postgres=# \c task\_schedule

task\_schedule=# CREATE TABLE usres(

task\_schedule(# email VARCHAR(50) NOT NULL PRIMARY KEY,

task\_schedule(# password VARCHAR(50) NOT NULL,

task\_schedule(# name VARCHAR(50) NOT NULL);

task\_schedule=# CREATE TABLE priority(

task\_schedule(# priority\_name VARCHAR(50) NOT NULL PRIMARY KEY,

task\_schedule(# color VARCHAR(50) NOT NULL,

task\_schedule(# email VARCHAR(50),

task\_schedule(# FOREIGN KEY (email) REFERENCES usres(email));

task\_schedule=# CREATE TABLE category(

task\_schedule(# category\_name VARCHAR(50) NOT NULL PRIMARY KEY,

task\_schedule(# email VARCHAR(50),

task\_schedule(# FOREIGN KEY (email) REFERENCES usres(email));

task\_schedule=# CREATE TABLE task(

task\_schedule(# task\_name VARCHAR(50) NOT NULL PRIMARY KEY,

task\_schedule(# status BOOLEAN NOT NULL,

task\_schedule(# data DATE,

task\_schedule(# priority\_name VARCHAR(50),

task\_schedule(# category\_name VARCHAR(50),

task\_schedule(# email VARCHAR (50),

task\_schedule(# FOREIGN KEY (priority\_name) REFERENCES priority(priority\_name),

task\_schedule(# FOREIGN KEY (category\_name) REFERENCES category(category\_name),

task\_schedule(# FOREIGN KEY (email) REFERENCES usres(email));

task\_schedule=# INSERT INTO usres VALUES(

task\_schedule(# 'admin@bk.com', 'ssss', 'Dastan'),

task\_schedule-# ('user@fr.com', 'dedede', 'Bakdaulet'),

task\_schedule-# ('user2@mail.ru', '456', 'Ramazan');

task\_schedule=# INSERT INTO priority VALUES

task\_schedule-# ('high', 'red', 'user2@mail.ru'),

task\_schedule-# ('medium', 'yellow', 'user@fr.com'),

task\_schedule-# ('low', 'green', 'admin@bk.com');

task\_schedule=# INSERT INTO category VALUES

task\_schedule-# ('Finish\_a\_work', 'user2@mail.ru'),

task\_schedule-# ('turn\_in\_work', 'user@fr.com'),

task\_schedule-# ('Check', 'admin@bk.com');

task\_schedule=# INSERT INTO task VALUES

task\_schedule-# ('make it to the deadline', true, '20022-05-04', 'high', 'Finish\_a\_work', 'user2@mail.ru'),

task\_schedule-# ('turn', true, '2022-05-04', 'medium', 'turn\_in\_work', 'user@fr.com'),

task\_schedule-# ('wait until everyone is done', true, '2022-05-04', 'low', 'Check', 'admin@bk.com');

Вопросы

1. Что такое PK и в чем отличие от FK?

PK - это столбец в таблице, который однозначно идентифицирует каждую строку в таблице.

PK должен содержать уникальные значения и не может быть пустым. Каждого таблица может иметь только один РК.

FK- обеспечивает логическую связь между таблицами.

1. Напишите все что знаете про объединение данных.

Операция объединения(JOIN)- позволяет запрашивать информацию из двух или более связанных таблиц. Это похоже на условие выбора, за исключением того, что сравниваются значения в двух разных таблицах.

INNER JOIN- возвращает все строки из нескольких таблиц , в которых выполнено условие соединения.

LEFT JOIN- возвращает все строки из левой таблицы с соответствующими строками в правой

таблице. Результат равен нулю в правой части , если совпадения нет.

RIGHT JOIN- возвращает все строки из правой таблицы с соответствующими строками в левой таблице.

FULL OUTER JOIN возвращает включая все значение с левого и правого таблица.

CROSS JOIN- создает результирующий набор, который представляет собой количество строк в первой таблице, умноженное на количество строк во второй таблице

1. Какие существуют связи таблиц? дайте примеры (создайте таблицы со связями) и скрины прикрепите к ответам.

Связь один ко многим в реляционных базах данных реализуется тогда, когда объекту А может принадлежать или же соответствовать несколько объектов Б, но объекту Б может соответствовать только один объект

Связь многие ко многим реализуется в том случае, когда нескольким объектам из таблицы

Связь один к одному – самая редко встречаемая связь между таблицами

4. Какие бывают агрегатные функции?

Агрегатные функции SQL извлекают информацию о данных в базе данных. Например, вы можете использовать COUNT, чтобы узнать, сколько записей находится в базе данных. Допустимыми агрегатными функциями являются: COUNT, MIN, MAX, AVG и SUM.

Count-выдает количество строк или не NULL значений полей которые выбрал запрос

Sum возвращает арифметическую сумму

AVG- возвращает средную значению данного данных

MAX- возвращает наибольшее значение из выбранного поля

MIN – возвращает наименьшее значение из выбранного поля

6. Что такое 1НФ, 2НФ, 3НФ (1NF, 2NF, 3NF)?

Relation is in First Normal Form (1NF)- если каждый столбец представляет собой одно атомарное значение.

2 NF- он находится в 1NF и не имеет частичных зависимостей. Вторая нормальная форма связана с аномалией модификации.

3 NF - если оно находится в 2NF и не имеет transitive dependencies.

7. В чем отличие materialized view от обычного view?

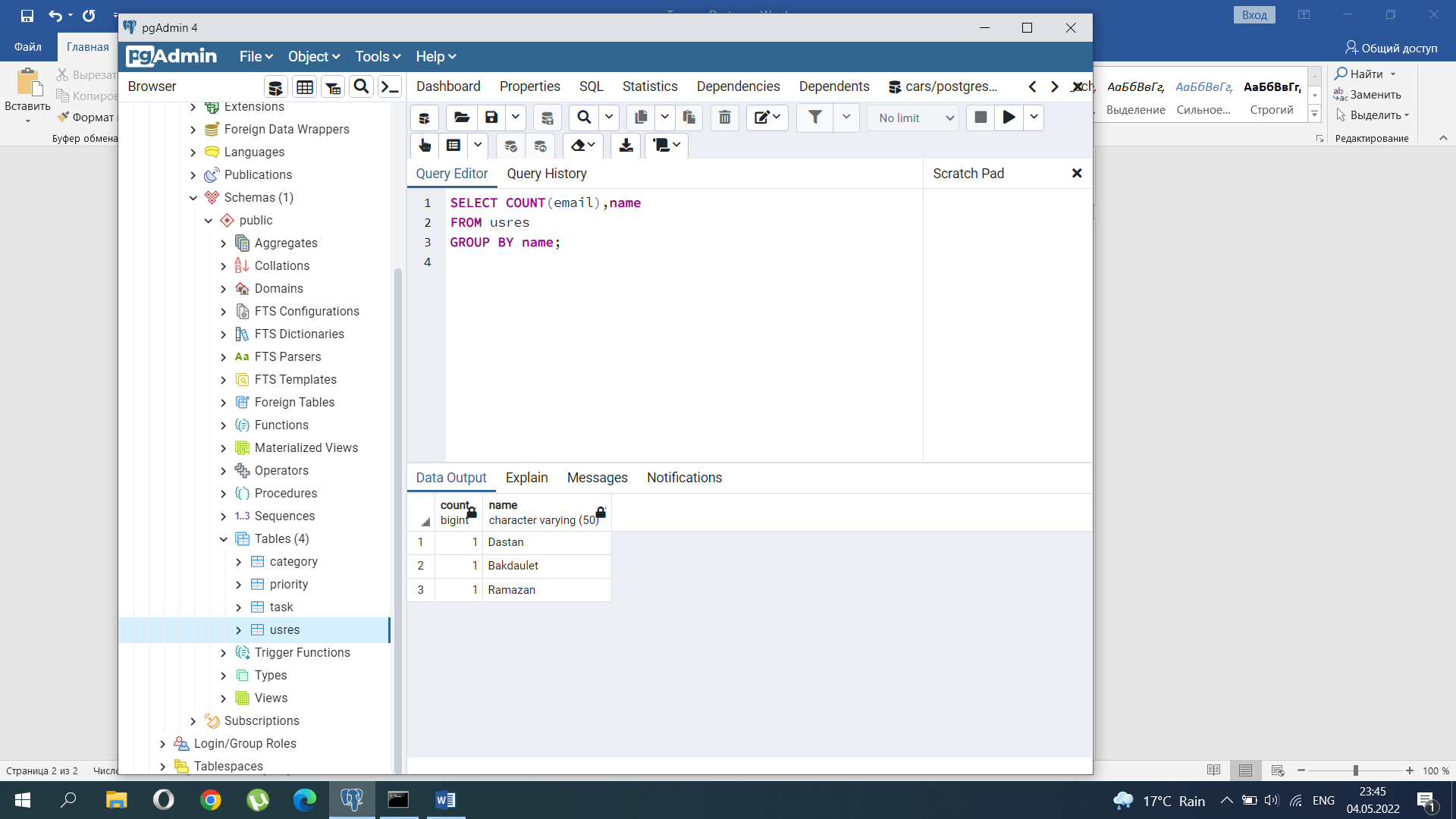
View- это виртуальная таблица которая создается с помощью команды Create View. это виртуальная таблица (представление). Она хранит результат выполнения SELECT запроса. С помощью вьюхи можно сохранить SELECT запрос и получить быстрый доступ к часто используемым данным.

Materialized view- это физическая копия исходных базовых таблиц. Материализованное представление выглядит как снимок или изображение исходных базовых таблиц. Как и Представление, оно также содержит данные, извлеченные из выражения запроса команды Create Materialized View. Но в отличие от представления, материализованное представление предварительно вычисляется и сохраняется на диске как объект, и они не обновляются каждый раз, когда используются. Вместо этого материализованное представление должно обновляться вручную или с помощью триггеров. Процесс обновления Материализованного представления называется Поддержкой Материализованного представления

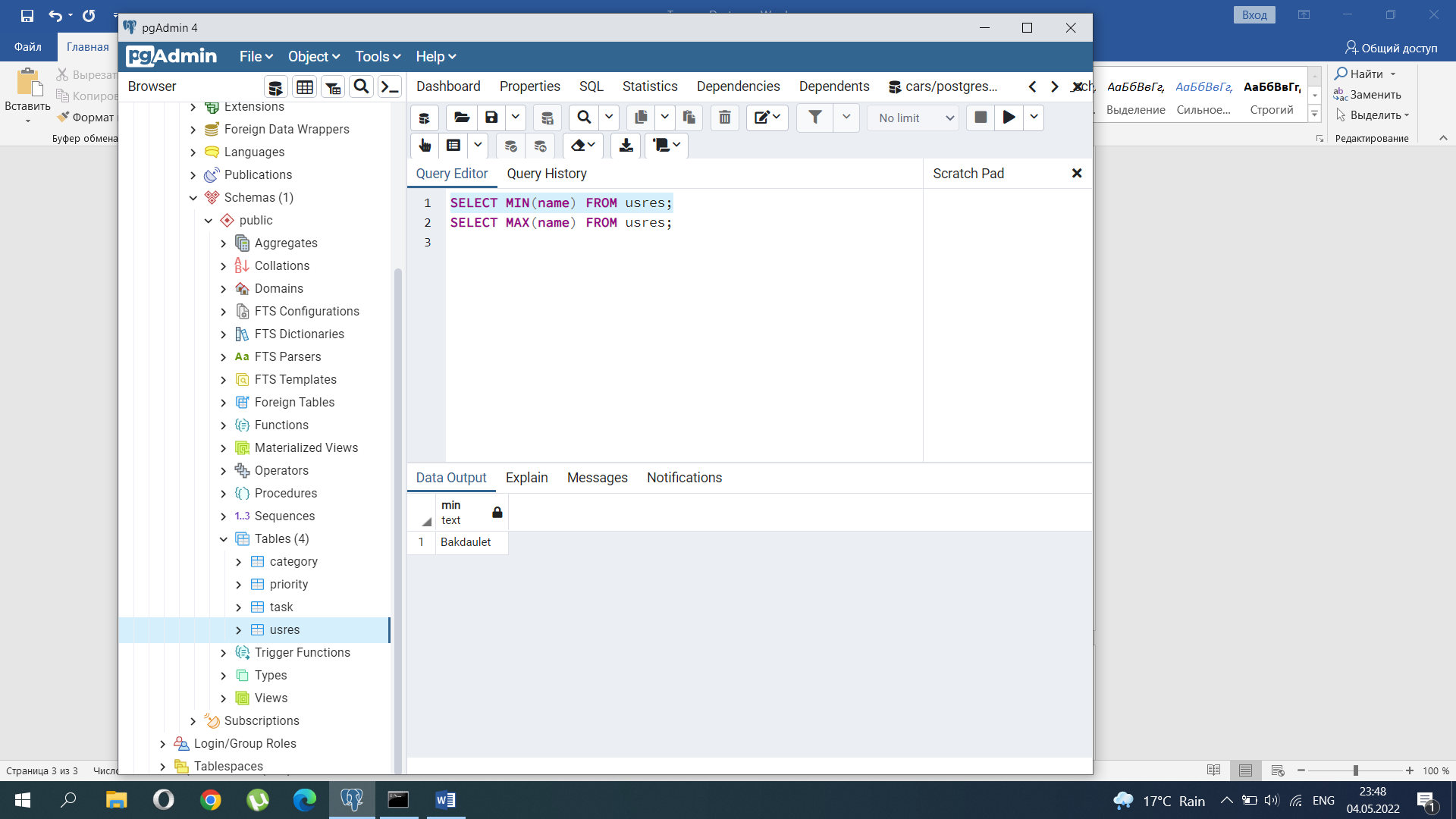
8. Можно ли объединить 3 (4, 5,<2)

9. Сделать запрос где есть:

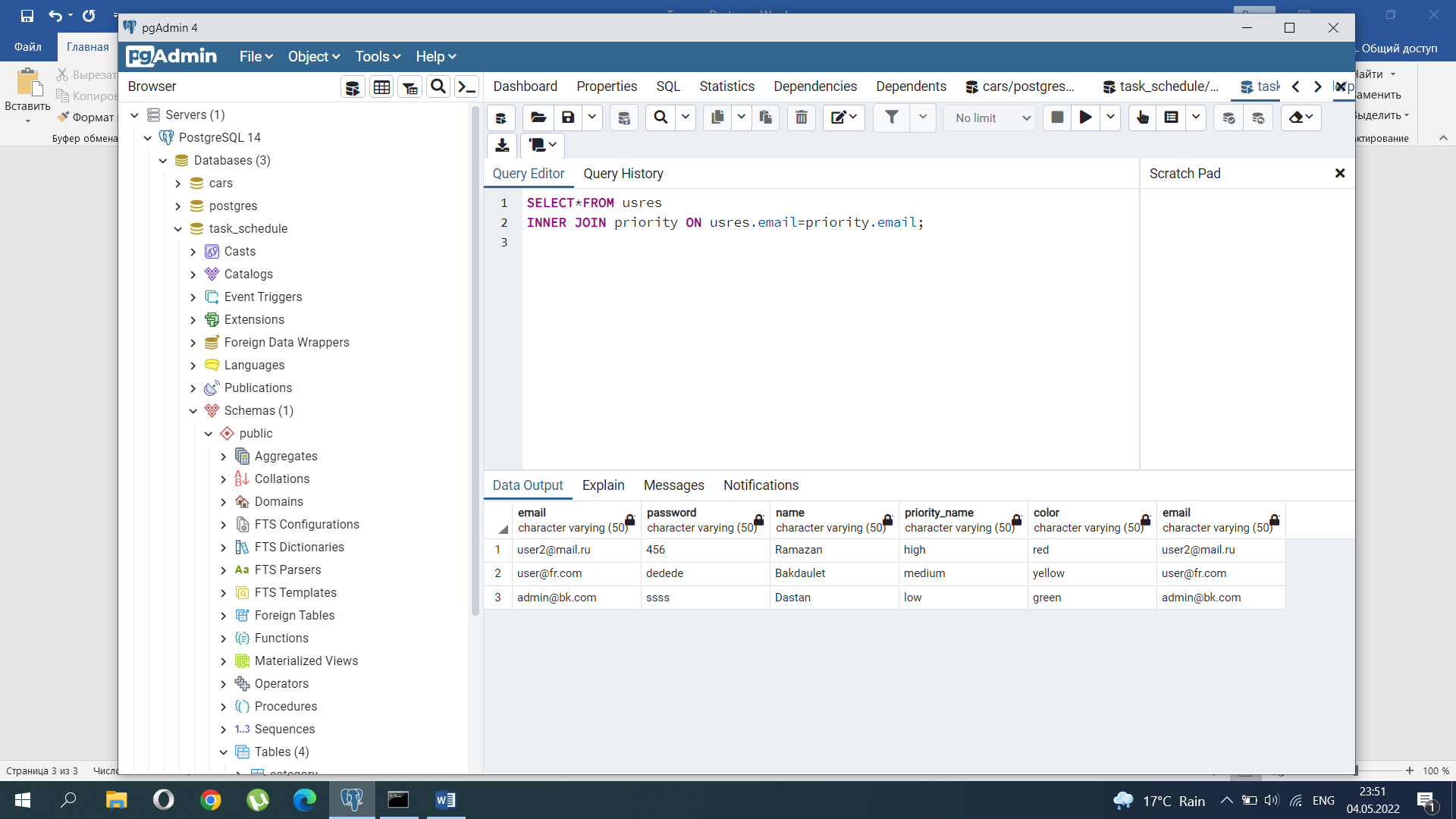
• группировка по нескольким полям



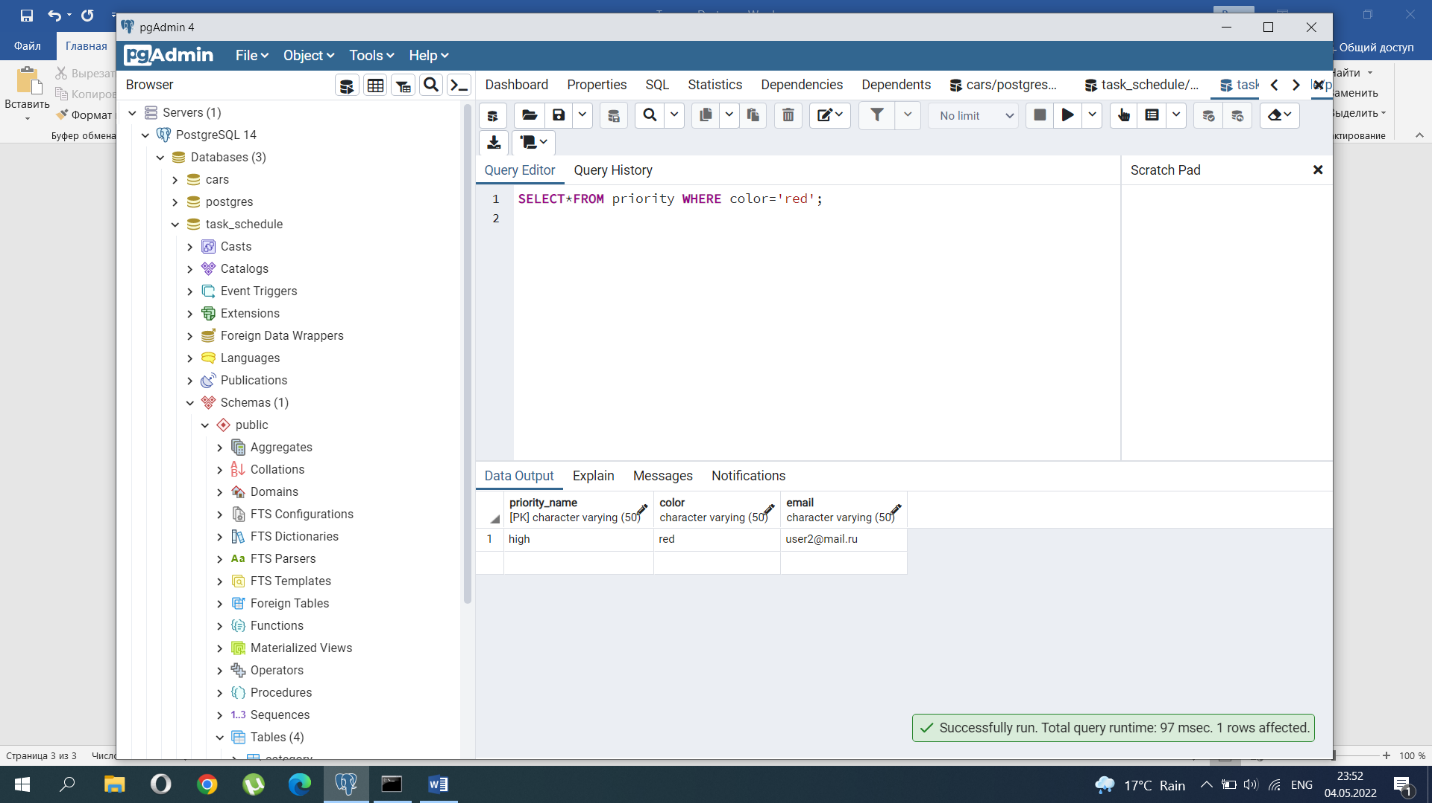
• выборка: функция агрегации + обычные поля



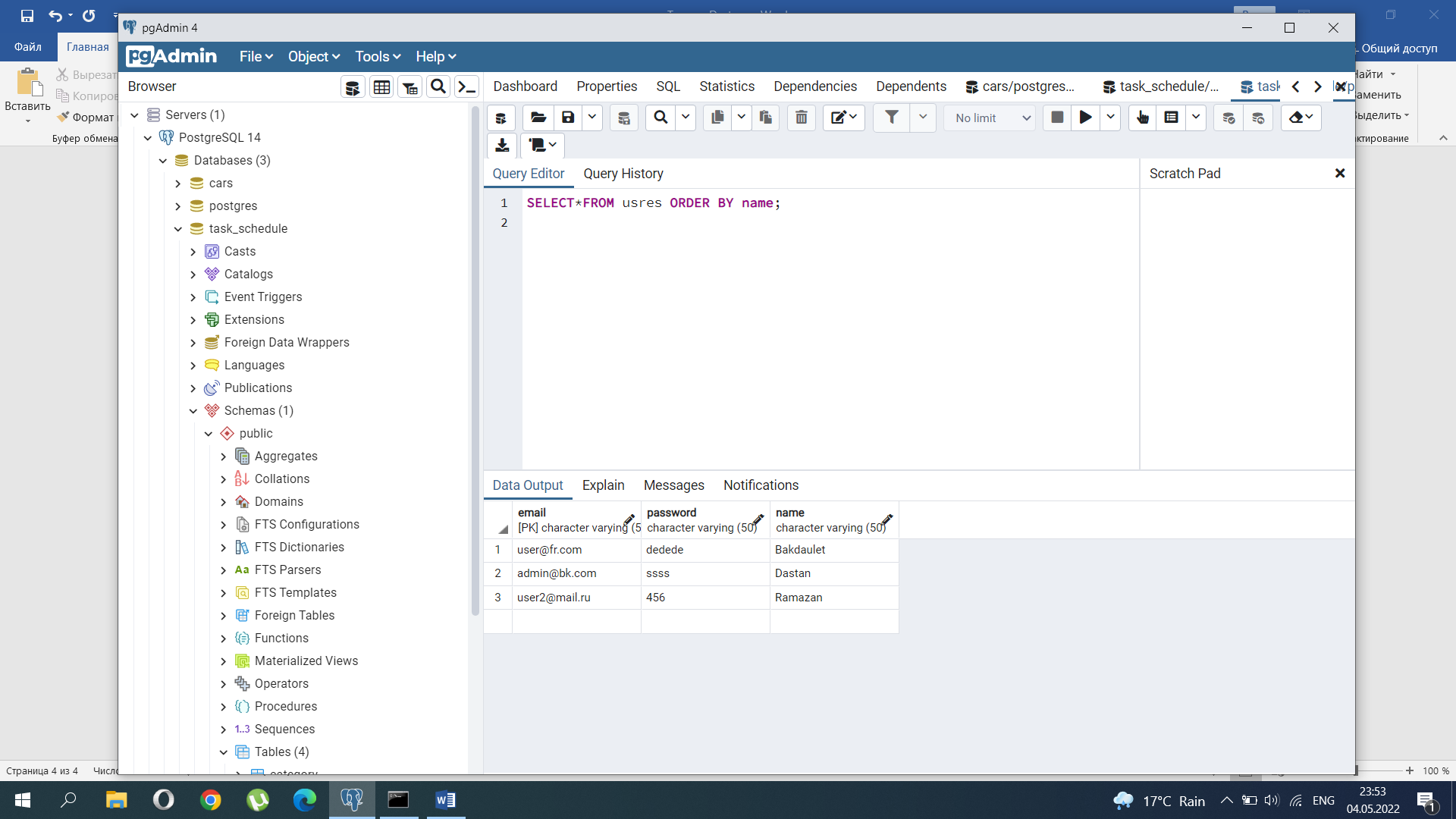
• join нескольких таблиц



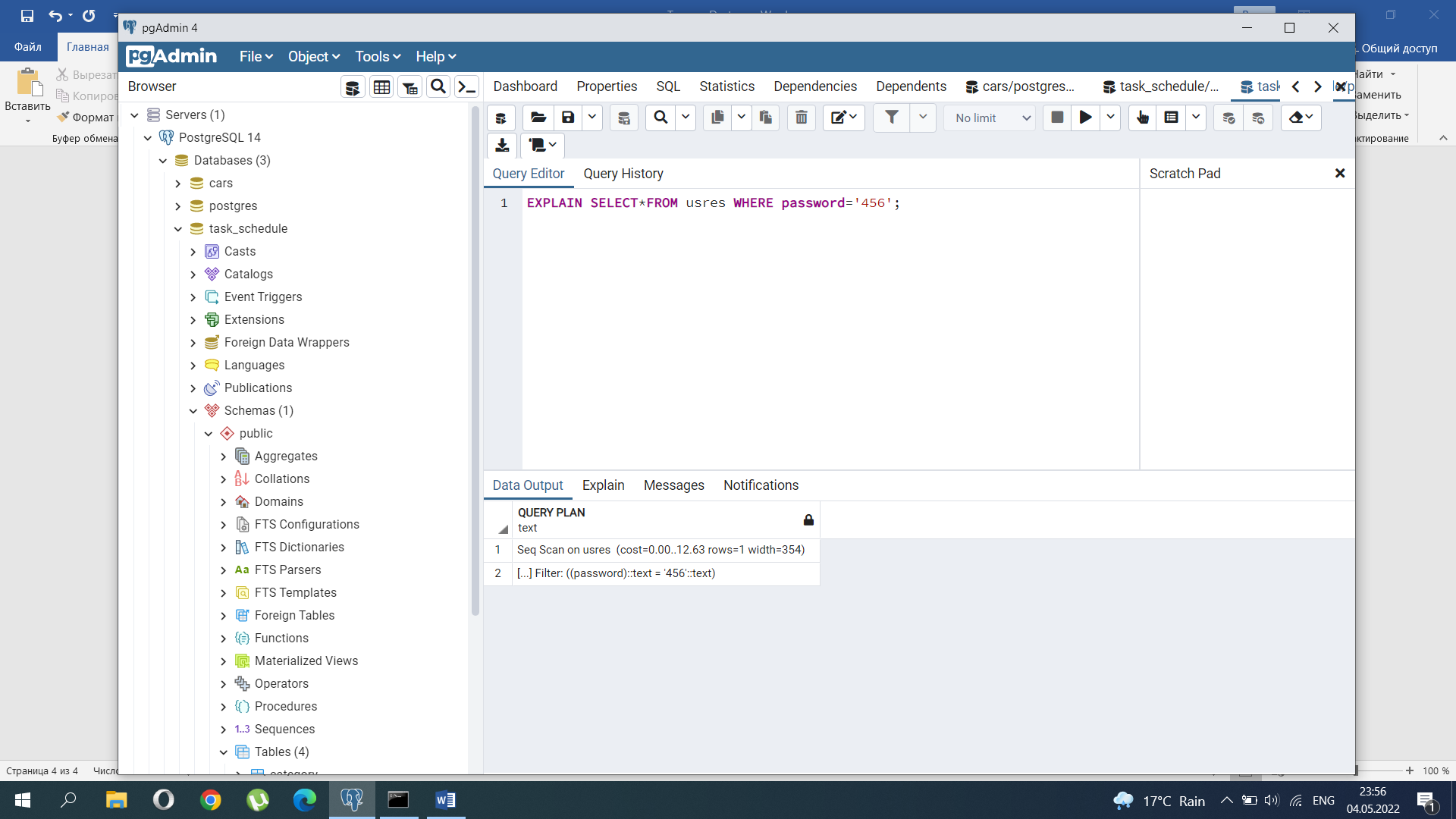
• условие where



• сортировка результата



10. Индексы создаются для тех полей, которые участвуют в условиях каких? (запросы)



11. Чтобы вручную обновить индексы для нужной таблицы, нужно выполнить команду

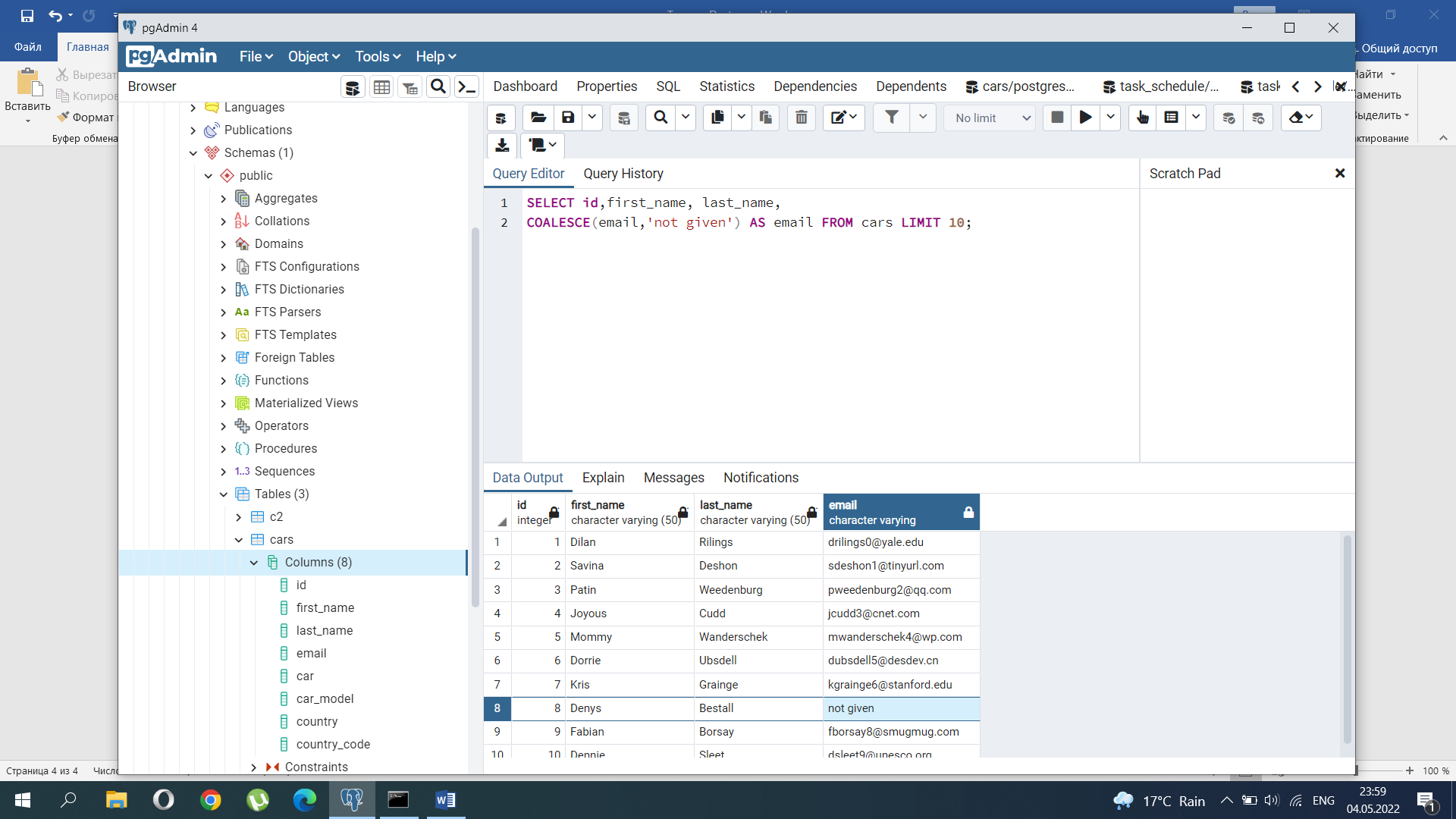
**ALTER TABLE tablename AUTO\_INCREMENT = 1**

12. Какая может быть причина(/ы) создания индексов в вашей БД?

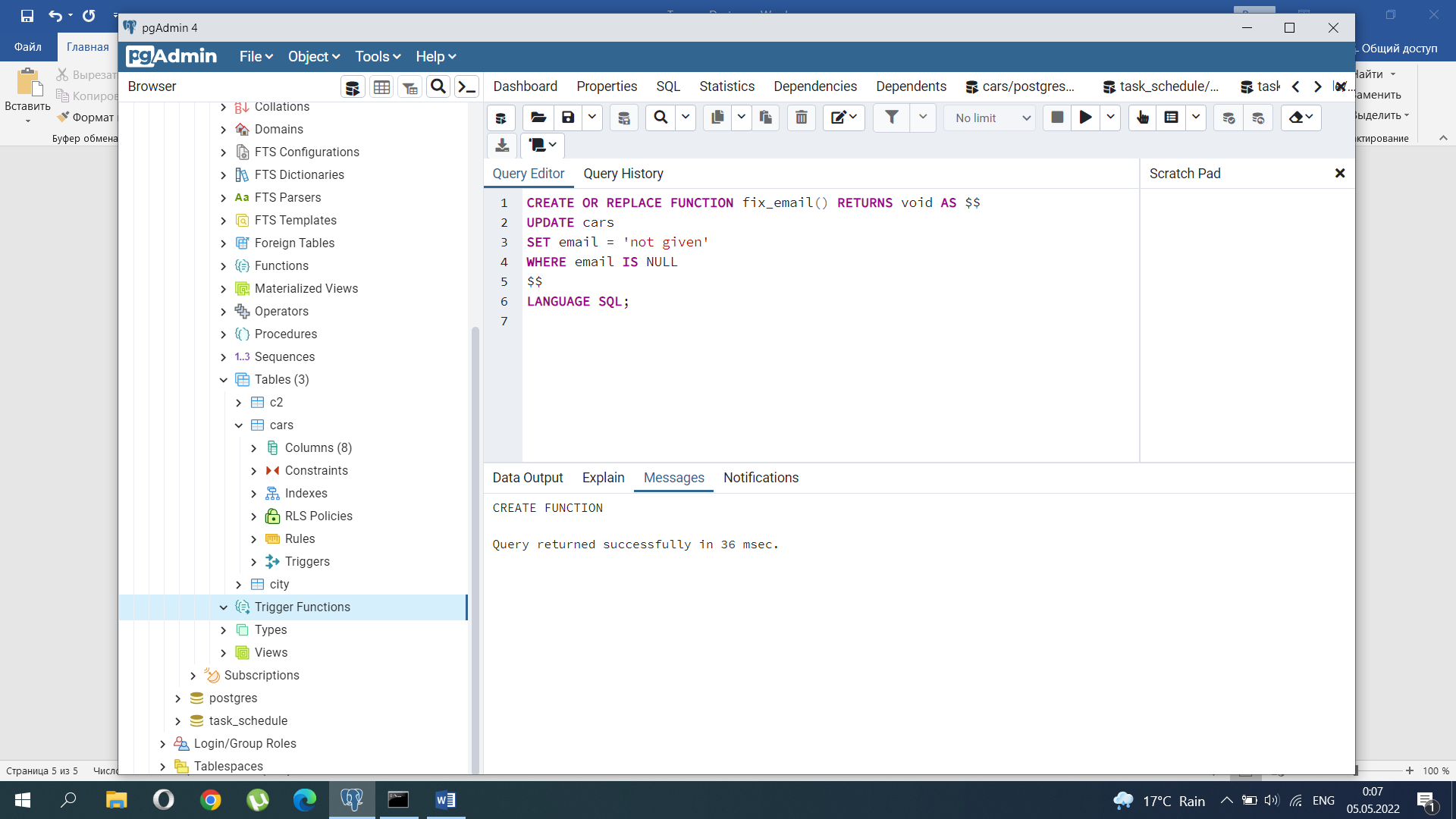
Нужно делать оптимизации производительности баз данных MySQL, создание индексов — один из лучших способов. Индексы SQL помогают повысить производительность базы данных, обеспечивая быстрый доступ к данным, хранящимся в базе данных.

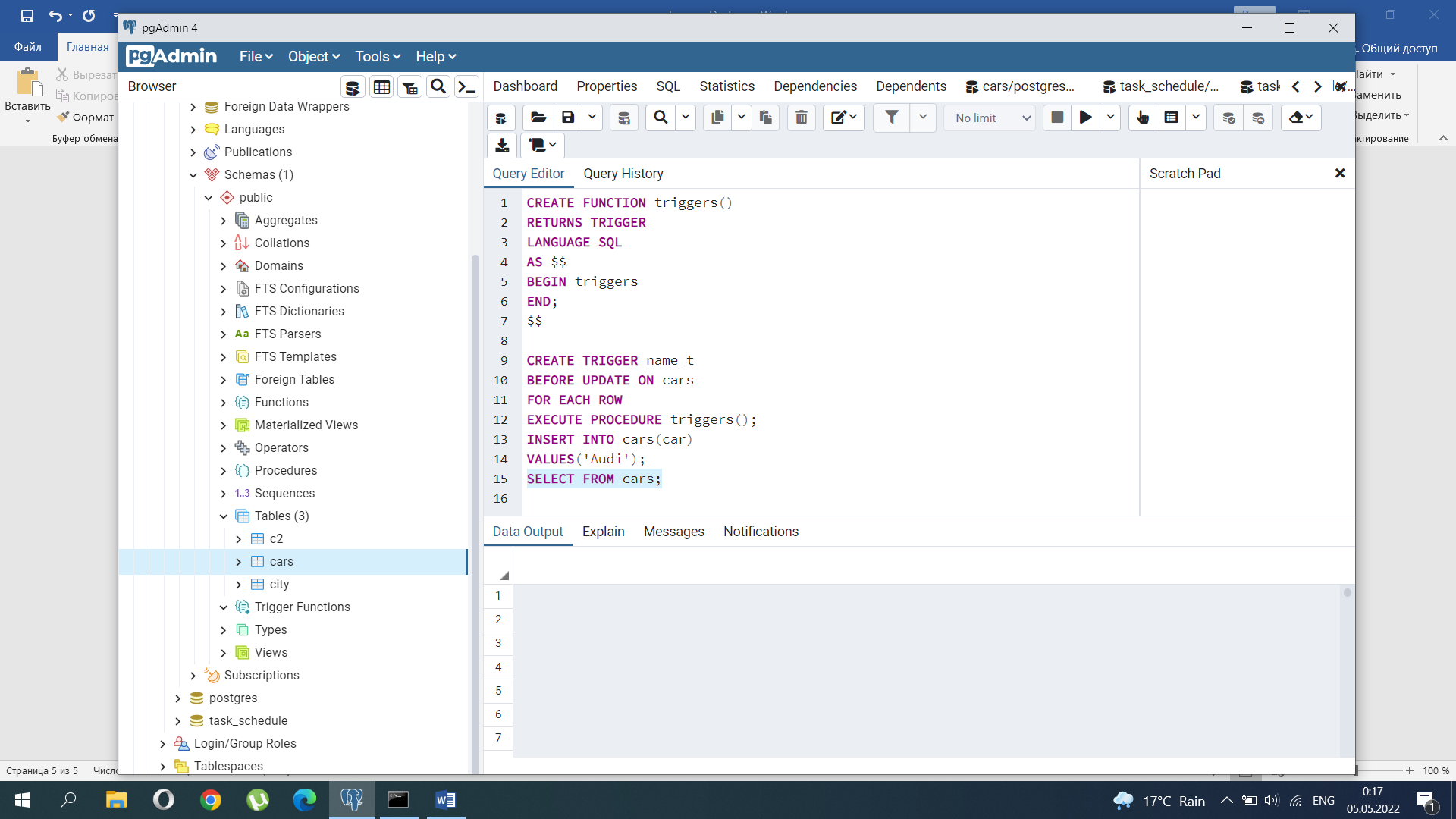
13. Что делает оператор сoalesce? Приведите пример запроса.

которая возвращает первое выражение из списка параметров, неравное NULL.



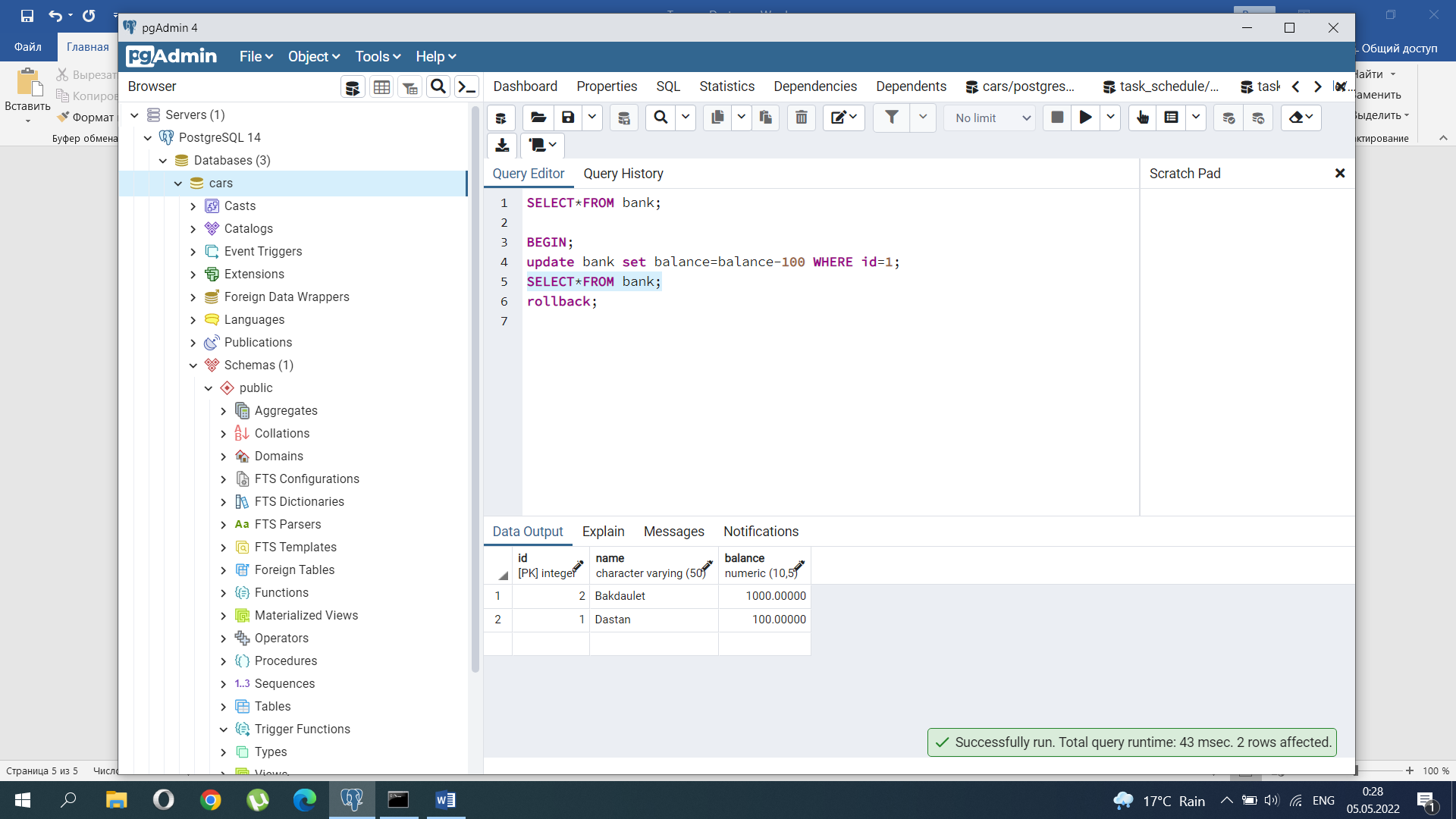
15. Напишите одну функцию и один триггер. Код и скрины БД вставить.





16. Напишите транзакцию и объясните в чем отличие от обычного запроса

Транзакция может содержать несколько запросов.  
Транзакция является атомарной, обычный запрос не обязательно



17 ACID- ACID (Atomicity - атомарность, consistency - консистентность, isolation - изолированность, durability - стойкость) это стандартный набор свойств, которые гарантируют, надежность транзакции. Atomicity - атомарность, Consistency - консистентность, Isolation - изолированность, Durability – стойкость