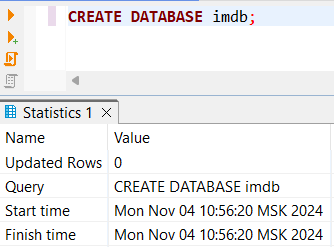
1. **Создать БД и таблицы**

CREATE DATABASE imdb;



CREATE TABLE imdb.actors

(

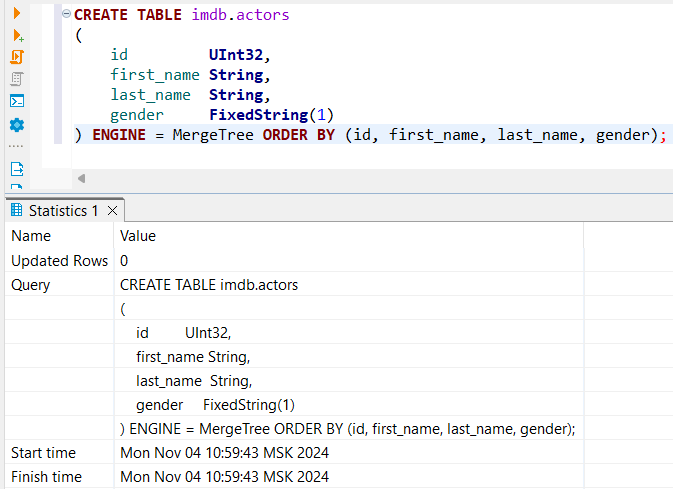
id UInt32,

first\_name String,

last\_name String,

gender FixedString(1)

) ENGINE = MergeTree ORDER BY (id, first\_name, last\_name, gender);



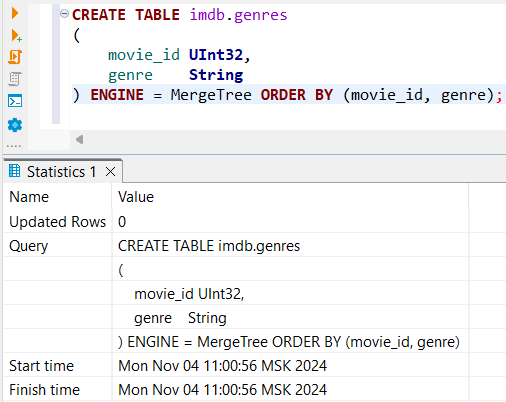
CREATE TABLE imdb.genres

(

    movie\_id UInt32,

    genre    String

) ENGINE = MergeTree ORDER BY (movie\_id, genre);



CREATE TABLE imdb.movies

(

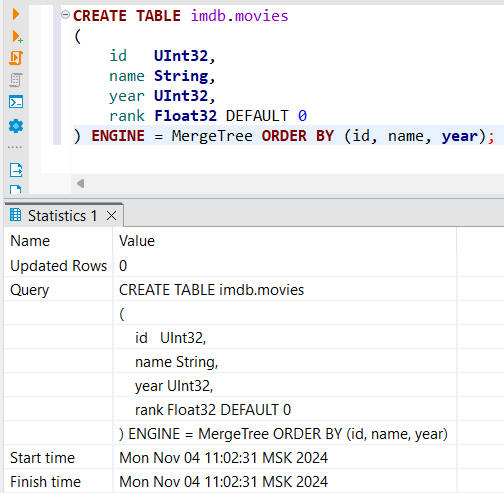
    id   UInt32,

    name String,

    year UInt32,

    rank Float32 DEFAULT 0

) ENGINE = MergeTree ORDER BY (id, name, year);



CREATE TABLE imdb.roles

(

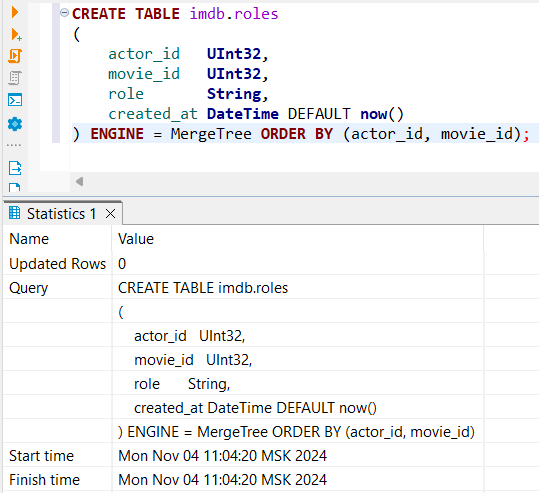
    actor\_id   UInt32,

    movie\_id   UInt32,

    role       String,

    created\_at DateTime DEFAULT now()

) ENGINE = MergeTree ORDER BY (actor\_id, movie\_id);

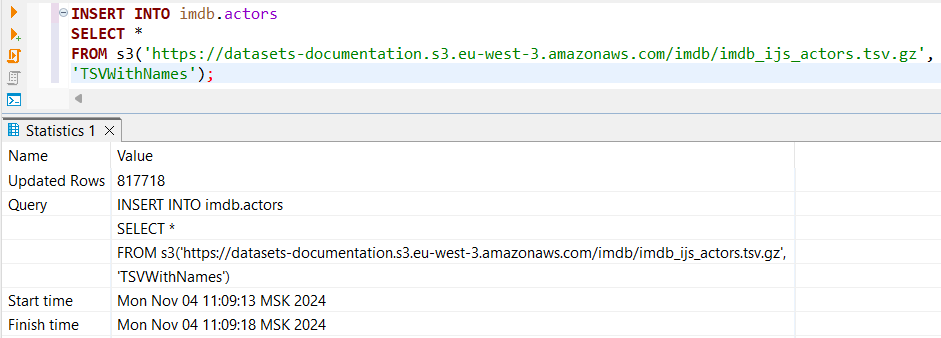


1. **Вставить тестовые данные, используя функцию S3**

INSERT INTO imdb.actors

SELECT \*

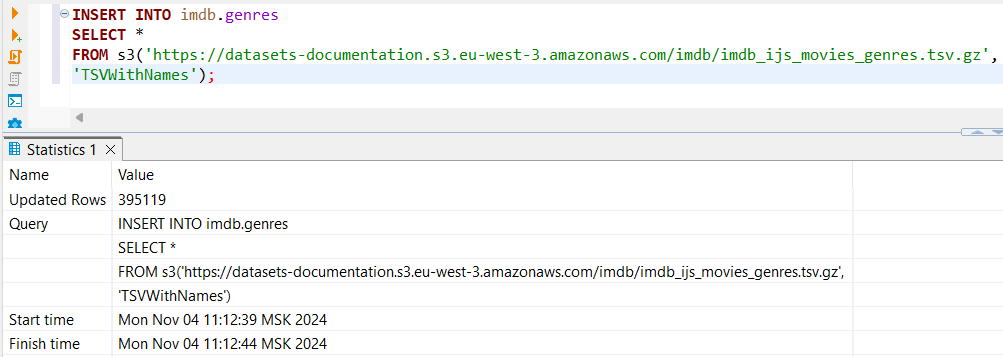
FROM s3('https://datasets-documentation.s3.eu-west-3.amazonaws.com/imdb/imdb\_ijs\_actors.tsv.gz', 'TSVWithNames');



INSERT INTO imdb.genres

SELECT \*

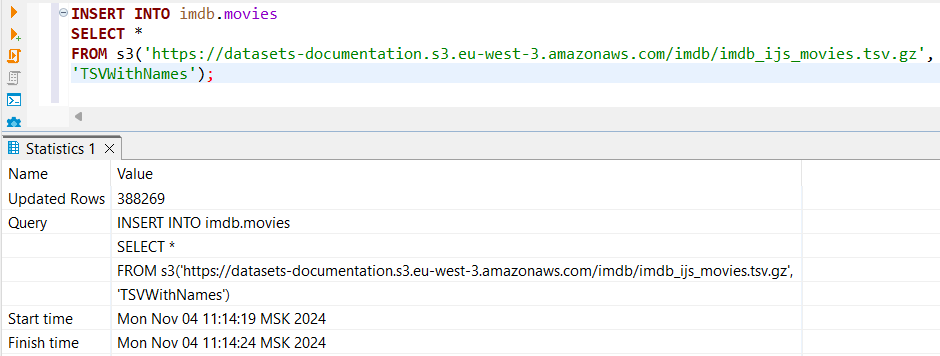
FROM s3('https://datasets-documentation.s3.eu-west-3.amazonaws.com/imdb/imdb\_ijs\_movies\_genres.tsv.gz', 'TSVWithNames');



INSERT INTO imdb.movies

SELECT \*

FROM s3('https://datasets-documentation.s3.eu-west-3.amazonaws.com/imdb/imdb\_ijs\_movies.tsv.gz', 'TSVWithNames');

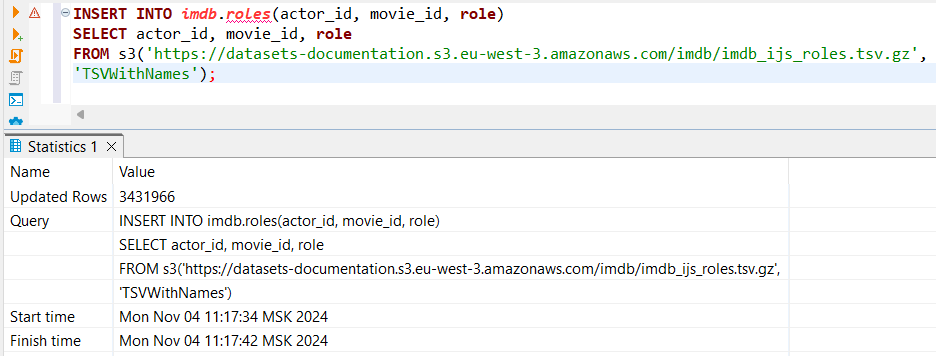


INSERT INTO imdb.roles(actor\_id, movie\_id, role)

SELECT actor\_id, movie\_id, role

FROM s3('https://datasets-documentation.s3.eu-west-3.amazonaws.com/imdb/imdb\_ijs\_roles.tsv.gz',

'TSVWithNames');



1. **Используя изученные материалы, построить запросы, отвечающие на следующие задачи:**
   * **Найти жанры для каждого фильма (не только для которых жанры определены)**

select

m.id as movie\_id,

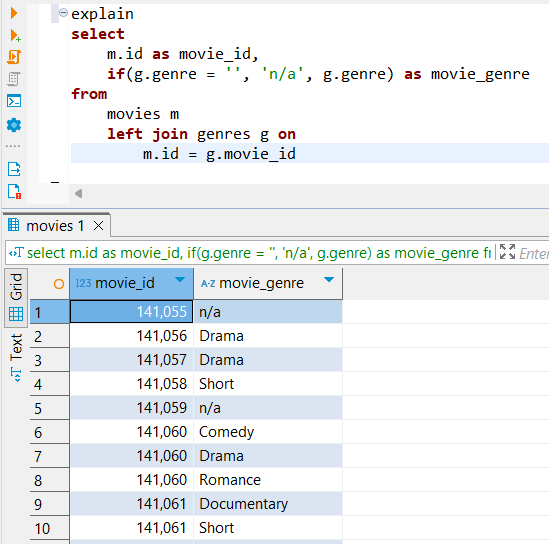
if(g.genre = '', 'n/a', g.genre) as movie\_genre

from

movies m

left join genres g on

m.id = g.movie\_id

****

* + **Найти жанры для каждого фильма (только для которых жанры определены)**

select

*m*.id as *movie\_id*,

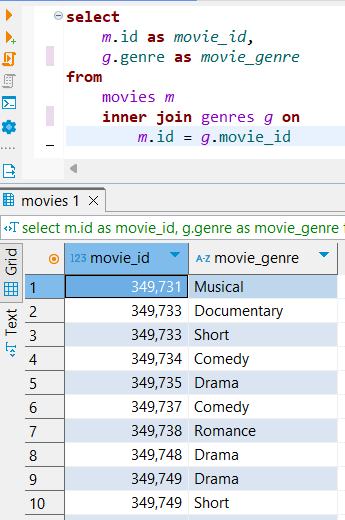
*g*.genre as *movie\_genre*

from

movies *m*

inner join genres *g* on

*m*.id = *g*.movie\_id

****

* + **Запросить все фильмы, у которых нет жанра**

select

m.id as movie\_id,

m.name as movie\_name

from

movies m

left join genres g on

m.id = g.movie\_id

where

g.genre = '';

select

m.id as movie\_id,

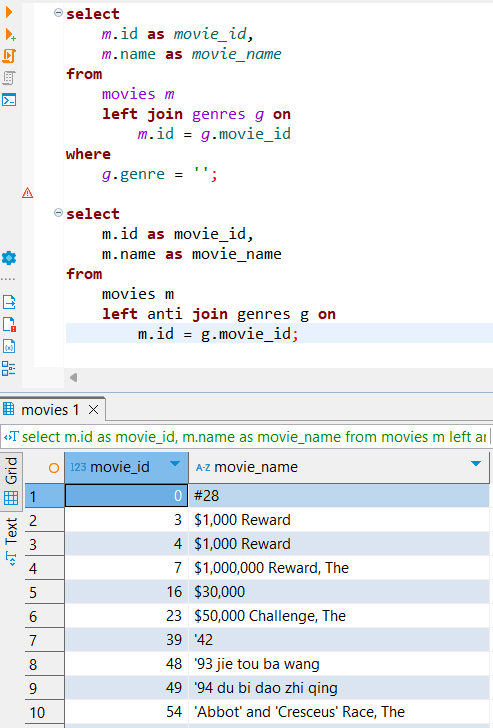
m.name as movie\_name

from

movies m

left anti join genres g on

m.id = g.movie\_id;

****

* + **Объединить каждую строку из таблицы “Фильмы” с каждой строкой из таблицы “Жанры”**

select

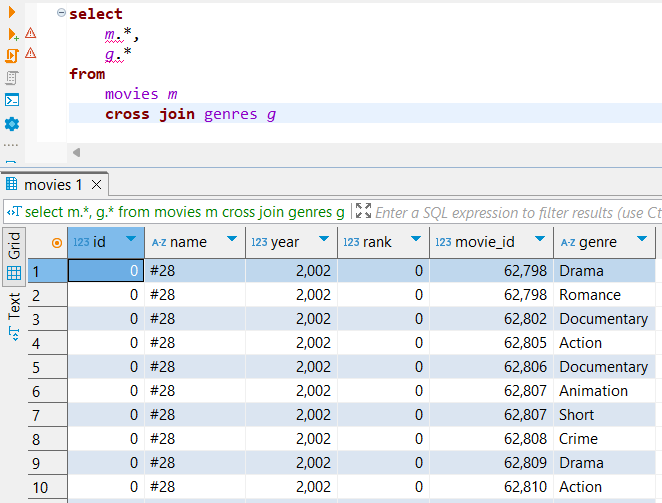
*m*.\*,

*g*.\*

from

movies *m*

cross join genres *g*



* + **Найти жанры для каждого фильма, НЕ используя INNER JOIN**

**не только для которых жанры определены – уже написан**

**только для которых жанры определены**

select

*m*.id as *movie\_id*,

*g*.genre as *movie\_genre*

from

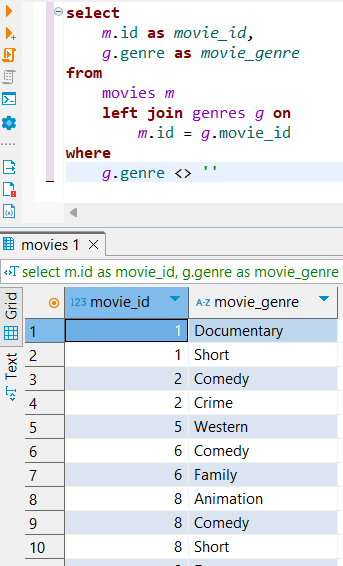
movies *m*

left join genres *g* on

*m*.id = *g*.movie\_id

where

*g*.genre <> ''

****

* + **Найти всех актеров и актрис, снявшихся в фильме в 2023 году**

SELECT

*a*.id,

*a*.first\_name,

*a*.last\_name,

*a*.gender

FROM

imdb.actors as a

left semi join (

select

actor\_id

from

imdb.roles as r

where

movie\_id in (select id from imdb.movies as m where m.year = 2023)

) as b on b.actor\_id = a.id

* + **Запросить все фильмы, у которых нет жанра, через ANTI JOIN**

select

*m*.id as *movie\_id*,

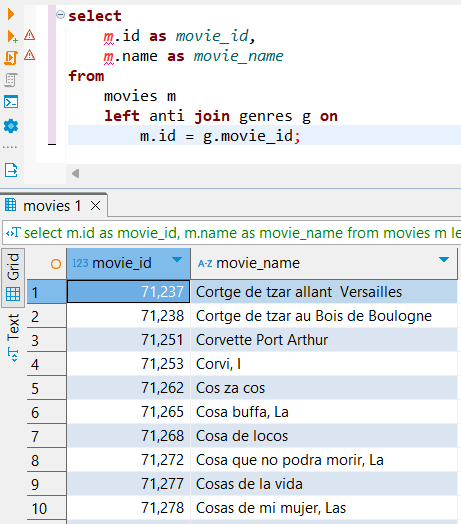
*m*.name as *movie\_name*

from

movies m

left anti join genres g on

m.id = g.movie\_id;

****