Министерство образования Новосибирской области ГБПОУ НСО «Новосибирский авиационный технический колледж имени Б.С.Галущака»

Практическая работа №9

Тема: Изучение PostgreSQL

Учебная дисциплина: МДК.11.01 Технология разработки и защиты баз данных

Работу выполнил:

студент группы ПР-21.102:

Воробьев П. А.

Проверил: Еремеева Л. А.

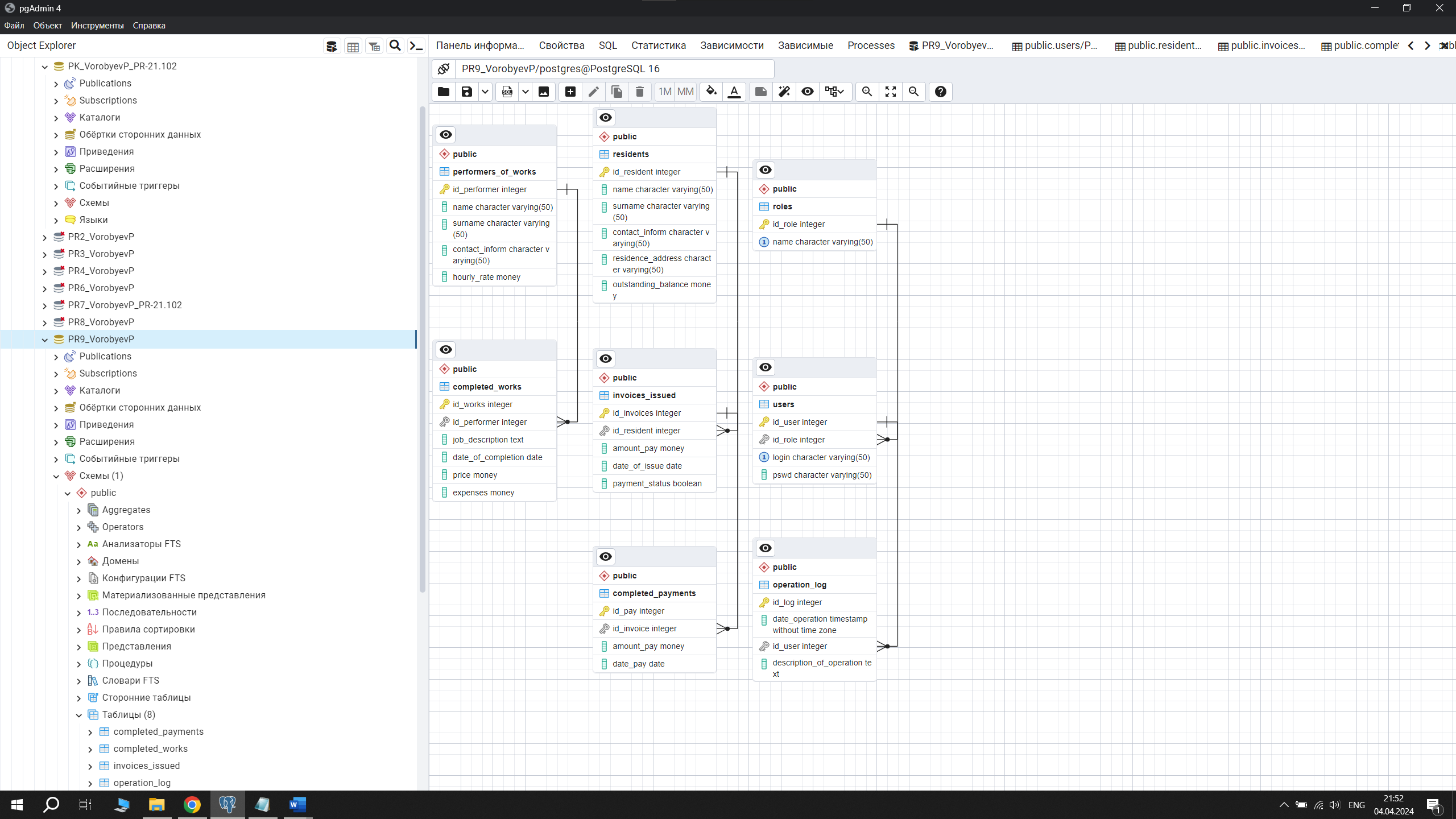
2024

Цель работы: изучить правила создания функций. Приобрести практические навыки создания функций в среде PostgreSQL. Научиться проектировать БД в целом на основе поставленного задания с использованием всех полученных ранее базовых знаний и навыков.

Задание. Общая часть: Разработать БД в соответствии с индивидуальным заданием. Создать функции, реализующие интерфейс для работы с [базой данных](https://pandia.org/text/category/bazi_dannih/). Проверить работоспособность функций путем выполнения этих функций с параметрами, обеспечивающими как успешное выполнение функции, так и невыполнение функции.

Вариант 8: База данных жилищной управляющей компании. Должна содержать следующие данные: информацию об исполнителях работ и [выполненных работах](https://pandia.org/text/category/vipolnenie_rabot/), жильцах, выставленных им счетах и выполненных ими платежах. Предусмотреть анализ следующих показателей: составить рейтинг злостных неплательщиков, рейтинг исполнителей работ с указанием их доли в статье расходов.

**Диаграмма базы данных:**



**Функции:**

**1. Добавление нового исполнителя работы:**

CREATE OR REPLACE FUNCTION add\_performer(

performer\_name VARCHAR(50),

performer\_surname VARCHAR(50),

performer\_contact VARCHAR(50),

performer\_hourly\_rate MONEY

) RETURNS VOID AS $$

BEGIN

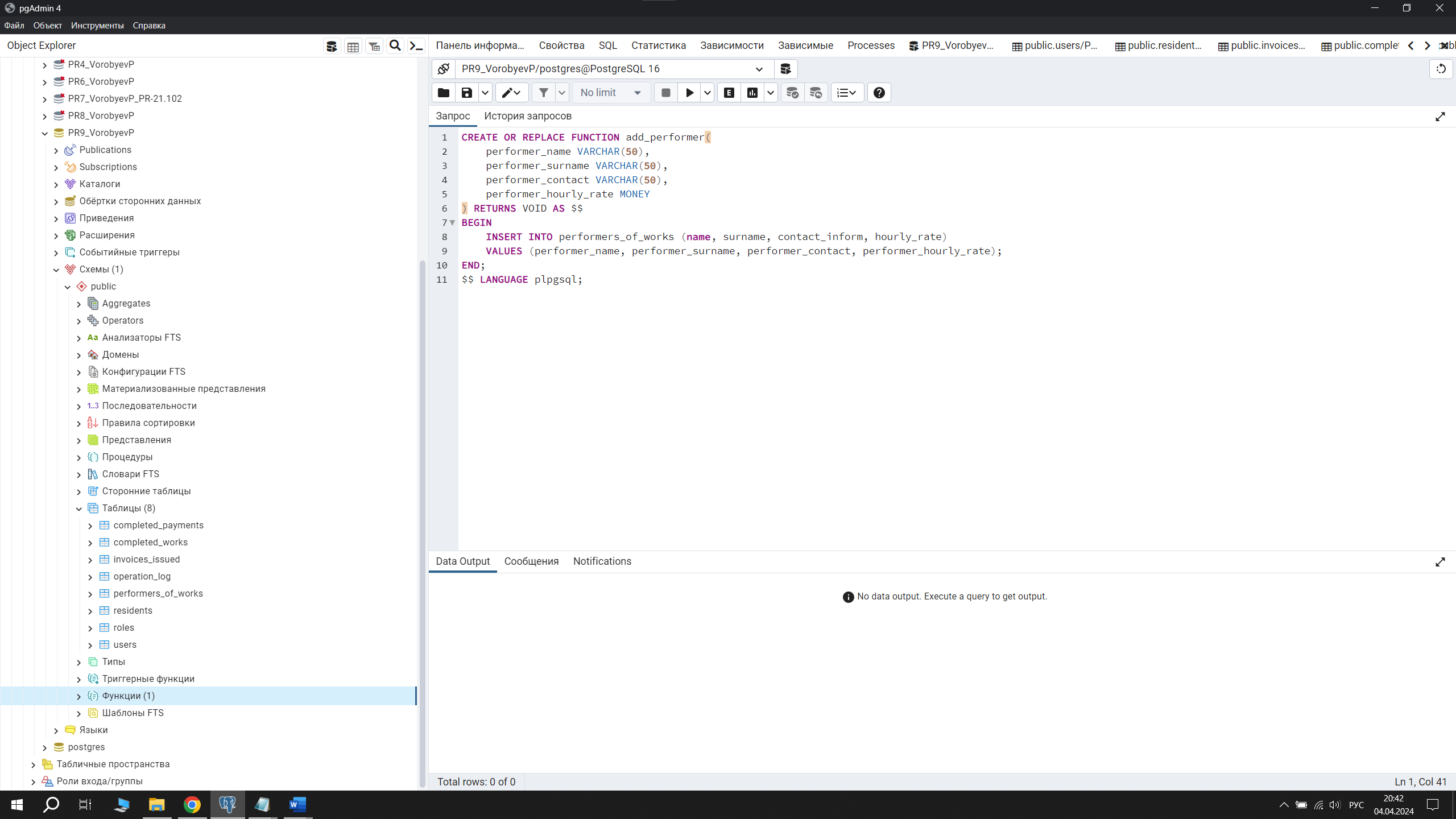
INSERT INTO performers\_of\_works (name, surname, contact\_inform, hourly\_rate)

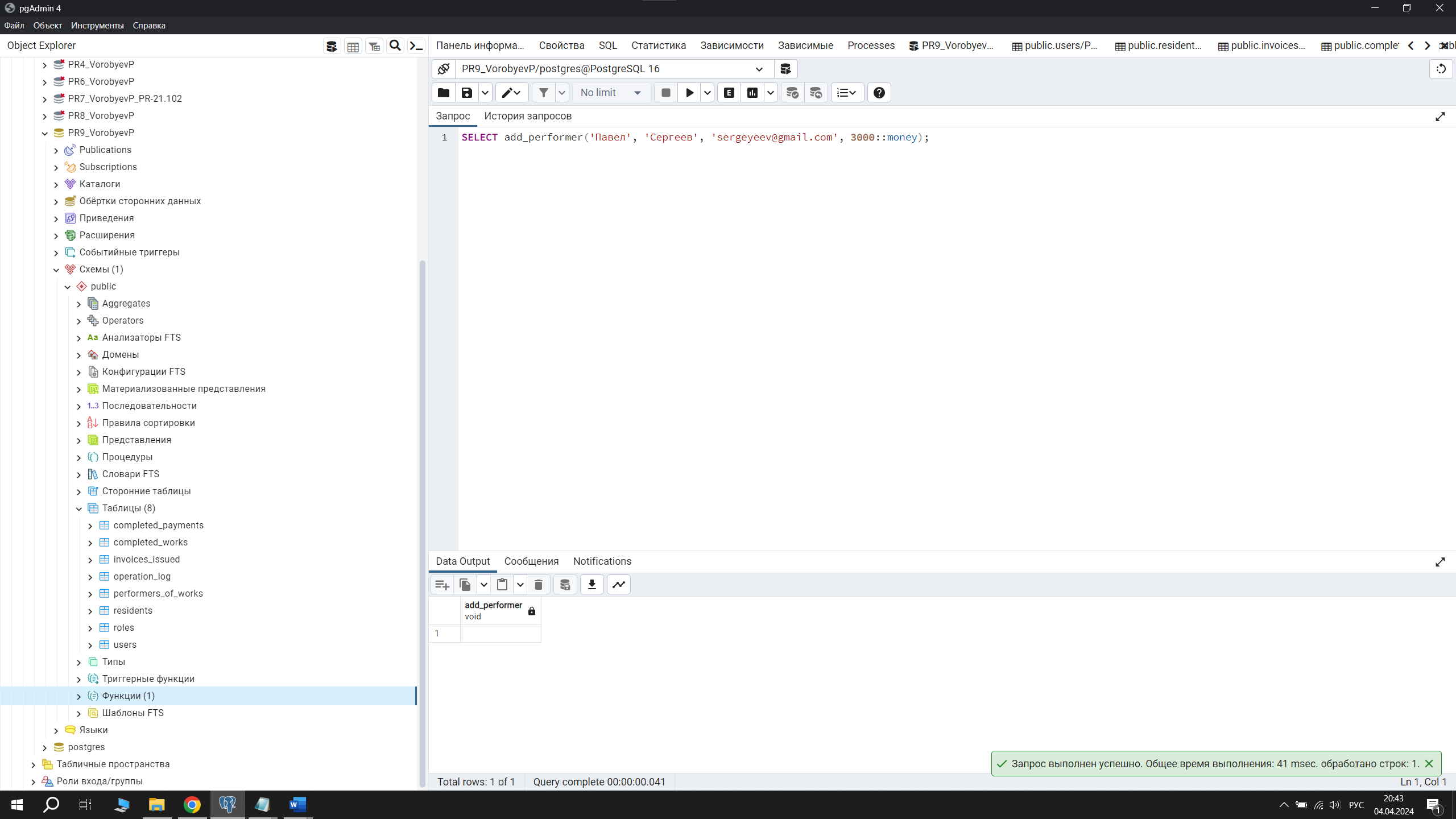
VALUES (performer\_name, performer\_surname, performer\_contact, performer\_hourly\_rate);

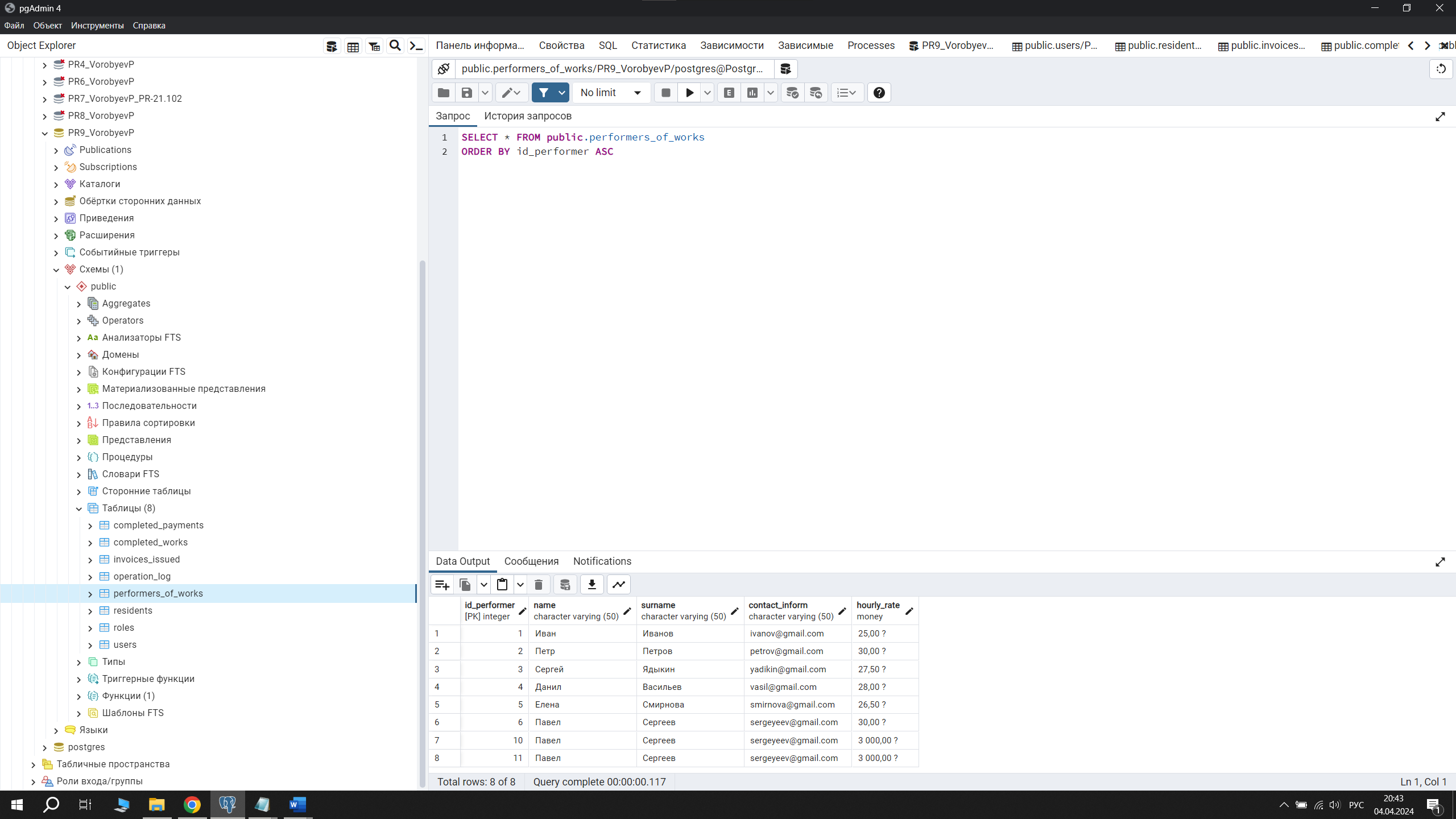
END;

$$ LANGUAGE plpgsql;

**Запрос для проверки:** SELECT add\_performer('Павел', 'Сергеев', 'sergeyeev@gmail.com', 3000::money);







**2. Добавление выполненной работы:**

CREATE OR REPLACE FUNCTION add\_completed\_work(

performer\_id INT,

description TEXT,

completion\_date DATE,

work\_price MONEY,

work\_expenses MONEY

) RETURNS VOID AS $$

BEGIN

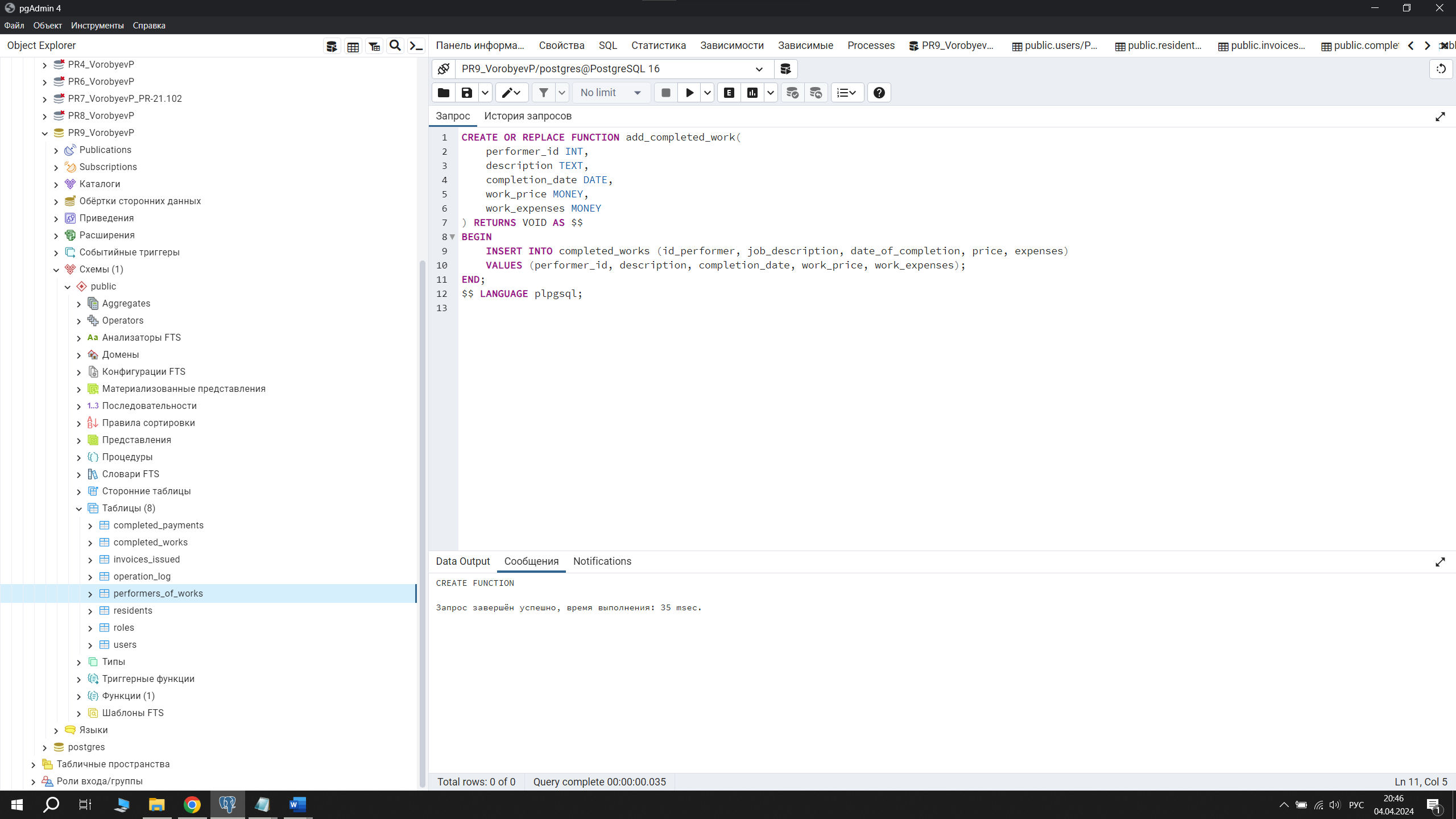
INSERT INTO completed\_works (id\_performer, job\_description, date\_of\_completion, price, expenses)

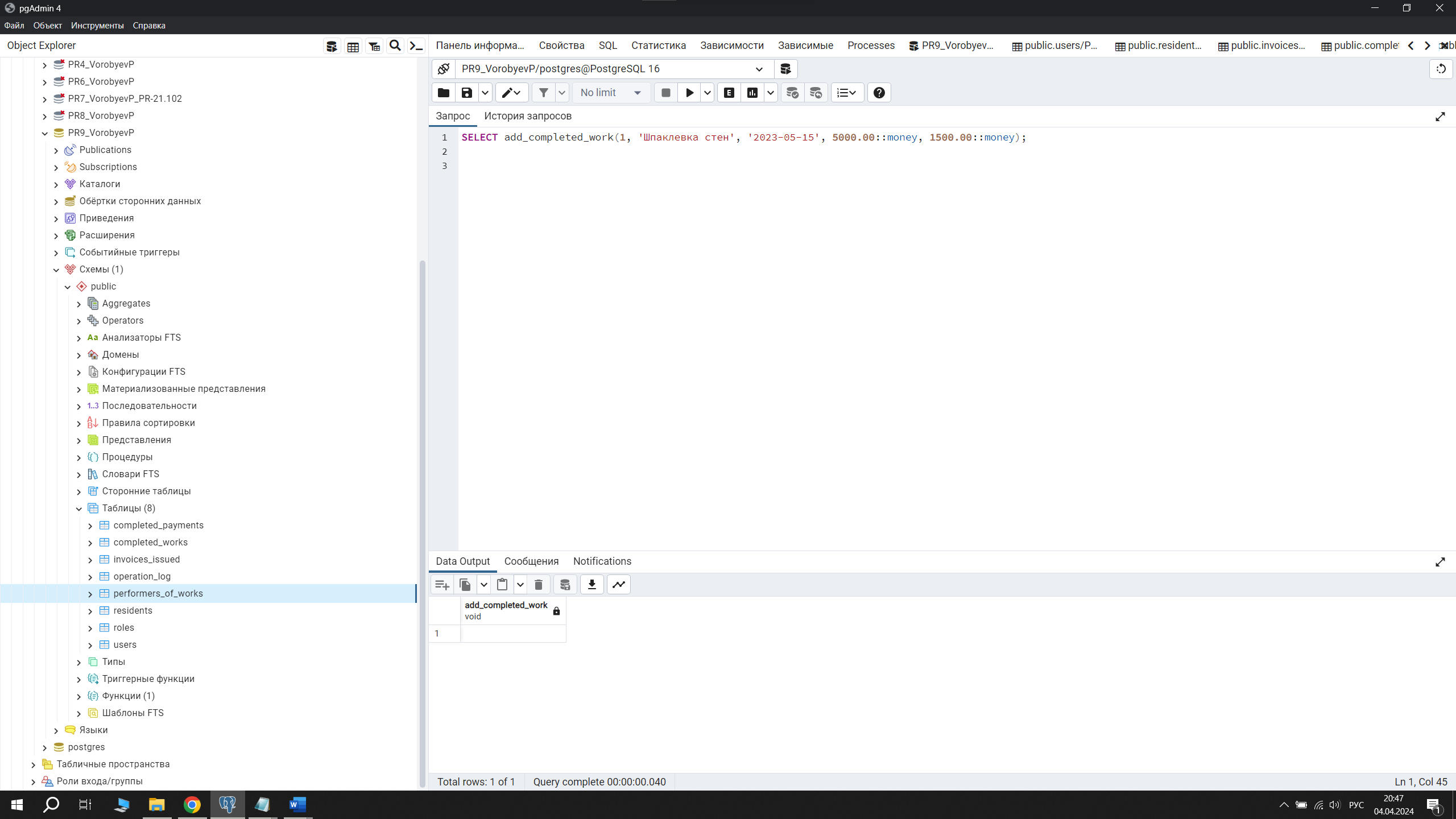
VALUES (performer\_id, description, completion\_date, work\_price, work\_expenses);

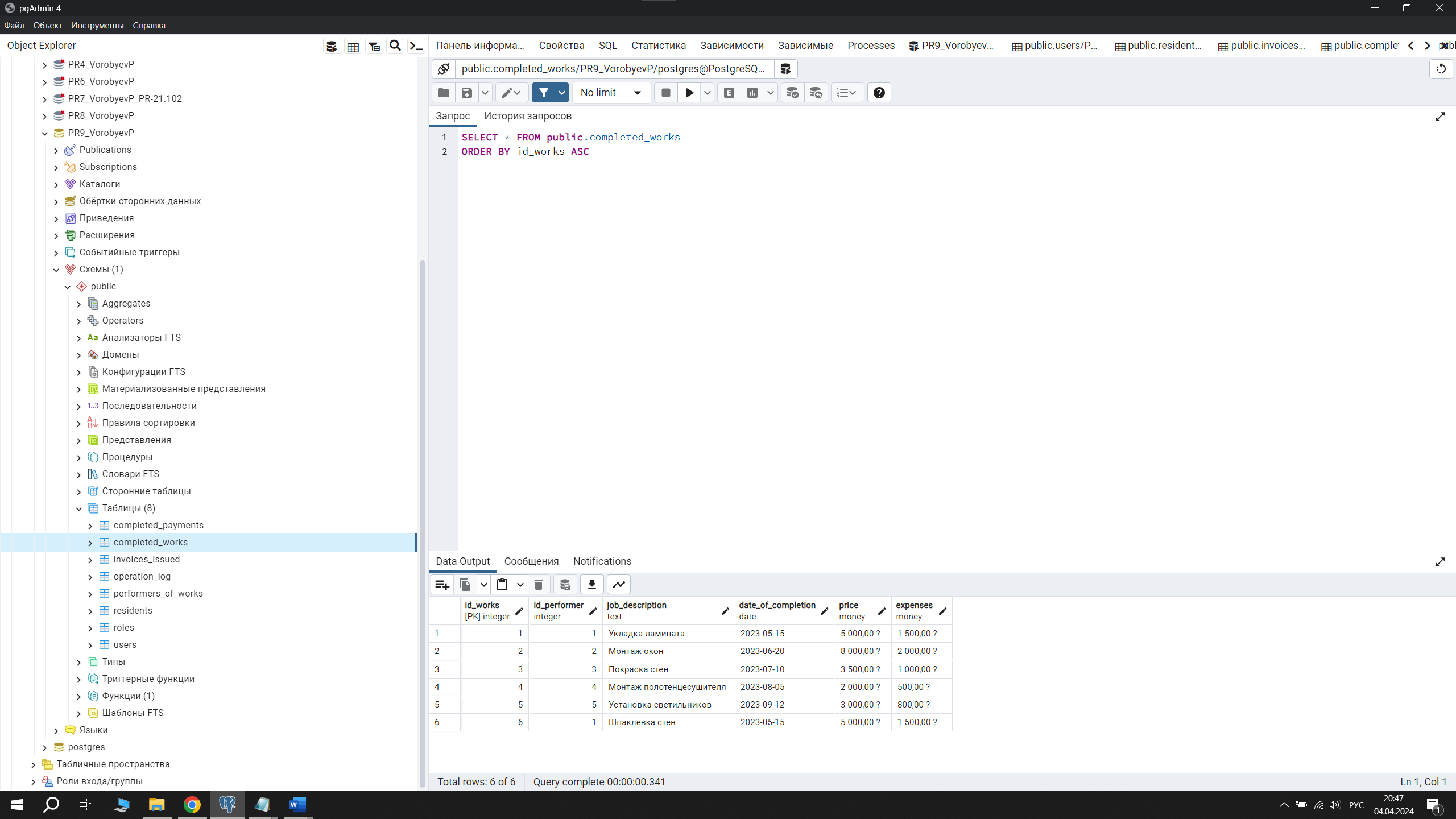
END;

$$ LANGUAGE plpgsql;

**Запрос для проверки:** SELECT add\_completed\_work(1, Шпаклевка стен', '2023-05-15', 5000.00::money, 1500.00::money);







**3. Добавление нового жильца:**

CREATE OR REPLACE FUNCTION add\_resident(

resident\_name VARCHAR(50),

resident\_surname VARCHAR(50),

resident\_contact VARCHAR(50),

resident\_address VARCHAR(50),

resident\_balance MONEY

) RETURNS VOID AS $$

BEGIN

INSERT INTO residents (name, surname, contact\_inform, residence\_address, outstanding\_balance)

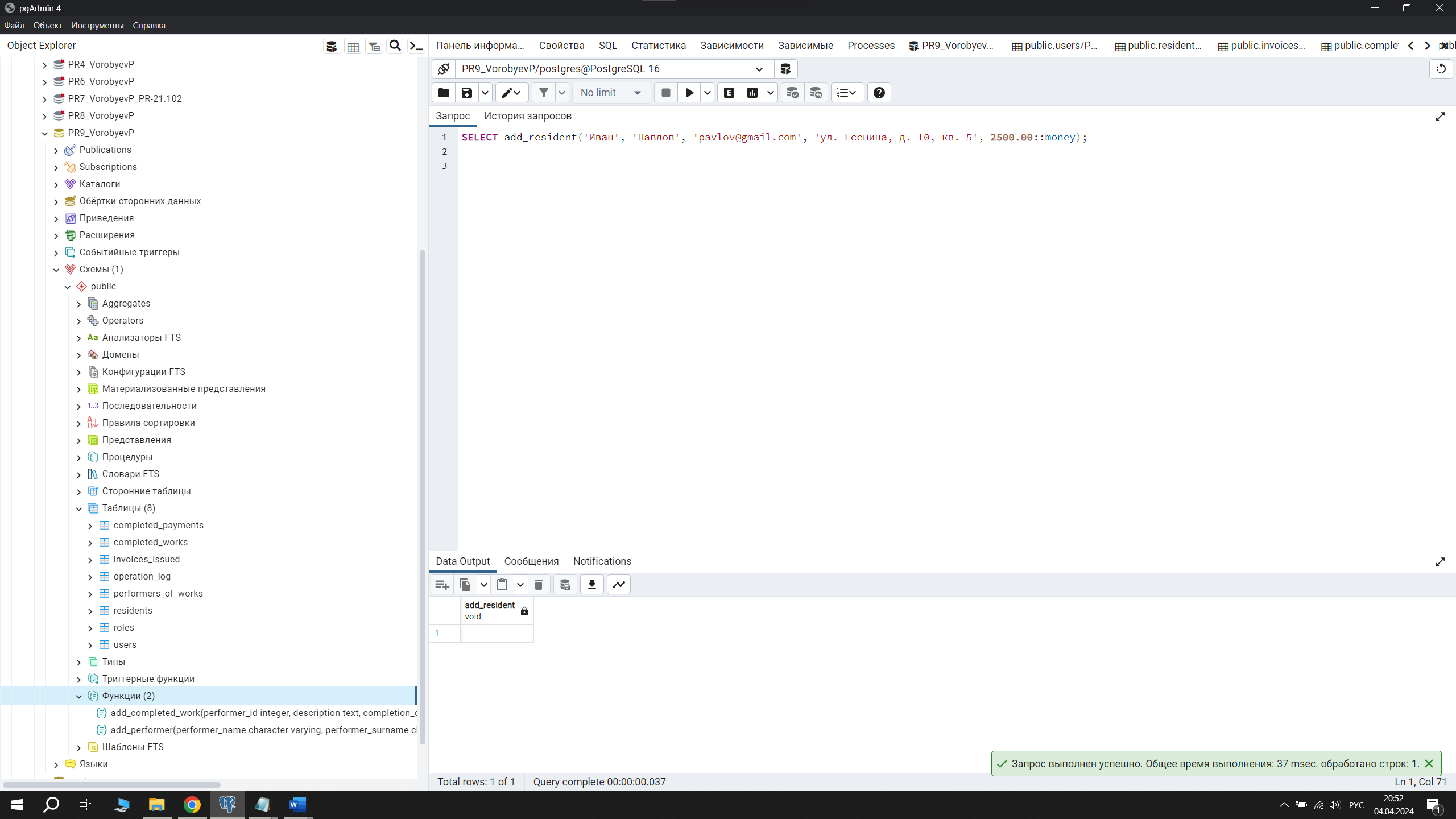
VALUES (resident\_name, resident\_surname, resident\_contact, resident\_address, resident\_balance);

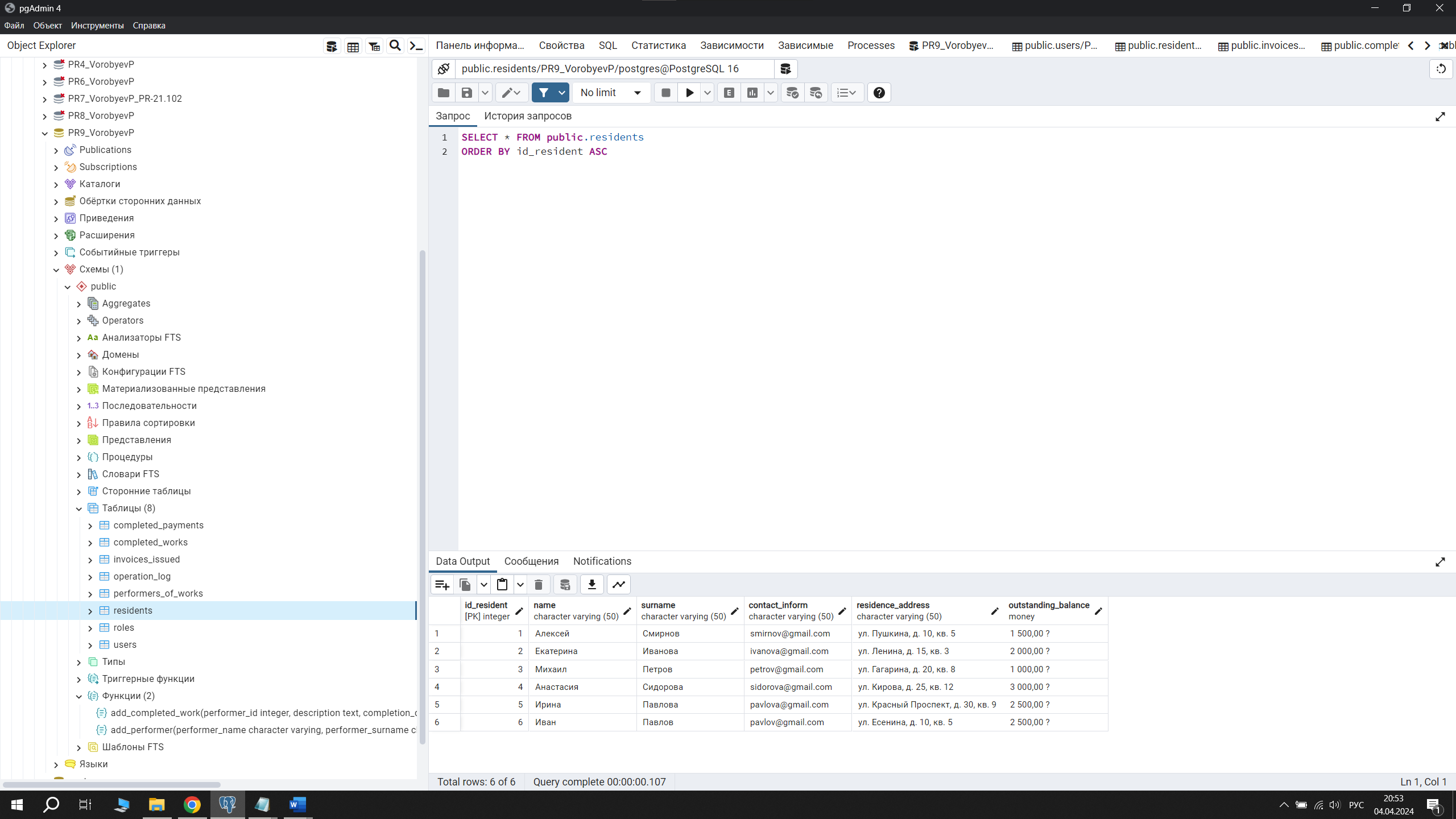
END;

$$ LANGUAGE plpgsql;

**Запрос для проверки:** SELECT add\_resident('Иван', 'Павлов', 'pavlov@gmail.com', 'ул. Есенина, д. 10, кв. 5', 2500.00::money);







**4. Выставление счета жильцу:**

CREATE OR REPLACE FUNCTION issue\_invoice(

id\_resident INT,

invoice\_amount MONEY,

issue\_date DATE,

payment\_status BOOLEAN

) RETURNS VOID AS $$

BEGIN

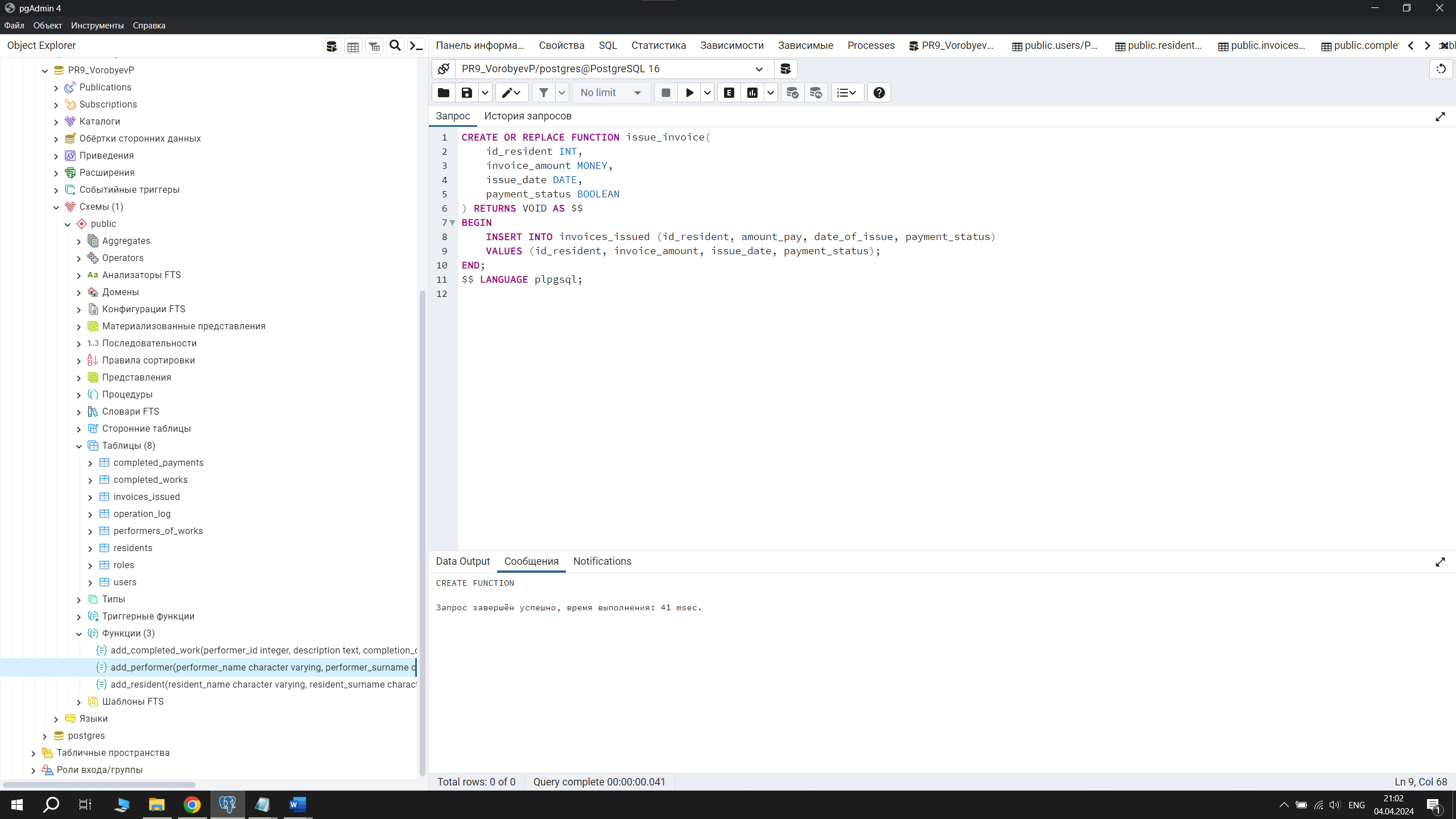
INSERT INTO invoices\_issued (id\_resident, amount\_pay, date\_of\_issue, payment\_status)

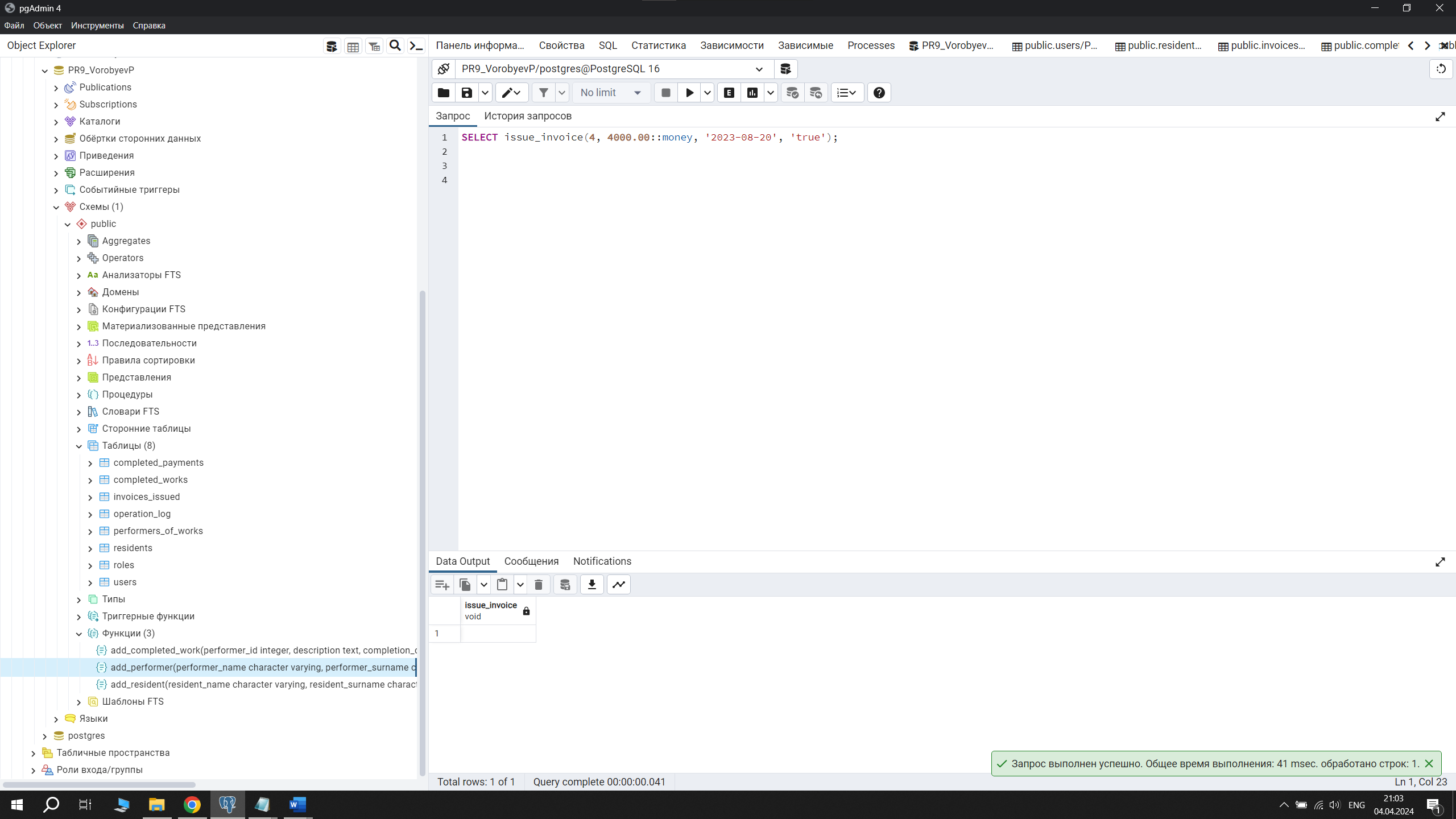
VALUES (id\_resident, invoice\_amount, issue\_date, payment\_status);

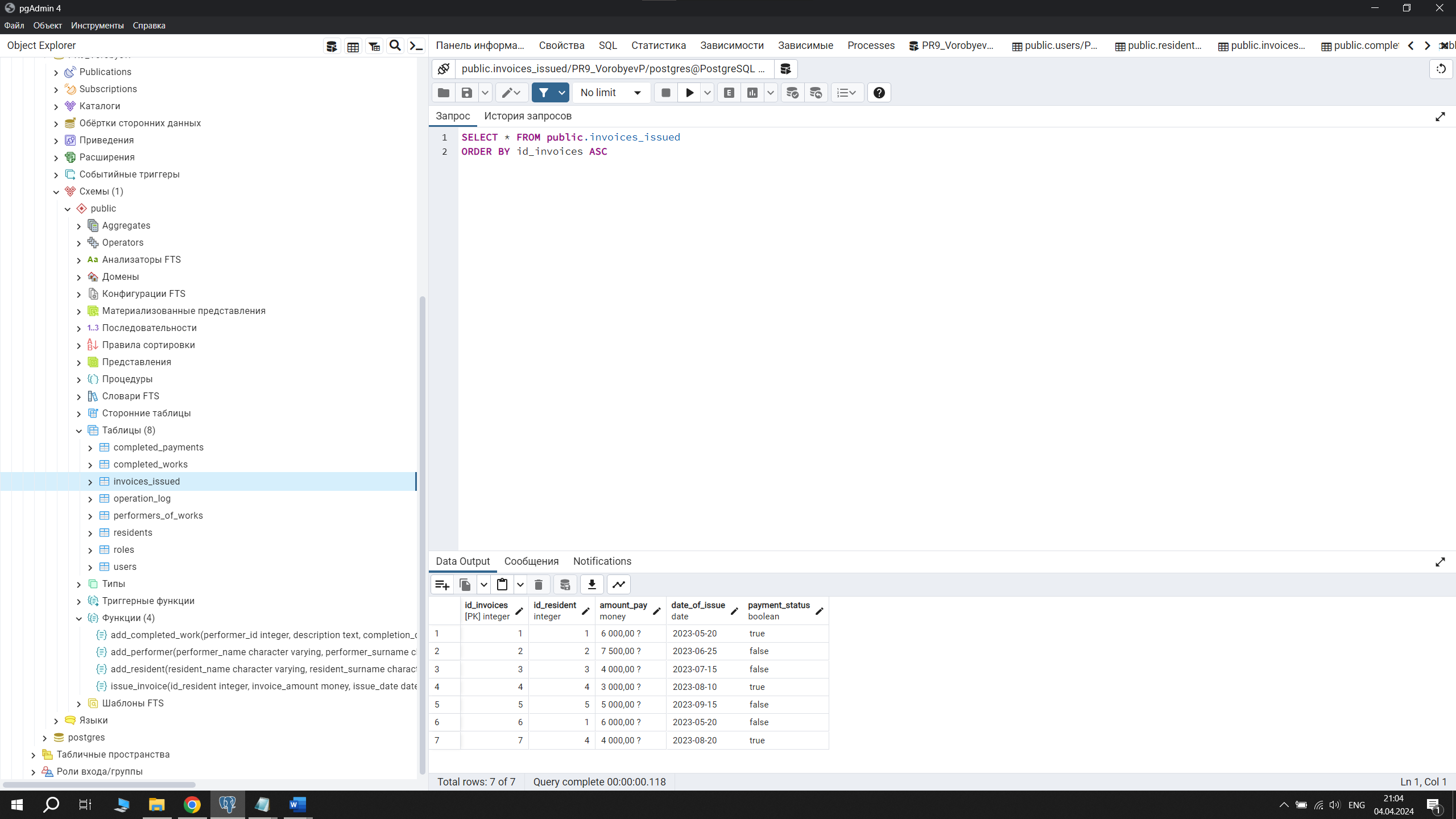
END;

$$ LANGUAGE plpgsql;

**Запрос для проверки:** SELECT issue\_invoice(4, 4000.00::money, '2023-08-20', 'true');







**5. Добавление выполненного платежа:**

CREATE OR REPLACE FUNCTION add\_payment(

id\_invoice INT,

payment\_amount MONEY,

payment\_date DATE

) RETURNS VOID AS $$

BEGIN

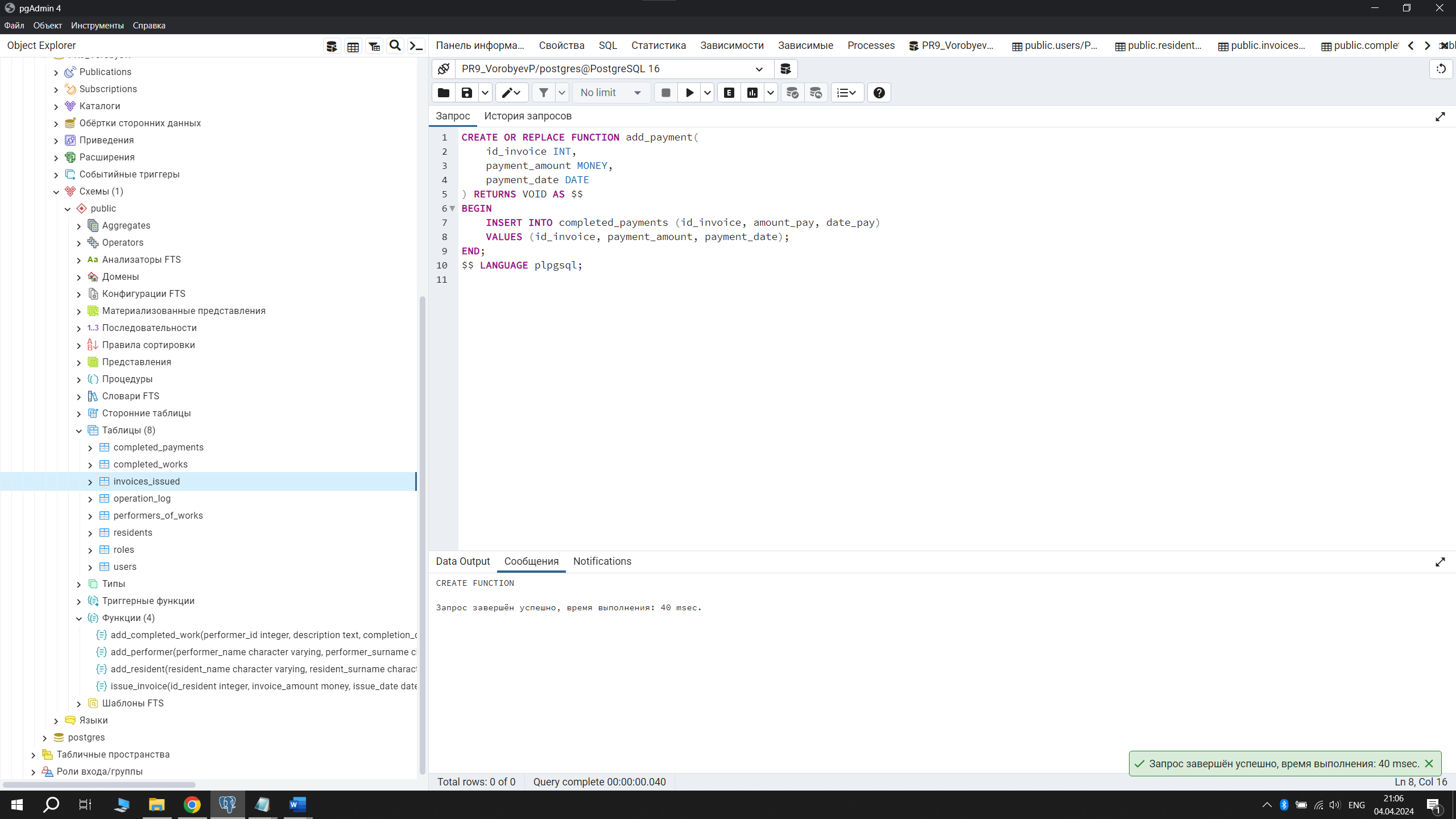
INSERT INTO completed\_payments (id\_invoice, amount\_pay, date\_pay)

VALUES (id\_invoice, payment\_amount, payment\_date);

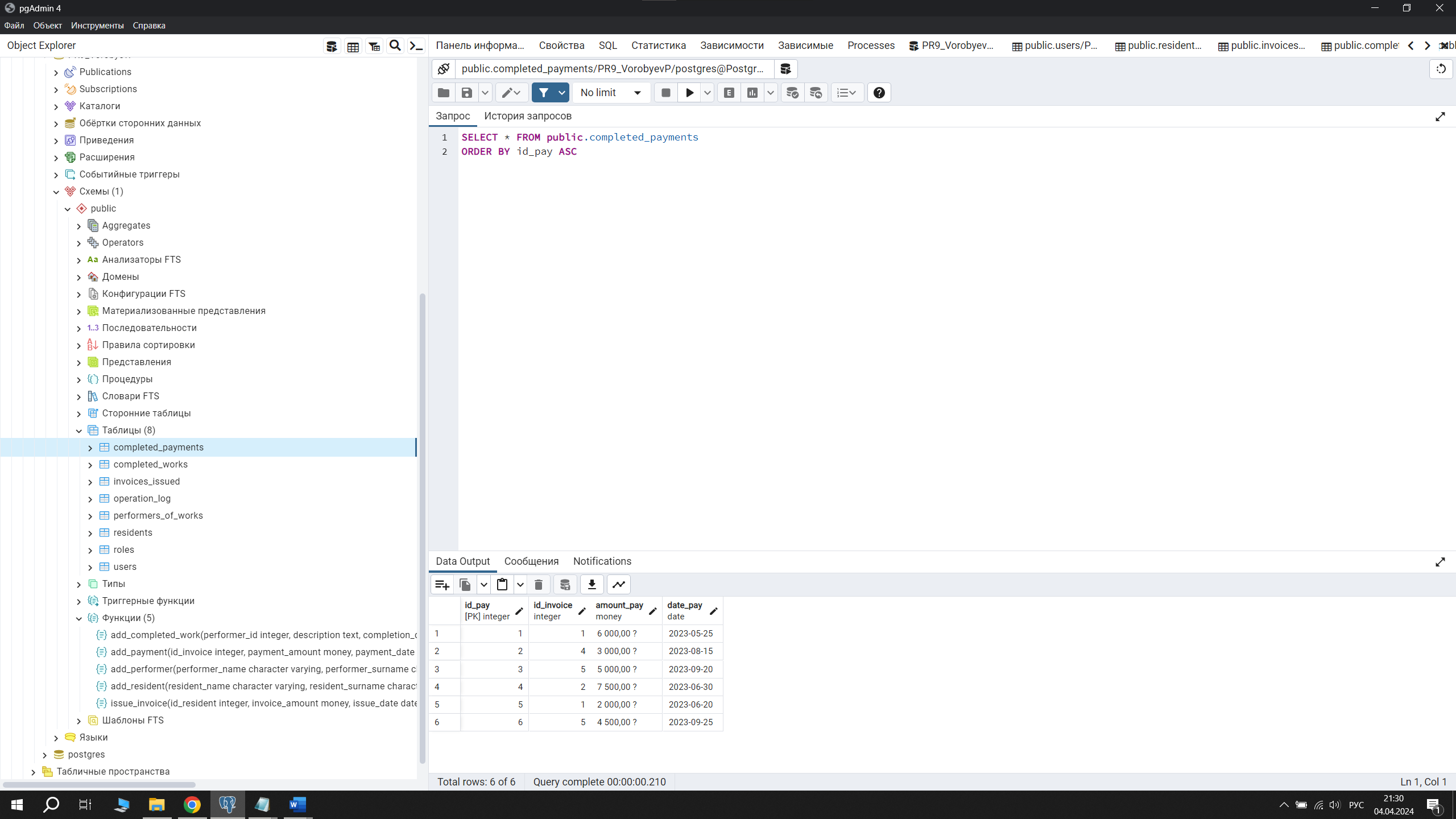
END;

$$ LANGUAGE plpgsql;

**Запрос для проверки:** SELECT add\_payment(5, 4500.00::money, '2023-09-25');







**Контрольные вопросы:**

**1. Что такое агрегатные функции?**

Агрегатные функции — это специальные функции PostgreSQL, которые позволяют обрабатывать группы строк и получать сводные значения.

Примеры агрегатных функций:

COUNT() - количество строк

SUM() - сумма

AVG() - среднее значение

MAX() - максимальное значение

MIN() - минимальное значение

**2. Как выполнить пользовательскую функцию?**

Существует два способа выполнить пользовательскую функцию:

Прямой вызов:

SELECT имя\_функции(параметры);

Используйте код с осторожностью.

Использование в запросе:

SELECT ...

FROM ...

WHERE ...

ORDER BY ...

GROUP BY ...

HAVING ...

Используйте код с осторожностью.

**3. Синтаксис команды создания функции**

CREATE FUNCTION имя\_функции(параметры) RETURNS тип\_возвращаемого\_значения AS $$

-- тело функции

$$ LANGUAGE язык;

**4. Как в теле функции связать переменную со значением входного параметра?**

В теле функции для связи переменной со значением входного параметра используется имя параметра.

**5. Синтаксис языка PL/pgSQL**

PL/pgSQL — это процедурный язык, разработанный специально для PostgreSQL.

Основные элементы PL/pgSQL:

Операторы: +, -, \*, /, =, >, <, >=, <=, !=, AND, OR, NOT

Ключевые слова: BEGIN, END, IF, THEN, ELSE, FOR, WHILE, LOOP, EXIT

Переменные: DECLARE имя\_переменной тип\_переменной;

Функции: CREATE FUNCTION имя\_функции(параметры) RETURNS тип\_возвращаемого\_значения AS $$ ... $$ LANGUAGE plpgsql;

Структуры управления: IF, THEN, ELSE, FOR, WHILE, LOOP, EXIT

**6. Возможно ли использование в теле функции запросов?**

Да, в теле функции можно использовать запросы.

Пример:

CREATE FUNCTION my\_function() RETURNS VARCHAR AS $$

DECLARE

result VARCHAR;

BEGIN

SELECT name INTO result FROM users WHERE id = 1;

-- ...

RETURN result;

END;

$$ LANGUAGE plpgsql;

**7. Как организовать условие в теле функции?**

Для организации условия в теле функции используются операторы IF, THEN, ELSE.

**8. Как создать цикл в теле функции?**

Для создания цикла в теле функции используются операторы FOR, WHILE, LOOP.

**9. Используются ли ограничения, наложенные на таблицу, при добавлении в нее записей через созданную функцию?**

Да, ограничения, наложенные на таблицу, будут использоваться при добавлении в нее записей через созданную функцию.

**10. Можно ли использовать созданные функции за пределами базы данных, в которой они созданы? Почему?**

Нет, созданные функции нельзя использовать за пределами базы данных, в которой они созданы. Функции PostgreSQL являются локальными объектами и не могут быть экспортированы.

Причины:

Функции могут использовать специфичные для PostgreSQL типы данных и функции.

Функции могут быть зависимы от других объектов базы данных.

Для использования функций в другой базе данных их необходимо создать в этой базе данных.