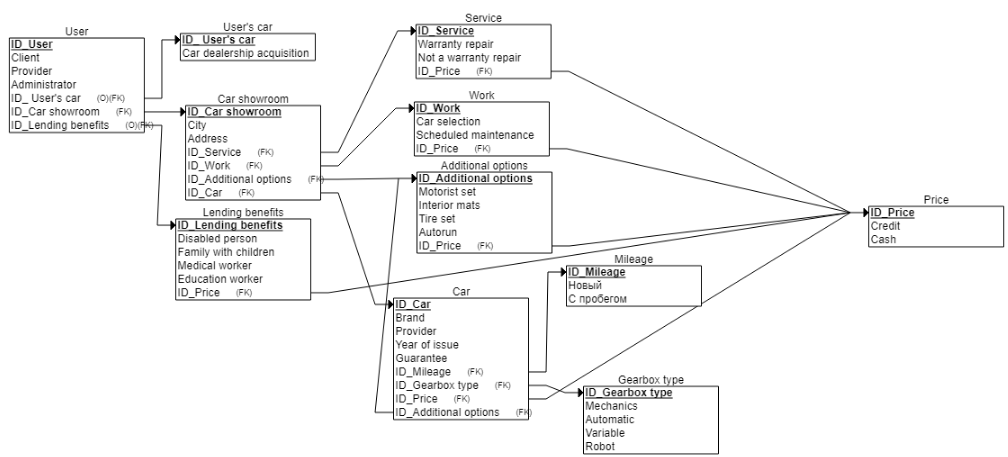
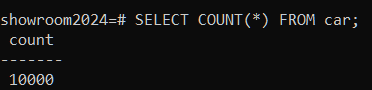
|  |  |  |
| --- | --- | --- |
| Лабораторная работа №5  Индексы, хранимые процедуры, представления, триггеры | Ф.И.О. | Сидский Н.А. |
| Группа | ИВТ-261 |
| Преподаватель | Аль-Мерри Гаис М. С. |
| Дата сдачи |  |



1. Добавление 10000 записей в основные таблицы.

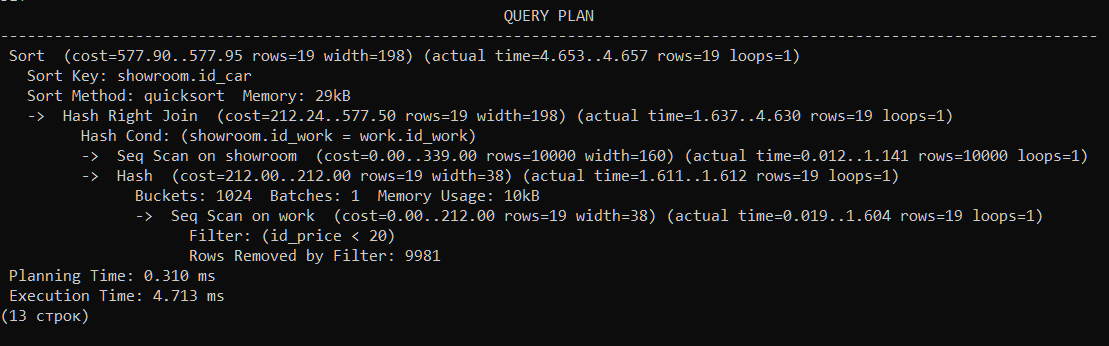
Для добавления обширного количества записей в нашу БД необходимо воспользоваться генератором случайных данных (например, mocaroo), он сгенерирует exel файл в формате CVS. Теперь мы можем скопировать данные при помощи следующего скрипта:

copy {TABLE NAME} from {PATH TO FILE} delimiter ',' CSV HEADER;



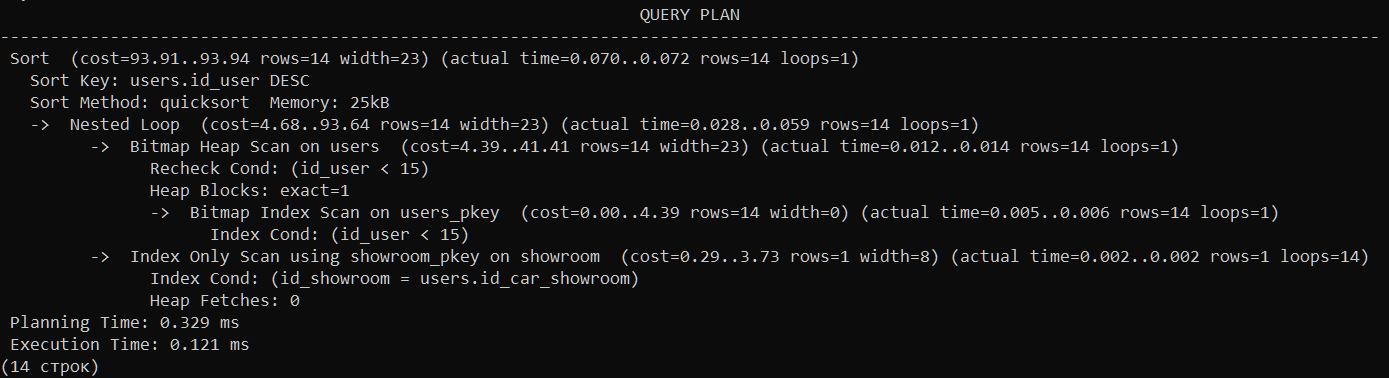
1. Измерение времени выполнения сложных запросов.

SET track\_io\_timing = TRUE; EXPLAIN(ANALYZE)

select \* from showroom right join work on showroom.id\_work = work.id\_work where id\_price < 20 order by id\_car asc;

SET track\_io\_timing = TRUE; EXPLAIN(ANALYZE)

select id\_user, client, id\_car\_showroom from users right join showroom on users.id\_car\_showroom = showroom.id\_showroom where id\_user < 15 order by id\_user desc;



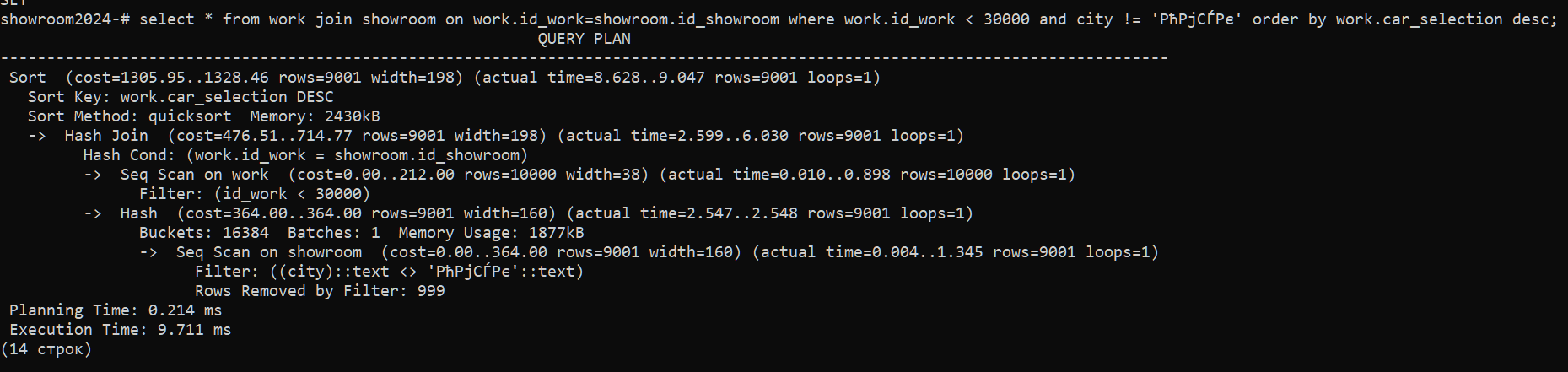
1. Анализ скорости выполнения запросов при помощи EXPLAIN и последующие визуализация и анализ.

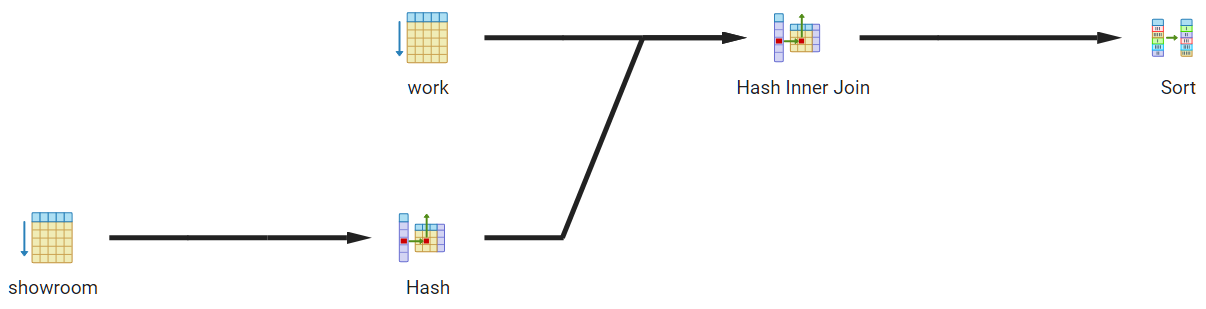
Запрос №1:

SET track\_io\_timing = TRUE; EXPLAIN(ANALYZE)

select \* from work join showroom on work.id\_work=showroom.id\_showroom where work.id\_work < 30000 and city != 'РћРјСЃРє' order by work.car\_selection desc;

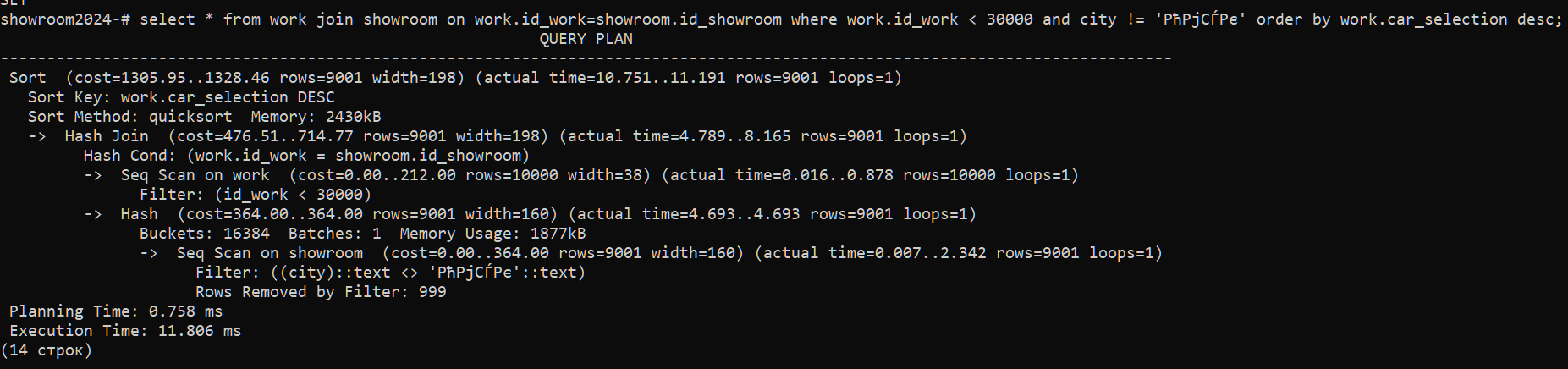
Объединяет таблицы work и showroom при помощи оператора JOIN. Условие объединения таблиц: work.id\_work=showroom.id\_showroom. Затем при помощи фильтрации WHERE выбираются только те строки, где work.id\_work < 30000 и city не является 'РћРјСЃРє'. Результаты сортируются по убыванию столбца car\_selection.





Добавим индексы в столбец car\_selection для ускорения сортировки:

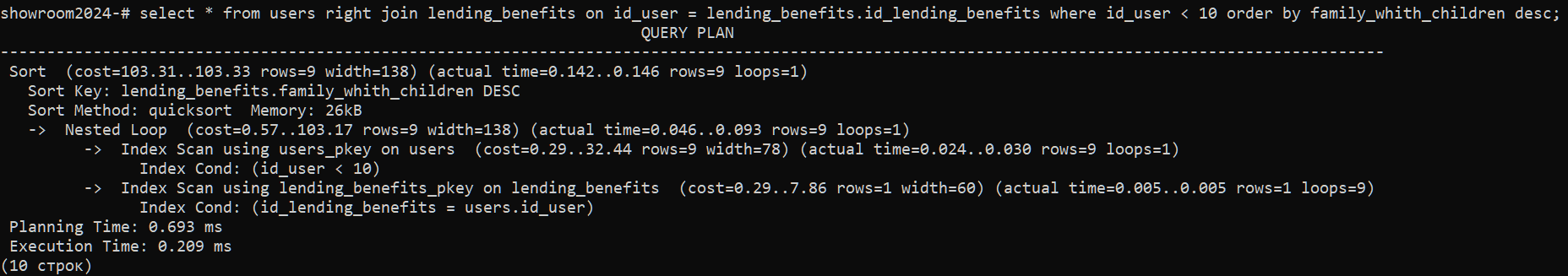
Create index selection\_index on work(car\_selection);

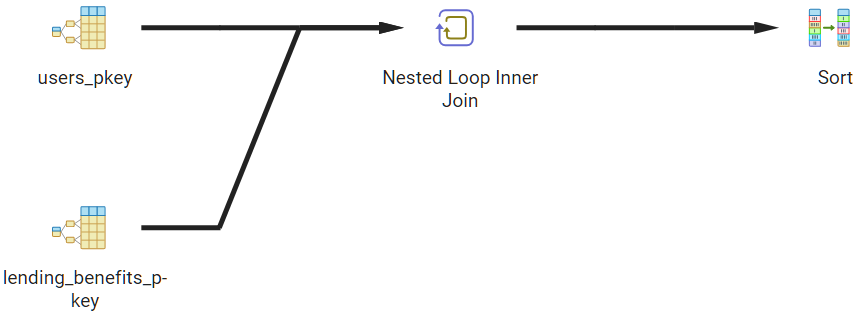


Запрос №2:

SET track\_io\_timing = TRUE; EXPLAIN(ANALYZE)

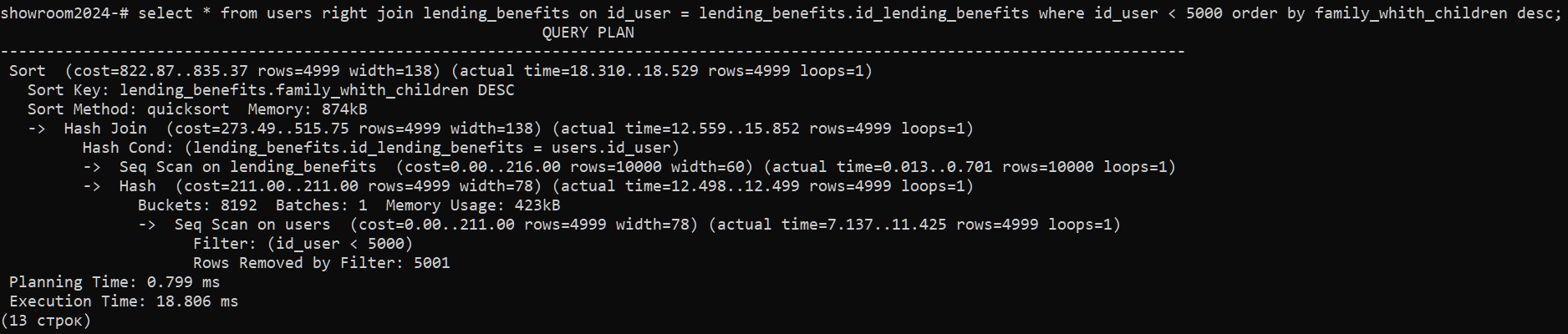
select \* from users right join lending\_benefits on id\_user = lending\_benefits.id\_lending\_benefits where id\_user < 5000 order by family\_whith\_children desc;





Для ускорения сортировки добавим индекс в столбец family\_whith\_children:

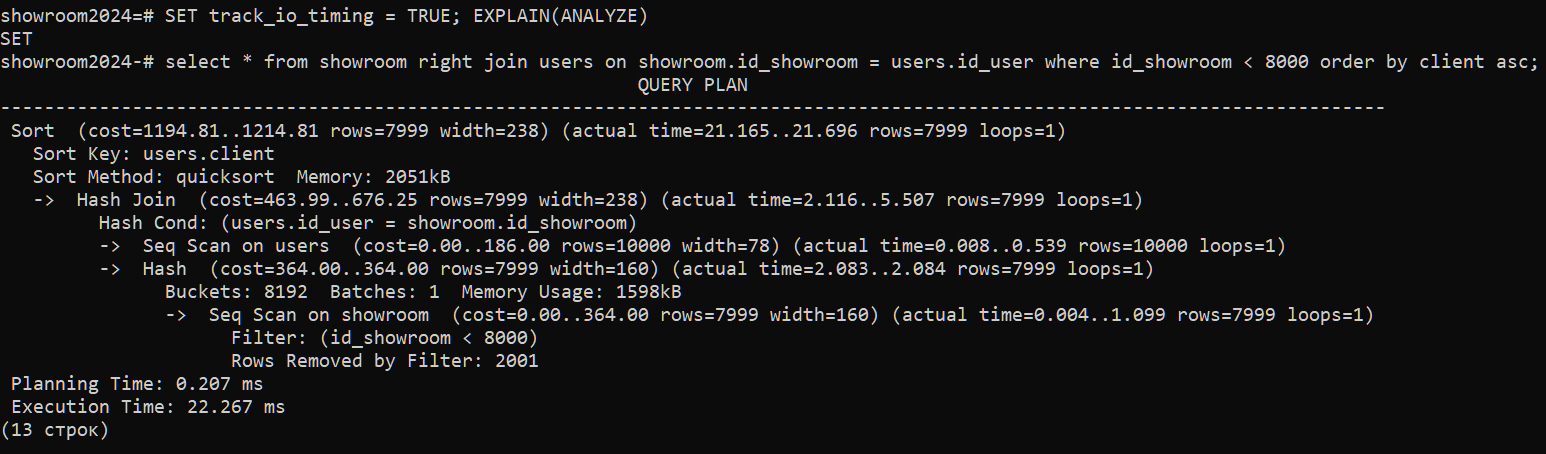
Create index family\_whith\_children\_index on lending\_benefits(family\_whith\_children);

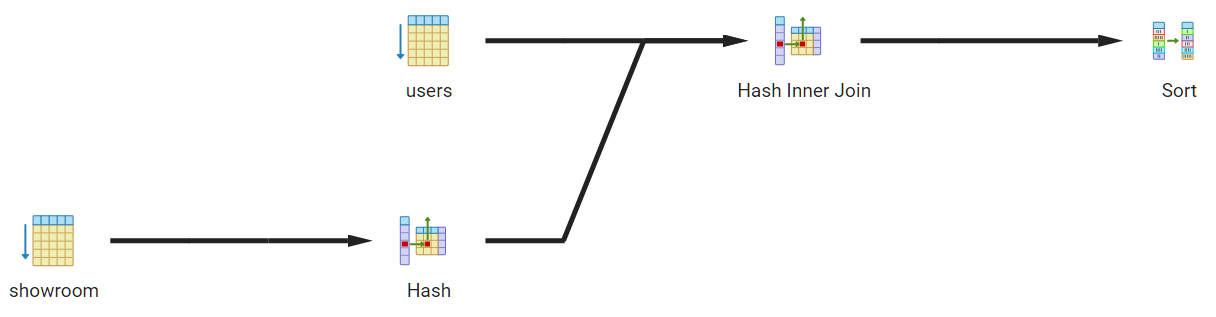


Запрос №3

SET track\_io\_timing = TRUE; EXPLAIN(ANALYZE)

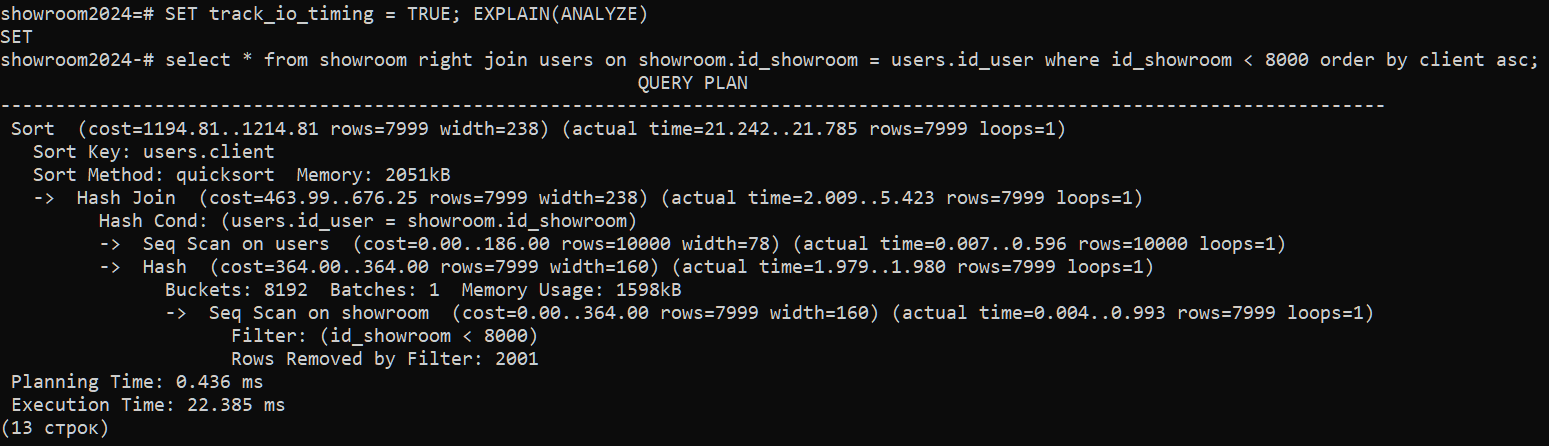
select \* from showroom right join users on showroom.id\_showroom = users.id\_user where id\_showroom < 8000 order by client asc;





Для ускорения сортировки добавим индекс в столбец client:

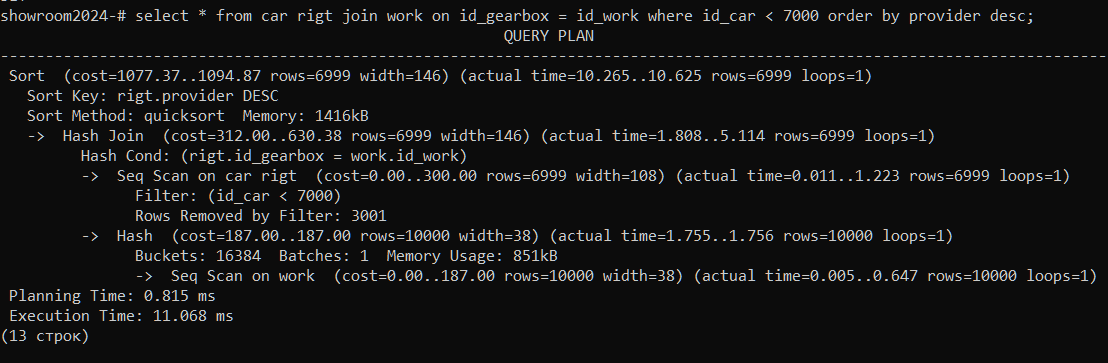
Create index client\_index on users(client);

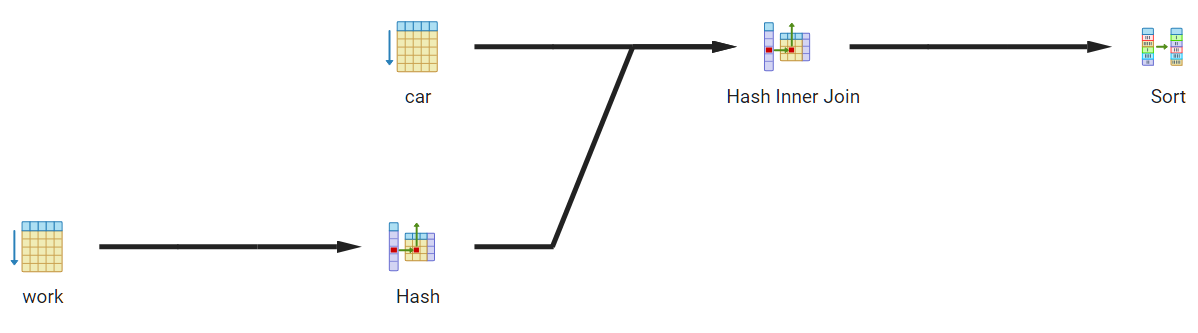


Запрос №4:

SET track\_io\_timing = TRUE; EXPLAIN(ANALYZE)

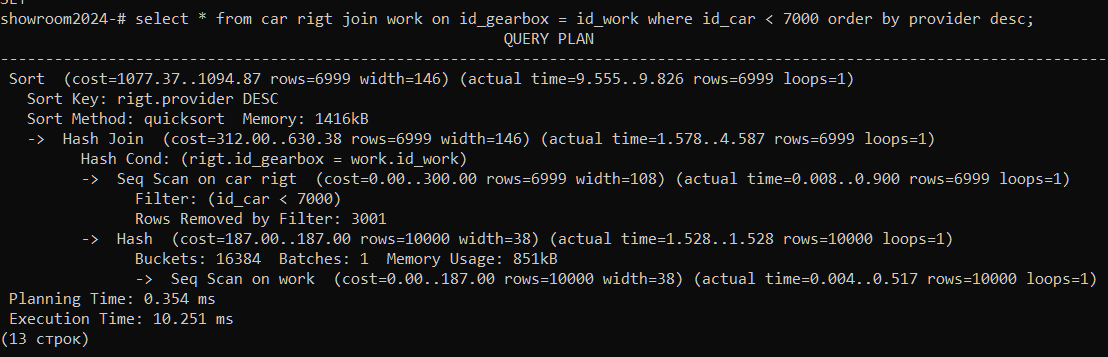
select \* from car rigt join work on id\_gearbox = id\_work where id\_car < 7000 order by provider desc;





Для ускорения сортировки добавим индексы в provider:

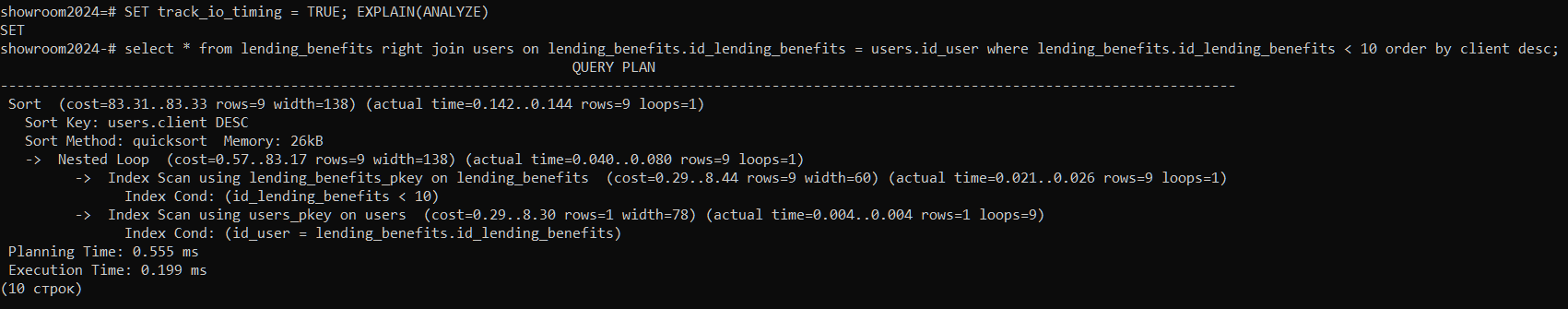
Create index provider\_index on car(provider);

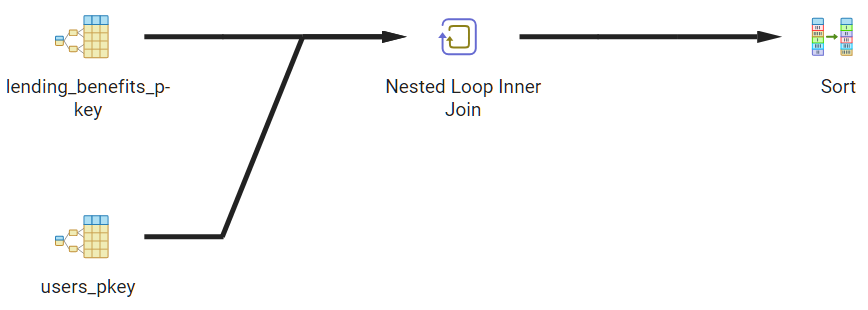


Запрос №5:

SET track\_io\_timing = TRUE; EXPLAIN(ANALYZE)

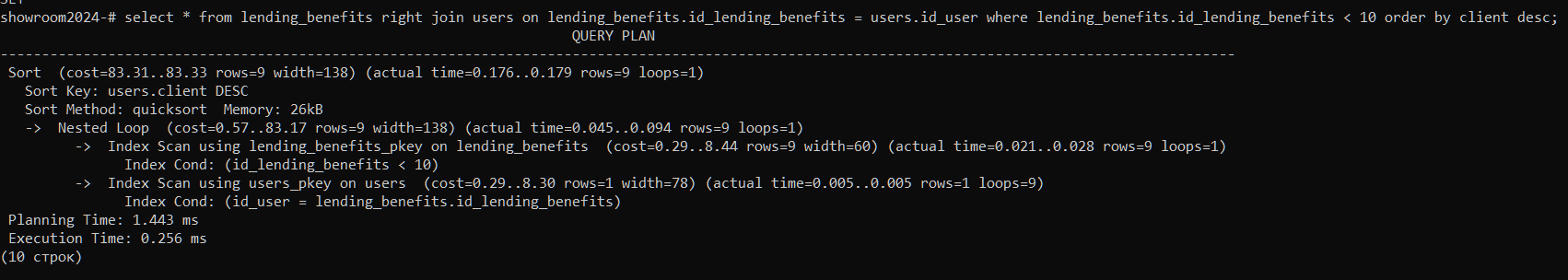
select \* from lending\_benefits right join users on lending\_benefits.id\_lending\_benefits = users.id\_user where lending\_benefits.id\_lending\_benefits < 10 order by client desc;





Для ускорения сортировки добавим индексы в client:

create index clients\_index on users(client);



**Хранимые процедуры и функции**

Функции

1. Данная функция выводит значение, равное разности наибольшего пробега и наименьшего из таблицы mileage по колонке mileage

create or replace function mileage\_maxmin()

returns int

as $$

declare

max\_val integer;

min\_val integer;

begin

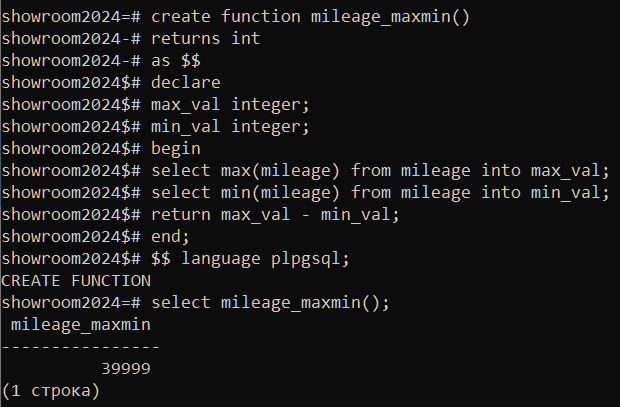
select max(mileage) from mileage into max\_val;

select min(mileage) from mileage into min\_val;

return max\_val – min\_val;

end;

$$ language plpgsql;



1. Данная функция выводит количество автомобилей с определенной маркой из таблицы car по колонке brand

CREATE OR REPLACE FUNCTION kolvo\_brand(val1 varchar)

RETURNS TABLE (provider varchar, counter bigint) AS $$

BEGIN

RETURN QUERY

SELECT car.provider, COUNT(\*) AS counter

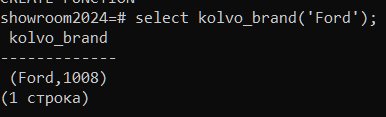
FROM car

WHERE car.provider = val1

GROUP BY car.provider;

END;

$$ LANGUAGE plpgsql;



1. Функция, которая определяет, нужна ли доп гарантия для авто с таким пробегом

CREATE OR REPLACE FUNCTION add\_garant(val1 integer)

returns varchar(100)

as $$

declare

val\_test varchar(100);

begin

if val1 < 20000 then

val\_test := 'ne nado';

else

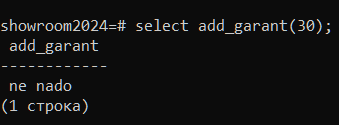
val\_test := 'nado';

end if;

return val\_test;

end;

$$ language plpgsql;



Процедуры

1. Процедура изменения данных в таблице price

CREATE or replace PROCEDURE lending\_benefits\_ (index1 integer, name1 varchar(40), name2 varchar(40))

AS $$

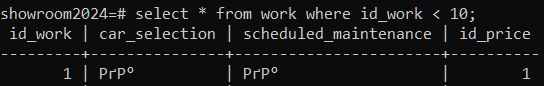
begin

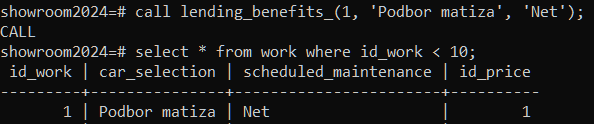
update work set car\_selection = name1 where id\_work = index1;

update work set scheduled\_maintenance = name2 where id\_work = index1;

end;

$$ language plpgsql;





1. Процедура добавления значений в таблицу

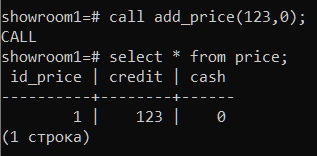
CREATE or replace PROCEDURE add\_price (a integer, b integer)

LANGUAGE SQL

AS $$

INSERT INTO price(credit, cash) VALUES (a, b);

$$;



1. Процедура удаления строки с заданным индексом

CREATE or replace PROCEDURE del\_price(index1 bigint)

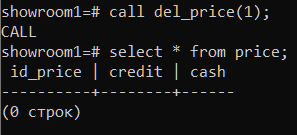
AS $$

begin

delete from price where id\_price = index1;

end;

$$ language plpgsql;



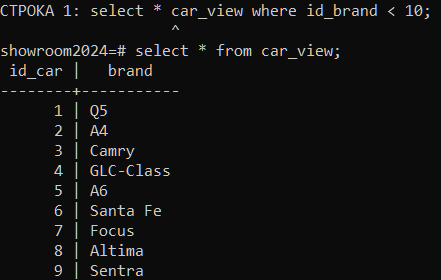
**Представления**

1. Представление, выводящее id автомобиля и его марку (brand) из таблицы car

CREATE VIEW car\_view AS

SELECT id\_car, provider

FROM car;



1. Представление, выводящее car\_dealership и id\_user

CREATE OR REPLACE VIEW record\_patient AS

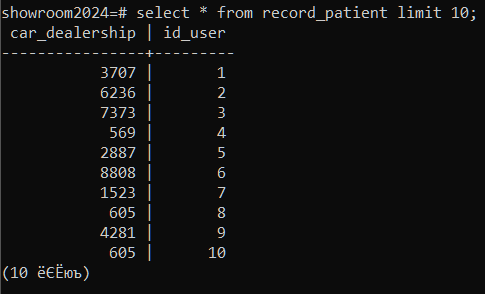
SELECT

c.car\_dealership,

u.id\_user

FROM users\_car c

JOIN users u ON u.id\_user= c.id\_users\_car;



1. Представление, выводящее address и id\_work

CREATE OR REPLACE VIEW carr AS

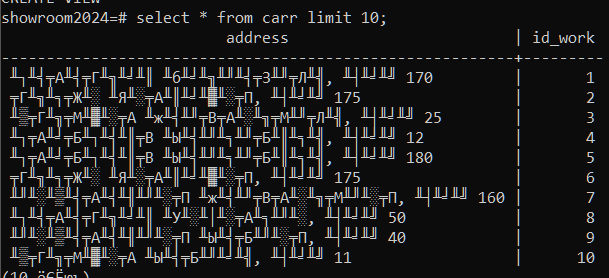
SELECT

c.address,

u.id\_work

FROM showroom c

JOIN work u ON u.id\_work = c.id\_showroom;



**Триггер**

CREATE OR REPLACE FUNCTION avg\_mileage()

RETURNS TRIGGER AS $$

BEGIN

NEW.avg\_mile = NEW.mileage + 5000;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER mile\_triger

BEFORE UPDATE ON mileage

FOR EACH ROW

EXECUTE FUNCTION avg\_mileage();

