Документация Tengri

Version stable, 2025-12-17

Оглавление

Архитектура . . . 4
- Идеология использования . . . 4
- Разделение Compute & Storage . . . 5
- Хранение . . . 6
- Вычисление . . . 6
- Диалект и протоколы PostgreSQL . . . 9
- Развитая система разграничения прав . . . 10

Результаты тестов . . . 12
- **TPC-H**, сравнение с Greenplum . . . 12
- **TPC-DS** . . . 15

Руководство пользователя . . . 20
- Вычислительные пулы . . . 22
- Управление пользователями и правами . . . 27
  - Операции с пользователями . . . 28
  - Операции с ролями . . . 31
  - Операции с привилегиями . . . 33
- Описание данных (DDL) . . . 36
  - Операции с таблицами . . . 37
  - Операции со схемами . . . 41
  - Операции с представлениями . . . 42
- Синтаксис запросов к данным . . . 43
  - `SELECT` . . . 45
  - `WITH` . . . 51
  - `FROM` . . . 53
  - `JOIN` . . . 56
  - `UNION` . . . 63
  - `WHERE` . . . 65
  - `CASE` . . . 66
  - `GROUP BY` . . . 69
  - `HAVING` . . . 71
  - `QUALIFY` . . . 71
  - `ORDER BY` . . . 74
  - `LIMIT` . . . 78
  - `LIKE` . . . 80
  - `SIMILAR TO` . . . 82
  - `AS` . . . 83
  - `DELETE` . . . 84
  - `UPDATE` . . . 87
  - Функции . . . 90
    - Агрегатные функции . . . 95

Числовые функции . . . 102
Текстовые функции . . . 121
Функции для регулярных выражений . . . 130
Функции для даты и времени . . . 137
Функции для JSON . . . 144
Функции для двоичных данных . . . 157
Функции для геоданных . . . 160
Оконные функции . . . 162
Утилиты . . . 173
Функции модуля Python `tngri` . . . 195
Типы данных . . . 202
Числовые типы . . . 203
Текстовый тип . . . 205
Типы для даты и времени . . . 206
Тип JSON . . . 210
Типы для массивов . . . 212
Двоичный тип . . . 214
Логический тип . . . 214
Тип для геоданных . . . 215
Руководство администратора . . . 217
Сценарии . . . 224
Начальная настройка системы . . . 224
Загрузка данных . . . 225
Автоматизация загрузки данных из множественных файлов . . . 229
Обращение к хранилищу Iceberg напрямую с помощью Python . . . 235
Аналитические сценарии . . . 238
Анализ данных из постов Instagram* . . . 238
Анализ геоданных спортивных трекеров . . . 252
Анализ текста на примере дата-сета FineWeb . . . 309
Алфавитный указатель . . . 346

Tengri **Data** — корпоративная аналитическая платформа.

Проблемы

Если для операционной деятельности крупных компаний существуют такие проверенные временем инструменты, как PostgreSQL, то выбор оптимального решения для аналитики данных по-прежнему представляет серьезный вызов.

Традиционные СУБД отлично справляются с объемом до **1 ТБ** анализируемых данных и поддерживают работу 10—20 аналитиков. Однако дальнейший рост потребностей аналитики, внедрение AI и ML, а также увеличение числа пользователей и доменов данных могут сильно увеличивать сроки и снижать надежность результатов.

▼ *Подробнее о **проблемах больших данных 3V** — Volume, Velocity, Variety*

Традиционно при работе с большими данным выделяют список главных проблем, состоящий из **3 V**:

1. Volume (Объем)

Данных для анализа становится все больше, речь идет о десятках и сотнях ТБ.

2. Velocity (Скорость)

Возрастают требования по скорости реагирования, растет число аналитиков и моделей, анализирующих данные. Увеличивается и скорость прироста данных.

3. Variety (Многообразие)

Увеличивается количество и разнообразие форматов источников данных.

Позже к этому списку базовых проблем добавлялись и другие **V**, среди которых:

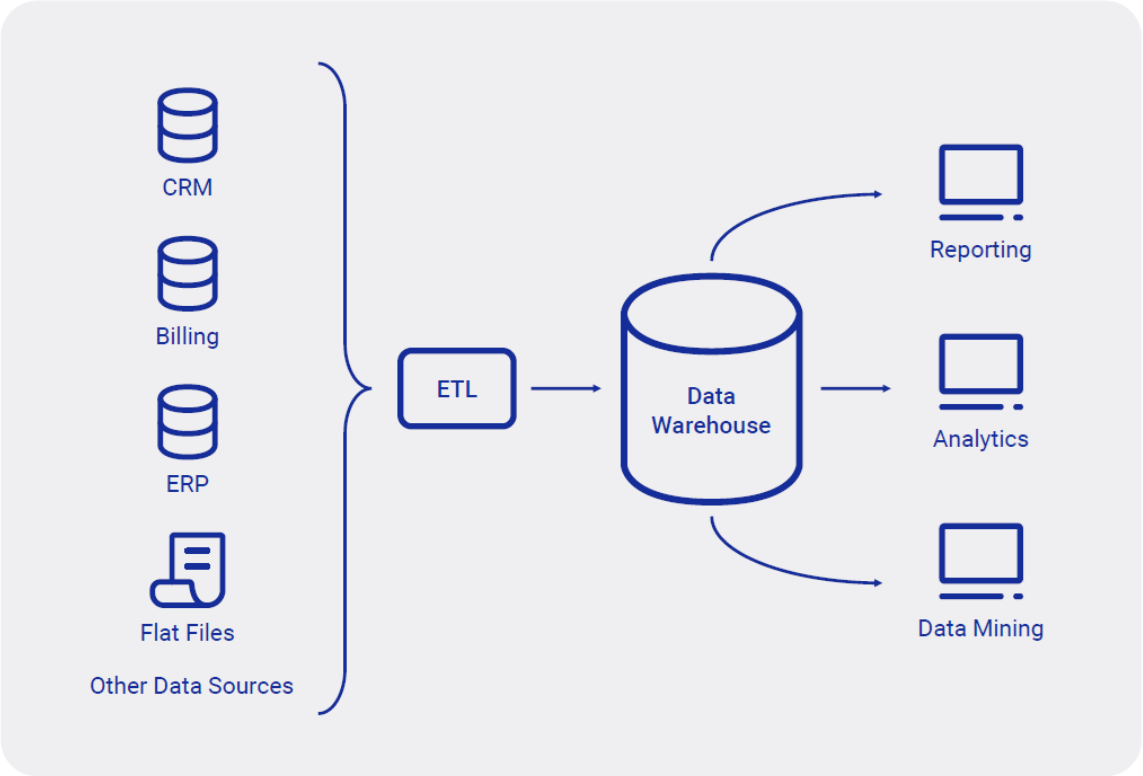
4. **Veracity** — достоверность данных
5. **Variability** — изменчивость данных
6. **Visualization** — визуализация данных
7. **Value** — ценность данных
8. **Venue** — место хранения и использования данных
9. **Vocabulary** — потребность в создании общего глоссария для работы с данными
10. **Vagueness** — неопределенность некоторых типов данных

Все эти ограничения и вызовы указывают на необходимость принципиально нового подхода при построении корпоративных систем анализа больших данных.

Что предлагает Tengri

Разделение

Принцип разделения **Compute & Storage** позволяет гибко и независимо масштабировать хранение данных и вычислительные ресурсы без кратного увеличения расходов на инфраструктуру и адаптивно встраиваться в современные гибридные инсталляции предприятий.



> Каждому запросу доступен свой ресурс: аналитикам, отчетной системе или сервису машинного обучения не приходится конкурировать
>
> — Николай Голов, директор по продукту Tengri Data

Удобство

Используя знакомый язык SQL и стандартные подключения ODBC/JDBC, аналитики могут получать ответы за считанные секунды — независимо от того, занимают ли данные **1 ГБ** или **1 ПБ**, и есть ли конкурирующие запросы или нет. При этом открытые форматы данных обеспечивают максимальную гибкость и масштабируемость.

> Аналитики могут сосредоточиться на решении бизнес-задач, не отвлекаясь на технические нюансы платформы.
>
> — Михаил Тюрин, директор департамента Tengri Data

Результат

Успех этого подхода на мировом рынке подтверждают коммерческие достижения **Snowflake** и **Databricks**, решения которых недоступны компаниям в РФ. Tengri предоставляет аналогичные, а

порой и превосходящие возможности, используя еще более современный технологический стек.

> Мы создаем платформу, открывающую путь к действительно неограниченной аналитике, снимая барьеры на пути к полноценному раскрытию потенциала данных.
>
> — Николай Голов, директор по продукту Tengri Data

Архитектура

В этом разделе описаны основные компоненты архитектуры Tengri.

- **Идеология использования**

  Об идеологии использования Tengri

- **Разделение Compute & Storage**

  О принципе разделения систем вычисления и хранения в Tengri

- **Хранение**

  О системе хранения данных в Tengri

- **Вычисление**

  О том, как устроено вычисление в Tengri

- **Диалект и протоколы PostgreSQL**

  О поддержке диалекта SQL PostgreSQL в Tengri

- **Развитая система разграничения прав**

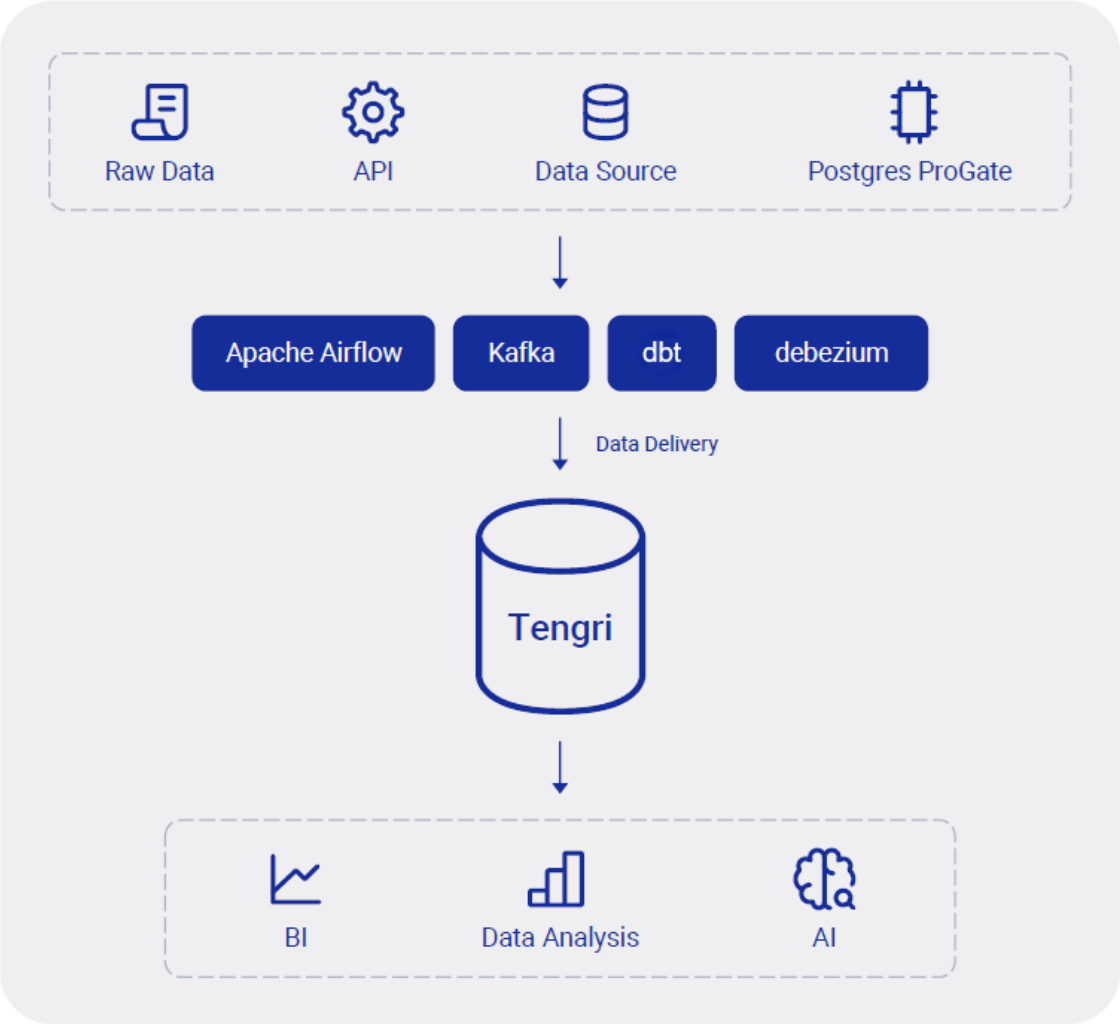
  О правах доступа и ролях пользователей в Tengri

Идеология использования

Общая схема использования Tengri

В Tengri данные:

1. поступают из множества разнообразных источников (API, CRM, ERP, JSON, биллинги, другие источники);
2. регулярно подгружаются в единую платформу;
3. анализируются и агрегируются;
4. порождают отчеты и графики;
5. служат для обучения моделей ML/AI.



В Tengri реализованы

- ☑ Поддержка OLAP SQL-запросов
- ☑ Простая интеграция с ETL/BI/ML-системами
- ☑ Неограниченное масштабирование
- ☑ Гарантии целостности и сохранности данных
- ☑ Система разграничения прав доступа

Разделение Compute & Storage

Tengri — аналитическая платформа нового поколения, основанная на парадигме разделения систем хранения и вычисления (**Compute & Storage**).

- Данные хранятся в универсальном масштабируемом кластере S3.
- Вычислительные мощности подбираются динамически при помощи изолированных вычислителей, запускаемых на платформе контейнеризации.

Такой подход позволяет:

- Гибко и независимо масштабировать хранение и вычислительные ресурсы без кратного увеличения расходов на инфраструктуру;
- Адаптивно встраиваться в современные гибридные инсталляции предприятий.

Описание систем вычисления и хранения представлено в разделах Вычисление и Хранение соответственно.

Хранение

Apache Iceberg в S3

Данные таблиц в Tengri хранятся в кластере **S3** в открытом формате **Apache Iceberg**.

- **Iceberg** поддерживает операции `INSERT`, `UPDATE` и `DELETE`, а также изменение схемы таблиц.
- **Iceberg** поддерживает ACID — набор требований к транзакционной системе, обеспечивающий наиболее надёжную и предсказуемую её работу.
- **Iceberg**-структура формируется на основе формата **Parquet** с поколоночной компрессией данных.
- **Iceberg** — это тысячи успешных внедрений крупнейшими мировыми компаниями для работы с сотнями **ТБ** и десятками **ПБ** данных.

Колоночный формат хранения

Благодаря использованию формата **Parquet** в Tengri поддерживаются:

Колоночный формат хранения данных

OLAP-запросы работают быстрее за счет чтения только нужных колонок таблиц.

Поколоночная компрессия данных

OLAP-запросы работают быстрее в 5 и более раз благодаря сжатию данных таблиц.

Вычисление

В Tengri вычисление производится в виртуальных вычислительных пулах.

Вычислители

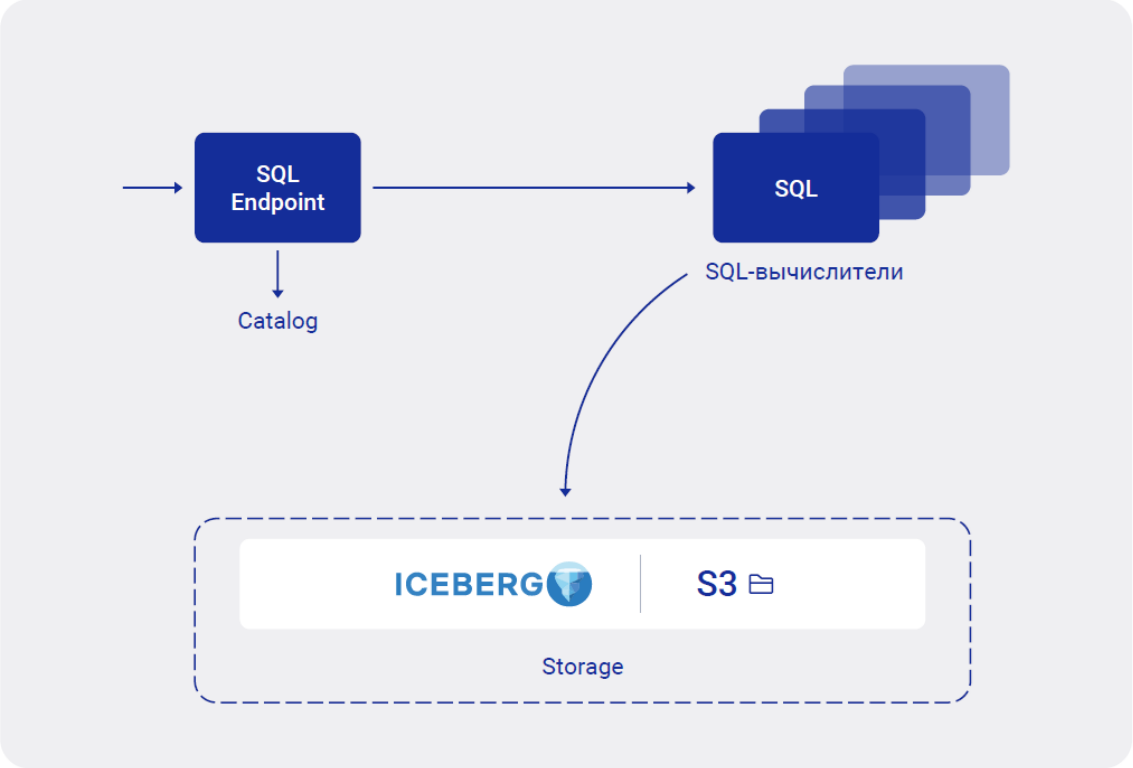
SQL-вычислитель — это легковесный контейнер, запускаемый на аппаратном кластере за ~100 миллисекунд.

SQL-вычислитель:

- получает эксклюзивные ядра (vCPU) и RAM-бюджет
- подгружает данные и выполняет SQL-запросы
- отключается системой после некоторого периода ожидания (если простаивает без нагрузки)

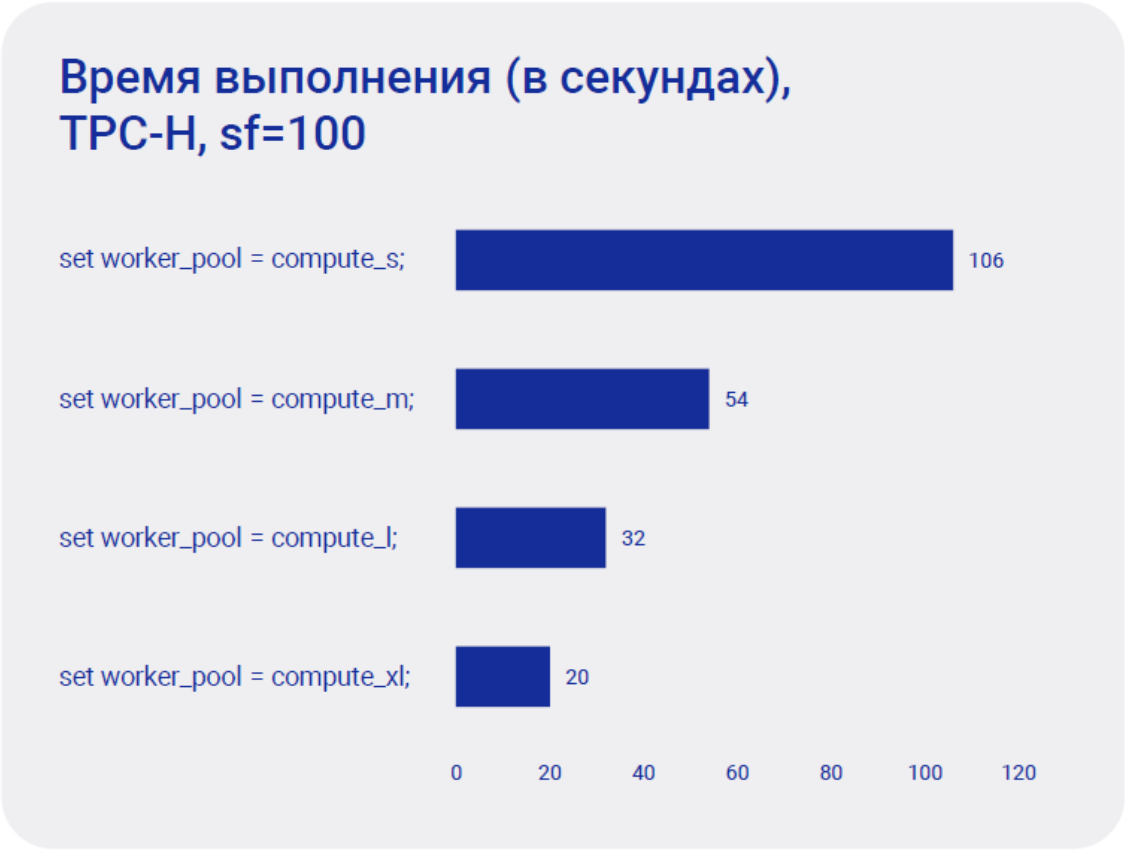
За счет такой архитектуры вычислителей:

- Каждый аналитик работает со своим SQL-вычислителем.
- Аналитики не мешают друг другу.



Каждый аналитик в каждой сессии может выбрать размер вычислителя для SQL-запроса: `XS` / `S` / `M` / `L` / `XL`. Каждый следующий размер вычислителя в два раза больше предыдущего.

Время выполнения теста **TPC-H** на вычислителях разного размера:



▼ *Код, использованный для выполнения этого теста*

```
SELECT profit.nation AS nation,
profit.o_year AS o_year,
SUM(profit.amount) AS sum_profit

FROM (SELECT s100_nation.n_name AS nation,

EXTRACT(YEAR FROM s100_orders.o_orderdate) AS o_year,

s100_lineitem.l_extendedprice * (1 - s100_lineitem.l_discount) -
s100_partsupp.ps_supplycost * s100_lineitem.l_quantity AS amount

FROM s100_part, s100_supplier, s100_lineitem,
s100_partsupp, s100_orders, s100_nation

WHERE s100_supplier.s_suppkey = s100_lineitem.l_suppkey
AND s100_partsupp.ps_suppkey = s100_lineitem.l_suppkey
AND s100_partsupp.ps_partkey = s100_lineitem.l_partkey
AND s100_part.p_partkey = s100_lineitem.l_partkey
AND s100_orders.o_orderkey = s100_lineitem.l_orderkey
AND s100_supplier.s_nationkey = s100_nation.n_nationkey
AND s100_part.p_name LIKE '%' || 'black' || '%')
AS profit

GROUP BY profit.nation, profit.o_year ORDER BY nation, o_year
DESC NULLS FIRST LIMIT 1000;
```

В Tengri также поддерживаются вычислители Python.

Ускорение вычислений

Для ускорения вычислений в Tengri поддержаны:

Morsel-driven parallelism

OLAP SQL-запросы работают быстрее за счет использования **Morsel**-алгоритмов, позволяющих загружать все vCPU, выделенные системой.

SIMD-инструкции

OLAP SQL-запросы работают быстрее за счет использования SIMD-инструкций современных CPU.

- Тесты производительности Tengri приведены в разделе Результаты тестов.
- Подробности о работе с вычислителями можно узнать в разделе Вычислительные пулы.

Диалект и протоколы PostgreSQL

PostgreSQL

В Tengri реализована поддержка SQL (диалект PostgreSQL).

В частности поддержаны:

- Основные элементы синтаксиса SQL диалекта PostgreSQL:
  `SELECT`, `INSERT`, `UPDATE`, `DELETE`, `JOIN`, `WITH`, `UNION`
- Оконные функции
- Работа с JSON посредством SQL
- Принципы ACID
- Транзакции
- Транзакционная целостность данных
- Изоляция данных на уровне **Repeatable read** (*Повторяющееся чтение*)
- JDBC-драйвер PostgreSQL

Загрузка данных

Гарантируется работоспособность инструментов загрузки данных (**Airflow**, **dbt**) и инструментов аналитики (BI, ML) за счет возможности подключаться через JDBC-драйвер PostgreSQL.

Все BI- и ETL-инструменты работают так, как если бы это был PostgreSQL.

> Аналитикам не нужно переучиваться на новые инструменты.

Развитая система разграничения прав

В Tengri реализована развитая система разграничения прав доступа.

Поддерживаемые принципы

В Tengri поддержаны принципы DAC и RBAC.

DAC

> **Избирательное управление доступом**: у каждого объекта есть владелец, который может предоставить права доступа (привилегии) на данный объект другим пользователям.

RBAC

> **Управление доступом на основе ролей**: права доступа (привилегии) приписаны ролям, которые, в свою очередь, приписаны пользователям.

Ключевые концепты системы разграничения прав

Защищаемый объект

Сущность, к которой может быть предоставлен доступ (привилегии). Если доступ к данному защищаемому объекту не разрешен, он будет запрещен.

Роль

Сущность, которой могут быть предоставлены права доступа (привилегии). Роли могут назначаться пользователям или другим ролям. Назначение роли другой роли создает **иерархию ролей**.

Привилегия

Определенный уровень доступа к объекту. Назначаются пользователям или ролям. Привилегии, назначенные ролям или пользователям, позволяют получить доступ к объектам, подлежащим защите. Могут быть отозваны у ролей или пользователей.

Для управления детализацией предоставляемого доступа может одновременно использоваться несколько различных привилегий.

Пользователь

Идентификатор, связанный с человеком или службой. Пользователь является объектом, которому могут быть предоставлены привилегии.

Подробности о работе с системой разграничения прав можно узнать в разделах:

- Операции с пользователями
- Операции с ролями
- Операции с привилегиями

Результаты тестов

Для оценки производительности использованы следующие тесты:

- TPC-H (*Transaction Processing Performance Council — High Performance*) — это бенчмарк для оценки производительности систем поддержки принятия решений и хранилищ данных. Он включает в себя набор бизнес-ориентированных аналитических запросов и параллельные модификации данных, моделирующие реальные сценарии использования таких систем.
- TPC-DS (*Transaction Processing Performance Council — Decision Support*) — это бенчмарк для оценки производительности систем поддержки принятия решений и аналитических баз данных. Он моделирует реальные сценарии работы с большими объемами данных, типичные для дата-центров розничных организаций и других крупных компаний.

Результаты:

- **TPC-H**, сравнение с Greenplum
- **TPC-DS**

TPC-H, сравнение с Greenplum

TPC-H (*Transaction Processing Performance Council — High Performance*) — это бенчмарк для оценки производительности систем поддержки принятия решений и хранилищ данных. Он включает в себя набор бизнес-ориентированных аналитических запросов и параллельные модификации данных, моделирующие реальные сценарии использования таких систем.

Параметры теста

Бенчмарк	TPC-H
Scaling factor	`100`
Запрос	случайный `Q1`—`Q10`
Параметры	случайные
Количество потоков	`1, 5, 10, 20`
Время	1 час

Примеры тестовых запросов

▼ *Посмотреть примеры тестовых SQL-запросов*

```
--Q1. Pricing Summary Report Query
--inputs = [{"depth": random.choice(range(90))}]
select
      l_returnflag,
```

```
      l_linestatus,
      sum(l_quantity) as sum_qty,
      sum(l_extendedprice) as sum_base_price,
      sum(l_extendedprice * (1-l_discount)) as sum_disc_price,
      sum(l_extendedprice * (1-l_discount) * (1+l_tax)) as sum_charge,
      avg(l_quantity) as avg_qty,
      avg(l_extendedprice) as avg_price,
      avg(l_discount) as avg_disc,
      count(*) as count_order
from
      s1_lineitem
where
      l_shipdate <= (DATE '1998-12-01') - :depth * interval '1' day
group by
      l_returnflag,
      l_linestatus
order by
      l_returnflag,
      l_linestatus;


--Q2. Minimum Cost Supplier Query
--inputs.size = random.choice(range(50))
--inputs.type = random.choice(['TIN','NICKEL', 'BRASS', 'STEEL', 'COPPER'])
--inputs.region = random.choice(['AFRICA','AMERICA','ASIA','EUROPE','MIDDLE EAST'])
select s_acctbal,s_name,n_name,p_partkey,p_mfgr,s_address,s_phone,s_comment
from s1_part, s1_supplier,s1_partsupp,s1_nation,s1_region
where p_partkey = ps_partkey
and s_suppkey = ps_suppkey
and p_size = :size
and p_type like '%' || :type
and s_nationkey = n_nationkey
and n_regionkey = r_regionkey
and r_name = :region
and ps_supplycost = (
    Select min(ps_supplycost)
    from s1_partsupp, s1_supplier, s1_nation, s1_region
    Where p_partkey = ps_partkey
    and s_suppkey = ps_suppkey
    and s_nationkey = n_nationkey
    and n_regionkey = r_regionkey
    and r_name = :region
)
order by s_acctbal desc,n_name,s_name,p_partkey;
```

Тестируемые системы

- Tengri
  - 100 vCPU
  - 200GB RAM

- **Greenplum** [1]
  - 1 координатор + 3 ноды по 30 vCPU, 64GB RAM

Результаты

Количество потоков	Tengri, запросов в час	Greenplum, запросов в час
1	302	304
5	1466	1082
10	2890	1444
20	4111	1800

При использовании системы одним аналитиком производительность Tengri и **Greenplum** сравнимая:

При использовании системы 10 аналитиками Tengri опережает **Greenplum** за счет эффективной утилизации ресурсов без конкуренции:

Tengri линейно масштабируется при увеличении количества аналитиков, эффективно утилизируя вычислительные ресурсы:

TPC-DS

TPC-DS (*Transaction Processing Performance Council — Decision Support*) — это бенчмарк для оценки производительности систем поддержки принятия решений и аналитических баз данных. Он моделирует реальные сценарии работы с большими объемами данных, типичные для дата-центров розничных организаций и других крупных компаний.

Параметры теста

Бенчмарк	TPC-DS
Тип теста	статический тест (без **ETL**)
Scaling factor	`100, 1000`
Запрос	случайный `Q1`—`Q100`
Параметры	случайные
Количество потоков	`1, 10, 20`
Время	1 час

Примеры тестовых запросов

▼ *Посмотреть примеры тестовых SQL-запросов*

```
-- start Q1
WITH customer_total_return
     AS (SELECT sr_customer_sk     AS ctr_customer_sk,
                sr_store_sk        AS ctr_store_sk,
                Sum(sr_return_amt) AS ctr_total_return
         FROM   s1_store_returns,
                s1_date_dim
         WHERE  sr_returned_date_sk = d_date_sk
                AND d_year = 2001
         GROUP  BY sr_customer_sk,
                   sr_store_sk)
SELECT c_customer_id
FROM   customer_total_return ctr1,
       s1_store,
       s1_customer
WHERE  ctr1.ctr_total_return > (SELECT Avg(ctr_total_return) * 1.2
                                FROM   customer_total_return ctr2
                                WHERE  ctr1.ctr_store_sk = ctr2.ctr_store_sk)
       AND s_store_sk = ctr1.ctr_store_sk
       AND s_state = 'TN'
       AND ctr1.ctr_customer_sk = c_customer_sk
ORDER  BY c_customer_id
LIMIT 100;

-- start Q2
WITH wscs
     AS (SELECT sold_date_sk,
                sales_price
         FROM   (SELECT ws_sold_date_sk    sold_date_sk,
                        ws_ext_sales_price sales_price
                 FROM   s1_web_sales)
         UNION ALL
```

```
         (SELECT cs_sold_date_sk    sold_date_sk,
                 cs_ext_sales_price sales_price
          FROM   s1_catalog_sales)),
     wswscs
     AS (SELECT d_week_seq,
                Sum(CASE
                      WHEN ( d_day_name = 'Sunday' ) THEN sales_price
                      ELSE NULL
                    END) sun_sales,
                Sum(CASE
                      WHEN ( d_day_name = 'Monday' ) THEN sales_price
                      ELSE NULL
                    END) mon_sales,
                Sum(CASE
                      WHEN ( d_day_name = 'Tuesday' ) THEN sales_price
                      ELSE NULL
                    END) tue_sales,
                Sum(CASE
                      WHEN ( d_day_name = 'Wednesday' ) THEN sales_price
                      ELSE NULL
                    END) wed_sales,
                Sum(CASE
                      WHEN ( d_day_name = 'Thursday' ) THEN sales_price
                      ELSE NULL
                    END) thu_sales,
                Sum(CASE
                      WHEN ( d_day_name = 'Friday' ) THEN sales_price
                      ELSE NULL
                    END) fri_sales,
                Sum(CASE
                      WHEN ( d_day_name = 'Saturday' ) THEN sales_price
                      ELSE NULL
                    END) sat_sales
         FROM   wscs,
                s1_date_dim
         WHERE  d_date_sk = sold_date_sk
         GROUP  BY d_week_seq)
SELECT d_week_seq1,
       Round(sun_sales1 / sun_sales2, 2),
       Round(mon_sales1 / mon_sales2, 2),
       Round(tue_sales1 / tue_sales2, 2),
       Round(wed_sales1 / wed_sales2, 2),
       Round(thu_sales1 / thu_sales2, 2),
       Round(fri_sales1 / fri_sales2, 2),
       Round(sat_sales1 / sat_sales2, 2)
FROM   (SELECT wswscs.d_week_seq d_week_seq1,
               sun_sales         sun_sales1,
               mon_sales         mon_sales1,
               tue_sales         tue_sales1,
               wed_sales         wed_sales1,
               thu_sales         thu_sales1,
               fri_sales         fri_sales1,
               sat_sales         sat_sales1
```

```
        FROM   wswscs,
               s1_date_dim
        WHERE  s1_date_dim.d_week_seq = wswscs.d_week_seq
               AND d_year = 1998) y,
       (SELECT wswscs.d_week_seq d_week_seq2,
               sun_sales         sun_sales2,
               mon_sales         mon_sales2,
               tue_sales         tue_sales2,
               wed_sales         wed_sales2,
               thu_sales         thu_sales2,
               fri_sales         fri_sales2,
               sat_sales         sat_sales2
        FROM   wswscs,
               s1_date_dim
        WHERE  s1_date_dim.d_week_seq = wswscs.d_week_seq
               AND d_year = 1998 + 1) z
WHERE  d_week_seq1 = d_week_seq2 - 53
ORDER  BY d_week_seq1;
```

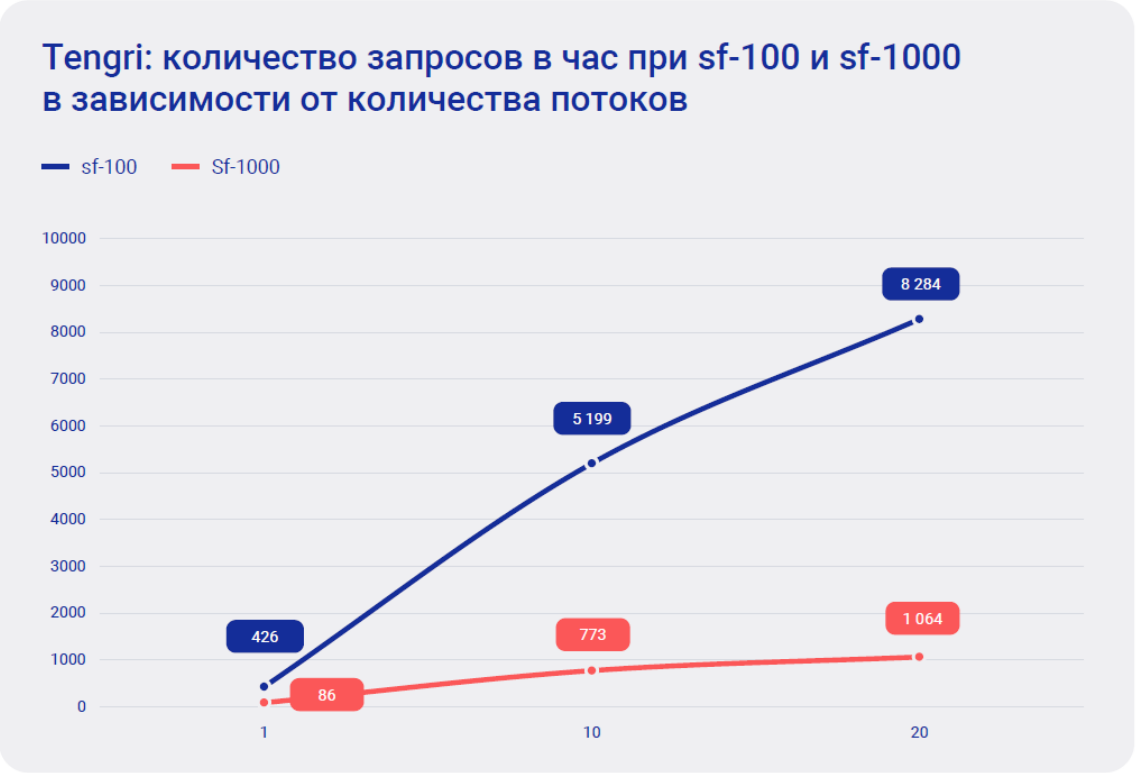
Тестируемая система

- Tengri
  - 200 vCPU
  - 400GB RAM

Результаты

Количество потоков	`sf=100`, запросов в час	`sf=1000`, запросов в час
`1`	426	86
`10`	5199	773
`20`	8284	1064

При переходе от значения **Scaling factor** в `100` к значению в `1000` работоспособность Tengri сохраняется, производительность снижается всего в 5-6 раз, хотя объем данных растет в 10 раз.



[1] Тест проведен на Cloudberry 1.6.0 (based on postgres 14), условно аналогичной Greenplum 7. Данные о производительности, содержащиеся на этом сайте, были получены в строго контролируемой среде Postgres Professional при моделировании предполагаемых аналитических нагрузок и носят справочный характер. Результаты, полученные в других условиях, могут отличаться.

Руководство пользователя

▼ *Вычислительные пулы*

- Создание вычислительного пула
- Сброс вычислительного пула
- Изменение атрибутов вычислительного пула
- Установка атрибутов вычислительного пула по умолчанию
- Вывод списка всех вычислительных пулов
- Вывод текущего вычислительного пула
- Установка текущего вычислительного пула

▼ *Управление пользователями и правами*

- Операции с пользователями
  - Создание нового пользователя
  - Изменение атрибутов пользователя
  - Сброс пользователя
  - Вывод информации о пользователе
  - Вывод списка всех пользователей
- Операции с ролями
  - Создание новой роли
  - Изменение атрибутов роли
  - Сброс роли
  - Вывод списка всех ролей
  - Установка активной роли
  - Предоставление ролям атрибутов
- Операции с привилегиями
  - Предоставление привилегий
  - Отзыв привилегий

▼ *Описание данных (DDL)*

- Операции с таблицами
  - Создание новой таблицы
  - Удаление таблицы
  - Вывод списка всех таблиц
- Операции со схемами
  - Создание новой схемы
  - Изменение атрибутов схемы
  - Удаление схемы

- Операции с представлениями
  - Создание нового представления
  - Удаление представления

▼ *Синтаксис запросов к данным*

- `SELECT`
- `WITH`
- `FROM`
- `JOIN`
- `UNION`
- `WHERE`
- `CASE`
- `GROUP BY`
- `HAVING`
- `QUALIFY`
- `ORDER BY`
- `LIMIT`
- `OFFSET`
- `LIKE`
- `ILIKE`
- `SIMILAR TO`
- `AS`
- `DELETE`
- `UPDATE`
- Функции
  - Агрегатные функции
  - Числовые функции
  - Текстовые функции
  - Функции для регулярных выражений
  - Функции для даты и времени
  - Функции для JSON
  - Функции для двоичных данных
  - Функции для геоданных
  - Оконные функции
  - Утилиты
  - Функции модуля Python `tngri`

▼ *Типы данных*

- Числовые типы
- Текстовый тип
- Типы для даты и времени
- Тип JSON
- Типы для массивов
- Двоичный тип
- Логический тип
- Тип для геоданных

Вычислительные пулы

В Tengri вычисление производится в виртуальных вычислительных пулах.

Вычислитель — это легковесный контейнер, входящий в вычислительный пул, содержащий программный код и обладающий выделенными вычислительными ресурсами для вычислительных задач:

- Выполнение SQL-запросов
- Выполнение кода на Python
- Запуск моделей AI

Операции с вычислительными пулами:

- `CREATE WORKER POOL` — Создание вычислительного пула
- `ALTER WORKER POOL` — Изменение атрибутов вычислительного пула
- `DROP WORKER POOL` — Сброс вычислительного пула
- `ALTER WORKER POOL SET DEFAULT` — Установка атрибутов вычислительного пула по умолчанию
- `SHOW WORKER POOLS` — Вывод списка всех вычислительных пулов
- `SHOW WORKER POOL` — Вывод текущего вычислительного пула
- `USE WORKER POOL` — Установка текущего вычислительного пула
- Предоставление привилегий на вычислительные пулы

Создание вычислительного пула

```
CREATE [OR REPLACE] WORKER POOL [IF NOT EXISTS] <pool_name>
    [<worker_pool_attribute> [ ... ]];
```

Создает новый вычислительный пул с указанным именем и атрибутами.

Если указан модификатор `OR REPLACE`, то конечное действие эквивалентно удалению существующего вычислительного пула и созданию нового с тем же именем.

Опциональный модификатор `IF NOT EXISTS` ограничивает запрос только теми случаями, в которых указанный объект еще не существует.

Модификаторы являются взаимоисключающими. Если указать их оба, это приведет к ошибке.

Можно указать следующие атрибуты вычислительного пула:

- `WORKER SIZE <size>` — устанавливает размер вычислителя для данного вычислительного пула: `XS`, `S`, `M`, `L`, `XL`. Чем больше размер, тем быстрее будут проходить вычисления.
- `MAX WORKER COUNT <count>` — устанавливает максимальное количество вычислителей.
- `WORKER IDLE TTL <seconds> SECONDS` — устанавливает максимальное время простоя вычислителей.
- `SCALING STRATEGY <strategy>` — задает стратегию масштабирования:
  - `STD` — если нет свободных вычислителей, сразу сообщить об их отсутствии.
  - `ECO` — если нет свободных вычислителей, ждать 10 минут, пока они появятся.

▼ *Посмотреть примеры*

Создаем вычислительный пул с именем `my_worker_pool` и размером вычислителей `S`:

```
CREATE WORKER POOL my_worker_pool
    WORKER SIZE S;
```

Создаем вычислительный пул с именем `my_worker_pool`, размером вычислителей `L`, максимальным количеством вычислителей `40`, максимальным временем простоя вычислителей `300` секунд и со стратегией масштабирования `ECO`:

```
CREATE WORKER POOL my_worker_pool
    WORKER SIZE L
    MAX WORKER COUNT 40
    WORKER IDLE TTL 300 SECONDS
    SCALING STRATEGY ECO;
```

Изменение атрибутов вычислительного пула

```
ALTER WORKER POOL [IF EXISTS] <pool_name> RENAME TO <new_pool_name>;

ALTER WORKER POOL [IF EXISTS] <pool_name> SET <worker_pool_attribute>;
```

Изменяет вычислительный пул с указанным именем.

Не рекомендуется изменять атрибуты вычислительных пулов, запущенных внутри вашей рабочей группы, так как это может привести к неожиданному падению производительности вычислений в случаях, когда вычислительные пулы используются несколькими пользователями.

> Поэтому выражения `ALTER WORKER POOL` следует использовать только администраторам.
>
> Если вам как пользователю требуется изменить размер используемого вами вычислительного пула (например, увеличить его), то рекомендуется выбрать для работы другой вычислительный пул большего размера.

Поддерживаемые действия:

- `RENAME TO` — переименовывает вычислительный пул в указанное имя `<new_pool_name>`. Все атрибуты и конфигурации вычислительного пула сохраняются.
- `SET` — обновляет атрибут вычислительного пула. Можно изменить следующие атрибуты:
  - `WORKER SIZE <size>` — задает размер вычислителя (`XS`, `S`, `M`, `L`, `XL`) для данного вычислительного пула. Чем больше размер, тем быстрее будут проходить вычисления.
  - `MAX WORKER COUNT <count>` — устанавливает максимальное количество вычислителей для данного вычислительного пула.
  - `WORKER IDLE TTL <seconds> SECONDS` — устанавливает время простоя вычислителей для данного вычислительного пула.
  - `SCALING STRATEGY <strategy>` — задает стратегию масштабирования:
    - `STD` — если нет свободных вычислителей, сразу сообщить об их отсутствии.
    - `ECO` — если нет свободных вычислителей, ждать 10 минут, пока они появятся.

Опциональный модификатор `IF EXISTS` ограничивает запрос только теми случаями, в которых указанный объект существует.

Сброс вычислительного пула

```
DROP WORKER POOL [IF EXISTS] <pool_name>;
```

Удаляет вычислительный пул с указанным именем.

Опциональный модификатор `IF EXISTS` ограничивает запрос только теми случаями, в которых указанный объект существует.

Установка атрибутов вычислительного пула по умолчанию

```
ALTER WORKER POOL SET DEFAULT <worker_pool_attributes>;
```

Устанавливает атрибуты по умолчанию для всех новых вычислительных пулов в системе.

Этот оператор позволяет задать общесистемные атрибуты по умолчанию, которые будут применяться к вычислительным пулам при их создании без явного указания этих атрибутов.

Можно указать следующие атрибуты вычислительных пулов:

- `WORKER SIZE <size>` — задает размер вычислителя по умолчанию (`XS`, `S`, `M`, `L`, `XL`) для данного вычислительного пула. Чем больше размер, тем быстрее будут проходить вычисления.
- `MAX WORKER COUNT <count>` — устанавливает максимальное количество вычислителей по умолчанию для данного вычислительного пула.
- `WORKER IDLE TTL <seconds> SECONDS` — устанавливает время простоя вычислителей по умолчанию для данного вычислительного пула.
- `SCALING STRATEGY <strategy>` — устанавливает стратегию масштабирования по умолчанию:
  - `STD` — если нет свободных вычислителей, сразу сообщить об их отсутствии.
  - `ECO` — если нет свободных вычислителей, ждать 10 минут, пока они появятся.

Вывод списка всех вычислительных пулов

```
SHOW WORKER POOLS;
```

Выводит список всех вычислительных пулов в системе.

Это утверждение отображает информацию обо всех доступных вычислительных пулах, включая их имена и конфигурации.

Чтобы увидеть вычислительный пул в ответе, требуется привилегия `MONITOR`.

Вывод текущего вычислительного пула

```
SHOW WORKER POOL;
```

Выводит вычислительный пул, который в данный момент используется для выполнения запроса.

Это утверждение отображает информацию о текущем активном вычислительном пуле, включая его имя и конфигурацию.

Установка текущего вычислительного пула

```
USE WORKER POOL <pool_name>;
```

Устанавливает указанный вычислительный пул в качестве текущего пула для выполнения запросов.

Этот оператор изменяет вычислительный пул, который будет использоваться для выполнения последующих запросов. Указанный пул должен существовать, а пользователь должен иметь необходимые привилегии для его использования.

Для использования этого оператора необходима привилегия `USAGE`.

Увеличение размеров вычислительного пула

Если вам как пользователю требуется изменить размер используемого вами вычислительного пула (например, увеличить его), то рекомендуется выбрать для работы другой вычислительный пул из списка доступных, не изменяя атрибуты текущего вычислительного пула.

Для этого нужно сделать следующее.

Выводим размер используемого вычислительного пула:

```
SHOW WORKER POOL;
```

```
+----------------+------+
| worker_pool    | size |
+----------------+------+
| compute_s      | S    |
+----------------+------+
```

Выводим размеры доступных для использования вычислительных пулов:

```
SHOW WORKER POOLS;
```

```
+----------------+-------+
| name           | value |
+----------------+-------+
| client_compute | S     |
+----------------+-------+
| compute_m      | M     |
+----------------+-------+
| compute_s      | S     |
+----------------+-------+
| compute_l      | L     |
+----------------+-------+
```

Переключаемся на вычислительный пул большего размера:

```
USE WORKER POOL compute_m;
```

```
+--------+
| status |
+--------+
| USE    |
```

```
+--------+
```

Управление пользователями и правами

В Tengri реализована развитая система разграничения прав доступа.

Поддержаны принципы:

- DAC — *избирательное управление доступом*
- RBAC — *управление доступом на основе ролей*

Ключевые концепты системы разграничения прав

Защищаемый объект

Сущность, к которой может быть предоставлен доступ (привилегии).

Роль

Сущность, которой могут быть предоставлены права доступа (привилегии). Роли могут назначаться пользователям или другим ролям.

Привилегия

Определенный уровень доступа к объекту. Назначаются пользователям или ролям.

Пользователь

Идентификатор, связанный с человеком или службой. Пользователь является объектом, которому могут быть предоставлены привилегии и/или роли.

Операции с объектами системы разграничения прав

- Операции с пользователями
  - Создание нового пользователя
  - Изменение атрибутов пользователя
  - Сброс пользователя
  - Вывод информации о пользователе
  - Вывод списка всех пользователей
- Операции с ролями
  - Создание новой роли
  - Изменение атрибутов роли
  - Сброс роли

    - Вывод списка всех ролей
    - Установка активной роли
    - Предоставление ролям атрибутов
- Операции с привилегиями
    - Предоставление привилегий
    - Отзыв привилегий

Операции с пользователями

- `CREATE USER` — Создание нового пользователя
- `ALTER USER` — Изменение атрибутов пользователя
- `DROP USER` — Сброс пользователя
- `DESCRIBE USER` — Вывод информации о пользователе
- `SHOW USERS` — Вывод списка всех пользователей
- `GRANT ROLE` — Назначение пользователям ролей
- `REVOKE ROLE` — Отзыв ролей у пользователей

Создание нового пользователя

```
CREATE [OR REPLACE] USER [IF NOT EXISTS] <user_name>
    [SET DEFAULT WORKER POOL <pool_id>]
    [IDENTIFIED BY (TRUST <ip_address> | PASSWORD <user_password>)];
```

Создает нового пользователя с указанным именем.

Модификатор `OR REPLACE`

Если указан опциональный модификатор `OR REPLACE`, то это действие эквивалентно удалению текущего пользователя и созданию нового пользователя с тем же именем пользователя. Вновь созданный пользователь не наследует никаких ролей и атрибутов.

Модификатор `IF NOT EXISTS`

Если указан опциональный модификатор `IF NOT EXISTS`, то запрос будет выполнен только в том случае, если пользователь с именем `<user_name>` не существует, в противном случае ничего не произойдет.

Модификаторы `OR REPLACE` и `IF NOT EXISTS` являются взаимоисключающими. Если указать их оба, это приведет к ошибке.

Параметр `SET DEFAULT WORKER POOL`

Опциональный параметр `SET DEFAULT WORKER POOL <pool_id>` устанавливает вычислительный пул `<pool_id>` по умолчанию для указанного пользователя.

Параметр `IDENTIFIED BY TRUST`

Опциональный параметр `IDENTIFIED BY TRUST <ip_address>` устанавливает проверку подлинности указанного пользователя через IP адрес `<ip_address>`.

▼ *Посмотреть примеры*

- ```
  CREATE USER ivanov IDENTIFIED BY TRUST '127.0.0.1';
  ```
- ```
  ALTER USER ivanov IDENTIFIED BY TRUST '127.0.0.1';
  ```

Параметр `IDENTIFIED BY PASSWORD`

Опциональный параметр `IDENTIFIED BY PASSWORD <user_password>` устанавливает проверку подлинности указанного пользователя через пароль `<user_password>` с использованием алгоритма `SCRAM-SHA-256`.

▼ *Посмотреть примеры*

- ```
  CREATE USER ivanov IDENTIFIED BY PASSWORD 'QWERTY123456';
  ```
- ```
  ALTER USER ivanov IDENTIFIED BY PASSWORD 'QWERTY123456';
  ```

Изменение атрибутов пользователя

```
ALTER USER [IF EXISTS] <user_name>
    RENAME TO <new_name>;

ALTER USER [IF EXISTS] <user_name>
    SET DEFAULT WORKER POOL <pool_id>;

ALTER USER [IF EXISTS] <user_name>
    IDENTIFIED BY (TRUST <ip_address> | PASSWORD <user_password>);
```

Изменяет атрибуты пользователя.

- `RENAME TO <new_name>` — атомарная операция, которая переименовывает пользователя на новое имя. Никакие другие свойства данного пользователя (предоставленные роли, привилегии и т.д.) при этом не изменяются.

- `SET DEFAULT WORKER POOL <pool_id>` — см. Параметр `SET DEFAULT WORKER POOL`.
- `IDENTIFIED BY TRUST <ip_address>` — см. Параметр `IDENTIFIED BY TRUST`.
- `IDENTIFIED BY PASSWORD <user_password>` — см. Параметр `IDENTIFIED BY PASSWORD`.

Опциональный модификатор `IF EXISTS` ограничивает запрос только теми случаями, в которых указанный объект существует.

Сброс пользователя

```
DROP USER [IF EXISTS] <user_name>;
```

Полностью удаляет пользователя из системы.

Не сохраняется ничего, включая предоставленные роли и права доступа.

Возможно, что некоторые объекты потеряют права доступа.

Опциональный модификатор `IF EXISTS` ограничивает запрос только теми случаями, в которых указанный объект существует.

Вывод информации о пользователе

```
DESC[RIBE] USER <user_name>;
```

Выводит всю информацию о пользователе, включая его атрибуты и значения по умолчанию.

Вывод списка всех пользователей

```
SHOW USERS;
```

Вывод списка всех зарегистрированных пользователей.

Назначение пользователям ролей

```
GRANT ROLE <role_name> TO <user_name>;
```

Назначает указанному пользователю указанную роль.

Отзыв ролей у пользователей

```
REVOKE ROLE <role_name> FROM <user_name>;
```

Отзывает указанную роль у указанного пользователя.

Операции с ролями

- `CREATE ROLE` — Создание новой роли
- `ALTER ROLE` — Изменение атрибутов роли
- `DROP ROLE` — Сброс роли
- `SHOW ROLES` — Вывод списка всех ролей
- `USE ROLE` — Установка активной роли
- `GRANT` — Предоставление ролям атрибутов

Создание новой роли

```
CREATE [OR REPLACE] ROLE [IF NOT EXISTS] <role_name>;
```

Создает новую роль с указанным именем.

Если указан модификатор `OR REPLACE`, то конечное действие эквивалентно удалению существующей роли и созданию новой роли с тем же именем. Вновь созданная роль не наследует никаких атрибутов или привилегий от предыдущей роли.

Опциональный модификатор `IF NOT EXISTS` ограничивает запрос только теми случаями, в которых указанный объект еще не существует.

Модификаторы являются взаимоисключающими. Если указать их оба, это приведет к ошибке.

Изменение атрибутов роли

```
ALTER ROLE [IF EXISTS] <role_name> RENAME TO <new_role_name>;

ALTER ROLE [IF EXISTS] <role_name> WITH <attribute>;
```

Изменяет существующую роль.

Действие может заключаться либо в переименовании роли, либо в изменении ее атрибутов.

Доступные атрибуты:

- `SUPERUSER` — предоставляет роли привилегии суперпользователя.
- `CREATEROLE` — позволяет роли создавать новые роли.

Опциональный модификатор `IF EXISTS` ограничивает запрос только теми случаями, в которых указанный объект существует.

Сброс роли

```
DROP ROLE [IF EXISTS] <role_name>;
```

Удаляет роль из системы.

Команда завершится неудачей, если на роль все еще ссылаются какие-либо объекты (например, она предоставлена пользователям или имеет право доступа для каких-то объектов).

Опциональный модификатор `IF EXISTS` ограничивает запрос только теми случаями, в которых указанный объект существует.

Вывод списка всех ролей

```
SHOW [TERSE] ROLES
    [LIKE '<pattern>']
    [STARTS WITH '<pattern>']
    [LIMIT <number> [FROM '<pattern>']];
```

Выводит список всех ролей в системе. Вывод можно отфильтровать и ограничить с помощью различных опций:

- `TERSE` — отображает упрощенный формат вывода.
- `LIKE '<pattern>'` — фильтрует роли, имена которых соответствуют указанному шаблону.
- `STARTS WITH '<pattern>'` — фильтрует роли, имена которых начинаются с указанного шаблона.
- `LIMIT <number> [FROM '<pattern>']` — ограничивает количество возвращаемых ролей, опционально начиная с определенного имени роли.

Шаблоны `<pattern>` представляют собой чувствительные к регистру строковые литералы, которые могут содержать символы подстановки.

Установка активной роли

```
USE ROLE <role_name>;
```

Устанавливает указанную роль в качестве активной роли для текущего сеанса.

Если роль установлена как активная, пользователь получает все привилегии, предоставленные этой роли, на время сессии.

Чтобы использовать указанную роль, пользователю должны быть предоставлены соответствующие права.

Предоставление ролям атрибутов

```
GRANT <role_attributes> TO ROLE <role_name>;
```

Предоставляет указанной роли `<role_name>` атрибуты `<role_attributes>`.

Это утверждение позволяет роли иметь специальные атрибуты или выполнять операции над другими ролями в соответствии с предоставленными привилегиями.

Операции с привилегиями

- `GRANT` — Предоставление привилегий
- `REVOKE` — Отзыв привилегий

Предоставление привилегий

Привилегии на схемы

```
GRANT <schema_privileges> ON SCHEMA <schema_name> TO [ROLE] <role_name>;
```

Предоставляет роли указанные привилегии на указанную схему.

`<schema_privileges>` — это список привилегий, разделенный запятыми.

Доступные привилегии на схемы:

- `ADMIN` — позволяет удалять схему и раздавать права на схему.
- `MONITOR` — разрешает доступ к метаинформации о схеме.
- `USAGE` — разрешает доступ к объектам в схеме.
- `MODIFY` — позволяет изменять свойства схемы.
- `CREATE TABLE` — позволяет создавать таблицы в схеме.
- `CREATE VIEW` — позволяет создавать представления в схеме.

Ключевое слово `ROLE` перед именем целевой роли не является обязательным.

▼ *Посмотреть пример*

Предоставим роли `junior_admin` привилегии `MONITOR` и `MODIFY` на схему `main_schema`.

```
GRANT MONITOR, MODIFY
    ON SCHEMA main_schema
    TO ROLE junior_admin;
```

Привилегии на таблицы и представления

Прямой синтаксис:

```
GRANT <table_privileges>
    ON (TABLE | VIEW) <table_name>
    TO [ROLE] <role_name>;
```

Темпоральный синтаксис:

```
GRANT <table_privileges>
    ON ALL [EXISTING] [[AND] FUTURE] (TABLES | VIEWS | TABLES AND VIEWS)
    IN [SCHEMA]<schema_name>
    TO [ROLE] <role_name>;
```

Предоставляет роли указанные привилегии на таблицу или представление внутри указанной схемы.

`<table_privileges>` — это список привилегий, разделенный запятыми.

Доступные привилегии на таблицы и представления:

- `SELECT` — позволяет читать данные из таблицы.
- `INSERT` — позволяет добавлять новые строки в таблицу.
- `UPDATE` — позволяет изменять существующие строки в таблице.
- `DELETE` — позволяет удалять строки из таблицы.
- `MODIFY` — позволяет изменять структуру таблицы.
- `OWNERSHIP` — позволяет изменять владельца таблицы.

Чтобы выдать привилегию на все существующие (или те, которые будут созданы в будущем) объекты внутри схемы, используется темпоральный синтаксис.

Модификатор `EXISTING` ограничивает предоставление привилегий только существующими объектами.

Модификатор `FUTURE` ограничивает предоставление привилегий объектами, которые будут созданы в будущем.

Если ни один модификатор не указан, то действие распространяется на все объекты — и уже существующие, и те, которые будут созданы в будущем.

Ключевое слово `ROLE` перед именем целевой роли не является обязательным.

▼ *Посмотреть пример*

Предоставим роли `junior_admin` привилегии `SELECT` и `INSERT` на все существующие таблицы и представления внутри схемы `main_schema`.

```
GRANT SELECT, INSERT
    ON EXISTING TABLES AND VIEWS
    IN SCHEMA main_schema
    TO ROLE junior_admin;
```

Привилегии на каталоги

```
GRANT <catalog_privileges> ON CATALOG TO [ROLE] <role_name>;
```

Предоставляет роли указанные привилегии на каталог.

`<catalog_privileges>` — это список привилегий, разделенный запятыми.

Доступные привилегии на каталоги:

- `ADMIN` — любые действия с каталогом.
- `CREATE USER` — создание пользователей.
- `CREATE WORKER POOL` — создание вычислительных пулов.
- `CREATE SCHEMA` — создание схем.

Модификатор `EXISTING` ограничивает предоставление привилегий только существующими объектами.

Модификатор `FUTURE` ограничивает предоставление привилегий объектами, которые будут созданы в будущем.

Ключевое слово `ROLE` перед именем целевой роли не является обязательным.

Привилегии на вычислительные пулы

```
GRANT <worker_pool_privileges> ON WORKER POOL <worker_pool_name> TO [ROLE] <role_name>;
```

Предоставляет роли указанные привилегии на вычислительный пул.

`<worker_pool_privileges>` — это список привилегий, разделенный запятыми.

Доступные привилегии на вычислительные пулы:

- `ALL` — все возможные действия с вычислительным пулом.
- `ADMIN` — изменение вычислительного пула (права на выполнение команды `ALTER WORKER POOL`).
- `USAGE` — использование вычислительного пула (права на выполнение команды `USE WORKER POOL`).
- `MONITOR` — мониторинг вычислительного пула.

▼ *Посмотреть пример*

Предоставим роли `junior_analyst` привилегии `USAGE` и `MONITOR` на вычислительный пул `compute_xl`.

```
GRANT USAGE, MONITOR
    ON WORKER POOL compute_xl
    TO junior_analyst;
```

Отзыв привилегий

```
REVOKE <schema_privileges>
    ON SCHEMA <schema_name> FROM ROLE <role_name>;

REVOKE <table_privileges>
    ON (TABLE | VIEW) <table_name> [IN SCHEMA <schema_name>] FROM ROLE <role_name>;

REVOKE <catalog_privileges>
    ON CATALOG FROM ROLE <role_name>;

REVOKE <worker_pool_privileges>
    ON WORKER POOL <worker_pool_name> FROM ROLE <role_name>;
```

Отзывает указанные привилегии на указанный объект у роли.

`<schema_privileges>`, `<table_privileges>`, `<catalog_privileges>` и `<worker_pool_privileges>` — это списки привилегий, разделенные запятыми. Конкретные привилегии зависят от целевого объекта.

Модификатор `EXISTING` ограничивает отзыв привилегий только существующими объектами.

Модификатор `FUTURE` ограничивает отзыв привилегий объектами, которые будут созданы в будущем.

▼ *Посмотреть пример*

Отзовем у роли `junior_admin` привилегии `SELECT` и `INSERT` на все существующие таблицы и представления внутри схемы `main_schema`.

```
REVOKE SELECT, INSERT
    ON EXISTING TABLES AND VIEWS
    IN SCHEMA main_schema
    FROM ROLE junior_admin;
```

Описание данных (DDL)

Команды DDL используются для описания данных — для создания, управления и удаления таблиц, схем и представлений.

- Операции с таблицами
- Операции со схемами
- Операции с представлениями

Операции с таблицами

- `CREATE TABLE` — Создание новой таблицы
- `INSERT` — Добавление данных в таблицу
- `DROP TABLE` — Удаление таблицы
- `TRUNCATE TABLE` — Удаление всех строк из таблицы
- `SHOW TABLES` — Вывод списка всех таблиц
- `DESCRIBE TABLE` — Вывод информации о таблице

Создание новой таблицы

```
CREATE [OR REPLACE] TABLE [IF NOT EXISTS] [<table_schema>.]<table_name>
    (<column_name> <column_type>
        [NOT NULL] [DEFAULT <default_expr>]
        [GENERATED BY DEFAULT AS IDENTITY] [GENERATED ALWAYS AS IDENTITY],
    ...
    )
    [WITH (<table_param>, ... )]
```

Создает новую таблицу с указанным именем и указанными столбцами.

```
CREATE [OR REPLACE] TABLE [IF NOT EXISTS] [<table_schema>.]<table_name> AS
    <select_expr>
    [WITH (<table_param>, ... )]
```

Создает новую таблицу с указанным именем на основе результата запроса `SELECT`.

Параметры

- `<table_name>` — имя создаваемой таблицы

- `<table_schema>` — схема создаваемой таблицы

- `<column_name>` — имя столбца создаваемой таблицы

- `<column_type>` — тип данных столбца создаваемой таблицы (см. раздел Типы данных)

- `NOT NULL` — данный столбец не принимает значения `NULL`

- `DEFAULT <default_expr>` — константа или константное выражение по умолчанию

- `GENERATED BY DEFAULT AS IDENTITY` — данный столбец является столбцом идентификации, то есть значения для него по умолчанию автоматически генерируется из неявной последовательности. Если команда `INSERT` выполняется для таблицы со столбцом идентификации и значение для этого столбца не указано явно, то в этот столбец вставляется значение, сгенерированное неявной последовательностью.

  ▼ *Посмотреть пример*

  Создадим таблицу `my_table` с двумя столбцами. Столбец `id` зададим как столбец идентификации (автоинкремент). Вставим в другой столбец числа от 5 до 10 с помощью функции `generate_series`. В итоге в столбец идентификации автоматически вставятся значения.

```
CREATE TABLE my_table (
    id INT GENERATED BY DEFAULT AS IDENTITY,
    numbers INT,
    );

INSERT INTO my_table (numbers)
    SELECT unnest(generate_series(5,10));

SELECT * FROM my_table;
```

```
+----+---------+
| id | numbers |
+----+---------+
| 0  | 5       |
+----+---------+
| 1  | 6       |
+----+---------+
| 2  | 7       |
+----+---------+
| 3  | 8       |
+----+---------+
| 4  | 9       |
+----+---------+
| 5  | 10      |
+----+---------+
```

- `<select_expr>` — выражение `SELECT`, результат выполнения которого будет записан в создаваемую таблицу

- `<table_param>` — параметры создаваемой таблицы

```
table_param ::= [<name> = <value>]
```

Возможные значения:

- `snapshot_ttl = <duration>` — глубина хранения снапшотов (версий таблицы).
  По умолчанию: 7 дней, но не более 1000 снапшотов.
  Например: `'1 week'`, `'2 days'`, `'4 days 3 hours 5 minutes 30 seconds'`
- `order_by = <column_name>` — столбец для сортировки, используемый для уплотнения данных таблицы. Данные в таблице хранятся с произвольной сортировкой, упорядоченность не гарантируется.

Если указан модификатор `OR REPLACE`, то конечное действие эквивалентно удалению существующей таблицы и созданию новой с тем же именем.

Опциональный модификатор `IF NOT EXISTS` ограничивает запрос только теми случаями, в которых указанный объект еще не существует.

Модификаторы являются взаимоисключающими. Если указать их оба, это приведет к ошибке.

Добавление данных в таблицу

```
INSERT INTO <target_table> [ ( <target_col_name> [ , ... ] ) ]
       {
         VALUES ( { <value> | DEFAULT | NULL } [ , ... ] ) [ , ( ... ) ]  |
         <query>
       }
```

Обновляет таблицу, вставляя в нее одну или несколько строк. Значения, вставляемые в каждый столбец таблицы, могут быть указаны явно или получены из вложенного запроса.

Параметры

- `<target_table>` — имя целевой таблицы, в которую будут вставлены строки

- `VALUES ( value | DEFAULT | NULL [ , ⋯ ] ) [ , ( ⋯ ) ]` — указывает одно или несколько значений для вставки в соответствующие столбцы целевой таблицы.
  - `value` — явно указанное значение; может быть литералом или выражением.
  - `DEFAULT` — значение по умолчанию для соответствующего столбца целевой таблицы.
  - `NULL` — пустое значение.

  Значения разделяются запятыми.

  Вы можете вставить несколько строк, указав дополнительные наборы значений в выражении.

- `query` — запрос, который возвращает значения для вставки в соответствующие столбцы. Это позволяет вставлять строки в целевую таблицу из одной или нескольких исходных таблиц.

Пример вставки явно заданных значений

Вставим в таблицу `capitals` значения для столбцов `country` и `capital`:

```
CREATE TABLE capitals (country VARCHAR, capital VARCHAR);
INSERT INTO capitals VALUES
    ('Russia', 'Moscow'),
    ('Italy', 'Rome'),
    ('Spain', 'Madrid'),
    ('France', 'Paris');

SELECT * FROM capitals;
```

```
+---------+---------+
| country | capital |
+---------+---------+
| France  | Paris   |
+---------+---------+
| Italy   | Rome    |
+---------+---------+
| Russia  | Moscow  |
+---------+---------+
| Spain   | Madrid  |
+---------+---------+
```

Пример вставки результатов вложенного запроса

Теперь создадим другую таблицу `capitals_m` и сделаем вставку строк из `capitals`, которые будут результатом вложенного запроса `SELECT`. Вставим такие строки из `capitals`, в которых значение `capital` содержит `M`.

```
CREATE TABLE capitals_m (country VARCHAR, capital VARCHAR);
INSERT INTO capitals_m (country, capital)
    SELECT * FROM capitals
        WHERE capital LIKE '%M%';

SELECT * FROM capitals_m;
```

```
+---------+---------+
| country | capital |
+---------+---------+
| Russia  | Moscow  |
+---------+---------+
```

```
| Spain   | Madrid  |
+---------+---------+
```

Удаление таблицы

```
DROP TABLE [IF EXISTS] <table_name>;
```

Удаляет таблицу с указанным именем.

Опциональный модификатор `IF EXISTS` ограничивает запрос только теми случаями, в которых указанный объект существует.

Удаление всех строк из таблицы

```
TRUNCATE TABLE [IF EXISTS] <table_name>;
```

Удаляет все строки из таблицы, но не удаляет саму таблицу (сохраняются колонки, типы данных колонок, привилегии на таблицу).

Опциональный модификатор `IF EXISTS` ограничивает запрос только теми случаями, в которых указанный объект существует.

Вывод списка всех таблиц

```
SHOW TABLES;
```

Выводит список всех существующих таблиц, доступных пользователю.

Вывод информации о таблице

```
DESC[RIBE] TABLE <table_name>;
```

Выводит всю информацию о таблице — имена столбцов, их типы данных и другие свойства.

Операции со схемами

- `CREATE SCHEMA` — Создание новой схемы
- `ALTER SCHEMA` — Изменение атрибутов схемы
- `DROP SCHEMA` — Удаление схемы

Создание новой схемы

```
CREATE [OR REPLACE] SCHEMA [IF NOT EXISTS] <schema_name>
```

Создает новую схему с указанным именем.

Если указан модификатор `OR REPLACE`, то конечное действие эквивалентно удалению существующей схемы со всеми объектами в ней и созданию новой схемы с тем же именем.

Опциональный модификатор `IF NOT EXISTS` ограничивает запрос только теми случаями, в которых указанный объект еще не существует.

Модификаторы являются взаимоисключающими. Если указать их оба, это приведет к ошибке.

Изменение атрибутов схемы

```
ALTER SCHEMA [IF EXISTS] <schema_name> RENAME TO <new_name>;
```

Изменяет схему с помощью указанного действия.

В настоящее время для изменения схем доступно только одно действие:

- Действие `RENAME TO` переименовывает схему в указанное имя `<new_name>`. Все атрибуты и права при этом сохраняются.

Опциональный модификатор `IF EXISTS` ограничивает запрос только теми случаями, в которых указанный объект существует.

Удаление схемы

```
DROP SCHEMA [IF EXISTS] <schema_name>;
```

Удаляет схему с указанным именем.

Опциональный модификатор `IF EXISTS` ограничивает запрос только теми случаями, в которых указанный объект существует.

Операции с представлениями

- `CREATE VIEW` — Создание нового представления
- `DROP VIEW` — Удаление представления

Создание нового представления

```
CREATE [OR REPLACE] VIEW [IF NOT EXISTS] [<view_schema>.]<view_name> AS
    <select_expr>
```

Создает новое представление с указанным именем на основе результата указанного запроса `SELECT`.

Параметры

- `<view_name>` — имя создаваемого представления

- `<view_schema>` — схема создаваемого представления

- `<select_expr>` — выражение `SELECT`, на основе результата выполнения которого будет создано представление

Если указан модификатор `OR REPLACE`, то конечное действие эквивалентно удалению существующего представления и созданию нового с тем же именем.

Опциональный модификатор `IF NOT EXISTS` ограничивает запрос только теми случаями, в которых указанный объект еще не существует.

Модификаторы являются взаимоисключающими. Если указать их оба, это приведет к ошибке.

Удаление представления

```
DROP VIEW [IF EXISTS] <view_name>;
```

Удаляет представление из системы.

Опциональный модификатор `IF EXISTS` ограничивает запрос только теми случаями, в которых указанный объект существует.

Синтаксис запросов к данным

Выражения SQL

Tengri поддерживает запросы к данным с использованием стандартных ключевых слов SQL и следующего базового синтаксиса:

```
[ WITH ... ]
```

```
SELECT
   ...
[ FROM ...
   [ JOIN ... ]
[ WHERE ... ]
[ GROUP BY ...
   [ HAVING ... ] ]
[ QUALIFY ... ]
[ ORDER BY ... ]
[ LIMIT ... ]
```

Подробные описания для поддерживаемых ключевых слов представлены в разделах:

- `SELECT`
- `WITH`
- `FROM`
- `JOIN`
- `UNION`
- `WHERE`
- `CASE`
- `GROUP BY`
- `HAVING`
- `QUALIFY`
- `ORDER BY`
- `LIMIT`
- `OFFSET`
- `LIKE`
- `ILIKE`
- `SIMILAR TO`
- `AS`
- `DELETE`
- `UPDATE`

Функции

- Агрегатные функции
- Числовые функции
- Текстовые функции
- Функции для регулярных выражений
- Функции для даты и времени

- Функции для JSON
- Функции для двоичных данных
- Функции для геоданных
- Оконные функции
- Утилиты
- Функции модуля Python `tngri`

SELECT

Выражение `SELECT` определяет список столбцов, которые будут возвращены запросом. Хотя оно стоит первым в запросе, логически выражения в этом выражении выполняются последними. Выражение `SELECT` может содержать произвольные выражения, преобразующие вывод, а также агрегатные и оконные функции.

Синтаксис

Выбор всех столбцов

```
[ ... ]
SELECT [ { ALL | DISTINCT } ]
       [ TOP <n> ]
       [{<object_name>|<alias>}.]*

       [ ILIKE '<pattern>' ]

       [ EXCLUDE
         {
           <col_name> | ( <col_name>, <col_name>, ... )
         }
       ]

       [ REPLACE
         {
           ( <expression> AS <col_name> [ , <expression> AS <col_name>, ... ] )
         }
       ]

       [ RENAME
         {
           <col_name> AS <col_alias>
           | ( <col_name> AS <col_alias>, <col_name> AS <col_alias>, ... )
         }
       ]
```

После `SELECT *` можно указать следующие комбинации ключевых слов. Ключевые слова должны быть в указанном ниже порядке:

```
SELECT * ILIKE ... REPLACE ...
```

```
SELECT * ILIKE ... RENAME ...
```

```
SELECT * ILIKE ... REPLACE ... RENAME ...
```

```
SELECT * EXCLUDE ... REPLACE ...
```

```
SELECT * EXCLUDE ... RENAME ...
```

```
SELECT * EXCLUDE ... REPLACE ... RENAME ...
```

```
SELECT * REPLACE ... RENAME ...
```

Выбор определенных столбцов

```
[ ... ]
SELECT [ { ALL | DISTINCT } ]
       [ TOP <n> ]
       {
         [{<object_name>|<alias>}.]<col_name>
         | [{<object_name>|<alias>}.]$<col_position>
         | <expression>
       }
       [ [ AS ] <col_alias> ]
       [ , ... ]
[ ... ]
```

В списке столбцов поддерживается конечная запятая. Например, поддерживается следующее выражение `SELECT`:

```
SELECT emp_id,
       name,
       department,
    FROM employees;
```

Параметры

- `ALL | DISTINCT`

Указывает, следует ли выполнять удаление дубликатов в наборе результатов:

- `ALL` включает все значения в набор результатов.
- `DISTINCT` удаляет дубликаты из набора результатов.

По умолчанию: `ALL`
Подробное описание см. в разделе Выражение `DISTINCT`.

- `<object_name>` или `<alias>`
  Указывает идентификатор объекта или псевдоним объекта (как определено в выражении `FROM`).

- `*` (звездочка)
  Звездочка является сокращением, обозначающим, что в выводе должны быть указаны все столбцы указанного объекта или все столбцы всех объектов, если `*` не сопровождается именем объекта или псевдонимом.
  Подробное описание см. в разделе Выражение со звездочкой `*`.

- `<col_name>`
  Указывает идентификатор столбца (как определено в выражении `FROM`).

- `$<col_position>`
  Указывает положение столбца (начиная с 1) в соответствии с определением в выражении `FROM`. Если на столбец есть ссылка из таблицы, это число не может превышать максимальное количество столбцов в таблице.

- `<expression>`
  Указывает выражение (например математическое), которое вычисляется как определенное значение для любой заданной строки.

- `[ AS ] <col_alias>`
  Указывает псевдоним столбца, назначенный результирующему выражению. Он используется в качестве отображаемого имени в списке `SELECT` верхнего уровня и в качестве имени столбца во встроенном представлении.

Не назначайте такой псевдоним столбца, который будет совпадать с именем другого столбца, на который есть ссылка в запросе. Например, если вы выбираете столбцы с именами `prod_id` и `product_id`, не назначайте псевдоним `product_id` столбцу `prod_id`.

См. также раздел `AS`

- `ILIKE <pattern>`
  Указывает, что в результаты запроса должны быть включены только столбцы, соответствующие заданному шаблону.
  В шаблоне можно использовать следующие выражения SQL:
  - Символ подчеркивания `_` для сопоставления с любым одиночным символом.
  - Символ процента `%` для сопоставления с любой последовательностью из нуля или более символов.

- `EXCLUDE <col_name>` или
  `EXCLUDE (<col_name_1>, <col_name_2>, ⋯)`
  Указывает столбцы, которые должны быть исключены из результатов запроса.

- `REPLACE (<expression> AS <col_name> [ , <expression> AS <col_name>, ⋯] )`
  Replaces the value of col_name with the value of the evaluated expression expr. Заменяет значение указанного столбца на результат применения выражения `<expression>` к его исходному значению.

- `RENAME <col_name> AS <col_alias>` или
  `RENAME (<col_name_1> AS <col_alias_1>, <col_name_2> AS <col_alias_2>, ⋯)`
  Указывает псевдонимы столбцов, которые будут использованы в результатах запроса.

- `TOP <n>`
  Указывает максимальное количество строк, которые будут результатом запроса.

Примеры

- Выбираем все столбцы из таблицы с именем `my_table`:

```
SELECT * FROM my_table;
```

- Выполняем арифметические действия над столбцами таблицы и указываем псевдоним:

```
SELECT column_1 + column_2 AS sum, sqrt(column_1) AS sq_root FROM my_table;
```

- Используем префиксные псевдонимы для получения того же результата:

```
SELECT
    sum: column_1 + column_2,
    sq_root: sqrt(column_1)
FROM my_table;
```

- Выбираем все уникальные имена из таблицы `employees` (*работники*):

```
SELECT DISTINCT name FROM employees;
```

- Выводим общее количество строк в таблице `employees`:

```
SELECT count(*) FROM employees;
```

- Выбираем все столбцы, кроме столбца `name`, из таблицы `employees`:

```
SELECT * EXCLUDE (name) FROM employees;
```

- Выбираем все столбцы из таблицы `employees`, но заменяем `name` на результат применения функции `lower(name)`:

```
SELECT * REPLACE (lower(name) AS name) FROM employees;
```

- Выбираем из таблицы все столбцы, соответствующие заданному регулярному выражению:

```
SELECT COLUMNS('number\d+') FROM employees;
```

- Вычисляем функцию по всем заданным столбцам таблицы:

```
SELECT min(COLUMNS(*)) FROM employees;
```

- Используем двойные кавычки ("), чтобы выбрать столбцы с пробелами или специальными символами:

```
SELECT "Фамилия Имя Отчество" FROM employees;
```

Список столбцов `SELECT`

Выражение `SELECT` содержит список выражений, которые определяют результат запроса. Список `SELECT` может ссылаться на любые столбцы в выражении `FROM` и объединять их с помощью выражений. Поскольку результатом SQL-запроса является таблица, каждое выражение в выражении `SELECT` также имеет имя. Выражения могут быть явно названы с помощью оператора `AS` (например, `expr AS name`). Если пользователь не указал имя, то выражения именуются системой автоматически.

Имена столбцов не чувствительны к регистру, если они задаются без кавычек.

Выражение со звездочкой `*`

Выражение со звездочкой (`*`) — это специальное выражение, которое расширяется до множества выражений на основе содержимого выражения `FROM`. В простейшем случае расширяется до всех выражений в выражении `FROM`.

- Выбираем все столбцы из таблицы с именем `my_table`:

```
SELECT * FROM my_table;
```

Выражение `DISTINCT`

Выражение `DISTINCT` можно использовать для получения только уникальных строк в результате — таким образом, все дубликаты будут отфильтрованы.

- Выбираем все уникальные имена из таблицы `employees`:

```
SELECT DISTINCT name FROM employees;
```

Запросы, начинающиеся с `SELECT DISTINCT`, выполняют дедупликацию, которая является дорогостоящей операцией. Поэтому используйте `DISTINCT` только в случае необходимости.

Выражение `DISTINCT ON`

Выражение `DISTINCT ON` возвращает только одну строку для каждого уникального значения в наборе выражений, как определено в выражении `ON`. Если присутствует условие `ORDER BY`, то возвращается первая строка, которая встречается в соответствии с условием `ORDER BY`. Если условие `ORDER BY` отсутствует, первая встречающаяся строка не определена и может быть любой строкой в таблице.

Выбираем сотрудника с наибольшей зарплатой для каждого отдела:

```
SELECT DISTINCT ON(department) name, salary
    FROM employees
    ORDER BY salary DESC;
```

При запросе больших наборов данных использование `DISTINCT` для всех столбцов может быть дорогостоящей операцией. Поэтому рассмотрите возможность использования `DISTINCT ON` для столбца (или набора столбцов), гарантирующего достаточную степень уникальности результатов. Например, использование `DISTINCT ON` для ключевого столбца (столбцов) таблицы гарантирует полную уникальность.

Агрегатные функции

Агрегатные функции — это функции, которые объединяют значения из нескольких строк в одно. Когда агрегатные функции присутствуют в выражении `SELECT`, запрос превращается в агрегатный запрос. В агрегатном запросе **все** выражения должны быть либо частью агрегатной функции, либо частью группы (как указано в выражении `GROUP BY`).

- Получаем общее количество строк в таблице сотрудников:

```
SELECT count(*) FROM employees;
```

- Получаем общее количество строк в таблице сотрудников, сгруппированных по отделам:

```
SELECT department, count(*)
    FROM employees
    GROUP BY department;
```

Подробное описание см. в разделе Агрегатные функции.

Оконные функции

Оконные функции — это функции, которые выполняют вычисления для набора строк, некоторым образом связанных с текущей строкой. Вызов оконной функции всегда содержит выражение `OVER`, следующее за названием и аргументами оконной функции. Выражение `OVER` определяет, как именно нужно разделить строки запроса для обработки оконной функцией.

- Получаем столбец `row_number`, содержащий инкрементные идентификаторы для каждой строки таблицы зарплат:

```
SELECT row_number() OVER ()
    FROM salaries;
```

- Вычисляем разницу между текущей и предыдущей суммой в порядке убывания времени:

```
SELECT amount - lag(amount) OVER (ORDER BY time)
    FROM salaries;
```

Подробное описание см. в разделе Оконные функции.

`WITH`

Выражение `WITH` позволяет указать общие табличные выражения (CTE). Регулярные (нерекурсивные) общие табличные выражения — это, по сути, представления (view), которые ограничены рамками конкретного запроса. CTE могут ссылаться друг на друга и быть вложенными. Рекурсивные CTE могут ссылаться сами на себя.

Синтаксис

Подзапрос

```
[ WITH
       <cte_name1> [ ( <cte_column_list> ) ] AS ( SELECT ...  )
   [ , <cte_name2> [ ( <cte_column_list> ) ] AS ( SELECT ...  ) ]
   [ , <cte_nameN> [ ( <cte_column_list> ) ] AS ( SELECT ...  ) ]
]
SELECT ...
```

Рекурсивное CTE:

```
[ WITH [ RECURSIVE ]
    <cte_name1> ( <cte_column_list> ) AS ( anchorClause UNION ALL recursiveClause)
 [, <cte_name2> ( <cte_column_list> ) AS ( anchorClause UNION ALL recursiveClause)]
 [, <cte_nameN> ( <cte_column_list> ) AS ( anchorClause UNION ALL recursiveClause)]
]
SELECT ...
```

где:

```
anchorClause ::=
    SELECT <anchor_column_list> FROM ...

recursiveClause ::=
    SELECT <recursive_column_list> FROM ... [ JOIN ... ]
```

Параметры

- `<cte_name1>` , `<cte_nameN>`
  Имя CTE.

- `<cte_column_list>`
  Имена столбцов в CTE (общее табличное выражение).

- `<anchor_column_list>`
  Столбцы, используемые в анкерном выражении для рекурсивного CTE. Столбцы в этом списке должны соответствовать столбцам, определенным в `<cte_column_list>`.

- `<recursive_column_list>`

Столбцы, используемые в рекурсивном выражении для рекурсивного CTE. Столбцы в этом списке должны соответствовать столбцам, определенным в `<cte_column_list>`.

Примеры базовых CTE

Создадим CTE с именем `cte` и используем его в основном запросе:

```
WITH cte AS (SELECT 33 AS amount)
    SELECT * FROM cte;
```

```
+--------+
| amount |
+--------+
| 33     |
+--------+
```

Создадим два CTE `cte1` и `cte2`, где второй CTE ссылается на первый CTE:

```
WITH
    cte1 AS (SELECT 33 AS i),
    cte2 AS (SELECT i * 2 AS amount_doubled FROM cte1)
SELECT * FROM cte2;
```

```
+----------------+
| amount_doubled |
+----------------+
| 66             |
+----------------+
```

Для столбцов CTE можно задавать имена:

```
WITH my_cte(amount) AS (SELECT 33 AS i)
SELECT * FROM my_cte;
```

```
+--------+
| amount |
+--------+
| 33     |
+--------+
```

FROM

Выражение `FROM` указывает источник данных, с которыми должна работать остальная часть запроса. С точки зрения логики, выражение `FROM` — это место, с которого начинается выполнение

запроса.

Выражение `FROM` может содержать одну таблицу, комбинацию из нескольких таблиц, объединенных с помощью выражения `JOIN`, или другой запрос `SELECT` в узле подзапроса.

Синтаксис

```
SELECT ...
FROM objectReference [ JOIN objectReference [ ... ] ]
[ ... ]
```

где:

```
objectReference ::=
   {
      [<namespace>.]<object_name>
           [ AT | BEFORE ( <object_state> ) ]
           [ CHANGES ( <change_tracking_type> ) ]
           [ MATCH_RECOGNIZE ]
           [ PIVOT | UNPIVOT ]
           [ [ AS ] <alias_name> ]
     | <table_function>
           [ PIVOT | UNPIVOT ]
           [ [ AS ] <alias_name> ]
     | ( VALUES (...) )
     | ( <subquery> )
           [ [ AS ] <alias_name> ]
     | DIRECTORY( @<stage_name> )
   }
```

Параметры

- `[<namespace>.]<object_name>`
  Указывает имя объекта (таблицы или представления), к которому производится запрос.

- `<table_function>`
  Указывает системную табличную функцию, табличную функцию UDF или метод класса для вызова в выражении `FROM`.

- `VALUES`
  Выражение `VALUES` может содержать литеральные значения или выражения, которые будут использоваться в выражении `FROM`. Это выражение также может содержать псевдонимы таблиц и столбцов (не показаны на схеме выше).

- `<subquery>`
  Подзапрос в выражении `FROM`.

- `DIRECTORY( @stage_name )`
  Указывает имя этапа, включающего таблицу каталогов.

- `[ AS ] <alias_name>`
  Указывает имя, заданное для объекта, к которому она относится. Может использоваться с любыми другими подзапросами в выражении `FROM`. Оператор `AS` может опускаться.

- `JOIN`
  Указывает на выполнение соединения между двумя (или более) таблицами (или представлениями или табличными функциями). Соединение может быть внутренним внешним или иметь другой тип. Соединение может использовать ключевое слово `JOIN` или альтернативный поддерживаемый синтаксис соединения.
  Подробное описание см. в разделе `JOIN`.

Примеры

- Выбираем все столбцы из таблицы с именем `my_table`:

```
SELECT *
FROM my_table;
```

- Выбираем все столбцы из таблицы с именем `my_table` через псевдоним `mt`:

```
SELECT mt.*
FROM my_table mt;
```

- Используем префиксный псевдоним:

```
SELECT mt.*
FROM mt: my_table;
```

- Выбираем все столбцы из таблицы `my_table` в схеме `my_schema`:

```
SELECT *
FROM my_schema.my_table;
```

- Выбираем столбец `i` из табличной функции `range`, где первый столбец функции диапазона переименован в `i`:

```
SELECT t.i
FROM range(1000) AS t(i);
```

- Выбираем все столбцы из подзапроса:

```
SELECT *
FROM (SELECT * FROM my_table);
```

- Объединяем две таблицы:

```
SELECT *
FROM my_table
JOIN other_table
    ON my_table.key = other_table.key;
```

  Подробное описание см. в разделе `JOIN`.

`JOIN`

Соединения (*JOINS*) — это фундаментальная реляционная операция, используемая для горизонтального соединения двух таблиц или отношений. Отношения называются левая и правая стороны соединения в зависимости от того, как они записаны в выражении `JOIN`. Каждая строка результата содержит столбцы из обоих отношений.

В соединении используется правило для сопоставления пар строк из каждого отношения. Часто это предикат, но могут быть указаны и другие правила.

Синтаксис

Могут использоваться следующие варианты синтаксиса:

```
SELECT ...
FROM <object_ref1> [
                     {
                       INNER
                       | { LEFT | RIGHT | FULL } [ OUTER ]
                     }
                   ]
                   JOIN <object_ref2>
  [ ON <condition> ]
[ ... ]
```

```
SELECT *
FROM <object_ref1> [
                     {
                       INNER
```

```
                      | { LEFT | RIGHT | FULL } [ OUTER ]
                    }
                  ]
                  JOIN <object_ref2>
  [ USING( <column_list> ) ]
[ ... ]
```

```
SELECT ...
FROM <object_ref1> [
                    {
                      | NATURAL [ { LEFT | RIGHT | FULL } [ OUTER ] ]
                      | CROSS
                    }
                  ]
                  JOIN <object_ref2>
[ ... ]
```

Параметры

- `<object_ref1>` и `<object_ref2>`
  Каждый `<object_ref>` представляет собой таблицу или источник данных, подобный таблице.

- `JOIN`
  Ключевое слово `JOIN`, указывающее, что таблицы должны быть соединены. `JOIN` комбинируется с другими ключевыми словами (например, `INNER` или `OUTER`), чтобы указать тип соединения.

- `ON <condition>`
  Булево выражение, которое определяет условия соединения.
  Подробное описание см. в разделе Условное соединение.

- `USING <column_list>`
  Список столбцов, по которым производится соединение. Столбцы должны быть одноименными в соединяемых таблицах.
  Подробное описание см. в разделе Условное соединение.

`OUTER JOIN` — внешнее соединение

Строки, не имеющие совпадений, могут быть возвращены, если указано соединение `OUTER`. Внешние соединения могут иметь один из типов:

- `LEFT` (Все строки из левого отношения встречаются хотя бы один раз)
- `RIGHT` (Все строки из правого отношения встречаются хотя бы один раз)
- `FULL` (Все строки из обоих отношений встречаются хотя бы один раз)

Соединение, которое не является внешним (`OUTER`), является внутренним (`INNER`) — возвращаются только те строки, которые удовлетворяют условию.

Если возвращается непарная строка, атрибуты другой таблицы устанавливаются в значение `NULL`.

`CROSS JOIN` — перекрёстное соединение (декартово произведение)

Простейшим типом соединения является `CROSS JOIN`. Для этого типа соединения нет никаких условий, и он просто возвращает все возможные пары.

Вернем все пары строк:

```
SELECT a.*, b.*
FROM a
CROSS JOIN b;
```

Это эквивалентно запросу, в котором просто опущено условие `JOIN`:

```
SELECT a.*, b.*
FROM a, b;
```

Условное соединение

Большинство соединений задается предикатом, который соединяет атрибуты с одной стороны с атрибутами с другой стороны. Условия могут быть явно указаны с помощью операторов `ON` и `WHERE`.

Покажем использование условных соединений на примере двух таблиц — `capitals` со странами и их столицами и `population` со странами и их населением (в миллионах).

```
CREATE TABLE capitals (cap_country VARCHAR, capital VARCHAR);
CREATE TABLE population (pop_country VARCHAR, population_mil BIGINT);
INSERT INTO capitals VALUES
    ('Russia', 'Moscow'),
    ('Italy', 'Rome'),
    ('Spain', 'Madrid'),
    ('France', 'Paris');
INSERT INTO population VALUES
    ('Russia', 143),
    ('Spain', 48),
    ('Brazil', 211);
```

Выведем страны с их столицей и населением, используя `JOIN` с оператором `ON`:

```
SELECT *
FROM capitals
```

```
JOIN population ON (cap_country = pop_country);
```

```
+-------------+---------+-------------+----------------+
| cap_country | capital | pop_country | population_mil |
+-------------+---------+-------------+----------------+
| Russia      | Moscow  | Russia      | 143            |
+-------------+---------+-------------+----------------+
| Spain       | Madrid  | Spain       | 48             |
+-------------+---------+-------------+----------------+
```

Теперь выведем те же данные, используя `JOIN` с оператором `WHERE`:

```
SELECT t1.*, t2.*
FROM capitals t1, population t2
WHERE t1.cap_country = t2.pop_country
```

```
+-------------+---------+-------------+----------------+
| cap_country | capital | pop_country | population_mil |
+-------------+---------+-------------+----------------+
| Russia      | Moscow  | Russia      | 143            |
+-------------+---------+-------------+----------------+
| Spain       | Madrid  | Spain       | 48             |
+-------------+---------+-------------+----------------+
```

С оператором `WHERE` можно использовать не только равенство, но и другие предикаты.

Если имена столбцов в двух таблицах одинаковы и значения в них должны быть равны, то можно использовать более простой синтаксис `USING`.

Зададим те же таблицы, но с одноименными столбцами `country`:

```
CREATE TABLE capitals (country VARCHAR, capital VARCHAR);
CREATE TABLE population (country VARCHAR, population_mil BIGINT);
INSERT INTO capitals VALUES
    ('Russia', 'Moscow'),
    ('Italy', 'Rome'),
    ('Spain', 'Madrid'),
    ('France', 'Paris');
INSERT INTO population VALUES
    ('Russia', 143),
    ('Spain', 48),
    ('Brazil', 211);
```

Теперь выведем страны с их столицей и населением через более простой запрос с `USING`:

```
SELECT *
FROM capitals
```

```
JOIN population USING (country);
```

```
+---------+---------+----------------+
| country | capital | population_mil |
+---------+---------+----------------+
| Russia  | Moscow  | 143            |
+---------+---------+----------------+
| Spain   | Madrid  | 48             |
+---------+---------+----------------+
```

`NATURAL JOIN` — естественное соединение

Естественные соединения объединяют две таблицы на основе столбцов с одинаковыми именами.

Покажем использование такого типа соединения на примере той же пары таблиц с одноименными столбцами `country`.

Чтобы объединить таблицы по их общему столбцу, выполним команду:

```
SELECT *
FROM capitals
NATURAL JOIN population;
```

```
+---------+---------+---------+----------------+
| country | capital | country | population_mil |
+---------+---------+---------+----------------+
| Russia  | Moscow  | Russia  | 143            |
+---------+---------+---------+----------------+
| Spain   | Madrid  | Spain   | 48             |
+---------+---------+---------+----------------+
```

Обратите внимание, что в результат были включены только строки, в которых в обеих таблицах присутствовал один и тот же атрибут `country`.

Мы можем выполнить аналогичный запрос с помощью выражения `JOIN` с ключевым словом `USING`:

```
SELECT *
FROM capitals
JOIN population
USING (country);
```

```
+---------+---------+----------------+
| country | capital | population_mil |
+---------+---------+----------------+
| Russia  | Moscow  | 143            |
+---------+---------+----------------+
```

```
| Spain   | Madrid  | 48             |
+---------+---------+----------------+
```

Обратите внимание, что в таком случае одноименные колонки из разных таблиц "схлопнутся" в одну.

`SEMI JOIN` и `ANTI JOIN` — полусоединение и антисоединение

Полусоединения возвращают строки из левой таблицы, которые имеют хотя бы одно совпадение в правой таблице.

Антисоединения возвращают строки из левой таблицы, которые не имеют ни одного совпадения в правой таблице.

При использовании полу- или антисоединений результат никогда не будет содержать больше строк, чем в левой таблице. Полусоединения обеспечивают ту же логику, что и оператор `IN`. Антисоединения обеспечивают ту же логику, что и оператор `NOT IN`, за исключением того, что антисоединения игнорируют значения `NULL` из правой таблицы.

Пример полусоединения

Выведем список таких пар "страна-столица" из таблицы `capitals`, для которых название страны присутствует в таблице `population`:

```
SELECT *
FROM capitals
SEMI JOIN population
    USING (country);
```

```
+---------+---------+
| country | capital |
+---------+---------+
| Russia  | Moscow  |
+---------+---------+
| Spain   | Madrid  |
+---------+---------+
```

Этот запрос эквивалентен следующему:

```
SELECT *
FROM capitals
WHERE country IN (SELECT country FROM population);
```

```
+---------+---------+
```

```
| country | capital |
+---------+---------+
| Russia  | Moscow  |
+---------+---------+
| Spain   | Madrid  |
+---------+---------+
```

Пример антисоединения

Выведем список таких пар "страна-столица" из таблицы `capitals`, для которых названия страны нет в таблице `population`:

```
SELECT *
FROM capitals
ANTI JOIN population
    USING (country);
```

```
+---------+---------+
| country | capital |
+---------+---------+
| Italy   | Rome    |
+---------+---------+
| France  | Paris   |
+---------+---------+
```

Этот запрос эквивалентен следующему:

```
SELECT *
FROM capitals
WHERE country NOT IN
    (SELECT country FROM population WHERE country IS NOT NULL);
```

```
+---------+---------+
| country | capital |
+---------+---------+
| Italy   | Rome    |
+---------+---------+
| France  | Paris   |
+---------+---------+
```

Замкнутое соединение (Self-Join)

Tengri позволяет использовать замкнутое соединение (соединение таблицы с самой собой) для всех типов соединений. Обратите внимание, что таблицы должны быть указаны через псевдонимы, использование одного и того же имени таблицы без псевдонимов приведет к ошибке:

```
CREATE TABLE t (x INTEGER);
SELECT * FROM t JOIN t USING(x);
```

```
Binder Error:
Duplicate alias "t" in query!
```

Добавление псевдонимов позволяет успешно обработать запрос:

```
CREATE TABLE t (num INTEGER);
SELECT * FROM t t1 JOIN t t2 USING(num);
```

```
+-----+
| num |
+-----+
```

`UNION`

Выражение `UNION` добавляет результаты одного запроса к результатам другого. При этом дублирующиеся строки убираются из результата, если только не добавлен оператор `ALL`.

Синтаксис

```
SELECT ...
UNION [ALL]
SELECT ...
[UNION [ALL]
SELECT ...
...
];
```

Примеры

- Выведем список всех стран, которые встречаются в столбцах `country` в двух таблицах — столиц и количества населения:

```
CREATE TABLE capitals (country VARCHAR, capital VARCHAR);
CREATE TABLE population (country VARCHAR, population_mil BIGINT);
INSERT INTO capitals VALUES
    ('Russia', 'Moscow'),
    ('Italy', 'Rome'),
    ('Spain', 'Madrid'),
    ('France', 'Paris');
INSERT INTO population VALUES
```

```
    ('Russia', 143),
    ('Spain', 48),
    ('Brazil', 211);

SELECT country FROM capitals
UNION
SELECT country FROM population;
```

```
+---------+
| country |
+---------+
| Russia  |
+---------+
| Italy   |
+---------+
| France  |
+---------+
| Spain   |
+---------+
| Brazil  |
+---------+
```

- Теперь для тех же двух таблиц выведем список стран, которые встречаются в столбцах `country`, но без сокращения повторяющихся вхождений. Для этого используем оператор `ALL`:

```
SELECT country FROM capitals
UNION ALL
SELECT country FROM population;
```

```
+---------+
| country |
+---------+
| Russia  |
+---------+
| Italy   |
+---------+
| Spain   |
+---------+
| France  |
+---------+
| Russia  |
+---------+
| Spain   |
+---------+
| Brazil  |
+---------+
```

`WHERE`

Выражение `WHERE` определяет фильтры, которые будут применены к данным. Это позволяет выбрать только ту часть данных, которая вас интересует. Логически условие `WHERE` применяется сразу после условия `FROM`.

Синтаксис

```
...
WHERE <predicate>
[ ... ]
```

Параметры

- `<predicate>`
  Булево выражение. Выражение может содержать логические операторы, такие как `AND`, `OR` и `NOT`.

Примеры

- Выберем из таблицы столиц строки, в которых значение `country` — `'Italy'`:

```
SELECT *
FROM capitals
WHERE country = 'Italy';
```

```
+---------+---------+
| country | capital |
+---------+---------+
| Italy   | Rome    |
+---------+---------+
```

- Выберем из таблицы дней недели строки, которые соответствуют заданному выражению `LIKE`, чувствительному к регистру:

```
SELECT *
FROM weekdays
WHERE name LIKE '%S%';
```

```
+--------+----------+
| number |   name   |
+--------+----------+
| 6      | Saturday |
+--------+----------+
| 7      | Sunday   |
```

```
+--------+----------+
```

- Выберем из таблицы дней недели строки, которые соответствуют заданному выражению `ILIKE`, не чувствительному к регистру:

```
SELECT *
FROM weekdays
WHERE name ILIKE '%S%';
```

```
+--------+-----------+
| number |    name   |
+--------+-----------+
| 2      | Tuesday   |
+--------+-----------+
| 3      | Wednesday |
+--------+-----------+
| 4      | Thursday  |
+--------+-----------+
| 6      | Saturday  |
+--------+-----------+
| 7      | Sunday    |
+--------+-----------+
```

- Выберем все строки, соответствующие заданному составному выражению:

```
SELECT *
FROM weekdays
WHERE number > 5 OR number = 3;
```

```
+--------+-----------+
| number |    name   |
+--------+-----------+
| 3      | Wednesday |
+--------+-----------+
| 6      | Saturday  |
+--------+-----------+
| 7      | Sunday    |
+--------+-----------+
```

`CASE`

Описание

Выражение `CASE` представляет собой общее условное выражение. Условия записываются через конструкции `WHEN ⋯ THEN`.

Синтаксис

```
CASE
    WHEN <condition1> THEN <result1>
  [ WHEN <condition2> THEN <result2> ]
  [ ... ]
  [ ELSE <result3> ]
END
```

```
CASE <expression>
    WHEN <value1> THEN <result1>
  [ WHEN <value2> THEN <result2> ]
  [ ... ]
  [ ELSE <result3> ]
END
```

Параметры

- `<condition>`
  Указывает условие — выражение, которое должно вычисляться в булево значение (True, False или NULL).

- `<expression>`
  Общее выражение.

- `<value>`
  Значение, совпадение с которым проверяется для выражения `<expression>`. Значение может быть литералом или выражением. Значение должно иметь тот же тип данных, что и `<expression>`, или иметь тип данных, который можно преобразовать в тип данных `<expression>`.

- `<result>`
  Результат, который возвращает выражение. Если одно условие истинно, то возвращается соответствующий результат. Если истинно более одного условия, то возвращается результат, заданный в первом истинном условии. Если присутствует опциональный оператор `ELSE` и не найдено ни одного совпадения в `WHEN`, то выражение возвращает результат по `ELSE`. Если оператор `ELSE` отсутствует и не найдено ни одного совпадения в `WHEN`, то результатом будет `NULL`.

Примеры

- Создадим таблицу из чисел с `1` до `10`. В колонку `result` запишем результат сравнения чисел с числом `5`.

```
CREATE TABLE demo.numbers AS
    SELECT unnest(generate_series(1,10)) AS number;

SELECT
    number,
    CASE
        WHEN number < 5 THEN 'less than 5'
        WHEN number > 5 THEN 'more than 5'
        ELSE 'equal to 5'
    END AS result
FROM demo.numbers;
```

```
+--------+-------------+
| number | result      |
+--------+-------------+
| 1      | less than 5 |
+--------+-------------+
| 2      | less than 5 |
+--------+-------------+
| 3      | less than 5 |
+--------+-------------+
| 4      | less than 5 |
+--------+-------------+
| 5      | equal to 5  |
+--------+-------------+
| 6      | more than 5 |
+--------+-------------+
| 7      | more than 5 |
+--------+-------------+
| 8      | more than 5 |
+--------+-------------+
| 9      | more than 5 |
+--------+-------------+
| 10     | more than 5 |
+--------+-------------+
```

- Создадим таблицу городов с указанием стран и населения. Подсчитаем для городов из России среднее значение населения, а для городов из Испании — максимальное значение:

```
CREATE TABLE demo.cities(city_name VARCHAR, city_country VARCHAR, city_population BIGINT);

INSERT INTO demo.cities VALUES
    ('Moscow', 'Russia', 13000000),
    ('Saint Petersburg', 'Russia', 5600000),
    ('Madrid', 'Spain', 3400000),
    ('Barcelona', 'Spain', 1700000);

SELECT
    avg(CASE WHEN city_country == 'Russia' THEN city_population END)
```

```
    AS "Russia avg city population",
    max(CASE WHEN city_country == 'Spain' THEN city_population END)
    AS "Spain max city population"
FROM demo.cities;
```

```
+----------------------------+--------------------------+
| russia avg city population | spain max city population |
+----------------------------+--------------------------+
| 9300000                    | 3400000                  |
+----------------------------+--------------------------+
```

GROUP BY

Выражение `GROUP BY` определяет, какие столбцы должны использоваться для группировки при выполнении любых агрегаций в выражении `SELECT`. Если указано условие `GROUP BY`, запрос всегда является агрегированным, даже если в условии `SELECT` нет явно заданного агрегирования.

Если указано условие `GROUP BY`, все кортежи, имеющие совпадающие данные в группирующих столбцах (т. е. все кортежи, принадлежащие одной группе), будут объединены. Значения самих группирующих столбцов не изменяются, а любые другие столбцы могут быть объединены с помощью агрегатной функции (например, `count`, `sum`, `avg`, и т. д.).

GROUP BY ALL

Используйте `GROUP BY ALL` для группировки всех столбцов в выражении `SELECT`, которые не обернуты в агрегатные функции. Это упрощает синтаксис, позволяя хранить список столбцов в одном месте, и предотвращает ошибки, сохраняя гранулярность `SELECT` в соответствии с гранулярностью `GROUP BY` (например, предотвращает дублирование).

Синтаксис

Могут использоваться следующие варианты синтаксиса:

```
SELECT ...
  FROM ...
  [ ... ]
  GROUP BY groupItem [ , groupItem [ , ... ] ]
  [ ... ]
```

```
SELECT ...
  FROM ...
  [ ... ]
  GROUP BY ALL
  [ ... ]
```

где:

```
groupItem ::= { <column_alias> | <position> | <expression> }
```

Параметры

- `<column_alias>`
  Псевдоним столбца, заданный в выражении `SELECT`.

- `<position>`
  Позиция выражения в выражении `SELECT`.

- `<expression>`
  Любое выражение, заданное на таблицах в текущей области видимости.

- `GROUP BY ALL`
  Указывает, что все элементы в выражении `SELECT`, которые не используют агрегатные функции, должны использоваться для группировки.

Примеры

- Вычислим количество записей в таблице `employees` (*работников*), относящихся к каждому отделу:

```
SELECT department, count(*)
    FROM employees
    GROUP BY department;
```

- Вычислим среднюю зарплату для каждого отдела каждого подразделения:

```
SELECT division, department, avg(salary)
    FROM employees
    GROUP BY division, department;
```

- Сгруппируем данные по отделам и подразделениям, чтобы увидеть все уникальные пары "отдел — подразделение":

```
SELECT division, department
    FROM employees
    GROUP BY ALL;
```

HAVING

Выражение `HAVING` может использоваться после выражения `GROUP BY` для задания критериев фильтрации после завершения группировки. По синтаксису выражение `HAVING` идентично выражению `WHERE`, но если выражение `WHERE` используется до группировки, то выражение `HAVING` — после нее.

Синтаксис

```
SELECT ...
FROM ...
GROUP BY ...
HAVING <predicate>
[ ... ]
```

Параметры

- `<predicate>`
  Булево выражение.

Примеры

- Подсчитаем количество записей в таблице `employees` (*работников*), которые содержат каждый из различных отделов, отбрасывая отделы с количеством меньше 10:

  ```
  SELECT department, count(*)
  FROM employees
  GROUP BY department
  HAVING count(*) >= 10;
  ```

- Вычислим среднюю зарплату для каждого отдела каждого подразделения , но только для тех отделов, где средняя зарплата больше, чем медианная зарплата:

  ```
  SELECT division, department, avg(salary)
  FROM employees
  GROUP BY division, department
  HAVING avg(salary) > median(salary);
  ```

QUALIFY

Выражение `QUALIFY` используется для фильтрации результатов оконных функций. Эта фильтрация результатов аналогична тому, как выражение `HAVING` фильтрует результаты агрегатных функций, применяемых в запросах с выражением `GROUP BY`.

Выражение `QUALIFY` позволяет избежать необходимости использования подзапроса или выражения

с `WITH` для выполнения этой фильтрации (так же, как `HAVING` позволяет избегать подзапросов).

Синтаксис

```
QUALIFY <predicate>
```

Общая форма выражений с `QUALIFY` обычно выглядит так (допускаются некоторые вариации в порядке):

```
SELECT <column_list>
  FROM <data_source>
  [GROUP BY ...]
  [HAVING ...]
  QUALIFY <predicate>
  [ ... ]
```

Параметры

- `<column_list>`
  Список выражения `SELECT`.

- `<data_source>`
  Источником данных обычно является таблица, но это может быть и другой источник данных, подобный таблице, например представление, пользовательская табличная функция и т.д.

- `<predicate>`
  Предикат — это выражение, которое фильтрует результат после вычисления агрегатных и оконных функций. Предикат подобен выражению `HAVING`, но без самого ключевого слова `HAVING`. Кроме того, предикат может содержать оконные функции.

Примеры

Создадим и заполним таблицу:

```
CREATE TABLE qt (i INTEGER, p CHAR(1), o INTEGER);
INSERT INTO qt (i, p, o) VALUES
    (1, 'A', 1),
    (2, 'A', 2),
    (3, 'B', 1),
    (4, 'B', 2);
```

В этом запросе используется вложенная структура, а не `QUALIFY`:

```
SELECT *
    FROM (
         SELECT i, p, o,
                ROW_NUMBER() OVER (PARTITION BY p ORDER BY o) AS row_num
            FROM qt
        )
    WHERE row_num = 1;
```

```
+---+---+---+---------+
| I | P | O | ROW_NUM |
|---+---+---+---------|
| 1 | A | 1 |       1 |
| 3 | B | 1 |       1 |
+---+---+---+---------+
```

А в этом запросе используется `QUALIFY`:

```
SELECT i, p, o
    FROM qt
    QUALIFY ROW_NUMBER() OVER (PARTITION BY p ORDER BY o) = 1;
```

```
+---+---+---+
| I | P | O |
|---+---+---|
| 1 | A | 1 |
| 3 | B | 1 |
+---+---+---+
```

Используем `QUALIFY` для ссылки на оконные функции, которые находятся в выражении `SELECT`:

```
SELECT i, p, o, ROW_NUMBER() OVER (PARTITION BY p ORDER BY o) AS row_num
    FROM qt
    QUALIFY row_num = 1;
```

```
+---+---+---+---------+
| I | P | O | ROW_NUM |
|---+---+---+---------|
| 1 | A | 1 |       1 |
| 3 | B | 1 |       1 |
+---+---+---+---------+
```

Выражение `QUALIFY` также может сочетаться с агрегатными функциями и может содержать подзапросы:

```
SELECT c2, SUM(c3) OVER (PARTITION BY c2) AS r
    FROM t1
    WHERE c3 < 4
    GROUP BY c2, c3
    HAVING sum(c1) > 3
    QUALIFY r IN (
        SELECT min(c1)
        FROM test
        GROUP BY c2
        HAVING min(c1) > 3);
```

ORDER BY

Выражение `ORDER BY` является модификатором вывода. Логически оно применяется в самом конце запроса (непосредственно перед `LIMIT`, если оно присутствует). Выражение `ORDER BY` сортирует строки по критериям сортировки в порядке возрастания или убывания. Кроме того, в каждом выражении `ORDER BY` можно указать, следует ли перемещать значения `NULL` в начало или в конец.

Выражение `ORDER BY` может содержать одно или несколько выражений, разделенных запятыми. Если выражений нет, то будет выдана ошибка. Выражения могут начинаться либо с произвольного скалярного выражения (которое может быть именем столбца), либо с номера позиции столбца (где индексация начинается с 1), либо с ключевого слова `ALL`. За каждым выражением может следовать модификатор порядка (`ASC` или `DESC`, по умолчанию — `ASC`), и/или модификатор порядка `NULL` (`NULLS FIRST` или `NULLS LAST`, по умолчанию — `NULLS LAST`).

ORDER BY ALL

Ключевое слово `ALL` указывает, что вывод должен быть отсортирован по каждому столбцу в порядке слева направо. Направление сортировки может быть изменено с помощью `ORDER BY ALL ASC` или `ORDER BY ALL DESC` и/или `NULLS FIRST` или `NULLS LAST`. Обратите внимание, что `ALL` не может использоваться в сочетании с другими выражениями в выражении `ORDER BY` — оно должно быть само по себе.

Модификатор порядка значений NULL

По умолчанию сортировка производится с параметрами `ASC` и `NULLS LAST`, то есть значения сортируются в порядке возрастания, а значения `NULL` располагаются последними. Это идентично порядку сортировки по умолчанию в PostgreSQL. Порядок сортировки по умолчанию можно изменить с помощью следующих параметров конфигурации.

Используйте параметр `default_null_order`, чтобы изменить порядок сортировки по умолчанию `NULL` на один из следующих вариантов:

- `NULLS_FIRST`
- `NULLS_LAST`
- `NULLS_FIRST_ON_ASC_LAST_ON_DESC`
- `NULLS_LAST_ON_ASC_FIRST_ON_DESC`:

Например:

```
SET default_null_order = 'NULLS_FIRST';
```

Используйте параметр `default_order`, чтобы изменить направление сортировки по умолчанию на один из следующих вариантов:

- `DESC`
- `ASC`

Например:

```
SET default_order = 'DESC';
```

Коллации (схемы сопоставления)

Текст по умолчанию сортируется с использованием коллации двоичного сравнения. Это означает, что значения сортируются по их двоичным представлениям в UTF-8. Хотя это хорошо работает для ASCII-текста (например, для данных на английском языке), порядок сортировки может быть неправильным для других языков. Для этой цели предусмотрены коллации.

Синтаксис

```
SELECT ...
FROM ...
ORDER BY orderItem [ , orderItem , ... ]
[ ... ]
```

где:

```
orderItem ::= { <column_alias> | <position> | <expression> }
    [ { ASC | DESC } ] [ NULLS { FIRST | LAST } ]
```

Параметры

- `<column_alias>`
  Псевдоним столбца, заданный в списке `SELECT`.

- `<position>`
  Позиция выражения в списке `SELECT`.

- `<expression>`

Любое выражение, заданное на таблицах в текущей области видимости.

- `{ ASC | DESC }`
  Опционально возвращает значения ключа сортировки в порядке возрастания (от наименьшего к наибольшему) или убывания (от наибольшего к наименьшему).
  По умолчанию: `ASC`

- `NULLS { FIRST | LAST }`
  Опционально указывает, возвращаются ли значения `NULL` до/после значений, отличных от `NULL`, в зависимости от порядка сортировки (`ASC` или `DESC`).
  По умолчанию: зависит от порядка сортировки (`ASC` или `DESC`), см. Модификатор порядка значений `NULL`.

Примеры

- Выведем дни недели, упорядоченные по их названию, используя порядок сортировки по умолчанию и стандартный порядок для значений `NULL`:

```
SELECT *
FROM weekdays
ORDER BY name;
```

```
+--------+-----------+
| number |    name   |
+--------+-----------+
| 5      | Friday    |
+--------+-----------+
| 1      | Monday    |
+--------+-----------+
| 6      | Saturday  |
+--------+-----------+
| 7      | Sunday    |
+--------+-----------+
| 4      | Thursday  |
+--------+-----------+
| 2      | Tuesday   |
+--------+-----------+
| 3      | Wednesday |
+--------+-----------+
| 8      | null      |
+--------+-----------+
```

- Выведем дни недели, упорядоченные по их названию в порядке убывания со значениями `NULL` в начале:

```
SELECT *
FROM weekdays
ORDER BY name DESC NULLS FIRST;
```

```
+--------+-----------+
| number |    name   |
+--------+-----------+
| 8      | null      |
+--------+-----------+
| 3      | Wednesday |
+--------+-----------+
| 2      | Tuesday   |
+--------+-----------+
| 4      | Thursday  |
+--------+-----------+
| 7      | Sunday    |
+--------+-----------+
| 6      | Saturday  |
+--------+-----------+
| 1      | Monday    |
+--------+-----------+
| 5      | Friday    |
+--------+-----------+
```

- Теперь рассмотрим ситуацию, при которой в нашей таблице дни недели пронумерованы начиная с воскресенья (как делается в некоторых календарных системах). Чтобы привести ее к привычному виду, упорядочим дни недели сначала по типу (`weekend` или нет), а затем по номеру:

```
SELECT *
FROM weekdays
ORDER BY weekend, number;
```

```
+--------+-----------+---------+
| number |    name   | weekend |
+--------+-----------+---------+
| 2      | Monday    | false   |
+--------+-----------+---------+
| 3      | Tuesday   | false   |
+--------+-----------+---------+
| 4      | Wednesday | false   |
+--------+-----------+---------+
| 5      | Thursday  | false   |
+--------+-----------+---------+
| 6      | Friday    | false   |
+--------+-----------+---------+
| 1      | Sunday    | true    |
+--------+-----------+---------+
```

```
| 7      | Saturday  | true    |
+--------+-----------+---------+
```

- Покажем разницу в сортировке с использованием разных коллаций. Возьмем названия двух финских городов в их шведском и финском вариантах и упорядочим их по их шведским названиям, используя сначала правила коллации для английского:

```
SELECT *
FROM finnish_cities
ORDER BY swed_name COLLATE EN
```

```
+-------------+-----------+
|  swed_name  |  fin_name |
+-------------+-----------+
| Åbo         | Turku     |
+-------------+-----------+
| Helsingfors | Helsinki  |
+-------------+-----------+
```

Теперь сделаем то же самое, но используем правила коллации для шведского:

```
SELECT *
FROM finnish_cities
ORDER BY swed_name COLLATE SV
```

```
+-------------+-----------+
|  swed_name  |  fin_name |
+-------------+-----------+
| Helsingfors | Helsinki  |
+-------------+-----------+
| Åbo         | Turku     |
+-------------+-----------+
```

В шведском алфавите буква `å` идет в конце, а в английских правилах коллации она идет в начале. Отсюда мы получаем разницу в порядке сортировки.

`LIMIT`

Выражение `LIMIT`

Выражение `LIMIT` является модификатором вывода. Логически оно применяется в самом конце запроса. Выражение `LIMIT` ограничивает количество выводимых строк.

Обратите внимание, что хотя `LIMIT` можно использовать без условия `ORDER BY`, в таком случае результаты могут быть не детерминированными. Тем не менее, это может быть полезно,

например, когда вы хотите получить быстрый срез данных.

Выражение `OFFSET`

Выражение `OFFSET` указывает, с какой позиции начинать считывание значений, т. е. первые `OFFSET` значений игнорируются.

Синтаксис

```
SELECT ...
FROM ...
[ ORDER BY ... ]
LIMIT <count> [ OFFSET <start> ]
[ ... ]
```

Параметры

- `<count>`
  Количество возвращаемых строк. Должно быть неотрицательной целой величиной.
  Значение `NULL` также принимается и рассматривается как неограниченное.

- `OFFSET <start>`
  Номер строки, после которой возвращаются ограниченные/извлеченные строки. Должен быть неотрицательной целой величиной.
  Если `OFFSET` опущено, вывод начинается с первой строки в наборе результатов.
  Значение `NULL` также принимается и рассматривается как неограниченное (т. е. строки пропускаться не будут).

Примеры

- Выберем первые 5 дней из таблицы дней недели:

  ```
  SELECT *
  FROM weekdays
  LIMIT 5;
  ```

  ```
  +--------+-----------+
  | number |    name   |
  +--------+-----------+
  | 1      | Monday    |
  +--------+-----------+
  | 2      | Tuesday   |
  +--------+-----------+
  | 3      | Wednesday |
  ```

```
+--------+-----------+
| 4      | Thursday  |
+--------+-----------+
| 5      | Friday    |
+--------+-----------+
```

- Выберем 5 строк из таблицы дней недели, начиная с позиции 1 (т. е. игнорируя первую строку):

```
SELECT *
FROM weekdays
LIMIT 5
OFFSET 1;
```

```
+--------+-----------+
| number |    name   |
+--------+-----------+
| 2      | Tuesday   |
+--------+-----------+
| 3      | Wednesday |
+--------+-----------+
| 4      | Thursday  |
+--------+-----------+
| 5      | Friday    |
+--------+-----------+
| 6      | Saturday  |
+--------+-----------+
```

`LIKE`

Описание

Выражение с оператором `LIKE` возвращает `TRUE`, если текстовая строка соответствует заданному шаблону.

Если шаблон не содержит знаков процента или подчеркивания, то он интерпретируется буквально, и в этом случае `LIKE` действует как оператор равенства:

```
SELECT
    'Tengri' LIKE 'Tengri' AS result;
```

```
+--------+
| result |
+--------+
| true   |
+--------+
```

Если в шаблоне есть специальные символы, то он интерпретируется не буквально, а как регулярное выражение:

- Подчеркивание `_` в шаблоне соответствует любому отдельному символу.
- Знак процента `%` в шаблоне соответствует любой последовательности из нуля или более символов.

Сопоставление шаблона `LIKE` всегда применяется ко всей строке. Поэтому, если требуется сопоставить последовательность в любом месте строки (накрыть шаблоном подстроку), то шаблон должен начинаться и заканчиваться знаком процента `%`.

Можно также использовать противоположные по значению выражения `<string> NOT LIKE <pattern>` и `NOT string LIKE <pattern>`:

```
SELECT
    NOT 'Tengri' LIKE 'Tengri' AS result_1,
    'Tengri' NOT LIKE 'Tengri' AS result_2;
```

```
+----------+----------+
| result_1 | result_2 |
+----------+----------+
| false    | false    |
+----------+----------+
```

Оператор `ILIKE`

Оператор `ILIKE` можно использовать вместо `LIKE`, чтобы сделать сопоставление нечувствительным к регистру:

```
SELECT
    'Tengri' ILIKE 'tengri' AS result_1,
    'Tengri' ILIKE '%NGRi'  AS result_2;
```

```
+----------+----------+
| result_1 | result_2 |
+----------+----------+
| true     | true     |
+----------+----------+
```

Примеры

- Покажем несколько примеров работы шаблонов `LIKE`:

```
SELECT
    'Tengri' LIKE 'TNGRi'  AS result_1,
```

```
    'Tengri' LIKE 'T%'     AS result_2,
    'TNGRi'  LIKE 'T%'     AS result_3,
    'Tengri' LIKE 'T_____' AS result_4;
```

```
+----------+----------+----------+----------+
| result_1 | result_2 | result_3 | result_4 |
+----------+----------+----------+----------+
| false    | true     | true     | true     |
+----------+----------+----------+----------+
```

- Выберем из таблицы столиц страны, в которых столица начинается на `M`:

```
CREATE TABLE capitals (country VARCHAR, capital VARCHAR);
INSERT INTO capitals VALUES
    ('Russia', 'Moscow'),
    ('Italy', 'Rome'),
    ('Spain', 'Madrid'),
    ('France', 'Paris');

SELECT country FROM capitals WHERE capital LIKE 'M%';
```

```
+---------+
| country |
+---------+
| Russia  |
+---------+
| Spain   |
+---------+
```

Полезные ссылки

- `SIMILAR TO`
- Текстовые функции
- Функции для регулярных выражений

`SIMILAR TO`

Описание

Выражение с оператором `SIMILAR TO` возвращает `TRUE` или `FALSE` в зависимости от того, накладывается ли его шаблон на данную текстовую строку.

Этот оператор похож на оператор `LIKE`, но в данном случае шаблон интерпретируется как регулярное выражение. Как и в случае с `LIKE`, выражение с `SIMILAR TO` проверяется успешно, только если шаблон накладывается на всю строку. Это отличается от обычного поведения регулярных

выражений, где шаблон может накладываться на любую часть строки.

В регулярных выражениях используется синтаксис RE2.

Можно также использовать противоположные по значению выражения `<string> NOT SIMILAR TO <pattern>` и `NOT string SIMILAR TO <pattern>`

Выражения с оператором `SIMILAR TO` полностью синонимичны выражениям с оператором `~`.

Примеры

```
SELECT
    'Tengri' SIMILAR TO 'TNGRi'  AS result_1, -- false expected
    'Tengri' SIMILAR TO 'T.*'    AS result_2,
    'Tengri' SIMILAR TO 'T.+i'   AS result_3,
    'TNGRi'  SIMILAR TO 'T.+i'   AS result_4,
    'T.+i'   SIMILAR TO 'T\.\+i' AS result_5;
```

```
+----------+----------+----------+----------+----------+
| result_1 | result_2 | result_3 | result_4 | result_5 |
+----------+----------+----------+----------+----------+
| false    | true     | true     | true     | true     |
+----------+----------+----------+----------+----------+
```

```
SELECT
    'Tengri' NOT SIMILAR TO 'TNGRi'  AS result_1,
    NOT 'T.{3}i' SIMILAR TO 'T.{3}i' AS result_2;
```

```
+----------+----------+
| result_1 | result_2 |
+----------+----------+
| true     | true     |
+----------+----------+
```

Полезные ссылки

- Оператор `~`
- `LIKE`
- Текстовые функции
- Функции для регулярных выражений

AS

Оператор `AS` в SQL-запросах может использоваться в нескольких разных функциях.

Задание имен для столбцов в запросе

Оператор `AS` можно использовать, чтобы задать имена (псевдонимы) для столбцов в таблице, которая будет выдана в качестве результата запроса или использована в самом запросе:

```
SELECT
    name AS Student
    FROM students
```

Если задаваемое имя содержит пробел, то его нужно заключить в кавычки:

```
SELECT
    name AS "Student Name"
    FROM students
```

Задание имен для таблиц в запросе

Оператор `AS` можно использовать, чтобы задать имена (псевдонимы) для таблиц, которые используются в запросе:

```
SELECT o.OrderID, o.OrderDate, c.CustomerName
    FROM
        Customers AS c,
        Orders AS o
    WHERE
        c.CustomerName="Tengri" AND c.CustomerID=o.CustomerID;
```

Здесь в запросе для краткости использованы псевдонимы `c` и `o` для таблиц `Customers` и `Orders`, к которым делается запрос.

Задание запроса при создании таблицы

При создании таблицы оператор `AS` можно использовать для того, чтобы задать запрос к данным, результат которого будет записан в создаваемую таблицу:

```
CREATE TABLE films_recent AS
    SELECT * FROM films WHERE date_prod >= '2025-01-01';
```

`DELETE`

Выражение `DELETE` удаляет строки из таблицы. Условие для выбора удаляемых строк можно задать через выражение `WHERE`.

В качестве результата запроса возвращается количество удаленных строк.

Вместо удаления или изменения строк в существующих таблицах (выражения `DELETE` и `UPDATE`) рекомендуется создавать новые таблицы на основе существующих с применением необходимых фильтров (выражение `CREATE TABLE`) и при необходимости удалять существующие (выражение `TRUNCATE`).

Синтаксис

```
DELETE FROM <table_name>
    [ WHERE <condition> ]
```

Параметры

- `[ WHERE <condition> ]`

  Указывает условие, которое будет использоваться для выбора строк для удаления. Если этот параметр опущен, все строки в таблице удаляются, но таблица не удаляется.

  Разрешенные операторы внутри условия `<condition>`:

  - `=` равно
  - `>` больше
  - `<` меньше
  - `in (⋯)` входит в список
  - их комбинации

Примеры

Пример 1

Создадим таблицу с числами от `1` до `100` и удалим из нее все строки без дополнительных условий:

```
CREATE TABLE demo.numbers (numbers BIGINT);

INSERT INTO demo.numbers (numbers)
    SELECT unnest(generate_series(1,100));

DELETE FROM demo.numbers;
```

```
+-------+
| count |
+-------+
| 100   |
+-------+
```

Проверим содержимое таблицы:

```
SELECT * FROM demo.numbers;
```

```
+---------+
| numbers |
+---------+
0 rows
```

Пример 2

Создадим таблицу с числами от `1` до `100` и удалим из нее строки с числами меньше `51` или больше `55`:

```
CREATE TABLE demo.numbers (numbers BIGINT);

INSERT INTO demo.numbers (numbers)
    SELECT unnest(generate_series(1,100));

DELETE FROM demo.numbers
    WHERE numbers < 51 OR numbers > 55;
```

```
+-------+
| count |
+-------+
| 95    |
+-------+
```

Проверим содержимое таблицы:

```
SELECT * FROM demo.numbers
```

```
+---------+
| numbers |
+---------+
| 51      |
+---------+
| 52      |
+---------+
| 53      |
+---------+
| 54      |
+---------+
| 55      |
```

```
+---------+
```

Пример 3

Создадим таблицу с числами от `1` до `10` и удалим из нее строки с числами из списка `(2,4,6,8,10,12)`:

```
CREATE TABLE demo.numbers (numbers BIGINT);

INSERT INTO demo.numbers (numbers)
    SELECT unnest(generate_series(1,10));

DELETE FROM demo.numbers
    WHERE numbers IN (2,4,6,8,10,12);
```

```
+-------+
| count |
+-------+
| 5     |
+-------+
```

Проверим содержимое таблицы:

```
SELECT * FROM demo.numbers;
```

```
+---------+
| numbers |
+---------+
| 1       |
+---------+
| 3       |
+---------+
| 5       |
+---------+
| 7       |
+---------+
| 9       |
+---------+
```

`UPDATE`

Выражение `UPDATE` обновляет указанные строки в таблице и записывает в них новые значения. Условие для выбора обновляемых строк можно задать через выражение `WHERE`.

В качестве результата запроса возвращается количество обновленных строк.

Вместо удаления или изменения строк в существующих таблицах (выражения `DELETE` и `UPDATE`) рекомендуется создавать новые таблицы на основе существующих с применением необходимых фильтров (выражение `CREATE TABLE`) и при необходимости удалять существующие (выражение `TRUNCATE`).

Синтаксис

```
UPDATE <table_name>
    SET <col_name> = <value> [ , <col_name> = <value> , ... ]
    [ WHERE <condition> ]
```

Параметры

- `[ SET <col_name> = <value> ]`

  Указывает колонку для обновления и значения, которые необходимо записать. Можно указывать несколько выражений через `,` для любого количества колонок.

- `[ WHERE <condition> ]`

  Указывает условие, которое будет использоваться для выбора строк для обновления.

  Разрешенные операторы внутри условия `<condition>`:

  - `=` равно
  - `>` больше
  - `<` меньше
  - `in (…)` входит в список
  - их комбинации

Примеры

Пример 4

Создадим таблицу с числами от `1` до `10` и обновим колонку `numbers`, записав в нее значения из нее самой, умноженные на `100`:

```
CREATE TABLE demo.numbers (numbers BIGINT);

INSERT INTO demo.numbers (numbers)
    SELECT unnest(generate_series(1,10));

UPDATE demo.numbers
    SET numbers = numbers*100;
```

```
+-------+
| Count |
+-------+
| 10    |
+-------+
```

Проверим содержимое таблицы:

```
SELECT * FROM demo.numbers;
```

```
+---------+
| numbers |
+---------+
| 100     |
+---------+
| 200     |
+---------+
| 300     |
+---------+
| 400     |
+---------+
| 500     |
+---------+
| 600     |
+---------+
| 700     |
+---------+
| 800     |
+---------+
| 900     |
+---------+
| 1000    |
+---------+
```

Пример 5

Создадим таблицу с двумя колонками `num_1` и `num_2`. Запишем в колонку `num_1` числа от `1` до `10`, а колонку `num_2` оставим пустой (там будут значения `NULL`). Обновим колонки `num_1` и `num_2`: запишем в `num_1` значения из нее самой, умноженные на `100`, а в `num_2` — значения из `num_1`, умноженные на `-1`. Сделаем все это с условием, что значения из `num_1` входят в список `(2,4,6,8,10,12)`:

```
CREATE TABLE demo.numbers (num_1 BIGINT, num_2 BIGINT);

INSERT INTO demo.numbers (num_1)
    SELECT unnest(generate_series(1,10));

UPDATE demo.numbers
```

```
    SET num_1 = num_1*100, num_2 = num_1*-1
    WHERE num_1 in (2,4,6,8,10,12);
```

```
+-------+
| Count |
+-------+
| 5     |
+-------+
```

Проверим содержимое таблицы:

```
SELECT * FROM demo.numbers;
```

```
+-------+-------+
| num_1 | num_2 |
+-------+-------+
| 200   | -2    |
+-------+-------+
| 400   | -4    |
+-------+-------+
| 600   | -6    |
+-------+-------+
| 800   | -8    |
+-------+-------+
| 1000  | -10   |
+-------+-------+
| 1     | null  |
+-------+-------+
| 3     | null  |
+-------+-------+
| 5     | null  |
+-------+-------+
| 7     | null  |
+-------+-------+
| 9     | null  |
+-------+-------+
```

Функции

Агрегатные функции

- `any_value`
- `array_agg`
- `avg`
- `count`

- `count(argument)`
- `count_if`
- `max`
- `min`
- `median`
- `sum`

Числовые функции

- `abs`
- `add`
- `ceil`
- `cos`
- `divide`
- `even`
- `exp`
- `floor`
- `fmod`
- `gcd`
- `greatest`
- `isfinite`
- `isinf`
- `isnan`
- `lcm`
- `least`
- `lgamma`
- `ln`
- `log`
- `log2`
- `multiply`
- `nextafter`
- `pi`
- `pow`
- `radians`
- `random`
- `round_even`
- `round`

- `setseed`
- `sign`
- `signbit`
- `sin`
- `sqrt`
- `subtract`
- `trunc`
- Оператор `+`
- Оператор `-`
- Оператор `*`
- Оператор `/`
- Оператор `%`
- Оператор `^`

Текстовые функции

- `contains`
- `length`
- `strlen`
- `trim`
- `ltrim`
- `rtrim`
- `lower`
- `upper`
- `replace`
- `split`
- `substring`
- `chr`

Функции для регулярных выражений

- `regexp_extract`
- `regexp_extract_all`
- `regexp_full_match`
- `regexp_matches`
- `regexp_replace`
- `regexp_split_to_array`

- `regexp_split_to_table`
- Оператор `~`

Функции для даты и времени

- `current_time`
- `now`
- `datepart`
- `date_diff`
- `date_trunc`
- `strptime`

Функции для JSON

- `json_array_length`
- `json_contains`
- `json_exists`
- `json_extract`
- `json_extract_string`
- `json_group_array`
- `json_group_object`
- `json_keys`
- `json_transform`
- `json_transform_strict`
- `json_valid`
- `json_value`
- `json`
- `read_json`

Функции для двоичных данных

- `concat`
- `md5`
- `sha1`
- `sha256`
- Оператор `||`

Функции для геоданных

- `ST_Point`
- `ST_Distance`
- `ST_Distance_Sphere`

Оконные функции

- `cume_dist`
- `dense_rank`
- `first_value`
- `lag`
- `last_value`
- `lead`
- `nth_value`
- `ntile`
- `percent_rank`
- `rank`
- `row_number`

Утилиты

- `coalesce`
- `generate_series`
- `hash`
- `read_csv`
- `read_parquet`
- `read_xlsx`
- `cast`
- `try_cast`
- `unnest`
- Оператор `::`

Функции модуля Python `tngri`

- `tngri.run_notebook`
- `tngri.sql`
- `tngri.upload_df`

- `tngri.upload_file`
- `tngri.upload_s3`

Агрегатные функции

Агрегатные функции — это функции, которые объединяют значения из нескольких строк в одно.

Агрегатные функции отличаются от скалярных функций и оконных функций тем, что они изменяют кардинальность результата. Поэтому в запросах их можно использовать только в выражениях `SELECT` и `HAVING`.

Выражение `DISTINCT` в агрегатных функциях

Когда используется выражение `DISTINCT`, при вычислении значения агрегатной функции учитываются только уникальные значения. Часто это выражение используется в сочетании с агрегатной функцией `count()` для получения количества уникальных элементов, но может использоваться и с другими агрегатными функциями.

Пример

```
CREATE TABLE cities(city_name VARCHAR);

INSERT INTO cities VALUES
    ('Moscow'),
    ('Moscow'),
    ('Paris'),
    ('Madrid');

SELECT
    count(DISTINCT city_name) AS distinct_cities_num,
    count(city_name) AS cities_num
FROM cities;
```

```
+---------------------+------------+
| distinct_cities_num | cities_num |
+---------------------+------------+
| 3                   | 4          |
+---------------------+------------+
```

Некоторые агрегатные функции нечувствительны к повторяющимся значениям (например, `min()` и `max()`), и для них выражение `DISTINCT` игнорируется.

`any_value()`

Описание Возвращает первое значение из `argument`, отличное от `NULL`.

Использование `any_value(argument)`

На работу этой функции влияет порядок сортировки в таблице.

▼ *Посмотреть пример*

```
CREATE TABLE numbers(number_asc BIGINT, number_desc BIGINT);

INSERT INTO numbers VALUES
    (NULL,  3),
    (1,     2),
    (2,     1),
    (3,     NULL);

SELECT
    any_value(number_asc) AS "any value from number asc",
    any_value(number_desc) AS "any value from number desc"
FROM numbers;
```

```
+--------------------------+---------------------------+
| any value from number asc | any value from number desc |
+--------------------------+---------------------------+
| 1                        | 3                         |
+--------------------------+---------------------------+
```

`array_agg()`

Описание Возвращает список, содержащий все значения столбца.

Использование `array_agg(argument)`

Псевдонимы `list()`

На работу этой функции влияет порядок сортировки в таблице.

▼ *Посмотреть пример*

```
CREATE TABLE numbers(number BIGINT);

INSERT INTO numbers VALUES
    (1),
    (2),
    (3),
    (NULL);

SELECT
```

```
    array_agg(number) AS array_agg,
    list(number) AS list
FROM numbers;
```

```
+--------------+--------------+
|   array_agg  |     list     |
+--------------+--------------+
| {1,2,3,None} | {1,2,3,None} |
+--------------+--------------+
```

avg()

Описание Вычисляет среднее значение всех непустых значений в `argument`.

Использование `avg(argument)`

Псевдонимы `mean()`

▼ *Посмотреть пример*

```
CREATE TABLE numbers(number BIGINT);

INSERT INTO numbers VALUES
    (1),
    (2),
    (3),
    (NULL);

SELECT
    avg(number) AS average,
    mean(number) AS mean
FROM numbers;
```

```
+---------+------+
| average | mean |
+---------+------+
| 2       | 2    |
+---------+------+
```

count()

Описание Вычисляет количество строк в группе.

Использование `count()`

Псевдонимы `count(*)`

▼ *Посмотреть пример*

```
CREATE TABLE numbers(number BIGINT);

INSERT INTO numbers VALUES
    (1),
    (2),
    (3),
    (NULL);

SELECT
    count() AS rows_count,
    count(*) AS rows_count_star
FROM numbers;
```

```
+------------+-----------------+
| rows_count | rows_count_star |
+------------+-----------------+
| 4          | 4               |
+------------+-----------------+
```

`count(argument)`

Описание Вычисляет количество непустых значений в `argument`.

Использование `count(argument)`

▼ *Посмотреть пример*

```
CREATE TABLE numbers(number BIGINT);

INSERT INTO numbers VALUES
    (1),
    (2),
    (3),
    (NULL);

SELECT
    count() AS rows_count,
    count(number) AS values_count
FROM numbers;
```

```
+------------+--------------+
| rows_count | values_count |
+------------+--------------+
| 4          | 3            |
+------------+--------------+
```

`count_if()`

Описание Возвращает количество записей, удовлетворяющих условию, или `NULL`, если ни одна запись не удовлетворяет условию.

Использование `count_if(<condition>)`

▼ *Посмотреть пример*

```
CREATE TABLE text_table(text_data VARCHAR);

INSERT INTO text_table VALUES
('Tengri'),
('Tengri'),
('TNGRi'),
(NULL);

SELECT
    COUNT_IF(TRUE) AS row_number,
    COUNT_IF(text_data = 'Tengri') AS tengri_number
FROM text_table;
```

```
+------------+---------------+
| row_number | tengri_number |
+------------+---------------+
| 4          | 2             |
+------------+---------------+
```

`max()`

Описание Возвращает максимальное значение, имеющееся в `argument`.

Использование `max(argument)`

▼ *Посмотреть пример*

```
CREATE TABLE numbers(number BIGINT);

INSERT INTO numbers VALUES
    (1),
    (2),
    (3),
    (NULL);

SELECT
    max(number) AS max,
    min(number) AS min
FROM numbers;
```

```
+-----+-----+
```

```
| max | min |
+-----+-----+
| 3   | 1   |
+-----+-----+
```

`min()`

Описание Возвращает минимальное значение, имеющееся в `argument`.

Использование `min(argument)`

▼ *Посмотреть пример*

```
CREATE TABLE numbers(number BIGINT);

INSERT INTO numbers VALUES
    (1),
    (2),
    (3),
    (NULL);

SELECT
    max(number) AS max,
    min(number) AS min
FROM numbers;
```

```
+-----+-----+
| max | min |
+-----+-----+
| 3   | 1   |
+-----+-----+
```

`median()`

Описание Возвращает медианное значение всех непустых значений в `argument`.

Использование `median(argument)`

В случае четного количества значений берется среднее между двумя центральными значениями.

▼ *Посмотреть пример*

Покажем разницу между медианным значением и средним значением (`avg()`) на примерах наборов из четного (`_even`) и нечетного (`_odd`) количества чисел, содержащих в том числе пустые значения.

```
CREATE TABLE numbers(
    number_even BIGINT,
    number_odd BIGINT);
```

```
INSERT INTO numbers VALUES
    (1,    1),
    (2,    2),
    (3,    10),
    (10,   NULL),
    (NULL, NULL);

SELECT
    median(number_even) AS median_even,
    avg(number_even) AS avg_even,
    median(number_odd) AS median_odd,
    avg(number_odd) AS avg_odd
FROM numbers;
```

```
+-------------+----------+------------+-------------------+
| median_even | avg_even | median_odd |      avg_odd      |
+-------------+----------+------------+-------------------+
| 2.5         | 4        | 2          | 4.333333333333333 |
+-------------+----------+------------+-------------------+
```

`sum()`

Описание Вычисляет сумму всех непустых значений в `argument`.

Использование `sum(argument)`

В случае булевых значений подсчитывает количество значений `True`.

▼ *Посмотреть пример*

```
CREATE TABLE numbers(
    number BIGINT,
    boolean BOOL);

INSERT INTO numbers VALUES
    (1,    True),
    (2,    False),
    (3,    False),
    (NULL, NULL);

SELECT
    sum(number) AS sum_number,
    sum(boolean) AS sum_boolean
FROM numbers;
```

```
+------------+-------------+
| sum_number | sum_boolean |
+------------+-------------+
| 6          | 1           |
```

```
+-----------+------------+
```

Числовые функции

Числовые функции — это функции для работы с данными числовых типов: `BIGINT`, `NUMERIC` и `DOUBLE`.

`abs()`

Описание Вычисляет модуль числа.

Использование `abs(num)`

▼ *Посмотреть примеры*

```
SELECT
    abs(-1)  AS result_1,
    abs(0)   AS result_2,
    abs(1.1) AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 1        | 0        | 1.1      |
+----------+----------+----------+
```

`add()`

Описание Складывает числа.

Использование `add(num, num)`

См. также Оператор `+`.

▼ *Посмотреть примеры*

```
SELECT
    add(1, 1)      AS result_1,
    add(-1.1, 2.1) AS result_2,
    add(1)         AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 2        | 1.0      | 1        |
+----------+----------+----------+
```

`ceil()`

Описание Округляет число в бОльшую сторону.

Использование `ceil(num)`

Псевдонимы `ceiling()`

▼ *Посмотреть примеры*

```
SELECT
    ceil(0.1)  AS result_1,
    ceil(-0.1) AS result_2,
    ceiling(1) AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 1        | 0        | 1        |
+----------+----------+----------+
```

`cos()`

Описание Вычисляет косинус угла, заданного в радианах.

Использование `cos(num)`

▼ *Посмотреть примеры*

```
SELECT
    cos(0)      AS result_1,
    cos(pi())   AS result_2,
    cos(pi()/3) AS result_3;
```

```
+----------+----------+--------------------+
| result_1 | result_2 | result_3           |
+----------+----------+--------------------+
| 1        | -1       | 0.5000000000000001 |
+----------+----------+--------------------+
```

`divide()`

Описание Возвращает результат деления в виде целого числа.

Использование `divide(num, num)`

▼ *Посмотреть примеры*

```
SELECT
```

```
    divide(7, 2)  AS result_1,
    divide(7, -2) AS result_2,
    divide(7, 0)  AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 3        | -3       | null     |
+----------+----------+----------+
```

even()

Описание Округляет до ближайшего четного числа в сторону от нуля.

Использование `even(num)`

▼ *Посмотреть примеры*

```
SELECT
    even(2.1)  AS result_1,
    even(-2.1) AS result_2,
    even(0)    AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 4        | -4       | 0        |
+----------+----------+----------+
```

exp()

Описание Вычисляет экспоненту числа.

Использование `exp(num)`

Вычисляет экспоненциальное значение числа: `e^x`.

▼ *Посмотреть примеры*

```
SELECT
    exp(0)  AS result_1,
    exp(1)  AS result_2,
    exp(-1) AS result_3;
```

```
+----------+-------------------+---------------------+
| result_1 | result_2          | result_3            |
+----------+-------------------+---------------------+
| 1        | 2.718281828459045 | 0.36787944117144233 |
+----------+-------------------+---------------------+
```

floor()

Описание Округляет число в меньшую сторону.

Использование `floor(num)`

▼ *Посмотреть примеры*

```
SELECT
    floor(0.9)  AS result_1,
    floor(-0.9) AS result_2,
    floor(1)    AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 0        | -1       | 1        |
+----------+----------+----------+
```

fmod()

Описание Возвращает остаток от деления первого аргумента на второй.

Использование `fmod(num, num)`

▼ *Посмотреть примеры*

```
SELECT
    fmod(3, 2)   AS result_1,
    fmod(3.1, 2) AS result_2,
    fmod(-10, 4) AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 1        | 1.1      | 2        |
+----------+----------+----------+
```

gcd()

Описание Вычисляет наибольший общий делитель двух чисел.

Использование `gcd(num, num)`

Псевдонимы `greatest_common_divisor()`

▼ *Посмотреть примеры*

```
SELECT
    gcd(12, 9)  AS result_1,
    gcd(-12, 9) AS result_2,
    gcd(12, 0)  AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 3        | 3        | 12       |
+----------+----------+----------+
```

`greatest()`

Описание Возвращает наибольшее число из указанных в аргументах.

Использование `greatest(num[, num, ...])`

▼ *Посмотреть примеры*

```
SELECT
    greatest(1, 2, 3, 4, 4) AS result_1,
    greatest(1, -1)         AS result_2,
    greatest(0)             AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 4        | 1        | 0        |
+----------+----------+----------+
```

`isfinite()`

Описание Проверяет, является ли число конечным.

Использование `isfinite(num)`

▼ *Посмотреть примеры*

```
SELECT
    isfinite(1)                  AS result_1,
    isfinite('Infinity'::DOUBLE) AS result_2,
    isfinite(NULL)               AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| true     | false    | null     |
+----------+----------+----------+
```

`isinf()`

Описание Проверяет, является ли число бесконечным.

Использование `isinf(num)`

▼ *Посмотреть примеры*

```
SELECT
    isinf(1)                  AS result_1,
    isinf('Infinity'::DOUBLE) AS result_2,
    isinf(NULL)               AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| false    | true     | null     |
+----------+----------+----------+
```

`isnan()`

Описание Проверяет, имеет ли аргумент значение `NaN` (*Not a Number*).

Использование `isnan(num)`

▼ *Посмотреть примеры*

```
SELECT
    isnan('NaN'::DOUBLE) AS result_1,
    isnan(1.1)           AS result_2,
    isnan(NULL)          AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| true     | false    | null     |
+----------+----------+----------+
```

`lcm()`

Описание Вычисляет наименьшее общее кратное двух чисел.

Использование `lcm(num, num)`

Псевдонимы `least_common_multiple()`

▼ *Посмотреть примеры*

```
SELECT
    lcm(3, 7)     AS result_1,
    lcm(333, 777) AS result_2,
    lcm(37, 0)    AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 21       | 2331     | 0        |
+----------+----------+----------+
```

`least()`

Описание Возвращает наименьшее число из указанных в аргументах.

Использование `least(num[, num, ...])`

▼ *Посмотреть примеры*

```
SELECT
    least(1, 1, 2, 3, 4) AS result_1,
    least(1, -1)         AS result_2,
    least(0)             AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 1        | -1       | 0        |
+----------+----------+----------+
```

`lgamma()`

Описание Вычисляет логарифм гамма-функции.

Использование `lgamma(num)`

▼ *Посмотреть примеры*

```
SELECT
    lgamma(1)   AS result_1,
    lgamma(11)  AS result_2,
    lgamma(1.1) AS result_3;
```

```
+----------+--------------------+-----------------------+
| result_1 | result_2           | result_3              |
+----------+--------------------+-----------------------+
| 0        | 15.104412573075518 | -0.049872441259839764 |
+----------+--------------------+-----------------------+
```

ln()

Описание Вычисляет натуральный логарифм числа.

Использование `ln(num)`

▼ *Посмотреть примеры*

```
SELECT
    ln(1)   AS result_1,
    ln(11)  AS result_2,
    ln(1.1) AS result_3;
```

```
+----------+--------------------+----------------------+
| result_1 | result_2           | result_3             |
+----------+--------------------+----------------------+
| 0        | 2.3978952727983707 | 0.09531017980432493 |
+----------+--------------------+----------------------+
```

log()

Описание Вычисляет логарифм числа по основанию `10`.

Использование `log(num)`

Псевдонимы `log10()`

▼ *Посмотреть примеры*

```
SELECT
    log(1)    AS result_1,
    log(100)  AS result_2,
    log(0.01) AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 0        | 2        | -2       |
+----------+----------+----------+
```

`log2()`

Описание Вычисляет логарифм числа по основанию `2`.

Использование `log2(num)`

▼ *Посмотреть примеры*

```
SELECT
    log2(1)    AS result_1,
    log2(2)    AS result_2,
    log2(4096) AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 0        | 1        | 12       |
+----------+----------+----------+
```

`multiply()`

Описание Перемножает два числа.

Использование `multiply(num, num)`

См. также Оператор `*`.

▼ *Посмотреть примеры*

```
SELECT
    multiply(2, 2)      AS result_1,
    multiply(0, 2)      AS result_2,
    multiply(0.2, -0.2) AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 4        | 0        | -0.04    |
+----------+----------+----------+
```

`nextafter()`

Описание Возвращает следующее значение с переменной точностью (типа `DOUBLE`) после первого числа в направлении второго числа.

Использование `nextafter(num, num)`

▼ *Посмотреть примеры*

```
SELECT
    nextafter(1::DOUBLE, 2)  AS result_1,
    nextafter(1::BIGINT, 2)  AS result_2,
    nextafter(-1::BIGINT, 0) AS result_3;
```

```
+-------------------+-------------------+--------------------+
|      result_1     |      result_2     |       result_3     |
+-------------------+-------------------+--------------------+
| 1.0000000000000002 | 1.0000000000000002 | -0.9999999999999999 |
+-------------------+-------------------+--------------------+
```

`pi()`

Описание Возвращает значение числа π.

Использование `pi()`

▼ *Посмотреть примеры*

```
SELECT
    pi()   AS result_1,
    pi()/2 AS result_2,
    2*pi() AS result_3;
```

```
+------------------+-------------------+------------------+
|      result_1    |      result_2     |      result_3    |
+------------------+-------------------+------------------+
| 3.141592653589793 | 1.5707963267948966 | 6.283185307179586 |
+------------------+-------------------+------------------+
```

`pow()`

Описание Возводит первый аргумент в степень, заданную вторым аргументом.

Использование `pow(num, num)`

Псевдонимы `power()`

См. также Оператор `^`.

▼ *Посмотреть примеры*

```
SELECT
    pow(2, 5)   AS result_1,
    pow(25, -1) AS result_2,
    pow(25, 0)  AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 32       | 0.04     | 1        |
+----------+----------+----------+
```

`radians()`

Описание Переводит градусы в радианы.

Использование `radians(num)`

▼ *Посмотреть примеры*

```
SELECT
    radians(0)         AS result_1,
    radians(180)       AS result_2,
    radians(-180/pi()) AS result_3;
```

```
+----------+-------------------+----------+
| result_1 |      result_2     | result_3 |
+----------+-------------------+----------+
| 0        | 3.141592653589793 | -1       |
+----------+-------------------+----------+
```

`random()`

Описание Возвращает произвольное число (типа `DOUBLE`) в интервале от `0` до `1`.

Использование `random()`

См. также `setseed()`.

▼ *Посмотреть примеры*

```
SELECT
    random() AS result;
```

```
+--------------------+
|       result       |
+--------------------+
| 0.5656213557274057 |
+--------------------+
```

Функцию `random()` удобно использовать для выбора из таблицы произвольного сэмпла заданной длины.

Покажем на примере, как выбрать из таблицы с 1 миллионом строк случайный сэмпл размером 10:

```
CREATE TABLE demo.numbers (numbers BIGINT);

INSERT INTO demo.numbers (numbers)
    SELECT unnest(generate_series(1,1000000));

SELECT * FROM demo.numbers
ORDER BY random()
LIMIT 10;
```

```
+---------+
| numbers |
+---------+
| 265453  |
+---------+
| 16766   |
+---------+
| 861234  |
+---------+
| 455059  |
+---------+
| 898869  |
+---------+
| 454774  |
+---------+
| 465029  |
+---------+
| 885538  |
+---------+
| 465844  |
+---------+
| 50905   |
+---------+
```

`round_even()`

Описание	Округляет число из первого аргумента до ближайшего четного с указанной во втором аргументе точностью.
Использование	`round_even(num, num)`
Псевдонимы	`roundbankers()`

Второй аргумент указывает на количество десятичных знаков точности и может быть отрицательным числом.

Подробнее об округлении до ближайшего четного числа см. здесь.

▼ *Посмотреть примеры*

```
SELECT
    round_even(4.5, 0)    AS result_1,
    round_even(3.5, 0)    AS result_2,
    round_even(-4.5, 0)   AS result_3,
    round_even(-3.5, 0)   AS result_4,
    round_even(4.45, 1)   AS result_5,
    round_even(4.35, 1)   AS result_6,
    round_even(35.35, -1) AS result_7;
```

```
+----------+----------+----------+----------+----------+----------+----------+
| result_1 | result_2 | result_3 | result_4 | result_5 | result_6 | result_7 |
+----------+----------+----------+----------+----------+----------+----------+
| 4        | 4        | -4       | -4       | 4.4      | 4.4      | 40       |
+----------+----------+----------+----------+----------+----------+----------+
```

`round()`

Описание Округляет число из первого аргумента с указанной во втором аргументе точностью.

Использование `round(num, num)`

Второй аргумент указывает на количество десятичных знаков точности и может быть отрицательным числом.

▼ *Посмотреть примеры*

```
SELECT
    round(4.5, 0)   AS result_1,
    round(4.45, 1)  AS result_2,
    round(44.5, -1) AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 5        | 4.5      | 40       |
+----------+----------+----------+
```

`setseed()`

Описание Фиксирует начальное значение для функции `random()`.

Использование `setseed(num)`

См. также `random()`.

▼ *Посмотреть примеры*

```
SELECT
    setseed(0.5) AS seed,
    random() AS random;
```

```
+------+-------------------+
| seed | random            |
+------+-------------------+
| null | 0.8511131886287325 |
+------+-------------------+
```

`sign()`

Описание Возвращает `-1`, `1` или `0` в зависимости от знака аргумента.

Использование `sign(num)`

▼ *Посмотреть примеры*

```
SELECT
    sign(10)  AS result_1,
    sign(-10) AS result_2,
    sign(0)   AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 1        | -1       | 0        |
+----------+----------+----------+
```

`signbit()`

Описание Определяет, установлен ли бит знака у вещественного числа.

Использование `signbit(num)`

▼ *Посмотреть примеры*

```
SELECT
    signbit(-1)                  AS result_1,
    signbit(-'Infinity'::DOUBLE) AS result_2,
    signbit(0)                   AS result_3,
    signbit(1)                   AS result_4,
    signbit('Infinity'::DOUBLE)  AS result_5;
```

```
+----------+----------+----------+----------+----------+
| result_1 | result_2 | result_3 | result_4 | result_5 |
+----------+----------+----------+----------+----------+
```

```
| true     | true     | false    | false    | false    |
+----------+----------+----------+----------+----------+
```

sin()

Описание Вычисляет синус угла, заданного в радианах.

Использование `sin(num)`

▼ *Посмотреть примеры*

```
SELECT
    sin(0)          AS result_1,
    sin(pi()/2)     AS result_2,
    sin((3*pi())/2) AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 0        | 1        | -1       |
+----------+----------+----------+
```

sqrt()

Описание Вычисляет квадратный корень.

Использование `sqrt(num)`

Число `num` должно быть неотрицательным.

▼ *Посмотреть примеры*

```
SELECT
    sqrt(4)   AS result_1,
    sqrt(144) AS result_2,
    sqrt(0)   AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 2        | 12       | 0        |
+----------+----------+----------+
```

subtract()

Описание Вычитает второй аргумент из первого.

Использование `subtract(num, num)`

См. также Оператор `-`.

▼ *Посмотреть примеры*

```
SELECT
    subtract(1, 2)     AS result_1,
    subtract(1.1, 2.2) AS result_2,
    subtract(-1, -2)   AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| -1       | -1.1     | 1        |
+----------+----------+----------+
```

`trunc()`

Описание Отбрасывает все знаки после десятичного разделителя.

Использование `trunc(num)`

Не следует путать с округлением `round`.

▼ *Посмотреть примеры*

```
SELECT
    trunc(1.99)  AS result_1,
    trunc(-11.9) AS result_2,
    trunc(0.119) AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 1        | -11      | 0        |
+----------+----------+----------+
```

Оператор `+`

Описание Прибавляет правый аргумент к левому.

Использование `<num> + <num> [+ ...]` или `TIME + INTERVAL [+ ...]`

Если используется с типами для для даты и времени, то прибавляет интервал к значению времени. Возвращает значение типа `TIME`.

См. также `add()`.

▼ *Посмотреть примеры*

```
SELECT
    3 + 2      AS result_1,
    3 + 2 + -1 AS result_2,
    1.1 + 1.9  AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 5        | 4        | 3.0      |
+----------+----------+----------+
```

```
SELECT
    TIME '12:11:10'     + INTERVAL 3 hours AS result_time_1,
    INTERVAL '12:11:10' + TIME '1:1:1'     AS result_time_2;
```

```
+---------------+---------------+
| result_time_1 | result_time_2 |
+---------------+---------------+
| 15:11:10      | 13:12:11      |
+---------------+---------------+
```

Оператор `-`

Описание Вычитает правый аргумент из левого.

Использование `<num> - <num> [- ...]` или `TIME - INTERVAL [- ...]`

Если используется с типами для для даты и времени, то вычитает интервал из значения времени. Возвращает значение типа `TIME`.

См. также `subtract()`.

▼ *Посмотреть примеры*

```
SELECT
    3 - 2      AS result_1,
    3 - 2 - +1 AS result_2,
    1.2 - 0.2  AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 1        | 0        | 1.0      |
+----------+----------+----------+
```

```
SELECT
    TIME '12:11:10' - INTERVAL 3 HOUR AS result_time_1,
    TIME '12:11:10' - INTERVAL 3 HOUR - INTERVAL 1 HOUR AS result_time_2;
```

```
+---------------+---------------+
| result_time_1 | result_time_2 |
+---------------+---------------+
| 09:11:10      | 08:11:10      |
+---------------+---------------+
```

Оператор *

Описание Перемножает аргументы.

Использование `<num> * <num>[ * <num>, ...]`

См. также `multiply()`.

▼ *Посмотреть примеры*

```
SELECT
    3*2     AS result_1,
    3*+2*-2 AS result_2,
    3*0     AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 6        | -12      | 0        |
+----------+----------+----------+
```

Оператор /

Описание Делит левый аргумент на правый.

Использование `<num> / <num>[ / <num>, ...]`

Возвращает результат в виде числа с переменной точностью (типа `DOUBLE`).

▼ *Посмотреть примеры*

```
SELECT
    3/2     AS result_1,
    3/+2/-2 AS result_2,
    3/1     AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 1.5      | -0.75    | 3        |
+----------+----------+----------+
```

Оператор `%`

Описание Возвращает остаток от деления левого аргумента на правый.

Использование `<num> % <num>[ % <num>, ...]`

▼ *Посмотреть примеры*

```
SELECT
    3 % 2       AS result_1,
    15 % 10 % 3 AS result_2,
    5 % 2.4     AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 1        | 2        | 0.2      |
+----------+----------+----------+
```

Оператор `^`

Описание Возводит левый аргумент в степень, заданную правым аргументом.

Использование `<num> ^ <num>[ ^ <num>, ...]`

См. также `pow()`.

▼ *Посмотреть примеры*

```
SELECT
    2^3   AS result_1,
    2^3^2 AS result_2,
    1^0   AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 8        | 64       | 1        |
+----------+----------+----------+
```

Текстовые функции

Текстовые функции — это функции для работы с текстовыми строками (данными типа `VARCHAR`).

`concat()`

Описание Конкатенирует несколько строк, массивов или двоичных значений.

Использование `concat(argument1, argument2, ...)`

Пустые значения (`NULL`) игнорируются.

См. также Оператор `||`.

▼ *Посмотреть пример*

```
SELECT
    concat('\xAA'::BLOB, '\xff'::BLOB)          AS result_blob,
    concat('I', ' ', 'love', ' ', 'Tengri')     AS result_string,
    concat(['T', 'e'], ['n', 'g', 'r', 'i'])    AS result_array;
```

```
+-------------+---------------+---------------+
| result_blob | result_string |  result_array |
+-------------+---------------+---------------+
| \xAA\xFF    | I love Tengri | {T,e,n,g,r,i} |
+-------------+---------------+---------------+
```

`contains()`

Описание Возвращает `true`, если в указанной строке `string` содержится искомая подстрока `search_string`.

Использование `contains(string, search_string)`

▼ *Посмотреть пример*

```
SELECT
    contains('I love Tengri', 'Tengri') AS check_name,
    contains('I love Tengri', 'TNGRi')  AS check_nickname;
```

```
+------------+----------------+
| check_name | check_nickname |
+------------+----------------+
| true       | false          |
+------------+----------------+
```

`length()`

Описание Возвращает количество символов в строке.

Использование `length(string)`

Псевдонимы `char_length()`, `character_length()`

▼ *Посмотреть пример*

```
SELECT
    length('I love Tengri ') AS length;
```

```
+--------+
| length |
+--------+
| 15     |
+--------+
```

`strlen()`

Описание Возвращает количество байтов в строке.

Использование `strlen(string)`

▼ *Посмотреть пример*

```
SELECT
    strlen('Tengri ') AS strlen;
```

```
+--------+
| strlen |
+--------+
| 11     |
+--------+
```

`trim()`

Описание Удаляет все вхождения любого из указанных символов с обеих сторон строки.

Использование `trim(string[, characters])`

Если подлежащие удалению символы не указаны, то по умолчанию таким символом является пробел.

▼ *Посмотреть примеры*

```
SELECT
```

```
    '"' || trim('  Tengri ') || '"' AS trim;
```

```
+----------+
|   trim   |
+----------+
| "Tengri" |
+----------+
```

```
SELECT
    trim('[Tengri]', '{([])}') AS trim_brackets;
```

```
+---------------+
| trim_brackets |
+---------------+
| Tengri        |
+---------------+
```

`ltrim()`

Описание Удаляет все вхождения любого из указанных символов в начале строки.

Использование `ltrim(string[, characters])`

Если подлежащие удалению символы не указаны, то по умолчанию таким символом является пробел.

▼ *Посмотреть примеры*

```
SELECT
    '"' || ltrim(' Tengri ') || '"' AS ltrim;
```

```
+-----------+
|   ltrim   |
+-----------+
| "Tengri " |
+-----------+
```

```
SELECT
    ltrim('{{([Tengri])}}', '{([])}') AS ltrim_brackets;
```

```
+----------------+
| ltrim_brackets |
+----------------+
| Tengri])}}     |
```

```
+----------------+
```

`rtrim()`

Описание Удаляет все вхождения любого из указанных символов в конце строки.

Использование `rtrim(string[, characters])`

Если подлежащие удалению символы не указаны, то по умолчанию таким символом является пробел.

▼ *Посмотреть примеры*

```
SELECT
    '"' || rtrim(' Tengri ') || '"' AS ltrim;
```

```
+-----------+
|   ltrim   |
+-----------+
| " Tengri" |
+-----------+
```

```
SELECT
    rtrim('{{{[Tengri])}}}', '{([])}') AS rtrim_brackets;
```

```
+----------------+
| rtrim_brackets |
+----------------+
| {{{[Tengri     |
+----------------+
```

`lower()`

Описание Преобразует строку в нижний регистр.

Использование `lower(string)`

Псевдонимы `lcase()`

▼ *Посмотреть пример*

```
SELECT
    lower('TNGRi') AS lower;
```

```
+-------+
| lower |
```

```
+-------+
| tngri |
+-------+
```

`upper()`

Описание Преобразует строку в верхний регистр.

Использование `upper(string)`

Псевдонимы `ucase()`

▼ *Посмотреть пример*

```
SELECT
    upper('Tengri') AS upper;
```

```
+--------+
|  upper |
+--------+
| TENGRI |
+--------+
```

`replace()`

Описание Заменяет все подстроки на указанную строку.

Использование `replace(string, source, target)`

Заменяет в строке `string` все подстроки `source` на `target`.

▼ *Посмотреть пример*

```
SELECT
    replace('Tengri', 'engr', 'NGR')       AS result_1,
    replace('my name is Tengri', 'n', 'N') AS result_2;
```

```
+----------+-------------------+
| result_1 | result_2          |
+----------+-------------------+
| TNGRi    | my Name is TeNgri |
+----------+-------------------+
```

`split()`

Описание Делит строку на две части по заданному сепаратору.

Использование `split(string, separator)`

Псевдонимы `str_split`, `string_split`, `string_to_array`

▼ *Посмотреть пример*

```
SELECT
    split('I love Tengri', ' ') AS words;
```

```
+-----------------+
|      words      |
+-----------------+
| {I,love,Tengri} |
+-----------------+
```

`substring()`

Описание Извлекает подстроку из строки.

Использование `substring(string, start[, length])`

Псевдонимы `substr`

Извлекает подстроку из указанной строки `string`, начиная с символа `start` и до конца строки. Если задан дополнительный аргумент `length`, извлекается подстрока длиной в `length` символов. Обратите внимание, что нумерация символов в исходной строке начинается с `1`.

▼ *Посмотреть пример*

```
SELECT
    substring('I love Tengri', 1)       AS substring_1,
    substring('I love Tengri', 3)       AS substring_2,
    substring('I love Tengri', 3, 4)    AS substring_3;
```

```
+---------------+-------------+-------------+
| substring_1   | substring_2 | substring_3 |
+---------------+-------------+-------------+
| I love Tengri | love Tengri | love        |
+---------------+-------------+-------------+
```

`chr()`

Описание Возвращает символ, соответствующий значению кода ASCII или кода Unicode, заданному в `argument`.

Использование `chr(argument)`

▼ *Посмотреть пример*

```
SELECT
```

```
    chr(84) || chr(78) || chr(71) || chr(82) || chr(105) AS chr;
```

```
+-------+
|  chr  |
+-------+
| TNGRi |
+-------+
```

md5()

Описание Возвращает хеш MD5 данных из `argument` в виде строки (`VARCHAR`).

Использование `md5(argument)`

В `argument` могут быть двоичные данные или строка.

▼ *Посмотреть примеры*

```
SELECT
    md5('\xAA\xFF'::BLOB) as md5_hash;
```

```
+----------------------------------+
|             md5_hash             |
+----------------------------------+
| 1fab7f7621f5ddc051ebd1f2c63c4665 |
+----------------------------------+
```

```
SELECT
    md5('Tengri') as md5_hash;
```

```
+----------------------------------+
|             md5_hash             |
+----------------------------------+
| 846b02d31131a10bd6ac0ba189c65bef |
+----------------------------------+
```

sha1()

Описание Возвращает хеш SHA-1 данных из `argument` в виде строки (`VARCHAR`).

Использование `sha1(argument)`

В `argument` могут быть двоичные данные или строка.

▼ *Посмотреть примеры*

```
SELECT
```

```
    sha1('\xAA\xFF'::BLOB) as sha1_hash;
```

```
+------------------------------------------+
|                 sha1_hash                |
+------------------------------------------+
| e89b0db325637edfacde04a76005c492e2c5aeca |
+------------------------------------------+
```

```
SELECT
    sha1('Tengri') as sha1_hash;
```

```
+------------------------------------------+
|                 sha1_hash                |
+------------------------------------------+
| b514525a19995a2442d7565bfd9bb42d9dc71a13 |
+------------------------------------------+
```

sha256()

Описание Возвращает хеш SHA-256 данных из `argument` в виде строки (`VARCHAR`).

Использование `sha256(argument)`

В `argument` могут быть двоичные данные или строка.

▼ *Посмотреть пример*

```
SELECT
    sha256('\xAA\xFF'::BLOB) as sha256_hash;
```

```
+----------------------------------------------------------------+
|                           sha256_hash                          |
+----------------------------------------------------------------+
| 768318522cac43261e8ef4946c2296a3643d523a8d5bda8ff5b82aa64470421a |
+----------------------------------------------------------------+
```

```
SELECT
    sha256('Tengri') as sha256_hash;
```

```
+----------------------------------------------------------------+
|                           sha256_hash                          |
+----------------------------------------------------------------+
| 8aaacef66663b14ee7c5a03dbaec7b40f0f3bf17bd12d2ed4f9aaad0e10a0d77 |
+----------------------------------------------------------------+
```

Оператор ||

Описание	Конкатенирует несколько строк, массивов или двоичных значений.
Использование	`argument1 || argument2 || ...`

Пустые значения (`NULL`) игнорируются.

См. также concat().

▼ *Посмотреть примеры*

```
SELECT
    '\xAA'::BLOB || '\xff'::BLOB as result_blob,
    'I' || ' ' || 'love' || ' ' || 'Tengri' as result_string,
    ['T', 'e'] || ['n', 'g', 'r', 'i'] as result_array;
```

```
+-------------+---------------+---------------+
| result_blob | result_string |  result_array |
+-------------+---------------+---------------+
|             | I love Tengri | {T,e,n,g,r,i} |
+-------------+---------------+---------------+
```

Обратите внимание, что значения в столбце `result_blob` в выводе не отображаются (так как имеют тип `BLOB`).

С помощью выражения `DESCRIBE` выведем типы данных для всех столбцов в таблице, созданной так же, как в предыдущем примере:

```
CREATE TABLE concat AS

SELECT
    '\xAA'::BLOB || '\xff'::BLOB as result_blob,
    'I' || ' ' || 'love' || ' ' || 'Tengri' as result_string,
    ['T', 'e'] || ['n', 'g', 'r', 'i'] as result_array;

DESCRIBE TABLE concat;
```

```
+---------------+-------------+------+------+---------+-------+
|  column_name  | column_type | null |  key | default | extra |
+---------------+-------------+------+------+---------+-------+
| result_blob   | BLOB        | YES  | null | null    | null  |
+---------------+-------------+------+------+---------+-------+
| result_string | VARCHAR     | YES  | null | null    | null  |
+---------------+-------------+------+------+---------+-------+
| result_array  | VARCHAR[]   | YES  | null | null    | null  |
+---------------+-------------+------+------+---------+-------+
```

Функции для регулярных выражений

Функции для работы с текстовыми строками с помощью регулярных выражений. В регулярных выражениях используется синтаксис RE2.

`regexp_extract()`

Описание	Извлекает из строки подстроку по заданному регулярному выражению.
Использование	`regexp_extract(string, regexp[, group_number, options] | [, name_list, options])`

Если в строке `string` имеется подстрока, накрытая шаблоном регулярного выражения `regexp`, возвращает эту подстроку. Если подстрок, накрытых шаблоном, несколько, то возвращает первую подстроку. Если таких подстрок не найдено, возвращает пустую строку.

Можно задать номер возвращаемой группы и параметры или список имен групп и параметры (см. примеры).

▼ *Опции регулярных выражений*

- `'c'` — сопоставление с учетом регистра (используется по умолчанию, но может быть задано явно)
- `'i'` — сопоставление без учета регистра
- `'l'` — сопоставление литералов вместо фрагментов регулярных выражений

▼ *Посмотреть примеры*

```
SELECT
    regexp_extract('Tengri', '..')                              AS result_1,
    regexp_extract('Tengri', 'n.*')                             AS result_2,
    regexp_extract('Tengri', '.*')                              AS result_3,
    regexp_extract('Tengri', '^.{5}')                           AS result_4,
    regexp_extract('Tengri', '[TNGRi]$')                        AS result_5,
    regexp_extract('Tengri, TNGRi', 't.', 0, 'i')               AS result_6,
    regexp_extract('Tengri; TNGRi;', '(.*); (.*);', ['1', '2']) AS result_7,
    regexp_extract('Tengri', '.{7}')                            AS result_empty;
```

```
+----------+----------+----------+----------+----------+----------
+-----------------------------------------+--------------+
| result_1 | result_2 | result_3 | result_4 | result_5 | result_6 | result_7
| result_empty |
+----------+----------+----------+----------+----------+----------
+-----------------------------------------+--------------+
| Te       | ngri     | Tengri   | Tengr    | i        | Te       | {"1": "Tengri", "2":
"TNGRi"}" |              |
+----------+----------+----------+----------+----------+----------
+-----------------------------------------+--------------+
```

`regexp_extract_all()`

Описание	Извлекает из строки все не пересекающиеся подстроки по заданному регулярному выражению. Возвращает массив подстрок.
Использование	`regexp_extract_all(string, regexp[, <num>, options])`

Если подстрок в `string`, накрытых шаблоном `regexp` не найдено, возвращает пустой список.

Параметры

- `<num>` — номер группы внутри шаблона, которую следует вернуть (для каждой подстроки). По умолчанию (если параметр не задан) возвращается вся подстрока. Нумерация групп начинается с `1`. Группы в шаблоне выделяются круглыми скобками.

  ▼ *Подробнее о параметре на примерах*

  Извлечем из текста все сочетания букв с точкой после них:

```
SELECT
    regexp_extract_all('My name is Tengri. My nickname is TNGRi.',
                       '(\w+)(\.)')
AS result;
```

```
+------------------+
|      result      |
+------------------+
| {Tengri.,TNGRi.} |
+------------------+
```

  Теперь извлечем из того же текста с помощью того же регулярного выражения сочетания букв перед точкой, но без самой точки. Для этого зададим номер группы `1`:

```
SELECT
    regexp_extract_all('My name is Tengri. My nickname is TNGRi.',
                       '(\w+)(\.)',
                       1)
AS result;
```

```
+----------------+
|     result     |
+----------------+
| {Tengri,TNGRi} |
+----------------+
```

▼ *Опции регулярных выражений*

- `'c'` — сопоставление с учетом регистра (используется по умолчанию, но может быть задано

явно)

- `'i'` — сопоставление без учета регистра
- `'l'` — сопоставление литералов вместо фрагментов регулярных выражений

▼ *Посмотреть еще примеры*

```
SELECT
    regexp_extract_all('My name is Tengri. My nickname is TNGRi.', '\w+')
AS words;
```

```
+----------------------------------------+
|                  words                 |
+----------------------------------------+
| {My,name,is,Tengri,My,nickname,is,TNGRi} |
+----------------------------------------+
```

Используем комбинацию функций `regexp_extract_all` и `unnest`, чтобы извлечь данные из структурированного текста:

```
CREATE TABLE text_table(text_data VARCHAR);

INSERT INTO text_table VALUES
('Name: "Tengri", Nickname:"TNGRi"'),
('Country: "Russia", Capital:"Moscow"');

SELECT
    unnest(
        regexp_extract_all(text_data, '(\w+):\s*"(.*?)"', 1)
          ) AS key,
    unnest(
        regexp_extract_all(text_data, '(\w+):\s*"(.*?)"', 2)
          ) AS value
FROM text_table;
```

```
+----------+--------+
|    key   | value  |
+----------+--------+
| Name     | Tengri |
+----------+--------+
| Nickname | TNGRi  |
+----------+--------+
| Country  | Russia |
+----------+--------+
| Capital  | Moscow |
+----------+--------+
```

`regexp_full_match()`

Описание Проверяет, накрывает ли регулярное выражение строку полностью.

Использование `regexp_full_match(string, regexp[, options])`

Если шаблон регулярного выражения `regexp` полностью накрывает строку `string`, то возвращает `true`, иначе — `false`.

▼ *Опции регулярных выражений*

- `'c'` — сопоставление с учетом регистра (используется по умолчанию, но может быть задано явно)
- `'i'` — сопоставление без учета регистра
- `'l'` — сопоставление литералов вместо фрагментов регулярных выражений

▼ *Посмотреть примеры*

```
SELECT
    regexp_full_match('Tengri', 'gri$')     AS result_1, -- false expected
    regexp_full_match('Tengri', '.')        AS result_2, -- false expected
    regexp_full_match('Tengri', '.*')       AS result_3, -- true expected
    regexp_full_match('Tengri', '^\w+gri$') AS result_4, -- true expected
    regexp_full_match('Tengri', 't.*', 'i') AS result_5; -- true expected
```

```
+----------+----------+----------+----------+----------+
| result_1 | result_2 | result_3 | result_4 | result_5 |
+----------+----------+----------+----------+----------+
| false    | false    | true     | true     | true     |
+----------+----------+----------+----------+----------+
```

`regexp_matches()`

Описание Проверяет, содержится ли регулярное выражение внутри строки.

Использование `regexp_matches(string, regexp[, options])`

Если внутри строки `string` найдена хотя бы одна подстрока, накрытая шаблоном `regexp`, то возвращает `true`, иначе — `false`.

▼ *Опции регулярных выражений*

- `'c'` — сопоставление с учетом регистра (используется по умолчанию, но может быть задано явно)
- `'i'` — сопоставление без учета регистра
- `'l'` — сопоставление литералов вместо фрагментов регулярных выражений

▼ *Посмотреть примеры*

```
SELECT
SELECT
```

```
    regexp_matches('Tengri', '.+T')     AS result_1, -- false expected
    regexp_matches('Tengri', '.*T')     AS result_2, -- true expected
    regexp_matches('Tengri', 'T')       AS result_3, -- true expected
    regexp_matches('Tengri', 't', 'i')  AS result_4, -- true expected
    regexp_matches('T*ngri', 'T*', 'l') AS result_5; -- true expected
```

```
+----------+----------+----------+----------+----------+
| result_1 | result_2 | result_3 | result_4 | result_5 |
+----------+----------+----------+----------+----------+
| false    | true     | true     | true     | true     |
+----------+----------+----------+----------+----------+
```

`regexp_replace()`

Описание Заменяет подстроку, накрытую регулярным выражением, на указанную строку.

Использование `regexp_replace(string, regexp, target[, options])`

Если внутри строки `string` найдена подстрока, накрытая шаблоном `regexp`, то она заменяется на строку `target`. Если таких подстрок найдено несколько, то заменяется только первая. Если подстрок не найдено, возвращается исходная строка `string`.

▼ *Опции регулярных выражений*

- `'c'` — сопоставление с учетом регистра (используется по умолчанию, но может быть задано явно)
- `'i'` — сопоставление без учета регистра
- `'l'` — сопоставление литералов вместо фрагментов регулярных выражений

▼ *Посмотреть примеры*

```
SELECT
    regexp_replace('Tengri', '.', 't')             AS result_1,
    regexp_replace('Tengri', '.*', 't')            AS result_2,
    regexp_replace('Tengri', 'e.*r', 'NGR')        AS result_3,
    regexp_replace('Tengri', 'a', 't')             AS result_4,
    regexp_replace('Tengri', 'tengr', 'TNGR', 'i') AS result_5;
```

```
+----------+----------+----------+----------+----------+
| result_1 | result_2 | result_3 | result_4 | result_5 |
+----------+----------+----------+----------+----------+
| tengri   | t        | TNGRi    | Tengri   | TNGRi    |
+----------+----------+----------+----------+----------+
```

`regexp_split_to_array()`

Описание Разрезает строку на части, разделенные регулярным выражением, и

возвращает части в виде массива.

Использование `regexp_split_to_array(string, regexp[, options])`

Псевдонимы `string_split_regex()`

Если в строке `string` найдены подстроки, накрытые шаблоном `regexp`, то возвращаются части исходной строки, не накрытые шаблоном, в виде массива. Если найденные подстроки стоят в начале или в конце исходной строки, то в результирующий массив попадут пустые строки для начала и конца строки. Если подстрок не найдено, то будет возвращен массив из одной исходной строки.

▼ *Опции регулярных выражений*

- `'c'` — сопоставление с учетом регистра (используется по умолчанию, но может быть задано явно)
- `'i'` — сопоставление без учета регистра
- `'l'` — сопоставление литералов вместо фрагментов регулярных выражений

▼ *Посмотреть примеры*

```
SELECT
    string_split_regex('My name is Tengri. My nickname is TNGRi.', '\.\s')
AS sentences,
    string_split_regex('My name is Tengri. My nickname is TNGRi.', '[\.\s]+')
AS words;
```

```
+-----------------------------------------+-----------------------------------------+
|                 sentences               |                  words                  |
+-----------------------------------------+-----------------------------------------+
| {My name is Tengri,My nickname is TNGRi.} | {My,name,is,Tengri,My,nickname,is,TNGRi,} |
+-----------------------------------------+-----------------------------------------+
```

`regexp_split_to_table()`

Описание Разрезает строку на части, разделенные регулярным выражением, и возвращает части в виде строк.

Использование `regexp_split_to_table(string, regexp[, options])`

Если в строке `string` найдены подстроки, накрытые шаблоном `regexp`, то возвращаются части исходной строки, не накрытые шаблоном, в виде столбца, где каждая часть записана в свою ячейку. Если найденные подстроки стоят в начале или в конце исходной строки, то в результат попадут пустые строки для начала и конца строки. Если подстрок не найдено, то будет возвращен столбец с одной ячейкой, в которой записана исходная строка.

▼ *Опции регулярных выражений*

- `'c'` — сопоставление с учетом регистра (используется по умолчанию, но может быть задано явно)
- `'i'` — сопоставление без учета регистра

- `'l'` — сопоставление литералов вместо фрагментов регулярных выражений

▼ *Посмотреть примеры*

```
SELECT
    regexp_split_to_table('My name is Tengri. My nickname is TNGRi.', '\.\s')
AS sentences,
    regexp_split_to_table('My name is Tengri. My nickname is TNGRi.', '[\.\s]+')
AS words;
```

```
+----------------------+----------+
|       sentences      |   words  |
+----------------------+----------+
| My name is Tengri    | My       |
+----------------------+----------+
| My nickname is TNGRi. | name     |
+----------------------+----------+
| null                 | is       |
+----------------------+----------+
| null                 | Tengri   |
+----------------------+----------+
| null                 | My       |
+----------------------+----------+
| null                 | nickname |
+----------------------+----------+
| null                 | is       |
+----------------------+----------+
| null                 | TNGRi    |
+----------------------+----------+
| null                 |          |
+----------------------+----------+
```

Оператор `~`

Описание	Проверяет, накрывает ли регулярное выражение строку полностью.
Использование	`string ~ regexp`

Может использоваться с оператором отрицания `!`: `string !~ regexp`.

Полностью синонимичен оператору `SIMILAR TO`.

▼ *Посмотреть примеры*

```
SELECT
    'Tengri' !~ 'TNGRi' AS result_1,
    'Tengri' ~ 'T.*'    AS result_2,
    'Tengri' ~ 'T.+i'   AS result_3,
    'TNGRi'  ~ 'T.+i'   AS result_4,
    'T.+i'   ~ 'T.+i'   AS result_5;
```

```
+----------+----------+----------+----------+----------+
| result_1 | result_2 | result_3 | result_4 | result_5 |
+----------+----------+----------+----------+----------+
| true     | true     | true     | true     | true     |
+----------+----------+----------+----------+----------+
```

Функции для даты и времени

Функции для даты и времени — это функции для работы с данными типов `DATE`, `TIME`, `TIMESTAMP` и `TIMESTAMPTZ`.

`current_time()`

Описание Возвращает текущее время в виде значения типа `TIME`.

Использование `current_time` или `current_time()`

Псевдонимы `get_current_time()`

▼ *Посмотреть пример*

```
SELECT
    current_time       AS cur_time_1,
    current_time()     AS cur_time_2,
    get_current_time() AS cur_time_3;
```

```
+-----------------+-----------------+-----------------+
|    cur_time_1   |    cur_time_2   |    cur_time_3   |
+-----------------+-----------------+-----------------+
| 10:33:24.016000 | 10:33:24.016000 | 10:33:24.016000 |
+-----------------+-----------------+-----------------+
```

`now()`

Описание Возвращает текущее время в виде значения типа `TIMESTAMPTZ`

Использование `now()`

▼ *Посмотреть пример*

```
SELECT
    now() AS cur_time;
```

```
+---------------------------------+
| cur_time                        |
+---------------------------------+
| 2025-08-26 13:38:58.461000+00:00 |
```

```
+---------------------------------+
```

`datepart()`

Описание Возвращает указанную часть от значения даты или времени в виде значения типа `BIGINT`.

Использование `datepart('<part>', (TIME | DATE | ...) <date_time>)`

Псевдонимы `date_part()`

Аргументами могут быть значения типов: `TIME`, `DATE`, `TIMESTAMP` или `TIMESTAMPTZ`.

▼ *Части могут быть указаны с помощью следующих литералов*

- `century` — век
- `day` — день
- `decade` — десятилетие
- `hour` — час
- `microseconds` — микросекунды
- `millennium` — тысячелетие
- `milliseconds` — миллисекунды
- `minute` — минута
- `month` — месяц
- `quarter` — квартал
- `second` — секунда
- `year` — год

▼ *Посмотреть примеры*

```
SELECT
    datepart('milliseconds', TIMESTAMP '2025-02-25 00:00:00.1') AS milliseconds,
    datepart('hour',         TIME '2025-02-25 00:00:00')        AS hour,
    datepart('millennium',   DATE '2025-02-25')                 AS millennium;
```

```
+--------------+------+------------+
| milliseconds | hour | millennium |
+--------------+------+------------+
| 100          | 0    | 3          |
+--------------+------+------------+
```

`date_diff()`

Описание Возвращает количество единиц времени между двумя моментами времени в виде значения типа `BIGINT`.

Использование `date_diff('<part>', start, end)`

Аргументами могут быть значения типов: `TIME`, `DATE`, `TIMESTAMP` или `TIMESTAMPTZ`.

▼ *Единицы могут быть указаны с помощью следующих литералов*

- `century` — век
- `day` — день
- `decade` — десятилетие
- `hour` — час
- `microseconds` — микросекунды
- `millennium` — тысячелетие
- `milliseconds` — миллисекунды
- `minute` — минута
- `month` — месяц
- `quarter` — квартал
- `second` — секунда
- `year` — год

▼ *Посмотреть примеры*

```
SELECT
    date_diff('second', TIME '01:02:03', TIME '03:02:01') AS diff_in_seconds,
    date_diff('minute', TIME '01:02:03', TIME '03:02:01') AS diff_in_minutes,
    date_diff('hour',   TIME '01:02:03', TIME '03:02:01') AS diff_in_hours,
    date_diff('day',    TIMESTAMP '2025-02-25 01:02:03', TIMESTAMP '2025-02-26 03:02:01')
AS diff_in_days,
    date_diff('day',    TIMESTAMP '2025-02-26 01:02:03', TIMESTAMP '2025-02-25 03:02:01')
AS diff_in_days,
    date_diff('day',    DATE '2024-02-27', DATE '2025-02-27') AS diff_in_days;
```

```
+-----------------+-----------------+---------------+-------------+-------------
+--------------+
| diff_in_seconds | diff_in_minutes | diff_in_hours | diff_in_days | diff_in_days |
diff_in_days |
+-----------------+-----------------+---------------+-------------+-------------
+--------------+
| 7198            | 120             | 2             | 1            | -1           | 366
|
+-----------------+-----------------+---------------+-------------+-------------
+--------------+
```

`date_trunc()`

Описание Сокращает момент времени до указанной точности.

Использование `date_trunc('<part>', <time_stamp>)`

Сокращает момент времени `TIMESTAMP` до указанной единицы точности и возвращает начальный момент времени для данной единицы в виде значения типа `TIMESTAMP` или `DATE`.

Некоторые примеры использования описаны в этом сценарии.

▼ *Единицы могут быть указаны с помощью следующих литералов*

- `century` — век
- `day` — день
- `decade` — десятилетие
- `hour` — час
- `microseconds` — микросекунды
- `millennium` — тысячелетие
- `milliseconds` — миллисекунды
- `minute` — минута
- `month` — месяц
- `quarter` — квартал
- `second` — секунда
- `year` — год

▼ *Посмотреть примеры*

```
SELECT
    date_trunc('minute',    TIMESTAMP '2025-02-25 01:02:03') AS minute,
    date_trunc('hour',      TIMESTAMP '2025-02-25 01:02:03') AS hour,
    date_trunc('day',       TIMESTAMP '2025-02-25 01:02:03') AS day,
    date_trunc('month',     TIMESTAMP '2025-02-25 01:02:03') AS month,
    date_trunc('quarter',   TIMESTAMP '2025-02-25 01:02:03') AS quarter,
    date_trunc('year',      TIMESTAMP '2025-02-25 01:02:03') AS year;
```

```
+---------------------+---------------------+------------+------------+------------
+------------+
| minute              | hour                | day        | month      | quarter    | year
|
+---------------------+---------------------+------------+------------+------------
+------------+
| 2025-02-25 01:02:00 | 2025-02-25 01:00:00 | 2025-02-25 | 2025-02-01 | 2025-01-01 | 2025-
01-01 |
+---------------------+---------------------+------------+------------+------------
+------------+
```

`strptime()`

Описание Конвертирует текст в момент времени по указанному формату.

Использование `strptime(<string>, <format>)`

Конвертирует текст в момент времени `TIMESTAMP` по указанному формату. В случае невозможности конвертации выдает ошибку.

Некоторые примеры использования описаны в этом сценарии.

▼ *Формат может быть указан с помощью следующих выражений*

Выражение	Описание	Пример
`%a`	Сокращенное название дня недели.	*Sun, Mon, ...*
`%A`	Полное название дня недели.	*Sunday, Monday, ...*
`%b`	Сокращенное название месяца.	*Jan, Feb, ..., Dec*
`%B`	Полное название месяца.	*January, February, ...*
`%c`	Представление даты и времени в формате ISO	*1992-03-02 10:30:20*
`%d`	День месяца в виде десятичного числа с нулями в начале.	*01, 02, ..., 31*
`%-d`	День месяца в виде десятичного числа.	*1, 2, ..., 30*
`%f`	Микросекунда в виде десятичного числа с нулями в начале.	*000000 - 999999*
`%g`	Миллисекунда в виде десятичного числа с нулями в начале.	*000 - 999*
`%G`	Год по стандарту ISO 8601 с указанием века, в котором находится большая часть недели по стандарту ISO (см. `%V`).	*0001, 0002, ..., 2013, 2014, ..., 9998, 9999*
`%H`	Час (24-часовой формат) в виде десятичного числа с нулями в начале.	*00, 01, ..., 23*
`%-H`	Час (24-часовой формат) в виде десятичного числа.	*0, 1, ..., 23*
`%I`	Час (12-часовой формат) в виде десятичного числа с нулями в начале.	*01, 02, ..., 12*
`%-I`	Час (12-часовой формат) в виде десятичного числа.	*1, 2, ... 12*
`%j`	День года в виде десятичного числа с нулями в начале.	*001, 002, ..., 366*
`%-j`	День года в виде десятичного числа.	*1, 2, ..., 366*
`%m`	Месяц в виде десятичного числа с нулями в начале.	*01, 02, ..., 12*
`%-m`	Месяц в виде десятичного числа.	*1, 2, ..., 12*
`%M`	Минута в виде десятичного числа с нулями в начале.	*00, 01, ..., 59*
`%-M`	Минута в виде десятичного числа.	*0, 1, ..., 59*
`%n`	Наносекунда в виде десятичного числа с нулями в начале.	*000000000 - 999999999*
`%p`	AM или PM в соответствии с локалью.	*AM, PM*
`%S`	Секунды в виде десятичного числа с нулями в начале	*00, 01, ..., 59*
`%-S`	Секунды в виде десятичного числа.	0_, 1, ..., 59_
`%u`	День недели по стандарту ISO 8601 в виде десятичного числа, где 1 — понедельник.	*1, 2, ..., 7*

Выражение	Описание	Пример
`%U`	Номер недели в году. Неделя 01 начинается в первое воскресенье года, поэтому может быть неделя 00. Обратите внимание, что это не соответствует стандарту даты недели в ISO-8601.	*00, 01, ..., 53*
`%V`	Неделя ISO 8601 в виде десятичного числа, где понедельник является первым днем недели. Неделя 01 — это неделя, содержащая 4 января. Обратите внимание, что `%V` несовместим с выражением для года `%Y`. Вместо этого используйте год ISO `%G`.	*01, ..., 53*
`%w`	День недели в виде десятичного числа.	*0, 1, ..., 6*
`%W`	Номер недели в году. Неделя 01 начинается в первый понедельник года, поэтому может быть неделя 00. Обратите внимание, что это не соответствует стандарту даты недели в ISO-8601.	*00, 01, ..., 53*
`%x`	Представление даты в формате ISO	*1992-03-02*
`%X`	Представление времени в формате ISO	*10:30:20*
`%y`	Год без века в виде десятичного числа с нулями в начале.	*00, 01, ..., 99*
`%-y`	Год без века в виде десятичного числа.	*0, 1, ..., 99*
`%Y`	Год с указанием века в виде десятичного числа.	*2013, 2019 и т. д.*
`%z`	Смещение времени от UTC в формате ±HH:MM, ±HHMM или ±HH.	*-0700*
`%Z`	Название часового пояса.	*Europe/Amsterdam*
`%%`	Буквальный символ `%`.	*%*

▼ *Посмотреть примеры*

```
SELECT
    strptime('Feb 25, 2025, 01:02:03 AM', '%b %d, %Y, %H:%M:%S %p') AS timestamp_1,
    strptime('2025-02-25, 01:02:03',      '%x, %X')                 AS timestamp_2,
    strptime('01:02, 01.02.99',           '%H:%M, %d.%m.%y')        AS timestamp_3,
    strptime('01:02%PM--01.02.99',        '%H:%M%%p--%d.%m.%y')     AS timestamp_4,
    strptime('1:1, 1.1.1',                '%-H:%-M, %-d.%-m.%-y')   AS timestamp_5,
    strptime('1',                         '%Y')                     AS timestamp_6;
```

```
+---------------------+---------------------+---------------------+---------------------
+---------------------+---------------------+
| timestamp_1         | timestamp_2         | timestamp_3         | timestamp_4         |
timestamp_5         | timestamp_6         |
+---------------------+---------------------+---------------------+---------------------
+---------------------+---------------------+
| 2025-02-25 01:02:03 | 2025-02-25 01:02:03 | 1999-02-01 01:02:00 | 1999-02-01 13:02:00 |
2001-01-01 01:01:00 | 0001-01-01 00:00:00 |
+---------------------+---------------------+---------------------+---------------------
```

```
+--------------------+--------------------+
```

Оператор `+`

Описание Прибавляет правый аргумент к левому.

Использование `<num> + <num> [+ ...]` или `TIME + INTERVAL [+ ...]`

Если используется с типами для для даты и времени, то прибавляет интервал к значению времени. Возвращает значение типа `TIME`.

См. также `add()`.

▼ *Посмотреть примеры*

```
SELECT
    3 + 2      AS result_1,
    3 + 2 + -1 AS result_2,
    1.1 + 1.9  AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 5        | 4        | 3.0      |
+----------+----------+----------+
```

```
SELECT
    TIME '12:11:10'     + INTERVAL 3 hours AS result_time_1,
    INTERVAL '12:11:10' + TIME '1:1:1'     AS result_time_2;
```

```
+--------------+--------------+
| result_time_1 | result_time_2 |
+--------------+--------------+
| 15:11:10      | 13:12:11      |
+--------------+--------------+
```

Оператор `-`

Описание Вычитает правый аргумент из левого.

Использование `<num> - <num> [- ...]` или `TIME - INTERVAL [- ...]`

Если используется с типами для для даты и времени, то вычитает интервал из значения времени. Возвращает значение типа `TIME`.

См. также `subtract()`.

▼ *Посмотреть примеры*

```
SELECT
    3 - 2      AS result_1,
    3 - 2 - +1 AS result_2,
    1.2 - 0.2  AS result_3;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| 1        | 0        | 1.0      |
+----------+----------+----------+
```

```
SELECT
    TIME '12:11:10' - INTERVAL 3 HOUR AS result_time_1,
    TIME '12:11:10' - INTERVAL 3 HOUR - INTERVAL 1 HOUR AS result_time_2;
```

```
+---------------+---------------+
| result_time_1 | result_time_2 |
+---------------+---------------+
| 09:11:10      | 08:11:10      |
+---------------+---------------+
```

Функции для JSON

Функции для работы с файлами расширения `.json` и с типом данных `JSON`.

Путь внутри структуры JSON

Во многих функциях для JSON используется путь внутри структуры JSON как один из аргументов. Путь может задаваться в любой из двух нотаций по следующим стандартам:

- JSONPath
  - `$.key1.key2` — обращение к значению ключа `key2`
  - `$.key1.key2[i]` — обращение к `i`-му элементу списка в значении ключа `key2`
- JSON Pointer
  - `/key1/key2` — обращение к значению ключа `key2`
  - `/key1/key2/i` — обращение к `i`-му элементу списка в значении ключа `key2`

Нумерация элементов списка в структуре JSON начинается с `0`.

В примерах ниже используется структура JSON из этого примера:

```
CREATE TABLE js_table(js_data JSON);

INSERT INTO js_table VALUES
('{
  "first_name": "John",
  "last_name": "Smith",
  "is_alive": true,
  "age": 27,
  "address": {
    "street_address": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postal_code": "10021-3100"
  },
  "phone_numbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [
    "Catherine",
    "Thomas",
    "Trevor"
  ],
  "spouse": null
}');
```

`json_array_length()`

Описание Возвращает количество элементов в массиве по указанному пути в JSON или `0`, если по указанному пути не массив.

Использование `json_array_length(json[, path])`

Если во втором аргументе задан список путей, то результатом будет список длин массивов по указанным путям.

▼ *Посмотреть примеры*

```
SELECT
    json_array_length(js_data, '$.phone_numbers')                AS phone_numbers,
    json_array_length(js_data, '$.children')                     AS children,
    json_array_length(js_data, ['$.phone_numbers', '$.children']) AS lists,
    json_array_length(js_data)                                   AS js_data,
    json_array_length(js_data, '$.first_name')                   AS first_name,
FROM js_table;
```

```
+---------------+----------+-------+---------+------------+
| phone_numbers | children | lists | js_data | first_name |
+---------------+----------+-------+---------+------------+
| 2             | 3        | {2,3} | 0       | 0          |
+---------------+----------+-------+---------+------------+
```

`json_contains()`

Описание Проверяет, есть ли в структуре JSON подструктура JSON, заданная во втором аргументе.

Использование `json_contains(json, json)`

Оба аргумента должны иметь тип `JSON`. Второй аргумент может быть числовым значением или текстовой строкой, при этом текстовая строка должна быть заключена в двойные кавычки.

▼ *Посмотреть примеры*

```
SELECT
    json_contains(js_data, '27')                      AS result_1,
    json_contains(js_data, '"John"')                  AS result_2,
    json_contains(js_data, '{"first_name": "John"}')  AS result_3,
    json_contains(js_data, '{"first_name": "Smith"}') AS result_4, -- false expected
FROM js_table;
```

```
+----------+----------+----------+----------+
| result_1 | result_2 | result_3 | result_4 |
+----------+----------+----------+----------+
| true     | true     | true     | false    |
+----------+----------+----------+----------+
```

`json_exists()`

Описание Проверяет, есть ли в структуре JSON указанный путь.

Использование `json_exists(json, path)`

Возвращает `BOOL` или массив `BOOL[]` для случаев, когда путь указывает на элементы списка в структуре JSON.

▼ *Посмотреть примеры*

```
SELECT
    json_exists(js_data, 'first_name')                AS result_1,
    json_exists(js_data, '$.first_name')              AS result_2,
    json_exists(js_data, '/first_name')               AS result_3,
    json_exists(js_data, '$.address.street_address') AS result_4,
    json_exists(js_data, '/address/street_address')  AS result_5,
    json_exists(js_data, '$.street_address')          AS result_6, -- false expected
```

```
    json_exists(js_data, '$.phone_numbers[*].type')  AS result_7,
    json_exists(js_data, '$..street_address')        AS result_8,
FROM js_table;
```

```
+----------+----------+----------+----------+----------+----------+-------------+----------
+
| result_1 | result_2 | result_3 | result_4 | result_5 | result_6 | result_7    | result_8
|
+----------+----------+----------+----------+----------+----------+-------------+----------
+
| true     | true     | true     | true     | true     | false    | {True,True} | {True}
|
+----------+----------+----------+----------+----------+----------+-------------+----------
+
```

`json_extract()`

Описание Извлекает данные из структуры JSON по указанному пути. Возвращает данные в виде `JSON`.

Использование `json_extract(json, path)` или `json_extract(json, [path1, path2, ...])`

Псевдонимы `json_extract_path`

Если в качестве второго аргумента задан список путей, то возвращает список значений.

▼ *Посмотреть примеры*

```
SELECT
    json_extract(js_data, 'first_name')                AS first_name,
    json_extract(js_data, ['first_name', 'last_name']) AS all_names,
    json_extract(js_data, '$.address.street_address')  AS street_address,
    json_extract(js_data, '$.phone_numbers[*].number') AS phone_numbers,
    json_extract(js_data, '$.children[0]')             AS child_1,
FROM js_table;
```

```
+------------+------------------+----------------+--------------------------------
+-----------+
| first_name | all_names        | street_address | phone_numbers                  |
child_1   |
+------------+------------------+----------------+--------------------------------
+-----------+
| John       | {"John","Smith"} | 21 2nd Street  | {"212 555-1234","646 555-4567"} |
Catherine |
+------------+------------------+----------------+--------------------------------
+-----------+
```

`json_extract_string()`

Описание Извлекает данные из структуры JSON по указанному пути. Возвращает данные в виде `VARCHAR`.

Использование `json_extract_string(json, path)` или `json_extract_string(json, [path1, path2, ...])`

Псевдонимы `json_extract_path_text`

Если в качестве второго аргумента задан список путей, то возвращает список значений.

▼ *Посмотреть примеры*

```
SELECT
    json_extract_string(js_data, 'first_name')               AS first_name,
    json_extract_string(js_data, ['first_name', 'last_name']) AS all_names,
    json_extract_string(js_data, '$.address.street_address')  AS street_address,
    json_extract_string(js_data, '$.phone_numbers[*].number') AS phone_numbers,
    json_extract_string(js_data, '$.children[0]')             AS child_1,
FROM js_table;
```

```
+------------+--------------+----------------+-----------------------------+-----------+
| first_name | all_names    | street_address | phone_numbers               | child_1   |
+------------+--------------+----------------+-----------------------------+-----------+
| John       | {John,Smith} | 21 2nd Street  | {212 555-1234,646 555-4567} | Catherine |
+------------+--------------+----------------+-----------------------------+-----------+
```

`json_group_array()`

Описание Возвращает список JSON, содержащий все значения столбца.

Использование `json_group_array(argument)`

Функция изменяет кардинальность данных.

▼ *Посмотреть примеры*

```
CREATE TABLE example (name VARCHAR);
INSERT INTO example VALUES ('Tengri'), ('TNGRi');

SELECT json_group_array(name) AS tengti_names
FROM example;
```

```
+--------------------+
| tengti_names       |
+--------------------+
| ["Tengri","TNGRi"] |
+--------------------+
```

`json_group_object()`

Описание Возвращает структуру JSON, содержащую все пары `key-value` из столбцов, указанных в аргументах.

Использование `json_group_object(argument1, argument2)`

Функция изменяет кардинальность данных.

▼ *Посмотреть примеры*

```
CREATE TABLE example (name VARCHAR, letters_num BIGINT);
INSERT INTO example VALUES
    ('Tengri', 6),
    ('TNGRi', 5);

SELECT json_group_object(name, letters_num) AS result
FROM example;
```

```
+------------------------+
| result                 |
+------------------------+
| {"Tengri":6,"TNGRi":5} |
+------------------------+
```

`json_keys()`

Описание Возвращает все ключи из указанной структуры JSON в виде `VARCHAR[]`.

Использование `json_keys(json[, path])`

Если во втором аргументе указан путь, то возвращает ключи структуры JSON по указанному пути. Если задан список путей, то результатом будет список списков ключей.

▼ *Посмотреть примеры*

```
SELECT
    json_keys(js_data) AS all_keys,
FROM js_table;
```

```
+-----------------------------------------------------------------------------+
| all_keys                                                                    |
+-----------------------------------------------------------------------------+
| {first_name,last_name,is_alive,age,address,phone_numbers,children,spouse} |
+-----------------------------------------------------------------------------+
```

```
SELECT
    json_keys(js_data, '$.address') AS address_keys,
```

```
FROM js_table;
```

```
+----------------------------------------+
| address_keys                           |
+----------------------------------------+
| {street_address,city,state,postal_code} |
+----------------------------------------+
```

```
SELECT
    json_keys(js_data, ['$.address',
                        '$.phone_numbers[0]'])
    AS address_and_phone_keys,
FROM js_table;
```

```
+-------------------------------------------------------------------------+
| address_and_phone_keys                                                  |
+-------------------------------------------------------------------------+
| {['street_address', 'city', 'state', 'postal_code'],['type', 'number']} |
+-------------------------------------------------------------------------+
```

`json_transform()`

Описание Трансформирует структуру JSON в соответствии с указанной структурой.

Использование `json_transform(json, string)`

Псевдонимы `from_json()`

Целевая структура указывается в виде текстовой строки во втором аргументе.

В случаях пропусков значений в исходной структуре JSON в результат проставляются значения `NULL`.

В случаях невозможности преобразовать типы данных в исходной структуре к указанным проставляется значение `NULL`.

▼ *Посмотреть примеры*

```
CREATE TABLE example (js_data JSON);
INSERT INTO example VALUES
    ('{
        "first_name": "John",
        "last_name": "Smith",
        "age": 27
      }'),
    ('{
        "first_name": "John",
        "is_alive": true,
        "age": 28
```

```
      }');

SELECT
    json_transform(js_data,
        '{
        "first_name": "VARCHAR",
        "last_name": "VARCHAR",
        "is_alive": "BOOL",
        "age": "BIGINT"
        }')
    AS result
FROM example;
```

```
+--------------------------------------------------------------------------+
| result                                                                   |
+--------------------------------------------------------------------------+
| {"first_name": "John", "last_name": "Smith", "is_alive": null, "age": 27} |
+--------------------------------------------------------------------------+
| {"first_name": "John", "last_name": null, "is_alive": true, "age": 28}    |
+--------------------------------------------------------------------------+
```

`json_transform_strict()`

Описание	Трансформирует структуру JSON в соответствии с указанной структурой. Выдает ошибку в случае несоответствия структур или типов.
Использование	`json_transform_strict(json, string)`
Псевдонимы	`from_json_strict()`

Целевая структура указывается в виде текстовой строки во втором аргументе.

В случаях, если в исходной структуре JSON отсутствуют значения из указанной структуры, выдает ошибку.

В случаях невозможности преобразовать типы данных в исходной структуре к указанным выдает ошибку.

▼ *Посмотреть примеры*

```
CREATE TABLE example (js_data JSON);
INSERT INTO example VALUES
    ('{
        "first_name": "John",
        "last_name": "Smith",
        "age": 27
      }'),
    ('{
        "first_name": "John",
        "is_alive": true,
        "age": 28
```

```
      }');
SELECT
    json_transform_strict(js_data,
        '{"first_name": "VARCHAR", "age": "BIGINT"}')
    AS result
FROM example;
```

```
+----------------------------------+
| result                           |
+----------------------------------+
| {"first_name": "John", "age": 27} |
+----------------------------------+
| {"first_name": "John", "age": 28} |
+----------------------------------+
```

```
CREATE TABLE example (js_data JSON);
INSERT INTO example VALUES
    ('{
        "first_name": "John",
        "last_name": "Smith",
        "age": 27
      }'),
    ('{
        "first_name": "John",
        "is_alive": true,
        "age": 28
      }');
SELECT
    json_transform_strict(js_data,
        '{"first_name": "BIGINT", "age": "BIGINT"}') -- error expected
    AS result
FROM example;
```

```
ERROR: InvalidInputException: Invalid Input Error:
Failed to cast value to numerical: "John"
```

```
CREATE TABLE example (js_data JSON);
INSERT INTO example VALUES
    ('{
        "first_name": "John",
        "last_name": "Smith",
        "age": 27
      }'),
    ('{
        "first_name": "John",
        "is_alive": true,
        "age": 28
      }');
```

```
SELECT
    json_transform_strict(js_data,
        '{"first_name": "VARCHAR", "last_name": "VARCHAR"}') -- error expected
    AS result
FROM example;
```

```
ERROR: InvalidInputException: Invalid Input Error:
Object {"first_name":"John","is_alive":true,"age":28} does not have key "last_name"
```

`json_valid()`

Описание Проверяет, является ли аргумент валидной структурой JSON.

Использование `json_valid(json)`

▼ *Посмотреть примеры*

```
SELECT
    json_valid(js_data)                  AS result_1,
    json_valid('{"first_name": "John"}') AS result_2,
    json_valid('{"first_name"}')         AS result_3, -- false expected
FROM js_table;
```

```
+----------+----------+----------+
| result_1 | result_2 | result_3 |
+----------+----------+----------+
| true     | true     | false    |
+----------+----------+----------+
```

`json_value()`

Описание Извлекает значения из структуры JSON по указанному пути.

Использование `json_value(json, path)`

Если по указанному пути значение не скалярное (список или вложенная структура), то возвращает `NULL`.

▼ *Посмотреть примеры*

```
SELECT
    json_value(js_data, 'first_name')      AS first_name,
    json_value(js_data, '$.phone_numbers[*].number') AS phone_numbers,
    json_value(js_data, '$.children[0]')   AS child_1,
    json_value(js_data, '$.phone_numbers') AS phone_numbers, -- NULL expected
    json_value(js_data, '$.address')       AS address,       -- NULL expected
FROM js_table;
```

```
+------------+--------------------------------+----------+--------------+---------+
| first_name | phone_numbers                  | child_1  | phone_numbers | address |
+------------+--------------------------------+----------+--------------+---------+
| John       | {"212 555-1234","646 555-4567"} | Catherine | null          | null    |
+------------+--------------------------------+----------+--------------+---------+
```

json()

Описание Сокращает запись структуры JSON (убирает пробелы и переносы строк).

Использование `json(json)`

▼ *Посмотреть примеры*

```
SELECT
    json('{
            "first_name": "John",
            "last_name": "Smith"
          }
        ') AS result_1,
    json(js_data) AS result_2,
FROM js_table;
```

```
+-------------------------------------------------
+-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-------------------------+
| result_1                                          | result_2
|
+-------------------------------------------------
+-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-------------------------+
| "{""first_name"":""John"",""last_name"":""Smith""}" |
"{""first_name"":""John"",""last_name"":""Smith"",""is_alive"":true,""age"":27,""address"":
{""street_address"":""21 2nd Street"",""city"":""New
York"",""state"":""NY"",""postal_code"":""10021-
3100""},""phone_numbers"":[{""type"":""home"",""number"":""212 555-
1234""},{""type"":""office"",""number"":""646 555-
4567""}],""children"":[""Catherine"",""Thomas"",""Trevor""],""spouse"":null}" |
+-------------------------------------------------
+-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
```

```
------------------------+
```

`read_json()`

Описание	Читает файл `.json` и записывает прочитанные данные в таблицу.
Использование	`read_json(filename[, columns = {column_name: 'column_type', ...}])`
Псевдонимы	`read_json_auto()`

Типы данных для столбцов определяются автоматически.

Параметры

- `columns` — опциональный параметр, в котором можно указать имена столбцов и их типы. В таком случае в результирующую таблицу будут добавлены только указанные столбцы.

Подробнее загрузка данных в Tengri описана на странице Загрузка данных.

В примерах используется файл `tengri_data_types.json` с типами данных Tengri и их краткими описаниями и файл `json_example_from_wikipedia.json` с этим примером.

▼ *Посмотреть примеры*

Прочитаем файл `tengri_data_types.json` и выведем первые пять строк таблицы:

```
SELECT *
FROM read_json(
    '<path>/tengri_data_types.json'
)
LIMIT 5
```

```
+----------+-----------+----------+--------------------------+
| name     | type      | category | description              |
+----------+-----------+----------+--------------------------+
| BIGINT   | data type | numeric  | Целые числа.             |
+----------+-----------+----------+--------------------------+
| BIGINT[] | data type | array    | Массивы целых чисел.     |
+----------+-----------+----------+--------------------------+
| BLOB     | data type | blob     | Двоичные объекты.        |
+----------+-----------+----------+--------------------------+
| BOOL     | data type | boolean  | Булевы значения.         |
+----------+-----------+----------+--------------------------+
| BOOL[]   | data type | array    | Массивы булевых значений. |
+----------+-----------+----------+--------------------------+
```

Прочитаем файл `tengri_data_types.json`, зададим только нужные колонки (ключи исходного файла) и выведем первые пять строк таблицы:

```
SELECT *
FROM read_json(
    '<path>/tengri_data_types.json',
    columns = {name: 'VARCHAR', category: 'VARCHAR'})
LIMIT 5
```

```
+----------+----------+
| name     | category |
+----------+----------+
| BIGINT   | numeric  |
+----------+----------+
| BIGINT[] | array    |
+----------+----------+
| BLOB     | blob     |
+----------+----------+
| BOOL     | boolean  |
+----------+----------+
| BOOL[]   | array    |
+----------+----------+
```

Прочитаем файл `json_example_from_wikipedia.json` и выведем всю построенную по нему таблицу. Обратим внимание на то, что вложенные структуры JSON записываются в ячейки таблицы и никак не разворачиваются (столбцы `address`, `phone_numbers`, `children`). Для разворачивания вложенных структур можно использовать функцию `unnest`.

```
SELECT *
FROM read_json(
    '<path>/json_example_from_wikipedia.json'
)
```

```
+------------+-----------+----------+-----
+-----------------------------------------------------------------------------------------
-----------+
-----------------------------------------------------------------------------------------
+---------------------------+--------+
| first_name | last_name | is_alive | age | address
| phone_numbers
| children                  | spouse |
+------------+-----------+----------+-----
+-----------------------------------------------------------------------------------------
-----------+
-----------------------------------------------------------------------------------------
+---------------------------+--------+
| John       | Smith     | true     | 27  | {"street_address": "21 2nd Street", "city":
"New York", "state": "NY", "postal_code": "10021-3100"} | {{'type': 'home', 'number': '212
555-1234'},{'type': 'office', 'number': '646 555-4567'}} | {Catherine,Thomas,Trevor} | null
|
+------------+-----------+----------+-----
```

```
+-----------------------------------------------------------------------------------------
----------+
-----------------------------------------------------------------------------------------
+-------------------------+--------+
```

Полезные ссылки

- Тип JSON

Функции для двоичных данных

Функции для двоичных данных — это функции для работы со значениями двоичного типа (`BLOB`).

`concat()`

Описание Конкатенирует несколько строк, массивов или двоичных значений.

Использование `concat(argument1, argument2, ...)`

Пустые значения (`NULL`) игнорируются.

См. также Оператор `||`.

▼ *Посмотреть пример*

```
SELECT
    concat('\xAA'::BLOB, '\xff'::BLOB)          AS result_blob,
    concat('I', ' ', 'love', ' ', 'Tengri')     AS result_string,
    concat(['T', 'e'], ['n', 'g', 'r', 'i'])    AS result_array;
```

```
+-------------+---------------+---------------+
| result_blob | result_string |  result_array |
+-------------+---------------+---------------+
| \xAA\xFF    | I love Tengri | {T,e,n,g,r,i} |
+-------------+---------------+---------------+
```

`md5()`

Описание Возвращает хеш MD5 данных из `argument` в виде строки (`VARCHAR`).

Использование `md5(argument)`

В `argument` могут быть двоичные данные или строка.

▼ *Посмотреть примеры*

```
SELECT
    md5('\xAA\xFF'::BLOB) as md5_hash;
```

```
+----------------------------------+
|             md5_hash             |
+----------------------------------+
| 1fab7f7621f5ddc051ebd1f2c63c4665 |
+----------------------------------+
```

```
SELECT
    md5('Tengri') as md5_hash;
```

```
+----------------------------------+
|             md5_hash             |
+----------------------------------+
| 846b02d31131a10bd6ac0ba189c65bef |
+----------------------------------+
```

sha1()

Описание Возвращает хеш SHA-1 данных из `argument` в виде строки (`VARCHAR`).

Использование `sha1(argument)`

В `argument` могут быть двоичные данные или строка.

▼ *Посмотреть примеры*

```
SELECT
    sha1('\xAA\xFF'::BLOB) as sha1_hash;
```

```
+------------------------------------------+
|                 sha1_hash                |
+------------------------------------------+
| e89b0db325637edfacde04a76005c492e2c5aeca |
+------------------------------------------+
```

```
SELECT
    sha1('Tengri') as sha1_hash;
```

```
+------------------------------------------+
|                 sha1_hash                |
+------------------------------------------+
| b514525a19995a2442d7565bfd9bb42d9dc71a13 |
+------------------------------------------+
```

sha256()

Описание Возвращает хеш SHA-256 данных из `argument` в виде строки (`VARCHAR`).

Использование `sha256(argument)`

В `argument` могут быть двоичные данные или строка.

▼ *Посмотреть пример*

```
SELECT
    sha256('\xAA\xFF'::BLOB) as sha256_hash;
```

```
+------------------------------------------------------------------+
|                            sha256_hash                           |
+------------------------------------------------------------------+
| 768318522cac43261e8ef4946c2296a3643d523a8d5bda8ff5b82aa64470421a |
+------------------------------------------------------------------+
```

```
SELECT
    sha256('Tengri') as sha256_hash;
```

```
+------------------------------------------------------------------+
|                            sha256_hash                           |
+------------------------------------------------------------------+
| 8aaacef66663b14ee7c5a03dbaec7b40f0f3bf17bd12d2ed4f9aaad0e10a0d77 |
+------------------------------------------------------------------+
```

Оператор ||

Описание Конкатенирует несколько строк, массивов или двоичных значений.

Использование `argument1 || argument2 || ...`

Пустые значения (`NULL`) игнорируются.

См. также concat().

▼ *Посмотреть примеры*

```
SELECT
    '\xAA'::BLOB || '\xff'::BLOB as result_blob,
    'I' || ' ' || 'love' || ' ' || 'Tengri' as result_string,
    ['T', 'e'] || ['n', 'g', 'r', 'i'] as result_array;
```

```
+-------------+---------------+--------------+
| result_blob | result_string |  result_array |
```

```
+------------+--------------+--------------+
|            | I love Tengri | {T,e,n,g,r,i} |
+------------+--------------+--------------+
```

Обратите внимание, что значения в столбце `result_blob` в выводе не отображаются (так как имеют тип `BLOB`).

С помощью выражения `DESCRIBE` выведем типы данных для всех столбцов в таблице, созданной так же, как в предыдущем примере:

```
CREATE TABLE concat AS

SELECT
    '\xAA'::BLOB || '\xff'::BLOB as result_blob,
    'I' || ' ' || 'love' || ' ' || 'Tengri' as result_string,
    ['T', 'e'] || ['n', 'g', 'r', 'i'] as result_array;

DESCRIBE TABLE concat;
```

```
+--------------+-------------+------+------+---------+-------+
|  column_name  | column_type | null |  key | default | extra |
+--------------+-------------+------+------+---------+-------+
| result_blob   | BLOB        | YES  | null | null    | null  |
+--------------+-------------+------+------+---------+-------+
| result_string | VARCHAR     | YES  | null | null    | null  |
+--------------+-------------+------+------+---------+-------+
| result_array  | VARCHAR[]   | YES  | null | null    | null  |
+--------------+-------------+------+------+---------+-------+
```

Функции для геоданных

Функции для работы с геопространственными данными и инструментами ГИС.

`ST_Point()`

Описание Создает точку типа `GEOMETRY`.

Использование `ST_Point(num, num)`

Создает точку типа `GEOMETRY` из двух чисел типа `DOUBLE`. Аргументами могут быть, например, географические координаты, заданные в градусах в виде десятичной дроби.

Некоторые примеры использования описаны в этом сценарии.

▼ *Посмотреть пример*

Создадим таблицу с названиями организаций и координатами их офисов. Вычислим расстояние от нулевого километра Москвы (`55.75579845052788, 37.617679973467204`) до офисов этих организаций и выведем эту информацию в виде таблицы:

```
CREATE TABLE demo.table (organization VARCHAR,
                         latitude DOUBLE,
                         longitude DOUBLE);

INSERT INTO demo.table VALUES
    ('Postgres Professional LLC', 55.69189394353437, 37.564623398131985),
    ('Oracle Corporation', 30.243622717202587, -97.72199761339736);

SELECT
    organization,
    round(ST_Distance_Sphere(ST_Point(latitude, longitude),
                             ST_Point(55.75579845052788, 37.617679973467204)
                            )/1000)
    AS "distance from moscow, km"
FROM demo.table;
```

```
+--------------------------+--------------------------+
| organization             | distance from moscow, km |
+--------------------------+--------------------------+
| Postgres Professional LLC | 8                        |
+--------------------------+--------------------------+
| Oracle Corporation        | 9557                     |
+--------------------------+--------------------------+
```

ST_Distance()

Описание Вычисляет расстояние между двумя точками на плоскости.

Использование `ST_Distance(GEOMETRY, GEOMETRY)`

Вычисляет расстояние на плоскости между двумя точками типа `GEOMETRY`.

▼ *Посмотреть пример*

```
SELECT
    ST_Distance('POINT (0 0)'::GEOMETRY, 'POINT (5 12)'::GEOMETRY)
        AS distance;
```

```
+----------+
| distance |
+----------+
| 13       |
+----------+
```

ST_Distance_Sphere()

Описание Вычисляет расстояние между двумя точками на сфере.

Использование `ST_Distance_Sphere(GEOMETRY, GEOMETRY)`

Вычисляет расстояние по гаверсину (большому кругу сферы) между двумя точками типа `GEOMETRY`.

Возвращает расстояние в метрах. Входные данные должны быть в координатах, заданных в градусах в виде десятичной дроби (формат WGS84, EPSG:4326) с порядком осей: широта (*latitude*), долгота (*longitude*).

Некоторые примеры использования описаны в этом сценарии.

▼ *Посмотреть пример*

Создадим таблицу с названиями организаций и координатами их офисов. Вычислим расстояние от нулевого километра Москвы (`55.75579845052788, 37.617679973467204`) до офисов этих организаций и выведем эту информацию в виде таблицы:

```
CREATE TABLE demo.table (organization VARCHAR,
                         latitude DOUBLE,
                         longitude DOUBLE);

INSERT INTO demo.table VALUES
    ('Postgres Professional LLC', 55.69189394353437, 37.564623398131985),
    ('Oracle Corporation', 30.243622717202587, -97.72199761339736);

SELECT
    organization,
    round(ST_Distance_Sphere(ST_Point(latitude, longitude),
                             ST_Point(55.75579845052788, 37.617679973467204)
                            )/1000)
    AS "distance from moscow, km"
FROM demo.table;
```

```
+--------------------------+--------------------------+
| organization             | distance from moscow, km |
+--------------------------+--------------------------+
| Postgres Professional LLC | 8                        |
+--------------------------+--------------------------+
| Oracle Corporation        | 9557                     |
+--------------------------+--------------------------+
```

Полезные ссылки

- Тип для геоданных

Оконные функции

Оконные функции — это функции, которые выполняют вычисления для набора строк, некоторым образом связанных с текущей строкой. При этом в отличие от агрегатных функций, значения, вычисленные оконными функциями, могут проставляться в каждую строку исходной таблицы.

Вызов оконной функции всегда содержит выражение `OVER`, следующее за названием и аргументами оконной функции. Выражение `OVER` определяет, как именно нужно разделить строки набора данных для обработки оконной функцией.

Выражение `OVER` может дополняться выражением `PARTITION BY`, которое разделяет строки набора данных по группам, объединяя строки в группы по совпадению значений указанных столбцов. Значение оконной функции вычисляется по строкам, попадающим в одну группу с текущей строкой.

`cume_dist()`

Описание Кумулятивное распределение.

Использование `cume_dist([ORDER BY column])`

Кумулятивное распределение: количество строк группы, предшествующих текущей строке или с тем же значением, что и в текущей строке, деленное на общее количество строк группы.

Если указано условие `ORDER BY`, значение вычисляется с использованием указанного порядка.

▼ *Посмотреть пример*

```
CREATE TABLE t(gr VARCHAR, num BIGINT);

INSERT INTO t VALUES
    ('A', 1),
    ('A', 2),
    ('A', 2),
    ('A', 3),
    ('B', 4);

SELECT gr, num,
    cume_dist(ORDER BY num) OVER (PARTITION BY gr) AS cume_dist
    FROM t ORDER BY num;
```

```
+-------+--------+-----------+
| gr    | num    | cume_dist |
+-------+--------+-----------+
| A     | 1      | 0.25      |
+-------+--------+-----------+
| A     | 2      | 0.75      |
+-------+--------+-----------+
| A     | 2      | 0.75      |
+-------+--------+-----------+
| A     | 3      | 1         |
+-------+--------+-----------+
| B     | 4      | 1         |
+-------+--------+-----------+
```

`dense_rank()`

Описание Ранг текущей строки внутри группы без пропусков.

Использование `dense_rank()`

Псевдонимы `rank_dense()`

Возвращает ранг текущей строки внутри группы без пропусков. По сути эта функция считает группы строк с совпадающими значениями по данному столбцу и назначает им номера внутри группы.

▼ *Посмотреть пример*

```
CREATE TABLE t(gr VARCHAR, num BIGINT);

INSERT INTO t VALUES
    ('A', 1),
    ('A', 2),
    ('A', 2),
    ('A', 5),
    ('B', 4);

SELECT gr, num,
    dense_rank() OVER (PARTITION BY gr ORDER BY num) AS dense_rank
    FROM t ORDER BY gr, num;
```

```
+----+--------+------------+
| gr | num    | dense_rank |
+----+--------+------------+
| A  | 1      | 1          |
+----+--------+------------+
| A  | 2      | 2          |
+----+--------+------------+
| A  | 2      | 2          |
+----+--------+------------+
| A  | 5      | 3          |
+----+--------+------------+
| B  | 4      | 1          |
+----+--------+------------+
```

`first_value()`

Описание Возвращает значение, вычисленное по указанному выражению для первой строки группы.

Использование `first_value(expression[ ORDER BY column][ IGNORE NULLS])`

Если указано условие `ORDER BY`, значение вычисляется с использованием указанного порядка.

Параметры

- `IGNORE NULLS` — значение вычисляется без учета строк со значением `NULL`.

▼ *Посмотреть пример*

```
CREATE TABLE t(gr VARCHAR, num BIGINT);

INSERT INTO t VALUES
    ('A', 1),
    ('A', 2),
    ('A', 3),
    ('A', NULL),
    ('B', 4);

SELECT gr, num,
    first_value(num) OVER (PARTITION BY gr) AS first_value
    FROM t ORDER BY gr, num;
```

```
+----+------+-------------+
| gr | num  | first_value |
+----+------+-------------+
| A  | 1    | 1           |
+----+------+-------------+
| A  | 2    | 1           |
+----+------+-------------+
| A  | 3    | 1           |
+----+------+-------------+
| A  | null | 1           |
+----+------+-------------+
| B  | 4    | 4           |
+----+------+-------------+
```

`lag()`

Описание	Возвращает значение, вычисленное для строки, сдвинутой на `offset` строк от текущей к началу группы.
Использование	`lag(expression[, offset[, default]][ ORDER BY column][ IGNORE NULLS])`

Если такой строки нет, возвращается значение `default` (оно должно быть совместимого по типу).

Оба аргумента, `offset` и `default`, являются опциональными. Если они не указываются, то используются следующие значения по умолчанию:

- `offset: 1`
- `default: NULL`

Если указано условие `ORDER BY`, значение вычисляется с использованием указанного порядка.

Параметры

- `IGNORE NULLS` — значение вычисляется без учета строк со значением `NULL`.

▼ *Посмотреть примеры*

```
CREATE TABLE t(gr VARCHAR, num BIGINT);

INSERT INTO t VALUES
    ('A', 1),
    ('A', 2),
    ('A', 3),
    ('B', 4),
    ('B', 5);

SELECT gr, num,
    lag(num) OVER (PARTITION BY gr) AS lag
    FROM t ORDER BY gr, num;
```

```
+----+-----+------+
| gr | num | lag  |
+----+-----+------+
| A  | 1   | null |
+----+-----+------+
| A  | 2   | 1    |
+----+-----+------+
| A  | 3   | 2    |
+----+-----+------+
| B  | 4   | null |
+----+-----+------+
| B  | 5   | 4    |
+----+-----+------+
```

```
CREATE TABLE t(gr VARCHAR, num BIGINT);

INSERT INTO t VALUES
    ('A', 1),
    ('A', 2),
    ('A', 3),
    ('B', 4),
    ('B', 5);

SELECT gr, num,
    lag(num, 1, -1) OVER (PARTITION BY gr) AS lag
    FROM t ORDER BY gr, num;
```

```
+----+-----+-----+
| gr | num | lag |
```

```
+----+-----+-----+
| A  | 1   | -1  |
+----+-----+-----+
| A  | 2   | 1   |
+----+-----+-----+
| A  | 3   | 2   |
+----+-----+-----+
| B  | 4   | -1  |
+----+-----+-----+
| B  | 5   | 4   |
+----+-----+-----+
```

`last_value()`

Описание	Возвращает значение, вычисленное по указанному выражению для последней строки группы.
Использование	`last_value(expression[ ORDER BY column][ IGNORE NULLS])`

Если указано условие `ORDER BY`, значение вычисляется с использованием указанного порядка.

Параметры

- `IGNORE NULLS` — значение вычисляется без учета строк со значением `NULL`.

▼ *Посмотреть пример*

```
CREATE TABLE t(gr VARCHAR, num BIGINT);

INSERT INTO t VALUES
    ('A', 1),
    ('A', 2),
    ('A', 3),
    ('A', NULL),
    ('B', 4),
    ('B', 5);

SELECT gr, num,
    last_value(num IGNORE NULLS) OVER (PARTITION BY gr) AS last_value
    FROM t ORDER BY gr, num;
```

```
+----+------+------------+
| gr | num  | last_value |
+----+------+------------+
| A  | 1    | 3          |
+----+------+------------+
| A  | 2    | 3          |
+----+------+------------+
| A  | 3    | 3          |
+----+------+------------+
```

```
| A  | null | 3          |
+----+------+------------+
| B  | 4    | 5          |
+----+------+------------+
| B  | 5    | 5          |
+----+------+------------+
```

`lead()`

Описание	Возвращает значение, вычисленное для строки, сдвинутой на `offset` строк от текущей к концу группы.
Использование	`lead(expression[, offset[, default]][ ORDER BY column][ IGNORE NULLS])`

Если такой строки нет, возвращается значение `default` (оно должно быть совместимого по типу).

Оба аргумента, `offset` и `default`, являются опциональными. Если они не указываются, то используются следующие значения по умолчанию:

- `offset: 1`
- `default: NULL`

Если указано условие `ORDER BY`, значение вычисляется с использованием указанного порядка.

Параметры

- `IGNORE NULLS` — значение вычисляется без учета строк со значением `NULL`.

▼ *Посмотреть примеры*

```sql
CREATE TABLE t(gr VARCHAR, num BIGINT);

INSERT INTO t VALUES
    ('A', 1),
    ('A', 2),
    ('A', 3),
    ('B', 4),
    ('B', 5);

SELECT gr, num,
    lead(num) OVER (PARTITION BY gr) AS lead
    FROM t ORDER BY gr, num;
```

```
+----+-----+------+
| gr | num | lead |
+----+-----+------+
| A  | 1   | 2    |
+----+-----+------+
| A  | 2   | 3    |
+----+-----+------+
```

```
| A  | 3   | null |
+----+-----+------+
| B  | 4   | 5    |
+----+-----+------+
| B  | 5   | null |
+----+-----+------+
```

```
CREATE TABLE t(gr VARCHAR, num BIGINT);

INSERT INTO t VALUES
    ('A', 1),
    ('A', 2),
    ('A', 3),
    ('B', 4),
    ('B', 5);

SELECT gr, num,
    lead(num, 1, -1) OVER (PARTITION BY gr) AS lead
    FROM t ORDER BY gr, num;
```

```
+----+-----+------+
| gr | num | lead |
+----+-----+------+
| A  | 1   | 2    |
+----+-----+------+
| A  | 2   | 3    |
+----+-----+------+
| A  | 3   | -1   |
+----+-----+------+
| B  | 4   | 5    |
+----+-----+------+
| B  | 5   | -1   |
+----+-----+------+
```

`nth_value()`

Описание	Возвращает значение, вычисленное для n-ой строки внутри группы (считая с 1).
Использование	`nth_value(expression, n[ ORDER BY column][ IGNORE NULLS])`

Если такой строки нет, возвращается значение `NULL`.

Если указано условие `ORDER BY`, значение вычисляется с использованием указанного порядка.

Параметры

- `IGNORE NULLS` — значение вычисляется без учета строк со значением `NULL`.

▼ *Посмотреть примеры*

```
CREATE TABLE t(gr VARCHAR, num BIGINT);

INSERT INTO t VALUES
    ('A', 1),
    ('A', 2),
    ('A', 3),
    ('B', 4),
    ('B', 5);

SELECT gr, num,
    nth_value(num, 3) OVER (PARTITION BY gr) AS third_value
    FROM t ORDER BY gr, num;
```

```
+----+-----+-------------+
| gr | num | third_value |
+----+-----+-------------+
| A  | 1   | 3           |
+----+-----+-------------+
| A  | 2   | 3           |
+----+-----+-------------+
| A  | 3   | 3           |
+----+-----+-------------+
| B  | 4   | null        |
+----+-----+-------------+
| B  | 5   | null        |
+----+-----+-------------+
```

`ntile()`

Описание Разбивает каждую группу на `num` подгрупп равного (насколько возможно) размера и возвращает номер подгруппы.

Использование `ntile(num[ ORDER BY ordering])`

Разбиение происходит так, чтобы размеры подгрупп были максимально близкими.

Если указано условие `ORDER BY`, значение вычисляется с использованием указанного порядка.

▼ *Посмотреть пример*

```
CREATE TABLE t(gr VARCHAR, num BIGINT);

INSERT INTO t VALUES
    ('A', 1),
    ('A', 2),
    ('A', 3),
    ('B', 4),
    ('B', 5);
```

```
SELECT gr, num,
    ntile(2) OVER (PARTITION BY gr) AS group_number
    FROM t ORDER BY gr, num;
```

```
+----+-----+--------------+
| gr | num | group_number |
+----+-----+--------------+
| A  | 1   | 1            |
+----+-----+--------------+
| A  | 2   | 1            |
+----+-----+--------------+
| A  | 3   | 2            |
+----+-----+--------------+
| B  | 4   | 1            |
+----+-----+--------------+
| B  | 5   | 2            |
+----+-----+--------------+
```

`percent_rank()`

Описание Вычисляет относительный ранг текущей строки внутри группы.

Использование ` percent_rank([ORDER BY column])`

Возвращает значение типа `DOUBLE` от `0` до `1` включительно. Относительный ранг вычисляется по формуле `(rank - 1) / (общее число строк группы - 1)`.

Если указано условие `ORDER BY`, значение вычисляется с использованием указанного порядка.

▼ *Посмотреть пример*

```
CREATE TABLE t(gr VARCHAR, num BIGINT);

INSERT INTO t VALUES
    ('A', 1),
    ('A', 2),
    ('A', 3),
    ('B', 4),
    ('B', 5);

SELECT gr, num,
    percent_rank(ORDER BY num) OVER (PARTITION BY gr) AS percent_rank,
    FROM t ORDER BY gr, num;
```

```
+----+-----+--------------+
| gr | num | percent_rank |
+----+-----+--------------+
| A  | 1   | 0            |
+----+-----+--------------+
| A  | 2   | 0.5          |
```

```
+----+-----+-------------+
| A  | 3   | 1           |
+----+-----+-------------+
| B  | 4   | 0           |
+----+-----+-------------+
| B  | 5   | 1           |
+----+-----+-------------+
```

`rank()`

Описание Возвращает ранг (с пропусками) текущей строки внутри группы.

Использование `rank([ORDER BY column])`

Если указано условие `ORDER BY`, значение вычисляется с использованием указанного порядка.

▼ *Посмотреть пример*

```
CREATE TABLE t(gr VARCHAR, num BIGINT);

INSERT INTO t VALUES
    ('A', 1),
    ('A', 2),
    ('A', 2),
    ('A', 3),
    ('B', 4),
    ('B', 5);

SELECT gr, num,
    rank(ORDER BY num) OVER (PARTITION BY gr) AS rank,
    FROM t ORDER BY gr, num;
```

```
+----+-----+------+
| gr | num | rank |
+----+-----+------+
| A  | 1   | 1    |
+----+-----+------+
| A  | 2   | 2    |
+----+-----+------+
| A  | 2   | 2    |
+----+-----+------+
| A  | 3   | 4    |
+----+-----+------+
| B  | 4   | 1    |
+----+-----+------+
| B  | 5   | 2    |
+----+-----+------+
```

`row_number()`

Описание Возвращает номер текущей строки в её группе (считая с `1`).

Использование `row_number([ORDER BY column])`

Если указано условие `ORDER BY`, значение вычисляется с использованием указанного порядка.

▼ *Посмотреть пример*

```
CREATE TABLE t(gr VARCHAR, num BIGINT);

INSERT INTO t VALUES
    ('A', 1),
    ('A', 2),
    ('A', 2),
    ('A', 3),
    ('B', 4),
    ('B', 5);

SELECT gr, num,
    row_number() OVER (PARTITION BY gr) AS row_number,
    FROM t ORDER BY gr, num;
```

```
+----+-----+------------+
| gr | num | row_number |
+----+-----+------------+
| A  | 1   | 1          |
+----+-----+------------+
| A  | 2   | 2          |
+----+-----+------------+
| A  | 2   | 3          |
+----+-----+------------+
| A  | 3   | 4          |
+----+-----+------------+
| B  | 4   | 1          |
+----+-----+------------+
| B  | 5   | 2          |
+----+-----+------------+
```

Утилиты

Утилиты — это функции для работы с данными различных типов, которые сложно отнести к конкретной категории. Их описания собраны в этом разделе.

`coalesce()`

Описание Возвращает первое значение, отличное от `NULL`, из списка значений аргументов.

Использование `coalesce(argument1[, argument2, ...])`

Если в единственном аргументе значение `NULL`, то возвращается `NULL`.

▼ *Посмотреть пример*

```
SELECT
    coalesce(NULL, 'Tengri', NULL) AS result_1,
    coalesce(NULL, '', NULL) AS result_2,
    coalesce('Tengri') AS result_3,
    coalesce(NULL) AS result_4;
```

```
+----------+----------+----------+----------+
| result_1 | result_2 | result_3 | result_4 |
+----------+----------+----------+----------+
| Tengri   |          | Tengri   | null     |
+----------+----------+----------+----------+
```

`generate_series()`

Описание Создает список значений в диапазоне между `start` и `stop`.

Использование `generate_series([start,] stop[, step])`

Параметры `start` и `stop` обрабатываются как "включительные".
Значение по умолчанию для `start` — `0`, для `step` — 1.

▼ *Посмотреть пример*

```
SELECT
    generate_series(10) AS stop,
    generate_series(5, 10) AS start_stop,
    generate_series(5, 10, 2) AS start_stop_step;
```

```
+-------------------------+----------------+-----------------+
| stop                    | start_stop     | start_stop_step |
+-------------------------+----------------+-----------------+
| {0,1,2,3,4,5,6,7,8,9,10} | {5,6,7,8,9,10} | {5,7,9}         |
+-------------------------+----------------+-----------------+
```

`hash()`

Описание Возвращает хеш данных из `argument` в виде числа.

Использование `hash(argument)`

▼ *Посмотреть пример*

```
SELECT
```

```
    hash('Tengri') AS hash;
```

```
+---------------------+
|         hash        |
+---------------------+
| 15418814193266442000 |
+---------------------+
```

`read_csv()`

Описание Читает файл `.csv` и записывает прочитанные данные в таблицу.

Использование

```
read_csv(filepath[,
    skip = 5,
    header = FALSE,
    all_varchar = TRUE]
    )
```

Читает файл `.csv` по указанному пути `filepath` и записывает прочитанные данные в таблицу. Пропуск начальных строк, приведение типов и настройки заголовков колонок могут регулироваться опциональными параметрами.

Для загрузки файлов `.csv` с локального компьютера воспользуйтесь Мастером загрузки.

Параметры

- `skip` — опциональный параметр, указывающий количество начальных строк файла, которые следует пропустить при чтении (включая первую строку!)
- `header` — опциональный параметр, указывающий на то, нужно ли интерпретировать первую строку таблицы как строку заголовков колонок (по умолчанию — `TRUE`)
- `all_varchar` — опциональный параметр; если `TRUE`, то приведение типов будет отключено и тип всех колонок будет текстовым (по умолчанию — `FALSE`)

▼ *Посмотреть примеры*

В примерах используется файл `strava_activities.csv` из сценария Анализ геоданных спортивных трекеров.

Выведем первые пять строк из файла без использования опциональных параметров функции:

```
SELECT *
    FROM read_csv('abogdanov/734b6e098011_strava_activities.csv')
    LIMIT 5
```

```
+------------+--------------------------+----------------------+--------------
+---------------------+-------------+----------+---------------+----------------
```

```
+--------+----------------------+---------------------+------------------------
+---------------+------------+----------------+-------------+------------+----------
+--------------+---------------+----------------+--------------+---------------
+-----------+--------------+-----------------------+-----------------------
+-------------+-----------------+-----------------+--------------------+----------
+--------------+----------+-----------------+--------------------+------------------
+------------+----------------+-------------+--------------+------------
+--------------------+------+------------+-----------------------+------------
+--------------------------+--------------------------+-----------+--------------------
+-------------+-------------------------+-------------------------+------------------
+--------------------+---------------------+----------+----------+-----------------
+------------+----------+--------------+-------------------------+-------------
+-------------+------------+---------+------+--------------------------
+--------------------+-------------+--------------------+----------+--------------
+------------+------------+--------------+---------+----------------------+---------------
+------------------------+-------------------------------+----------------+------------
+-------------+-------------+---------------+-----------+---------------------------
+------------+-------------+----------+----------+-------------+----------+------------
+-------+
| Activity ID | Activity Date              | Activity Name         | Activity Type |
Activity Description | Elapsed Time | Distance | Max Heart Rate | Relative Effort | Commute
| Activity Private Note | Activity Gear        | Filename                 | Athlete Weight
| Bike Weight | Elapsed Time_1 | Moving Time | Distance_1 | Max Speed | Average Speed |
Elevation Gain | Elevation Loss | Elevation Low | Elevation High | Max Grade | Average
Grade | Average Positive Grade | Average Negative Grade | Max Cadence | Average Cadence |
Max Heart Rate_1 | Average Heart Rate | Max Watts | Average Watts | Calories | Max
Temperature | Average Temperature | Relative Effort_1 | Total Work | Number of Runs |
Uphill Time | Downhill Time | Other Time | Perceived Exertion | Type | Start Time |
Weighted Average Power | Power Count | Prefer Perceived Exertion | Perceived Relative
Effort | Commute_1 | Total Weight Lifted | From Upload | Grade Adjusted Distance | Weather
Observation Time | Weather Condition | Weather Temperature | Apparent Temperature |
Dewpoint | Humidity | Weather Pressure | Wind Speed | Wind Gust | Wind Bearing |
Precipitation Intensity | Sunrise Time | Sunset Time | Moon Phase | Bike    | Gear |
Precipitation Probability | Precipitation Type | Cloud Cover | Weather Visibility | UV
Index | Weather Ozone | Jump Count | Total Grit | Average Flow | Flagged | Average Elapsed
Speed | Dirt Distance | Newly Explored Distance | Newly Explored Dirt Distance | Activity
Count | Total Steps | Carbon Saved | Pool Length | Training Load | Intensity | Average
Grade Adjusted Pace | Timer Time | Total Cycles | Recovery | With Pet | Competition | Long
Run | For a Cause | Media |
+-------------+--------------------------+----------------------+--------------
+---------------------+-------------+----------+---------------+-----------------
+---------+----------------------+---------------------+-------------------------
+---------------+------------+---------------+------------+------------+----------
+--------------+---------------+----------------+--------------+---------------
+-----------+--------------+-----------------------+-----------------------
+-------------+-----------------+-----------------+--------------------+----------
+--------------+----------+-----------------+--------------------+------------------
+------------+----------------+-------------+--------------+------------
+--------------------+------+------------+-----------------------+------------
+--------------------------+--------------------------+-----------+--------------------
+-------------+-------------------------+-------------------------+------------------
+--------------------+---------------------+----------+----------+-----------------
+------------+----------+--------------+-------------------------+-------------
```

```
+-------------+------------+---------+------+--------------------------
+--------------------+-------------+--------------------+----------+--------------
+------------+------------+-------------+---------+----------------------+--------------
+------------------------+------------------------------+----------------+------------
+-------------+-------------+--------------+-----------+----------------------------
+------------+-------------+----------+----------+-------------+----------+------------
+-------+
| 187782986   | Aug 31, 2014, 10:54:43 AM | Из дома к родителям   | Ride          | null
| 2462         | 19.41    | null           | null            | false   | null
| Merida S-Presso 900d | activities/187782986.gpx | 83             | 11          | 2462
| 2286        | 19415.4    | 13.5      | null          | 158.2          | null           |
118.2         | 206            | 12        | 0.3           | null                    | null
| null        | null           | null             | null               | null      | 192
| null     | null            | null                | null              | null       | null
| null        | null          | null       | null               | null | null       | null
| null        | null                        | null                      | 0         | null
| null        | null                     | null                     | null             |
null                | null                | null     | null     | null             | null
| null      | null         | null                     | null         | null        | null
| 1512511 | null | null                       | null               | null        | null
| null     | null          | null       | null       | null         | null    | null
| null          | null                     | null                          | null           |
null        | null         | null        | null          | null      | null
| null       | null         | null     | null     | null        | null     | null        |
null  |
+-------------+-------------------------+---------------------+--------------
+---------------------+-------------+----------+---------------+----------------
+---------+----------------------+---------------------+-------------------------
+----------------+-------------+----------------+-------------+------------+-----------
+---------------+----------------+----------------+--------------+----------------
+-----------+--------------+------------------------+-----------------------
+-------------+-----------------+------------------+--------------------+-----------
+--------------+----------+-----------------+--------------------+------------------
+------------+---------------+-------------+--------------+------------
+--------------------+------+------------+----------------------+-------------
+--------------------------+--------------------------+----------+--------------------
+-------------+------------------------+-------------------------+------------------
+--------------------+---------------------+----------+----------+-----------------
+------------+-----------+--------------+------------------------+--------------
+-------------+------------+---------+------+---------------------------
+-------------------+-------------+--------------------+----------+--------------
+------------+------------+-------------+---------+----------------------+--------------
+------------------------+------------------------------+----------------+------------
+--------------+-------------+---------------+-----------+----------------------------
+------------+-------------+----------+----------+-------------+----------+------------
+-------+
| 187840092   | Aug 31, 2014, 12:48:49 PM | Послеобеденный заезд  | Ride          | null
| 2396         | 15.9     | null           | null            | false   | null
| null                 | activities/187840092.gpx | 83             | null        | 2396
| 1920        | 15906.6    | 13.4      | null          | 93.9           | null           |
122.5         | 214.6          | 8.8       | -0.5          | null                    | null
| null        | null           | null             | null               | null      | 178
| null     | null            | null                | null              | null       | null
```

```
| null        | null          | null       | null               | null | null       | null
| null        | null                      | null                     | 0         | null
| null        | null                    | null                       | null              |
null                | null                  | null     | null     | null             | null
| null      | null         | null                     | null         | null        | null
| null    | null | null                       | null               | null        | null
| null     | null          | null       | null       | null         | null    | null
| null          | null                    | null                         | null           |
null        | null         | null        | null          | null      | null
| null       | null         | null     | null     | null        | null     | null        |
null  |
+-------------+-------------------------+----------------------+--------------
+----------------------+-------------+----------+----------------+-----------------
+---------+----------------------+----------------------+-------------------------
+---------------+-------------+---------------+-------------+-----------+----------
+--------------+----------------+---------------+--------------+---------------
+-----------+--------------+-----------------------+----------------------
+-------------+-----------------+------------------+--------------------+----------
+--------------+----------+-----------------+--------------------+------------------
+------------+---------------+-------------+--------------+-----------
+--------------------+------+-----------+----------------------+------------
+--------------------------+--------------------------+----------+--------------------
+-------------+-----------------------+--------------------------+------------------
+---------------------+---------------------+----------+----------+-----------------
+------------+-----------+--------------+-----------------------+-------------
+-------------+------------+---------+------+--------------------------
+--------------------+-------------+--------------------+----------+--------------
+------------+------------+-------------+---------+----------------------+--------------
+------------------------+-------------------------------+----------------+-------------
+-------------+-------------+--------------+----------+---------------------------
+------------+-------------+----------+----------+-------------+----------+-------------
+-------+
| 187844596   | Aug 31, 2014, 1:29:23 PM  | Послеобеденный заезд  | Ride          | null
| 103          | 0.7      | null           | null            | false   | null
| null                | activities/187844596.gpx | 83             | null        | 103
| 103         | 708.4      | 8.4       | null          | 0             | null          |
127.4         | 136            | 3.7       | 1.2           | null                  | null
| null        | null           | null             | null               | null      | 172
| null     | null           | null               | null              | null       | null
| null        | null          | null       | null               | null | null       | null
| null        | null                      | null                     | 0         | null
| null        | null                    | null                       | null              |
null                | null                  | null     | null     | null             | null
| null      | null         | null                     | null         | null        | null
| null    | null | null                       | null               | null        | null
| null     | null          | null       | null       | null         | null    | null
| null          | null                    | null                         | null           |
null        | null         | null        | null          | null      | null
| null       | null         | null     | null     | null        | null     | null        |
null  |
+-------------+-------------------------+----------------------+--------------
+----------------------+-------------+----------+----------------+-----------------
+---------+----------------------+----------------------+-------------------------
```

```
+---------------+------------+---------------+------------+-----------+----------
+--------------+---------------+---------------+--------------+---------------
+----------+--------------+----------------------+----------------------
+-------------+-----------------+-----------------+--------------------+----------
+--------------+----------+----------------+--------------------+------------------
+------------+---------------+------------+--------------+------------
+--------------------+------+-----------+----------------------+------------
+--------------------------+--------------------------+----------+--------------------
+-------------+-------------------------+-------------------------+-------------------
+--------------------+---------------------+----------+----------+-----------------
+------------+----------+--------------+-----------------------+-------------
+-------------+-----------+---------+------+--------------------------
+--------------------+------------+-------------------+----------+--------------
+------------+-----------+-------------+---------+----------------------+---------------
+-------------------------+-------------------------------+---------------+------------
+-------------+-------------+--------------+----------+--------------------------
+------------+-------------+----------+----------+------------+----------+------------
+-------+
| 187850580   | Aug 31, 2014, 1:37:59 PM  | Крылатское малый круг | Ride          | null
| 561          | 4.41     | null           | null            | false   | null
| null                 | activities/187850580.gpx | 83             | null        | 561
| 561         | 4414.1     | 13.6      | null          | 46.7           | null           |
137.6         | 179.4          | 10.4      | -0.1          | null                    | null
| null        | null            | null             | null               | null      | 192
| 120      | null            | null               | null               | null       | null
| null        | null          | null       | null               | null | null       | null
| null        | null                       | null                      | 0         | null
| null        | null                    | null                    | null             |
null                | null                | null     | null     | null             | null
| null      | null         | null                    | null         | null        | null
| null    | null | null                      | null              | null        | null
| null     | null          | null       | null       | null         | null    | null
| null          | null                    | null                         | null           |
null        | null         | null        | null          | null      | null
| null       | null         | null     | null     | null        | null     | null        |
null  |
+-------------+-------------------------+---------------------+--------------
+--------------------+-------------+----------+---------------+----------------
+---------+----------------------+---------------------+-------------------------
+---------------+-------------+----------------+-------------+-----------+----------
+---------------+---------------+---------------+--------------+---------------
+-----------+--------------+-----------------------+----------------------
+-------------+-----------------+-----------------+--------------------+----------
+--------------+----------+----------------+--------------------+------------------
+------------+---------------+------------+--------------+------------
+--------------------+------+-----------+----------------------+------------
+--------------------------+--------------------------+----------+--------------------
+-------------+-------------------------+-------------------------+-------------------
+--------------------+---------------------+----------+----------+-----------------
+------------+-----------+-------------+-------------------------+-------------
+-------------+-----------+---------+------+--------------------------
+-------------------+-------------+--------------------+----------+--------------
+------------+-----------+-------------+---------+----------------------+--------------
```

```
+------------------------+-----------------------------+---------------+------------
+--------------+-------------+---------------+-----------+---------------------------
+------------+-------------+----------+----------+-------------+----------+------------
+-------+
| 187867604   | Aug 31, 2014, 1:59:58 PM  | Послеобеденный заезд  | Ride          | null
| 886          | 5.03     | null           | null            | false   | null
| null                 | activities/187867604.gpx | 83             | null        | 886
| 820          | 5036.3     | 10.6      | null          | 34.6           | null          |
123           | 156.7          | 7.3       | 0.3           | null                   | null
| null        | null           | null             | null               | null      | 127
| null     | null           | null                | null              | null       | null
| null        | null          | null       | null               | null | null       | null
| null        | null                       | null                     | 0         | null
| null        | null                     | null                     | null              |
null               | null                | null     | null     | null             | null
| null      | null         | null                    | null         | null        | null
| null    | null | null                      | null              | null        | null
| null     | null          | null       | null       | null         | null    | null
| null          | null                     | null                          | null          |
null        | null         | null        | null          | null      | null
| null       | null         | null     | null     | null        | null     | null        |
null  |
+-------------+-------------------------+----------------------+--------------
+----------------------+-------------+----------+----------------+-----------------
+---------+-----------------------+----------------------+-------------------------
+---------------+-------------+---------------+-------------+-----------+----------
+---------------+----------------+---------------+--------------+---------------
+-----------+--------------+------------------------+-----------------------
+-------------+-----------------+-----------------+--------------------+----------
+--------------+----------+-----------------+--------------------+------------------
+------------+---------------+-------------+--------------+-----------
+--------------------+------+-----------+----------------------+-------------
+-------------------------+--------------------------+----------+--------------------
+-------------+------------------------+-------------------------+------------------
+--------------------+---------------------+----------+----------+-----------------
+------------+----------+--------------+-----------------------+-------------
+-------------+-----------+---------+------+--------------------------
+------------------+-------------+--------------------+----------+--------------
+------------+------------+-------------+---------+----------------------+--------------
+------------------------+-----------------------------+---------------+-------------
+--------------+-------------+---------------+-----------+---------------------------
+------------+-------------+----------+----------+-------------+----------+------------
+-------+
```

Выведем первые пять строк из файла `.xlsx`, игнорируя строку заголовков:

```
SELECT *
    FROM read_csv('abogdanov/734b6e098011_strava_activities.csv',
        header = FALSE
        )
    LIMIT 5
```

```
+-------------+--------------------------+----------------------+--------------
+---------------------+--------------+----------+----------------+----------------
+----------+----------------------+---------------------+-------------------------
+----------------+-------------+--------------+-------------+----------+-----------
+--------------+----------------+----------------+--------------+---------------
+-----------+---------------+-------------------------+-----------------------
+-------------+-----------------+----------------+--------------------+-----------
+--------------+----------+-----------------+---------------------+-----------------
+------------+----------------+-------------+---------------+------------
+--------------------+----------+------------+-----------------------+-------------
+---------------------------+----------------------------+----------+---------------------
+-------------+-------------------------+--------------------------+-------------------
+---------------------+---------------------+----------+----------+-----------------
+------------+-----------+--------------+-------------------------+--------------
+-------------+------------+-----------+----------+---------------------------
+--------------------+-------------+--------------------+----------+---------------
+------------+------------+--------------+----------+----------------------
+--------------+-------------------------+-------------------------------+----------------
+-------------+--------------+-------------+---------------+-----------
+------------------------------+------------+--------------+----------+----------
+-------------+----------+-------------+----------+
| column00    | column01                  | column02              | column03      |
column04             | column05     | column06 | column07       | column08        |
column09 | column10              | column11             | column12                  |
column13       | column14    | column15     | column16    | column17 | column18  | column19
| column20       | column21       | column22      | column23       | column24  | column25
| column26               | column27               | column28    | column29        |
column30       | column31           | column32  | column33      | column34 | column35
| column36            | column37        | column38   | column39       | column40    |
column41      | column42   | column43           | column44 | column45   | column46
| column47    | column48                  | column49                  | column50 | column51
| column52    | column53               | column54                  | column55          |
column56            | column57             | column58 | column59 | column60         |
column61   | column62  | column63     | column64                | column65     | column66
| column67   | column68  | column69 | column70                  | column71           |
column72    | column73           | column74 | column75      | column76   | column77   |
column78     | column79 | column80              | column81      | column82                |
column83                     | column84       | column85    | column86     | column87    |
column88      | column89  | column90                     | column91   | column92     |
column93 | column94 | column95    | column96 | column97    | column98 |
+-------------+--------------------------+----------------------+--------------
+---------------------+-------------+----------+----------------+----------------
+----------+----------------------+---------------------+-------------------------
+----------------+-------------+--------------+-------------+----------+-----------
+--------------+----------------+----------------+--------------+---------------
+-----------+---------------+-------------------------+-----------------------
+-------------+-----------------+----------------+--------------------+-----------
+--------------+----------+-----------------+---------------------+-----------------
+------------+----------------+-------------+---------------+------------
+--------------------+----------+------------+-----------------------+-------------
+---------------------------+----------------------------+----------+---------------------
+-------------+-------------------------+--------------------------+-------------------
```

```
+--------------------+---------------------+----------+----------+-----------------
+------------+----------+-------------+------------------------+-------------
+------------+-----------+-----------+----------+--------------------------
+--------------------+-------------+--------------------+----------+--------------
+------------+-----------+-------------+----------+----------------------
+--------------+------------------------+------------------------------+---------------
+-------------+-------------+-------------+--------------+-----------
+----------------------------+-----------+-------------+----------+----------
+-------------+----------+-------------+----------+
| Activity ID | Activity Date              | Activity Name         | Activity Type |
Activity Description | Elapsed Time | Distance | Max Heart Rate | Relative Effort | Commute
| Activity Private Note | Activity Gear        | Filename                 | Athlete Weight
| Bike Weight | Elapsed Time | Moving Time | Distance | Max Speed | Average Speed |
Elevation Gain | Elevation Loss | Elevation Low | Elevation High | Max Grade | Average
Grade | Average Positive Grade | Average Negative Grade | Max Cadence | Average Cadence |
Max Heart Rate | Average Heart Rate | Max Watts | Average Watts | Calories | Max
Temperature | Average Temperature | Relative Effort | Total Work | Number of Runs | Uphill
Time | Downhill Time | Other Time | Perceived Exertion | Type     | Start Time | Weighted
Average Power | Power Count | Prefer Perceived Exertion | Perceived Relative Effort |
Commute  | Total Weight Lifted | From Upload | Grade Adjusted Distance | Weather
Observation Time | Weather Condition | Weather Temperature | Apparent Temperature |
Dewpoint | Humidity | Weather Pressure | Wind Speed | Wind Gust | Wind Bearing |
Precipitation Intensity | Sunrise Time | Sunset Time | Moon Phase | Bike      | Gear     |
Precipitation Probability | Precipitation Type | Cloud Cover | Weather Visibility | UV
Index | Weather Ozone | Jump Count | Total Grit | Average Flow | Flagged  | Average Elapsed
Speed | Dirt Distance | Newly Explored Distance | Newly Explored Dirt Distance | Activity
Count | Total Steps | Carbon Saved | Pool Length | Training Load | Intensity | Average
Grade Adjusted Pace | Timer Time | Total Cycles | Recovery | With Pet | Competition | Long
Run | For a Cause | Media    |
+-------------+--------------------------+----------------------+--------------
+----------------------+-------------+----------+---------------+-----------------
+----------+----------------------+---------------------+-------------------------
+----------------+-------------+-------------+-------------+----------+-----------
+---------------+----------------+----------------+---------------+----------------
+-----------+---------------+------------------------+----------------------
+-------------+-----------------+----------------+--------------------+-----------
+---------------+----------+-----------------+--------------------+-----------------
+------------+----------------+-------------+--------------+------------
+--------------------+----------+------------+-------------------------+-------------
+---------------------------+---------------------------+----------+---------------------
+-------------+-------------------------+-------------------------+--------------------
+---------------------+---------------------+----------+----------+------------------
+------------+-----------+-------------+-------------------------+--------------
+-------------+------------+-----------+----------+---------------------------
+--------------------+-------------+--------------------+----------+--------------
+------------+------------+-------------+----------+----------------------
+--------------+-------------------------+------------------------------+---------------
+-------------+-------------+-------------+--------------+-----------
+----------------------------+-----------+-------------+----------+----------
+-------------+----------+-------------+----------+
| 187782986   | Aug 31, 2014, 10:54:43 AM | Из дома к родителям   | Ride          | null
| 2462         | 19.41    | null           | null            | false    | null
| Merida S-Presso 900d | activities/187782986.gpx | 83.0           | 11.0        | 2462.0
```

```
| 2286.0      | 19415.4  | 13.5      | null          | 158.2          | null           |
118.2         | 206.0          | 12.0      | 0.3           | null                    | null
| null        | null            | null           | null               | null      | 192.0
| null     | null            | null                | null            | null       | null
| null        | null          | null       | null               | null     | null       |
null                  | null        | null                       | null
| 0.0      | null                | null        | null                    | null
| null              | null                | null                 | null     | null     |
null             | null       | null      | null         | null                    | null
| null        | null       | 1512511.0 | null     | null                    | null
| null        | null               | null     | null          | null       | null       |
null         | null     | null                  | null          | null                    |
null                         | null           | null        | null         | null        |
null          | null      | null                         | null       | null         | null
| null     | null        | null     | null        | null     |
+-------------+--------------------------+----------------------+--------------
+---------------------+-------------+----------+---------------+----------------
+----------+----------------------+---------------------+-------------------------
+----------------+-------------+--------------+-------------+----------+-----------
+---------------+----------------+----------------+--------------+---------------
+-----------+---------------+-------------------------+-----------------------
+-------------+-----------------+----------------+--------------------+-----------
+---------------+----------+-----------------+---------------------+-----------------
+------------+----------------+-------------+---------------+------------
+--------------------+----------+------------+------------------------+-------------
+---------------------------+---------------------------+----------+---------------------
+-------------+-------------------------+--------------------------+-------------------
+---------------------+----------------------+----------+----------+------------------
+------------+-----------+--------------+-------------------------+--------------
+-------------+------------+-----------+----------+---------------------------
+--------------------+-------------+--------------------+----------+--------------
+------------+------------+--------------+----------+----------------------
+--------------+-------------------------+-------------------------------+----------------
+-------------+--------------+-------------+---------------+-----------
+-----------------------------+------------+--------------+----------+----------
+-------------+----------+-------------+----------+
| 187840092   | Aug 31, 2014, 12:48:49 PM | Послеобеденный заезд  | Ride          | null
| 2396         | 15.90    | null           | null            | false    | null
| null                 | activities/187840092.gpx | 83.0           | null        | 2396.0
| 1920.0      | 15906.6  | 13.4      | null          | 93.9           | null           |
122.5         | 214.6          | 8.8       | -0.5          | null                    | null
| null        | null            | null           | null               | null      | 178.0
| null     | null            | null                | null            | null       | null
| null        | null          | null       | null               | null     | null       |
null                  | null        | null                       | null
| 0.0      | null                | null        | null                    | null
| null              | null                | null                 | null     | null     |
null             | null       | null      | null         | null                    | null
| null        | null       | null      | null     | null                    | null
| null        | null               | null     | null          | null       | null       |
null         | null     | null                  | null          | null                    |
null                         | null           | null        | null         | null        |
null          | null      | null                         | null       | null         | null
| null     | null        | null     | null        | null     |
```

```
| null     | null        | null     | null        | null     |
+-------------+---------------------------+---------------------+--------------
+---------------------+-------------+----------+----------------+----------------
+----------+----------------------+---------------------+-------------------------
+---------------+-------------+-------------+-------------+----------+-----------
+--------------+----------------+----------------+--------------+---------------
+-----------+---------------+------------------------+-----------------------
+-------------+-----------------+----------------+--------------------+-----------
+---------------+----------+-----------------+--------------------+-----------------
+------------+---------------+-------------+--------------+------------
+--------------------+----------+------------+-----------------------+-------------
+---------------------------+---------------------------+----------+--------------------
+-------------+------------------------+-------------------------+-------------------
+---------------------+----------------------+----------+----------+-----------------
+------------+-----------+-------------+-----------------------+-------------
+-------------+------------+-----------+----------+--------------------------
+--------------------+-------------+--------------------+----------+--------------
+------------+------------+-------------+----------+----------------------
+--------------+-------------------------+-----------------------------+----------------
+-------------+-------------+-------------+---------------+-----------
+------------------------------+------------+-------------+----------+----------
+-------------+----------+-------------+----------+
| 187844596   | Aug 31, 2014, 1:29:23 PM  | Послеобеденный заезд  | Ride          | null
| 103          | 0.70     | null           | null            | false    | null
| null                | activities/187844596.gpx | 83.0           | null        | 103.0
| 103.0       | 708.4    | 8.4       | null          | 0.0            | null           |
127.4         | 136.0          | 3.7       | 1.2           | null                    | null
| null        | null            | null           | null               | null      | 172.0
| null     | null            | null                | null            | null       | null
| null        | null          | null       | null               | null     | null       |
null                  | null        | null                       | null
| 0.0      | null               | null        | null                    | null
| null              | null               | null                | null     | null     |
null             | null       | null      | null         | null                     | null
| null        | null       | null      | null     | null                       | null
| null        | null               | null     | null          | null       | null       |
null         | null     | null                  | null          | null                     |
null                         | null           | null        | null         | null        |
null          | null      | null                          | null       | null         | null
| null     | null        | null     | null        | null     |
+-------------+---------------------------+---------------------+--------------
+---------------------+-------------+----------+----------------+----------------
+----------+----------------------+---------------------+-------------------------
+---------------+-------------+-------------+-------------+----------+-----------
+--------------+----------------+----------------+--------------+---------------
+-----------+---------------+------------------------+-----------------------
+-------------+-----------------+----------------+--------------------+-----------
+---------------+----------+-----------------+--------------------+-----------------
+------------+---------------+-------------+--------------+------------
+--------------------+----------+------------+-----------------------+-------------
+---------------------------+---------------------------+----------+--------------------
+-------------+------------------------+-------------------------+-------------------
+---------------------+----------------------+----------+----------+-----------------
```

```
+-----------+----------+-------------+-----------------------+-------------
+-------------+-----------+----------+----------+--------------------------
+-------------------+------------+-------------------+----------+--------------
+-----------+-----------+-------------+----------+----------------------
+--------------+------------------------+----------------------------+---------------
+-------------+-------------+------------+--------------+-----------
+----------------------------+-----------+-------------+----------+----------
+-------------+----------+------------+----------+
| 187850580   | Aug 31, 2014, 1:37:59 PM  | Крылатское малый круг | Ride          | null
| 561          | 4.41     | null           | null            | false    | null
| null                 | activities/187850580.gpx | 83.0           | null        | 561.0
| 561.0       | 4414.1   | 13.6      | null          | 46.7           | null           |
137.6         | 179.4          | 10.4      | -0.1          | null                    | null
| null        | null            | null           | null               | null      | 192.0
| 120.0    | null            | null               | null            | null       | null
| null        | null          | null       | null               | null     | null       |
null                  | null        | null                       | null
| 0.0      | null                | null        | null                    | null
| null              | null                | null                 | null     | null     |
null             | null       | null      | null         | null                     | null
| null        | null       | null      | null     | null                       | null
| null        | null               | null     | null          | null       | null       |
null         | null     | null                  | null          | null                    |
null                         | null           | null        | null         | null        |
null          | null      | null                         | null       | null         | null
| null     | null        | null     | null        | null     |
+-------------+--------------------------+----------------------+--------------
+----------------------+-------------+----------+----------------+-----------------
+----------+----------------------+----------------------+--------------------------
+----------------+-------------+--------------+-------------+----------+-----------
+---------------+----------------+----------------+---------------+----------------
+-----------+---------------+------------------------+-----------------------
+-------------+-----------------+----------------+--------------------+-----------
+---------------+----------+-----------------+---------------------+-----------------
+------------+----------------+-------------+---------------+------------
+--------------------+----------+------------+-----------------------+-------------
+---------------------------+---------------------------+----------+---------------------
+-------------+------------------------+--------------------------+-------------------
+---------------------+----------------------+----------+----------+-------------------
+------------+-----------+--------------+-------------------------+--------------
+-------------+------------+-----------+----------+--------------------------
+-------------------+-------------+--------------------+----------+--------------
+------------+------------+--------------+----------+----------------------
+---------------+------------------------+-----------------------------+----------------
+-------------+--------------+-------------+---------------+-----------
+-----------------------------+------------+--------------+----------+----------
+-------------+----------+-------------+----------+
```

`read_parquet()`

Описание Читает файл `.parquet` и записывает прочитанные данные в таблицу.

Использование `read_parquet(filepath)`

Читает файл `.parquet` по указанному пути `filepath` и записывает прочитанные данные в таблицу.

Для загрузки файлов `.parquet` с локального компьютера воспользуйтесь Мастером загрузки.

▼ *Посмотреть пример*

Загрузим через Мастер загрузки один из файлов с данными отсюда и прочитаем его:

```
SELECT *
    FROM read_parquet('abogdanov/05e4fddda22d_green_tripdata_2025-01.parquet')
    LIMIT 5
```

```
+----------+----------------------+----------------------+-------------------
+------------+-------------+-------------+-----------------+--------------+-------------
+-------+---------+------------+-------------+-----------+----------------------
+-------------+-------------+-----------+----------------------+-------------------+
| VendorID | lpep_pickup_datetime | lpep_dropoff_datetime | store_and_fwd_flag | RatecodeID
| PULocationID | DOLocationID | passenger_count | trip_distance | fare_amount | extra |
mta_tax | tip_amount | tolls_amount | ehail_fee | improvement_surcharge | total_amount |
payment_type | trip_type | congestion_surcharge | cbd_congestion_fee |
+----------+----------------------+----------------------+-------------------
+------------+-------------+-------------+-----------------+--------------+-------------
+-------+---------+------------+-------------+-----------+----------------------
+-------------+-------------+-----------+----------------------+-------------------+
| 2        | 2025-01-01 00:03:01  | 2025-01-01 00:17:12   | N                  | 1
| 75           | 235          | 1               | 5.93          | 24.7        | 1     | 0.5
| 6.8        | 0            | null      | 1                    | 34           | 1
| 1         | 0                    | 0                 |
+----------+----------------------+----------------------+-------------------
+------------+-------------+-------------+-----------------+--------------+-------------
+-------+---------+------------+-------------+-----------+----------------------
+-------------+-------------+-----------+----------------------+-------------------+
| 2        | 2025-01-01 00:19:59  | 2025-01-01 00:25:52   | N                  | 1
| 166          | 75           | 1               | 1.32          | 8.6         | 1     | 0.5
| 0          | 0            | null      | 1                    | 11.1         | 2
| 1         | 0                    | 0                 |
+----------+----------------------+----------------------+-------------------
+------------+-------------+-------------+-----------------+--------------+-------------
+-------+---------+------------+-------------+-----------+----------------------
+-------------+-------------+-----------+----------------------+-------------------+
| 2        | 2025-01-01 00:05:29  | 2025-01-01 00:07:21   | N                  | 5
| 171          | 73           | 1               | 0.41          | 25.55       | 0     | 0
| 0          | 0            | null      | 1                    | 26.55        | 2
| 2         | 0                    | 0                 |
+----------+----------------------+----------------------+-------------------
+------------+-------------+-------------+-----------------+--------------+-------------
+-------+---------+------------+-------------+-----------+----------------------
+-------------+-------------+-----------+----------------------+-------------------+
| 2        | 2025-01-01 00:52:24  | 2025-01-01 01:07:52   | N                  | 1
| 74           | 223          | 1               | 4.12          | 21.2        | 1     | 0.5
```

```
| 6.13       | 6.94         | null      | 1                    | 36.77        | 1
| 1          | 0                    | 0                  |
+----------+---------------------+---------------------+------------------
+------------+-------------+-------------+----------------+--------------+-------------
+-------+---------+-----------+-------------+----------+---------------------
+-------------+-------------+----------+---------------------+--------------------+
| 2        | 2025-01-01 00:25:05  | 2025-01-01 01:01:10   | N                   | 1
| 66           | 158          | 1               | 4.71          | 33.8        | 1     | 0.5
| 7.81       | 0            | null      | 1                    | 46.86        | 1
| 1          | 2.75                 | 0                  |
+----------+---------------------+---------------------+------------------
+------------+-------------+-------------+----------------+--------------+-------------
+-------+---------+-----------+-------------+----------+---------------------
+-------------+-------------+----------+---------------------+--------------------+
```

`read_xlsx()`

Описание Читает файл `.xlsx` и записывает прочитанные данные в таблицу.

Использование

```
read_xlsx(filepath[,
    sheet = 'Лист1',
    range = 'A5:E100',
    stop_at_empty = TRUE,
    header = TRUE,
    all_varchar = TRUE]
    )
```

Читает файл `.xlsx` по указанному пути `filepath` и записывает прочитанные данные в таблицу. Если указан диапазон для чтения или лист файла, то выбирает только указанные данные. Приведение типов и настройки заголовков колонок могут регулироваться опциональными параметрами.

Для загрузки файлов `.xlsx` с локального компьютера воспользуйтесь Мастером загрузки.

Параметры

- `sheet` — опциональный параметр, указывающий имя листа файла `.xlsx` для чтения. По умолчанию берется первый лист.
- `range` — опциональный параметр, указывающий диапазон клеток таблицы файла `.xlsx` для чтения. Диапазон указывается от левой верхней клетки до правой нижней, например: `range = 'A5:E100'`.
  Возможно нестрогое указание диапазона, например:
  - `range = 'A:E100'` — выберет клетки от первой в колонке `A` до `E100`
  - `range = 'B:E'` — выберет клетки от первой в колонке `B` до последней в колонке `E` (в этом случае рекомендуется использовать ограничение `stop_at_empty = TRUE`, см. пример ниже)
- `stop_at_empty` — опциональный параметр, указывающий на то, нужно ли прервать чтение на первой пустой строке таблицы
- `header` — опциональный параметр, указывающий на то, нужно ли интерпретировать первую

строку таблицы как строку заголовков колонок (по умолчанию — `TRUE`, если в первой строке нет пустых клеток)

- `all_varchar` — опциональный параметр; если `TRUE`, то приведение типов будет отключено и тип всех колонок будет текстовым (по умолчанию — `FALSE`)

▼ *Посмотреть примеры*

В примерах используется файл `instagram_example_result_1.xlsx` из сценария Анализ данных из постов Instagram*.

Выведем первые пять строк из файла без использования опциональных параметров функции:

```
SELECT *
    FROM read_xlsx('abogdanov/b95f3e020363_instagram_example_result_1.xlsx')
    LIMIT 5
```

```
+------------------+-------+----------+----------+--------------+------------
+--------------+
| artist_name      | songs | ig_posts | comments | tagged_users | likescount |
repliescount |
+------------------+-------+----------+----------+--------------+------------
+--------------+
| "telfarglobal"   | 9     | 136      | 985      | 120          | 634        | 156
|
+------------------+-------+----------+----------+--------------+------------
+--------------+
| "everlane"       | 1     | 95       | 532      | 6            | 513        | 65
|
+------------------+-------+----------+----------+--------------+------------
+--------------+
| "asos"           | 3     | 93       | 503      | 60           | 583        | 85
|
+------------------+-------+----------+----------+--------------+------------
+--------------+
| "victoriassecret" | 1     | 83       | 686      | 49           | 185        | 16
|
+------------------+-------+----------+----------+--------------+------------
+--------------+
| "summerfridays"  | 2     | 65       | 574      | 48           | 173        | 18
|
+------------------+-------+----------+----------+--------------+------------
+--------------+
```

Выведем первые пять строк первых четырех колонок из файла `.xlsx`:

```
SELECT *
    FROM read_xlsx('abogdanov/b95f3e020363_instagram_example_result_1.xlsx',
        range = 'A1:D6'
        )
```

```
+------------------+-------+----------+----------+
| artist_name      | songs | ig_posts | comments |
+------------------+-------+----------+----------+
| "telfarglobal"   | 9     | 136      | 985      |
+------------------+-------+----------+----------+
| "everlane"       | 1     | 95       | 532      |
+------------------+-------+----------+----------+
| "asos"           | 3     | 93       | 503      |
+------------------+-------+----------+----------+
| "victoriassecret" | 1     | 83       | 686      |
+------------------+-------+----------+----------+
| "summerfridays"  | 2     | 65       | 574      |
+------------------+-------+----------+----------+
```

Выведем колонки с `A` по `C` из файла `.xlsx`, для этого используем диапазон по именам колонок и параметр `stop_at_empty` (иначе у диапазона не будет явной правой границы, так как не указана конечная клетка диапазона):

```
SELECT *
    FROM read_xlsx('abogdanov/b95f3e020363_instagram_example_result_1.xlsx',
        range = 'A:C',
        stop_at_empty = TRUE
        )
```

```
+-----------------------------------------------------------------------+-------+----------+
| artist_name                                                           | songs | ig_posts |
+-----------------------------------------------------------------------+-------+----------+
| "telfarglobal"                                                        | 9     | 136      |
+-----------------------------------------------------------------------+-------+----------+
| "everlane"                                                            | 1     | 95       |
+-----------------------------------------------------------------------+-------+----------+
| "asos"                                                                | 3     | 93       |
+-----------------------------------------------------------------------+-------+----------+
| "victoriassecret"                                                     | 1     | 83       |
+-----------------------------------------------------------------------+-------+----------+
| "summerfridays"                                                       | 2     | 65       |
+-----------------------------------------------------------------------+-------+----------+
| ...                                                                   | ...   | ...      |
+-----------------------------------------------------------------------+-------+----------+
226 rows
```

`cast()`

Описание Приводит тип данных к указанному.

Использование `cast(argument AS DATA_TYPE)`

Приводит тип данных аргумента к указанному. В случае невозможности приведения выдает ошибку.

См. также `try_cast()`, Оператор `::`.

▼ *Посмотреть примеры*

- ```
  SELECT
      cast('1' AS BIGINT)                        AS result_1,
      cast('00:00:00.0' AS TIME)                 AS result_2,
      cast('2025-12-03T00:00:00.0' AS TIMESTAMP) AS result_3;
  ```

  ```
  +----------+----------+---------------------+
  | result_1 | result_2 | result_3            |
  +----------+----------+---------------------+
  | 1        | 00:00:00 | 2025-12-03 00:00:00 |
  +----------+----------+---------------------+
  ```

- ```
  SELECT
      cast('Tengri' AS BIGINT); -- error expected
  ```

  ```
  ERROR: ConversionException: Conversion Error: Could not convert string 'Tengri' to INT64
  ```

`try_cast()`

Описание Приводит тип данных к указанному.

Использование `try_cast(argument AS DATA_TYPE)`

Приводит тип данных аргумента к указанному. В случае невозможности приведения возвращает `NULL`.

См. также `cast()`, Оператор `::`.

▼ *Посмотреть примеры*

```
SELECT
    try_cast('1' AS BIGINT)                        AS result_1,
    try_cast('00:00:00.0' AS TIME)                 AS result_2,
    try_cast('2025-01-01T00:00:00.0' AS TIMESTAMP) AS result_3,
    try_cast('Tengri' AS BIGINT)                   AS result_4, -- null expected
    try_cast('00:00:00.0' AS TIMESTAMP)            AS result_5; -- null expected
```

```
+----------+----------+---------------------+----------+----------+
| result_1 | result_2 | result_3            | result_4 | result_5 |
+----------+----------+---------------------+----------+----------+
| 1        | 00:00:00 | 2025-01-01 00:00:00 | null     | null     |
+----------+----------+---------------------+----------+----------+
```

`unnest()`

Описание	Разворачивает списки или структуры из `argument` в множество отдельных значений.
Использование	`unnest(argument) [, recursive := true] [, max_depth := <num>]`

Применение функции к списку дает одну строку на каждый элемент списка. Обычные скалярные выражения в том же выражении `SELECT` повторяются для каждой выводимой строки.

Когда несколько списков разворачиваются в одном выражении `SELECT`, они разворачиваются каждый в отдельный столбец. Если один список длиннее другого, более короткий список заполняется значениями `NULL`.

Функция изменяет кардинальность данных.

Параметры

- `recursive := true`
  Включает рекурсивный режим. Если этот режим включен (значение `true`), то функция полностью разворачивает списки, а затем полностью разворачивает вложенные структуры. Это может быть полезно для полного "уплощения" столбцов, которые содержат списки внутри списков или структуры внутри списков. Обратите внимание, что списки внутри структур не разворачиваются.

  ▼ *Подробнее о параметре на примерах*

  Покажем работу этого параметра на двух примерах с одними и теми же данными с включенным параметром и без него:

```
SELECT
    unnest([[1, 2, 3], [4, 5]], recursive := true) AS result;
```

```
+--------+
| result |
+--------+
| 1      |
+--------+
| 2      |
+--------+
| 3      |
+--------+
| 4      |
+--------+
| 5      |
+--------+
```

```
SELECT
```

```
    unnest([[1, 2, 3], [4, 5]]) AS result;
```

```
+---------+
|  result |
+---------+
| {1,2,3} |
+---------+
| {4,5}   |
+---------+
```

- `max_depth := <num>`
  Параметр `max_depth` позволяет ограничить максимальную глубину рекурсивного развертывания. Рекурсивный режим автоматически включен, если указана максимальная глубина.

  ▼ *Подробнее о параметре на примерах*

  Покажем работу этого параметра на трех примерах с одними и теми же данными: с глубиной развертывания по умолчанию (1), с глубиной развертывания `2` и с глубиной развертывания `3`:

```
SELECT
    unnest([[['T', 'e'], ['n', 'g']], [['r', 'i'], []], [['!', '!', '!']]])
AS result;
```

```
+-------------------------+
|          result         |
+-------------------------+
| {['T', 'e'],['n', 'g']} |
+-------------------------+
| {['r', 'i'],[]}         |
+-------------------------+
| {['!', '!', '!']}       |
+-------------------------+
```

```
SELECT
    unnest([[['T', 'e'], ['n', 'g']], [['r', 'i'], []], [['!', '!', '!']]],
    max_depth := 2)
AS result;
```

```
+---------+
|  result |
+---------+
| {T,e}   |
+---------+
| {n,g}   |
```

```
+---------+
| {r,i}   |
+---------+
| {}      |
+---------+
| {!,!,!} |
+---------+
```

```
SELECT
    unnest([['T', 'e'], ['n', 'g']], [['r', 'i'], []], [['!', '!', '!']]],
    max_depth := 3)
AS result;
```

```
+--------+
| result |
+--------+
| T      |
+--------+
| e      |
+--------+
| n      |
+--------+
| g      |
+--------+
| r      |
+--------+
| i      |
+--------+
| !      |
+--------+
| !      |
+--------+
| !      |
+--------+
```

▼ *Посмотреть еще примеры*

```
SELECT
    unnest([1,2,3])       AS numbers,
    unnest(['a','b','c']) AS letters;
```

```
+---------+---------+
| numbers | letters |
+---------+---------+
| 1       | a       |
+---------+---------+
| 2       | b       |
+---------+---------+
```

```
| 3       | c       |
+---------+---------+
```

```
SELECT
    unnest([1,2,3])   AS numbers,
    unnest(['a','b']) AS letters;
```

```
+---------+---------+
| numbers | letters |
+---------+---------+
| 1       | a       |
+---------+---------+
| 2       | b       |
+---------+---------+
| 3       | null    |
+---------+---------+
```

```
SELECT
    unnest([{'column_a': 1, 'column_b': 84},
            {'column_a': 100, 'column_b': NULL, 'column_c':22}],
            recursive := true);
```

```
+----------+----------+----------+
| column_a | column_b | column_c |
+----------+----------+----------+
| 1        | 84       | null     |
+----------+----------+----------+
| 100      | null     | 22       |
+----------+----------+----------+
```

```
SELECT
    unnest([{'column_a': 1, 'column_b': 84},
            {'column_a': 100, 'column_b': NULL, 'column_c':22}])
AS result;
```

```
+-----------------------------------------------------+
|                        result                       |
+-----------------------------------------------------+
| {"column_a": 1, "column_b": 84, "column_c": null}   |
+-----------------------------------------------------+
| {"column_a": 100, "column_b": null, "column_c": 22} |
+-----------------------------------------------------+
```

Оператор `::`

Описание Приводит тип данных к указанному.

Использование `argument::DATA_TYPE`

Приводит тип данных аргумента к указанному. В случае невозможности приведения выдает ошибку.

См. также `cast()`, `try_cast()`.

▼ *Посмотреть примеры*

- ```
  SELECT
      '1'::BIGINT                         AS result_1,
      '00:00:00.0'::TIME                  AS result_2,
      '2025-01-01T00:00:00.0'::TIMESTAMP  AS result_3;
  ```

  ```
  +----------+----------+---------------------+
  | result_1 | result_2 | result_3            |
  +----------+----------+---------------------+
  | 1        | 00:00:00 | 2025-01-01 00:00:00 |
  +----------+----------+---------------------+
  ```

- ```
  SELECT
      'Tengri'::BIGINT; -- error expected
  ```

  ```
  ERROR: ConversionException: Conversion Error: Could not convert string 'Tengri' to INT64
  ```

Функции модуля Python `tngri`

Для удобной работы с данными в ячейках типа Python создан модуль `tngri`. В нем доступны функции, описанные на этой странице.

Функции модуля `tngri` доступны только в ячейках типа Python и для их работы необходимо импортировать модуль: `import tngri`.

`tngri.run_notebook()`

Описание Запускает указанный ноутбук из другого ноутбука.

Использование `tngri.run_notebook('<notebook_name>')`

Запускает указанный ноутбук из другого ноутбука. Возвращает результаты выполнения всех ячеек запущенного ноутбука единым текстовым блоком.

Функцией удобно пользоваться при необходимости оркестрации запусков нескольких ноутбуков. С

ее помощью можно реализовать разную логику:

- Запуск ноутбука по условию
- Запуск ноутбука сразу после выполнения другого ноутбука
- Выделение части вычислений в отдельный ноутбук и запуск этой части из других ноутбуков

▼ *Посмотреть пример*

```
import tngri

result = tngri.run_notebook('My Notebook')

print(result)
```

```
<Output of cell 1 of My Notebook>
<Output of cell 2 of My Notebook>
done at 2025-12-12 14:09:14.497666
```

`tngri.sql()`

Описание Выполняет указанный запрос SQL внутри ячейки типа Python.

Использование `tngri.sql('<SQL_query>')`

Функцией удобно пользоваться при необходимости выполнить какие-либо запросы SQL непосредственно внутри ячейки типа Python, например, внутри цикла или в других сложных конструкциях, не создавая для этого отдельную ячейку типа SQL и используя любые локальные переменные и функции Python внутри текста запроса SQL.

Один из сценариев использования описан здесь.

▼ *Посмотреть примеры*

Пример 6

Создадим таблицу с именем из переменной `table_name` и запишем в нее в цикле:

- индекс (начиная с `1`)
- слово из фразы, заданной в переменной `test_phrase`
- результат применения к этому слову заданной функции `length_in_chars`

В каждой итерации цикла будем выводить значение индекса, добавленное слово и результат запроса SQL с текущим количеством строк в создаваемой таблице.

```
import tngri

def length_in_chars(text):
    if len(text) == 1:
```

```
        return '1 character'
    else:
        return f'{len(text)} characters'

table_name = 'my_table'
test_phrase = 'I love Tengri'

tngri.sql(f'CREATE OR REPLACE TABLE {table_name} \
          (index INT, word VARCHAR, length VARCHAR)'
         )

ind = 0
for word in test_phrase.split(' '):
    ind += 1
    tngri.sql(f"INSERT INTO {table_name} VALUES \
              ({ind}, '{word}', '{length_in_chars(word)}')"
             )
    print(f'Added word: "{word}"')
    print(tngri.sql(f'SELECT count(*) FROM {table_name}'))
```

```
Step: 1
Added word: "I"
shape: (1, 1)
+----------+
| column_0 |
| ---      |
| i64      |
+----------+
| 1        |
+----------+
Step: 2
Added word: "love"
shape: (1, 1)
+----------+
| column_0 |
| ---      |
| i64      |
+----------+
| 2        |
+----------+
Step: 3
Added word: "Tengri"
shape: (1, 1)
+----------+
| column_0 |
| ---      |
| i64      |
+----------+
| 3        |
+----------+
```

Теперь в ячейке типа SQL выведем созданную таблицу, упорядочив ее по индексу:

```
SELECT * FROM my_table
    ORDER BY index
```

```
+-------+--------+--------------+
| index | word   | length       |
+-------+--------+--------------+
| 1     | I      | 1 character  |
+-------+--------+--------------+
| 2     | love   | 4 characters |
+-------+--------+--------------+
| 3     | Tengri | 6 characters |
+-------+--------+--------------+
```

Пример 7

Выполним итеративную загрузку данных из файлов `.parquet` из хранилища S3 в таблицу по маске путей к файлам.

В каждой итерации цикла будем выводить результат запроса с количеством строк в создаваемой таблице.

```
import tngri

for i in range(1,4):
    file_name = f"s3://prostore/Stage/<lake_path>/{i}.parquet"
    tngri.sql(f"INSERT INTO raw_dyntest SELECT * FROM read_parquet('{file_name}')")
    print(tngri.sql("SELECT count(*) FROM raw_dyntest"))
```

```
shape: (1, 1)
+----------+
| column_0 |
| ---      |
| i64      |
+----------+
| 10000000 |
+----------+
shape: (1, 1)
+----------+
| column_0 |
| ---      |
| i64      |
+----------+
| 20000000 |
+----------+
shape: (1, 1)
```

```
+----------+
| column_0 |
| ---      |
| i64      |
+----------+
| 30000000 |
+----------+
```

`tngri.upload_df()`

Описание Загружает данные из DataFrame в Tengri.

Использование `tngri.upload_df(DataFrame)`

Загружает данные из указанного DataFrame в Tengri (в хранилище S3).
Возвращает строку с именем файла `.parquet`, в который были загружены данные.

Подробнее загрузка данных в Tengri с помощью Python описана здесь.

▼ *Посмотреть примеры*

- Создадим DataFrame, и загрузим его в Tengri:

```
import tngri
import pandas

my_df = pandas.DataFrame(range(100))

tngri.upload_df(my_df)
```

```
UploadedFile(s3_path='s3://prostore/Stage/ijwsajclddxw.parquet', _client=None)
```

- Создадим DataFrame, загрузим его в Tengri и запишем имя файла `.parquet` в переменную `file_name`:

```
import tngri
import pandas

my_df = pandas.DataFrame(range(100))
file_name = tngri.upload_df(my_df)

print(my_df)
print(file_name)
```

```
0
0    0
1    1
2    2
3    3
```

```
4    4
..  ..
95  95
96  96
97  97
98  98
99  99

[100 rows x 1 columns]
tdiiuetablyx.parquet
```

`tngri.upload_file()`

Описание Загружает данные из файла в Tengri.

Использование `tngri.upload_file(FilePath[, FileName])`

- `FilePath` — путь к файлу, который нужно загрузить в Tengri
- `FileName` — имя для файла в Tengri

Загружает данные из файла по указанному пути в Tengri (в хранилище S3).
Возвращает строку с путем и именем файла внутри хранилища S3, в который были загружены данные.

Один из сценариев использования описан здесь.

▼ *Посмотреть примеры*

- Загрузим в Tengri данные из файла `.json`, доступного по URL:

```
import tngri
import urllib.request

urllib.request.urlretrieve(

'https://tngri.postgrespro.ru/documentation/ru/stable/_attachments/tengri_data_types.json',
    'my_file.json'
    )

tngri.upload_file('my_file.json')
```

```
UploadedFile(s3_path='s3://prostore/Stage/pxfihzbonctd.json', _client=None)
```

Выведем первые 5 строк таблицы, прочитав ее из загруженного файла:

```
SELECT * FROM read_json("pxfihzbonctd.json")
    LIMIT 5
```

```
+----------+-----------+----------+---------------------------+
| name     | type      | category | description               |
+----------+-----------+----------+---------------------------+
| BIGINT   | data type | numeric  | Целые числа.              |
+----------+-----------+----------+---------------------------+
| BIGINT[] | data type | array    | Массивы целых чисел.      |
+----------+-----------+----------+---------------------------+
| BLOB     | data type | blob     | Двоичные объекты.         |
+----------+-----------+----------+---------------------------+
| BOOL     | data type | boolean  | Булевы значения.          |
+----------+-----------+----------+---------------------------+
| BOOL[]   | data type | array    | Массивы булевых значений. |
+----------+-----------+----------+---------------------------+
```

- Загрузим в Tengri данные из файла `.json`, доступного по URL, и сохраним имя загруженного файла в переменную:

```
import tngri
import urllib.request

urllib.request.urlretrieve(

'https://tngri.postgrespro.ru/documentation/ru/stable/_attachments/tengri_data_types.json',
    'my_file.json'
    )

file_name = tngri.upload_file('my_file.json')
print(file_name)
```

```
pxfihzbonctd.json
```

Выведем первые 5 строк таблицы, прочитав ее из загруженного файла:

```
SELECT * FROM read_json("pxfihzbonctd.json")
    LIMIT 5
```

```
+----------+-----------+----------+---------------------------+
| name     | type      | category | description               |
+----------+-----------+----------+---------------------------+
| BIGINT   | data type | numeric  | Целые числа.              |
+----------+-----------+----------+---------------------------+
| BIGINT[] | data type | array    | Массивы целых чисел.      |
+----------+-----------+----------+---------------------------+
| BLOB     | data type | blob     | Двоичные объекты.         |
+----------+-----------+----------+---------------------------+
| BOOL     | data type | boolean  | Булевы значения.          |
```

```
+----------+----------+----------+--------------------------+
| BOOL[]   | data type | array    | Массивы булевых значений. |
+----------+----------+----------+--------------------------+
```

`tngri.upload_s3()`

Описание Загружает файл из указанного бакета S3 в Tengri.

Использование

```
tngri.upload_s3(
    object = 's3://<file_path>.parquet',
    access_key = '<access_key>',
    secret_key = '<secret_key>',
    [filename = '<new_name>.parquet']
    )
```

- `object` — путь к файлу в хранилище S3.
- `access_key` — ключ доступа для S3.
- `secret_key` — секретный ключ для S3.

Расширение файла при этом может быть любым. Оно останется тем же, каким оно было в начальном файле. При необходимости для загружаемого файла можно задать путь и имя через опциональный параметр `filename`.

Пример использования описан здесь.

Типы данных

	Имя типа	Псевдонимы	
Числовые типы	`BIGINT`	`INTEGER`	Целые числа
	`NUMERIC`	`DECIMAL`	Вещественные числа с заданной точностью
	`DOUBLE`	`FLOAT`	Вещественные числа с переменной точностью
Текстовый тип	`VARCHAR`	`CHAR`, `BPCHAR`, `STRING`, `TEXT`	Текстовые строки
Типы для даты и времени	`DATE`		Даты
	`TIME`		Время
	`TIMESTAMP`		Моменты времени
	`TIMESTAMPTZ`		Моменты времени с часовым поясом
Тип JSON	`JSON`		Данные в JSON

	Имя типа	Псевдонимы	
Типы для массивов	`BIGINT[]`	`INTEGER[]`	Массивы целых чисел
	`VARCHAR[]`	`CHAR[]`, `BPCHAR[]`, `STRING[]`, `TEXT[]`	Массивы текстовых строк
	`BOOL[]`	`BOOLEAN[]`	Массивы булевых значений
Двоичный тип	`BLOB`	`BYTEA`	Двоичные объекты
Логический тип	`BOOL`	`BOOLEAN`	Булевы значения
Тип для геоданных	`GEOMETRY`	`GEOMETRY`	Геопространственные данные

Числовые типы

Имя	Псевдоним	Диапазон	Описание
`BIGINT`	`INTEGER`	`- 2^63` … `2^63 - 1`	Целые числа
`NUMERIC`	`DECIMAL`	`WIDTH`: 37	Вещественные числа с заданной точностью
`DOUBLE`	`FLOAT`	`1E-307` … `1E+308`	Вещественные числа с переменной точностью

Целочисленный тип

- `BIGINT`
  - Псевдоним: `INTEGER`

Тип данных `BIGINT` хранит целые числа, то есть числа без дробных компонентов, различных диапазонов. Попытки сохранить значения за пределами допустимого диапазона приведут к ошибке.

Диапазон возможных значений для типа `BIGINT`: от `- 2^63` до `2^63 - 1`.

Тип с заданной точностью

- `NUMERIC`
  - Псевдоним: `DECIMAL`

Тип данных `NUMERIC (WIDTH, SCALE)` представляет собой точное десятичное значение с фиксированным десятичным разделителем — точкой.

При создании значения типа `NUMERIC` можно указать параметры `WIDTH` и `SCALE`.

- Параметр `WIDTH` определяет максимальное общее количество цифр.
- Параметр `SCALE` определяет количество цифр после десятичной точки.

Например, тип `NUMERIC(3, 2)` может содержать значение `1.23`, но не может содержать значение `12.3` или `1.234`. Если параметры `WIDTH` и `SCALE` не заданы явно, то используются значения по умолчанию:

`NUMERIC(37,18)`.

Сложение, вычитание и умножение двух десятичных чисел с фиксированной точкой возвращает другое десятичное число с фиксированной точкой с требуемыми `WIDTH` и `SCALE` или выдает ошибку, если требуемая `WIDTH` превышает максимальную поддерживаемую `WIDTH`, которая составляет `37`.

Если вам требуется хранение чисел с известным количеством десятичных знаков, а также точные операции сложения, вычитания и умножения (например, для **денежных сумм**), то следует использовать тип данных с заданной точностью `NUMERIC`.

Тип с переменной точностью

- `DOUBLE`
  - Псевдоним: `FLOAT`

Тип данных `DOUBLE` — числовой тип с переменной точностью. Этот тип хранит число с плавающей десятичной точкой двойной точности (8 байтов).

Диапазон возможных значений для типа `DOUBLE`: от `1E-307` до `1E+308`.

Как и в случае с типом данных `NUMERIC`, при преобразовании литералов или приведении других типов данных к типу с переменной точностью входные данные, которые не могут быть представлены точно, сохраняются в виде приближенных значений.

В то время как умножение, сложение и вычитание для типа `NUMERIC` являются точными операциями, те же операции для типа с переменной точностью являются только приблизительными.

Если вам нужно выполнять быстрые или сложные вычисления, тип данных с переменной точностью может быть более подходящим, чем тип `NUMERIC`. Однако, если вы используете результаты этих вычислений для принятия важных решений, то вам следует тщательно проанализировать реализацию этих вычислений в пограничных случаях (диапазоны, бесконечные значения, антипереполнения, недопустимые операции и т.п.). Они могут обрабатываться иначе, чем вы ожидаете.

Примеры

Создадим таблицу `numbers` со столбцами, имеющими числовые типы `BIGINT`, `NUMERIC(4,3)`, `DOUBLE` и заполним одну строку значениями `2^62`, `1.1` и `3.14159265358979323846`:

```
CREATE TABLE numbers(
    bigint BIGINT,
    numeric NUMERIC(4,3),
    double DOUBLE);

INSERT INTO numbers VALUES
    (2^62, 1.1, 3.14159265358979323846);
```

```
SELECT * FROM numbers;
```

```
+--------------------+---------+-------------------+
|        bigint      | numeric |       double      |
+--------------------+---------+-------------------+
| 4611686018427388000 | 1.100   | 3.141592653589793 |
+--------------------+---------+-------------------+
```

Обратите внимание, что в программном выводе в столбце `numeric` исходное число отображается с указанной точностью (3 знака после десятичного разделителя), а в столбце `double` исходное число отображается с максимально возможной точностью, поэтому количество знаков после разделителя меньше, чем в введенном числе.

Полезные ссылки

- Числовые функции

Текстовый тип

- `VARCHAR`
  - Псевдонимы: `CHAR`, `BPCHAR`, `STRING`, `TEXT`

Описание

Для хранения текстовых строк используется тип `VARCHAR`. Этот тип позволяет хранить символы Unicode. Данные кодируются в формате UTF-8.

Чтобы задать значение текстовой строки в выражении `INSERT`, следует использовать одинарные кавычки.

Пример

Создадим таблицу `varchar_table` и заполним текстовыми строками две ячейки:

```
CREATE TABLE varchar_table(text_column VARCHAR);

INSERT INTO varchar_table VALUES
    ('My name is Tengri'),
    ('My nickname is TNGRi');

SELECT * FROM varchar_table;
```

```
+----------------------+
|      text_column     |
+----------------------+
```

```
| My name is Tengri    |
+----------------------+
| My nickname is TNGRi |
+----------------------+
```

Обратите внимание, что в программном выводе текстовые строки отображаются без кавычек.

Полезные ссылки

- `LIKE`
- Текстовые функции
- Функции для регулярных выражений

Типы для даты и времени

Имя	Формат	Описание
`DATE`	`YYYY-MM-DD`	Даты
`TIME`	`[YYYY-MM-DD ]HH:MM[:SS][.MS]`	Время
`TIMESTAMP`	`YYYY-MM-DD hh:mm[:ss][.zzzzzz]`	Моменты времени
`TIMESTAMPTZ`	`YYYY-MM-DD hh:mm[:ss][.zzzzzz][+-TT[:tt] | ZONE]`	Моменты времени с часовым поясом

Даты

- `DATE`

Тип `DATE` используется для хранения даты (сочетание года, месяца и дня). Даты считаются по Григорианскому календарю, даже для времени до его введения.

Данные для этого типа должны быть отформатированы в соответствии со стандартом ISO 8601: `YYYY-MM-DD`.

Время

- `TIME`

Тип `TIME` используется для хранения времени (часы, минуты, секунды и микросекунды внутри суток).

Данные для этого типа должны быть отформатированы в соответствии со стандартом ISO 8601: `[YYYY-MM-DD ]HH:MM:SS[.MS]`.

Тип `TIME` следует использовать только в редких случаях, когда часть даты в метке времени может игнорироваться. В большинстве случаев для представления конкретных моментов времени следует использовать тип `TIMESTAMP`.

Моменты времени

- `TIMESTAMP`
- `TIMESTAMPTZ`

Типы `TIMESTAMP` и `TIMESTAMPTZ` используются для хранения временных меток — конкретных моментов времени. Они объединяют информацию о дате (тип `DATE`) и времени (тип `TIME`).

Для типа `TIMESTAMP` данные должны быть отформатированы в соответствии со стандартом ISO 8601:
`YYYY-MM-DD hh:mm:ss[.zzzzzz]`.

Для типа `TIMESTAMPTZ` (*TIMESTAMP TIME ZONE*) данные должны быть отформатированы так же, но допускается добавление часового пояса:
`YYYY-MM-DD hh:mm:ss[.zzzzzz][+-TT[:tt] | ZONE]`.

Десятичные знаки, выходящие за пределы поддерживаемой точности, игнорируются и не приводят к ошибке.

Часовые пояса могут задаваться либо через отступ в часах от указанного времени (`+01`, `-01`), либо через явное указание идентификатора из списка часовых поясов `tz_database`.

Примеры

Создадим таблицу `dates` со столбцами, имеющими типы `DATE`, `TIME` и `TIMESTAMP` и запишем в них различные валидные значения:

```
CREATE TABLE dates(
    date DATE,
    time TIME,
    timestamp TIMESTAMP);

INSERT INTO dates VALUES
    ('2025-01-02', '1:1', '2025-01-02 1:1:00'),
    ('2025-01-02', '1:01:11', '2025-01-02 1:1:11.111'),
    ('2025-01-02', '01:01:11.1234567', '2025-01-02 1:1:11.1234567');

SELECT * FROM dates;
```

```
+------------+-----------------+----------------------------+
|    date    |       time      |          timestamp         |
+------------+-----------------+----------------------------+
| 2025-01-02 | 01:01:00        | 2025-01-02 01:01:00        |
+------------+-----------------+----------------------------+
| 2025-01-02 | 01:01:11        | 2025-01-02 01:01:11.111000 |
+------------+-----------------+----------------------------+
| 2025-01-02 | 01:01:11.123456 | 2025-01-02 01:01:11.123456 |
+------------+-----------------+----------------------------+
```

Обратите внимание, что в программном выводе в столбце `time` время отображается в стандартном виде и с обязательным значением секунд (несмотря на различные способы ввода). В столбце `timestamp` время также отображается в стандартном виде. Микросекунды отображаются, только если они были заданы и с точностью в 6 знаков.

Создадим таблицу `time_stamps` со столбцами, имеющими типы `TIMESTAMP` и `TIMESTAMPTZ` и запишем в них одни и те же пары значений — с указанием часового пояса и без:

```
CREATE TABLE time_stamps(
    timestamp TIMESTAMP,
    timestamptz TIMESTAMPTZ);

INSERT INTO time_stamps VALUES
    ('2025-01-02 0:0:0',       '2025-01-02 0:0:0'),
    ('2025-01-02 0:0:0+01:00', '2025-01-02 0:0:0+01:00');

SELECT * FROM time_stamps;
```

```
+---------------------+---------------------------+
|      timestamp      |        timestamptz        |
+---------------------+---------------------------+
| 2025-01-02 00:00:00 | 2025-01-02 00:00:00+00:00 |
+---------------------+---------------------------+
| 2025-01-01 23:00:00 | 2025-01-01 23:00:00+00:00 |
+---------------------+---------------------------+
```

Обратите внимание, что в обоих случаях указание на часовой пояс обрабатывается верно, но в случае `TIMESTAMPTZ` оно сохраняется в таблице, а в случае типа `TIMESTAMP` — нет.

Для демонстрации работы с часовыми поясами вставим в таблицу `time_stamps` значения времени с разными указаниями часового пояса. Для наглядности вставим одни и те значения в две колонки с типами `VARCHAR` и `TIMESTAMPTZ`:

```
CREATE TABLE time_stamps(
    timestampt_text VARCHAR,
    timestamptz TIMESTAMPTZ);

INSERT INTO time_stamps VALUES
    ('2025-01-01 0:0:0 UTC', '2025-01-01 0:0:0 UTC'),
    ('2025-01-01 0:0:0 CET', '2025-01-01 0:0:0 CET'),
    ('2025-01-01 0:0:0+01' , '2025-01-01 0:0:0+01' ),
    ('2025-01-01 0:0:0-01' , '2025-01-01 0:0:0-01' ),
    ('2025-01-01 0:0:0+25' , '2025-01-01 0:0:0+25' ),
    ('2025-01-01 0:0:0-25' , '2025-01-01 0:0:0-25' );
```

```
SELECT * FROM time_stamps;
```

```
+----------------------+--------------------------+
|    timestampt_text   |        timestamptz       |
+----------------------+--------------------------+
| 2025-01-01 0:0:0 UTC | 2025-01-01 00:00:00+00:00 |
+----------------------+--------------------------+
| 2025-01-01 0:0:0 CET | 2024-12-31 23:00:00+00:00 |
+----------------------+--------------------------+
| 2025-01-01 0:0:0+01  | 2024-12-31 23:00:00+00:00 |
+----------------------+--------------------------+
| 2025-01-01 0:0:0-01  | 2025-01-01 01:00:00+00:00 |
+----------------------+--------------------------+
| 2025-01-01 0:0:0+25  | 2024-12-30 23:00:00+00:00 |
+----------------------+--------------------------+
| 2025-01-01 0:0:0-25  | 2025-01-02 01:00:00+00:00 |
+----------------------+--------------------------+
```

Попробуем ввести в таблицу `dates` невалидные данные:

```
CREATE TABLE dates(
    date DATE);

INSERT INTO dates VALUES
    ('2025-12-13'),
    ('2025-13-13');
```

```
ERROR: ConversionException: Conversion Error: date field value out of range:
"2025-13-13"
```

Данные не могут быть введены, так как задан невалидный номер месяца — `13`.

```
CREATE TABLE dates(
    time TIME);

INSERT INTO dates VALUES
    ('1:1'),
    ('24:1');
```

```
ERROR: ConversionException: Conversion Error: time field value out of range:
"24:1", expected format is ([YYYY-MM-DD ]HH:MM:SS[.MS])
```

Данные не могут быть введены, так как задано невалидное значение часа — `24`.

```
CREATE TABLE dates(
    timestamp TIMESTAMP);

INSERT INTO dates VALUES
    ('2025-01-01 1');
```

```
ERROR: ConversionException: Conversion Error: invalid timestamp field format:
"2025-01-01 1", expected format is (YYYY-MM-DD HH:MM:SS[.US][±HH:MM| ZONE])
```

Данные не могут быть введены, так как задано невалидное значение времени — `1`.

```
CREATE TABLE dates(
    timestamptz TIMESTAMPTZ);

INSERT INTO dates VALUES
    ('2025-01-01 0:0:0 Europe/Moscow'),
    ('2025-01-01 0:0:0 Asia/Srednekolymsk'),
    ('2025-01-01 0:0:0 MSK');
```

```
ERROR: NotImplementedException: Not implemented Error: Unknown TimeZone 'MSK'
```

Данные не могут быть введены, так как задано невалидное значение часового пояса `MSK`. Значения `Europe/Moscow` и `Asia/Srednekolymsk` при этом являются валидными, так как входят в список идентификаторов часовых поясов `tz_database`.

Полезные ссылки

- Функции для даты и времени

Тип JSON

- `JSON`

Описание

Тип `JSON` используется для хранения данных JSON согласно стандартному синтаксису, описанному в спецификации.

Для хранения таких данных можно использовать и тип `VARCHAR`, но при использовании типа `JSON` данные будут проверяться на соответствие вводимых значений формату JSON, поэтому использовать специальный тип `JSON` в таких случаях удобнее. К тому же для типа `JSON` доступны

специальные функции, позволяющие обращаться напрямую к данным в структуре JSON.

Кавычки внутри текста в формате JSON должны быть двойными. А обрамляющие этот текст кавычки внутри выражения `INSERT` должны быть одинарными (см. пример ниже).

Примеры

Создадим таблицу `js_table` и вставим в столбец `js_data` данные в формате JSON из этого примера:

```
CREATE TABLE js_table(name VARCHAR, js_data JSON);

INSERT INTO js_table VALUES
('John Smith',
'{
  "first_name": "John",
  "last_name": "Smith",
  "is_alive": true,
  "age": 27,
  "address": {
    "street_address": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postal_code": "10021-3100"
  },
  "phone_numbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [
    "Catherine",
    "Thomas",
    "Trevor"
  ],
  "spouse": null
}');
```

Выведем имена полей верхнего уровня из загруженных данных JSON с помощью функции `json_keys`:

```
SELECT
    json_keys(js_data) AS json_fields
FROM js_table;
```

```
+-------------------------------------------------------------------------+
|                               json_fields                               |
+-------------------------------------------------------------------------+
| {first_name,last_name,is_alive,age,address,phone_numbers,children,spouse} |
+-------------------------------------------------------------------------+
```

Выведем в отдельные столбцы полное имя из текстового столбца `name`, а из данных JSON возьмем возраст `age` и количество детей — длину массива в поле `children`. Для этого используем фунции `json_extract` и `json_array_length`.

```
SELECT
    name,
    json_extract(js_data, 'age') AS age,
    json_array_length(js_data, 'children') AS children_num
FROM js_table;
```

```
+------------+-----+--------------+
|    name    | age | children_num |
+------------+-----+--------------+
| John Smith | 27  | 3            |
+------------+-----+--------------+
```

Полезные ссылки

- Функции для JSON

Типы для массивов

- `BIGINT[]`
  - Псевдоним: `INTEGER[]`
- `VARCHAR[]`
  - Псевдонимы: `CHAR[]`, `BPCHAR[]`, `STRING[]`, `TEXT[]`
- `BOOL[]`
  - Псевдонимы: `BOOLEAN[]`

Описание

Типы `BIGINT[]`, `VARCHAR[]` и `BOOL[]` используются для хранения массивов целых чисел, массивов строк и массивов булевых значений соответственно.

Длина массивов в одном столбце таблицы может не совпадать.

Массивы могут использоваться для хранения векторов, таких как эмбеддинги (векторные

представления) текстов или изображений.

Примеры

Создадим таблицу `array_table` и вставим в нее значения типа `VARCHAR` и типа `VARCHAR[]`:

```
CREATE TABLE array_table(text_column VARCHAR, array_column VARCHAR[]);

INSERT INTO array_table VALUES
    ('I love Tengri', ['I','love', 'Tengri']),
    ('I adore Tengri', ['I','adore', 'Tengri']);

SELECT * FROM array_table;
```

```
+----------------+------------------+
|   text_column  |   array_column   |
+----------------+------------------+
| I love Tengri  | {I,love,Tengri}  |
+----------------+------------------+
| I adore Tengri | {I,adore,Tengri} |
+----------------+------------------+
```

Обратите внимание, что в программном выводе массивы отображаются через фигурные скобки `{}`, а текстовые значения в них — без кавычек.

Теперь создадим таблицу `array_table` со столбцами типа `BIGINT` и `BIGINT[]` и вставим в нее числовые значения и массивы чисел:

```
CREATE TABLE array_table(number_column BIGINT, array_column BIGINT[]);

INSERT INTO array_table VALUES
    (2025, [2,0,2,5]),
    (0025, [0,0,2,5]);

SELECT * FROM array_table;
```

```
+---------------+--------------+
| number_column | array_column |
+---------------+--------------+
| 2025          | {2,0,2,5}    |
+---------------+--------------+
| 25            | {0,0,2,5}    |
+---------------+--------------+
```

Обратите внимание, что числовое значение `0025` в программном выводе приведено к нормальному

виду `25`, а массивы в программном выводе отображаются через фигурные скобки `{}`.

Двоичный тип

- `BLOB`
  - Псевдоним: `BYTEA`

Описание

Тип данных `BLOB` предназначен для хранения произвольных двоичных объектов. Он может содержать любые двоичные данные без каких-либо ограничений. Что на самом деле представляют собой эти данные, для Tengri остается неизвестным.

`BLOB` обычно используются для хранения нетекстовых объектов, тип которых явно не поддерживается, например для изображений. Хотя тип `BLOB` может хранить объекты размером до 4 ГБ, не рекомендуется хранить очень большие объекты в Tengri. Обычно более удобно хранить большие файлы в файловой системе, а пути к файлам хранить в Tengri в виде типа `VARCHAR`.

Полезные ссылки

- Функции для двоичных данных

Логический тип

- `BOOL`
  - Псевдоним: `BOOLEAN`

Описание

Тип `BOOL` представляет собой утверждение истинности и может принимать значения:
`true`, `false` (булевы значения) или `NULL`.

Булевы значения могут быть явно заданы с помощью литералов `true` и `false`. Однако чаще всего они создаются в результате сравнения. Например, сравнение `i > 10` возвращает булево значение.

Булевы значения могут использоваться в выражениях `WHERE` и `HAVING` для фильтрации результатов. В этом случае выражения, результат которых оценивается как `true`, пройдут фильтр, а выражения, результат которых оценивается как `false` или `NULL`, будут отброшены.

Логические операторы

Для комбинирования булевых значений могут использоваться логические операторы `AND` и `OR`.

Table 1. Таблица истинности для оператора `AND`

X	X AND true	X AND false	X AND NULL
`true`	`true`	`false`	`NULL`
`false`	`false`	`false`	`false`
`NULL`	`NULL`	`false`	`NULL`

Table 2. Таблица истинности для оператора `OR`

X	X OR true	X OR false	X OR NULL
`true`	`true`	`true`	`true`
`false`	`true`	`false`	`NULL`
`NULL`	`true`	`NULL`	`NULL`

Тип для геоданных

- `GEOMETRY`

Описание

Тип для чтения и записи геопространственных данных, преобразования между системами координат и работы с инструментами ГИС.

Примеры

Вычислим расстояние на плоскости между двумя точками, заданными в целочисленных координатах.

```
SELECT
    ST_Distance('POINT (0 0)'::GEOMETRY, 'POINT (5 12)'::GEOMETRY)
        AS distance;
```

```
+----------+
| distance |
+----------+
| 13       |
+----------+
```

Вычислим расстояние между офисами Postgress Pro и Oracle в километрах.

```
SELECT
    round(ST_Distance_Sphere('POINT (55.69189394353437, 37.564623398131985)'::GEOMETRY,
                             'POINT (30.243622717202587, -97.72199761339736)'::GEOMETRY
                             )/1000)
    AS "distance between postgress and oracle, km";
```

```
+-----------------------------------------+
| distance between postgress and oracle, km |
+-----------------------------------------+
| 9561                                    |
+-----------------------------------------+
```

Полезные ссылки

- Функции для геоданных

Руководство администратора

Установка

1. Получаем архив с необходимыми файлами и распаковываем:

```
tar xf example_client.tar.gz
cd example_client
```

```
example_client/
├──── client.crt
├──── client.key
├──── install.sh
└──── license.txt

1 directory, 4 files
```

2. Запускаем скрипт для установки сертификата, файла конфигурации и репозитория APT:

```
sudo ./install.sh
```

3. Устанавливаем пакеты:

   a. Только Tengri

   ```
   sudo apt install tengri
   ```

   b. Tengri с локальным S3 (MinIO)

   ```
   sudo apt install tengri tengri-minio
   ```

- После установки Tengri будет готов к работе на порту `3000`.
  Логин и пароль по умолчанию: `admin / admin`.
- Пакет `tengri-minio` установит локальный MinIO с дефолтными настройками. Подробности о дальнейшей настройке можно узнать в документации MinIO.

Настройка конфигурации

- Дополнительные настройки конфигурации Tengri можно задать в файле `/etc/tengri/tengri.conf` (по умолчанию файл отсутствует).

▼ *Полный список возможных настроек в файле* `tengri.conf`

```
# TNGRi configuration parameters

# =========
# Licencing
# =========
#
TNGRI_LICENSE_FILE=/opt/tengri/license.txt

# ============================
# S3 configuration parameters
# ============================
#
#
# Access key id and secret key of user that will connect to S3
# rename SECRET_ACCESS_KEY to SECRET_KEY
TNGRI_S3_ACCESS_KEY_ID=minioadmin
TNGRI_S3_SECRET_ACCESS_KEY=minioadmin
#
# Region to send to S3 server. Generally you don't need to change that, just make
# sure that target S3 has this same region
TNGRI_S3_REGION=eu-central-1
TNGRI_S3_DEFAULT_REGION=eu-central-1
#
# API endpoint URL for target S3. Must be in FQDN form
#     TNGRI_S3_ENDPOINT_URL=http://example.com
TNGRI_S3_ENDPOINT_URL=http://127.0.0.1:9002
#
# Bucket name to use. Must exist before start, so make sure that it exists and
# has full access rights for this bucket. Must be in full form, i.e.
#     TNGRI_S3_BUCKET_NAME=s3://example:9000
TNGRI_S3_BUCKET_NAME=prostore

# ======================
# TNGRi server parameters
# ======================
#
#
# Iceberg catlog path in S3 bucket
TNGRI_WAREHOUSE_PATH=s3://prostore/iceberg/
#
# Name for the default schema in TNGRi.
TNGRI_DEFAULT_SCHEMA=public
#
# URI for catalog endpoint. Must be either PostgreSQL or SQLite connection string, i.e.
#     TNGRI_CATALOG_URI=postgresql://user:password@example.com/database
TNGRI_CATALOG_URI=postgresql://prostore:prostore@127.0.0.1:5432/prostore
#
# Path to writable directory where cached files will be placed.
#
# WARNING: Should be placed on SSD storage. Tngri uses aggressive caching strategies
# to reduce S3 load
#
```

```
TNGRI_CACHE_DIR=/var/lib/tengri/cache
#
# Path to writable directory where python kernels will be stored.
TNGRI_KERNEL_DIR=/var/lib/tengri/kernels
#
# If set to false then no agent is required to process queries.
# Must be true for production environments
TNGRI_REMOTE_SESSIONS=true
#
# Set to true for more log output
TNGRI_DEBUG=false
#
# Host to bind main server to
TNGRI_HOST=0.0.0.0
#
# Port to bind PostgreSQL protocol implementation to
TNGRI_PORT=5433
#
# Host to bind RPC server for communication between agent and server
TNGRI_RPC_HOST=0.0.0.0
#
# Port to bind RPC server for communication between agent and server
TNGRI_RPC_PORT=57776

# ============================
# Web application configuration
# ============================
#
#
# SSL configuration. If set then Web communication will be done over SSL, otherwise
# it will be plaintext.
#
# Path to SSL key.
# TNGRI_SSL_KEY=
#
# Path to SSL certificate file.
# TNGRI_SSL_CERT=
#
# Enables TNGRi web app.
TNGRI_ENABLE_WEBSOCKET=true
#
# Path to static files for web app.
TNGRI_WEBAPP_DIR=/usr/share/tengri-web
#
# Host to bind HTTP server to
TNGRI_WEBAPP_HOST=0.0.0.0
#
# Port to bind HTTP server to
TNGRI_WEBAPP_PORT=3000
#
# Host to bind Websocket server to. It is used extensively by webapp and python clients
TNGRI_WEBSOCKET_HOST=0.0.0.0
#
```

```
# Port to bind Websocket server to
TNGRI_WEBSOCKET_PORT=3001
#
# Host to bind PyLSP server to. It is used by webapp to provide code completion
TNGRI_PYLSP_HOST=0.0.0.0
#
# Port to bind PyLSP server to
TNGRI_PYLSP_PORT=3003
#
# Host to bind CRUD. Used by webapp to send messages in chats, manage scheduled
# notebooks and manage external catalogs
TNGRI_CRUD_HOST=0.0.0.0
#
# Port to bind CRUD to
TNGRI_CRUD_PORT=3004

# ============================
# TNGRi services configuration
# ============================
#
# Enable TNGRi's Iceberg REST catalog API provider
TNGRI_ENABLE_REST_CATALOG=true

# Host to bind REST catalog server to
TNGRI_REST_CATALOG_HOST=0.0.0.0

# Port to bind REST catalog to
TNGRI_REST_CATALOG_PORT=3005

# Enable compaction daemon
TNGRI_ENABLE_COMPACTION=true

# Enable notebook CRON daemon
TNGRI_ENABLE_CRON=true

# ============================
# TNGRi deployment information
# ============================
#
# Needed for external clients to connect to TNGRi services. For example, python clients
# connecting from user's local machines. Set these to the values that point to this
# installation, for example, if using reverse proxy for HTTP or Postgres protocol.
#
# Change the name of site deployment
# TNGRI_SITE_NAME=tngri
#
# Websocket address to connect to. Must be in FQDN form, i.e.
#     TNGRI_SITE_WS_ADDR=ws://example.com:3001
TNGRI_SITE_WS_ADDR=ws://0.0.0.0:3001
#
# Postgres protocol address to connect to, i.e.
#     TNGRI_SITE_PSQL_ADDR=example.com:5433
# TNGRI_SITE_PSQL_ADDR=
```

```
#
# S3 address to connect to. Must be in FQDN form, i.e.
#     TNGRI_SITE_S3_ADDR=http://example.com:9000
# TNGRI_SITE_S3_ADDR=
#
# S3 access key for public usage (Stage only)
# TNGRI_SITE_PUBLIC_LOADER_ACCESS_KEY=public_loader
#
# S3 secret key for public usage (Stage only)
# TNGRI_SITE_PUBLIC_LOADER_SECRET_KEY=public_loader
#
```

- Дополнительные настройки конфигурации MinIO можно задать в файле `/etc/default/tengri-minio` в соответствии с документацией MinIO.

  После обновления файла конфигурации MinIO необходимо выполнить:

  ```
  sudo systemctl daemon-reload
  sudo systemctl restart minio
  ```

Сброс пароля администратора

В случае если утрачен доступ к паролю администратора, его можно сбросить.

Для этого на сервере, где был установлен пакет, нужно выполнить:

```
sudo -u tengri bash -c 'export $(egrep -v "^#|^$|.*#" /var/lib/tengri/tengri.defaults |
xargs); prostore reset-admin-password'"
```

После этого текущий пароль администратора сбросится, и новый автоматически сгенерированный пароль будет показан в логе вывода.

Настройка ролей для пользователей

Создание ролей для пользователей

Чтобы создать роль `analyst`, выполняем запрос:

```
CREATE ROLE analyst;
```

- Подробнее о создании ролей.

Назначение ролей пользователям

Чтобы назначить пользователю `tengri_user` роль `analyst`, выполняем запрос:

```
GRANT ROLE analyst TO tengri_user;
```

- Подробнее о назначении ролей.

Предоставление ролям привилегий

Чтобы предоставить роли `analyst` привилегии для чтения и записи внутри схемы `analytical_sandbox`, выполняем запрос:

```
GRANT
    USAGE,
    MODIFY,
    CREATE TABLE,
    CREATE VIEW
ON SCHEMA analytical_sandbox
TO ROLE analyst;
```

После этого у всех пользователей с ролью `analyst` появятся права на чтение и изменение таблиц и представлений внутри схемы `analytical_sandbox`.

При необходимости скорректируйте список предоставляемых привилегий для данной роли.

- Подробнее о предоставлении привилегий.

Отзыв привилегий у ролей

Чтобы отозвать у роли `analyst` привилегию для записи внутри схемы `analytical_sandbox`, выполняем запрос:

```
REVOKE
    MODIFY
ON SCHEMA analytical_sandbox
FROM ROLE analyst;
```

При необходимости скорректируйте список отзываемых привилегий для данной роли.

- Подробнее об отзыве привилегий.

Отзыв ролей у пользователей

Чтобы отозвать у пользователя `tengri_user` роль `analyst`, выполняем запрос:

```
REVOKE ROLE analyst FROM tengri_user;
```

- Подробнее об отзыве ролей.

Полезные ссылки

- Начальная настройка системы
- Управление пользователями и правами
- Операции с пользователями
- Операции с ролями
- Операции с привилегиями
- Вычислительные пулы

Сценарии

- Начальная настройка системы
- Загрузка данных
- Автоматизация загрузки данных из множественных файлов
- Обращение к хранилищу Iceberg напрямую с помощью Python
- Аналитические сценарии
  - Анализ данных из постов Instagram*
  - Анализ геоданных спортивных трекеров
  - Анализ текста на примере дата-сета FineWeb

Начальная настройка системы

Предположим, что нам нужно произвести начальные настройки системы для работы одного пользователя, у которого будет одна роль. Пользователь будет работать на одном вычислительном пуле внутри одной схемы.

Чтобы описанные ниже действия были возможны, их должен производить либо администратор, либо пользователь с привилегиями на создание других пользователей, ролей и вычислительных пулов.

1. Создадим роль `analyst`:

```
CREATE ROLE analyst;
```

2. Создадим вычислительный пул `compute_s` и установим для него размер вычислителей `S`:

```
CREATE WORKER POOL compute_s WORKER SIZE S;
```

3. Создадим пользователя `tengri_user`, зададим его пароль для идентификации и назначим для него вычислительный пул по умолчанию `compute_s`:

```
CREATE USER tengri_user
    IDENTIFIED BY PASSWORD '***'
    DEFAULT WORKER POOL compute_s;
```

4. Назначим пользователю `tengri_user` роль `analyst`:

```
GRANT ROLE analyst TO tengri_user;
```

5. Предоставим роли `analyst` привилегии `USAGE` и `MONITOR` на вычислительный пул `compute_s`:

```
GRANT USAGE, MONITOR ON WORKER POOL compute_s TO ROLE analyst;
```

Далее:

a. если в системе уже есть схема `analytical_sandbox` и наш пользователь будет работать только в ней, то предоставим ему привилегию на работу со всеми таблицами внутри этой схемы, как существующими, так и теми, которые будут созданы в будущем:

```
GRANT ALL ON SCHEMA analytical_sandbox TO ROLE analyst;
```

b. если же пользователь будет работать в новой схеме, то создадим эту схему `analytical_sandbox_new` и предоставим ему привилегию на работу со всеми таблицами внутри этой схемы:

```
CREATE SCHEMA analytical_sandbox_new;

GRANT ALL ON SCHEMA analytical_sandbox_new TO ROLE analyst;
```

6. Если предполагается, что пользователь `tengri_user` будет создавать новые схемы, то предоставим ему привилегию на создание схемы внутри каталога:

```
GRANT CREATE SCHEMA ON CATALOG TO ROLE analyst;
```

После этого пользователь `tengri_user` сможет:

- Создавать таблицы внутри схемы `analytical_sandbox` (или `analytical_sandbox_new`)
- Читать таблицы внутри схемы `analytical_sandbox` (или `analytical_sandbox_new`)

С такими настройками пользователь `tengri_user` сможет:

- Удалять таблицы внутри схемы `analytical_sandbox` (или `analytical_sandbox_new`)
- Удалить схему `analytical_sandbox` (или `analytical_sandbox_new`)

Загрузка данных

Загрузить данные в Tengri можно следующими способами:

- С помощью мастера загрузки — кнопка **[Upload file]**.
- Через модуль Python `tngri`.

Загрузка данных из локального файла с помощью мастера загрузки

Поддерживаемые расширения файлов для загрузки:

- `.csv`
- `.json`
- `.parquet`
- `.xlsx`
- `.zip`

Чтобы загрузить данные из файла:

1. Нажмите **[Upload file]** (*Загрузить файл*) в интерфейсе Tengri.
2. Выберете или переместите файл в открывшуюся область загрузки.
3. В открывшемся окне **Parse file** (*Распознать файл*) проверьте правильность распознанных данных.
   При необходимости выберете нужные настройки распознавания и нажмите **[Next]** (*Далее*).
4. После этого вы увидите ячейку с кодом загрузки файла для вставки в ноутбук.
   Для файла `file_name.csv` и для пользователя `user_name` она будет иметь вид:

   ```
   select *
   from read_csv(
       "user_name/<id>_file_name.csv"
   )
   ```

   Нажмите **[Add cell]** (*Добавить ячейку*), чтобы добавить эту ячейку в ваш ноутбук.

После добавления ячейки в ноутбук данные из файла будут доступны для работы.

Загрузка данных из файла через модуль Python `tngri`

Чтобы загружать данные из нелокальных файлов, можно использовать модуль Python `tngri`.

Загрузка данных из файла по URL

Пример кода на Python для загрузки в Tengri данных из файла `iris.csv`, расположенного по указанному URL:

```
import polars ①
import tngri
```

```
df = polars.read_csv(
    "https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv" ②
)

tngri.upload_df(df) ③
```

① Импортируем необходимые модули Python

② Считываем файл по указанному URL и записываем его в переменную `df` с помощью модуля `polars`

③ Загружаем данные из `df` в Tengri с помощью функции `tngri.upload_df`

Помимо файлов `.csv`, этот способ может использоваться и для других расширений — `.json`, `.xlsx` и прочих (подробнее см. в документации Polars по загрузке данных).

После этого в ячейке вывода появится сообщение вида:

```
UploadedFile(s3_path='s3://<path>/<file_id>.parquet')
```

Теперь данные из файла доступны для работы. Чтобы работать с ними, нужно использовать функцию `read_parquet` и указать id загруженного файла `.parquet` с предыдущего шага:

```
SELECT * FROM read_parquet('<file_id>.parquet');
```

Расширение `.parquet` будет у загруженного таким образом файла в любом случае вне зависимости от расширения начального файла.

Загрузка данных из файла, сохраненного в S3

Пример кода на Python для загрузки в Tengri данных из файла `my_file.parquet`, расположенного в вашем бакете S3:

```
import tngri ①

tngri.upload_s3(
    object="s3://my_folders/my_file.parquet", ②
    access_key="***",
    secret_key="***"
)
```

① Импортируем модуль Python `tngri`

② Задаем параметры вашего бакета S3 (путь к файлу и ключи для доступа)

Функция `tngri.upload_s3` загружает файл из вашего бакета S3 в Tengri.

После этого данные из файла будут доступны для работы.

Расширение файла при этом может быть любым. Оно останется тем же, каким оно было в

начальном файле. Для работы с разными расширениями нужно использовать разные функции.

Работать с данными из файла в нашем примере можно с помощью функции `read_parquet`:

```
SELECT * FROM read_parquet('my_file.parquet');
```

При необходимости для загружаемого файла можно задать путь и имя внутри Tengri через опциональный параметр `filename` функции `tngri.upload_s3`:
`filename="new_path/new_name.parquet"`

Работа с данными из загруженных файлов

Покажем базовые варианты работы с данными из загруженных файлов типа `.csv` с использованием функции `read_csv`

- Проверяем, что загруженный файл `.csv` доступен:

```
SELECT * FROM read_csv('customer_country.csv');
```

- Создаем таблицу с данными из загруженного файла `.csv`:

```
CREATE OR REPLACE TABLE customer_country AS
    SELECT * FROM read_csv('customer_country.csv');
```

- Загружаем данные из файла `.csv` в существующую таблицу:

```
INSERT INTO customer_country
    SELECT * FROM read_csv('customer_country.csv');
```

Работа с данными из загруженных файлов разных расширений

Для работы с загруженными файлами других расширений нужно использовать соответствующие функции.

- Создаем таблицу с данными из загруженного файла `.parquet` с помощью функции `read_parquet`:

```
CREATE OR REPLACE TABLE customer_country AS
    SELECT * FROM read_parquet('customer_country.parquet');
```

- Создаем таблицу с данными из загруженного файла `.json` с помощью функции `read_json`:

```
CREATE OR REPLACE TABLE customer_country AS
```

```
    SELECT * FROM read_json('customer_country.json');
```

- Создаем таблицу с данными из загруженного файла `.xlsx` с помощью функции `read_xlsx`:

```
CREATE OR REPLACE TABLE customer_country AS
    SELECT * FROM read_xlsx('customer_country.xlsx');
```

Пример работы с данными из загруженного файла `.zip` можно увидеть здесь.

Автоматизация загрузки данных из множественных файлов

В этом примере покажем, как загрузить в Tengri данные, которые хранятся в виде множества файлов `.parquet` в хранилище S3.

С помощью модуля Python `boto3` выведем пути всех файлов `.parquet` в конкретном хранилище:

```
import boto3
import boto3.s3.transfer

path = 'Stage/<lake_path>/DataLake/2049/'
s3_client = boto3.client('s3',
                         endpoint_url='***',
                         aws_access_key_id='***',
                         aws_secret_access_key='***',
                         region_name='us-east-1')

response = s3_client.list_buckets()

for bucket in response['Buckets']:
    print(f'Bucket: {bucket["Name"]}')
    response = s3_client.list_objects_v2(
    Bucket=bucket["Name"],
    Prefix='Stage')

    for content in response.get('Contents', []):
        file_path = str(content['Key'])
        if path in file_path:
            print(file_path)
```

```
Bucket: prostore
Stage/<lake_path>/DataLake/2049/1.parquet
Stage/<lake_path>/DataLake/2049/2.parquet
Stage/<lake_path>/DataLake/2049/3.parquet
Stage/<lake_path>/DataLake/2049/4.parquet
Stage/<lake_path>/DataLake/2049/5.parquet
```

```
Bucket: s3bench-asusnode21
```

Попробуем сделать запрос SQL к данным из одного из этих файлов и убедимся, что запрос обрабатывается, а данные отображаются в выводе:

```
SELECT *
    FROM read_parquet('s3://prostore/Stage/<lake_path>/DataLake/2049/1.parquet')
LIMIT 5;
```

```
+-------------------------------------------------------------------------------------------
---------------------------------+-------+--------------------+--------------------------+
----------+----------+---------------------------+------------------------+
| f1
| f2    | f3                  | f4                          | jd1      | jd2      |
command_load_datetime      | command_load_session_id |
+-------------------------------------------------------------------------------------------
---------------------------------+-------+--------------------+--------------------------+
----------+----------+---------------------------+------------------------+
| {"id": 673, "age": 77, "name": "d1b7d724ef3a8451189fc47da62b9888", "email":
"b32c9c5d3b1768e7955144103d72c593@example.com"} | false | 0001-01-02 06:44:06 | <memory at
0x7f0bb8523880> | 7000-1-1 | 7000-2-1 | 2025-06-23 08:05:23.636063 | 195
|
+-------------------------------------------------------------------------------------------
---------------------------------+-------+--------------------+--------------------------+
----------+----------+---------------------------+------------------------+
| {"id": 221, "age": 34, "name": "4de35c863a2df55018eca39f130c0738", "email":
"3535ac8c3b04c33ac67ee70693707bdb@example.com"} | true  | 0001-01-02 02:59:53 | <memory at
0x7f0bb85219c0> | 7000-1-1 | 7000-2-1 | 2025-06-23 08:05:23.636063 | 195
|
+-------------------------------------------------------------------------------------------
---------------------------------+-------+--------------------+--------------------------+
----------+----------+---------------------------+------------------------+
| {"id": 612, "age": 27, "name": "94a027942dffccb4abe3923914592ba5", "email":
"10178786f4378e98f62be257c4215056@example.com"} | false | 0001-01-02 11:53:53 | <memory at
0x7f0bb8522c80> | 7000-1-1 | 7000-2-1 | 2025-06-23 08:05:23.636063 | 195
|
+-------------------------------------------------------------------------------------------
---------------------------------+-------+--------------------+--------------------------+
----------+----------+---------------------------+------------------------+
| {"id": 136, "age": 73, "name": "500522adeb682c1efcdce8af83610a27", "email":
"de507ff06310d2b35ea65c40f26f6a94@example.com"} | false | 0001-01-02 00:56:43 | <memory at
0x7f0bb8521fc0> | 7000-1-1 | 7000-2-1 | 2025-06-23 08:05:23.636063 | 195
|
+-------------------------------------------------------------------------------------------
---------------------------------+-------+--------------------+--------------------------+
----------+----------+---------------------------+------------------------+
| {"id": 150, "age": 67, "name": "7d283d7c78b5692f6162d8b65baee1ed", "email":
"df78cebca3c40d06e8b43ad8a7e98f56@example.com"} | true  | 0001-01-02 13:46:41 | <memory at
0x7f0bb8521600> | 7000-1-1 | 7000-2-1 | 2025-06-23 08:05:23.636063 | 195
|
```

```
+-------------------------------------------------------------------------------------------
--------------------------------+-------+--------------------+--------------------------+
----------+----------+---------------------------+------------------------+
```

Теперь нужно загрузить данные из всех файлов `.parquet` в одну таблицу.

Для этого сначала инициализируем эту таблицу — создадим ее, но не будем пока записывать в нее данные, чтобы сделать это на следующем шаге циклом.

Чтобы инициализировать таблицу, выполним такую команду с подзапросом `SELECT *` для одного из этих файлов и со значением параметра `LIMIT` — `0`:

```
CREATE OR REPLACE TABLE raw_dyntest AS
    SELECT *
        FROM read_parquet('s3://prostore/Stage/<lake_path>/DataLake/2049/1.parquet')
    LIMIT 0;
```

```
Done in 22.2 sec

+--------+
| status |
+--------+
| CREATE |
+--------+
```

Этот запрос создает таблицу с нужным нам набором столбцов и их типами, теперь в нее можно загружать данные в цикле.

Для этого в ячейке типа Python опишем цикл по именам всех файлов и в этом цикле с помощью функции `tngri.sql` выполним итеративную загрузку данных из каждого файла `.parquet` в инициализированную таблицу `raw_dyntest`.

В каждой итерации будем выводить количество строк в этой таблице, которое должно прирастать.

```
import tngri

for i in range(1,6):
    file_name = f"s3://prostore/Stage/<lake_path>/DataLake/2049/{i}.parquet"
    insert_sql = f"INSERT INTO raw_dyntest SELECT * FROM read_parquet('{file_name}')"
    print(insert_sql)
    tngri.sql(insert_sql)

    print(tngri.sql("SELECT count(*) FROM raw_dyntest"))
```

```
insert into raw_dyntest select * from
read_parquet('s3://prostore/Stage/<lake_path>/DataLake/2049/1.parquet')
shape: (1, 1)
+----------+
```

```
| column_0 |
| ---      |
| i64      |
+----------+
| 10000000 |
+----------+
insert into raw_dyntest select * from
read_parquet('s3://prostore/Stage/<lake_path>/DataLake/2049/2.parquet')
shape: (1, 1)
+----------+
| column_0 |
| ---      |
| i64      |
+----------+
| 20000000 |
+----------+
insert into raw_dyntest select * from
read_parquet('s3://prostore/Stage/<lake_path>/DataLake/2049/3.parquet')
shape: (1, 1)
+----------+
| column_0 |
| ---      |
| i64      |
+----------+
| 30000000 |
+----------+
insert into raw_dyntest select * from
read_parquet('s3://prostore/Stage/<lake_path>/DataLake/2049/4.parquet')
shape: (1, 1)
+----------+
| column_0 |
| ---      |
| i64      |
+----------+
| 40000000 |
+----------+
insert into raw_dyntest select * from
read_parquet('s3://prostore/Stage/<lake_path>/DataLake/2049/5.parquet')
shape: (1, 1)
+----------+
| column_0 |
| ---      |
| i64      |
+----------+
| 50000000 |
+----------+
```

Теперь выведем количество строк в заполненной таким образом таблице:

```
SELECT count(*) AS row_count
    FROM raw_dyntest;
```

```
+-----------+
| row_count |
+-----------+
| 50000000  |
+-----------+
```

Чтобы проверить, что запросы к этим данным на таком объеме (50 млн. строк) работают штатно, выведем распределение значений `true` и `false` в столбце `f2`:

```
SELECT f2, count(*) AS f2_cnt
    FROM raw_dyntest_ng
    GROUP BY f2;
```

```
+-------+----------+
| f2    | f2_cnt   |
+-------+----------+
| false | 24998577 |
+-------+----------+
| true  | 25001423 |
+-------+----------+
```

В однм из столбцов таблицы записаны данные в виде структуры JSON, и один из ключей этой структуры — `age` (*возраст*). Построим таблицу распределения значений возраста. Для этого используем функцию `json_extract`.

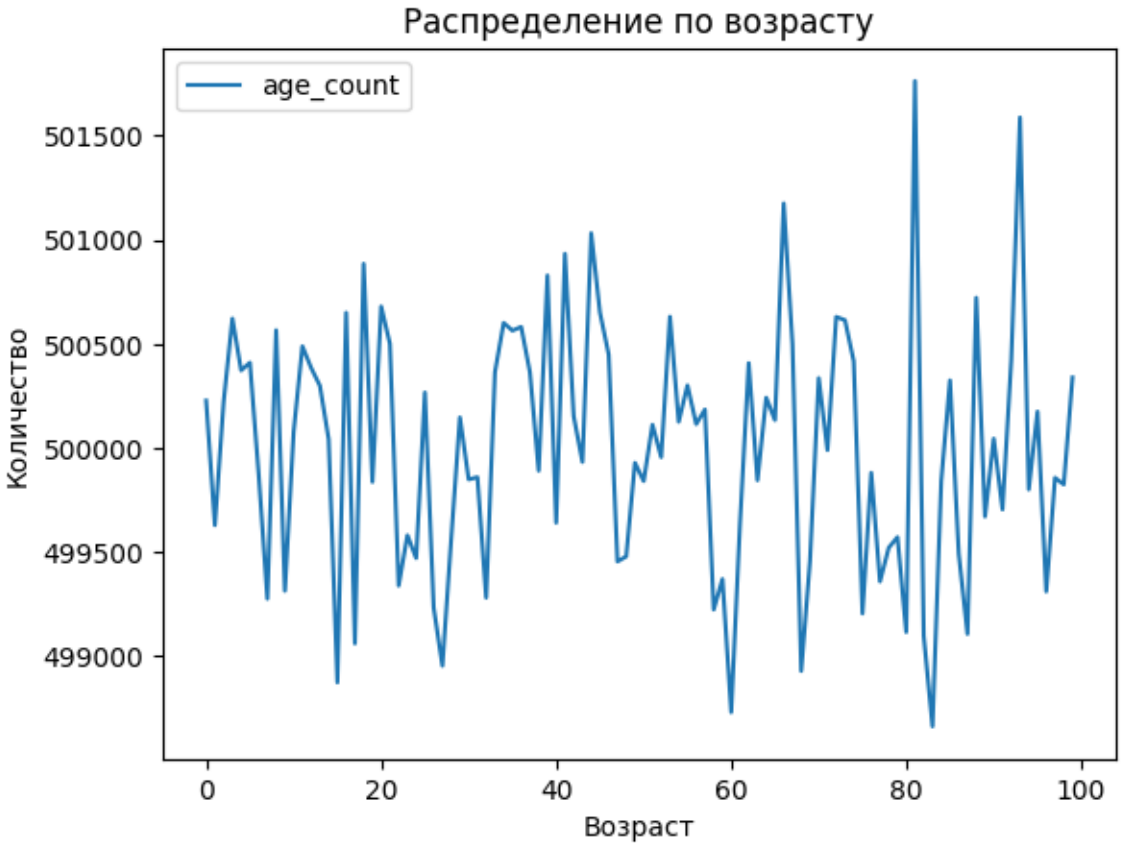
```
SELECT
    json_extract(f1, ['age'])[1]::INT AS age,
    count(*) AS age_count
    FROM raw_dyntest
    GROUP BY 1 ORDER BY 1;
```

```
Done in 9.9 sec

+-----+-----------+
| age | age_count |
+-----+-----------+
| 0   | 500228    |
+-----+-----------+
| 1   | 499629    |
+-----+-----------+
| 2   | 500222    |
+-----+-----------+
| 3   | 500622    |
+-----+-----------+
| 4   | 500373    |
+-----+-----------+
| 5   | 500410    |
```

```
+-----+-----------+
| 6   | 499876    |
+-----+-----------+
| 7   | 499276    |
+-----+-----------+
| 8   | 500566    |
+-----+-----------+
| 9   | 499314    |
+-----+-----------+
| ... | ...       |
+-----+-----------+
```

На основе этой таблицы построим график распределения значений по возрасту:

```
import matplotlib

plt = cell_output.plot(
    title='Распределение по возрасту',
    y='age_count',
    ylabel='Количество',
    x='age',
    xlabel='Возраст',
)

plt.get_figure()
```



Обращение к хранилищу Iceberg напрямую с помощью Python

В этом примере мы покажем, как обращаться к хранилищу Iceberg напрямую с помощью Python. Используя этот способ, вы сможете читать и редактировать существующие таблицы, создавать новые таблицы, в том числе на основе данных из существующих таблиц, и совершать другие операции непосредственно в хранилище Iceberg.

Для начала в ячейке типа SQL создадим тестовую таблицу с одним столбцом и запишем в строки 1 миллион чисел начиная с `1`. Для этого используем функции `unnest` и `generate_series`.

```
CREATE OR REPLACE TABLE demo.iceberg_test (numbers BIGINT);

INSERT INTO demo.iceberg_test (numbers)
    SELECT unnest(generate_series(1,1000000));
```

```
Done in 2.9 sec

+---------+
| Count   |
+---------+
| 1000000 |
+---------+
```

Выведем получившуюся таблицу:

```
SELECT * FROM demo.iceberg_test
    ORDER BY numbers
```

```
Done in 2.1 sec

+---------+
| numbers |
+---------+
| 1       |
+---------+
| 2       |
+---------+
| 3       |
+---------+
| 4       |
+---------+
| 5       |
+---------+
| ...     |
+---------+
```

```
999+ rows
```

Предположим, что на основе этой таблицы нам нужно создать новую таблицу, в которой должны быть все столбцы из исходной таблицы и один новый столбец. В новый столбец мы должны записать данные, вычисленные с помощью описанной на Python функции, передавая ей в качестве аргументов значения из столбцов исходной таблицы.

В качестве такой тестовой функции опишем функцию `check_odd`, которая возвращает строку `odd` для нечетных чисел и строку `even` — для четных.

Все это мы сделаем, обращаясь напрямую в Iceberg с помощью Python:

```
import tngri
import pyarrow
import pyiceberg
from contextlib import suppress
import pandas


# Задаем имя исходной таблицы
source_table_name = 'demo.iceberg_test'

# Задаем имя целевой таблицы
target_table_name = 'demo.iceberg_test_target'

# Задаем имя колонки в целевой таблице для записи новых данных
target_column = 'odd_or_even'

print(f'Source table: {source_table_name}')
print(f'Target table: {target_table_name}')

# Загружаем исходную таблицу
source_table = catalog.load_table(source_table_name)

# Создаем целевую таблицу, копируем схему из исходной
with suppress(Exception):
    catalog.drop_table(target_table_name)
sink = catalog.create_table(target_table_name, source_table.schema())

# Добавляем колону для записи новых данных (обязательно указать тип данных) ①
with sink.update_schema() as tx:
    tx.add_column(target_column, pyiceberg.schema.StringType())

# Тестовая функция для вычисления новых данных
def check_odd(num):
    if num % 2:
        return 'odd'
    else:
        return 'even'

# Делим исходную таблицу на батчи, чтобы не загружать в память целиком
table_batches = source_table.scan().to_arrow_batch_reader()
```

```
# Записываем новые данные по батчам
step = 0
for batch in table_batches:
    batch: pyarrow.RecordBatch

    step += 1
    print(f'Step: {step}')

    # Преобразуем батч в DataFrame
    part_df = batch.to_pandas()

    # Записываем новые данные в новую колонку DataFrame с тем же именем, что в таблице
    part_df[target_column] = part_df['numbers'].apply(lambda num: check_odd(num))

    # Выводим размер DataFrame с новыми данными
    print(f'Result dataframe shape: {part_df.shape}')

    # Преобразуем DataFrame обратно в таблицу
    sink_part = pyarrow.Table.from_pandas(part_df)

    # Аппендим данный батч в целевую таблицу
    sink.append(sink_part)

# Выводим длину целевой таблицы через scan().count()
print(f'Result table length: {sink.scan().count()}')
# Выводим длину целевой таблицы через tngri.sql
print(tngri.sql(f'SELECT count(*) FROM {target_table_name}')) ②
```

① Подробнее про типы данных в pyiceberg можно узнать здесь

② Подробное описание функции: `tngri.sql`

```
Done in 11.9 sec

Source table: demo.iceberg_test
Target table: demo.iceberg_test_target
Step: 1
Result dataframe shape: (16960, 2)
Step: 2
Result dataframe shape: (122880, 2)
Step: 3
Result dataframe shape: (122880, 2)
Step: 4
Result dataframe shape: (122880, 2)
Step: 5
Result dataframe shape: (122880, 2)
Step: 6
Result dataframe shape: (122880, 2)
Step: 7
Result dataframe shape: (122880, 2)
Step: 8
Result dataframe shape: (122880, 2)
```

```
Step: 9
Result dataframe shape: (122880, 2)
Result table length: 1000000
shape: (1, 1)
+----------+
| column_0 |
| ---      |
| i64      |
+----------+
| 1000000  |
+----------+
```

Проверим целевую таблицу. Для этого в ячейке типа SQL выведем ее первые пять строк, упорядочив строки по столбцу `numbers`:

```
SELECT * FROM demo.iceberg_test_target
    ORDER BY numbers
    LIMIT 5
```

```
Done in 2.1 sec

+---------+-------------+
| numbers | odd_or_even |
+---------+-------------+
| 1       | odd         |
+---------+-------------+
| 2       | even        |
+---------+-------------+
| 3       | odd         |
+---------+-------------+
| 4       | even        |
+---------+-------------+
| 5       | odd         |
+---------+-------------+
```

Аналитические сценарии

- Анализ данных из постов Instagram*
- Анализ геоданных спортивных трекеров
- Анализ текста на примере дата-сета FineWeb

Анализ данных из постов Instagram*

** Instagram принадлежит компании Meta, признанной экстремистской и запрещенной на территории РФ.*

В этом примере мы продемонстрируем, как можно проанализировать данные из датасета "instagram ads" на Kaggle. Данные в этом датасете представляют собой посты из социальной сети Instagram со всевозможными атрибутами (комментарии, лайки, информация об аудио и видео и прочее).

Загрузка исходных данных

Для начала загрузим данные. Это можно сделать либо в ручную через Мастер загрузки, либо с помощью, например, такого кода на Python.

Здесь мы задаем URL страницы датасета на Kaggle, проверяем, что скачанный файл является файлом `.zip`, и если это так, то распаковываем его содержимое во временную директорию, а затем загружаем в Tengri все распакованные файлы с помощью функции `tngri.upload_file`.

```
import urllib.request
import os
import zipfile
import tngri

zip_url = 'https://www.kaggle.com/api/v1/datasets/download/geraygench/instagram-ads'

urllib.request.urlretrieve(zip_url, 'data.zip')
temp_dir = 'temp'
uploaded_files = []

try:
    with zipfile.ZipFile('data.zip', 'r') as zObject:
        zObject.extractall(path=temp_dir)
    for file in os.listdir(temp_dir):
        cur_file = tngri.upload_file(os.path.join(temp_dir, file))
        uploaded_files.append(cur_file)
except:
    print('Not a valid zip file')

print('Uploaded files:')
i = 1
for file in uploaded_files:
    print(i, file)
    i += 1
```

```
Uploaded files:
1 niemkupghwwz.json
```

Теперь зададим имя таблицы и запишем в нее данные из загруженного файла с помощью функции `read_json`. На случай, если файлов загрузилось несколько, сделаем это в цикле.

```
table_name = 'instagram_source'

created = False
```

```
for file in uploaded_files:
    if not created:
        tngri.sql(f"create or replace table {table_name} as select * from
read_json('{file}')")
        created = True
    else:
        tngri.sql(f"insert into table {table_name} select * from read_json('{file}')")

print(f'Created table: {table_name}')
count = tngri.sql(f'select count(*) from {table_name}')
print(f'Created table row count: {count[0,0]}')
```

```
Created table: instagram_source
Created table row count: 2800
```

Выведем первые 5 строк загруженных данных, чтобы зрительно проверить, что загрузка прошла успешно.

```
SELECT * FROM instagram_source
    LIMIT 5;
```

```
Done in 1.1 sec
+-----------------------------------------------+---------------------+---------
+-------------+-----+
| inputUrl                                      | id                  | type    | shortCode
| ... |
+-----------------------------------------------+---------------------+---------
+-------------+-----+
| https://www.instagram.com/fjallravenofficial/ | 3351813129396710000 | Video   | C6ECEhTLlj7
| ... |
+-----------------------------------------------+---------------------+---------
+-------------+-----+
| https://www.instagram.com/fjallravenofficial/ | 3335943316421300000 | Video   | C5LpsWOIUc8
| ... |
+-----------------------------------------------+---------------------+---------
+-------------+-----+
| https://www.instagram.com/fjallravenofficial/ | 3319217982042080000 | Sidecar | C4QOysvtPUU
| ... |
+-----------------------------------------------+---------------------+---------
+-------------+-----+
| https://www.instagram.com/fjallravenofficial/ | 3372181927023210000 | Sidecar | C7MZZiWNffn
| ... |
+-----------------------------------------------+---------------------+---------
+-------------+-----+
| https://www.instagram.com/fjallravenofficial/ | 3369947094264730000 | Video   | C7EdQcJgkwr
| ... |
+-----------------------------------------------+---------------------+---------
+-------------+-----+
```

Теперь создадим таблицу `raw_instagram_post` и загрузим в нее эти данные.

Для тех столбцов, в которых мы обнаружили вложенную структуру JSON, укажем тип `JSON`. Для некоторых столбцов укажем удобные нам новые имена и нужные типы данных (например, `TIMESTAMP`). Зададим ограничение, что значение идентификатора `ig_post_id` не должно быть пустым, чтобы отбросить потенциальные невалидные фрагменты данных.

```
CREATE OR REPLACE TABLE raw_instagram_post AS
SELECT
    inputUrl,
    id AS ig_post_id,
    TYPE,
    shortCode,
    caption,
    hashtags,
    mentions,
    url,
    dimensionsHeight,
    dimensionsWidth,
    commentsCount,
    firstComment,
    displayUrl,
    images,
    videoUrl,
    alt,
    likesCount,
    videoViewCount,
    videoPlayCount,
    timestamp::TIMESTAMP AS post_ts,
    ownerFullName,
    ownerUsername,
    ownerId,
    productType,
    videoDuration,
    isSponsored,
    isPinned,
    locationName,
    locationId,
    error,
    description,
    paidPartnership,
    sponsors::JSON AS sponsors,
    taggedUsers::JSON AS taggedUsers,
    musicInfo::JSON AS musicInfo,
    coauthorProducers::JSON AS coauthorProducers,
    latestComments::JSON AS latestComments,
    childPosts::JSON AS childPosts
FROM instagram_source
WHERE ig_post_id IS NOT NULL;
```

```
Done in 1.7 sec
```

```
+--------+
| status |
+--------+
| CREATE |
+--------+
```

Выведем первые 5 строк таблицы, чтобы зрительно проверить, что данные в ней соответствуют нашим ожиданиям.

```
SELECT * FROM raw_instagram_post
    LIMIT 5;
```

```
Done in 1.2 sec
+-----------------------------------------------+--------------------+---------
+-------------+-----+
| inputurl                                      | ig_post_id         | type    | shortcode
| ... |
+-----------------------------------------------+--------------------+---------
+-------------+-----+
| https://www.instagram.com/fjallravenofficial/ | 3351813129396710000 | Video   | C6ECEhTLlj7
| ... |
+-----------------------------------------------+--------------------+---------
+-------------+-----+
| https://www.instagram.com/fjallravenofficial/ | 3335943316421300000 | Video   | C5LpsWOIUc8
| ... |
+-----------------------------------------------+--------------------+---------
+-------------+-----+
| https://www.instagram.com/fjallravenofficial/ | 3319217982042080000 | Sidecar | C4QOysvtPUU
| ... |
+-----------------------------------------------+--------------------+---------
+-------------+-----+
| https://www.instagram.com/fjallravenofficial/ | 3372181927023210000 | Sidecar | C7MZZiWNffn
| ... |
+-----------------------------------------------+--------------------+---------
+-------------+-----+
| https://www.instagram.com/fjallravenofficial/ | 3369947094264730000 | Video   | C7EdQcJgkwr
| ... |
+-----------------------------------------------+--------------------+---------
+-------------+-----+
```

Подготовка данных

Теперь проанализируем вложенную структуру JSON в столбце `musicinfo`, использовав для этого функции `unnest` и `json_keys`:

```
SELECT
    unnest(keys) AS musicinfo_keys,
    keys_count
```

```
FROM
    (SELECT
        json_keys(musicinfo) AS keys,
        count(*) AS keys_count
     FROM raw_instagram_post
     WHERE musicinfo IS NOT NULL
     GROUP BY 1) t;
```

```
Done in 1.1 sec
+--------------------------+------------+
| musicinfo_keys           | keys_count |
+--------------------------+------------+
| artist_name              | 1176       |
+--------------------------+------------+
| song_name                | 1176       |
+--------------------------+------------+
| uses_original_audio      | 1176       |
+--------------------------+------------+
| should_mute_audio        | 1176       |
+--------------------------+------------+
| should_mute_audio_reason | 1176       |
+--------------------------+------------+
| audio_id                 | 1176       |
+--------------------------+------------+
```

Извлечем данные из структуры JSON с помощью функции `json_extract` и запишем их в новую таблицу `silver_instagram_musicinfo`, столбцы которой — это ключи из исходной структуры JSON (внутри `musicinfo`), которые мы увидели на предыдущем шаге, а в столбец `ig_post_id` запишем id из исходной таблицы `instagram_post`.

Обратите внимание, что функцию `json_extract` мы используем здесь не более одного раза для одного столбца исходной таблицы. Это позволяет сэкономить время и вычислительные мощности, так как после однократного извлечения этих данных обращение к ним будет происходить практически моментально.

```
CREATE OR REPLACE TABLE silver_instagram_musicinfo AS
SELECT DISTINCT ig_post_id,
    arr[1] AS 'artist_name',
    arr[2] AS 'song_name',
    arr[3]::bool AS 'uses_original_audio',
    arr[4]::bool AS 'should_mute_audio',
    arr[5] AS 'should_mute_audio_reason',
    arr[6]::bigint AS 'audio_id'
FROM (SELECT ig_post_id, musicinfo,
    json_extract(musicinfo, ['artist_name',
                             'song_name',
                             'uses_original_audio',
                             'should_mute_audio',
                             'should_mute_audio_reason',
```

```
                            'audio_id'])
                            arr
    FROM raw_instagram_post
    WHERE musicinfo IS NOT NULL
) e;
```

```
Done in 1.1 sec
+--------+
| status |
+--------+
| CREATE |
+--------+
```

Выведем первые пять строк из получившейся таблицы, чтобы зрительно проверить данные в ней.

```
SELECT * FROM silver_instagram_musicinfo
    LIMIT 5;
```

```
Done in 1 sec
+---------------------+----------------------+--------------------------
+---------------------+-------------------+--------------------------+------------------+
| ig_post_id          | artist_name          | song_name                |
uses_original_audio | should_mute_audio | should_mute_audio_reason | audio_id         |
+---------------------+----------------------+--------------------------
+---------------------+-------------------+--------------------------+------------------+
| 3361389062866375005 | "fjallravenofficial" | "Original audio"         | true
| false             | ""                     | 1091589181917200 |
+---------------------+----------------------+--------------------------
+---------------------+-------------------+--------------------------+------------------+
| 3362166784222944940 | "Lyle Workman"       | "Sun Beyond the Darkness" | false
| false             | ""                     | 700744903595866  |
+---------------------+----------------------+--------------------------
+---------------------+-------------------+--------------------------+------------------+
| 3370702174072339576 | "victoriassecret"    | "Original audio"         | true
| false             | ""                     | 969258887793092  |
+---------------------+----------------------+--------------------------
+---------------------+-------------------+--------------------------+------------------+
| 3365659145996406849 | "victoriassecret"    | "Original audio"         | true
| false             | ""                     | 1856092344873079 |
+---------------------+----------------------+--------------------------
+---------------------+-------------------+--------------------------+------------------+
| 3367926439711140761 | "asos"               | "Original audio"         | true
| false             | ""                     | 1428332318047391 |
+---------------------+----------------------+--------------------------
+---------------------+-------------------+--------------------------+------------------+
```

Теперь посмотрим на столбец исходной таблицы с информацией о затэганных пользователях `taggedUsers`. Обратим внимание, что в нем данные представлены в виде массивов. Используя

функции `unnest` и `json_extract`, создадим таблицу `silver_instagram_taggeduser`:

```
CREATE OR REPLACE TABLE silver_instagram_taggeduser AS
SELECT DISTINCT
    ig_post_id,
    arr[1]::varchar AS 'full_name',
    arr[2]::bigint AS 'user_id',
    arr[3]::bool AS 'is_verified',
    arr[4] AS 'profile_pic_url',
    arr[5] AS 'username'
FROM
    (SELECT
        ig_post_id,
        json_extract(users, ['full_name',
                        'id','is_verified',
                        'profile_pic_url',
                        'username']) AS arr
    FROM
        (SELECT
            ig_post_id,
            unnest(json_extract(taggedUsers, '[*]')) AS users
        FROM raw_instagram_post
        WHERE taggedUsers IS NOT NULL)
    un) t;
```

```
Done in 1 sec
+--------+
| status |
+--------+
| CREATE |
+--------+
```

Выведем первые 5 строк созданной таблицы `silver_instagram_taggeduser` и визуально проверим данные в ней.

```
SELECT * FROM silver_instagram_taggeduser
    LIMIT 5;
```

```
Done in 1 sec
+---------------------+-------------------+-----------+-------------
+-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
---------------------------+--------------------+
| ig_post_id          | full_name         | user_id   | is_verified | profile_pic_url
| username           |
+---------------------+-------------------+-----------+-------------
+-----------------------------------------------------------------------------------------
```

```
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
---------------------------+--------------------+
| 3368467612563246843 | "Candice"          | 43114731  | true        |
"https://scontent.cdninstagram.com/v/t51.2885-
19/436426783_716232097252223_2080875243016028171_n.jpg?stp=dst-
jpg_s150x150&_nc_ht=scontent.cdninstagram.com&_nc_cat=1&_nc_ohc=VZz3Y6G6kCgQ7kNvgEfJgX3&edm=A
Ps17CUBAAAA&ccb=7-5&oh=00_AYDeE3YwxCQxVEK-
3DLy0AK8cSmaFu3SVuwsUluh0pQ2Hg&oe=66526CB7&_nc_sid=10d13b"   | "candiceswanepoel" |
+--------------------+------------------+-----------+-------------
+----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
---------------------------+--------------------+
| 3371457031159064630 | "Brittany Xavier" | 739181077 | true        |
"https://scontent.cdninstagram.com/v/t51.2885-
19/299498992_480162533527636_8112607220645623732_n.jpg?stp=dst-
jpg_s150x150&_nc_ht=scontent.cdninstagram.com&_nc_cat=104&_nc_ohc=D83lh8W9o-
EQ7kNvgEAZo1_&edm=APs17CUBAAAA&ccb=7-5&oh=00_AYCI_FerVMUvMEltLhAaANlfFdtezWw90dIOCff-
oNuiLQ&oe=66526DD9&_nc_sid=10d13b" | "brittanyxavier"   |
+--------------------+------------------+-----------+-------------
+----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
---------------------------+--------------------+
| 3350535229969664632 | "dev scudds"       | 200671402 | false       |
"https://scontent.cdninstagram.com/v/t51.2885-
19/316332449_539815587596795_7700370950656909913_n.jpg?stp=dst-
jpg_s150x150&_nc_ht=scontent.cdninstagram.com&_nc_cat=111&_nc_ohc=yt51b9U7YQcQ7kNvgGr0PqA&edm
=APs17CUBAAAA&ccb=7-
5&oh=00_AYC_DbQKZ56A_C_kK4oCamtB10dpgXWxNr9dupqkTHIKZA&oe=665259D4&_nc_sid=10d13b" |
"devscudds"        |
+--------------------+------------------+-----------+-------------
+----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
---------------------------+--------------------+
| 3355576548572106408 | "Roxie Nafousi"   | 182064334 | true        |
"https://scontent.cdninstagram.com/v/t51.2885-
19/432688376_299228906523099_7608649815950644359_n.jpg?stp=dst-
jpg_s150x150&_nc_ht=scontent.cdninstagram.com&_nc_cat=1&_nc_ohc=mNDTJLE0sakQ7kNvgFoinRd&edm=A
Ps17CUBAAAA&ccb=7-5&oh=00_AYDyLDXyYkSqF7jeJ7SDKO-
ATM7WbbH8OYqhw_GOjiGygQ&oe=66526921&_nc_sid=10d13b"   | "roxienafousi"     |
+--------------------+------------------+-----------+-------------
+----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------
---------------------------+--------------------+
| 3369046462609537802 | "ASOS MAN"         | 640874843 | true        |
"https://scontent.cdninstagram.com/v/t51.2885-
19/244277018_902025494057602_3929487210907869376_n.jpg?stp=dst-
jpg_s150x150&_nc_ht=scontent.cdninstagram.com&_nc_cat=110&_nc_ohc=5aRpL0ZaRW8Q7kNvgFG0_9g&edm
=APs17CUBAAAA&ccb=7-5&oh=00_AYBmZ6qVQtwRziMSsl8tGtlX7u4-RnYXa-
```

```
MxQIaIEKyc7Q&oe=66527A51&_nc_sid=10d13b" | "asos_man"         |
+--------------------+------------------+-----------+-------------
+-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
---------------------------+-------------------+
```

Теперь посмотрим на столбец исходной таблицы с информацией о комментариях `latestComments`. В нем данные представлены в виде массива комментариев. Чтобы узнать, какие ключи встречаются внутри этих массивов, используем функции `unnest`, `json_keys` и `json_extract` и выполним такой запрос:

```
SELECT
    unnest(keys) AS comment_keys
FROM
    (SELECT
        json_keys(comments) AS keys,
    FROM
        (SELECT
            ig_post_id,
            unnest(json_extract(latestComments, '[*]')) AS comments
        FROM raw_instagram_post
        WHERE latestComments IS NOT NULL) un
    GROUP BY 1) t;
```

```
Done in 1 sec
+--------------------+
| comment_keys       |
+--------------------+
| id                 |
+--------------------+
| text               |
+--------------------+
| ownerUsername      |
+--------------------+
| ownerProfilePicUrl |
+--------------------+
| timestamp          |
+--------------------+
| likesCount         |
+--------------------+
| repliesCount       |
+--------------------+
| replies            |
+--------------------+
```

Извлечем комментарии в таблицу `silver_instagram_comments` аналогичным образом, как мы это сделали выше для пользователей и исполнителей.

Обратим внимание, что в этой таблице `likesCount` и `repliesCount` — это количество лайков и ответов на данный комментарий (а не на исходный пост).

```
CREATE OR REPLACE TABLE silver_instagram_comments as
    SELECT DISTINCT
        ig_post_id,
        arr[1]::bigint AS comment_id,
        arr[2] AS 'text',
        arr[3] AS 'ownerUsername',
        arr[4] AS 'ownerProfilePicUrl',
        arr[5]::timestamp AS 'comment_ts',
        arr[6]::int AS 'likesCount',
        arr[7]::int AS 'repliesCount',
        arr[8] AS 'replies'
FROM (
SELECT
    ig_post_id,
    json_extract(comments, ['id',
                            'text',
                            'ownerUsername',
                            'ownerProfilePicUrl',
                            'timestamp',
                            'likesCount',
                            'repliesCount',
                            'replies']) AS arr
    FROM
        (SELECT
            ig_post_id,
            unnest(json_extract(latestComments, '[*]')) AS comments
        FROM raw_instagram_post
        WHERE latestComments IS NOT NULL)
    un) t;
```

```
Done in 1 sec
+--------+
| status |
+--------+
| CREATE |
+--------+
```

Проверим получившуюся таблицу:

```
SELECT * FROM silver_instagram_comments
    LIMIT 5;
```

```
Done in 1 sec
+--------------------+------------------
+-------------------------------------------------------------------
```

```
+---------------------------------+-----+
| ig_post_id          | comment_id        | text
| ownerusername                   | ... |
+---------------------+-------------------
+-------------------------------------------------------------------
+---------------------------------+-----+
| 3351813129396705531 | 18025641284105656 | "Your work apron is cool..\nDo Fjalraven sell
them to the public?" | "dai_stick"                      | ... |
+---------------------+-------------------
+-------------------------------------------------------------------
+---------------------------------+-----+
| 3367108632665594085 | 18010752380180096 | "Well done "
| "alpencore"                     | ... |
+---------------------+-------------------
+-------------------------------------------------------------------
+---------------------------------+-----+
| 3362086677098626407 | 18041534590828292 | "Still made in cheap labour countries"
| "leineleineleineleineleineleine" | ... |
+---------------------+-------------------
+-------------------------------------------------------------------
+---------------------------------+-----+
| 3362086677098626407 | 18028710974031364 | ""
| "arifwidie26"                   | ... |
+---------------------+-------------------
+-------------------------------------------------------------------
+---------------------------------+-----+
| 3361389062866375005 | 18012200105461376 | "Classic description of type 2 fun"
| "itskate_g"                     | ... |
+---------------------+-------------------
+-------------------------------------------------------------------
+---------------------------------+-----+
```

Результат 1: Популярность исполнителей

Теперь используем собранные выше данные, чтобы построить таблицу, в которой каждая строка соответствует одному исполнителю, а в столбцах представлена различная информация, так или иначе свидетельствующая о популярности этого исполнителя:

- `songs` — количество композиций данного исполнителя
- `ig_posts` — количество постов с композициями данного исполнителя
- `comments` — количество комментариев в постах с композициями данного исполнителя
- `tagged_users` — количество затэганных пользователей в постах с композициями данного исполнителя
- `likesCount` — количество лайков в постах с композициями данного исполнителя
- `repliesCount` — количество реплаев в постах с композициями данного исполнителя

Для этого используем выражение `LEFT JOIN`.

Упорядочим финальную таблицу по третьему столбцу (`ig_posts`) по убыванию.

```
SELECT artist_name,
    count(distinct song_name) AS songs,
    count(distinct im.ig_post_id) AS ig_posts,
    count(distinct comment_id) AS comments,
    count(distinct user_id) AS tagged_users,
    sum(likesCount) AS likesCount,
    sum(repliesCount) AS repliesCount
FROM silver_instagram_musicinfo im
    LEFT JOIN silver_instagram_taggeduser tu ON tu.ig_post_id=im.ig_post_id
    LEFT JOIN silver_instagram_comments ic ON ic.ig_post_id =im.ig_post_id
GROUP BY 1 ORDER BY 3 DESC;
```

```
Done in 1.6 sec
+------------------+-------+----------+----------+-------------+-----------+-------------
+
| artist_name      | songs | ig_posts | comments | tagged_users | likescount | repliescount
|
+------------------+-------+----------+----------+-------------+-----------+-------------
+
| "telfarglobal"   | 9     | 136      | 985      | 120          | 634        | 156
|
+------------------+-------+----------+----------+-------------+-----------+-------------
+
| "everlane"       | 1     | 95       | 532      | 6            | 513        | 65
|
+------------------+-------+----------+----------+-------------+-----------+-------------
+
| "asos"           | 3     | 93       | 503      | 60           | 583        | 85
|
+------------------+-------+----------+----------+-------------+-----------+-------------
+
| "victoriassecret" | 1     | 83       | 686      | 49           | 185        | 16
|
+------------------+-------+----------+----------+-------------+-----------+-------------
+
| "summerfridays"  | 2     | 65       | 574      | 48           | 173        | 18
|
+------------------+-------+----------+----------+-------------+-----------+-------------
+
| ...              | ...   | ...      | ...      | ...          | ...        | ...
|
+------------------+-------+----------+----------+-------------+-----------+-------------
+
```

Эту таблицу можно сохранить в формат `.xlsx`, нажав на кнопку **[Download output as XLSX]** на данной ячейке. После этого в любом приложении для просмотра файлов `.xlsx` можно упорядочить собранные данные по любому столбцу, наложить дополнительные фильтры, скопировать данные или их часть для вставки в аналитические отчеты.

Сохраненный таким образом файл можно скачать здесь.

Результат 2: Распределение комментариев и событий в них во времени

Проанализируем распределение комментариев и событий в них (лайков и реплаев) в зависимости от времени, прошедшего от публикации поста до публикации данного комментария к нему — "свежести" комментария относительно поста, к которому он относится.

Для этого построим таблицу, в которой каждая строка будет соответствовать всем таким парам "пост — комментарий к нему", что между ними прошло 0 дней, 1 день и так далее до 9. И для всех таких пар выведем в столбцы:

- `comments` — количество таких комментариев
- `likescount` — количество лайков у таких комментариев
- `repliescount` — количество реплаев у таких комментариев

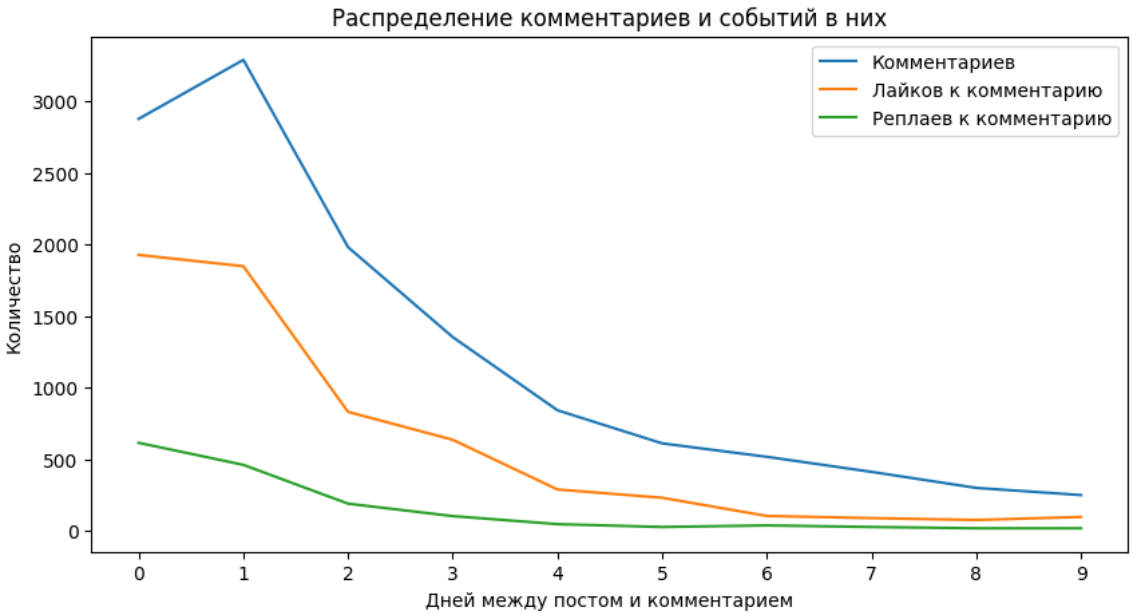
```
SELECT
    (comment_ts::DATE - post_ts::DATE)::INT AS days_till_comment,
    count(DISTINCT comment_id) AS comments,
    sum(c.likesCount)::int AS likesCount,
    sum(c.repliesCount)::int AS repliesCount
FROM raw_instagram_post p
LEFT JOIN silver_instagram_comments c ON p.ig_post_id=c.ig_post_id
GROUP BY 1
ORDER BY 1
LIMIT 10;
```

```
Done in 1.7 sec
+-------------------+----------+------------+--------------+
| days_till_comment | comments | likescount | repliescount |
+-------------------+----------+------------+--------------+
| 0                 | 2876     | 1926       | 614          |
+-------------------+----------+------------+--------------+
| 1                 | 3288     | 1847       | 460          |
+-------------------+----------+------------+--------------+
| 2                 | 1980     | 830        | 189          |
+-------------------+----------+------------+--------------+
| 3                 | 1352     | 635        | 102          |
+-------------------+----------+------------+--------------+
| 4                 | 841      | 288        | 46           |
+-------------------+----------+------------+--------------+
| 5                 | 610      | 230        | 26           |
+-------------------+----------+------------+--------------+
| 6                 | 516      | 103        | 37           |
+-------------------+----------+------------+--------------+
| 7                 | 411      | 88         | 26           |
+-------------------+----------+------------+--------------+
| 8                 | 299      | 75         | 17           |
+-------------------+----------+------------+--------------+
```

```
| 9                 | 249      | 96         | 17           |
+-------------------+----------+------------+--------------+
```

Теперь в ячейке типа Python построим график по данным из этой таблицы. Для этого воспользуемся модулем Python `matplotlib`. Чтобы взять данные из предыдущей ячейки, используем переменную `cell_output`.

```
import matplotlib

plt = cell_output.plot(
    title='Распределение комментариев и событий в них',
    y=[
        'comments',
        'likescount',
        'repliescount',
    ],
    label = ['Комментариев', 'Лайков к комментарию', 'Реплаев к комментарию'],
    ylabel='Количество',
    x='days_till_comment',
    xlabel = 'Дней между постом и комментарием',
    figsize=(10,5),
)
plt.set_xticks(cell_output['days_till_comment'])
plt.get_figure()
```



На графике хорошо видно, что количество комментариев к посту в первый день возрастает, а потом плавно снижается. Также видно, каким образом снижается активность внутри самих комментариев (лайки и реплаи к комментариям) в зависимости от устаревания комментария относительно даты поста.

Анализ геоданных спортивных трекеров

В этом примере мы покажем, как с помощью Tengri можно анализировать геоданные из

спортивных трекеров, таких как Garmin, Strava и других.

Наиболее популярные форматы записи и хранения геоданных в спортивных трекерах — `.gpx` и `.fit`. Помимо GPS-координат точек (долгота, широта, высота над уровнем моря) в этих форматах сохраняется также время прохождения спортсменом данной точки и другая информация о спортсмене и его оборудовании (пульс, мощность, частота вращения педалей, скорость, температура и прочее).

В качестве дата-сета для этого примера мы возьмем выгрузку архива всех данных из одного профиля Strava по состоянию на 9 сентября 2025. Как сделать такую выгрузку, можно узнать здесь.

Выгрузка данных из Strava представляет собой архив, в котором среди прочего есть папка `activities` со всеми активностями данного пользователя в виде файлов `.gpx` или `.fit` (в зависимости от использованных для записи треков устройств) и файл `activities.csv`, в котором представлена дополнительная информация из сервиса Strava (название активности, тип активности, комментарии, различные вычисленные показатели и прочее).

Всего в нашем дата-сете 2190 активностей (треков), причем часть из них представлена в виде файлов `.gpx`, а другая часть — в виде файлов `.fit`.

Загрузка исходных данных

Импортируем необходимые модули Python:

```
import os
import zipfile
import tngri
import pandas as pd
import gzip
import shutil
import datetime
import requests
import s3fs

os.system('pip install gpxpy')
os.system('pip install fitdecode')
os.system('pip install tilemapbase')
import gpxpy
import fitdecode
import tilemapbase
```

Для начала загрузим данные. Для этого воспользуемся мастером загрузки и выберем в нем файл zip на локальном компьютере с файлами треков. Результатом работы мастера загрузки будет путь к загруженному файлу вида: `"prostore/Stage/abogdanov/0f89889ad699_all_activities.zip"`. Этот путь мы используем для чтения загруженного файла в ячейке Python через модуль `s3fs`:

```
start = datetime.datetime.now()
temp_dir = 'temp'

# Чистим временную папку (чтобы при каждом запуске данные заливались с нуля)
```

```
if os.path.exists(temp_dir) and os.path.isdir(temp_dir):
    shutil.rmtree(temp_dir)
    print(f'Folder "{temp_dir}" cleaned')

# Читаем файл zip, загруженный через мастер загрузки
fs = s3fs.S3FileSystem()
with fs.open("prostore/Stage/abogdanov/0f89889ad699_all_activities.zip", "rb") as f:
    content = zipfile.ZipFile(f)

    # Распаковываем zip
    content.extractall(path=temp_dir)

downloaded = len(os.listdir(temp_dir))
print(f'Downloaded files total: {downloaded}')

# Распаковываем .gz
unpacked = 0
for file in os.listdir(temp_dir):
    file_path = os.path.join(temp_dir, file)
    if file.endswith('.gz'):
        unpacked += 1
        with gzip.open(file_path, 'rb') as f_in:
            with open(file_path[:-3], 'wb') as f_out:
                shutil.copyfileobj(f_in, f_out)

print(f'Unpacked .gz files total: {unpacked}')
print(f'Time from start: {str(datetime.datetime.now() - start)}')
```

```
Downloaded files total: 2190
Unpacked .gz files total: 1765
Time from start: 0:00:03.692377
```

Теперь приступим к парсингу файлов треков.

1. Зададим имена колонок в двух создаваемых таблицах — для всех треков и для всех точек из всех треков
2. Опишем вспомогательные функции
3. Пройдем циклом по всем файлам активностей и сделаем парсинг данных из них, причем парсинг будет различаться для файлов `.gpx` и `.fit`. Для парсинга используем удобные модули Python `gpxpy` и `fitdecode`.
4. Запишем сформированные DataFrame в таблицы с помощью функции `tngri.sql`

```
start = datetime.datetime.now()

①
# Колонки для таблицы треков
tracks_dict = {
    'track_id':[],
    'track_name':[],
```

```
    'track_length_km':[],
    'track_first_coord_lat':[],
    'track_first_coord_lon':[],
    'country':[]
    }
# Колонки для таблицы точек
points_dict = {
    'track_id':[],
    'latitude':[],
    'longitude':[],
    'elevation':[],
    'heart_rate':[],
    'temperature':[],
    'cadence':[],
    'power':[],
    'time':[],
    'speed':[]
    }

②
# Получение extension из точки для .gpx
def ext_from_point(point, type):
    res = pd.NA
    for ext in point.extensions:
        ext_list = [el.text for el in ext if type in el.tag]
        if len(ext_list) == 1:
            res = ext_list[0]
    return res

# Получение мощности из точки для .gpx
def power_from_point(point):
    res = pd.NA
    for ext in point.extensions:
        if ext.tag == 'power':
            res = ext.text
    return res

# Преобразование координат для .fit
def convert_coord(coord):
    if pd.isna(coord):
        return coord
    else:
        return coord/((2**32)/360)

# Преобразование скорости из м/с в км/ч
def convert_speed(speed):
    if pd.isna(speed):
        return speed
    else:
        return speed*3.6

# Определение страны по координатам
def get_country(coords):
```

```
    location = geolocator.reverse(f'{coords[0]}, {coords[1]}')
    if not location is None:
        return location.raw['address']['country']
    else:
        return 'Unknown'

③
# Начинаем парсинг gpx и fit
files_done = 0
for file in os.listdir(temp_dir):
    file_path = os.path.join(temp_dir, file)

    if file.endswith('.gpx'):
        files_done += 1
        with open(file_path) as f:
            gpx = gpxpy.parse(f)
            if not len(gpx.tracks) == 1:
                print(f'Not 1 track in file: {file_path}')

            cur_length_2d_km = gpx.tracks[0].segments[0].length_2d()/1000
            track_first_lat = gpx.tracks[0].segments[0].points[0].latitude
            track_first_lon = gpx.tracks[0].segments[0].points[0].longitude

            tracks_dict['track_first_coord_lat'].append(track_first_lat)
            tracks_dict['track_first_coord_lon'].append(track_first_lon)
            tracks_dict['track_id'].append(file)
            tracks_dict['track_name'].append(gpx.tracks[0].name)
            tracks_dict['track_length_km'].append(round(cur_length_2d_km, 2))
            tracks_dict['country'].append(get_country((track_first_lat, track_first_lon)))

            for point in gpx.tracks[0].segments[0].points:
                points_dict['track_id'].append(file)
                points_dict['latitude'].append(point.latitude)
                points_dict['longitude'].append(point.longitude)
                points_dict['elevation'].append(point.elevation)
                points_dict['heart_rate'].append(ext_from_point(point, 'hr'))
                points_dict['temperature'].append(ext_from_point(point, 'atemp'))
                points_dict['cadence'].append(ext_from_point(point, 'cad'))
                points_dict['power'].append(power_from_point(point))
                points_dict['time'].append(point.time)
                # В gpx нету моментальной скорости
                points_dict['speed'].append(pd.NA)

    elif file.endswith('.fit'):
        files_done += 1
        track_first_lat = 0
        track_first_lon = 0
        with fitdecode.FitReader(file_path) as fit_file:
            tracks_dict['track_id'].append(file)
            tracks_dict['track_name'].append(pd.NA)
            distance = 0
            for frame in fit_file:
                if not isinstance(frame, fitdecode.records.FitDataMessage):
```

```
                    continue
                if not frame.name == 'record':
                    continue
                if not (frame.has_field('position_lat') and
frame.has_field('position_long')):
                    continue
                cur_lat = convert_coord(frame.get_value('position_lat', fallback=pd.NA))
                cur_lon = convert_coord(frame.get_value('position_long', fallback=pd.NA))

                points_dict['track_id'].append(file)
                points_dict['latitude'].append(cur_lat)
                points_dict['longitude'].append(cur_lon)
                points_dict['elevation'].append(frame.get_value('altitude', fallback=pd.NA))
                #points_dict['heart_rate'].append(frame.get_value('heart_rate',
fallback=pd.NA))
                points_dict['heart_rate'].append(pd.NA)
                points_dict['temperature'].append(frame.get_value('temperature',
fallback=pd.NA))
                #points_dict['cadence'].append(frame.get_value('cadence', fallback=pd.NA))
                points_dict['cadence'].append(pd.NA)
                points_dict['power'].append(frame.get_value('power', fallback=pd.NA))
                points_dict['time'].append(frame.get_value('timestamp'))
                points_dict['speed'].append(convert_speed(frame.get_value('enhanced_speed',
fallback=pd.NA)))

                cur_dist = frame.get_value('distance', fallback=pd.NA)
                if type(cur_dist) == float:
                    distance = round((cur_dist/1000), 2)
                if track_first_lat == 0 and not pd.isna(cur_lat):
                    track_first_lat = cur_lat
                if track_first_lon == 0 and not pd.isna(cur_lon):
                    track_first_lon = cur_lon
            tracks_dict['track_length_km'].append(distance)
            tracks_dict['track_first_coord_lat'].append(track_first_lat)
            tracks_dict['track_first_coord_lon'].append(track_first_lon)
            tracks_dict['country'].append(get_country((track_first_lat, track_first_lon)))

    if files_done > 0 and files_done % 100 == 0 and file.endswith(('.fit', '.gpx')):
        print(f'Files done: {files_done}/{downloaded} \
        Points: {len(points_dict['track_id'])} \
        Time from start: {str(datetime.datetime.now() - start)}')

# Пишем собранные словари в DataFrame
df_tracks = pd.DataFrame.from_dict(tracks_dict)
df_points = pd.DataFrame.from_dict(points_dict)

df_points['time_text'] = df_points['time'].astype(str)
df_points = df_points.drop(columns=['time'])

# Имена таблиц для записи
table_names = ['demo_geo_tracks', 'demo_geo_points']
```

④

```
# Записываем сформированные DataFrame в таблицы
for frame, table_name in zip([df_tracks, df_points], table_names):
    print(f'Creating table: {table_name}')
    print(f'Size: {frame.shape}')

    file_name = tngri.upload_df(frame)
    tngri.sql(f"CREATE OR REPLACE TABLE {table_name} AS SELECT * FROM
read_parquet('{file_name}')")

print(f'Finished. Time from start: {str(datetime.datetime.now() - start)}')
```

```
Files done: 100/2190         Points: 561928         Time from start: 0:01:57.949657
Files done: 200/2190         Points: 1055973         Time from start: 0:03:48.225765
Files done: 300/2190         Points: 1525239         Time from start: 0:05:38.334613
Files done: 400/2190         Points: 2125124         Time from start: 0:07:28.989251
Files done: 500/2190         Points: 2677727         Time from start: 0:09:25.163544
Files done: 600/2190         Points: 3190749         Time from start: 0:11:12.439539
Files done: 700/2190         Points: 3682379         Time from start: 0:13:02.326586
Files done: 800/2190         Points: 4241999         Time from start: 0:14:49.970186
Files done: 900/2190         Points: 4767488         Time from start: 0:16:37.493898
Files done: 1000/2190         Points: 5252788         Time from start: 0:18:24.054901
Files done: 1100/2190         Points: 5782689         Time from start: 0:20:14.398042
Files done: 1200/2190         Points: 6333129         Time from start: 0:22:03.400943
Files done: 1300/2190         Points: 6862458         Time from start: 0:23:51.433175
Files done: 1400/2190         Points: 7413209         Time from start: 0:25:41.723606
Files done: 1500/2190         Points: 7888312         Time from start: 0:27:28.314530
Files done: 1600/2190         Points: 8469405         Time from start: 0:29:22.235098
Files done: 1700/2190         Points: 8945913         Time from start: 0:31:10.916823
Files done: 1800/2190         Points: 9444734         Time from start: 0:33:01.630686
Files done: 1900/2190         Points: 9931590         Time from start: 0:34:49.403797
Files done: 2000/2190         Points: 10481654         Time from start: 0:36:37.313329
Files done: 2100/2190         Points: 11042971         Time from start: 0:38:29.835817
Creating table: demo_geo_tracks
Size: (2190, 6)
Creating table: demo_geo_points
Size: (11473061, 10)
Finished. Time from start: 0:40:59.713264
```

Обратим внимание на то, что мы записываем данные в две таблицы, которые связаны через ID трека (имя файла активности). Причем для удобства дальнейшей работы с данными, в таблицу треков мы сразу же записываем координаты первой точки трека (колонки `track_first_coord_lat` и `track_first_coord_lon`) и страну, вычисленную по этим координатам.

Выполнение этой ячейки занимает около 40 минут. Но примерно половина этого времени уходит на то, чтобы обратиться к сервису `geopy` и определить страну по координатам точки начала трека. Если есть необходимость ускорить начальную загрузку, то это обращение полезно отключить.

Проверка загруженных данных

Проверяем таблицу треков:

```
SELECT count(*) AS "Количество треков"
    FROM demo_geo_tracks
```

```
+------------------+
| Количество треков |
+------------------+
| 2190             |
+------------------+
```

```
SELECT *
    FROM demo_geo_tracks
    LIMIT 10
```

```
Done in 8.2 sec.

+----------------+-----------+----------------+----------------------
+----------------------+--------------+--------------------+---------------------+
| track_id        | track_name | track_length_km | track_first_coord_lat |
track_first_coord_lon | country       | speed              | time_stamp          |
+----------------+-----------+----------------+----------------------
+----------------------+--------------+--------------------+---------------------+
| 8205566762.fit  | null       | 51.04           | 52.47870256192982     | 13.336647050455213
| Deutschland   | 36.619200000000006 | 2020-06-26 13:44:05 |
+----------------+-----------+----------------+----------------------
+----------------------+--------------+--------------------+---------------------+
| 5758691664.fit  | null       | 3.92            | 55.7587356492877      | 37.745437659323215
| Россия        | 36.781200000000005 | 2020-06-26 13:44:06 |
+----------------+-----------+----------------+----------------------
+----------------------+--------------+--------------------+---------------------+
| 7562241822.fit  | null       | 21.69           | 36.594818104058504    | 30.56384850293398
| Türkiye       | 37.0512            | 2020-06-26 13:44:07 |
+----------------+-----------+----------------+----------------------
+----------------------+--------------+--------------------+---------------------+
| 5752363946.fit  | null       | 62.85           | 55.764476750046015    | 37.71281814202666
| Россия        | 37.3068            | 2020-06-26 13:44:08 |
+----------------+-----------+----------------+----------------------
+----------------------+--------------+--------------------+---------------------+
| 5774225138.fit  | null       | 114.24          | 55.99621960893273     | 36.27057650126517
| Россия        | 37.4652            | 2020-06-26 13:44:09 |
+----------------+-----------+----------------+----------------------
+----------------------+--------------+--------------------+---------------------+
| 3602018438.fit  | null       | 34.76           | 55.76445009559393     | 37.712875390425324
| Россия        | 37.7856            | 2020-06-26 13:44:10 |
+----------------+-----------+----------------+----------------------
```

```
+----------------------+--------------+--------------------+---------------------+
| 2665481253.fit  | null       | 43.88           | 57.06134635023773     | 23.311248868703842
| Latvija       | 37.980000000000004 | 2020-06-26 13:44:11 |
+-----------------+------------+-----------------+-----------------------
+----------------------+--------------+--------------------+---------------------+
| 16185006845.fit | null       | 66.47           | 36.71585462987423     |
-4.2837147787213326   | España        | 38.037600000000005 | 2020-06-26 13:44:12 |
+-----------------+------------+-----------------+-----------------------
+----------------------+--------------+--------------------+---------------------+
| 5061483340.fit  | null       | 16.87           | 40.78166037797928     | -73.96295428276062
| United States | 38.037600000000005 | 2020-06-26 13:44:13 |
+-----------------+------------+-----------------+-----------------------
+----------------------+--------------+--------------------+---------------------+
| 1842519266.fit  | null       | 71.94           | 55.73788298293948     | 37.65193585306406
| Россия        | 38.419200000000004 | 2020-06-26 13:44:14 |
+-----------------+------------+-----------------+-----------------------
+----------------------+--------------+--------------------+---------------------+
```

Проверяем таблицу точек:

```
SELECT count(*) AS "Количество точек"
    FROM demo_geo_points
```

```
+------------------+
| Количество точек |
+------------------+
| 11473061         |
+------------------+
```

```
SELECT *
    FROM demo_geo_points
    LIMIT 10
```

```
Done in 8.3 sec.

+-----------------+--------------------+--------------------+-------------------+------------
+-------------+---------+-------+--------------------+---------------------------+
| track_id        | latitude           | longitude          | elevation         | heart_rate
| temperature | cadence | power | speed              | time_text                 |
+-----------------+--------------------+--------------------+-------------------+------------
+-------------+---------+-------+--------------------+---------------------------+
| 11078368333.fit | 36.822746274992824 | -4.126659873872995 | 64.39999999999998 | null
| 18          | null    | 290   | 17.9784            | 2023-12-08 15:17:16+00:00 |
+-----------------+--------------------+--------------------+-------------------+------------
+-------------+---------+-------+--------------------+---------------------------+
| 11078368333.fit | 36.82279204018414  | -4.126656521111727 | 64.79999999999995 | null
| 18          | null    | 227   | 18.576             | 2023-12-08 15:17:17+00:00 |
+-----------------+--------------------+--------------------+-------------------+------------
```

```
+-------------+---------+-------+--------------------+--------------------------+
| 11078368333.fit | 36.822841661050916 | -4.126651743426919 | 65                | null
| 18          | null    | 148   | 18.4356            | 2023-12-08 15:17:18+00:00 |
+-----------------+--------------------+--------------------+-------------------+-----------
+-------------+---------+-------+--------------------+--------------------------+
| 11078368333.fit | 36.82288742624223  | -4.126646546646953 | 65.20000000000005 | null
| 18          | null    | 204   | 18.511200000000002 | 2023-12-08 15:17:19+00:00 |
+-----------------+--------------------+--------------------+-------------------+-----------
+-------------+---------+-------+--------------------+--------------------------+
| 11078368333.fit | 36.82293319143355  | -4.126641768962145 | 65.39999999999998 | null
| 18          | null    | 222   | 18.8244            | 2023-12-08 15:17:20+00:00 |
+-----------------+--------------------+--------------------+-------------------+-----------
+-------------+---------+-------+--------------------+--------------------------+
| 11078368333.fit | 36.822978956624866 | -4.126635566353798 | 65.60000000000002 | null
| 18          | null    | 213   | 18.9216            | 2023-12-08 15:17:21+00:00 |
+-----------------+--------------------+--------------------+-------------------+-----------
+-------------+---------+-------+--------------------+--------------------------+
| 11078368333.fit | 36.82302857749164  | -4.126623664051294 | 65.79999999999995 | null
| 18          | null    | 219   | 19.040399999999998 | 2023-12-08 15:17:22+00:00 |
+-----------------+--------------------+--------------------+-------------------+-----------
+-------------+---------+-------+--------------------+--------------------------+
| 11078368333.fit | 36.823078114539385 | -4.126606900244951 | 66                | null
| 18          | null    | 223   | 19.1484            | 2023-12-08 15:17:23+00:00 |
+-----------------+--------------------+--------------------+-------------------+-----------
+-------------+---------+-------+--------------------+--------------------------+
| 11078368333.fit | 36.82312773540616  | -4.126583598554134 | 66.20000000000005 | null
| 18          | null    | 223   | 19.332             | 2023-12-08 15:17:24+00:00 |
+-----------------+--------------------+--------------------+-------------------+-----------
+-------------+---------+-------+--------------------+--------------------------+
| 11078368333.fit | 36.82317350059748  | -4.126557866111398 | 66.39999999999998 | null
| 18          | null    | 216   | 19.6452            | 2023-12-08 15:17:25+00:00 |
+-----------------+--------------------+--------------------+-------------------+-----------
+-------------+---------+-------+--------------------+--------------------------+
```

Мы видим, что полученные через SQL-запросы данные о количестве строк в таблицах треков и точек совпадают с данными из лога нашей первой ячейки — 2.190 треков и 11.473.061 точек.

Теперь загрузим файл `strava_activities.csv` через Мастер загрузки и проверим данные в сформированной из него таблице.

```
CREATE OR REPLACE TABLE demo_geo_strava_activities AS
    SELECT *
        FROM read_csv(
            'abogdanov/5fe14b4c4110_strava_activities.csv'
        );

SELECT *
    FROM demo_geo_strava_activities
    LIMIT 10
    ORDER BY distance DESC;
```

```
Done in 9 sec.

+-------------+--------------------------+-----------------------------+--------------
+-------------------------------------------------------------------------------------
+--------------+----------+---------------+-----------------+---------
+----------------------+----------------------+-------------------------------
+---------------+-------------+---------------+-------------+-----------+-----------
+---------------+----------------+----------------+--------------+----------------
+-----------+---------------+-----------------------+-------------------------+-------------
+-----------------+------------------+--------------------+-----------+--------------
+----------+-----------------+---------------------+--------------------+------------
+---------------+-------------+--------------+------------+--------------------+------
+-----------+-------------------------+-------------+---------------------------
+--------------------------+-----------+---------------------+-------------
+------------------------+-------------------------+-------------------
+---------------------+----------------------+----------+----------+-----------------
+------------+-----------+--------------+-------------------------+--------------
+-------------+------------+----------+------+--------------------------
+--------------------+-------------+--------------------+----------+---------------
+------------+------------+--------------+---------+-----------------------+---------------
+-------------------------+--------------------------------+----------------+-------------
+--------------+-------------+---------------+-----------+-----------------------------
+------------+--------------+----------+----------+-------------+----------+-------------
+-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
-----------------------------------------------------+
| activity id | activity date            | activity name               | activity type |
activity description                                                          | elapsed
time | distance | max heart rate | relative effort | commute | activity private note |
activity gear        | filename                      | athlete weight | bike weight | elapsed
time_1 | moving time | distance_1 | max speed | average speed | elevation gain | elevation
loss | elevation low | elevation high | max grade | average grade | average positive grade |
average negative grade | max cadence | average cadence | max heart rate_1 | average heart
rate | max watts | average watts | calories | max temperature | average temperature |
relative effort_1 | total work | number of runs | uphill time | downhill time | other time |
perceived exertion | type | start time | weighted average power | power count | prefer
perceived exertion | perceived relative effort | commute_1 | total weight lifted | from
upload | grade adjusted distance | weather observation time | weather condition | weather
temperature | apparent temperature | dewpoint | humidity | weather pressure | wind speed |
wind gust | wind bearing | precipitation intensity | sunrise time | sunset time | moon phase
| bike     | gear | precipitation probability | precipitation type | cloud cover | weather
visibility | uv index | weather ozone | jump count | total grit | average flow | flagged |
average elapsed speed | dirt distance | newly explored distance | newly explored dirt
distance | activity count | total steps | carbon saved | pool length | training load |
intensity | average grade adjusted pace | timer time | total cycles | recovery | with pet |
competition | long run | for a cause | media
|
+-------------+--------------------------+-----------------------------+--------------
+-------------------------------------------------------------------------------------
```

```
+-------------+----------+---------------+-----------------+---------
+----------------------+----------------------+------------------------------
+---------------+------------+---------------+------------+-----------+----------
+---------------+---------------+---------------+---------------+---------------
+-----------+--------------+----------------------+----------------------+-------------
+----------------+-----------------+--------------------+----------+--------------
+----------+----------------+--------------------+------------------+------------
+---------------+------------+--------------+-----------+--------------------+------
+------------+-----------------------+-------------+--------------------------
+------------------------+-----------+--------------------+-------------
+------------------------+--------------------------+------------------
+--------------------+----------------------+----------+----------+-----------------
+------------+-----------+-------------+-----------------------+-------------
+-------------+------------+----------+------+--------------------------
+--------------------+-------------+-------------------+----------+--------------
+------------+------------+-------------+---------+----------------------+--------------
+------------------------+-------------------------------+---------------+------------
+--------------+------------+--------------+-----------+---------------------------
+------------+-------------+----------+----------+-------------+----------+-------------
+-----------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------
-----------------------------------------------------+
| 8735873891  | Mar 18, 2023, 7:08:04 AM | La Primavera with RCC       | Ride          | null
| 31281        | 200.77   | 179            | 194             | false   | null
| Giant TCR Advanced 1 | activities/9376119823.fit.gz  | null           | 8           | 31281
| 23034       | 200777.4   | 16.188    | 8.717         | 295            | 372            |
26.8          | 71.6           | 47.4      | 0             | null                    | null
| 172         | 74              | null             | 131                | null      | 146
| 3248     | null            | 14                  | 194               | 3246159    | null
| null        | null          | null       | null               | null | null       | 176
| 26231       | 0                          | null                       | 0         | null
| 1           | null                     | 1679122816               | 3                  | 7.5
| 5.6                  | 4        | 0.78     | 1011.5           | 2.9        | 6.7       |
184          | 0                       | 1679116544   | 1679159808  | 0.89       | 4194513  |
null | 0                         | null               | 1           | 16093              | 0
| 340.5         | null       | null       | null         | 0       | 6.419                |
1138.9        | null                     | null                          | null           |
null        | null         | null        | null          | null      | null
| null       | null         | null     | null     | null        | null     | null        |
media/4cda8a70-bd40-4e47-9f16-f8cce644e23d.jpg|media/5e549f56-1cc5-414e-b1f6-
70923b28c261.jpg|media/aaefa66b-e6e3-425a-bf68-265c1d9f6355.mp4|media/bbf17907-6990-458b-
8c41-ccba2dd463e4.jpg|media/d90e8b35-f65d-4649-8fcc-37214ba3b428.jpg|media/054f30b2-d198-
4ecd-b520-2338c670d509.jpg
|
+-------------+--------------------------+--------------------------+--------------
+-----------------------------------------------------------------------------------
+--------------+----------+---------------+-----------------+---------
+----------------------+----------------------+------------------------------
+---------------+------------+---------------+------------+-----------+----------
+---------------+---------------+---------------+---------------+---------------
```

```
+----------+--------------+----------------------+----------------------+------------
+----------------+------------------+--------------------+-----------+--------------
+----------+----------------+--------------------+-------------------+------------
+----------------+-------------+---------------+------------+--------------------+------
+------------+-----------------------+-------------+--------------------------
+--------------------------+-----------+---------------------+-------------
+------------------------+--------------------------+-------------------
+---------------------+---------------------+----------+----------+-----------------
+------------+-----------+--------------+-------------------------+--------------
+-------------+------------+----------+------+--------------------------
+--------------------+-------------+--------------------+----------+--------------
+------------+------------+--------------+---------+----------------------+--------------
+------------------------+--------------------------------+----------------+------------
+--------------+-------------+---------------+-----------+----------------------------
+------------+-------------+----------+----------+-------------+----------+-------------
+-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------+
| 2556833948  | Jul 23, 2019, 6:00:52 AM | Giro delle Dolomiti Stage 3 | Ride          | null
| 37818        | 180.47   | 177            | 184             | false   | null
| Giant TCR Advanced 1 | activities/2713670395.fit.gz  | 83             | 8           | 37818
| 28854       | 180471     | 19.4      | null          | 3151           | 3175           |
225           | 2256           | 47.1      | 0             | null                   | null
| 251         | 71              | 177              | 121                | null      | 149
| 5603     | null            | 29                   | 184               | null       | null
| null        | null          | null       | null               | null | null       | null
| null        | null                        | null                      | null      | null
| null        | null                      | null                      | null              | null
| null                 | null     | null     | null             | null       | null      |
null         | null                     | null         | null        | null       | 4194513  |
null | null                        | null               | null        | null               |
null     | null          | null       | null       | null         | null    | null
| null          | null                     | null                          | null           |
null        | null         | null        | null          | null      | null
| null       | null         | null     | null     | null        | null     | null        |
media/c21cda2a-7c14-47f7-a840-24c4cb412025.jpg|media/ae791d8b-e9a1-497c-9309-c49bbc318702.jpg
|
+-------------+--------------------------+----------------------------+--------------
+-----------------------------------------------------------------------------------
+--------------+----------+----------------+-----------------+---------
+----------------------+---------------------+--------------------------------
+----------------+-------------+----------------+-----------------+------------+-----------
+---------------+----------------+----------------+---------------+----------------
+-----------+---------------+-----------------------+-----------------------+-------------
+-----------------+------------------+--------------------+-----------+--------------
+----------+-----------------+--------------------+-------------------+------------
+----------------+-------------+---------------+------------+--------------------+------
+------------+-----------------------+-------------+--------------------------
+--------------------------+-----------+---------------------+-------------
+------------------------+--------------------------+-------------------
```

```
+--------------------+---------------------+----------+----------+-----------------
+------------+-----------+--------------+-------------------------+-------------
+-------------+------------+----------+------+--------------------------
+--------------------+-------------+--------------------+----------+--------------
+------------+------------+-------------+---------+----------------------+--------------
+-------------------------+-------------------------------+----------------+------------
+-------------+-------------+---------------+-----------+----------------------------
+------------+-------------+----------+----------+------------+----------+------------
+-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------+
| 4242331242  | Oct 25, 2020, 5:34:56 AM | Alanya Camp Day 6           | Ride          |
Queen stage                                                                    | 33077
| 180.46   | 157            | 198             | false   | null                  | Giant TCR
Advanced 1 | activities/4538350403.fit.gz  | null           | 8           | 33077          |
25687       | 180461.1   | 22.8      | 7.025         | 3110           | 3119           | 22
| 1060.4         | 33.3      | 0             | null                  | null
| 108         | 73              | null             | 128                | null      | 190
| 4997     | null            | 27                  | 198               | null       | null
| null        | null          | null       | null               | null | null       | null
| null        | 0                         | null                       | 0         | null
| 1           | null                     | 1603602048               | 1                 | 25.2
| 25.2                | 6.6      | 0.3      | 1013.4           | 1.6        | 1.9       | 40
| 0                      | 1603598976   | 1603638272  | 0.32       | 4194513  | null | 0
| null               | 0           | 16093              | 0        | 264.3         | null
| null       | null         | null    | null                  | null          | null
| null                          | null           | null        | null         | null        |
null          | null      | null                         | null       | null         | null
| null     | null        | null     | null        | media/574b5612-10c1-49b3-b3e1-
bc97896153b0.jpg|media/1944d2ba-2e4a-4d15-b7ae-56ede81c8db2.jpg|media/363ad0a2-1fb7-4552-
ae66-a53ec25e6bea.jpg|media/fc1cbf56-0bc7-4a26-88d1-c2558db2f934.jpg|media/d3a84a2d-0d73-
4452-9be3-e53ff1d952f8.jpg|media/6013cbeb-193b-46ba-873d-5120c26a7535.jpg
|
+-------------+-------------------------+---------------------------+--------------
+-------------------------------------------------------------------------------
+--------------+----------+----------------+-----------------+---------
+----------------------+----------------------+--------------------------------
+----------------+-------------+----------------+-------------+------------+-----------
+---------------+----------------+----------------+--------------+----------------
+-----------+---------------+-----------------------+-------------------------+-------------
+-----------------+------------------+--------------------+-----------+--------------
+----------+-----------------+---------------------+-------------------+------------
+----------------+-------------+--------------+------------+--------------------+------
+------------+-------------------------+-------------+---------------------------
+-------------------------+-----------+---------------------+-------------
+------------------------+--------------------------+------------------
+---------------------+----------------------+----------+----------+-----------------
+------------+-----------+-------------+-------------------------+--------------
+------------+-----------+----------+------+--------------------------
+--------------------+-------------+--------------------+----------+--------------
```

```
+-----------+-----------+-------------+--------+----------------------+--------------
+------------------------+-------------------------------+----------------+------------
+-------------+------------+--------------+----------+---------------------------
+-----------+-------------+---------+---------+------------+---------+------------
+-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-------------------------------------------------------+
| 5109982540  | Apr 11, 2021, 7:09:45 AM | Desperados camp Day 8       | Ride          |
Queen stage                                                                  | 33941
| 179.42   | 156            | 232             | false   | null                  | Giant TCR
Advanced 1 | activities/5447314715.fit.gz  | null           | 8           | 33941          |
26284       | 179424.8   | 19.7      | 6.826         | 3192           | 3196           | -16
| 1030.8         | 27        | 0             | null                    | null
| 107         | 73              | null             | 128                | null      | 184
| 5125     | null            | 17                  | 232               | null       | null
| null        | null          | null       | null               | null | null       | null
| null        | 0                         | null                       | 0         | null
| 1           | null                    | 1618124416              | 1                  | 12.6
| 12.6                | -1.3     | 0.38     | 1015.5           | 2.1        | 2.3       |
151          | 0                       | 1618111488   | 1618158208  | 0.99       | 4194513  |
null | 0                         | null               | 0           | 16093              | 3
| 374.3         | null       | null       | null         | null    | null                  |
null          | null                    | null                           | null          |
null        | null         | null        | null          | null      | null
| null       | null         | null     | null     | null        | null     | null        |
media/7cab3cd2-33d2-4c90-a3a7-9fe1ab10717a.jpg|media/6ed80c53-4d61-4ed7-b86c-
08f85e81ef3d.jpg|media/90ab131a-463e-412c-89c8-cbf489112f6a.jpg|media/90caba00-d1d4-4b58-
80f5-672797383a42.jpg|media/4486c339-d16d-4c33-bac4-c790117bb62b.jpg|media/b633df5d-3cb5-
42cd-9ce8-166382ab7d08.jpg|media/24da2f65-2a38-49e6-aaad-4e1637465eee.jpg|media/72c68f98-
acbd-4ba8-9f3c-1841147ee3ec.jpg
|
+-------------+--------------------------+----------------------------+--------------
+-----------------------------------------------------------------------------
+--------------+----------+----------------+-----------------+---------
+----------------------+----------------------+-------------------------------
+----------------+-------------+----------------+-------------+------------+-----------
+---------------+---------------+----------------+---------------+----------------
+-----------+---------------+-----------------------+-----------------------+-------------
+-----------------+-----------------+--------------------+-----------+--------------
+----------+-----------------+---------------------+--------------------+------------
+----------------+-------------+---------------+------------+--------------------+------
+------------+-----------------------+-------------+----------------------------
+---------------------------+-----------+---------------------+-------------
+-------------------------+---------------------------+-------------------
+---------------------+---------------------+----------+----------+------------------
+------------+-----------+--------------+-----------------------+--------------
+-------------+------------+----------+------+----------------------------
+--------------------+-------------+-------------------+----------+---------------
+------------+------------+--------------+---------+----------------------+---------------
+-------------------------+-------------------------------+----------------+-------------
```

```
+-------------+------------+--------------+-----------+----------------------------
+-----------+-------------+----------+----------+-------------+----------+------------
+-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------+
| 9196092834  | Jun 3, 2023, 8:53:43 AM  | To Chartres and back        | Ride          | Эпик
соло                                                                    | 31048
| 178.89   | null           | null            | false   | null                  | Giant TCR
Advanced 1 | activities/9866271692.fit.gz  | null           | 8           | 31048          |
24765       | 178891.4   | 16.196    | 7.224         | 1273           | 1269           | 60.8
| 179.8          | 35.1      | 0             | null                    | null
| 158         | 77              | null             | null               | null      | 139
| 3394     | null            | 24                  | null              | 3393782    | null
| null        | null          | null       | null               | null | null       | 164
| 27934       | 0                         | null                       | 0         | null
| 1           | null                    | 1685779200               | 1                  | 16.4
| 15.8                | 9.8      | 0.65     | 1018.3           | 3.5        | 7.3       | 30
| 0                      | 1685764224   | 1685821696  | 0.5        | 4194513  | null | 0
| 1                  | 0           | 27927.8            | 3        | null          | null
| null       | null         | 0       | 5.762                 | 162.6         | null
| null                          | null           | null        | null         | null        |
null          | null      | null                         | null       | null         | null
| null     | null        | null     | null        | media/6767f454-38a4-4e8f-9744-
0224aa404404.jpg|media/9710109e-2167-4787-af1e-9bc0d48d8b37.jpg|media/f8bfef10-0af0-4f26-
884d-2c39ba52e84b.jpg|media/734bba38-006a-46d8-8a3a-821a6664b02c.jpg|media/efe18288-537a-
45af-9c8f-0753faee311b.jpg|media/44862277-2c3c-4076-a6c7-c0808d078bcb.jpg|media/7e33db5c-
3484-4baf-bcaf-fe45b519cbef.jpg|media/ba09fc14-0114-4089-9d30-
97a717a98f97.jpg|media/994a029a-6360-43a3-b2b1-f86bf4926cf0.jpg|media/197cd6db-63da-4dde-
bd3b-f66ff65553c5.jpg|media/a69ff087-470a-43e1-aaf9-6514757a67f9.jpg |
+-------------+--------------------------+----------------------------+--------------
+-----------------------------------------------------------------------------------
+--------------+----------+----------------+-----------------+---------
+-----------------------+----------------------+-------------------------------
+----------------+-------------+----------------+-------------+------------+-----------
+---------------+----------------+----------------+--------------+----------------
+-----------+---------------+------------------------+------------------------+-------------
+-----------------+------------------+--------------------+-----------+--------------
+----------+-----------------+--------------------+-------------------+------------
+----------------+-------------+--------------+------------+--------------------+------
+------------+-------------------------+-------------+---------------------------
+---------------------------+-----------+--------------------+-------------
+-------------------------+--------------------------+-------------------
+----------------------+----------------------+----------+----------+-----------------
+------------+-----------+--------------+-------------------------+--------------
+-------------+------------+----------+------+---------------------------
+--------------------+-------------+--------------------+----------+---------------
+------------+------------+--------------+---------+-----------------------+----------------
+-------------------------+-------------------------------+----------------+-------------
+--------------+-------------+--------------+-----------+----------------------------
+------------+--------------+----------+----------+-------------+----------+-------------
```

```
+-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
-----------------------------------------------------+
| 13705580758 | Feb 23, 2025, 8:14:47 AM | Morning Drop Ride           | Ride          | Was
dropped on the way back and had to do 80km ITT                                  | 21668
| 168.59   | 180            | 424             | false   | null                  | Rose xlite
04        | activities/14626116429.fit.gz | null           | 7.8         | 21668          |
19470       | 168594.1   | 19.88     | 8.659         | 1288           | 1299           | 31.4
| 184.4          | 12.9      | 0             | null                    | null
| 164          | 84              | 180              | 149                | null      | 193
| 3718     | null            | 18                  | 424               | 3711913    | null
| null        | null          | null       | null               | null | null       | 222
| 20512       | 0                         | null                      | 0         | null
| 1           | null                    | 1740297600               | 1                  | 12.6
| 13.3                 | 7        | 0.69     | 1027             | 2.4        | 3.8       |
339          | 0                        | 1740293760   | 1740333952  | 0.875      | 13283968 |
null | 0                          | 1                  | 0           | 34184              | 0
| null          | null       | null       | null         | 0       | 7.781                |
0             | null                     | null                          | null           |
null        | null         | null        | 293           | 72        | null
| null       | null         | null     | null     | null        | null     | null        |
media/ca9d74d0-33fc-4db2-bb39-2ffdaaf08d67.jpg|media/584a4f49-c9f9-4560-807d-72752f676e75.jpg
|
+-------------+--------------------------+----------------------------+--------------
+-----------------------------------------------------------------------------------
+--------------+----------+----------------+-----------------+---------
+-----------------------+----------------------+--------------------------------
+----------------+-------------+----------------+-------------+------------+-----------
+---------------+----------------+----------------+---------------+----------------
+-----------+---------------+------------------------+------------------------+-------------
+-----------------+------------------+--------------------+-----------+---------------
+----------+-----------------+---------------------+--------------------+------------
+----------------+-------------+---------------+------------+--------------------+------
+------------+-------------------------+-------------+---------------------------
+--------------------------+-----------+---------------------+-------------
+------------------------+--------------------------+-------------------
+---------------------+----------------------+----------+----------+-----------------
+------------+-----------+--------------+-------------------------+--------------
+-------------+------------+----------+------+--------------------------
+--------------------+-------------+--------------------+----------+---------------
+------------+------------+--------------+---------+-----------------------+---------------
+-------------------------+-------------------------------+----------------+-------------
+--------------+-------------+---------------+-----------+----------------------------
+------------+-------------+----------+----------+-------------+----------+-------------
+-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------+
```

```
| 5084894611  | Apr 7, 2021, 7:07:29 AM  | Desperados camp Day 4       | Ride          | Pre-
epic                                                                                  | 25473
| 164.2    | 165            | 248             | false   | null                 | Giant TCR
Advanced 1 | activities/5421087598.fit.gz  | null           | 8           | 25473          |
19934       | 164206.4   | 21.2      | 8.238         | 1736           | 1729           |
-36.8         | 572.4          | 21.3      | 0             | null                    | null
| 108         | 79              | null            | 135                | null      | 206
| 4253     | null            | 24                  | 248               | null       | null
| null        | null          | null       | null               | null | null       | null
| null        | 0                         | null                       | 0         | null
| 1           | null                    | 1617778816               | 1                  | 20.1
| 20.1                | 11.1     | 0.56     | 1011.8           | 1.7        | 2.9       | 86
| 0                      | 1617766144   | 1617812352  | 0.86       | 4194513  | null | 0
| null               | 0.11        | 16093              | 4        | 345.5         | null
| null       | null         | null    | null                 | null          | null
| null                          | null           | null        | null         | null        |
null          | null      | null                          | null       | null         | null
| null     | null        | null     | null        | media/68f7fee3-f6ed-4285-9bd5-
a64271b3876e.jpg
|
+-------------+--------------------------+----------------------------+--------------
+-------------------------------------------------------------------------------
+--------------+----------+---------------+-----------------+---------
+-----------------------+----------------------+-------------------------------
+----------------+-------------+----------------+-------------+------------+-----------
+---------------+----------------+----------------+--------------+----------------
+-----------+---------------+-----------------------+-------------------------+-------------
+-----------------+------------------+--------------------+-----------+--------------
+----------+-----------------+----------------------+--------------------+------------
+----------------+-------------+--------------+------------+--------------------+------
+------------+------------------------+-------------+---------------------------
+--------------------------+-----------+---------------------+-------------
+------------------------+--------------------------+-------------------
+---------------------+---------------------+----------+----------+-----------------
+------------+-----------+--------------+------------------------+--------------
+-------------+------------+----------+------+--------------------------
+--------------------+-------------+--------------------+----------+---------------
+------------+------------+--------------+---------+----------------------+--------------
+------------------------+-------------------------------+----------------+-------------
+--------------+-------------+--------------+-----------+----------------------------
+------------+--------------+----------+----------+-------------+----------+-------------
-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
-------------------------------------------------------+
| 6178861995  | Oct 28, 2021, 6:09:52 AM | Cyprus Camp Day 5           | Ride          | null
| 30438        | 161.65   | 159            | 114             | false   | null
| Giant TCR Advanced 1 | activities/6566187110.fit.gz  | null           | 8           | 30438
| 20881       | 161657.4   | 17.227    | 7.742         | 1484           | 1502           | -4
| 181.6          | 27.7      | 0             | null                    | null
| 142         | 76              | null            | 121               | null      | 145
```

```
| 2956     | null            | 20                  | 114               | 2953912    | null
| null        | null          | null       | null               | null | null       | 178
| 22443       | 0                         | null                      | 0         | null
| 1           | null                    | 1635400832               | 2                  | 22.3
| 22.3                | 14.2     | 0.6      | 1016.9           | 1.8        | 1.9       | 73
| 0                      | 1635393920   | 1635433088  | 0.75       | 4194513  | null | 0
| null               | 0.7         | 16093              | 1        | 285.2         | null
| null       | null         | 0       | 5.311                | 102.7         | null
| null                          | null           | null        | null         | null        |
null          | null      | null                        | null       | null         | null
| null     | null        | null     | null        | media/a74a5532-85bc-40fe-9a4c-
8645bca02811.jpg|media/c335908c-72d8-4a09-ac2f-1719bb665973.jpg
|
+-------------+-------------------------+----------------------------+--------------
+-----------------------------------------------------------------------------------
+--------------+----------+---------------+-----------------+---------
+----------------------+---------------------+------------------------------
+----------------+-------------+----------------+-------------+------------+-----------
+---------------+----------------+----------------+---------------+----------------
+-----------+---------------+----------------------+-----------------------+-------------
+-----------------+------------------+--------------------+-----------+--------------
+----------+-----------------+---------------------+-------------------+------------
+----------------+-------------+--------------+------------+--------------------+------
+------------+------------------------+-------------+---------------------------
+--------------------------+-----------+---------------------+-------------
+------------------------+--------------------------+-------------------
+---------------------+---------------------+----------+----------+-----------------
+------------+-----------+--------------+-------------------------+--------------
+-------------+------------+----------+------+---------------------------
+--------------------+-------------+--------------------+----------+---------------
+------------+------------+--------------+---------+----------------------+---------------
+-------------------------+-------------------------------+---------------+-------------
+--------------+-------------+--------------+-----------+----------------------------
+-------------+--------------+----------+----------+-------------+----------+-------------
+-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-----------------------------------------------------------------------------------
-------------------------------------------------------+
| 9325531707  | Jun 24, 2023, 5:04:12 AM | Coup d'etat Ride            | Ride          |
Wonderful ride with nice guys! Thank you [strava://athletes/71275936/10]Luis Chase | 31007
| 159.99   | 179            | 332             | false   | null                  | Giant TCR
Advanced 1 | activities/10003255376.fit.gz | null           | null        | null           |
null        | null       | null      | null          | null           | null           | null
| null           | null      | null          | null                    | null
| null        | null            | null             | null               | null      | null
| null     | null            | null                | null              | null       | null
| null        | null          | null       | null               | null | null       | null
| null        | null                        | null                      | null      | null
| null        | null                     | null                       | null              | null
| null                 | null     | null     | null             | null       | null      |
null         | null                    | null         | null        | null       | null     |
```

```
null | null                      | null               | null        | null               |
null     | null          | null       | null       | null         | null    | null
| null          | null                     | null                           | null           |
null        | null         | null        | null          | null      | null
| null       | null         | null     | null     | null        | null     | null        |
media/0df135af-edfe-4148-af00-1d09287558b1.jpg|media/50663e37-4f88-407d-9fe2-
b5c1187a2f69.jpg|media/c196fee3-6fa1-419d-aa9b-95050bead0b9.mp4|media/1e674155-6607-49fd-
b28c-5bea672a8cd9.jpg|media/addc9782-4c14-43d5-a978-0c728442fc7c.jpg
|
+------------+--------------------------+----------------------------+--------------
+-------------------------------------------------------------------------------
+-------------+----------+---------------+----------------+---------
+----------------------+----------------------+-------------------------------
+----------------+-------------+----------------+-------------+------------+-----------
+---------------+---------------+----------------+---------------+----------------
+-----------+---------------+------------------------+------------------------+-------------
+-----------------+------------------+--------------------+-----------+--------------
+----------+-----------------+--------------------+--------------------+------------
+----------------+-------------+--------------+------------+--------------------+------
+------------+--------------------------+-------------+---------------------------
+--------------------------+-----------+---------------------+-------------
+------------------------+--------------------------+-------------------
+---------------------+---------------------+----------+----------+-----------------
+------------+-----------+--------------+-------------------------+--------------
+-------------+------------+----------+------+---------------------------
+---------------------+-------------+--------------------+----------+--------------
+------------+------------+--------------+---------+-----------------------+--------------
+-------------------------+-------------------------------+----------------+-------------
+--------------+-------------+---------------+-----------+-----------------------------
+------------+--------------+----------+----------+-------------+----------+-------------
+-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
-----------------------------------------------------+
| 11993808841 | Jul 27, 2024, 7:16:28 AM | Morning Ride                | Ride          | null
| 22167        | 157.8    | 177            | 262             | false   | null
| Rose xlite 04        | activities/12786638321.fit.gz | null           | 7.8         | 22167
| 18925       | 157800.3   | 18.886    | 8.338         | 1451           | 1489           |
-6.8          | 624.2          | 23.4      | 0             | null                    | null
| 93          | 68              | 177              | 137                | null      | 0
| 4130     | null            | 28                  | 262               | 3204       | null
| null        | null          | null       | null               | null | null       | 5
| 19179       | 0                         | null                       | 0         | null
| 1           | null                     | 1722063616               | 2                  | 23
| 25.6                 | 20.7     | 0.87     | 1012.7           | 0.7        | 1.6       | 24
| 0                      | 1722057600   | 1722108416  | 0.625      | 13283968 | null | 0
| 1                  | 0.41        | 20127              | 1        | null          | null
| null       | null         | 0       | 7.119                 | 0            | null
| null                           | null           | null        | null         | null        |
0             | 2         | null                          | null       | null         | null
| null     | null        | null     | null        | media/db7a0b8e-b71f-4a38-902e-
```

```
3435ae00f790.jpg
|
+------------+-------------------------+---------------------------+--------------
+-------------------------------------------------------------------------------
+-------------+----------+---------------+-----------------+---------
+----------------------+----------------------+-----------------------------
+---------------+-------------+---------------+-------------+-----------+-----------
+---------------+---------------+---------------+--------------+---------------
+-----------+--------------+----------------------+-----------------------+-------------
+-----------------+-----------------+--------------------+-----------+--------------
+----------+-----------------+--------------------+-------------------+------------
+---------------+-------------+--------------+-----------+--------------------+------
+-----------+-----------------------+-------------+--------------------------
+--------------------------+-----------+--------------------+-------------
+------------------------+-------------------------+------------------
+--------------------+---------------------+----------+----------+-----------------
+-----------+-----------+--------------+------------------------+--------------
+-------------+------------+----------+------+--------------------------
+--------------------+-------------+--------------------+----------+--------------
+-----------+------------+--------------+---------+----------------------+---------------
+------------------------+-------------------------------+---------------+-------------
+--------------+-------------+---------------+-----------+----------------------------
+-----------+--------------+----------+----------+-------------+----------+-------------
+-------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------
-----------------------------------------------------+
```

В этой таблице мы видим все параметры активностей, которые выгружены из самого сервиса Strava (а не из файлов треков). В них встречаются самые разнообразные параметры, в том числе такие экзотические как `Moon Phase` (*фаза луны*) и `With Pet` (*с животным*) .

Обратим внимание, что все скорости представлены в м/с, время — в секундах, а расстояние — в километрах.

Очистка и подготовка данных

Создадим таблицу серебряного слоя `demo_geo_tracks_silver`. Для этого преобразуем время из колонки `activity date` в тип `TIMESTAMP` используя функцию `strptime`, которая позволяет задать конкретный шаблон текстового описания момента времени. И объединим таблицу треков, полученную из файлов активностей, с таблицей, полученной из файла `strava_activities.csv`, проверяя, что `track_id` содержится в `filename` с помощью функции `contains`:

```
CREATE OR REPLACE TABLE demo_geo_tracks_silver AS
    SELECT *,
        strptime("activity date", '%b %d, %Y, %H:%M:%S %p') as time_stamp
        FROM demo_geo_tracks
        JOIN demo_geo_strava_activities ON (contains(filename, track_id));
```

```
SELECT * FROM demo_geo_tracks_silver
    LIMIT 10;
```

```
Done in 9.4 sec.

+-----------------+------------+-----------------+----------------------
+-----------------------+--------------+-------------+--------------------------
+--------------------------------+--------------+-----------------------------------------
+--------------+----------+---------------+-----------------+---------
+-----------------------+----------------------+------------------------------
+----------------+-------------+----------------+-------------+------------+-----------
+---------------+---------------+----------------+--------------+----------------
+-----------+--------------+----------------------+-------------------------+-------------
+-----------------+------------------+--------------------+-----------+--------------
+----------+-----------------+---------------------+-------------------+------------
+----------------+-------------+---------------+------------+--------------------+------
+------------+-------------------------+-------------+---------------------------
+--------------------------+-----------+---------------------+-------------
+------------------------+-------------------------+-------------------
+---------------------+----------------------+----------+----------+-----------------
+------------+-----------+--------------+-------------------------+--------------
+-------------+------------+----------+------+---------------------------
+---------------------+-------------+--------------------+----------+---------------
+------------+------------+--------------+---------+-----------------------+---------------
+------------------------+-------------------------------+----------------+-------------
+--------------+-------------+---------------+-----------+-----------------------------
+------------+--------------+----------+----------+-------------+----------+-------------
+-----------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------
-----+--------------------+
| track_id        | track_name | track_length_km | track_first_coord_lat |
track_first_coord_lon | country       | activity id | activity date            | activity
name                   | activity type | activity description                    | elapsed
time | distance | max heart rate | relative effort | commute | activity private note |
activity gear        | filename                      | athlete weight | bike weight | elapsed
time_1 | moving time | distance_1 | max speed | average speed | elevation gain | elevation
loss | elevation low | elevation high | max grade | average grade | average positive grade |
average negative grade | max cadence | average cadence | max heart rate_1 | average heart
rate | max watts | average watts | calories | max temperature | average temperature |
relative effort_1 | total work | number of runs | uphill time | downhill time | other time |
perceived exertion | type | start time | weighted average power | power count | prefer
perceived exertion | perceived relative effort | commute_1 | total weight lifted | from
upload | grade adjusted distance | weather observation time | weather condition | weather
temperature | apparent temperature | dewpoint | humidity | weather pressure | wind speed |
wind gust | wind bearing | precipitation intensity | sunrise time | sunset time | moon phase
| bike     | gear | precipitation probability | precipitation type | cloud cover | weather
visibility | uv index | weather ozone | jump count | total grit | average flow | flagged |
average elapsed speed | dirt distance | newly explored distance | newly explored dirt
distance | activity count | total steps | carbon saved | pool length | training load |
```

```
intensity | average grade adjusted pace | timer time | total cycles | recovery | with pet |
competition | long run | for a cause | media
| time_stamp          |
+-----------------+------------+-----------------+----------------------
+----------------------+--------------+-------------+-------------------------
+------------------------------+--------------+--------------------------------------
+--------------+----------+----------------+-----------------+---------
+----------------------+----------------------+-----------------------------
+----------------+-------------+----------------+-------------+------------+-----------
+---------------+---------------+----------------+--------------+---------------
+-----------+--------------+----------------------+-----------------------+-------------
+-----------------+------------------+--------------------+-----------+--------------
+----------+-----------------+--------------------+-------------------+------------
+----------------+-------------+--------------+------------+--------------------+------
+------------+-----------------------+-------------+--------------------------
+--------------------------+-----------+---------------------+-------------
+------------------------+-------------------------+-------------------
+--------------------+---------------------+----------+----------+-----------------
+------------+-----------+--------------+-------------------------+--------------
+-------------+------------+----------+------+--------------------------
+--------------------+-------------+--------------------+----------+--------------
+------------+------------+--------------+---------+----------------------+--------------
+-------------------------+------------------------------+----------------+-------------
+--------------+-------------+---------------+-----------+-----------------------------
+------------+--------------+----------+----------+-------------+----------+-------------
+-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----+--------------------+
| 8205566762.fit  | null       | 51.04           | 52.47870256192982     | 13.336647050455213
| Deutschland   | 7688162488  | Aug 23, 2022, 4:39:47 PM | Berlin Recon                   |
Ride          | null                                    | 7740         | 51.04    | 174
| 73              | false   | null                  | Giant TCR Advanced 1 |
activities/8205566762.fit.gz  | null           | 8           | 7740           | 6518        |
51044.1    | 16.012    | 7.831         | 174            | 188            | 18.6          |
68.4           | 9.3       | 0             | null                     | null
| 116         | 73              | null             | 134                | null      | 168
| 1083     | null            | 22                  | 73                | 1082634    | null
| null        | null          | null       | null               | null | null       | 202
| 7738        | 0                         | null                       | 0         | null
| 1           | null                     | 1661270400               | 2                  | 24.5
| 24.6                | 15.6     | 0.58     | 1018.7           | 3.5        | 3.8       | 60
| 0                      | 1661227392   | 1661278592  | 0.89       | 4194513  | null | 0
| null               | 0.34        | 16093              | 1        | 301.9         | null
| null       | null         | 0       | 6.595                | 332.6         | null
| null                           | null           | null        | null         | null        |
null          | null      | null                          | null       | null         | null
| null     | null        | null     | null        | media/f6509beb-c34d-4eb4-97c1-
9c80b92308ec.jpg|media/f2fe2880-42a1-4794-bef0-b5d563437aca.jpg
| 2022-08-23 16:39:47 |
+-----------------+------------+-----------------+----------------------
+----------------------+--------------+-------------+-------------------------
+------------------------------+--------------+--------------------------------------
```

```
+-------------+----------+---------------+-----------------+---------
+----------------------+----------------------+------------------------------
+---------------+-------------+---------------+-------------+-----------+-----------
+--------------+---------------+----------------+--------------+---------------
+-----------+--------------+----------------------+----------------------+-------------
+----------------+-----------------+--------------------+-----------+--------------
+---------+-----------------+--------------------+------------------+------------
+---------------+------------+--------------+-----------+--------------------+------
+------------+-----------------------+-------------+--------------------------
+------------------------+-----------+---------------------+-------------
+------------------------+--------------------------+-------------------
+--------------------+----------------------+----------+----------+-----------------
+------------+-----------+-------------+-----------------------+-------------
+-------------+------------+----------+------+--------------------------
+--------------------+-------------+-------------------+----------+--------------
+------------+------------+-------------+---------+----------------------+--------------
+------------------------+-------------------------------+---------------+------------
+--------------+------------+--------------+-----------+--------------------------
+------------+-------------+----------+----------+-------------+----------+-------------
+-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
-----+--------------------+
| 5758691664.fit  | null       | 3.92            | 55.7587356492877      | 37.745437659323215
| Россия        | 5408223435  | Jun 3, 2021, 6:35:24 PM  | Ночной заезд                    |
Ride          | null                                    | 554          | 3.91     | 143
| 2               | false   | null                  | Giant TCR Advanced 1 |
activities/5758691664.fit.gz  | null           | 8           | 554            | 554         |
3918.7     | 11.3      | 7.073         | 15             | 20             | 110.2         |
130.8          | 4.7       | -0.2          | null                    | null
| 92          | 67              | null             | 122                | null      | 127
| 68       | null            | 18                 | 2                 | 68242      | null
| null        | null          | null       | null               | null | null       | 141
| 555         | null                        | null                       | 0         | null
| 1           | null                     | 1622743168               | 1                 | 17.6
| 17.6                | 6.2      | 0.47     | 1024.7           | 3.5        | 8.7       | 42
| 0                      | 1622681472   | 1622743552  | 0.8        | 4194513  | null | 0
| null               | 0.28        | 16093              | 0        | 339           | null
| null       | null         | null    | null                   | null          | null
| null                           | null           | null        | null         | null        |
null          | null      | null                           | null       | null         | null
| null     | null        | null     | null        | null
| 2021-06-03 18:35:24 |
+-----------------+------------+-----------------+----------------------
+----------------------+--------------+-------------+--------------------------
+------------------------------+--------------+----------------------------------------
+-------------+----------+---------------+-----------------+---------
+----------------------+--------------------+------------------------------
+---------------+-------------+---------------+-------------+-----------+-----------
+--------------+---------------+----------------+--------------+---------------
+-----------+--------------+----------------------+----------------------+-------------
+----------------+-----------------+--------------------+-----------+--------------
+---------+-----------------+--------------------+------------------+------------
```

```
+---------------+------------+--------------+-----------+--------------------+------
+-----------+------------------------+-------------+--------------------------
+--------------------------+----------+--------------------+-------------
+------------------------+-------------------------+------------------
+---------------------+---------------------+----------+----------+-----------------
+-----------+----------+--------------+-------------------------+--------------
+-------------+------------+----------+------+--------------------------
+--------------------+-------------+--------------------+----------+--------------
+-----------+-----------+--------------+---------+----------------------+---------------
+------------------------+-------------------------------+---------------+-------------
+--------------+-------------+--------------+-----------+----------------------------
+-----------+--------------+----------+----------+-------------+----------+-------------
+-------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------
-----+--------------------+
| 7562241822.fit  | null       | 21.69           | 36.594818104058504    | 30.56384850293398
| Türkiye       | 7103414491  | May 7, 2022, 3:04:49 PM  | Вечерний велозаезд             |
Ride          | Завтра будем страдать в длинную гору    | 3044         | 21.69    | 148
| 10              | false   | null                  | Giant TCR Advanced 1 |
activities/7562241822.fit.gz  | null           | 8           | 3044           | 2786        |
21692.4    | 16.224    | 7.786         | 105            | 119            | 2             | 45
| 8.1       | 0             | null                  | null                   | 129         |
66              | null             | 108                | null      | 126           | 345
| null            | 21                  | 10                | 343778     | null           |
null        | null          | null       | null               | null | null       | 175
| 3045        | 0                        | null                      | 0         | null
| 1           | null                    | 1651935616               | 2                 | 22.5
| 22.5                | 9.7      | 0.44     | 1013             | 3.5        | 4.6       |
196          | 0                       | 1651892352   | 1651942272  | 0.21       | 4194513  |
null | 0                         | null               | 0.45        | 16093              | 1
| 352.8         | null       | null       | null         | 0       | 7.126                |
645           | null                    | null                          | null           |
null        | null         | null        | null          | null      | null
| null       | null         | null     | null     | null        | null     | null        |
media/318cbe19-4ae0-4800-a873-349417c4df90.jpg
| 2022-05-07 15:04:49 |
+-----------------+------------+----------------+----------------------
+-----------------------+---------------+-------------+--------------------------
+--------------------------------+---------------+-----------------------------------------
+--------------+----------+----------------+-----------------+---------
+-----------------------+---------------------+--------------------------------
+----------------+-------------+----------------+-------------+------------+-----------
+---------------+----------------+----------------+---------------+----------------
+-----------+---------------+-----------------------+-----------------------+-------------
+-----------------+------------------+--------------------+-----------+---------------
+----------+-----------------+--------------------+-------------------+------------
+----------------+-------------+--------------+------------+--------------------+------
+------------+------------------------+-------------+--------------------------
+--------------------------+-----------+---------------------+-------------
+------------------------+-------------------------+-------------------
+---------------------+---------------------+----------+----------+-----------------
+------------+-----------+--------------+------------------------+--------------
```

```
+------------+------------+----------+------+--------------------------
+--------------------+-------------+--------------------+----------+--------------
+------------+------------+-------------+---------+----------------------+--------------
+------------------------+-----------------------------+---------------+-------------
+-------------+-------------+--------------+-----------+---------------------------
+------------+-------------+----------+----------+-------------+----------+-------------
+-----------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------
-----+--------------------+
| 5752363946.fit  | null       | 62.85           | 55.764476750046015    | 37.71281814202666
| Россия        | 5402167712  | Jun 2, 2021, 3:55:33 PM  | Вечерний велозаезд             |
Ride          | null                                    | 8485         | 62.85    | 169
| 125             | false   | null                  | Giant TCR Advanced 1 |
activities/5752363946.fit.gz  | null           | 8           | 8485           | 7359        |
62849.2    | 14.5      | 8.54          | 220            | 255            | 154.2         |
226.2          | 10.7      | 0             | null                     | null
| 168         | 81              | null             | 143                | null      | 208
| 1527     | null            | 16                  | 125               | 1530481    | null
| null        | null          | null       | null               | null | null       | 224
| 7711        | null                        | null                       | 0         | null
| 1           | null                     | 1622646016               | 2                  | 19.7
| 19.7                | 5.6      | 0.4      | 1023.4           | 4.3        | 6.3       | 38
| 0                      | 1622595200   | 1622657152  | 0.76       | 4194513  | null | 0
| null               | 0.68        | 16093              | 1        | 360           | null
| null       | null         | null    | null                  | null          | null
| null                           | null           | null        | null         | null        |
null          | null      | null                          | null       | null         | null
| null     | null        | null     | null        | null
| 2021-06-02 15:55:33 |
+-----------------+------------+-----------------+----------------------
+----------------------+--------------+-------------+--------------------------
+------------------------------+--------------+---------------------------------------
+--------------+----------+---------------+----------------+---------
+-----------------------+---------------------+-----------------------------
+----------------+------------+----------------+-------------+------------+-----------
+---------------+---------------+----------------+--------------+----------------
+-----------+--------------+-----------------------+-----------------------+-------------
+-----------------+------------------+--------------------+-----------+---------------
+----------+-----------------+--------------------+-------------------+------------
+----------------+-------------+---------------+------------+--------------------+------
+------------+-----------------------+-------------+---------------------------
+--------------------------+-----------+--------------------+-------------
+------------------------+--------------------------+-------------------
+---------------------+----------------------+----------+----------+-----------------
+------------+-----------+--------------+-------------------------+--------------
+-------------+------------+----------+------+--------------------------
+--------------------+-------------+--------------------+----------+---------------
+------------+------------+-------------+---------+----------------------+---------------
+-------------------------+-------------------------------+----------------+-------------
+--------------+-------------+---------------+-----------+----------------------------
+------------+-------------+----------+----------+-------------+----------+-------------
+-----------------------------------------------------------------------------------------
```

```
-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
-----+--------------------+
| 5774225138.fit  | null       | 114.24          | 55.99621960893273     | 36.27057650126517
| Россия        | 5423038640  | Jun 6, 2021, 8:24:11 AM  | Granfondo Волоколамск Группа Б |
Ride          | 6 место в личном зачете и 1 в командном | 10566        | 114.24   | 189
| 284             | false   | null                  | Giant TCR Advanced 1 |
activities/5774225138.fit.gz  | null           | 8           | 10566          | 10496       |
114243.9   | 22.1      | 10.885        | 917            | 894            | 152.8         |
262.8          | 9.6       | 0             | null                    | null
| 173         | 80              | null            | 154                | null      | 199
| 2027     | null            | 26                  | 284               | 2026580    | null
| null        | null          | null       | null               | null | null       | 242
| 10567       | 0                         | null                       | 0         | null
| 1           | null                     | 1622966400               | 2                  | 22.1
| 22.1                 | 10.2     | 0.47     | 1014.9           | 3.5        | 4         | 71
| 0.1                    | 1622940800   | 1623003392  | 0.89       | 4194513  | null | 0.01
| 5                  | 0.45        | 16093              | 5        | 341.8         | null
| null       | null         | null    | null                  | null          | null
| null                          | null           | null        | null         | null        |
null          | null      | null                          | null       | null         | null
| null     | null        | null     | null        | media/946f24a2-cd5f-4b18-ae4b-
822158a40244.jpg|media/7d11f14c-7845-489a-bb34-f71b882c47e6.jpg|media/08fe3d9d-2f68-4319-
b5ce-00c6a6282a49.jpg|media/679c1351-1ac7-4a72-a525-c3bc1766c202.jpg|media/ba7f9fd0-a470-
45d9-8083-00ab445251cd.jpg|media/df115e87-def4-46ee-9f7e-899fd370903e.jpg | 2021-06-06
08:24:11 |
+-----------------+------------+-----------------+----------------------
+----------------------+--------------+-------------+-------------------------
+--------------------------------+---------------+----------------------------------------
+--------------+----------+----------------+-----------------+---------
+-----------------------+----------------------+-------------------------------
+----------------+-------------+----------------+-------------+------------+-----------
+---------------+----------------+----------------+---------------+----------------
+-----------+---------------+-----------------------+-------------------------+-------------
+-----------------+------------------+--------------------+-----------+--------------
+----------+-----------------+--------------------+-------------------+------------
+----------------+-------------+---------------+------------+--------------------+------
+------------+------------------------+-------------+---------------------------
+---------------------------+-----------+---------------------+-------------
+---------------------------+--------------------------+-------------------
+----------------------+----------------------+----------+----------+------------------
+------------+-----------+--------------+------------------------+--------------
+-------------+------------+----------+------+---------------------------
+--------------------+-------------+--------------------+----------+---------------
+------------+------------+--------------+---------+-----------------------+---------------
+--------------------------+--------------------------------+---------------+-------------
+--------------+-------------+---------------+-----------+----------------------------
+------------+--------------+----------+----------+-------------+----------+-------------
+-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
-----+--------------------+
| 3602018438.fit  | null       | 34.76           | 55.76445009559393     | 37.712875390425324
```

```
| Россия        | 3370915442  | Apr 29, 2020, 2:03:19 PM | Дневной велозаезд              |
Ride          | null                                    | 4075         | 34.76    | 174
| 127             | false   | null                  | Giant TCR Advanced 1 |
activities/3602018438.fit.gz  | 83             | 8           | 4075           | 3946        |
34762.6    | 15        | 8.801         | 140            | 146            | 132           |
162.6          | 17.6      | 0             | null                     | null
| 104         | 86              | 174             | 159                | null      | 191
| 1088     | null            | 11                  | 127                | null       | null
| null        | null          | null       | null               | null | null       | null
| null        | null                        | null                        | 0         | null
| 1           | null                     | 1588168832               | 2                  | 11.5
| 11.5                | 0        | 0.45     | 1000.3           | 7.4        | 12.1      |
301          | 0                        | 1588125056   | 1588179840  | 0.21       | 4194513  |
null | 0                          | null               | 0.53        | 16093              | 1
| 367.2         | null       | null       | null         | null    | null                 |
null          | null                     | null                           | null           |
null        | null         | null        | null          | null      | null
| null       | null         | null     | null     | null        | null     | null        |
null
| 2020-04-29 14:03:19 |
+-----------------+------------+-----------------+----------------------
+----------------------+--------------+-------------+--------------------------
+--------------------------------+---------------+-----------------------------------------
+--------------+----------+----------------+-----------------+---------
+-----------------------+----------------------+--------------------------------
+----------------+-------------+----------------+-------------+------------+-----------
+---------------+----------------+----------------+---------------+----------------
+-----------+---------------+-----------------------+-----------------------+-------------
+-----------------+------------------+--------------------+-----------+---------------
+----------+-----------------+--------------------+-------------------+------------
+----------------+-------------+---------------+------------+--------------------+------
+------------+------------------------+-------------+---------------------------
+--------------------------+-----------+---------------------+-------------
+------------------------+--------------------------+-------------------
+---------------------+----------------------+----------+----------+-----------------
+------------+-----------+--------------+-------------------------+--------------
+-------------+------------+----------+------+--------------------------
+--------------------+-------------+--------------------+----------+---------------
+------------+------------+--------------+---------+-----------------------+---------------
+-------------------------+---------------------------------+----------------+-------------
+--------------+-------------+---------------+-----------+-----------------------------
+------------+--------------+----------+----------+-------------+----------+-------------
+-----------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------
-----+--------------------+
| 2665481253.fit  | null       | 43.88           | 57.06134635023773     | 23.311248868703842
| Latvija       | 2509891301  | Jul 6, 2019, 6:12:44 PM  | Night ride                   |
Ride          | null                                    | 4958         | 43.88    | 163
| 85              | false   | null                  | Giant TCR Advanced 1 |
activities/2665481253.fit.gz  | 83             | 8           | 4958           | 4958        |
43880      | 12.2      | null          | 51             | 43             | -1            | 6
| 7.7       | 0             | null                    | null                   | 101         |
```

```
89              | 163              | 143                | null      | 170           | 1395
| null            | 12                  | 85                 | null       | null           |
null        | null          | null       | null                | null | null       | null
| null        | null                        | null                        | null      | null
| null        | null                      | null                      | null              | null
| null                 | null     | null     | null             | null       | null      |
null         | null                      | null         | null        | null       | 4194513  |
null | null                        | null                | null        | null                |
null     | null          | null       | null       | null         | null    | null
| null          | null                      | null                            | null           |
null        | null         | null        | null          | null      | null
| null       | null         | null     | null     | null        | null     | null        |
null
| 2019-07-06 18:12:44 |
+-----------------+------------+-----------------+----------------------
+----------------------+---------------+-------------+--------------------------
+------------------------------+---------------+----------------------------------------
+--------------+----------+----------------+-----------------+---------
+----------------------+----------------------+-------------------------------
+----------------+-------------+----------------+-------------+------------+-----------
+---------------+----------------+----------------+--------------+----------------
+-----------+--------------+-----------------------+-------------------------+-------------
+-----------------+------------------+--------------------+-----------+--------------
+----------+-----------------+---------------------+--------------------+------------
+----------------+-------------+--------------+------------+--------------------+------
+------------+------------------------+-------------+---------------------------
+--------------------------+-----------+---------------------+-------------
+------------------------+--------------------------+------------------
+---------------------+----------------------+----------+----------+-----------------
+------------+-----------+--------------+-------------------------+--------------
+-------------+------------+----------+------+--------------------------
+--------------------+-------------+--------------------+----------+---------------
+------------+------------+--------------+---------+----------------------+--------------
+------------------------+-------------------------------+----------------+-------------
+--------------+-------------+---------------+-----------+----------------------------
+------------+--------------+----------+----------+-------------+----------+-------------
+-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----+--------------------+
| 5061483340.fit  | null       | 16.87           | 40.78166037797928     | -73.96295428276062
| United States | 4741945778  | Feb 6, 2021, 7:18:45 AM  | NYC                            |
Virtual Ride  | null                                    | 1976         | 16.87    | 157
| 16              | false   | null                  | Giant TCR Advanced 1 |
activities/5061483340.fit.gz  | null           | 8           | 1976           | 1953        |
16870.5    | 15.2      | 8.638         | 176            | 0              | 17            |
46.2           | 12.4      | -0.1          | null                    | null
| 103         | 89              | null             | 133                | null      | 169
| 313      | null            | null               | 16                 | null       | null
| null        | null          | null       | null               | null | null       | 175
| 1977        | null                        | null                       | 0         | null
| 1           | null                      | null                      | null              | null
| null                 | null     | null     | null             | null       | null      |
```

```
null         | null                    | null         | null        | null       | 4194513  |
null | null                        | null                | null        | null                |
null     | null          | null       | null       | null         | null    | null
| null          | null                      | null                            | null           |
null        | null         | null        | null          | null      | null
| null       | null         | null     | null     | null        | null     | null        |
media/9218c649-fc79-4378-a133-5d691bd50c38.jpg
| 2021-02-06 07:18:45 |
+-----------------+------------+-----------------+----------------------
+----------------------+--------------+-------------+-------------------------
+--------------------------------+--------------+---------------------------------------
+-------------+----------+---------------+-----------------+---------
+----------------------+----------------------+-------------------------------
+----------------+-------------+----------------+-------------+------------+-----------
+---------------+---------------+----------------+--------------+---------------
+-----------+--------------+----------------------+-------------------------+-------------
+-----------------+------------------+--------------------+-----------+--------------
+----------+-----------------+--------------------+--------------------+------------
+----------------+-------------+--------------+------------+--------------------+------
+------------+-------------------------+-------------+---------------------------
+--------------------------+-----------+---------------------+-------------
+------------------------+--------------------------+------------------
+---------------------+---------------------+----------+----------+-----------------
+------------+-----------+--------------+-------------------------+--------------
+-------------+------------+----------+------+---------------------------
+---------------------+-------------+--------------------+----------+--------------
+------------+------------+--------------+---------+----------------------+---------------
+-------------------------+--------------------------------+----------------+-------------
+--------------+-------------+---------------+-----------+-----------------------------
+------------+--------------+----------+----------+-------------+----------+-------------
+-----------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------
-----+---------------------+
| 1842519266.fit  | null       | 71.94           | 55.73788298293948     | 37.65193585306406
| Россия        | 1717234129  | Jul 21, 2018, 9:51:37 AM | Noon ride                       |
Ride          | null                                      | 9707         | 71.94    | 170
| 186             | false   | null                  | Giant TCR Advanced 1 |
activities/1842519266.fit.gz  | 83             | 8           | 9707           | 8353        |
71942      | 13.6      | null          | 313            | 299            | 110           |
193            | 33.3      | 0             | null                     | null
| 101         | 86              | 170              | 148                | null      | 178
| 2298     | null            | 27                  | 186                | null       | null
| null        | null          | null       | null               | null | null       | null
| null        | null                        | null                       | 0         | null
| null        | null                      | null                       | null              | null
| null                 | null     | null     | null             | null       | null      |
null         | null                      | null         | null        | null       | 4194513  |
null | null                        | null                | null        | null                |
null     | null          | null       | null       | null         | null    | null
| null          | null                      | null                            | null           |
null        | null         | null        | null          | null      | null
| null       | null         | null     | null     | null        | null     | null        |
```

```
null
| 2018-07-21 09:51:37 |
+-----------------+------------+-----------------+----------------------
+-----------------------+---------------+-------------+--------------------------
+--------------------------------+--------------+-----------------------------------------
+--------------+----------+----------------+-----------------+---------
+-----------------------+----------------------+------------------------------
+----------------+-------------+----------------+-------------+------------+-----------
+---------------+----------------+----------------+---------------+----------------
+-----------+---------------+----------------------+----------------------+-------------
+-----------------+------------------+--------------------+-----------+--------------
+----------+-----------------+--------------------+------------------+------------
+----------------+-------------+--------------+------------+--------------------+------
+------------+-------------------------+-------------+--------------------------
+--------------------------+-----------+--------------------+-------------
+-------------------------+---------------------------+------------------
+---------------------+----------------------+----------+----------+-----------------
+------------+-----------+-------------+------------------------+-------------
+-------------+------------+----------+------+--------------------------
+---------------------+-------------+--------------------+----------+---------------
+------------+------------+-------------+---------+----------------------+--------------
+-------------------------+--------------------------------+----------------+-------------
+--------------+-------------+--------------+-----------+----------------------------
+------------+--------------+----------+----------+-------------+----------+-------------
+-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
-----+--------------------+
| 11417255201.fit | null       | 30.91           | 36.71511836349964     |
-4.2892365250736475   | España        | 10674837301 | Feb 1, 2024, 2:24:07 PM  | Afternoon
Ride                 | Ride          | From new apartment                       | 4793
| 30.91    | 166            | 67              | false   | null                  | Rose xlite
04        | activities/11417255201.fit.gz | null           | 7.8         | 4793           |
4571        | 30913.6    | 15.33     | 6.763         | 658            | 676            | 8.8
| 509.8          | 20.3      | 0             | null                  | null
| 118         | 74              | null            | 137               | null      | 188
| 850      | null            | 15                  | 67                | 849803     | null
| null        | null          | null       | null              | null | null       | 227
| 4794        | 0                         | null                       | 0         | null
| 1           | null                     | 1706796032              | 1                 | 16.9
| 16.4                | 10.4     | 0.65     | 1033            | 2.7        | 5.9       |
128          | 0                       | 1706771968   | 1706809344  | 0.625      | 13283968 |
null | 0                         | 1                 | 0.29        | 26463              | 2
| null          | null       | null       | null         | 0       | 6.45                 |
109.7         | null                    | null                          | null           |
null        | null         | null        | null          | null      | null
| null       | null         | null     | null     | null        | null     | null        |
media/da2f0e86-c77e-42c2-b7c5-5d08d0ac7acf.jpg
| 2024-02-01 14:24:07 |
+-----------------+------------+-----------------+----------------------
+-----------------------+---------------+-------------+--------------------------
+--------------------------------+--------------+-----------------------------------------
+--------------+----------+----------------+-----------------+---------
```

```
+----------------------+---------------------+-----------------------------
+---------------+-------------+---------------+-------------+-----------+-----------
+---------------+---------------+---------------+--------------+---------------
+-----------+--------------+----------------------+-----------------------+-------------
+----------------+-----------------+--------------------+----------+--------------
+----------+----------------+--------------------+------------------+------------
+---------------+-------------+--------------+-----------+--------------------+------
+-----------+-----------------------+-------------+--------------------------
+--------------------------+-----------+---------------------+-------------
+------------------------+--------------------------+------------------
+--------------------+---------------------+----------+----------+-----------------
+-----------+-----------+-------------+------------------------+-------------
+-------------+-----------+----------+------+--------------------------
+--------------------+-------------+--------------------+----------+--------------
+-----------+-----------+-------------+---------+----------------------+--------------
+------------------------+------------------------------+---------------+-------------
+-------------+-------------+--------------+-----------+---------------------------
+-----------+-------------+----------+----------+------------+----------+-------------
+-----------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------
-----+--------------------+
```

Теперь создадим таблицу точек серебряного слоя — оставим в ней только нужные колонки, преобразуем данные в колонке `time_text` в тип `TIMESTAMP` (используя функции `strptime` и `regexp_extract`) и отфильтруем все точки, в которых значение `latitude` или `longitude` пустое:

```
CREATE OR REPLACE TABLE demo_geo_points_silver AS
    SELECT
        track_id,
        latitude,
        longitude,
        elevation,
        temperature,
        power,
        speed,
        strptime(regexp_extract(time_text, '^.{19}'), '%c') as time_stamp,
    FROM demo_geo_points
    WHERE latitude IS NOT NULL AND longitude IS NOT NULL;

SELECT *
    FROM demo_geo_points_silver
    LIMIT 10;
```

```
Done in 11.6 sec.

+---------------+------------------+------------------+--------------------
+-------------+-------+------------------+---------------------+
| track_id       | latitude           | longitude          | elevation          | temperature
| power | speed              | time_stamp          |
```

```
+---------------+-------------------+-------------------+-------------------
+------------+-------+-------------------+--------------------+
| 3922792040.fit | 55.84256364963949  | 37.73899080231786  | 167.60000000000002 | 24
| null  | 36.619200000000006 | 2020-06-26 13:44:05 |
+---------------+-------------------+-------------------+-------------------
+------------+-------+-------------------+--------------------+
| 3922792040.fit | 55.842567421495914 | 37.73883054032922  | 167.60000000000002 | 24
| null  | 36.781200000000005 | 2020-06-26 13:44:06 |
+---------------+-------------------+-------------------+-------------------
+------------+-------+-------------------+--------------------+
| 3922792040.fit | 55.842567421495914 | 37.73866650648415  | 167.60000000000002 | 24
| null  | 37.0512            | 2020-06-26 13:44:07 |
+---------------+-------------------+-------------------+-------------------
+------------+-------+-------------------+--------------------+
| 3922792040.fit | 55.842567421495914 | 37.73849870078266  | 167.60000000000002 | 24
| null  | 37.3068            | 2020-06-26 13:44:08 |
+---------------+-------------------+-------------------+-------------------
+------------+-------+-------------------+--------------------+
| 3922792040.fit | 55.84256364963949  | 37.73833081126213  | 167.60000000000002 | 24
| null  | 37.4652            | 2020-06-26 13:44:09 |
+---------------+-------------------+-------------------+-------------------
+------------+-------+-------------------+--------------------+
| 3922792040.fit | 55.8425560221076   | 37.738166777417064 | 167.60000000000002 | 24
| null  | 37.7856            | 2020-06-26 13:44:10 |
+---------------+-------------------+-------------------+-------------------
+------------+-------+-------------------+--------------------+
| 3922792040.fit | 55.842548394575715 | 37.73799511604011  | 167.39999999999998 | 24
| null  | 37.980000000000004 | 2020-06-26 13:44:11 |
+---------------+-------------------+-------------------+-------------------
+------------+-------+-------------------+--------------------+
| 3922792040.fit | 55.842544538900256 | 37.7378349378705   | 167.20000000000005 | 24
| null  | 38.037600000000005 | 2020-06-26 13:44:12 |
+---------------+-------------------+-------------------+-------------------
+------------+-------+-------------------+--------------------+
| 3922792040.fit | 55.84254076704383  | 37.737670904025435 | 167                | 24
| null  | 38.037600000000005 | 2020-06-26 13:44:13 |
+---------------+-------------------+-------------------+-------------------
+------------+-------+-------------------+--------------------+
| 3922792040.fit | 55.84253691136837  | 37.73749924264848  | 166.79999999999995 | 24
| null  | 38.419200000000004 | 2020-06-26 13:44:14 |
+---------------+-------------------+-------------------+-------------------
+------------+-------+-------------------+--------------------+
```

Анализ

На основе сформированных выше данных можно получить самую разнообразную статистику. Ограничимся здесь только нескольким примерами.

Статистика по странам

Выведем суммарный пробег в километрах по всем трекам с началом в данной стране, исключая виртуальные заезды. Упорядочим результат по убыванию суммарного пробега. Используем функции `round` и `sum`.

```
SELECT
    country AS "Country",
    round(sum(distance)) AS "Total distance, km",
FROM demo_geo_tracks_silver
WHERE NOT "Activity Type" = 'Virtual Ride'
GROUP BY 1
ORDER BY 2 DESC
```

```
+--------------------------------+--------------------+
| Country                        | Total distance, km |
+--------------------------------+--------------------+
| Россия                         | 36947              |
+--------------------------------+--------------------+
| España                         | 30437              |
+--------------------------------+--------------------+
| Türkiye                        | 8255               |
+--------------------------------+--------------------+
| Deutschland                    | 6033               |
+--------------------------------+--------------------+
| Italia                         | 2681               |
+--------------------------------+--------------------+
| France                         | 1833               |
+--------------------------------+--------------------+
| K□προς - Kıbrıs                | 1452               |
+--------------------------------+--------------------+
| Latvija                        | 1191               |
+--------------------------------+--------------------+
| Schweiz/Suisse/Svizzera/Svizra | 708                |
+--------------------------------+--------------------+
| Sverige                        | 615                |
+--------------------------------+--------------------+
| □□□□□□□□                       | 595                |
+--------------------------------+--------------------+
| Magyarország                   | 421                |
+--------------------------------+--------------------+
| Nederland                      | 408                |
+--------------------------------+--------------------+
| Österreich                     | 352                |
+--------------------------------+--------------------+
| Slovensko                      | 223                |
+--------------------------------+--------------------+
| Unknown                        | 56                 |
+--------------------------------+--------------------+
| United Kingdom                 | 48                 |
```

```
+------------------------------+--------------------+
```

Мы получили таблицу рейтинга стран в зависимости от суммарного пробега в каждой стране.

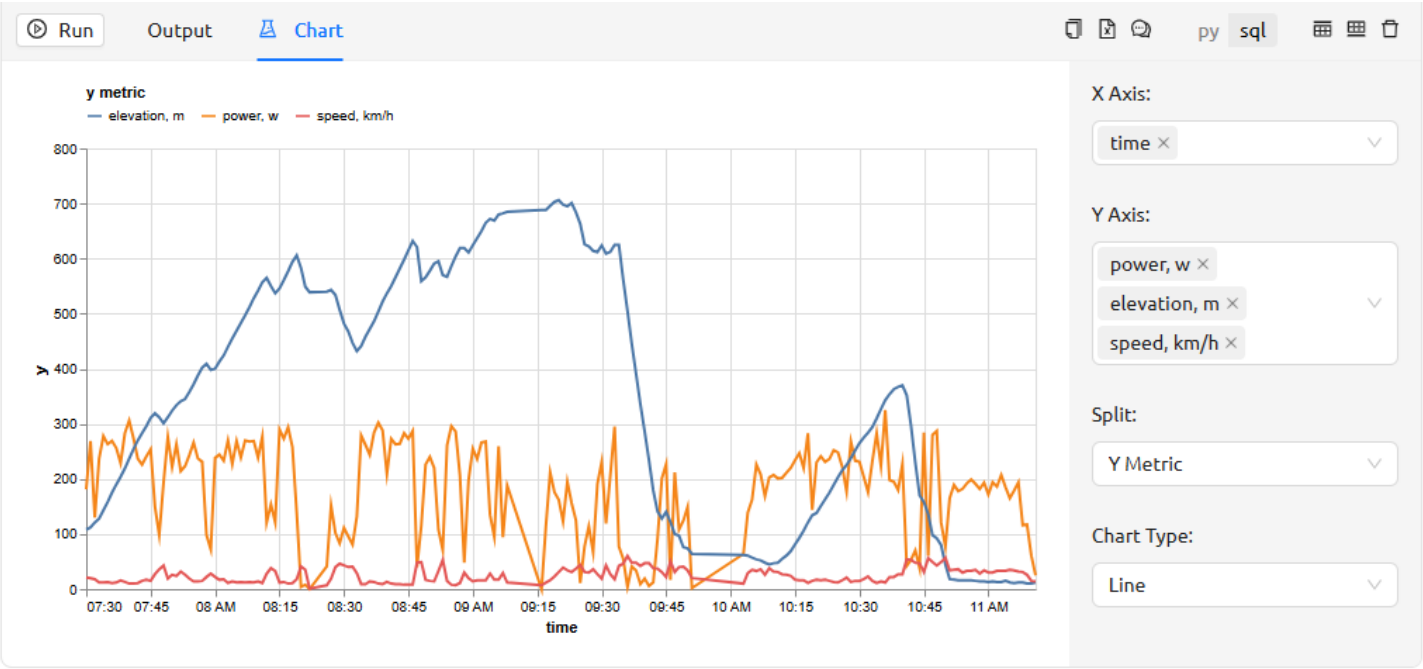
Анализ нескольких показателей внутри одной активности

Сформируем график, на котором будут одновременно показаны скорость, высота и мощность внутри одной заданной активности с поминутной сегментацией и дополнительным ограничением по времени начала. Используем функции `avg`, `round` и `date_trunc` с параметром `minute`.

```
SELECT
    round(avg(power)) AS "power, w",
    round(avg(elevation)) AS "elevation, m",
    round(avg(speed)) AS "speed, km/h",
    date_trunc('minute', time_stamp) as time,
FROM demo_geo_points_silver
WHERE track_id = '16627535673.fit' and time_stamp < (timestamp '2025-08-24 07:30:00')
GROUP BY TIME
ORDER BY TIME
```

```
Done in 8.5 sec.

+----------+--------------+-------------+---------------------+
| power, w | elevation, m | speed, km/h | time                |
+----------+--------------+-------------+---------------------+
| 181      | 108          | 21          | 2025-08-24 07:30:00 |
+----------+--------------+-------------+---------------------+
| 269      | 112          | 20          | 2025-08-24 07:31:00 |
+----------+--------------+-------------+---------------------+
| 130      | 121          | 18          | 2025-08-24 07:32:00 |
+----------+--------------+-------------+---------------------+
| 237      | 128          | 12          | 2025-08-24 07:33:00 |
+----------+--------------+-------------+---------------------+
| 278      | 143          | 12          | 2025-08-24 07:34:00 |
+----------+--------------+-------------+---------------------+
| ...      | ...          | ...         | ...                 |
+----------+--------------+-------------+---------------------+
201 rows
```

Для отображения графика воспользуемся вкладкой **Chart** в интерфейсе вывода ячейки:



Мы получили график, на котором одновременно отображаются несколько разных показателей для данной активности — скорость, высота и мощность. При наведении мышкой на линию графика показывается значение в данной точке. На таких графиках пользователи могут изучать показатели в их зависимости друг от друга. Например, в данном случае можно заметить, что на спусках скорость возрастает, а мощность падает, на подъемах же — наоборот.

Распределение мощности внутри одной активности

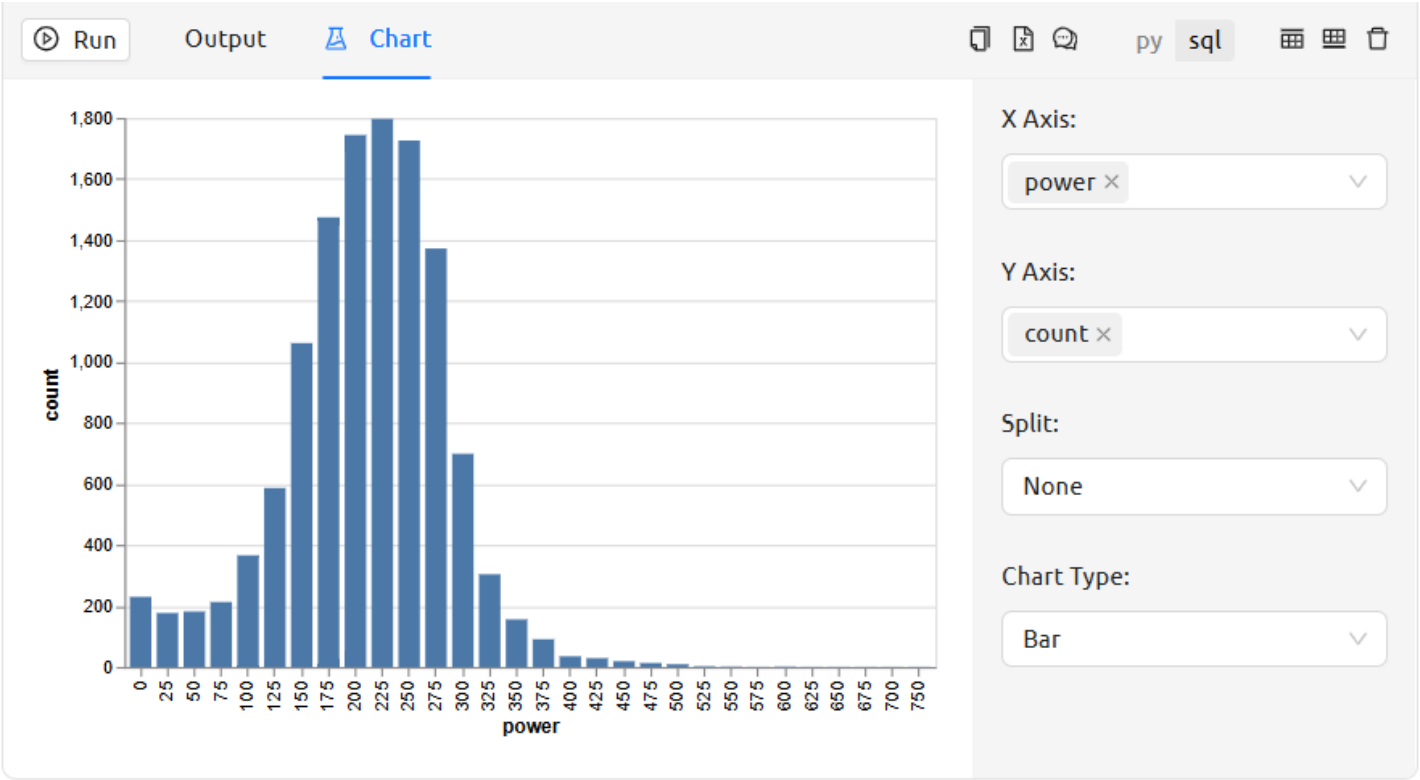
Выведем график распределения мощности внутри той же активности с шагом по 25 ватт. Для этого воспользуемся функцией `floor`.

```
SELECT
    floor(power/25)*25 AS power,
    count(*) AS count
FROM demo_geo_points_silver
WHERE track_id = '16627535673.fit' AND power > 0
GROUP BY 1
ORDER BY 1
```

```
Done in 8.2 sec.

+-------+-------+
| power | count |
+-------+-------+
| 0     | 230   |
+-------+-------+
| 25    | 177   |
+-------+-------+
| 50    | 182   |
+-------+-------+
| 75    | 213   |
+-------+-------+
```

```
| 100   | 366   |
+-------+-------+
| ...   | ...   |
+-------+-------+
30 rows
```

Для отображения графика воспользуемся вкладкой **Chart** в интерфейсе вывода ячейки:



Мы получили график распределения мощности внутри одной активности с шагом по 25 ватт.

Статистика по календарным периодам

Чтобы вывести подробную статистику по календарным периодам (месяцы, дни недели, часы внутри суток и прочее), создадим на основе `demo_geo_tracks_silver` новую таблицу, в которую добавим колонки, соответствующие разным периодам времени. Используем для этого функции `concat` и `regexp_extract`.

```
CREATE OR REPLACE TABLE demo_geo_tracks_silver_time AS
SELECT
    track_id,
    hour(time_stamp) AS hour,
    regexp_extract(string(time_stamp), '.{5} ') as day_of_year,
    month(time_stamp) AS month,
    monthname(time_stamp) AS month_name,
    dayname(time_stamp) AS day_name,
    concat(regexp_extract(string(day_of_year), '.{2} $') , ' ', month_name) as text_date,
FROM demo_geo_tracks_silver;

SELECT *
    FROM demo_geo_tracks_silver_time
```

```
    LIMIT 10;
```

```
+----------------+------+-------------+------------+-------+------+------------+-----------
+-------------+
| track_id       | hour | day_of_year | month_year | month | year | month_name | day_name  |
text_date   |
+----------------+------+-------------+------------+-------+------+------------+-----------
+-------------+
| 8205566762.fit | 16   | 08-23       | 2022-08    | 8     | 2022 | August     | Tuesday   |
23 August   |
+----------------+------+-------------+------------+-------+------+------------+-----------
+-------------+
| 5758691664.fit | 18   | 06-03       | 2021-06    | 6     | 2021 | June       | Thursday  |
03 June     |
+----------------+------+-------------+------------+-------+------+------------+-----------
+-------------+
| 7562241822.fit | 15   | 05-07       | 2022-05    | 5     | 2022 | May        | Saturday  |
07 May      |
+----------------+------+-------------+------------+-------+------+------------+-----------
+-------------+
| 5752363946.fit | 15   | 06-02       | 2021-06    | 6     | 2021 | June       | Wednesday |
02 June     |
+----------------+------+-------------+------------+-------+------+------------+-----------
+-------------+
| 5774225138.fit | 8    | 06-06       | 2021-06    | 6     | 2021 | June       | Sunday    |
06 June     |
+----------------+------+-------------+------------+-------+------+------------+-----------
+-------------+
| 3602018438.fit | 14   | 04-29       | 2020-04    | 4     | 2020 | April      | Wednesday |
29 April    |
+----------------+------+-------------+------------+-------+------+------------+-----------
+-------------+
| 2665481253.fit | 18   | 07-06       | 2019-07    | 7     | 2019 | July       | Saturday  |
06 July     |
+----------------+------+-------------+------------+-------+------+------------+-----------
+-------------+
| 5061483340.fit | 7    | 02-06       | 2021-02    | 2     | 2021 | February   | Saturday  |
06 February |
+----------------+------+-------------+------------+-------+------+------------+-----------
+-------------+
| 1842519266.fit | 9    | 07-21       | 2018-07    | 7     | 2018 | July       | Saturday  |
21 July     |
+----------------+------+-------------+------------+-------+------+------------+-----------
+-------------+
| 11417255201.fit | 14   | 02-01       | 2024-02    | 2     | 2024 | February   | Thursday  |
01 February |
+----------------+------+-------------+------------+-------+------+------------+-----------
+-------------+
```

Построим график, из которого будет видно, какова суммарная дистанция в километрах за все время в активностях, начинающихся в каждый час суток. Для объединения таблиц

`demo_geo_tracks_silver_time` и `demo_geo_tracks_silver` по колонке `track_id` воспользуемся выражением `JOIN` с параметром `USING`.

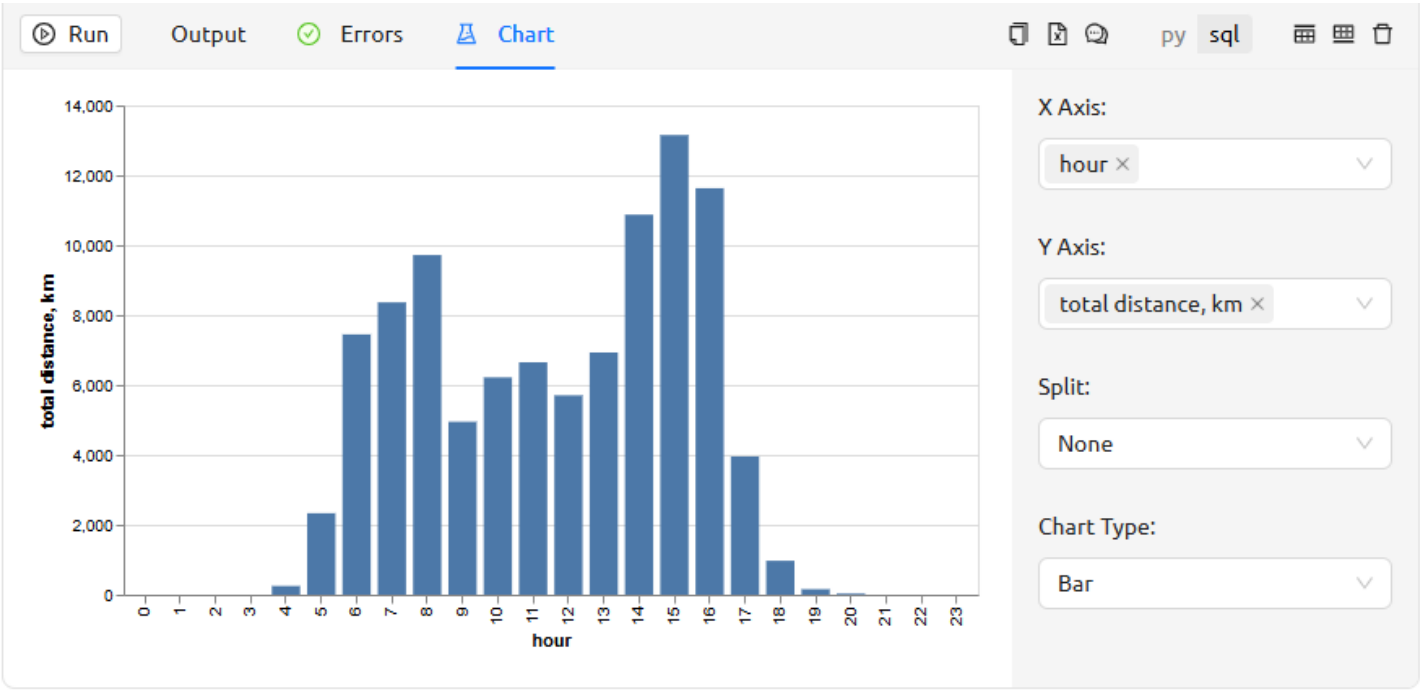
Для заполнения строк с пустыми часами используем функции `unnest` и `generate_series`.

```
CREATE OR REPLACE TABLE demo_geo_tracks_hour AS
SELECT
    distance,
    hour,
    time_stamp
FROM demo_geo_tracks_silver_time
    JOIN demo_geo_tracks_silver
        USING (track_id);

-- Добавляем пустые часы
INSERT INTO demo_geo_tracks_hour (hour, distance)
    SELECT unnest(generate_series(23)), 0;

SELECT
    round(sum(distance)) AS "Total distance, km",
    hour AS Hour
FROM demo_geo_tracks_hour
GROUP BY 2
ORDER BY 2;
```

```
Done in 9.5 sec.

+--------------------+------+
| Total distance, km | Hour |
+--------------------+------+
| 0                  | 0    |
+--------------------+------+
| 0                  | 1    |
+--------------------+------+
| 0                  | 2    |
+--------------------+------+
| 0                  | 3    |
+--------------------+------+
| 250                | 4    |
+--------------------+------+
| 2332               | 5    |
+--------------------+------+
| 7448               | 6    |
+--------------------+------+
| 8365               | 7    |
+--------------------+------+
| 9719               | 8    |
+--------------------+------+
| 4949               | 9    |
+--------------------+------+
| ...                | ...  |
```

```
+--------------------+------+
24 rows
```

Для отображения графика воспользуемся вкладкой **Chart** в интерфейсе вывода ячейки:



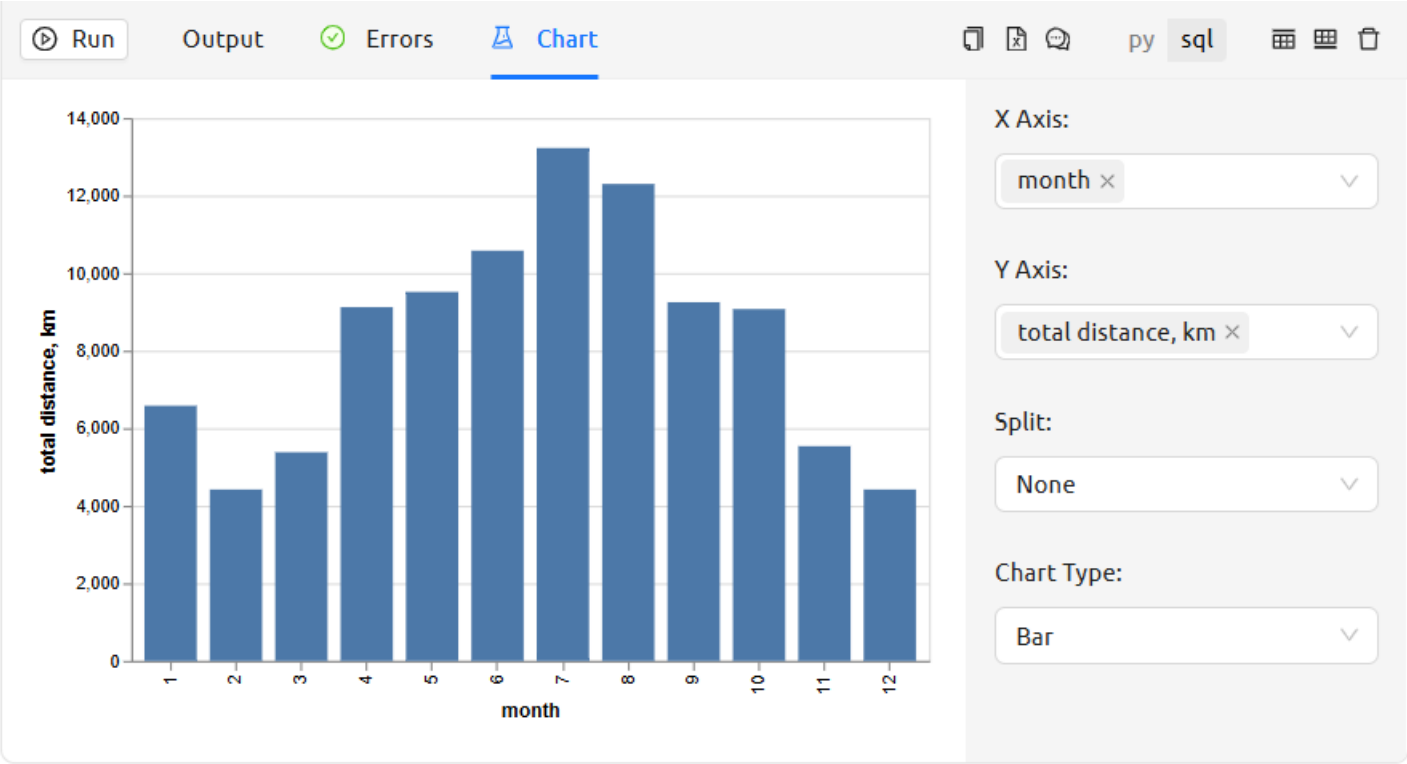
Теперь построим график, на котором покажем суммарную дистанцию в километрах для каждого месяца за все время.

```
SELECT
    round(sum(distance)) AS "Total distance, km",
    month,
FROM demo_geo_tracks_silver_time
    JOIN demo_geo_tracks_silver
        USING (track_id)
GROUP BY 2
ORDER BY 2
```

```
Done in 8.2 sec.

+--------------------+-------+
| total distance, km | month |
+--------------------+-------+
| 6580               | 1     |
+--------------------+-------+
| 4419               | 2     |
+--------------------+-------+
| 5380               | 3     |
+--------------------+-------+
| 9122               | 4     |
+--------------------+-------+
| 9514               | 5     |
+--------------------+-------+
```

```
| 10578              | 6     |
+--------------------+-------+
| 13225              | 7     |
+--------------------+-------+
| 12298              | 8     |
+--------------------+-------+
| 9250               | 9     |
+--------------------+-------+
| 9072               | 10    |
+--------------------+-------+
| 5537               | 11    |
+--------------------+-------+
| 4419               | 12    |
+--------------------+-------+
```

Для отображения графика воспользуемся вкладкой **Chart** в интерфейсе вывода ячейки:



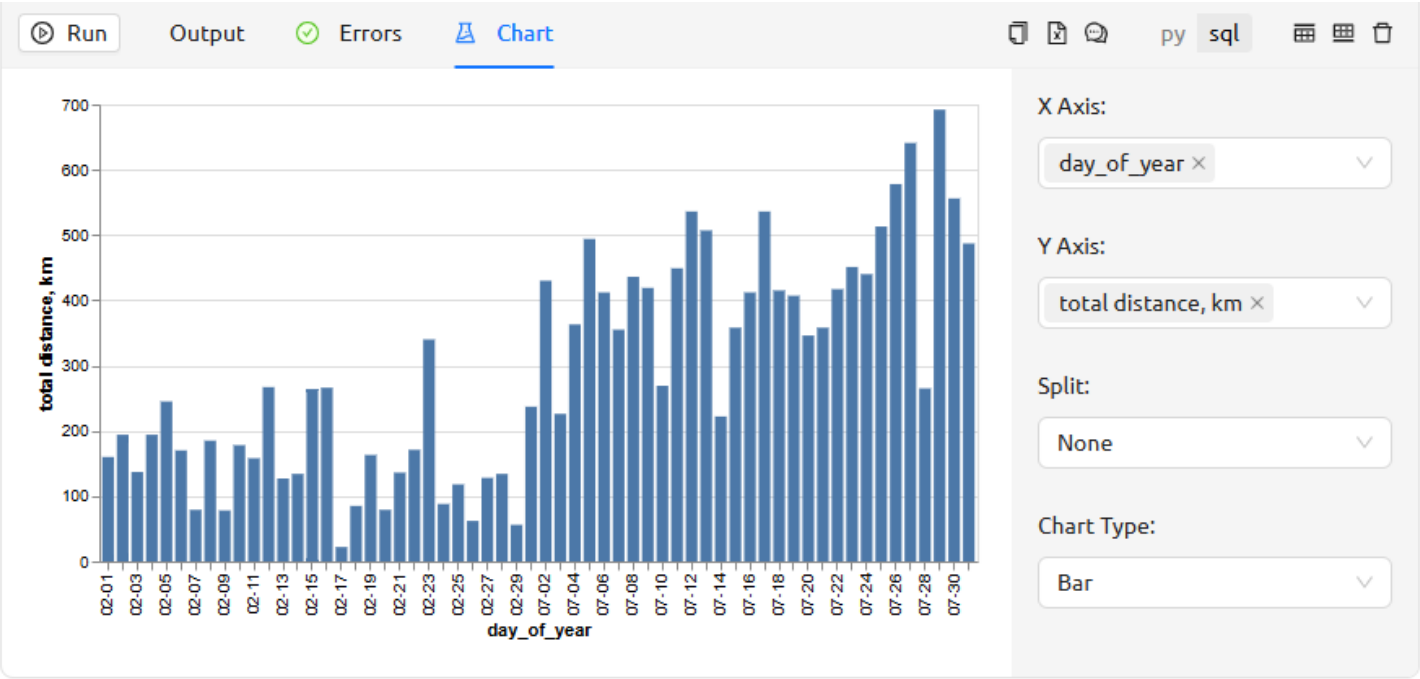
Ожидаемо в зимние месяцы суммарная дистанция значительно меньше, чем в летние.

Теперь выведем на одном графике суммарные дистанции в километрах для каждой даты в заданных месяцах. Для примера возьмем февраль и июль как месяцы с наименьшей и наибольшей суммарной дистанцией.

```
SELECT
    round(sum(distance)) AS "Total distance, km",
    day_of_year,
FROM demo_geo_tracks_silver_time
    JOIN demo_geo_tracks_silver
        USING (track_id)
WHERE month IN (2,7)
GROUP BY 2
```

```
ORDER BY 2
```

```
Done in 8.1 sec.

+--------------------+-------------+
| total distance, km | day_of_year |
+--------------------+-------------+
| 160                | 02-01       |
+--------------------+-------------+
| 194                | 02-02       |
+--------------------+-------------+
| 137                | 02-03       |
+--------------------+-------------+
| 194                | 02-04       |
+--------------------+-------------+
| 245                | 02-05       |
+--------------------+-------------+
| ...                | ...         |
+--------------------+-------------+
60 rows
```



Теперь выведем рейтинг дней недели — упорядочим дни недели по суммарной дистанции в километрах за все время начиная с максимальных.

```
SELECT
    round(sum(distance)) AS "Total distance, km",
    day_name as "Day of week",
FROM demo_geo_tracks_silver_time
    JOIN demo_geo_tracks_silver
        USING track_id
GROUP BY 2
```

```
ORDER BY 1 DESC
```

```
Done in 8.4 sec.

+--------------------+-------------+
| total distance, km | day of week |
+--------------------+-------------+
| 21636              | Saturday    |
+--------------------+-------------+
| 19006              | Sunday      |
+--------------------+-------------+
| 13817              | Thursday    |
+--------------------+-------------+
| 12453              | Tuesday     |
+--------------------+-------------+
| 12400              | Wednesday   |
+--------------------+-------------+
| 10696              | Friday      |
+--------------------+-------------+
| 9386               | Monday      |
+--------------------+-------------+
```

Изменения показателей за весь период

Теперь отобразим на одном графике изменения показателей в активностях на протяжении всего времени с сегментацией по кварталам. Для этого используем функцию `date_trunc` с параметром `quarter`. Возьмем следующие показатели:

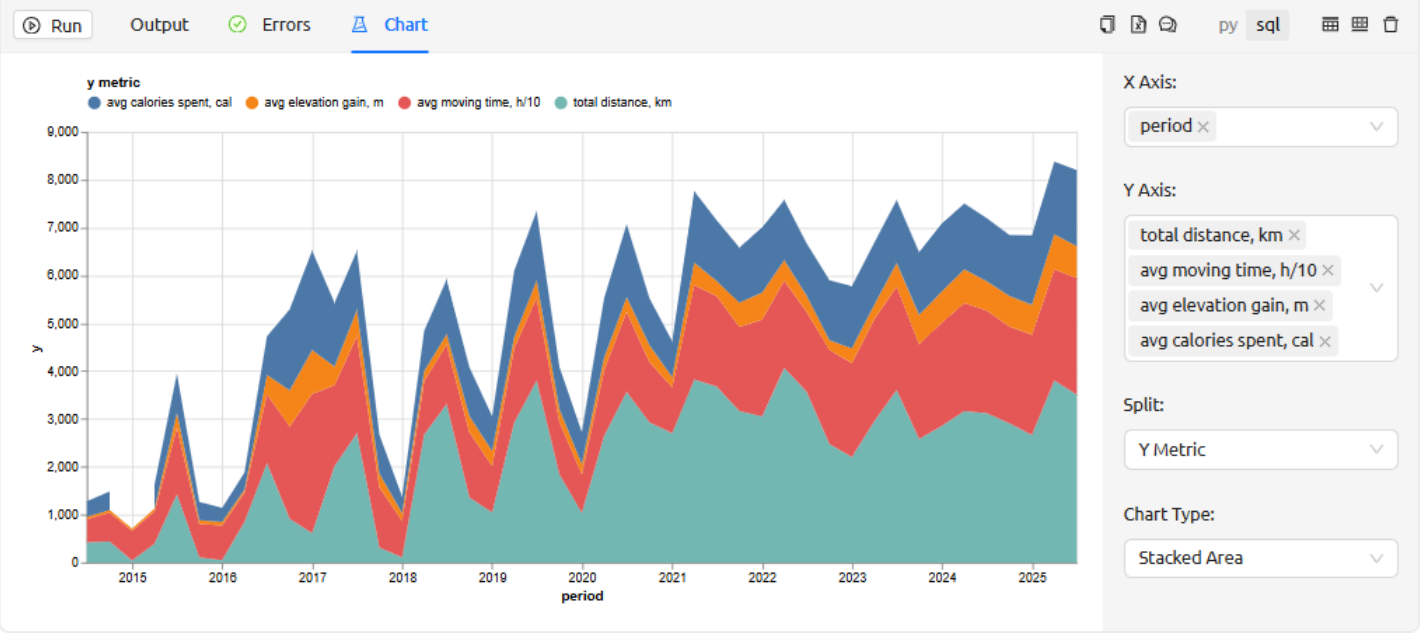
- суммарная дистанция
- среднее время одной активности
- средний набор высоты за одну активность
- среднее количество затраченных калорий за одну активность

```
SELECT
    round(avg("Elevation Gain")) AS "avg elevation gain, m",
    round(avg("Moving Time")/3.6) AS "avg moving time, h/10",
    round(avg(Calories)) AS "avg calories spent, cal",
    round(sum(distance)) AS "total distance, km",
    period
FROM (SELECT
        distance,
        "Elevation Gain",
        "Moving Time",
        Calories,
        date_trunc('quarter', time_stamp) AS period
        FROM demo_geo_tracks_silver
            JOIN demo_geo_tracks_silver_time
                USING (track_id)
```

```
        )
GROUP BY period
ORDER BY period
```

```
Done in 8.3 sec.

+----------------------+----------------------+-------------------------
+--------------------+------------+
| avg elevation gain, m | avg moving time, h/10 | avg calories spent, cal | total distance,
km | period     |
+----------------------+----------------------+-------------------------
+--------------------+------------+
| 52                   | 485                  | 322                     | 423
| 2014-07-01 |
+----------------------+----------------------+-------------------------
+--------------------+------------+
| 60                   | 597                  | 389                     | 438
| 2014-10-01 |
+----------------------+----------------------+-------------------------
+--------------------+------------+
| 46                   | 623                  | null                    | 41
| 2015-01-01 |
+----------------------+----------------------+-------------------------
+--------------------+------------+
| 60                   | 676                  | 502                     | 390
| 2015-04-01 |
+----------------------+----------------------+-------------------------
+--------------------+------------+
| 310                  | 1394                 | 816                     | 1426
| 2015-07-01 |
+----------------------+----------------------+-------------------------
+--------------------+------------+
| ...                  | ...                  | ...                     | ...
| ...        |
+----------------------+----------------------+-------------------------
+--------------------+------------+
45 rows
```

Так как изменения всех этих показателей в целом говорят об улучшении уровня подготовки спортсмена, то мы выберем тип графика **Stacked Area**, в котором значения показателей складываются:



На этом графике видно, во-первых, рост всех показателей, а во-вторых — уменьшение сезонного фактора со временем.

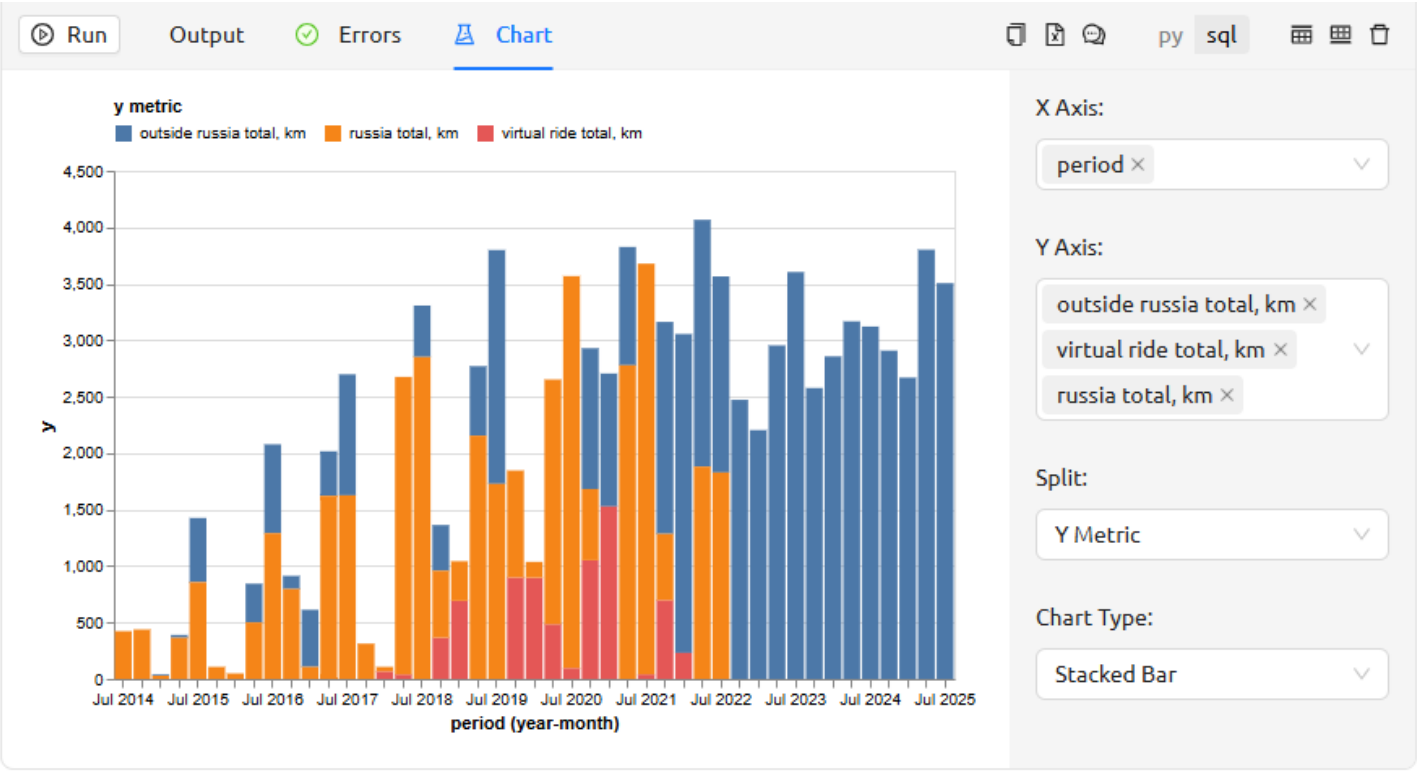
Выведем на одном графике статистику по суммарному пробегу в России / вне России и в виртуальных заездах с сегментацией по кварталам. Условия будем задавать с помощью выражения `CASE`.

```
SELECT
    round(sum(CASE WHEN country == 'Россия' AND NOT "Activity Type" = 'Virtual Ride'
        THEN distance END))
    AS "Russia total, km",
    round(sum(CASE WHEN NOT country == 'Россия' AND NOT "Activity Type" = 'Virtual Ride'
        THEN distance END))
    AS "Outside Russia total, km",
    round(sum(CASE WHEN "Activity Type" = 'Virtual Ride' THEN distance END))
    AS "Virtual Ride total, km",
    date_trunc('quarter', time_stamp) AS period
FROM demo_geo_tracks_silver
GROUP BY period
ORDER BY period
```

```
Done in 8.7 sec.

+------------------+--------------------------+-------------------------+------------+
| russia total, km | outside russia total, km | virtual ride total, km | period     |
+------------------+--------------------------+-------------------------+------------+
| 423              | null                     | null                    | 2014-07-01 |
+------------------+--------------------------+-------------------------+------------+
| 438              | null                     | null                    | 2014-10-01 |
+------------------+--------------------------+-------------------------+------------+
| 29               | 12                       | null                    | 2015-01-01 |
+------------------+--------------------------+-------------------------+------------+
| 365              | 25                       | null                    | 2015-04-01 |
+------------------+--------------------------+-------------------------+------------+
```

```
| 857              | 569                       | null                    | 2015-07-01 |
+------------------+---------------------------+-------------------------+------------+
| ...              | ...                       | ...                     | ...        |
+------------------+---------------------------+-------------------------+------------+
45 rows
```

Для такого графика хорошо подойдет тип **Stacked Bar**:



На этом графике можно заметить разные зависимости: сезонный фактор влияет на количество виртуальных заездов — зимой их больше; активности вне России сначала есть только в летние периоды, а позже активности вне России остаются единственными. Видно также общее снижение сезонного фактора, как и на предыдущем графике.

Анализ точек по их расстоянию от заданной

Нам известно, что пользователь, чей архив активностей мы изучаем, жил в Москве, поэтому попробуем проанализировать его активности в зависимости от расстояния от Москвы. Для этого возьмем координаты нулевого километра Москвы: `55.75579845052788, 37.617679973467204` и создадим таблицу, в которой для каждой активности будет вычислено расстояние ее начальной точки от Москвы в километрах. Используем для этого функции `ST_Point` и `ST_Distance_Sphere`.

```
CREATE OR REPLACE TABLE demo_geo_tracks_by_moscow AS
SELECT
    ST_Distance_Sphere(ST_Point(track_first_coord_lat, track_first_coord_lon),
                       ST_Point(55.75579845052788, 37.617679973467204)
                       )/1000
    AS km_from_moscow,
    track_id,
FROM demo_geo_tracks_silver;
```

```
SELECT * FROM demo_geo_tracks_by_moscow
    LIMIT 10;
```

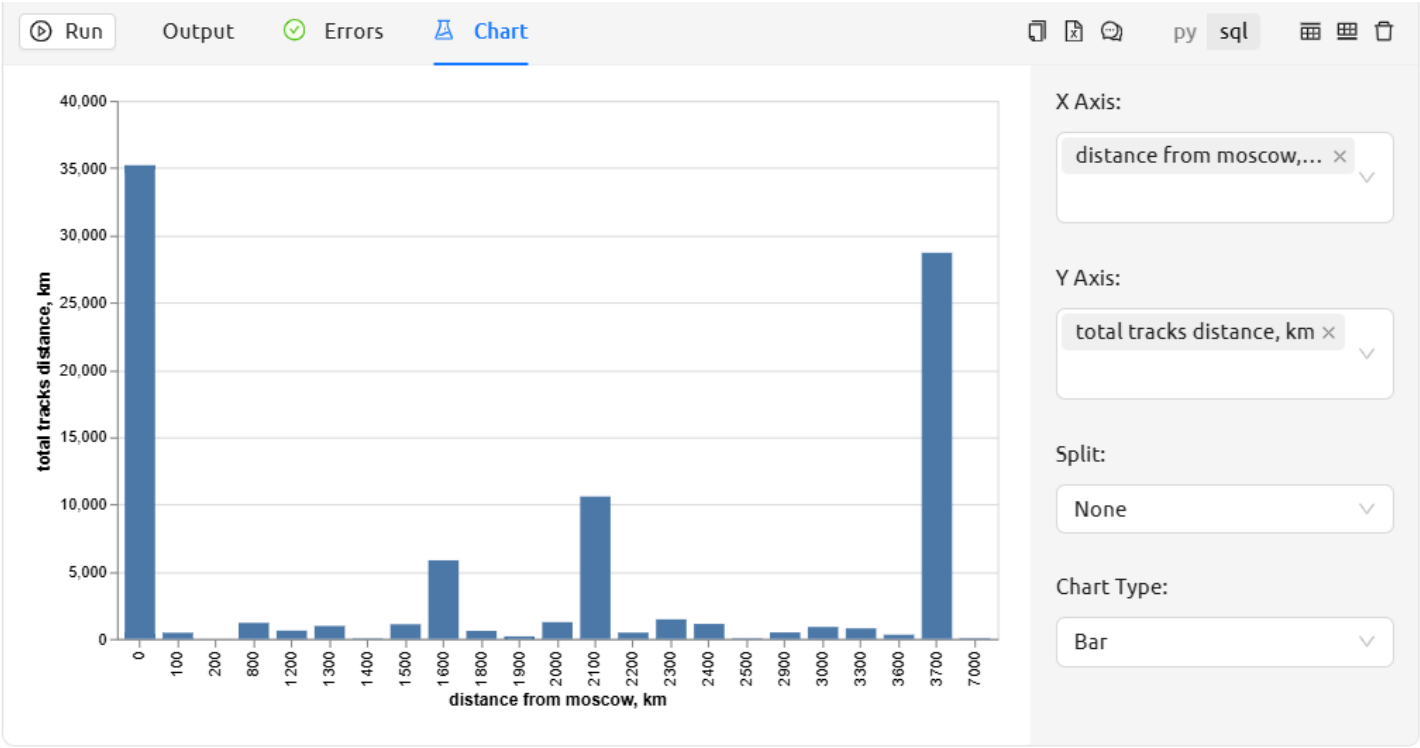
```
Done in 9.3 sec.

+--------------------+-----------------+
| km_from_moscow     | track_id        |
+--------------------+-----------------+
| 1614.8913738847054 | 8205566762.fit  |
+--------------------+-----------------+
| 8.000389153235107  | 5758691664.fit  |
+--------------------+-----------------+
| 2196.013494904038  | 7562241822.fit  |
+--------------------+-----------------+
| 6.0300128287340815 | 5752363946.fit  |
+--------------------+-----------------+
| 88.17899082303168  | 5774225138.fit  |
+--------------------+-----------------+
| 6.033077121995654  | 3602018438.fit  |
+--------------------+-----------------+
| 890.3163236663786  | 2665481253.fit  |
+--------------------+-----------------+
| 7501.918475014584  | 5061483340.fit  |
+--------------------+-----------------+
| 2.926600115019104  | 1842519266.fit  |
+--------------------+-----------------+
| 3770.6424243313804 | 11417255201.fit |
+--------------------+-----------------+
```

Посмотрим на распределение суммарной длины треков в зависимости от дальности начала трека от Москвы с шагом по 100 км. Для этого используем функцию `floor`.

```
SELECT
    floor(km_from_moscow/100)*100 AS "Distance from Moscow, km",
    round(sum(distance)) AS "Total tracks distance, km"
FROM demo_geo_tracks_by_moscow
    JOIN demo_geo_tracks_silver
        USING (track_id)
    WHERE NOT "Activity Type" = 'Virtual Ride'
GROUP BY 1
ORDER BY 1;
```

```
Done in 8.3 sec.

+--------------------------+---------------------------+
| distance from moscow, km | total tracks distance, km |
+--------------------------+---------------------------+
| 0                        | 35205                     |
+--------------------------+---------------------------+
```

```
| 100                       | 457                       |
+---------------------------+---------------------------+
| 200                       | 5                         |
+---------------------------+---------------------------+
| 800                       | 1191                      |
+---------------------------+---------------------------+
| 1200                      | 613                       |
+---------------------------+---------------------------+
| ...                       | ...                       |
+---------------------------+---------------------------+
29 rows
```



На графике мы видим, что есть несколько столбцов заметно выше соседних. Они соответствуют разным городам, из которых пользователь совершал регулярные тренировки. Кроме Москвы это три города:

- Малага — `36.71011042348239, -4.429510437480414`
- Берлин — `52.51622694136956, 13.377732359062023`
- Аланья — `36.544934513586334, 31.993451012738028`

Вычислим для всех активностей расстояния до этих городов:

```
CREATE OR REPLACE TABLE demo_geo_tracks_by_cities AS
SELECT
    track_id,
    distance,
    ST_Distance_Sphere(ST_Point(track_first_coord_lat, track_first_coord_lon),
                       ST_Point(55.75579845052788, 37.617679973467204)
                       )/1000
        AS km_from_moscow,
    ST_Distance_Sphere(ST_Point(track_first_coord_lat, track_first_coord_lon),
```

```
                        ST_Point(36.71011042348239, -4.429510437480414)
                        )/1000
        AS km_from_malaga,
    ST_Distance_Sphere(ST_Point(track_first_coord_lat, track_first_coord_lon),
                        ST_Point(52.51622694136956, 13.377732359062023)
                        )/1000
        AS km_from_berlin,
    ST_Distance_Sphere(ST_Point(track_first_coord_lat, track_first_coord_lon),
                        ST_Point(36.544934513586334, 31.993451012738028)
                        )/1000
        AS km_from_alanya,
FROM demo_geo_tracks_silver;

SELECT * FROM demo_geo_tracks_by_cities
    LIMIT 10;
```

```
+----------------+----------+--------------------+--------------------+--------------------
+--------------------+
| track_id       | distance | km_from_moscow     | km_from_malaga     | km_from_berlin     |
km_from_alanya     |
+----------------+----------+--------------------+--------------------+--------------------
+--------------------+
| 8205566762.fit | 51.04    | 1614.8913738847054 | 2235.249319891205  | 5.0145197840760245 |
2294.093997594629  |
+----------------+----------+--------------------+--------------------+--------------------
+--------------------+
| 5758691664.fit | 3.91     | 8.000389153235107  | 3787.2509319742358 | 1618.720214954235  |
2180.111417985302  |
+----------------+----------+--------------------+--------------------+--------------------
+--------------------+
| 7562241822.fit | 21.69    | 2196.013494904038  | 3104.0466510427395 | 2221.451446627775  |
127.78858146201004 |
+----------------+----------+--------------------+--------------------+--------------------
+--------------------+
| 5752363946.fit | 62.85    | 6.0300128287340815 | 3785.4665468428007 | 1616.7118606890685 |
2180.242964284234  |
+----------------+----------+--------------------+--------------------+--------------------
+--------------------+
| 5774225138.fit | 114.24   | 88.17899082303168  | 3707.0000971542977 | 1528.712391035889  |
2186.697705370458  |
+----------------+----------+--------------------+--------------------+--------------------
+--------------------+
| 3602018438.fit | 34.76    | 6.033077121995654  | 3785.4691666595877 | 1616.7152967306572 |
2180.24094049402   |
+----------------+----------+--------------------+--------------------+--------------------
+--------------------+
| 2665481253.fit | 43.88    | 890.3163236663786  | 3051.4212916480724 | 811.7057240823138  |
2370.646300384682  |
+----------------+----------+--------------------+--------------------+--------------------
+--------------------+
| 5061483340.fit | 16.87    | 7501.918475014584  | 5886.25300373306   | 6375.612772054101  |
```

```
8583.236350749687  |
+-----------------+----------+--------------------+--------------------+--------------------
+--------------------+
| 1842519266.fit  | 71.94    | 2.926600115019104  | 3780.985953704283  | 1612.7664280078745 |
2176.465565328427  |
+-----------------+----------+--------------------+--------------------+--------------------
+--------------------+
| 11417255201.fit | 30.91    | 3770.6424243313804 | 12.516233931245793 | 2232.9498881694144 |
3217.9897501282717 |
+-----------------+----------+--------------------+--------------------+--------------------
+--------------------+
```

Выведем статистику по суммарной дистанции треков с началом в разных городах. В качестве порогового значения расстояния возьмем `100 км`:

```
SELECT
    round(sum(CASE WHEN km_from_moscow < 100 THEN distance END))
        AS "Moscow total, km",
    round(sum(CASE WHEN km_from_malaga < 100 THEN distance END))
        AS "Malaga total, km",
    round(sum(CASE WHEN km_from_alanya < 100 THEN distance END))
        AS "Alanya total, km",
    round(sum(CASE WHEN km_from_berlin < 100 THEN distance END))
        AS "Berlin total, km",
    round(sum(CASE WHEN
        (
            km_from_moscow >= 100 AND
            km_from_malaga >= 100 AND
            km_from_alanya >= 100 AND
            km_from_berlin >= 100
        )
        THEN distance END))
        AS "Rest total, km",
    round(sum(distance)) AS "Total, km"
FROM demo_geo_tracks_by_cities
```

```
+------------------+------------------+------------------+------------------+----------------
+-----------+
| moscow total, km | malaga total, km | alanya total, km | berlin total, km | rest total, km
| total, km |
+------------------+------------------+------------------+------------------+----------------
+-----------+
| 35155            | 28969            | 8136             | 5841             | 21242
| 99343     |
+------------------+------------------+------------------+------------------+----------------
+-----------+
```

Отображение множества точек на карте

Для демонстрации отображения точек на карте создадим отдельную таблицу с точками одной активности в пределах Москвы с максимальной дистанцией за все время.

```
CREATE OR REPLACE TABLE demo_geo_points_for_map AS
    SELECT * FROM demo_geo_points_silver
        WHERE
        (track_id IN (
        SELECT track_id
            from demo_geo_tracks_by_distance
                JOIN demo_geo_tracks_silver
                    USING (track_id)
            WHERE km_from_moscow < 100
            ORDER BY distance DESC
            LIMIT 1
            )
    );

SELECT count(*) AS "Points for map total"
    FROM demo_geo_points_for_map;
```

```
Done in 9.7 sec.

+----------------------+
| points for map total |
+----------------------+
| 16458                |
+----------------------+
```

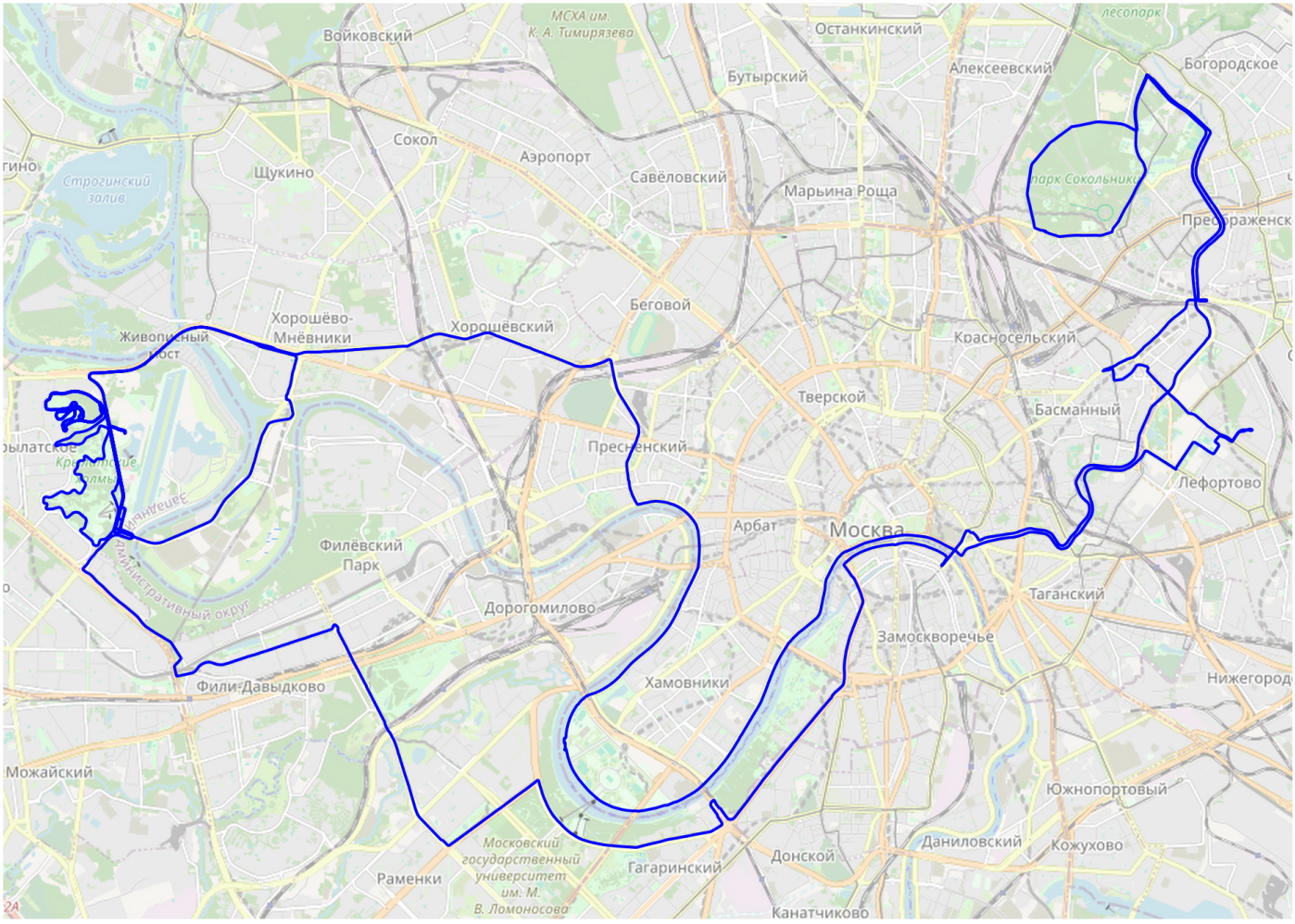
Теперь в ячейке типа Python запишем код, с помощью которого можно обратиться к созданной таблице с отфильтрованными точками и отобразить их на карте OSM, используя модуль Python TileMapBase:

```
import tngri
import pyarrow
import pyiceberg
from pyiceberg.catalog import load_catalog
from contextlib import suppress
import pandas as pd
import matplotlib.pyplot as plt
import tilemapbase
import warnings
warnings.filterwarnings("ignore")

def print_map(source_table_name):

    # Загружаем исходную таблицу
    source_table = catalog.load_table(source_table_name)
```

```
    # Преобразуем в DataFrame
    trip_df = source_table.scan().to_pandas()
    print(f'Размер исходной таблицы: {trip_df.shape}')

    tilemapbase.init(create=True)

    # Размер полей карты вокруг крайних точек трека
    expand=0.01
    extent = tilemapbase.Extent.from_lonlat(
        trip_df.longitude.min() - expand,
        trip_df.longitude.max() + expand,
        trip_df.latitude.min() - expand,
        trip_df.latitude.max() + expand,
    )

    track_names = trip_df['track_id'].unique()
    print(f'Количество треков: {len(track_names)}')

    # Выводим суммарную дистанцию треков в исходной таблице
    tracks_dist = tngri.sql(f'''
    select round(sum(distance)) as "Суммарная дистанция треков, км"
    from demo_geo_tracks_silver
    where track_id in (
            select track_id
            from {source_table_name})
    ''')
    print(f'Суммарная дистанция треков: {tracks_dist[0,0]} км')

    track_list = []
    for track_name in track_names:
        track = trip_df.loc[trip_df['track_id'] == track_name].apply(
            lambda x: tilemapbase.project(x.longitude, x.latitude), axis=1
        ).apply(pd.Series)
        track.columns = ["x", "y"]
        track_list.append(track)

    tiles = tilemapbase.tiles.build_OSM()

    fig, ax = plt.subplots(figsize=(8, 8), dpi=300)

    # height -- детализация карты
    plotter = tilemapbase.Plotter(extent, tiles, height=600)

    # alpha -- замутнение карты
    plotter.plot(ax, tiles, alpha=0.6)

    # Если треков несколько, то толщину линии делаем меньше
    if len(track_names) == 1:
        width = 1
    else:
        width = 0.5
    for track in track_list:
```

```
        ax.plot(track.x, track.y, color='blue', linewidth=width)

    plt.axis('off')
    fig.get_figure()

print_map('demo_geo_points_for_map')
```

```
Done in 30 sec.

Размер исходной таблицы: (16458, 8)
Количество треков: 1
Суммарная дистанция треков: 135.0 км
```



Обратите внимание, что для вывода суммарной дистанции отображаемых треков мы используем вызов запроса `SELECT` в ячейке Python через функцию `tngri.sql`.

Теперь перезапишем таблицу с координатами точек для отображения на карте, выбрав все активности с началом в Берлине, чтобы отобразить их все на одной карте.

```
CREATE OR REPLACE TABLE demo_geo_points_for_map AS
    SELECT
        latitude,
        longitude,
```

```
        track_id
    FROM demo_geo_points_silver
    WHERE
    (track_id IN (
        SELECT track_id
            FROM demo_geo_tracks_by_distance
            WHERE km_from_berlin < 100
        )
    );

SELECT * FROM demo_geo_points_for_map
    LIMIT 10;
```

```
Done in 9.6 sec.

+--------------------+--------------------+----------------+
| latitude           | longitude          | track_id       |
+--------------------+--------------------+----------------+
| 52.529693618416786 | 13.444848032668233 | 9440992383.fit |
+--------------------+--------------------+----------------+
| 52.52970124594867  | 13.44483470544219  | 9440992383.fit |
+--------------------+--------------------+----------------+
| 52.52970124594867  | 13.44481467269361  | 9440992383.fit |
+--------------------+--------------------+----------------+
| 52.52969739027321  | 13.444789862260222 | 9440992383.fit |
+--------------------+--------------------+----------------+
| 52.5296859908849   | 13.444773685187101 | 9440992383.fit |
+--------------------+--------------------+----------------+
| 52.52967827953398  | 13.4447565022856   | 9440992383.fit |
+--------------------+--------------------+----------------+
| 52.529674507677555 | 13.444727919995785 | 9440992383.fit |
+--------------------+--------------------+----------------+
| 52.529674507677555 | 13.44469740986824  | 9440992383.fit |
+--------------------+--------------------+----------------+
| 52.529670652002096 | 13.444674527272582 | 9440992383.fit |
+--------------------+--------------------+----------------+
| 52.529670652002096 | 13.444655416533351 | 9440992383.fit |
+--------------------+--------------------+----------------+
```

Выведем количество точек в созданной таблице:

```
SELECT count(*) AS "Points for map total"
    FROM demo_geo_points_for_map
```

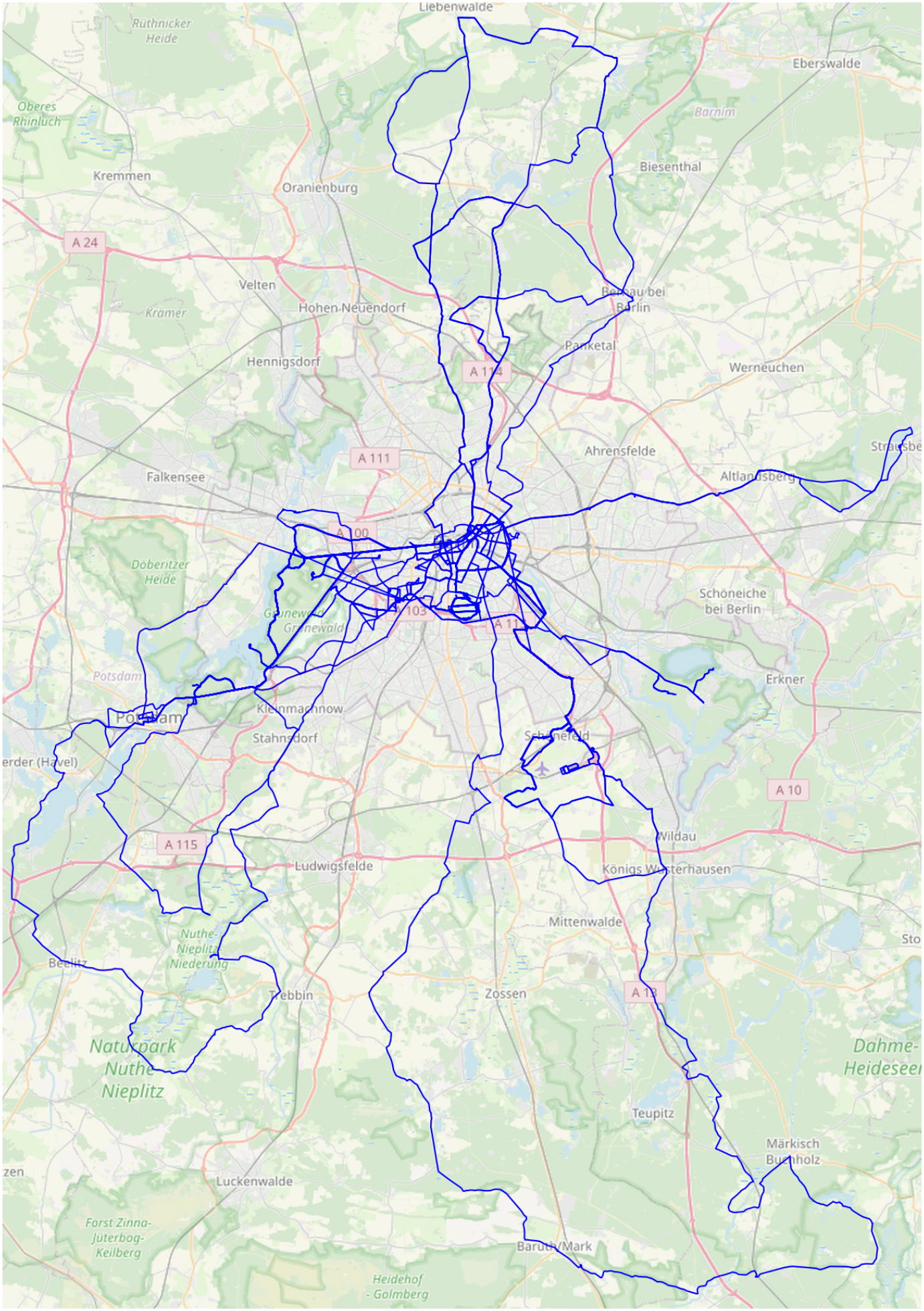
```
+----------------------+
| points for map total |
+----------------------+
| 708616               |
```

```
+----------------------+
```

Запустим в ячейке Python выполнение созданной ранее функции `print_map` для обновленной таблицы с тем же именем и отобразим карту со всеми активностями в Берлине:

```
print_map('demo_geo_points_for_map')
```

```
Done in 1 min. 6 sec.

Размер исходной таблицы: (708616, 3)
Количество треков: 108
Суммарная дистанция треков: 5841.0 км
```



Теперь отобразим на карте все треки в радиусе 3 км от Велотрека в Москве.

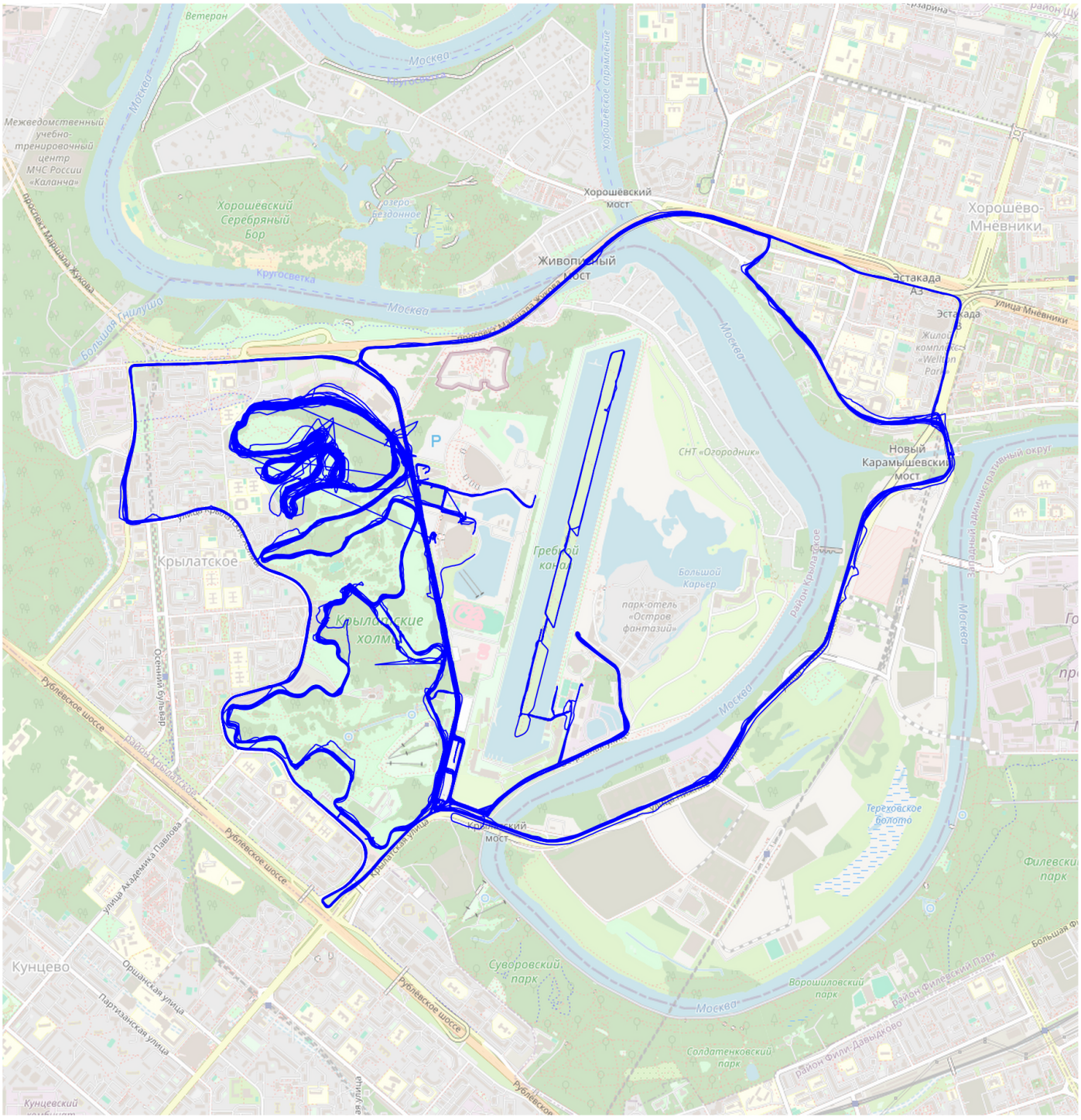
```
CREATE OR REPLACE TABLE demo_geo_points_for_map AS
SELECT
    latitude,
    longitude,
    track_id
FROM demo_geo_points_silver
WHERE track_id NOT IN
    (SELECT DISTINCT track_id
    FROM demo_geo_points_silver
    WHERE ST_Distance_Sphere(ST_Point(latitude, longitude),
                            ST_Point(55.76314006886495, 37.43303308055759)
                            )/1000 > 3);

SELECT count(*) AS "Points for map total"
    FROM demo_geo_points_for_map;
```

```
+----------------------+
| points for map total |
+----------------------+
| 215466               |
+----------------------+
```

Запустим отображение карты:

```
print_map('demo_geo_points_for_map')
```

```
Done in 29 sec.

Размер исходной таблицы: (215466, 3)
Количество треков: 43
Суммарная дистанция треков: 2063.0 км
```



Анализ текста на примере дата-сета FineWeb

В этом примере демонстрируется, как с помощью Tengri можно анализировать большие текстовые корпуса.

Для анализа мы будем использовать дата-сет FineWeb. Он состоит из более чем **18,5 трлн** токенов очищенных и дедуплицированных англоязычных текстов из веба от CommonCrawl. По сути этот дата-сет состоит из слепков всего англоязычного интернета на конкретный момент времени (с 2013 по 2025 годы). Он используется в основном для обучения LLM.

Загрузка исходных данных

Дата-сет **FineWeb** предоставляет разные возможности для загрузки данных. Есть следующие варианты:

1. Загрузить весь дата-сет целиком — около **108 ТБ**.
2. Загрузить определенные дампы — слепки на конкретный временной период. Каждый дамп занимает около **500 ГБ**.
3. Загрузить сэмплы определенного объема — случайно выбранные подмножества данных, взятые по всем дампам.
   Возможный объем сэмплов:
   - `10BT` — 10 млрд токенов gpt2 **(27,6 ГБ)**
   - `100BT` — 100 млрд токенов gpt2 **(277,4 ГБ)**
   - `350BT` — 350 млрд токенов gpt2 **(388 ГБ)**

Для этой демонстрации мы выбрали третий вариант — загрузку по произвольным сэмплам, взятым по всем временнЫм периодам, так как наш анализ будет опираться на возможную разницу показателей по разным периодам.

Приступим к загрузке. Для начала в ячейке типа Python импортируем необходимые для работы модули:

```
import tngri
import os
import datetime
from pathlib import Path

os.system('pip install huggingface_hub')
from huggingface_hub import snapshot_download
```

▼ *Посмотреть вывод*

```
Requirement already satisfied: huggingface_hub in
/data/python_venv/shared/lib/python3.13/site-packages (0.35.3)
Requirement already satisfied: filelock in /data/python_venv/shared/lib/python3.13/site-
packages (from huggingface_hub) (3.20.0)
Requirement already satisfied: fsspec>=2023.5.0 in
/data/python_venv/shared/lib/python3.13/site-packages (from huggingface_hub) (2025.9.0)
Requirement already satisfied: packaging>=20.9 in
/data/python_venv/shared/lib/python3.13/site-packages (from huggingface_hub) (25.0)
Requirement already satisfied: pyyaml>=5.1 in /data/python_venv/shared/lib/python3.13/site-
packages (from huggingface_hub) (6.0.3)
Requirement already satisfied: requests in /data/python_venv/shared/lib/python3.13/site-
packages (from huggingface_hub) (2.32.4)
Requirement already satisfied: tqdm>=4.42.1 in
/data/python_venv/shared/lib/python3.13/site-packages (from huggingface_hub) (4.67.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/data/python_venv/shared/lib/python3.13/site-packages (from huggingface_hub) (4.14.1)
```

```
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in
/data/python_venv/shared/lib/python3.13/site-packages (from huggingface_hub) (1.1.10)
Requirement already satisfied: charset_normalizer<4,>=2 in
/data/python_venv/shared/lib/python3.13/site-packages (from requests->huggingface_hub)
(3.4.2)
Requirement already satisfied: idna<4,>=2.5 in
/data/python_venv/shared/lib/python3.13/site-packages (from requests->huggingface_hub)
(3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/data/python_venv/shared/lib/python3.13/site-packages (from requests->huggingface_hub)
(2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in
/data/python_venv/shared/lib/python3.13/site-packages (from requests->huggingface_hub)
(2025.6.15)
```

Теперь опишем функцию на Python для скачивания сэмпла указанного размера и загрузки его в указанную таблицу. Для загрузки таблицы воспользуемся функциями `tngri.upload_file` и `tngri.sql`.

```
def fineweb_download(sample_size, table_name):
    my_dir = f'fineweb_{sample_size}'

①
    if os.path.exists(my_dir) and os.path.isdir(my_dir):
        shutil.rmtree(my_dir)

②
    snapshot_download("HuggingFaceFW/fineweb",
                      repo_type="dataset",
                      local_dir=my_dir,
                      allow_patterns=f"sample/{sample_size}/*")

③
    files_total = len([x for x in Path(my_dir).rglob('*') \
        if os.path.isfile(x) and str(x).endswith('.parquet')])

    created = False
    file_count = 0
    start = datetime.datetime.now()

④
    for subdir, dirs, files in os.walk(my_dir):
        for file in files:
            filepath = os.path.join(subdir, file)

            if not filepath.endswith('.parquet'):
                continue

            file_count += 1
            print(f'File {file_count}/{files_total}: {filepath}')
            print('Size:', size(os.path.getsize(filepath)))

⑤
```

```
            uploaded_file = tngri.upload_file(filepath)

⑥
            if not created:
                tngri.sql(f'create or replace table {table_name} as \
                            select * from read_parquet("{uploaded_file}")')
                created = True
            else:
                tngri.sql(f'insert into {table_name} select * \
                            from read_parquet("{uploaded_file}")')

⑦
            count = tngri.sql(f'select count(*) from {table_name}')
            print(f'Created table row count: {count[0,0]}')

            print(f'Time from start: {str(datetime.datetime.now() - start)}')
    print(f'Uploaded in {str(datetime.datetime.now() - start)}')
```

```
Done in 4 sec.
```

① Чистим временную папку (чтобы при каждом запуске данные заливались с нуля)

② Запускаем загрузку

③ Считаем, сколько файлов parquet загрузилось

④ Проходим по всем скачанным файлам и загружаем данные в таблицу

⑤ Загружаем файл в Tengri через функцию `tngri.upload_file`

⑥ На первом проходе создаем таблицу, на остальных — только добавляем данные; используем функцию `tngri.sql`

⑦ Выводим промежуточную длину наполняемой таблицы

Начнем с загрузки наименьшего по размеру сэмпла — `10BT`:

```
fineweb_download('10BT', 'demo.fineweb_sample_10BT')
```

▼ *Посмотреть вывод*

```
Done in 22 min.

Fetching 15 files:   0%|          | 0/15 [00:00<?, ?it/s]
Fetching 15 files:   7%|▋         | 1/15 [07:08<1:39:57, 428.36s/it]
Fetching 15 files:  20%|██        | 3/15 [09:03<30:44, 153.72s/it]
Fetching 15 files:  53%|█████▎    | 8/15 [09:17<05:09, 44.17s/it]
Fetching 15 files:  60%|██████    | 9/15 [10:41<05:06, 51.08s/it]
Fetching 15 files:  67%|██████▋   | 10/15 [11:00<03:43, 44.63s/it]
Fetching 15 files:  73%|███████▎  | 11/15 [12:23<03:32, 53.06s/it]
Fetching 15 files:  80%|████████  | 12/15 [12:43<02:15, 45.16s/it]
Fetching 15 files:  93%|█████████▎| 14/15 [12:45<00:27, 27.35s/it]
Fetching 15 files: 100%|██████████| 15/15 [12:45<00:00, 51.04s/it]
```

```
File 1/15: fineweb_10BT/sample/10BT/011_00000.parquet
Size: 1G
Created table row count: 1033752
Time from start: 0:00:29.632723
File 2/15: fineweb_10BT/sample/10BT/014_00000.parquet
Size: 548M
Created table row count: 1315416
Time from start: 0:00:45.758067
File 3/15: fineweb_10BT/sample/10BT/002_00000.parquet
Size: 2G
Created table row count: 2362045
Time from start: 0:01:17.201461
File 4/15: fineweb_10BT/sample/10BT/000_00000.parquet
Size: 1G
Created table row count: 3410626
Time from start: 0:01:48.977265
File 5/15: fineweb_10BT/sample/10BT/009_00000.parquet
Size: 2G
Created table row count: 4449266
Time from start: 0:02:20.716597
File 6/15: fineweb_10BT/sample/10BT/003_00000.parquet
Size: 1G
Created table row count: 5495871
Time from start: 0:02:52.690474
File 7/15: fineweb_10BT/sample/10BT/012_00000.parquet
Size: 2G
Created table row count: 6530586
Time from start: 0:03:25.198067
File 8/15: fineweb_10BT/sample/10BT/008_00000.parquet
Size: 1G
Created table row count: 7565226
Time from start: 0:03:56.557413
File 9/15: fineweb_10BT/sample/10BT/007_00000.parquet
Size: 2G
Created table row count: 8608873
Time from start: 0:04:30.311380
File 10/15: fineweb_10BT/sample/10BT/013_00000.parquet
Size: 2G
Created table row count: 9655610
Time from start: 0:05:04.350260
File 11/15: fineweb_10BT/sample/10BT/005_00000.parquet
Size: 1G
Created table row count: 10702268
Time from start: 0:05:38.026093
File 12/15: fineweb_10BT/sample/10BT/006_00000.parquet
Size: 1G
Created table row count: 11745931
Time from start: 0:06:11.046026
File 13/15: fineweb_10BT/sample/10BT/001_00000.parquet
Size: 1G
Created table row count: 12792546
Time from start: 0:06:43.937408
```

```
File 14/15: fineweb_10BT/sample/10BT/004_00000.parquet
Size: 1G
Created table row count: 13833162
Time from start: 0:07:17.067437
File 15/15: fineweb_10BT/sample/10BT/010_00000.parquet
Size: 2G
Created table row count: 14868862
Time from start: 0:07:50.150215
Uploaded in 0:07:50.150305
```

За 22 минуты мы загрузили сэмпл наименьшего размера и получили таблицу размером в **14,8 млн** строк. Выведем несколько строк этой таблицы, чтобы проверить загруженные данные:

```
SELECT * FROM demo.fineweb_sample_10BT
    LIMIT 5
```

```
+-----------------------------------------------------
+-----------------------------------------------+-----------------
+-----------------------------------------------------------------------------------------------------
-------------+----------------------+
-----------------------------------------------------------------------------------------------------
-----------------------------------+----------+----------------+-------------+
| text                                                  | id
| dump            | url
| date                 | file_path
| language | language_score | token_count |
+-----------------------------------------------------
+-----------------------------------------------+-----------------
+-----------------------------------------------------------------------------------------------------
-------------+----------------------+
-----------------------------------------------------------------------------------------------------
-----------------------------------+----------+----------------+-------------+
| Thank you guys for the responses! Close combat is  ... | <urn:uuid:0d2b3414-c50e-47ec-a145-
1ecd22a348b8> | CC-MAIN-2020-24 |
http://www.40konline.com/index.php?topic=223837.0;prev_next=prev
| 2020-05-25T11:49:14Z | s3://commoncrawl/crawl-data/CC-MAIN-2020-
24/segments/1590347388427.15/warc/CC-MAIN-20200525095005-20200525125005-00534.warc.gz | en
| 0,9705834389   | 320         |
+-----------------------------------------------------
+-----------------------------------------------+-----------------
+-----------------------------------------------------------------------------------------------------
-------------+----------------------+
-----------------------------------------------------------------------------------------------------
-----------------------------------+----------+----------------+-------------+
| MEDAL OF HONOR RECIPIENT THOMAS J. KINSMAN PASSES  ... | <urn:uuid:a0f3a248-df51-4269-84d1-
d26d8730155c> | CC-MAIN-2020-24 | https://themedalofhonor.com/press_releases/medal-of-honor-
recipient-thomas-j-kinsman-passes-away-at-72/ | 2020-05-25T12:07:08Z |
s3://commoncrawl/crawl-data/CC-MAIN-2020-24/segments/1590347388427.15/warc/CC-MAIN-
20200525095005-20200525125005-00534.warc.gz | en       | 0,9769160748   | 470         |
+-----------------------------------------------------
```

```
+-----------------------------------------------+----------------
+-----------------------------------------------------------------------------------------------
-------------+----------------------+
-----------------------------------------------------------------------------------------------
-----------------------------------+----------+----------------+-------------+
| These Buckwheat and Chocolate Bliss Balls are an e ... | <urn:uuid:a90e6084-06a0-4363-9018-
58514f5df315> | CC-MAIN-2020-24 | https://www.dhmaya.com/buckwheat-and-chocolate-bliss-balls/
| 2020-05-25T11:51:30Z | s3://commoncrawl/crawl-data/CC-MAIN-2020-
24/segments/1590347388427.15/warc/CC-MAIN-20200525095005-20200525125005-00534.warc.gz | en
| 0,7672950029   | 974         |
+-----------------------------------------------------
+-----------------------------------------------+----------------
+-----------------------------------------------------------------------------------------------
-------------+----------------------+
-----------------------------------------------------------------------------------------------
-----------------------------------+----------+----------------+-------------+
| PANAMA CITY BEACH, Fla. (WJHG/WECP) - A three-car  ... | <urn:uuid:3fa75684-1dda-406a-a126-
9cbad9dab78c> | CC-MAIN-2020-24 | https://www.wjhg.com/content/news/Three-508574761.html
| 2020-05-25T09:54:25Z | s3://commoncrawl/crawl-data/CC-MAIN-2020-
24/segments/1590347388427.15/warc/CC-MAIN-20200525095005-20200525125005-00534.warc.gz | en
| 0,9612885118   | 99          |
+-----------------------------------------------------
+-----------------------------------------------+----------------
+-----------------------------------------------------------------------------------------------
-------------+----------------------+
-----------------------------------------------------------------------------------------------
-----------------------------------+----------+----------------+-------------+
| New ways to say goodbye How do you pay tribute to  ... | <urn:uuid:0645fbdd-e306-4dec-b308-
fc35202c8959> | CC-MAIN-2020-24 | http://www.petcremationservices.co.uk/new-ways-to-say-
goodbye/                                          | 2020-05-27T08:04:27Z |
s3://commoncrawl/crawl-data/CC-MAIN-2020-24/segments/1590347392142.20/warc/CC-MAIN-
20200527075559-20200527105559-00134.warc.gz | en       | 0,9408096671   | 251         |
+-----------------------------------------------------
+-----------------------------------------------+----------------
+-----------------------------------------------------------------------------------------------
-------------+----------------------+
-----------------------------------------------------------------------------------------------
-----------------------------------+----------+----------------+-------------+
```

Теперь мы можем увидеть все колонки, которые есть в исходных данных. Их подробные описания можно увидеть здесь.

Поскольку загрузка наименьшего сэмпла заняла не очень много времени, то попробуем скачать сэмпл большего размера — `100BT`.

```
fineweb_download('100BT', 'demo.fineweb_sample_100BT')
```

▼ *Посмотреть вывод*

```
Done in 8 h. 32 min.
```

```
Fetching 150 files:   0%|          | 0/150 [00:00<?, ?it/s]
Fetching 150 files:   1%|          | 1/150 [07:23<18:21:06, 443.40s/it]
Fetching 150 files:   6%|▌         | 9/150 [10:39<2:16:43, 58.18s/it]
Fetching 150 files:   7%|▋         | 10/150 [10:54<2:01:29, 52.07s/it]
Fetching 150 files:   7%|▋         | 11/150 [14:01<2:53:22, 74.84s/it]
Fetching 150 files:   8%|▊         | 12/150 [14:30<2:31:35, 65.91s/it]
Fetching 150 files:  10%|▉         | 15/150 [15:09<1:34:21, 41.94s/it]
Fetching 150 files:  12%|█▏        | 18/150 [15:25<1:00:30, 27.50s/it]
Fetching 150 files:  13%|█▎        | 19/150 [19:38<2:17:42, 63.07s/it]
Fetching 150 files:  13%|█▎        | 20/150 [21:06<2:26:45, 67.74s/it]
Fetching 150 files:  15%|█▍        | 22/150 [21:28<1:42:47, 48.18s/it]
Fetching 150 files:  16%|█▌        | 24/150 [21:43<1:13:02, 34.79s/it]
Fetching 150 files:  17%|█▋        | 25/150 [22:27<1:16:02, 36.50s/it]
Fetching 150 files:  17%|█▋        | 26/150 [23:14<1:19:54, 38.67s/it]
Fetching 150 files:  18%|█▊        | 27/150 [24:50<1:46:51, 52.12s/it]
Fetching 150 files:  19%|█▊        | 28/150 [26:08<1:59:16, 58.66s/it]
Fetching 150 files:  19%|█▉        | 29/150 [27:47<2:19:51, 69.35s/it]
Fetching 150 files:  20%|██        | 30/150 [29:17<2:30:03, 75.03s/it]
Fetching 150 files:  22%|██▏       | 33/150 [29:58<1:21:44, 41.92s/it]
Fetching 150 files:  23%|██▎       | 35/150 [31:06<1:15:14, 39.25s/it]
Fetching 150 files:  24%|██▍       | 36/150 [31:27<1:07:39, 35.61s/it]
Fetching 150 files:  25%|██▍       | 37/150 [34:00<1:55:06, 61.12s/it]
Fetching 150 files:  26%|██▌       | 39/150 [35:24<1:39:33, 53.82s/it]
Fetching 150 files:  27%|██▋       | 40/150 [36:01<1:32:07, 50.25s/it]
Fetching 150 files:  28%|██▊       | 42/150 [36:07<58:08, 32.30s/it]
Fetching 150 files:  29%|██▉       | 43/150 [38:11<1:32:29, 51.86s/it]
Fetching 150 files:  30%|███       | 45/150 [40:15<1:37:26, 55.68s/it]
Fetching 150 files:  31%|███▏      | 47/150 [41:30<1:24:42, 49.35s/it]
Fetching 150 files:  32%|███▏      | 48/150 [43:07<1:40:08, 58.91s/it]
Fetching 150 files:  33%|███▎      | 50/150 [43:13<1:04:03, 38.43s/it]
Fetching 150 files:  34%|███▍      | 51/150 [44:13<1:10:53, 42.96s/it]
Fetching 150 files:  35%|███▍      | 52/150 [45:20<1:19:01, 48.39s/it]
Fetching 150 files:  35%|███▌      | 53/150 [46:34<1:28:38, 54.83s/it]
Fetching 150 files:  36%|███▌      | 54/150 [47:37<1:30:56, 56.84s/it]
Fetching 150 files:  37%|███▋      | 55/150 [48:01<1:15:55, 47.95s/it]
Fetching 150 files:  37%|███▋      | 56/150 [49:44<1:38:56, 63.15s/it]
Fetching 150 files:  39%|███▉      | 59/150 [51:34<1:14:05, 48.85s/it]
Fetching 150 files:  40%|████      | 60/150 [52:13<1:10:13, 46.81s/it]
Fetching 150 files:  41%|████      | 61/150 [53:26<1:18:17, 52.78s/it]
Fetching 150 files:  41%|████▏     | 62/150 [54:06<1:12:40, 49.55s/it]
Fetching 150 files:  42%|████▏     | 63/150 [54:33<1:03:22, 43.71s/it]
Fetching 150 files:  43%|████▎     | 64/150 [55:03<57:20, 40.01s/it]
Fetching 150 files:  43%|████▎     | 65/150 [55:07<42:23, 29.92s/it]
Fetching 150 files:  44%|████▍     | 66/150 [55:37<42:11, 30.14s/it]
Fetching 150 files:  45%|████▍     | 67/150 [58:17<1:33:23, 67.51s/it]
Fetching 150 files:  45%|████▌     | 68/150 [58:49<1:18:19, 57.31s/it]
Fetching 150 files:  46%|████▌     | 69/150 [59:22<1:07:25, 49.95s/it]
Fetching 150 files:  47%|████▋     | 70/150 [1:00:04<1:03:30, 47.63s/it]
Fetching 150 files:  47%|████▋     | 71/150 [1:01:11<1:10:25, 53.49s/it]
Fetching 150 files:  48%|████▊     | 72/150 [1:03:19<1:38:13, 75.56s/it]
Fetching 150 files:  50%|█████     | 75/150 [1:04:56<1:04:15, 51.40s/it]
Fetching 150 files:  51%|█████▏    | 77/150 [1:05:55<53:19, 43.82s/it]
Fetching 150 files:  52%|█████▏    | 78/150 [1:07:22<1:02:45, 52.30s/it]
```

```
Fetching 150 files:  53%|□□□□□□□    | 80/150 [1:08:40<55:26, 47.53s/it]
Fetching 150 files:  55%|□□□□□□□    | 82/150 [1:10:14<53:41, 47.38s/it]
Fetching 150 files:  55%|□□□□□□□    | 83/150 [1:12:02<1:06:12, 59.29s/it]
Fetching 150 files:  57%|□□□□□□□    | 85/150 [1:13:03<53:02, 48.96s/it]
Fetching 150 files:  57%|□□□□□□□    | 86/150 [1:14:07<55:22, 51.92s/it]
Fetching 150 files:  58%|□□□□□□□    | 87/150 [1:14:38<49:29, 47.14s/it]
Fetching 150 files:  59%|□□□□□□□    | 88/150 [1:16:25<1:03:56, 61.89s/it]
Fetching 150 files:  61%|□□□□□□□    | 91/150 [1:17:31<40:48, 41.51s/it]
Fetching 150 files:  61%|□□□□□□□□   | 92/150 [1:18:58<48:41, 50.37s/it]
Fetching 150 files:  62%|□□□□□□□□   | 93/150 [1:19:23<42:34, 44.81s/it]
Fetching 150 files:  63%|□□□□□□□□   | 94/150 [1:20:24<45:25, 48.66s/it]
Fetching 150 files:  63%|□□□□□□□□   | 95/150 [1:21:27<47:52, 52.22s/it]
Fetching 150 files:  65%|□□□□□□□□   | 97/150 [1:22:05<33:56, 38.42s/it]
Fetching 150 files:  65%|□□□□□□□□   | 98/150 [1:22:34<31:17, 36.11s/it]
Fetching 150 files:  66%|□□□□□□□□   | 99/150 [1:24:04<42:02, 49.46s/it]
Fetching 150 files:  67%|□□□□□□□□   | 100/150 [1:24:40<38:21, 46.04s/it]
Fetching 150 files:  67%|□□□□□□□□   | 101/150 [1:25:35<39:37, 48.52s/it]
Fetching 150 files:  68%|□□□□□□□□   | 102/150 [1:27:12<49:36, 62.00s/it]
Fetching 150 files:  69%|□□□□□□□□   | 103/150 [1:27:36<40:04, 51.16s/it]
Fetching 150 files:  69%|□□□□□□□□   | 104/150 [1:29:23<51:39, 67.38s/it]
Fetching 150 files:  71%|□□□□□□□□   | 106/150 [1:30:13<35:22, 48.24s/it]
Fetching 150 files:  71%|□□□□□□□□□  | 107/150 [1:31:05<35:06, 48.98s/it]
Fetching 150 files:  72%|□□□□□□□□□  | 108/150 [1:32:08<36:57, 52.80s/it]
Fetching 150 files:  73%|□□□□□□□□□  | 110/150 [1:32:40<24:39, 36.98s/it]
Fetching 150 files:  74%|□□□□□□□□□  | 111/150 [1:32:49<19:48, 30.47s/it]
Fetching 150 files:  75%|□□□□□□□□□  | 112/150 [1:34:38<31:47, 50.21s/it]
Fetching 150 files:  76%|□□□□□□□□□  | 114/150 [1:37:44<40:50, 68.07s/it]
Fetching 150 files:  77%|□□□□□□□□□  | 116/150 [1:39:33<35:42, 63.00s/it]
Fetching 150 files:  79%|□□□□□□□□□  | 119/150 [1:39:37<18:20, 35.50s/it]
Fetching 150 files:  80%|□□□□□□□□□  | 120/150 [1:40:04<17:04, 34.15s/it]
Fetching 150 files:  81%|□□□□□□□□□  | 121/150 [1:40:59<18:29, 38.25s/it]
Fetching 150 files:  81%|□□□□□□□□□□ | 122/150 [1:42:55<25:52, 55.45s/it]
Fetching 150 files:  82%|□□□□□□□□□□ | 123/150 [1:44:12<27:19, 60.71s/it]
Fetching 150 files:  83%|□□□□□□□□□□ | 125/150 [1:46:21<25:55, 62.24s/it]
Fetching 150 files:  84%|□□□□□□□□□□ | 126/150 [1:48:13<29:22, 73.42s/it]
Fetching 150 files:  87%|□□□□□□□□□□ | 130/150 [1:49:04<13:05, 39.26s/it]
Fetching 150 files:  88%|□□□□□□□□□□ | 132/150 [1:49:23<09:13, 30.76s/it]
Fetching 150 files:  89%|□□□□□□□□□□ | 133/150 [1:51:55<14:31, 51.28s/it]
Fetching 150 files:  90%|□□□□□□□□□□ | 135/150 [1:51:56<08:40, 34.69s/it]
Fetching 150 files:  91%|□□□□□□□□□□ | 136/150 [1:54:37<13:40, 58.63s/it]
Fetching 150 files: 100%|□□□□□□□□□□□| 150/150 [1:54:41<00:00, 12.47s/it]
Fetching 150 files: 100%|□□□□□□□□□□□| 150/150 [1:54:41<00:00, 45.87s/it]

File 1/150: fineweb_100BT/sample/100BT/010_00006.parquet
Size: 2G
Created table row count: 1043719
Time from start: 0:00:28.852910
File 2/150: fineweb_100BT/sample/100BT/006_00001.parquet
Size: 1G
Created table row count: 2094339
Time from start: 0:01:01.744534
File 3/150: fineweb_100BT/sample/100BT/011_00000.parquet
Size: 1G
```

```
Created table row count: 3133065
Time from start: 0:01:33.096181
File 4/150: fineweb_100BT/sample/100BT/009_00008.parquet
Size: 1G
Created table row count: 4176711
Time from start: 0:02:03.285131
File 5/150: fineweb_100BT/sample/100BT/001_00004.parquet
Size: 1G
Created table row count: 5228342
Time from start: 0:02:33.440667
File 6/150: fineweb_100BT/sample/100BT/014_00002.parquet
Size: 228M
Created table row count: 5346224
Time from start: 0:02:45.836067
File 7/150: fineweb_100BT/sample/100BT/010_00004.parquet
Size: 2G
Created table row count: 6387939
Time from start: 0:03:16.229568
File 8/150: fineweb_100BT/sample/100BT/008_00007.parquet
Size: 1G
Created table row count: 7430532
Time from start: 0:03:47.389115
File 9/150: fineweb_100BT/sample/100BT/002_00002.parquet
Size: 2G
Created table row count: 8484171
Time from start: 0:04:17.673468
File 10/150: fineweb_100BT/sample/100BT/005_00003.parquet
Size: 2G
Created table row count: 9532816
Time from start: 0:04:48.737682
File 11/150: fineweb_100BT/sample/100BT/008_00006.parquet
Size: 1G
Created table row count: 10573420
Time from start: 0:05:19.362057
File 12/150: fineweb_100BT/sample/100BT/004_00006.parquet
Size: 2G
Created table row count: 11623027
Time from start: 0:05:50.208816
File 13/150: fineweb_100BT/sample/100BT/006_00008.parquet
Size: 2G
Created table row count: 12674674
Time from start: 0:06:21.136123
File 14/150: fineweb_100BT/sample/100BT/014_00000.parquet
Size: 289M
Created table row count: 12824334
Time from start: 0:06:35.916579
File 15/150: fineweb_100BT/sample/100BT/002_00000.parquet
Size: 1G
Created table row count: 13876952
Time from start: 0:07:06.777706
File 16/150: fineweb_100BT/sample/100BT/000_00001.parquet
Size: 2G
Created table row count: 14933571
```

```
Time from start: 0:07:38.816585
File 17/150: fineweb_100BT/sample/100BT/008_00008.parquet
Size: 2G
Created table row count: 15976203
Time from start: 0:08:09.023863
File 18/150: fineweb_100BT/sample/100BT/010_00009.parquet
Size: 2G
Created table row count: 17016928
Time from start: 0:08:40.462770
File 19/150: fineweb_100BT/sample/100BT/014_00008.parquet
Size: 92M
Created table row count: 17065178
Time from start: 0:08:49.528764
File 20/150: fineweb_100BT/sample/100BT/011_00008.parquet
Size: 2G
Created table row count: 18107879
Time from start: 0:09:20.749646
File 21/150: fineweb_100BT/sample/100BT/005_00006.parquet
Size: 1G
Created table row count: 19157553
Time from start: 0:09:54.081489
File 22/150: fineweb_100BT/sample/100BT/009_00005.parquet
Size: 1G
Created table row count: 20203222
Time from start: 0:10:26.292212
File 23/150: fineweb_100BT/sample/100BT/000_00000.parquet
Size: 2G
Created table row count: 21261862
Time from start: 0:10:59.173029
File 24/150: fineweb_100BT/sample/100BT/003_00005.parquet
Size: 1G
Created table row count: 22311481
Time from start: 0:11:30.492799
File 25/150: fineweb_100BT/sample/100BT/012_00004.parquet
Size: 2G
Created table row count: 23355169
Time from start: 0:12:01.600806
File 26/150: fineweb_100BT/sample/100BT/003_00002.parquet
Size: 2G
Created table row count: 24407778
Time from start: 0:12:33.296834
File 27/150: fineweb_100BT/sample/100BT/001_00007.parquet
Size: 2G
Created table row count: 25459381
Time from start: 0:13:06.284683
File 28/150: fineweb_100BT/sample/100BT/010_00001.parquet
Size: 2G
Created table row count: 26503114
Time from start: 0:13:39.525053
File 29/150: fineweb_100BT/sample/100BT/005_00008.parquet
Size: 1G
Created table row count: 27551779
Time from start: 0:14:13.082079
```

```
File 30/150: fineweb_100BT/sample/100BT/004_00002.parquet
Size: 2G
Created table row count: 28599409
Time from start: 0:14:46.372963
File 31/150: fineweb_100BT/sample/100BT/011_00007.parquet
Size: 2G
Created table row count: 29639150
Time from start: 0:15:19.568540
File 32/150: fineweb_100BT/sample/100BT/009_00000.parquet
Size: 2G
Created table row count: 30681832
Time from start: 0:15:53.101321
File 33/150: fineweb_100BT/sample/100BT/006_00002.parquet
Size: 2G
Created table row count: 31733500
Time from start: 0:16:26.300927
File 34/150: fineweb_100BT/sample/100BT/006_00007.parquet
Size: 2G
Created table row count: 32782146
Time from start: 0:17:00.225888
File 35/150: fineweb_100BT/sample/100BT/012_00008.parquet
Size: 1G
Created table row count: 33828841
Time from start: 0:17:32.810276
File 36/150: fineweb_100BT/sample/100BT/002_00007.parquet
Size: 1G
Created table row count: 34877448
Time from start: 0:18:08.108476
File 37/150: fineweb_100BT/sample/100BT/005_00007.parquet
Size: 2G
Created table row count: 35930049
Time from start: 0:18:42.200643
File 38/150: fineweb_100BT/sample/100BT/003_00000.parquet
Size: 2G
Created table row count: 36981675
Time from start: 0:19:15.925095
File 39/150: fineweb_100BT/sample/100BT/000_00004.parquet
Size: 1G
Created table row count: 38036322
Time from start: 0:19:50.572792
File 40/150: fineweb_100BT/sample/100BT/007_00008.parquet
Size: 2G
Created table row count: 39083982
Time from start: 0:20:25.948724
File 41/150: fineweb_100BT/sample/100BT/001_00002.parquet
Size: 2G
Created table row count: 40136583
Time from start: 0:21:01.833808
File 42/150: fineweb_100BT/sample/100BT/001_00001.parquet
Size: 2G
Created table row count: 41190213
Time from start: 0:21:37.824188
File 43/150: fineweb_100BT/sample/100BT/010_00005.parquet
```

```
Size: 2G
Created table row count: 42232927
Time from start: 0:22:12.425037
File 44/150: fineweb_100BT/sample/100BT/013_00002.parquet
Size: 2G
Created table row count: 43286646
Time from start: 0:22:47.126034
File 45/150: fineweb_100BT/sample/100BT/014_00009.parquet
Size: 91M
Created table row count: 43334180
Time from start: 0:22:57.208541
File 46/150: fineweb_100BT/sample/100BT/006_00005.parquet
Size: 2G
Created table row count: 44383846
Time from start: 0:23:31.219982
File 47/150: fineweb_100BT/sample/100BT/007_00005.parquet
Size: 2G
Created table row count: 45431510
Time from start: 0:24:08.005149
File 48/150: fineweb_100BT/sample/100BT/006_00004.parquet
Size: 2G
Created table row count: 46481147
Time from start: 0:24:44.025810
File 49/150: fineweb_100BT/sample/100BT/014_00004.parquet
Size: 160M
Created table row count: 46563933
Time from start: 0:24:55.228824
File 50/150: fineweb_100BT/sample/100BT/009_00007.parquet
Size: 1G
Created table row count: 47606598
Time from start: 0:25:30.684821
File 51/150: fineweb_100BT/sample/100BT/012_00002.parquet
Size: 2G
Created table row count: 48647292
Time from start: 0:26:06.884349
File 52/150: fineweb_100BT/sample/100BT/007_00009.parquet
Size: 1G
Created table row count: 49692952
Time from start: 0:26:42.211974
File 53/150: fineweb_100BT/sample/100BT/011_00005.parquet
Size: 1G
Created table row count: 50733683
Time from start: 0:27:17.721789
File 54/150: fineweb_100BT/sample/100BT/008_00004.parquet
Size: 2G
Created table row count: 51774333
Time from start: 0:27:53.212798
File 55/150: fineweb_100BT/sample/100BT/002_00001.parquet
Size: 2G
Created table row count: 52824930
Time from start: 0:28:29.285450
File 56/150: fineweb_100BT/sample/100BT/001_00009.parquet
Size: 1G
```

```
Created table row count: 53878559
Time from start: 0:29:05.236868
File 57/150: fineweb_100BT/sample/100BT/003_00008.parquet
Size: 1G
Created table row count: 54930168
Time from start: 0:29:41.001157
File 58/150: fineweb_100BT/sample/100BT/000_00005.parquet
Size: 2G
Created table row count: 55985752
Time from start: 0:30:17.224921
File 59/150: fineweb_100BT/sample/100BT/003_00004.parquet
Size: 1G
Created table row count: 57038397
Time from start: 0:30:53.156811
File 60/150: fineweb_100BT/sample/100BT/009_00009.parquet
Size: 2G
Created table row count: 58083070
Time from start: 0:31:28.512663
File 61/150: fineweb_100BT/sample/100BT/010_00002.parquet
Size: 1G
Created table row count: 59120789
Time from start: 0:32:05.104801
File 62/150: fineweb_100BT/sample/100BT/006_00003.parquet
Size: 2G
Created table row count: 60169446
Time from start: 0:32:40.673072
File 63/150: fineweb_100BT/sample/100BT/008_00002.parquet
Size: 2G
Created table row count: 61215038
Time from start: 0:33:17.144214
File 64/150: fineweb_100BT/sample/100BT/011_00004.parquet
Size: 1G
Created table row count: 62255748
Time from start: 0:33:54.737086
File 65/150: fineweb_100BT/sample/100BT/012_00006.parquet
Size: 1G
Created table row count: 63300415
Time from start: 0:34:30.432307
File 66/150: fineweb_100BT/sample/100BT/002_00004.parquet
Size: 2G
Created table row count: 64356037
Time from start: 0:35:07.015967
File 67/150: fineweb_100BT/sample/100BT/003_00001.parquet
Size: 1G
Created table row count: 65410650
Time from start: 0:35:43.453072
File 68/150: fineweb_100BT/sample/100BT/001_00005.parquet
Size: 1G
Created table row count: 66464266
Time from start: 0:36:20.613623
File 69/150: fineweb_100BT/sample/100BT/004_00003.parquet
Size: 1G
Created table row count: 67513885
```

```
Time from start: 0:36:57.109028
File 70/150: fineweb_100BT/sample/100BT/013_00008.parquet
Size: 2G
Created table row count: 68567615
Time from start: 0:37:35.467682
File 71/150: fineweb_100BT/sample/100BT/008_00001.parquet
Size: 2G
Created table row count: 69608215
Time from start: 0:38:11.936897
File 72/150: fineweb_100BT/sample/100BT/003_00006.parquet
Size: 1G
Created table row count: 70657819
Time from start: 0:38:49.276731
File 73/150: fineweb_100BT/sample/100BT/013_00001.parquet
Size: 2G
Created table row count: 71710531
Time from start: 0:39:28.260212
File 74/150: fineweb_100BT/sample/100BT/010_00007.parquet
Size: 2G
Created table row count: 72751248
Time from start: 0:40:03.988011
File 75/150: fineweb_100BT/sample/100BT/011_00009.parquet
Size: 1G
Created table row count: 73789934
Time from start: 0:40:39.744685
File 76/150: fineweb_100BT/sample/100BT/005_00005.parquet
Size: 1G
Created table row count: 74844590
Time from start: 0:41:17.292479
File 77/150: fineweb_100BT/sample/100BT/004_00007.parquet
Size: 1G
Created table row count: 75894188
Time from start: 0:41:54.396646
File 78/150: fineweb_100BT/sample/100BT/012_00000.parquet
Size: 2G
Created table row count: 76935865
Time from start: 0:42:30.812184
File 79/150: fineweb_100BT/sample/100BT/007_00003.parquet
Size: 2G
Created table row count: 77983496
Time from start: 0:43:06.936659
File 80/150: fineweb_100BT/sample/100BT/000_00006.parquet
Size: 1G
Created table row count: 79040077
Time from start: 0:43:43.145352
File 81/150: fineweb_100BT/sample/100BT/008_00000.parquet
Size: 2G
Created table row count: 80081683
Time from start: 0:44:19.564529
File 82/150: fineweb_100BT/sample/100BT/007_00006.parquet
Size: 1G
Created table row count: 81127356
Time from start: 0:44:56.718055
```

```
File 83/150: fineweb_100BT/sample/100BT/012_00001.parquet
Size: 2G
Created table row count: 82169051
Time from start: 0:45:33.547747
File 84/150: fineweb_100BT/sample/100BT/007_00000.parquet
Size: 1G
Created table row count: 83216714
Time from start: 0:46:09.748628
File 85/150: fineweb_100BT/sample/100BT/014_00006.parquet
Size: 154M
Created table row count: 83295712
Time from start: 0:46:22.085088
File 86/150: fineweb_100BT/sample/100BT/013_00006.parquet
Size: 1G
Created table row count: 84348411
Time from start: 0:46:57.700685
File 87/150: fineweb_100BT/sample/100BT/013_00000.parquet
Size: 1G
Created table row count: 85400116
Time from start: 0:47:36.255688
File 88/150: fineweb_100BT/sample/100BT/003_00009.parquet
Size: 1G
Created table row count: 86452753
Time from start: 0:48:13.492316
File 89/150: fineweb_100BT/sample/100BT/005_00000.parquet
Size: 1G
Created table row count: 87502424
Time from start: 0:48:51.413627
File 90/150: fineweb_100BT/sample/100BT/004_00004.parquet
Size: 2G
Created table row count: 88552059
Time from start: 0:49:28.871954
File 91/150: fineweb_100BT/sample/100BT/004_00005.parquet
Size: 2G
Created table row count: 89595711
Time from start: 0:50:06.647498
File 92/150: fineweb_100BT/sample/100BT/011_00003.parquet
Size: 2G
Created table row count: 90636384
Time from start: 0:50:43.440582
File 93/150: fineweb_100BT/sample/100BT/000_00008.parquet
Size: 2G
Created table row count: 91690943
Time from start: 0:51:21.197407
File 94/150: fineweb_100BT/sample/100BT/006_00000.parquet
Size: 2G
Created table row count: 92740591
Time from start: 0:51:58.057183
File 95/150: fineweb_100BT/sample/100BT/009_00004.parquet
Size: 2G
Created table row count: 93782266
Time from start: 0:52:36.577706
File 96/150: fineweb_100BT/sample/100BT/001_00000.parquet
```

```
Size: 2G
Created table row count: 94834913
Time from start: 0:53:14.052083
File 97/150: fineweb_100BT/sample/100BT/002_00005.parquet
Size: 1G
Created table row count: 95890524
Time from start: 0:53:52.523632
File 98/150: fineweb_100BT/sample/100BT/003_00007.parquet
Size: 1G
Created table row count: 96939135
Time from start: 0:54:29.820786
File 99/150: fineweb_100BT/sample/100BT/002_00006.parquet
Size: 2G
Created table row count: 97990770
Time from start: 0:55:07.436848
File 100/150: fineweb_100BT/sample/100BT/012_00009.parquet
Size: 1G
Created table row count: 99034470
Time from start: 0:55:47.194622
File 101/150: fineweb_100BT/sample/100BT/007_00001.parquet
Size: 2G
Created table row count: 100083108
Time from start: 0:56:25.127151
File 102/150: fineweb_100BT/sample/100BT/002_00009.parquet
Size: 2G
Created table row count: 101135723
Time from start: 0:57:02.752588
File 103/150: fineweb_100BT/sample/100BT/002_00003.parquet
Size: 1G
Created table row count: 102188381
Time from start: 0:57:39.727832
File 104/150: fineweb_100BT/sample/100BT/004_00001.parquet
Size: 2G
Created table row count: 103236981
Time from start: 0:58:16.268214
File 105/150: fineweb_100BT/sample/100BT/000_00009.parquet
Size: 1G
Created table row count: 104293603
Time from start: 1:12:25.726401
File 106/150: fineweb_100BT/sample/100BT/009_00006.parquet
Size: 2G
Created table row count: 105337243
Time from start: 1:18:07.728719
File 107/150: fineweb_100BT/sample/100BT/005_00004.parquet
Size: 2G
Created table row count: 106391859
Time from start: 1:27:36.670984
File 108/150: fineweb_100BT/sample/100BT/007_00007.parquet
Size: 2G
Created table row count: 107440533
Time from start: 1:32:42.870671
File 109/150: fineweb_100BT/sample/100BT/009_00001.parquet
Size: 1G
```

```
Created table row count: 108483175
Time from start: 1:36:35.532009
File 110/150: fineweb_100BT/sample/100BT/008_00003.parquet
Size: 2G
Created table row count: 109524824
Time from start: 1:40:59.236812
File 111/150: fineweb_100BT/sample/100BT/005_00001.parquet
Size: 1G
Created table row count: 110572500
Time from start: 1:45:23.668655
File 112/150: fineweb_100BT/sample/100BT/007_00002.parquet
Size: 2G
Created table row count: 111619149
Time from start: 1:49:43.723661
File 113/150: fineweb_100BT/sample/100BT/011_00001.parquet
Size: 1G
Created table row count: 112660866
Time from start: 1:54:21.210991
File 114/150: fineweb_100BT/sample/100BT/000_00007.parquet
Size: 1G
Created table row count: 113715430
Time from start: 1:58:52.989134
File 115/150: fineweb_100BT/sample/100BT/014_00001.parquet
Size: 244M
Created table row count: 113841900
Time from start: 2:02:41.656932
File 116/150: fineweb_100BT/sample/100BT/014_00005.parquet
Size: 169M
Created table row count: 113929476
Time from start: 2:06:35.322410
File 117/150: fineweb_100BT/sample/100BT/010_00003.parquet
Size: 1G
Created table row count: 114969219
Time from start: 2:11:41.393656
File 118/150: fineweb_100BT/sample/100BT/011_00006.parquet
Size: 2G
Created table row count: 116008910
Time from start: 2:16:12.055538
File 119/150: fineweb_100BT/sample/100BT/001_00008.parquet
Size: 2G
Created table row count: 117059491
Time from start: 2:21:16.509081
File 120/150: fineweb_100BT/sample/100BT/008_00005.parquet
Size: 2G
Created table row count: 118100096
Time from start: 2:25:43.547424
File 121/150: fineweb_100BT/sample/100BT/012_00007.parquet
Size: 2G
Created table row count: 119139777
Time from start: 2:31:04.039562
File 122/150: fineweb_100BT/sample/100BT/010_00008.parquet
Size: 2G
Created table row count: 120181505
```

```
Time from start: 2:35:50.879853
File 123/150: fineweb_100BT/sample/100BT/005_00002.parquet
Size: 2G
Created table row count: 121235177
Time from start: 2:41:09.794652
File 124/150: fineweb_100BT/sample/100BT/013_00009.parquet
Size: 2G
Created table row count: 122288897
Time from start: 2:46:04.723648
File 125/150: fineweb_100BT/sample/100BT/012_00005.parquet
Size: 1G
Created table row count: 123328570
Time from start: 2:51:28.132650
File 126/150: fineweb_100BT/sample/100BT/012_00003.parquet
Size: 2G
Created table row count: 124374252
Time from start: 2:56:41.151501
File 127/150: fineweb_100BT/sample/100BT/014_00007.parquet
Size: 143M
Created table row count: 124447824
Time from start: 3:03:14.479777
File 128/150: fineweb_100BT/sample/100BT/013_00003.parquet
Size: 2G
Created table row count: 125500577
Time from start: 3:12:44.314742
File 129/150: fineweb_100BT/sample/100BT/004_00000.parquet
Size: 2G
Created table row count: 126550174
Time from start: 3:22:12.152731
File 130/150: fineweb_100BT/sample/100BT/009_00003.parquet
Size: 2G
Created table row count: 127593826
Time from start: 3:31:56.180134
File 131/150: fineweb_100BT/sample/100BT/005_00009.parquet
Size: 1G
Created table row count: 128641459
Time from start: 3:42:47.612067
File 132/150: fineweb_100BT/sample/100BT/000_00003.parquet
Size: 2G
Created table row count: 129696028
Time from start: 3:52:25.733665
File 133/150: fineweb_100BT/sample/100BT/002_00008.parquet
Size: 1G
Created table row count: 130749657
Time from start: 3:59:48.563870
File 134/150: fineweb_100BT/sample/100BT/014_00003.parquet
Size: 214M
Created table row count: 130859092
Time from start: 4:06:40.370777
File 135/150: fineweb_100BT/sample/100BT/003_00003.parquet
Size: 2G
Created table row count: 131908729
Time from start: 4:15:31.843165
```

```
File 136/150: fineweb_100BT/sample/100BT/001_00003.parquet
Size: 1G
Created table row count: 132960371
Time from start: 4:24:42.440355
File 137/150: fineweb_100BT/sample/100BT/001_00006.parquet
Size: 2G
Created table row count: 134014982
Time from start: 4:33:08.637923
File 138/150: fineweb_100BT/sample/100BT/004_00008.parquet
Size: 1G
Created table row count: 135062623
Time from start: 4:42:35.145125
File 139/150: fineweb_100BT/sample/100BT/013_00005.parquet
Size: 2G
Created table row count: 136116356
Time from start: 4:51:22.587171
File 140/150: fineweb_100BT/sample/100BT/000_00002.parquet
Size: 2G
Created table row count: 137175935
Time from start: 5:01:00.468476
File 141/150: fineweb_100BT/sample/100BT/009_00002.parquet
Size: 2G
Created table row count: 138217602
Time from start: 5:10:04.730768
File 142/150: fineweb_100BT/sample/100BT/011_00002.parquet
Size: 2G
Created table row count: 139258309
Time from start: 5:18:23.841829
File 143/150: fineweb_100BT/sample/100BT/010_00000.parquet
Size: 2G
Created table row count: 140299012
Time from start: 5:25:47.866903
File 144/150: fineweb_100BT/sample/100BT/013_00007.parquet
Size: 2G
Created table row count: 141352724
Time from start: 5:34:58.433079
File 145/150: fineweb_100BT/sample/100BT/004_00009.parquet
Size: 2G
Created table row count: 142400324
Time from start: 5:44:22.919270
File 146/150: fineweb_100BT/sample/100BT/007_00004.parquet
Size: 2G
Created table row count: 143446963
Time from start: 5:54:32.355219
File 147/150: fineweb_100BT/sample/100BT/006_00009.parquet
Size: 2G
Created table row count: 144494613
Time from start: 6:03:11.414636
File 148/150: fineweb_100BT/sample/100BT/006_00006.parquet
Size: 1G
Created table row count: 145542270
Time from start: 6:15:49.798877
File 149/150: fineweb_100BT/sample/100BT/013_00004.parquet
```

```
Size: 2G
Created table row count: 146597987
Time from start: 6:22:12.993619
File 150/150: fineweb_100BT/sample/100BT/008_00009.parquet
Size: 1G
Created table row count: 147639585
Time from start: 6:28:43.843504
Uploaded in 6:28:43.843592
```

За 8,5 часов мы загрузили сэмпл среднего размера и получили таблицу размером в **147 млн** строк. Тексты, собранные в этой таблице, представляют собой долю в `100 млрд / 18,5 трлн` — то есть, около **0,5%** от текстов всего дата-сета.

Ниже мы будем использовать и этот сэмпл, и предыдущий меньшего размера.

Очистка и подготовка данных

Для дальнейшей работы нам будут нужны не все колонки исходных таблиц. Мы оставим только те колонки, которые могут понадобиться для нашего анализа:

- `text` — собственно полный текст некоторой веб-страницы.
- `date` — дата, в которую был получен данный текст. Мы преобразуем ее в тип `TIMESTAMP` с помощью функций `strptime` и `regexp_extract`.
- `url` — адрес веб-страницы, на которой размещается данный текст. Может понадобиться для анализа, связанного с географическим распределением.
- `dump` — идентификатор дампа. Может понадобиться для разбиения текстов на подгруппы.
- `id` — идентификатор текста. Сохраним его для того, чтобы иметь возможность при необходимости сгруппировать предложения (на которые мы разобьем тексты) по исходным текстам.

Наш анализ будет заключаться в том, что мы будем вычислять частоту упоминаний в текстах некоторых слов или сочетаний слов в зависимости от периода времени. Чтобы эта частота была более показательной, мы разобьем тексты из колонки `text` на предложения с помощью регулярных выражений и будем вычислять частоту упоминаний не по текстам, а по предложениям. Мы будем подсчитывать количество таких предложений, в которых искомое слово или сочетание слов встретилось хотя бы один раз.

Для разбиения на предложения используем функцию `regexp_split_to_table`.

Создадим таблицу очищенных данных с разбиением текстов на предложения:

```
CREATE OR REPLACE TABLE demo.fineweb_sample_100BT_sent AS
    SELECT
        regexp_split_to_table(text, '(\.\s)|(\?\s)|(\!\s)|(\n)') AS sentence,
        url,
        dump,
        id,
        strptime(regexp_extract(date, '^.{10}'), '%x') AS time_stamp
```

```
    FROM demo.fineweb_sample_100BT
```

```
Done in 14 min. 20 sec.

+--------+
| status |
+--------+
| CREATE |
+--------+
```

Проверим, сколько строк в созданной таблице с предложениями:

```
SELECT count(*) AS "row count"
    FROM demo.fineweb_sample_100BT_sent
```

```
Done in 15 min. 50 sec.

+------------+
| row count  |
+------------+
| 4944294138 |
+------------+
```

Мы получили таблицу очищенных и подготовленных данных длиной **4,9 млрд** строк.

Проверим данные в созданной таблице:

```
SELECT * FROM demo.fineweb_sample_100BT_sent
    LIMIT 5
```

```
Done in 8.3 sec.

+-----------------------------------------------------------------------------------------
-------------------------------------------------------------+
-------------------------------------------------------------------------+
-----------------+-------------------------------------------------+---------------------+
| sentence
| url                                                                          | dump
| id                                                | time_stamp          |
+-----------------------------------------------------------------------------------------
-------------------------------------------------------------+
-------------------------------------------------------------------------+
-----------------+-------------------------------------------------+---------------------+
| - There has been an injunction petition against the appointment of a new EC boss
| https://yen.com.gh/113014-supreme-court-petitioned-injunction-ec-boss.html | CC-MAIN-2019-
09 | <urn:uuid:1f15e5e0-1d05-4b0a-a457-3e007cba4dd3> | 2019-02-17 00:00:00 |
+-----------------------------------------------------------------------------------------
```

```
-----------------------------------------------------------------+
-----------------------------------------------------------------------------+
-----------------+-------------------------------------------------+---------------------+
| - A concerned citizen, Fafali Nyonator, who earlier sued the attorney general, is now
begging the Supreme Court over a replacement for Charlotte Osei     |
https://yen.com.gh/113014-supreme-court-petitioned-injunction-ec-boss.html | CC-MAIN-2019-09
| <urn:uuid:1f15e5e0-1d05-4b0a-a457-3e007cba4dd3> | 2019-02-17 00:00:00 |
+-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------+
-----------------------------------------------------------------------------+
-----------------+-------------------------------------------------+---------------------+
| - Charlotte Osei was sacked by the president on the basis of misbehaviour and incompetence
| https://yen.com.gh/113014-supreme-court-petitioned-injunction-ec-boss.html | CC-MAIN-2019-
09 | <urn:uuid:1f15e5e0-1d05-4b0a-a457-3e007cba4dd3> | 2019-02-17 00:00:00 |
+-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------+
-----------------------------------------------------------------------------+
-----------------+-------------------------------------------------+---------------------+
| A concerned citizen, Fafali Nyonator, has petitioned the Supreme Court to stop Nana Addo
from appointing a new electoral commissioner                     | https://yen.com.gh/113014-
supreme-court-petitioned-injunction-ec-boss.html | CC-MAIN-2019-09 | <urn:uuid:1f15e5e0-1d05-
4b0a-a457-3e007cba4dd3> | 2019-02-17 00:00:00 |
+-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------+
-----------------------------------------------------------------------------+
-----------------+-------------------------------------------------+---------------------+
| Fafali Nyonator, who had earlier prayed the court to reverse the sacking of Charlotte Osei,
has now filed this new injunction to fast-track the situation | https://yen.com.gh/113014-
supreme-court-petitioned-injunction-ec-boss.html | CC-MAIN-2019-09 | <urn:uuid:1f15e5e0-1d05-
4b0a-a457-3e007cba4dd3> | 2019-02-17 00:00:00 |
+-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------+
-----------------------------------------------------------------------------+
-----------------+-------------------------------------------------+---------------------+
```

Аналогичное разбиение на предложения мы сделали и для малого сэмпла (`10BT`), записав данные в таблицу `demo.fineweb_sample_10BT_sent`.

Вот ее размер:

```
SELECT count(*) AS "row count"
    FROM demo.fineweb_sample_10BT_sent
```

```
+-----------+
| row count |
+-----------+
| 497812391 |
+-----------+
```

Малый сэмпл имеет размер **497 млн** строк, то есть примерно в 10 раз меньше сэмпла `100BT`, как и

ожидалось.

Анализ

На основе сформированных выше данных за счет их объема и информации о временнОм периоде можно получить самую разнообразную статистику. Ограничимся здесь только нескольким примерами: построим графики с трендами по разным наборам близких сущностей, таких как виды спорта, разные социальные сети, популярные музыкальные исполнители и прочие общемировые явления.

Виды спорта

Возьмем четыре вида спорта (`basketball`, `yoga`, `baseball`, `soccer`) и вычислим частоту их упоминаний (количество предложений, где они встречаются, деленное на общее количество предложений в данном временном периоде) с сегментацией по кварталам. Для этого используем функцию `date_trunc` с параметром `quarter`.

Чтобы найти ключевое слово в предложении будем использовать функцию `regexp_matches`, описав регулярное выражение так, чтобы нашлись только полные слова, не являющиеся частями других слов. Условия будем задавать с помощью выражения `CASE`.

Для начала сделаем это на малом сэмпле (таблица `demo.fineweb_sample_10BT_sent`).

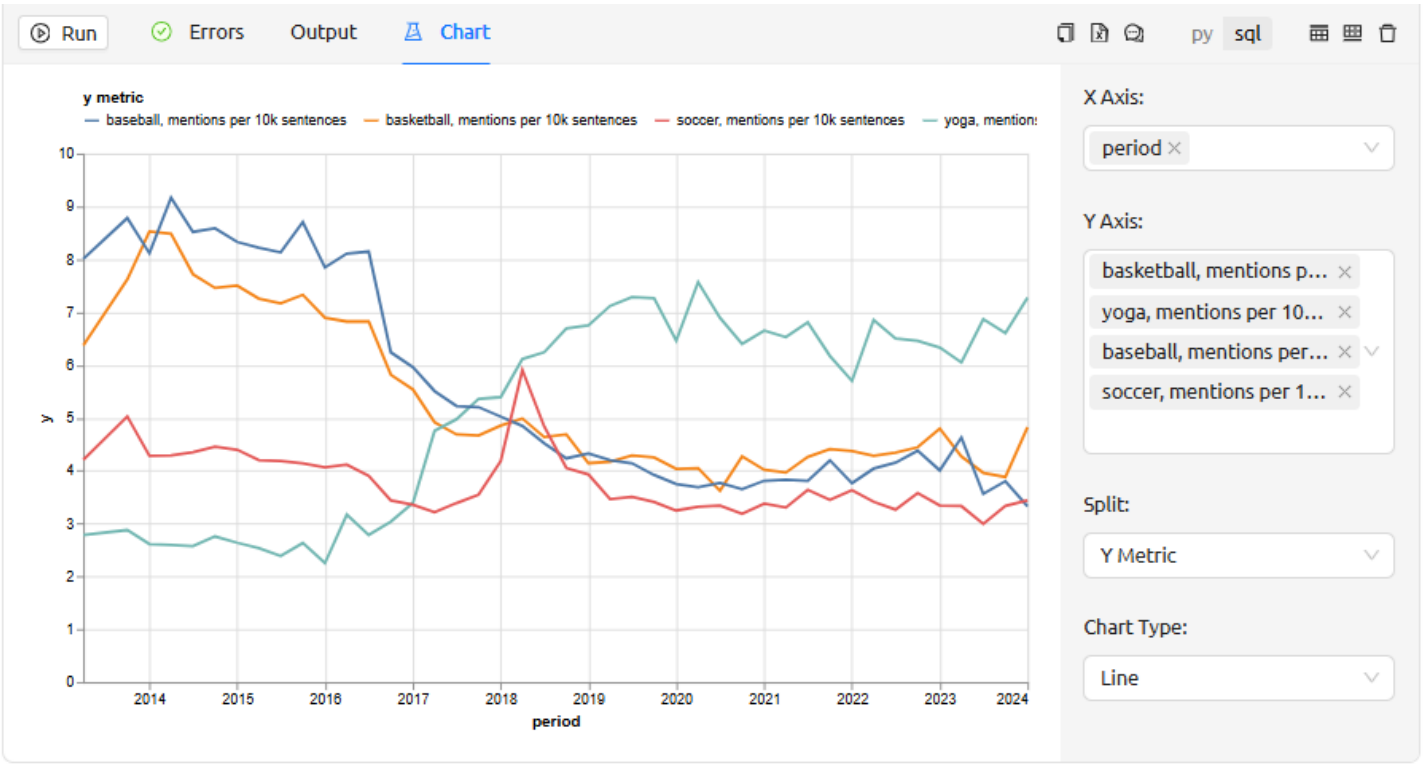
```
SELECT
    date_trunc('quarter', time_stamp) AS period,
    count(*) AS sent_cnt,
    (count(CASE
               WHEN regexp_matches(sentence, '(?:^|[[:punct:]]| )basketball(?:[[:punct:]]|
|$)', 'i') THEN 1
           END)/sent_cnt)*10^4 AS "basketball, mentions per 10k sentences",
    (count(CASE
               WHEN regexp_matches(sentence, '(?:^|[[:punct:]]| )yoga(?:[[:punct:]]| |$)',
'i') THEN 1
           END)/sent_cnt)*10^4 AS "yoga, mentions per 10k sentences",
    (count(CASE
               WHEN regexp_matches(sentence, '(?:^|[[:punct:]]| )baseball(?:[[:punct:]]|
|$)', 'i') THEN 1
           END)/sent_cnt)*10^4 AS "baseball, mentions per 10k sentences",
    (count(CASE
               WHEN regexp_matches(sentence, '(?:^|[[:punct:]]| )soccer(?:[[:punct:]]| |$)',
'i') THEN 1
           END)/sent_cnt)*10^4 AS "soccer, mentions per 10k sentences",
FROM demo.fineweb_sample_10BT_sent
GROUP BY period
ORDER BY period;
```

```
Done in 2 min. 12 sec.

+------------+----------+---------------------------------------
```

```
+---------------------------------+-----------------------------------
+-----------------------------------+
| period     | sent_cnt | basketball, mentions per 10k sentences | yoga, mentions per 10k
sentences | baseball, mentions per 10k sentences | soccer, mentions per 10k sentences |
+------------+----------+----------------------------------------
+---------------------------------+-----------------------------------
+-----------------------------------+
| 2013-04-01 | 4707333  | 6.373035432165092                      | 2.785016483856145
| 8.008781193087465                   | 4.204079040084906                 |
+------------+----------+----------------------------------------
+---------------------------------+-----------------------------------
+-----------------------------------+
| 2013-10-01 | 4627481  | 7.617535328616152                      | 2.878455902898359
| 8.780154905012036                   | 5.026492815421609                 |
+------------+----------+----------------------------------------
+---------------------------------+-----------------------------------
+-----------------------------------+
| 2014-01-01 | 4702545  | 8.52729745276228                       | 2.6113519381526387
| 8.114754882728395                   | 4.280660791124806                 |
+------------+----------+----------------------------------------
+---------------------------------+-----------------------------------
+-----------------------------------+
| 2014-04-01 | 4279243  | 8.48280875846499                       | 2.596253589712012
| 9.167509300126214                   | 4.290478479488077                 |
+------------+----------+----------------------------------------
+---------------------------------+-----------------------------------
+-----------------------------------+
| 2014-07-01 | 13423596 | 7.719243040389475                      | 2.5753158840596813
| 8.517836800213594                   | 4.352038008295244                 |
+------------+----------+----------------------------------------
+---------------------------------+-----------------------------------
+-----------------------------------+
| 2014-10-01 | 13341406 | 7.458734109433443                      | 2.7575804229329353
| 8.58530202888661                    | 4.456801629453447                 |
+------------+----------+----------------------------------------
+---------------------------------+-----------------------------------
+-----------------------------------+
| 2015-01-01 | 11438818 | 7.504271857459398                      | 2.6375102742258862
| 8.327783517492804                   | 4.397307484042495                 |
+------------+----------+----------------------------------------
+---------------------------------+-----------------------------------
+-----------------------------------+
| 2015-04-01 | 10339615 | 7.256556457856506                      | 2.5349106325525663
| 8.218874687307023                   | 4.199382665602153                 |
+------------+----------+----------------------------------------
+---------------------------------+-----------------------------------
+-----------------------------------+
| 2015-07-01 | 11986615 | 7.163823981999923                      | 2.389331767141933
| 8.131570088803219                   | 4.187170439694609                 |
+------------+----------+----------------------------------------
+---------------------------------+-----------------------------------
+-----------------------------------+
| ...        | ...      | ...                                    | ...
```

```
| ...                                  | ...                                  |
+------------+----------+---------------------------------------
+----------------------------------+-------------------------------------
+----------------------------------+

43 rows
```

Теперь построим график на основе полученных результатов:



Мы видим, что частота упоминаний йоги за последние годы выросла и обогнала другие виды спорта, а частота упоминаний баскетбола и бейсбола упала и сравнялась с частотой для футбола (`soccer`). Также мы видим пик упоминаний `soccer`, который приходится на 2018 год, — именно тогда проводился чемпионат мира по футболу.

> Отметим, что порядок абсолютных величин чисел, которые мы реально сравниваем в этом анализе, — `10^4` для каждого периода. Это достаточно большой порядок для такого сравнения. Но для каких-то других наборов слов порядок их упоминаний может быть сильно ниже, и в таком случае сравнение может не быть статистически значимым.

Теперь вычислим те же данные для большего сэмпла — `100BT`.

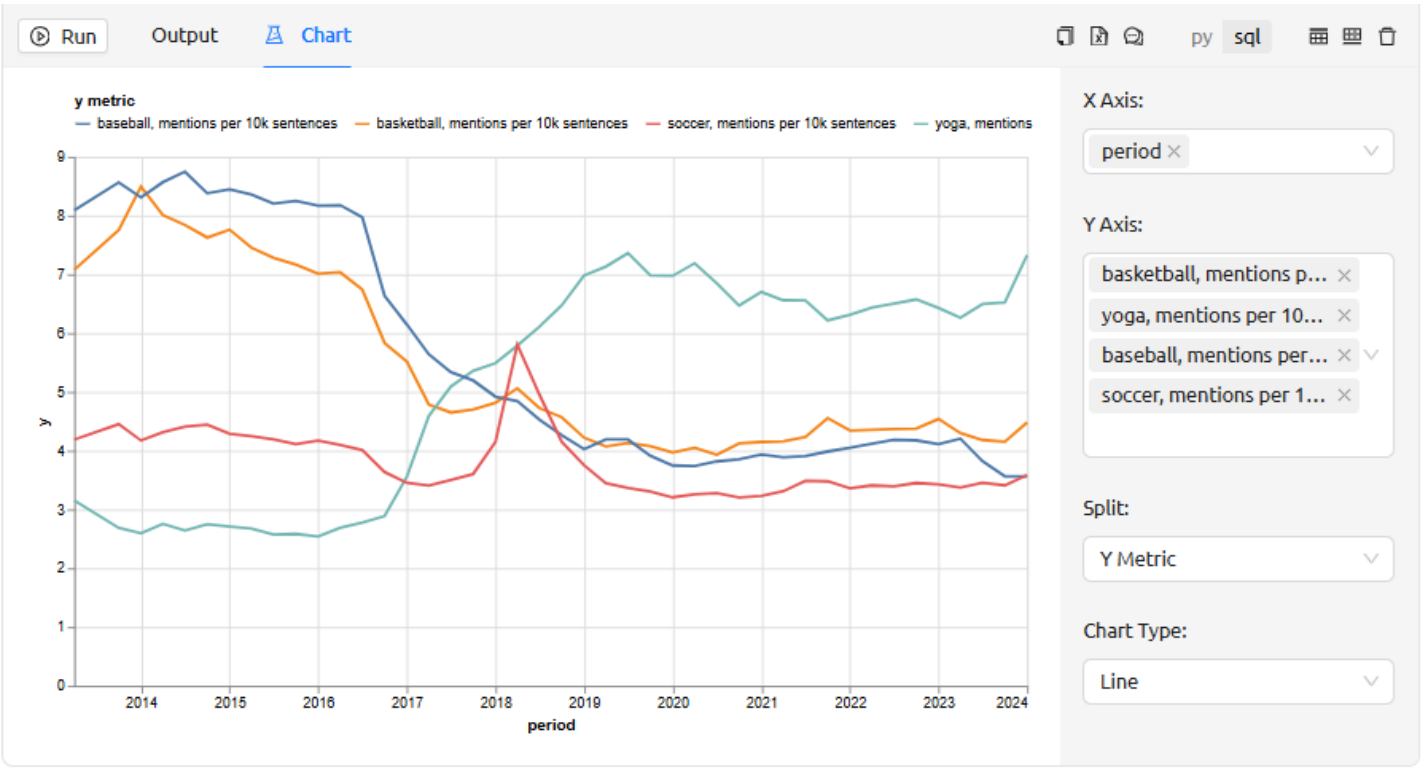
```
SELECT
    date_trunc('quarter', time_stamp) AS period,
    count(*) AS sent_cnt,
    (count(CASE
                WHEN regexp_matches(sentence, '(?:^|[[:punct:]]| )basketball(?:[[:punct:]]|
|$)', 'i') THEN 1
            END)/sent_cnt)*10^4 AS "basketball, mentions per 10k sentences",
    (count(CASE
```

```
               WHEN regexp_matches(sentence, '(?:^|[[:punct:]]| )yoga(?:[[:punct:]]| |$)',
'i') THEN 1
           END)/sent_cnt)*10^4 AS "yoga, mentions per 10k sentences",
    (count(CASE
               WHEN regexp_matches(sentence, '(?:^|[[:punct:]]| )baseball(?:[[:punct:]]|
|$)', 'i') THEN 1
           END)/sent_cnt)*10^4 AS "baseball, mentions per 10k sentences",
    (count(CASE
               WHEN regexp_matches(sentence, '(?:^|[[:punct:]]| )soccer(?:[[:punct:]]| |$)',
'i') THEN 1
           END)/sent_cnt)*10^4 AS "soccer, mentions per 10k sentences",
FROM demo.fineweb_sample_100BT_sent
GROUP BY period
ORDER BY period;
```

```
Done in 21 min 10 sec.

+------------+-----------+---------------------------------------
+-----------------------------------+-------------------------------------
+-----------------------------------+
| period     | sent_cnt  | basketball, mentions per 10k sentences | yoga, mentions per 10k
sentences | baseball, mentions per 10k sentences | soccer, mentions per 10k sentences |
+------------+-----------+---------------------------------------
+-----------------------------------+-------------------------------------
+-----------------------------------+
| 2013-04-01 | 45851328  | 7.0796204637737                       | 3.1523623481527077
| 8.095076330177395                  | 4.189191641297717                 |
+------------+-----------+---------------------------------------
+-----------------------------------+-------------------------------------
+-----------------------------------+
| 2013-10-01 | 46286108  | 7.753514294180881                     | 2.6859030791701044
| 8.564340730484403                  | 4.452091759367628                 |
+------------+-----------+---------------------------------------
+-----------------------------------+-------------------------------------
+-----------------------------------+
| 2014-01-01 | 46377153  | 8.495993706211332                     | 2.593518407652147
| 8.308185713771607                  | 4.173175528907521                 |
+------------+-----------+---------------------------------------
+-----------------------------------+-------------------------------------
+-----------------------------------+
| 2014-04-01 | 42865408  | 8.010188541772424                     | 2.757234924720651
| 8.569147411357894                  | 4.313034883512598                 |
+------------+-----------+---------------------------------------
+-----------------------------------+-------------------------------------
+-----------------------------------+
| 2014-07-01 | 136422305 | 7.843145591184667                     | 2.6447288073603508
| 8.749155792375742                  | 4.409249645796558                 |
+------------+-----------+---------------------------------------
+-----------------------------------+-------------------------------------
+-----------------------------------+
| 2014-10-01 | 132901499 | 7.627528715834876                     | 2.7515867221332093
```

```
| 8.380417138861617                    | 4.4450965899188235                   |
+------------+-----------+--------------------------------------
+-----------------------------------+--------------------------------------
+-----------------------------------+
| 2015-01-01 | 113174872 | 7.762677213365879                    | 2.708993565285411
| 8.447899989672619                    | 4.289291354356469                    |
+------------+-----------+--------------------------------------
+-----------------------------------+--------------------------------------
+-----------------------------------+
| 2015-04-01 | 103388014 | 7.453958831243242                    | 2.6753584801425823
| 8.360059996896739                    | 4.245850007332572                    |
+------------+-----------+--------------------------------------
+-----------------------------------+--------------------------------------
+-----------------------------------+
| 2015-07-01 | 120472087 | 7.281520739322794                    | 2.5748703099996932
| 8.204224103795926                    | 4.192506435121357                    |
+------------+-----------+--------------------------------------
+-----------------------------------+--------------------------------------
+-----------------------------------+
| ...        | ...       | ...                                  | ...
| ...                                  | ...                                  |
+------------+-----------+--------------------------------------
+-----------------------------------+--------------------------------------
+-----------------------------------+

43 rows
```

Построим график:



В данном случае порядок абсолютных чисел, которые сравниваются, — `10^5`. И мы можем отметить, что на этом графике линии более сглаженные, чем на предыдущем. Это важный момент — на большем объеме данных на графике становится меньше случайных флуктуаций, и остаются

только реальные тренды. Это свидетельствует о том, что, во-первых, данные консистентны, а во-вторых о том, что наш анализ раскрывает реальные закономерности, присутствующие в этих текстах.

Далее мы будем производить все вычисления только на большем сэмле — `100BT`, так как очевидно, что результаты анализа на нем получаются более качественными, хотя время расчетов заметно увеличивается.

Iphone против Android

Теперь возьмем два слова: `iphone` и `android` и вычислим частоту упоминаний для этой пары аналогичным образом, только в качестве периода сегментации возьмем год, а не квартал (используем параметр `year` для функции `date_trunc`).

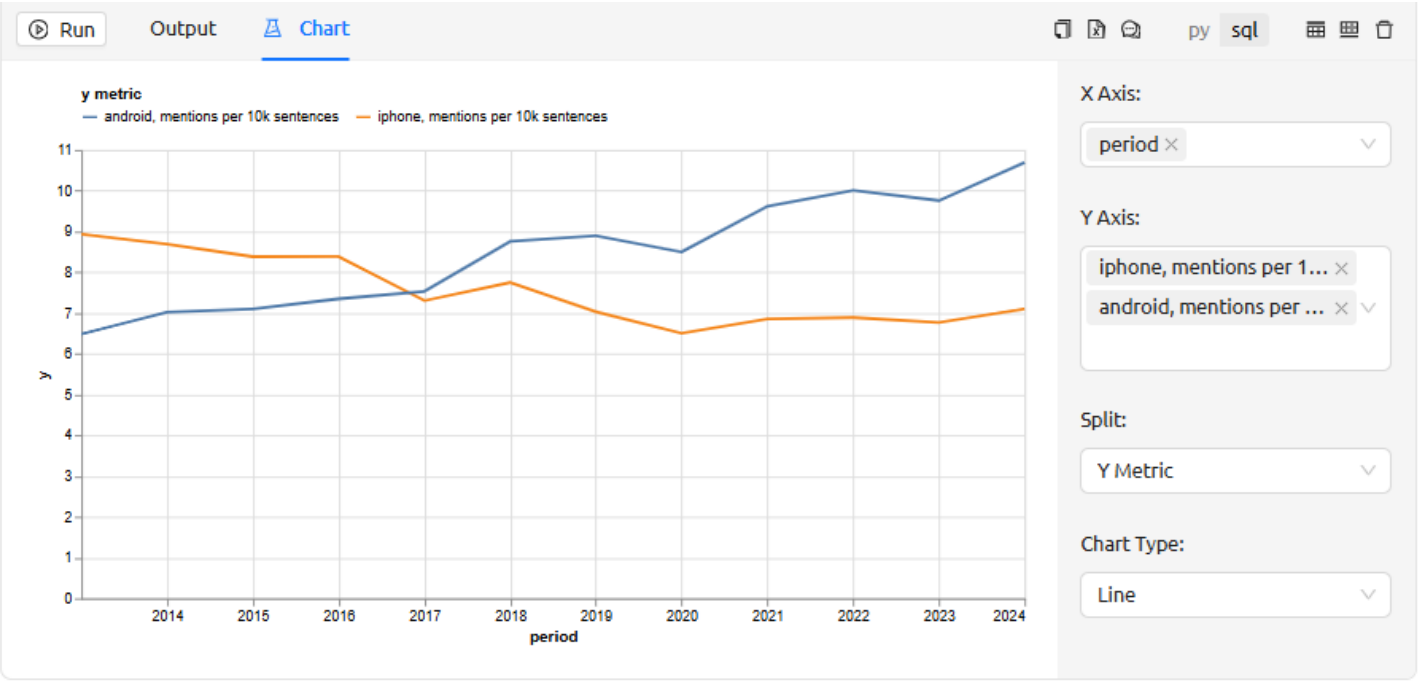
Обратите внимание, что в случае такого "грубого" анализа мы выбираем только такие слова или сочетания слов, которые должны быть однозначны в любом контексте, так как мы никак не проверяем контекст и возможные значения одного и того же слова. Поэтому таким способом не удастся проанализировать тренды для потенциально многозначных слов, таких, как `Apple`, так как мы не сможем отделить контексты со значением `яблоко` от контекстов со значением `Компания Apple`. Для таких случаев следует использовать более "умные" способы текстового анализа. В этом смысле слово `android` тоже не идеально, так как оно может означать человекоподобного робота, а не операционную систему от Google. Однако таких значений, вероятно, довольно мало в общем объеме, поэтому для целей этой демонстрации мы пренебрегаем этим фактом.

```
SELECT
    date_trunc('year', time_stamp) AS period,
    count(*) AS sent_cnt,
    (count(CASE
            WHEN regexp_matches(sentence, '(?:^|[[:punct:]]| )iphone(?:[[:punct:]]| |$)',
'i') THEN 1
        END)/sent_cnt)*10^4 AS "iphone, mentions per 10k sentences",
    (count(CASE
            WHEN regexp_matches(sentence, '(?:^|[[:punct:]]| )android(?:[[:punct:]]| |$)',
'i') THEN 1
        END)/sent_cnt)*10^4 AS "android, mentions per 10k sentences",
FROM demo.fineweb_sample_100BT_sent
GROUP BY period
ORDER BY period;
```

```
Done in 15 min.

+------------+-----------+-----------------------------------
+-------------------------------------+
| period     | sent_cnt  | iphone, mentions per 10k sentences | android, mentions per 10k
sentences |
+------------+-----------+-----------------------------------
+-------------------------------------+
```

```
| 2013-01-01 | 92137436  | 8.932308470142363                 | 6.485203256578575
|
+------------+-----------+-----------------------------------
+-----------------------------------+
| 2014-01-01 | 358566365 | 8.693676552735225                 | 7.024139032114738
|
+------------+-----------+-----------------------------------
+-----------------------------------+
| 2015-01-01 | 412559722 | 8.383974041944889                 | 7.096621031754524
|
+------------+-----------+-----------------------------------
+-----------------------------------+
| 2016-01-01 | 378850555 | 8.389930958395983                 | 7.349467918820919
|
+------------+-----------+-----------------------------------
+-----------------------------------+
| 2017-01-01 | 695789958 | 7.304862540140311                 | 7.5303185102881285
|
+------------+-----------+-----------------------------------
+-----------------------------------+
| 2018-01-01 | 632081943 | 7.747349934975124                 | 8.756791838934086
|
+------------+-----------+-----------------------------------
+-----------------------------------+
| 2019-01-01 | 580360066 | 7.030893817563251                 | 8.898475795541728
|
+------------+-----------+-----------------------------------
+-----------------------------------+
| 2020-01-01 | 457671733 | 6.504247007975037                 | 8.49984764079804
|
+------------+-----------+-----------------------------------
+-----------------------------------+
| 2021-01-01 | 524184633 | 6.855828602667183                 | 9.621171019715872
|
+------------+-----------+-----------------------------------
+-----------------------------------+
| 2022-01-01 | 390360895 | 6.890623611261062                 | 10.010813198898932
|
+------------+-----------+-----------------------------------
+-----------------------------------+
| 2023-01-01 | 373505149 | 6.769973604835097                 | 9.760695427521402
|
+------------+-----------+-----------------------------------
+-----------------------------------+
| 2024-01-01 | 48225683  | 7.098292418170625                 | 10.690776530837313
|
+------------+-----------+-----------------------------------
+-----------------------------------+
```

Построим график:



Мы видим, что частота упоминаний этих слов в первые годы нашего периода наблюдений имела одно соотношение, а в последние годы — соотношение изменилось, и `android` стал упоминаться чаще, чем `iphone`. Причем мы имеем явный момент времени, когда это соотношение изменилось — 2017 год.

В данном случае абсолютный порядок сравниваемых чисел (количество предложений с найденным ключевым словом в каждом временном периоде) — `10^6`. А значит, это еще более достоверные результаты, чем в предыдущем примере.

Мы сознательно не занимаемся здесь интерпретацией полученных результатов — то есть не делаем утверждения в духе *Android выиграл у Iphone* и т.п. Так как этих данных может быть недостаточно для таких выводов. Все, что мы точно можем сказать, — что в данном дата-сете частота упоминаний данных последовательностей символов меняется обнаруженным нами образом.

Instagram, Tiktok и Facebook

Сраним аналогичным образом частоту упоминаний названий трех социальных сетей — `Instagram`, `Tiktok` и `Facebook` [1]. Сегментируем временные периоды так же, как и в предыдущем примере, — по году.

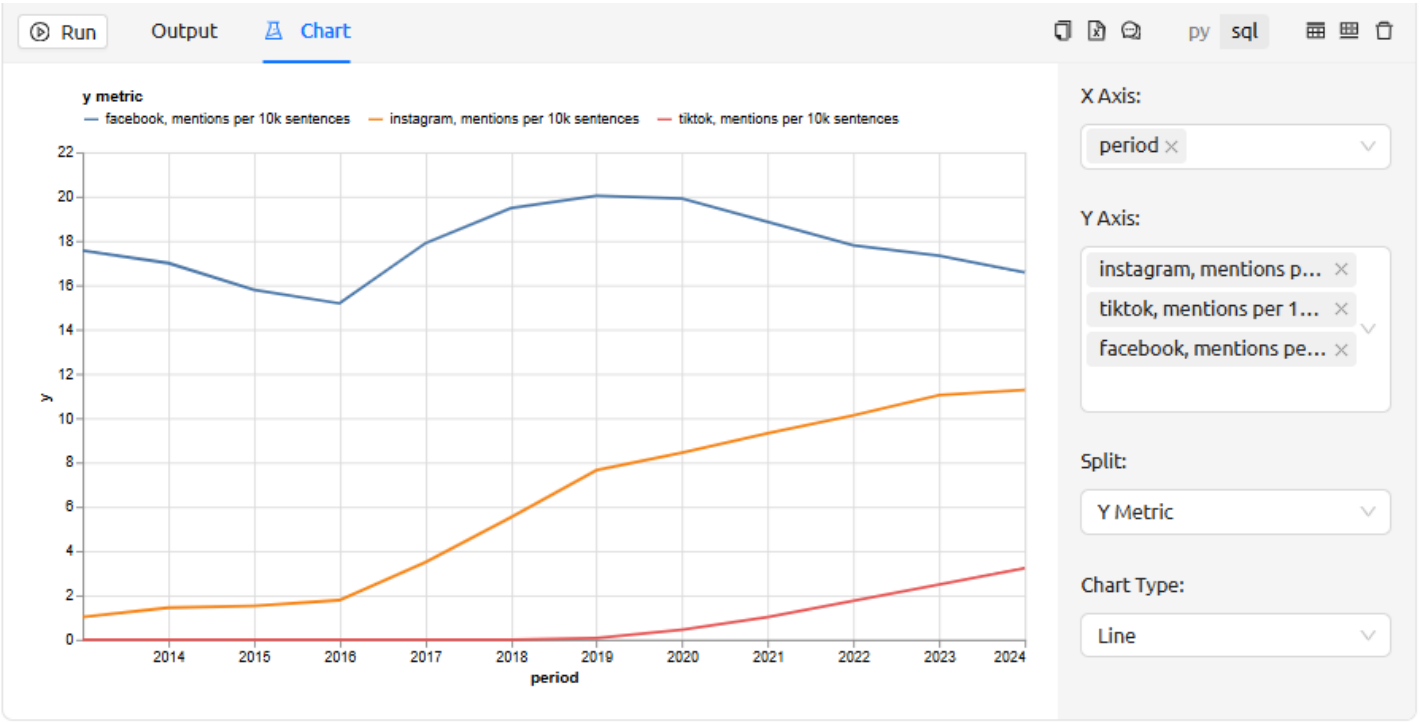
```
SELECT
    date_trunc('year', time_stamp) AS period,
    count(*) AS sent_cnt,
    (count(CASE
                WHEN regexp_matches(sentence, '(?:^|[[:punct:]]| )instagram(?:[[:punct:]]|
|$)', 'i') THEN 1
            END)/sent_cnt)*10^4 AS "instagram, mentions per 10k sentences",
    (count(CASE
                WHEN regexp_matches(sentence, '(?:^|[[:punct:]]| )tiktok(?:[[:punct:]]| |$)',
'i') THEN 1
            END)/sent_cnt)*10^4 AS "tiktok, mentions per 10k sentences",
```

```
    (count(CASE
                WHEN regexp_matches(sentence, '(?:^|[[:punct:]]| )facebook(?:[[:punct:]]|
|$)', 'i') THEN 1
            END)/sent_cnt)*10^4 AS "facebook, mentions per 10k sentences",
FROM demo.fineweb_sample_100BT_sent
GROUP BY period
ORDER BY period;
```

```
Done in 18 min.

+------------+-----------+---------------------------------------
+-----------------------------------+-----------------------------------+
| period     | sent_cnt  | instagram, mentions per 10k sentences | tiktok, mentions per 10k
sentences | facebook, mentions per 10k sentences |
+------------+-----------+---------------------------------------
+-----------------------------------+-----------------------------------+
| 2013-01-01 | 92137436  | 1.0193467940653351                    | 0.0015194692415795O3
| 17.571359376659885                |
+------------+-----------+---------------------------------------
+-----------------------------------+-----------------------------------+
| 2014-01-01 | 358566365 | 1.4433869166730124                    | 0.0020916630035837298
| 17.000172339087076                |
+------------+-----------+---------------------------------------
+-----------------------------------+-----------------------------------+
| 2015-01-01 | 412559722 | 1.5202647436338927                    | 0.001793679703904784
| 15.789156460600873                |
+------------+-----------+---------------------------------------
+-----------------------------------+-----------------------------------+
| 2016-01-01 | 378850555 | 1.7838960272870659                    | 0.001821298638456528
| 15.182239867643853                |
+------------+-----------+---------------------------------------
+-----------------------------------+-----------------------------------+
| 2017-01-01 | 695789958 | 3.4984264604750157                    | 0.0018827520934126501
| 17.897743215201732                |
+------------+-----------+---------------------------------------
+-----------------------------------+-----------------------------------+
| 2018-01-01 | 632081943 | 5.519062897830637                     | 0.0023889307655795508
| 19.484483200938396                |
+------------+-----------+---------------------------------------
+-----------------------------------+-----------------------------------+
| 2019-01-01 | 580360066 | 7.6474076353833755                    | 0.07197256056553002
| 20.044349502158887                |
+------------+-----------+---------------------------------------
+-----------------------------------+-----------------------------------+
| 2020-01-01 | 457671733 | 8.440044952481259                     | 0.45615664011305673
| 19.919429894089614                |
+------------+-----------+---------------------------------------
+-----------------------------------+-----------------------------------+
| 2021-01-01 | 524184633 | 9.314599651760489                     | 1.0264513038481233
| 18.85459698319695                 |
+------------+-----------+---------------------------------------
```

```
+-----------------------------------+-------------------------------------+
| 2022-01-01 | 390360895 | 10.123042678237534                    | 1.7644441562211297
| 17.806343025215167                    |
+------------+-----------+---------------------------------------
+-----------------------------------+-------------------------------------+
| 2023-01-01 | 373505149 | 11.046086007237346                    | 2.491960291556784
| 17.33796714004604                     |
+------------+-----------+---------------------------------------
+-----------------------------------+-------------------------------------+
| 2024-01-01 | 48225683  | 11.278430209065158                    | 3.2198610852229916
| 16.58514613468512                     |
+------------+-----------+---------------------------------------
+-----------------------------------+-------------------------------------+
```

Построим график на основе этих данных:



Telegram, Whatsapp и Snapchat

Сравним анаолгичным образом частоту упоминаний трех мессенджеров — `Telegram`, `Whatsapp` и `Snapchat`.

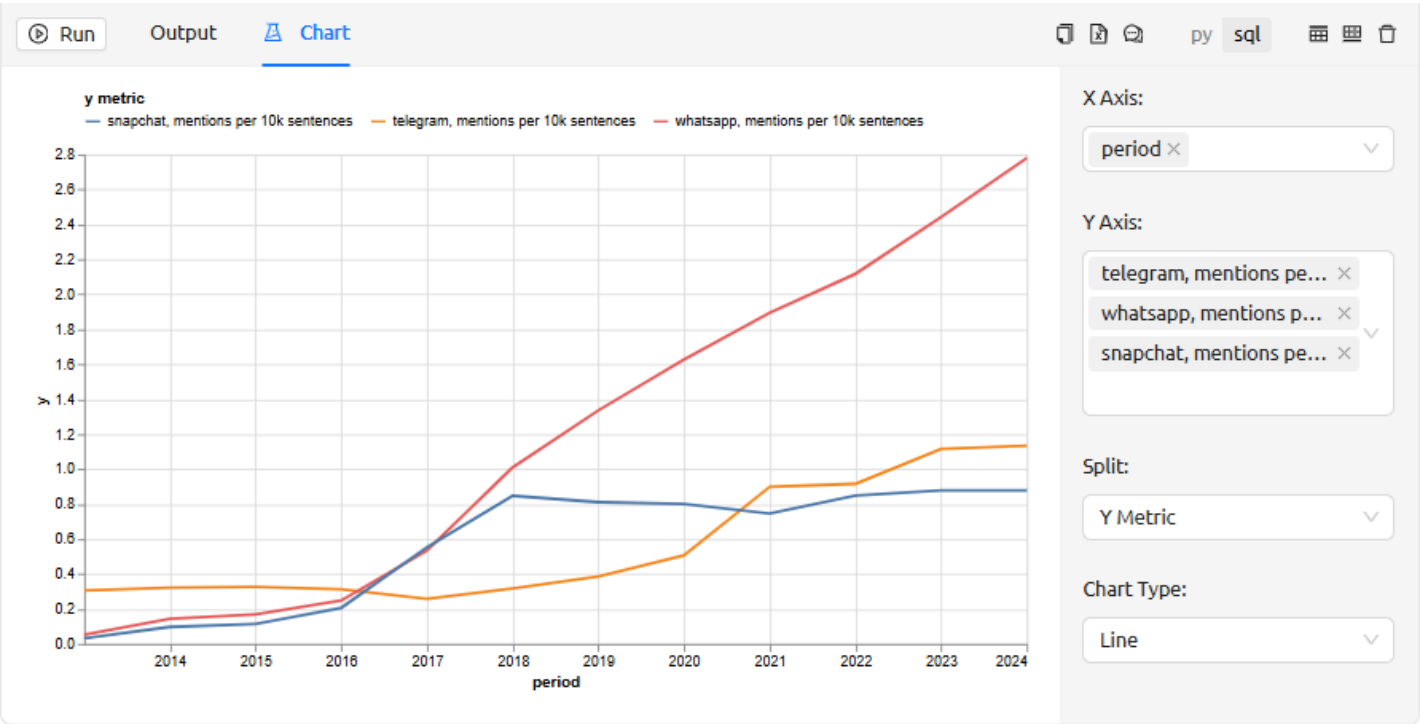
```
SELECT
    date_trunc('year', time_stamp) AS period,
    count(*) AS sent_cnt,
    (count(CASE
               WHEN regexp_matches(sentence, '(?:^|[[:punct:]]| )telegram(?:[[:punct:]]|
|$)', 'i') THEN 1
           END)/sent_cnt)*10^4 AS "Telegram, mentions per 10k sentences",
    (count(CASE
               WHEN regexp_matches(sentence, '(?:^|[[:punct:]]| )whatsapp(?:[[:punct:]]|
|$)', 'i') THEN 1
           END)/sent_cnt)*10^4 AS "Whatsapp, mentions per 10k sentences",
```

```
    (count(CASE
                WHEN regexp_matches(sentence, '(?:^|[[:punct:]]| )snapchat(?:[[:punct:]]|
|$)', 'i') THEN 1
            END)/sent_cnt)*10^4 AS "Snapchat, mentions per 10k sentences",
FROM demo.fineweb_sample_100BT_sent
GROUP BY period
ORDER BY period;
```

```
Done in 17 min.

+------------+-----------+-------------------------------------
+-------------------------------------+-------------------------------------+
| period     | sent_cnt  | telegram, mentions per 10k sentences | whatsapp, mentions per 10k
sentences | snapchat, mentions per 10k sentences |
+------------+-----------+-------------------------------------
+-------------------------------------+-------------------------------------+
| 2013-01-01 | 92137436  | 0.3060645186610142                   | 0.051770487730958784
| 0.03169178703865821                  |
+------------+-----------+-------------------------------------
+-------------------------------------+-------------------------------------+
| 2014-01-01 | 358566365 | 0.3217256587912254                   | 0.1441016365268951
| 0.09722049640657177                  |
+------------+-----------+-------------------------------------
+-------------------------------------+-------------------------------------+
| 2015-01-01 | 412559722 | 0.32681328983443514                  | 0.16872708674163786
| 0.11406833360237721                  |
+------------+-----------+-------------------------------------
+-------------------------------------+-------------------------------------+
| 2016-01-01 | 378850555 | 0.3132897614469642                   | 0.2481189449491502
| 0.20617628499976726                  |
+------------+-----------+-------------------------------------
+-------------------------------------+-------------------------------------+
| 2017-01-01 | 695789958 | 0.2574052671222944                   | 0.5349459211367348
| 0.5540177686784034                   |
+------------+-----------+-------------------------------------
+-------------------------------------+-------------------------------------+
| 2018-01-01 | 632081943 | 0.31631974653640754                  | 1.0093469795576806
| 0.8475799790407871                   |
+------------+-----------+-------------------------------------
+-------------------------------------+-------------------------------------+
| 2019-01-01 | 580360066 | 0.3853814435261299                   | 1.334981583657067
| 0.8109276078275173                   |
+------------+-----------+-------------------------------------
+-------------------------------------+-------------------------------------+
| 2020-01-01 | 457671733 | 0.506236202269455                    | 1.626689931492885
| 0.8016225900497115                   |
+------------+-----------+-------------------------------------
+-------------------------------------+-------------------------------------+
| 2021-01-01 | 524184633 | 0.8994922214745659                   | 1.8942371399124935
| 0.7451954433772956                   |
+------------+-----------+-------------------------------------
```

```
+------------------------------------+------------------------------------+
| 2022-01-01 | 390360895 | 0.9157935760957818                  | 2.116707924855024
| 0.8491885438473543                 |
+------------+-----------+------------------------------------
+------------------------------------+------------------------------------+
| 2023-01-01 | 373505149 | 1.1152724430045273                  | 2.4411176189702273
| 0.8787027458087331                 |
+------------+-----------+------------------------------------
+------------------------------------+------------------------------------+
| 2024-01-01 | 48225683  | 1.1342503951680685                  | 2.7792245057472797
| 0.8791995750480092                 |
+------------+-----------+------------------------------------
+------------------------------------+------------------------------------+
```

Построим график:



Музыкальные исполнители

Теперь сравним частоту упоминаний музыкальных исполнителей Taylor Swift, Billie Eilish и Lady Gaga. С той же сегментацией — по году.

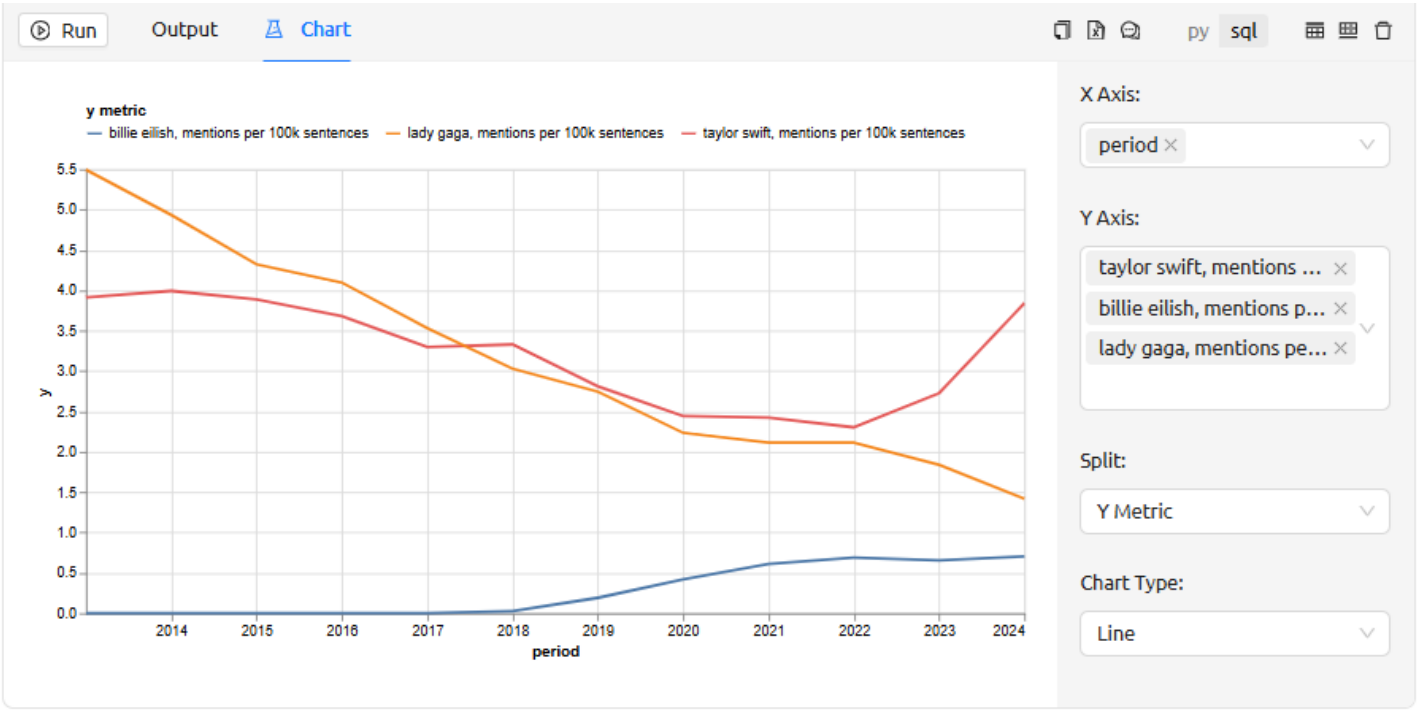
```
SELECT
    date_trunc('year', time_stamp) AS period,
    count(*) AS sent_cnt,
    (count(CASE
                WHEN regexp_matches(sentence, '(?:^|[[:punct:]]| )taylor swift(?:[[:punct:]]|
|$)', 'i') THEN 1
            END)/sent_cnt)*10^5 AS "Taylor Swift, mentions per 100k sentences",
    (count(CASE
                WHEN regexp_matches(sentence, '(?:^|[[:punct:]]| )billie
eilish(?:[[:punct:]]| |$)', 'i') THEN 1
            END)/sent_cnt)*10^5 AS "Billie Eilish, mentions per 100k sentences",
```

```
    (count(CASE
                WHEN regexp_matches(sentence, '(?:^|[[:punct:]]| )lady gaga(?:[[:punct:]]|
|$)', 'i') THEN 1
            END)/sent_cnt)*10^5 AS "Lady Gaga, mentions per 100k sentences",
FROM demo.fineweb_sample_100BT_sent
GROUP BY period
ORDER BY period;
```

```
Done in 17 min.

+------------+-----------+-----------------------------------------
+-----------------------------------------+---------------------------------------+
| period     | sent_cnt  | taylor swift, mentions per 100k sentences | billie eilish,
mentions per 100k sentences | lady gaga, mentions per 100k sentences |
+------------+-----------+-----------------------------------------
+-----------------------------------------+---------------------------------------+
| 2013-01-01 | 92137436  | 3.9126332970672206                        | 0
| 5.4928813083099035                      |
+------------+-----------+-----------------------------------------
+-----------------------------------------+---------------------------------------+
| 2014-01-01 | 358566365 | 3.9928452296411012                        | 0
| 4.933256916052346                       |
+------------+-----------+-----------------------------------------
+-----------------------------------------+---------------------------------------+
| 2015-01-01 | 412559722 | 3.8854980612964445                        | 0
| 4.321798529813824                       |
+------------+-----------+-----------------------------------------
+-----------------------------------------+---------------------------------------+
| 2016-01-01 | 378850555 | 3.6819267692507403                        | 0.0007918689732419686
| 4.09607424225629                        |
+------------+-----------+-----------------------------------------
+-----------------------------------------+---------------------------------------+
| 2017-01-01 | 695789958 | 3.296253378810621                         | 0.003018152210814172
| 3.531956694321823                       |
+------------+-----------+-----------------------------------------
+-----------------------------------------+---------------------------------------+
| 2018-01-01 | 632081943 | 3.329473374941831                         | 0.026578832358765864
| 3.0299868888993085                      |
+------------+-----------+-----------------------------------------
+-----------------------------------------+---------------------------------------+
| 2019-01-01 | 580360066 | 2.812391988390187                         | 0.19074365464697562
| 2.7451923268614418                      |
+------------+-----------+-----------------------------------------
+-----------------------------------------+---------------------------------------+
| 2020-01-01 | 457671733 | 2.445202356423441                         | 0.417329684636652
| 2.2345710391513296                      |
+------------+-----------+-----------------------------------------
+-----------------------------------------+---------------------------------------+
| 2021-01-01 | 524184633 | 2.424718162235786                         | 0.609327286402919
| 2.1156667521003043                      |
+------------+-----------+-----------------------------------------
```

```
+-------------------------------------------+---------------------------------------+
| 2022-01-01 | 390360895 | 2.3032532497908123                    | 0.6873126981635801
| 2.1136850810837498                    |
+------------+-----------+---------------------------------------
+-------------------------------------------+---------------------------------------+
| 2023-01-01 | 373505149 | 2.72472816700045                      | 0.6548771835003538
| 1.8395998069627684                    |
+------------+-----------+---------------------------------------
+-------------------------------------------+---------------------------------------+
| 2024-01-01 | 48225683  | 3.83820380522138                      | 0.7029449432577244
| 1.4183313899359393                    |
+------------+-----------+---------------------------------------
+-------------------------------------------+---------------------------------------+
```

Построим график на основе этих данных:



Обратите внимание, что в данном случае абсолютные цифры на порядок меньше, чем в предыдущих примерах, поэтому в запросе мы умножаем результаты на `10^5` и используем подписи для графика `mentions per 100k sentences` (*упоминаний на 100 тысяч предложений*).

Мы показали, как с помощью Tengri можно анализировать текстовые корпуса очень большого объема таким простым инструментом, как регулярные выражения.

[1] Instagram и Facebook принадлежат компании Meta, признанной экстремистской и запрещенной на территории РФ.

Алфавитный указатель

Общий список выражений SQL, функций и типов данных со сквозным алфавитным порядком.

`abs`	Функция	Вычисляет модуль числа.
`add`	Функция	Складывает числа.
`ALTER ROLE`	Выражение SQL	Изменение атрибутов роли.
`ALTER SCHEMA`	Выражение SQL	Изменение атрибутов схемы.
`ALTER USER`	Выражение SQL	Изменение атрибутов пользователя.
`ALTER WORKER POOL`	Выражение SQL	Изменение атрибутов вычислительного пула.
`ALTER WORKER POOL SET DEFAULT`	Выражение SQL	Установка атрибутов вычислительного пула по умолчанию.
`any_value`	Функция	Возвращает первое значение из `argument`, отличное от `NULL`.
`array_agg`	Функция	Возвращает список, содержащий все значения столбца.
`AS`	Выражение SQL	Элемент запроса `SELECT`.
`avg`	Функция	Вычисляет среднее значение всех непустых значений в `argument`.
`BIGINT`	Тип данных	Целые числа.
`BIGINT[]`	Тип данных	Массивы целых чисел.
`BLOB`	Тип данных	Двоичные объекты.
`BOOL`	Тип данных	Булевы значения.
`BOOL[]`	Тип данных	Массивы булевых значений.
`BOOLEAN`	Тип данных	Псевдоним типа данных `BOOL`.
`BOOLEAN[]`	Тип данных	Псевдоним типа данных `BOOL[]`.
`BPCHAR`	Тип данных	Псевдоним типа данных `VARCHAR`.
`BPCHAR[]`	Тип данных	Псевдоним типа данных `VARCHAR[]`.
`BYTEA`	Тип данных	Псевдоним типа данных `BLOB`.
`CASE`	Выражение SQL	Элемент запроса `SELECT`.
`cast`	Функция	Приводит тип данных к указанному.
`ceil`	Функция	Округляет число в бОльшую сторону.
`ceiling`	Функция	Псевдоним функции `ceil`.
`CHAR`	Тип данных	Псевдоним типа данных `VARCHAR`.
`CHAR[]`	Тип данных	Псевдоним типа данных `VARCHAR[]`.
`char_length`	Функция	Псевдоним функции `length`.
`character_length`	Функция	Псевдоним функции `length`.
`chr`	Функция	Возвращает символ, соответствующий значению кода ASCII или кода Unicode, заданному в `argument`.

coalesce	Функция	Возвращает первое значение, отличное от NULL, из списка значений аргументов.
concat	Функция	Конкатенирует несколько строк, массивов или двоичных значений.
contains	Функция	Возвращает true, если в указанной строке string содержится искомая подстрока search_string.
cos	Функция	Вычисляет косинус угла, заданного в радианах.
count	Функция	Вычисляет количество строк в группе.
count(argument)	Функция	Вычисляет количество непустых значений в argument.
count_if	Функция	Возвращает количество записей, удовлетворяющих условию, или NULL, если ни одна запись не удовлетворяет условию.
CREATE ROLE	Выражение SQL	Создание новой роли.
CREATE SCHEMA	Выражение SQL	Создание новой схемы.
CREATE TABLE	Выражение SQL	Создание новой таблицы.
CREATE USER	Выражение SQL	Создание нового пользователя.
CREATE VIEW	Выражение SQL	Создание нового представления.
CREATE WORKER POOL	Выражение SQL	Создание вычислительного пула.
cume_dist	Функция	Кумулятивное распределение.
current_time	Функция	Возвращает текущее время в виде значения типа TIME.
DATE	Тип данных	Даты.
date_diff	Функция	Возвращает количество единиц времени между двумя моментами времени в виде значения типа BIGINT.
date_part	Функция	Псевдоним функции datepart.
date_trunc	Функция	Сокращает момент времени до указанной точности.
datepart	Функция	Возвращает указанную часть от значения даты или времени в виде значения типа BIGINT.
DECIMAL	Тип данных	Псевдоним типа данных NUMERIC.
DELETE	Выражение SQL	Элемент запроса SELECT.
dense_rank	Функция	Ранг текущей строки внутри группы без пропусков.
DESCRIBE TABLE	Выражение SQL	Вывод информации о таблице.
DESCRIBE USER	Выражение SQL	Вывод информации о пользователе.
divide	Функция	Возвращает результат деления в виде целого числа.
DOUBLE	Тип данных	Вещественные числа с переменной точностью.
DROP ROLE	Выражение SQL	Сброс роли.
DROP SCHEMA	Выражение SQL	Удаление схемы.
DROP TABLE	Выражение SQL	Удаление таблицы.
DROP USER	Выражение SQL	Сброс пользователя.

`DROP VIEW`	Выражение SQL	Удаление представления.
`DROP WORKER POOL`	Выражение SQL	Сброс вычислительного пула.
`even`	Функция	Округляет до ближайшего четного числа в сторону от нуля.
`exp`	Функция	Вычисляет экспоненту числа.
`first_value`	Функция	Возвращает значение, вычисленное по указанному выражению для первой строки группы.
`FLOAT`	Тип данных	Псевдоним типа данных `DOUBLE`.
`floor`	Функция	Округляет число в меньшую сторону.
`fmod`	Функция	Возвращает остаток от деления первого аргумента на второй.
`FROM`	Выражение SQL	Элемент запроса `SELECT`.
`from_json`	Функция	Псевдоним функции `json_transform`.
`from_json_strict`	Функция	Псевдоним функции `json_transform_strict`.
`gcd`	Функция	Вычисляет наибольший общий делитель двух чисел.
`generate_series`	Функция	Создает список значений в диапазоне между `start` и `stop`.
`GEOMETRY`	Тип данных	Геопространственные данные.
`get_current_time`	Функция	Псевдоним функции `current_time`.
`GRANT`	Выражение SQL	Предоставление ролям атрибутов.
`GRANT ROLE`	Выражение SQL	Назначение пользователям ролей.
`greatest`	Функция	Возвращает наибольшее число из указанных в аргументах.
`greatest_common_divisor`	Функция	Псевдоним функции `gcd`.
`GROUP BY`	Выражение SQL	Элемент запроса `SELECT`.
`hash`	Функция	Возвращает хеш данных из `argument` в виде числа.
`HAVING`	Выражение SQL	Элемент запроса `SELECT`.
`ILIKE`	Выражение SQL	Элемент запроса `SELECT`.
`INSERT`	Выражение SQL	Добавление данных в таблицу.
`INTEGER`	Тип данных	Псевдоним типа данных `BIGINT`.
`INTEGER[]`	Тип данных	Псевдоним типа данных `BIGINT[]`.
`isfinite`	Функция	Проверяет, является ли число конечным.
`isinf`	Функция	Проверяет, является ли число бесконечным.
`isnan`	Функция	Проверяет, имеет ли аргумент значение `NaN` (*Not a Number*).
`JOIN`	Выражение SQL	Элемент запроса `SELECT`.
`JSON`	Тип данных	Данные в JSON.
`json`	Функция	Сокращает запись структуры JSON (убирает пробелы и переносы строк).

json_array_length	Функция	Возвращает количество элементов в массиве по указанному пути в JSON или 0, если по указанному пути не массив.
json_contains	Функция	Проверяет, есть ли в структуре JSON подструктура JSON, заданная во втором аргументе.
json_exists	Функция	Проверяет, есть ли в структуре JSON указанный путь.
json_extract	Функция	Извлекает данные из структуры JSON по указанному пути. Возвращает данные в виде JSON.
json_extract_path	Функция	Псевдоним функции json_extract.
json_extract_string	Функция	Извлекает данные из структуры JSON по указанному пути. Возвращает данные в виде VARCHAR.
json_extract_string_path	Функция	Псевдоним функции json_extract_string.
json_group_array	Функция	Возвращает список JSON, содержащий все значения столбца.
json_group_object	Функция	Возвращает структуру JSON, содержащую все пары key-value из столбцов, указанных в аргументах.
json_keys	Функция	Возвращает все ключи из указанной структуры JSON в виде VARCHAR[].
json_transform	Функция	Трансформирует структуру JSON в соответствии с указанной структурой.
json_transform_strict	Функция	Трансформирует структуру JSON в соответствии с указанной структурой. Выдает ошибку в случае несоответствия структур или типов.
json_valid	Функция	Проверяет, является ли аргумент валидной структурой JSON.
json_value	Функция	Извлекает значения из структуры JSON по указанному пути.
lag	Функция	Возвращает значение, вычисленное для строки, сдвинутой на offset строк от текущей к началу группы.
last_value	Функция	Возвращает значение, вычисленное по указанному выражению для последней строки группы.
lcase	Функция	Псевдоним функции lower.
lcm	Функция	Вычисляет наименьшее общее кратное двух чисел.
lead	Функция	Возвращает значение, вычисленное для строки, сдвинутой на offset строк от текущей к концу группы.
least	Функция	Возвращает наименьшее число из указанных в аргументах.
least_common_multiple	Функция	Псевдоним функции lcm.
length	Функция	Возвращает количество символов в строке.
lgamma	Функция	Вычисляет логарифм гамма-функции.
LIKE	Выражение SQL	Элемент запроса SELECT.
LIMIT	Выражение SQL	Элемент запроса SELECT.
list	Функция	Псевдоним функции array_agg.

ln	Функция	Вычисляет натуральный логарифм числа.
log	Функция	Вычисляет логарифм числа по основанию 10.
log10	Функция	Псевдоним функции log.
log2	Функция	Вычисляет логарифм числа по основанию 2.
lower	Функция	Преобразует строку в нижний регистр.
ltrim	Функция	Удаляет все вхождения любого из указанных символов в начале строки.
max	Функция	Возвращает максимальное значение, имеющееся в argument.
md5	Функция	Возвращает хеш MD5 данных из argument в виде строки (VARCHAR).
mean	Функция	Псевдоним функции avg.
median	Функция	Возвращает медианное значение всех непустых значений в argument.
min	Функция	Возвращает минимальное значение, имеющееся в argument.
multiply	Функция	Перемножает два числа.
nextafter	Функция	Возвращает следующее значение с переменной точностью (типа DOUBLE) после первого числа в направлении второго числа.
now	Функция	Возвращает текущее время в виде значения типа TIMESTAMPTZ.
nth_value	Функция	Возвращает значение, вычисленное для n-ой строки внутри группы (считая с 1).
ntile	Функция	Разбивает каждую группу на num подгрупп равного (насколько возможно) размера и возвращает номер подгруппы.
NUMERIC	Тип данных	Вещественные числа с заданной точностью.
OFFSET	Выражение SQL	Элемент запроса SELECT.
ORDER BY	Выражение SQL	Элемент запроса SELECT.
percent_rank	Функция	Вычисляет относительный ранг текущей строки внутри группы.
pi	Функция	Возвращает значение числа π.
pow	Функция	Возводит первый аргумент в степень, заданную вторым аргументом.
power	Функция	Псевдоним функции pow.
QUALIFY	Выражение SQL	Элемент запроса SELECT.
radians	Функция	Переводит градусы в радианы.
random	Функция	Возвращает произвольное число (типа DOUBLE) в интервале от 0 до 1.
rank	Функция	Возвращает ранг (с пропусками) текущей строки внутри группы.

rank_dense	Функция	Псевдоним функции dense_rank.
read_csv	Функция	Читает файл .csv и записывает прочитанные данные в таблицу.
read_json	Функция	Читает файл .json и записывает прочитанные данные в таблицу.
read_json_auto	Функция	Псевдоним функции read_json.
read_parquet	Функция	Читает файл .parquet и записывает прочитанные данные в таблицу.
read_xlsx	Функция	Читает файл .xlsx и записывает прочитанные данные в таблицу.
regexp_extract	Функция	Извлекает из строки подстроку по заданному регулярному выражению.
regexp_extract_all	Функция	Извлекает из строки все не пересекающиеся подстроки по заданному регулярному выражению. Возвращает массив подстрок.
regexp_full_match	Функция	Проверяет, накрывает ли регулярное выражение строку полностью.
regexp_matches	Функция	Проверяет, содержится ли регулярное выражение внутри строки.
regexp_replace	Функция	Заменяет подстроку, накрытую регулярным выражением, на указанную строку.
regexp_split_to_array	Функция	Разрезает строку на части, разделенные регулярным выражением, и возвращает части в виде массива.
regexp_split_to_table	Функция	Разрезает строку на части, разделенные регулярным выражением, и возвращает части в виде строк.
replace	Функция	Заменяет все подстроки на указанную строку.
REVOKE	Выражение SQL	Отзыв привилегий.
REVOKE ROLE	Выражение SQL	Отзыв ролей у пользователей.
round	Функция	Округляет число из первого аргумента с указанной во втором аргументе точностью.
round_even	Функция	Округляет число из первого аргумента до ближайшего четного с указанной во втором аргументе точностью.
roundbankers	Функция	Псевдоним функции round_even.
row_number	Функция	Возвращает номер текущей строки в её группе (считая с 1).
rtrim	Функция	Удаляет все вхождения любого из указанных символов в конце строки.
SELECT	Выражение SQL	Запрос SELECT.
setseed	Функция	Фиксирует начальное значение для функции random().
sha1	Функция	Возвращает хеш SHA-1 данных из argument в виде строки (VARCHAR).

sha256	Функция	Возвращает хеш SHA-256 данных из argument в виде строки (VARCHAR).
SHOW ROLES	Выражение SQL	Вывод списка всех ролей.
SHOW TABLES	Выражение SQL	Вывод списка всех таблиц.
SHOW USERS	Выражение SQL	Вывод списка всех пользователей.
SHOW WORKER POOL	Выражение SQL	Вывод текущего вычислительного пула.
SHOW WORKER POOLS	Выражение SQL	Вывод списка всех вычислительных пулов.
sign	Функция	Возвращает -1, 1 или 0 в зависимости от знака аргумента.
signbit	Функция	Определяет, установлен ли бит знака у вещественного числа.
SIMILAR TO	Выражение SQL	Элемент запроса SELECT.
sin	Функция	Вычисляет синус угла, заданного в радианах.
split	Функция	Делит строку на две части по заданному сепаратору.
sqrt	Функция	Вычисляет квадратный корень.
ST_Distance	Функция	Вычисляет расстояние между двумя точками на плоскости.
ST_Distance_Sphere	Функция	Вычисляет расстояние между двумя точками на сфере.
ST_Point	Функция	Создает точку типа GEOMETRY.
str_split	Функция	Псевдоним функции split.
STRING	Тип данных	Псевдоним типа данных VARCHAR.
STRING[]	Тип данных	Псевдоним типа данных VARCHAR[].
string_split	Функция	Псевдоним функции split.
string_split_regex	Функция	Псевдоним функции regexp_split_to_array.
string_to_array	Функция	Псевдоним функции split.
strlen	Функция	Возвращает количество байтов в строке.
strptime	Функция	Конвертирует текст в момент времени по указанному формату.
substr	Функция	Псевдоним функции substring.
substring	Функция	Извлекает подстроку из строки.
subtract	Функция	Вычитает второй аргумент из первого.
sum	Функция	Вычисляет сумму всех непустых значений в argument.
TEXT	Тип данных	Псевдоним типа данных VARCHAR.
TEXT[]	Тип данных	Псевдоним типа данных VARCHAR[].
TIME	Тип данных	Время.
TIMESTAMP	Тип данных	Моменты времени.
TIMESTAMPTZ	Тип данных	Моменты времени с часовым поясом.
tngri.run_notebook	Функция Python	Запускает указанный ноутбук из другого ноутбука.

`tngri.sql`	Функция Python	Выполняет указанный запрос SQL внутри ячейки типа Python.
`tngri.upload_df`	Функция Python	Загружает данные из DataFrame в Tengri.
`tngri.upload_file`	Функция Python	Загружает данные из файла в Tengri.
`tngri.upload_s3`	Функция Python	Загружает файл из указанного бакета S3 в Tengri.
`trim`	Функция	Удаляет все вхождения любого из указанных символов с обеих сторон строки.
`trunc`	Функция	Отбрасывает все знаки после десятичного разделителя.
`TRUNCATE TABLE`	Выражение SQL	Удаление всех строк из таблицы.
`try_cast`	Функция	Приводит тип данных к указанному.
`ucase`	Функция	Псевдоним функции `upper`.
`UNION`	Выражение SQL	Элемент запроса `SELECT`.
`unnest`	Функция	Разворачивает списки или структуры из `argument` в множество отдельных значений.
`UPDATE`	Выражение SQL	Элемент запроса `SELECT`.
`upper`	Функция	Преобразует строку в верхний регистр.
`USE ROLE`	Выражение SQL	Установка активной роли.
`USE WORKER POOL`	Выражение SQL	Установка текущего вычислительного пула.
`VARCHAR`	Тип данных	Текстовые строки.
`VARCHAR[]`	Тип данных	Массивы текстовых строк.
`WHERE`	Выражение SQL	Элемент запроса `SELECT`.
`WITH`	Выражение SQL	Элемент запроса `SELECT`.
`%`	Оператор	Возвращает остаток от деления левого аргумента на правый.
`*`	Оператор	Перемножает аргументы.
`+`	Оператор	Прибавляет правый аргумент к левому.
`-`	Оператор	Вычитает правый аргумент из левого.
`/`	Оператор	Делит левый аргумент на правый.
`::`	Оператор	Приводит тип данных к указанному.
`^`	Оператор	Возводит левый аргумент в степень, заданную правым аргументом.
`||`	Оператор	Конкатенирует несколько строк, массивов или двоичных значений.
`~`	Оператор	Проверяет, накрывает ли регулярное выражение строку полностью.

Всего объектов: 231