

***Федеральное агентство по рыболовству***

***Федеральное государственное бюджетное образовательное***

***учреждение высшего образования***

***«Астраханский государственный технический университет»***

**Система менеджмента качества в области образования, воспитания, науки и инноваций сертифицирована**

**ООО «ДКС РУС» по международному стандарту ISO 9001:2015**

|  |  |
| --- | --- |
| Институт | Информационных технологий и коммуникаций |
| Направление | 09.03.01 Информатика и вычислительная техника |
| Профиль | «Автоматизированные системы обработки информации и управления» |
| Кафедра | «Автоматизированные системы обработки информации и управления» |

**Лабораторная работа № 13**

**«Пользовательские функции в PostgreSQL. Часть 1.**»

по дисциплине «СУБД PostgreSQL»

|  |  |  |
| --- | --- | --- |
|  |  | Работа выполнена студентом группы ДИНРБ-31  Кузургалиев Р.А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Фамилия И.О.) подпись |
|  |  | Проверил работу:  ст. преподаватель Мамлеева А.Р,\_\_\_\_\_\_\_\_\_\_\_\_  (ученая степень, ученое звание, Фамилия И.О.) |

Работа защищена

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

**АСТРАХАНЬ – 2024**

**СОДЕРЖАНИЕ ОТЧЕТА**

*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.

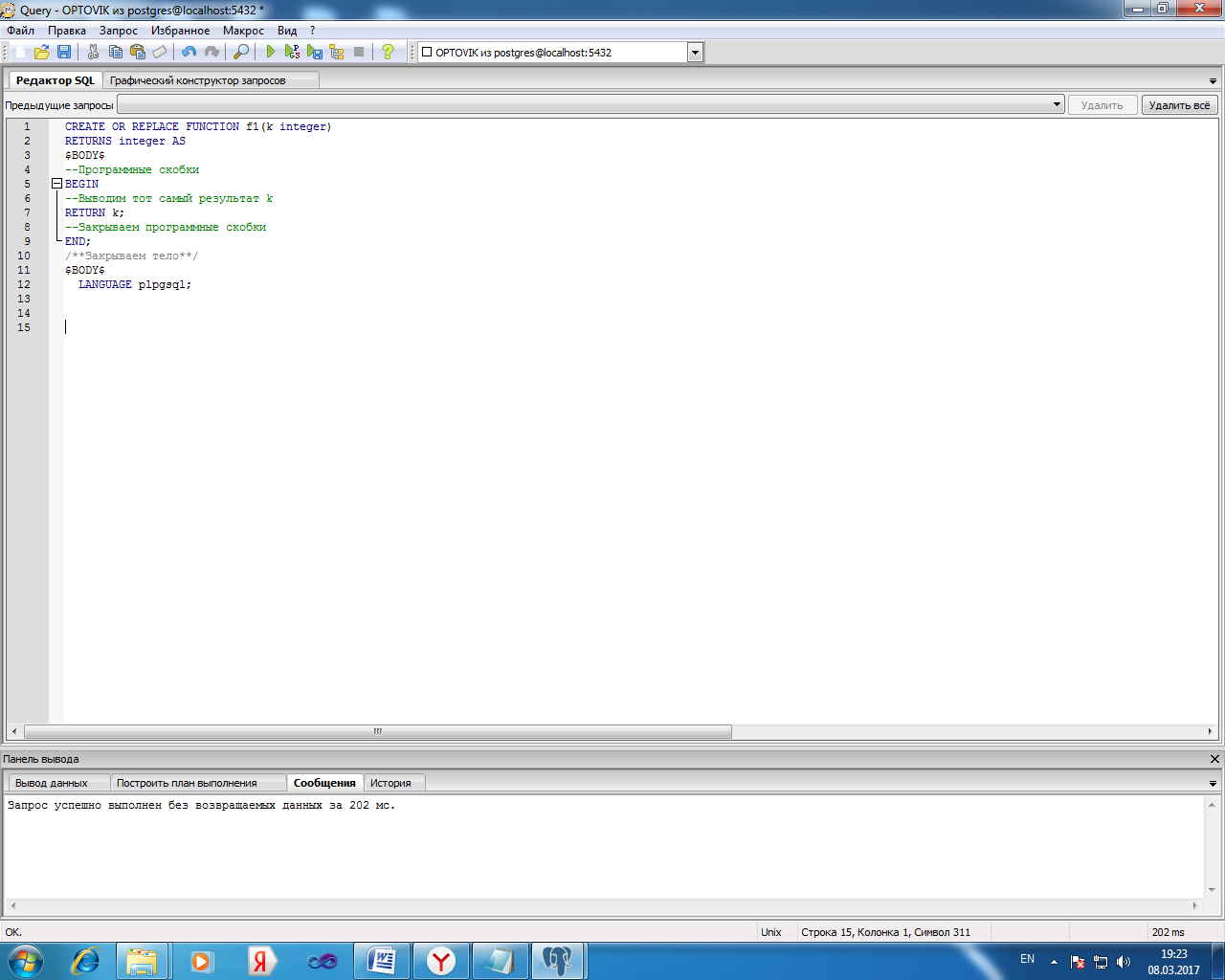
**Часть 1. Знакомство с функцией.**

PostgreSQL достаточно серьёзно отличается от других языков запросов SQL. Язык SQL в PostgreSQL является типизированным. Подход PostgreSQL похож на языки программирования (Pascal, C/C++, Basic) и на HTML.

В PostgreSQL **нет хранимых процедур**.  Вместо них есть **хранимые функции**.

Функции могут не иметь выходных данных.Каждая функция может выводить только определённый тип данных.

***Пример № 1***:



В данном примере описана простая функция, которая вводит и выводит целое число.

Рассмотрим подробнее каждую часть кода

.

**1 CREATE OR REPLACE FUNCTION f1(k integer)**

Конструкция CREATE OR REPLACE FUNCTION создаёт или изменяет функцию, если она создана. Можно указывать схему. В примере схема по умолчанию - PUBLIC. f1 - название функции. Переменная k целочисленного типа integer является входными данными.

**2 RETURNS integer AS**

Эта конструкция указывает, что на выходе будут данные в виде целочисленного числа.

**3 $BODY$**

Далее следует тело функции (по аналогии с HTML).

**4 Комментарий**. Однострочный начинается с двух знаков минус .

**5 BEGIN**

Программные скобки являются словесными синтаксическими конструкциями.

**7 RETURN k;**

Возврат целочисленного значения.

**9 END;**

Закрытие программных скобок.

**11 $BODY$**

Конец тела функции.

**12 LANGUAGE plpgsql;**

PostgreSQL - это мощная СУБД, писать запросы под которую можно на разных языках. Например, доступен TCL. Чтобы сервер понял, на каком именно языке запросов написана инструкция, необходимо это указать.

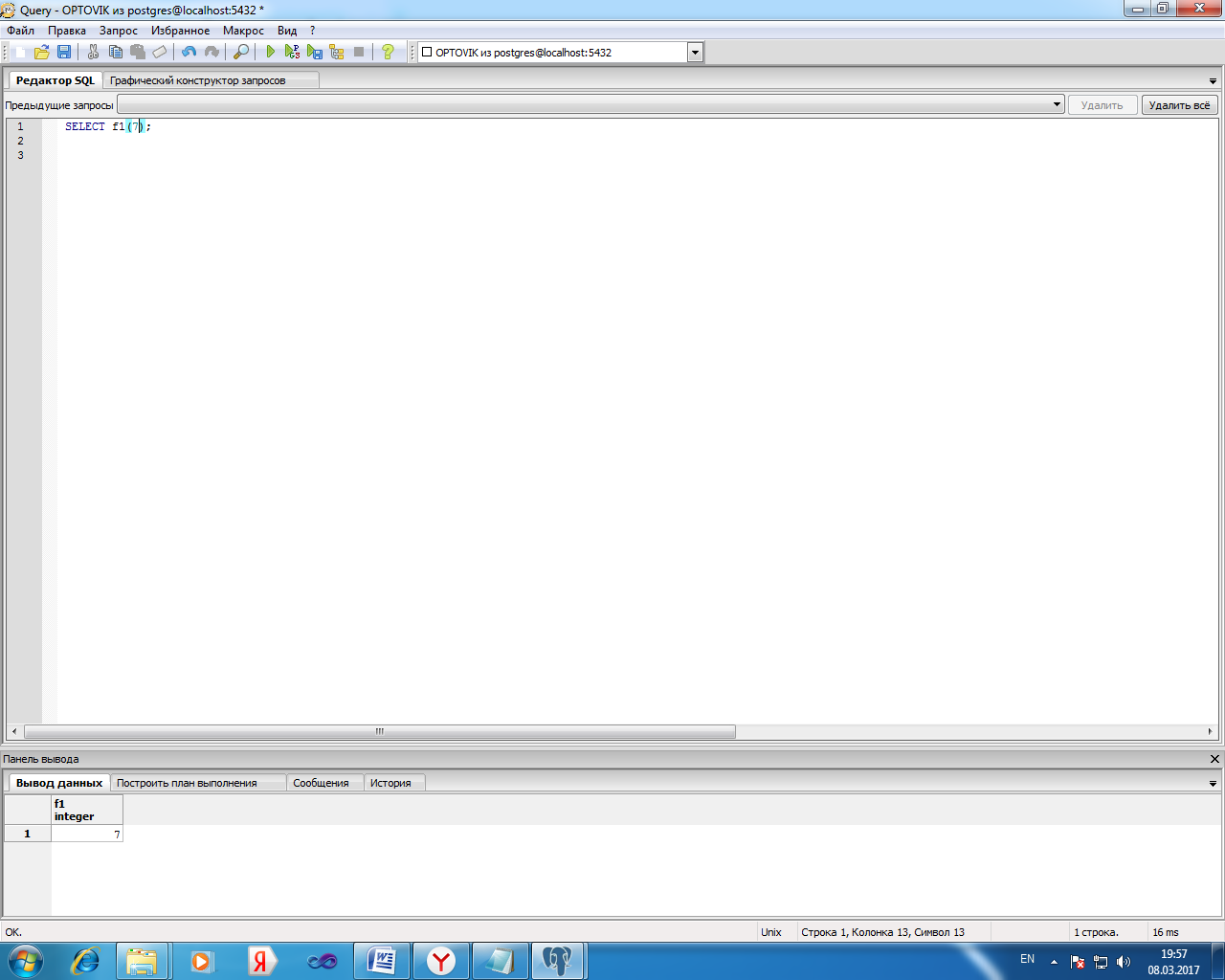
***Примечание.***

Просто скопировать код не получится. У pgAdmin имеется контроль правильной сборки при копировании из разных источников с разным форматированием. Если вы хотите, чтобы указанный пример заработал, необходимо переписать его вручную или скопировать сначала в блокнот, а уже после скопировать в редактор кода pgAdmin

Вызов функции f1:

SELECT f1(7);

Результат вызова функции f1:

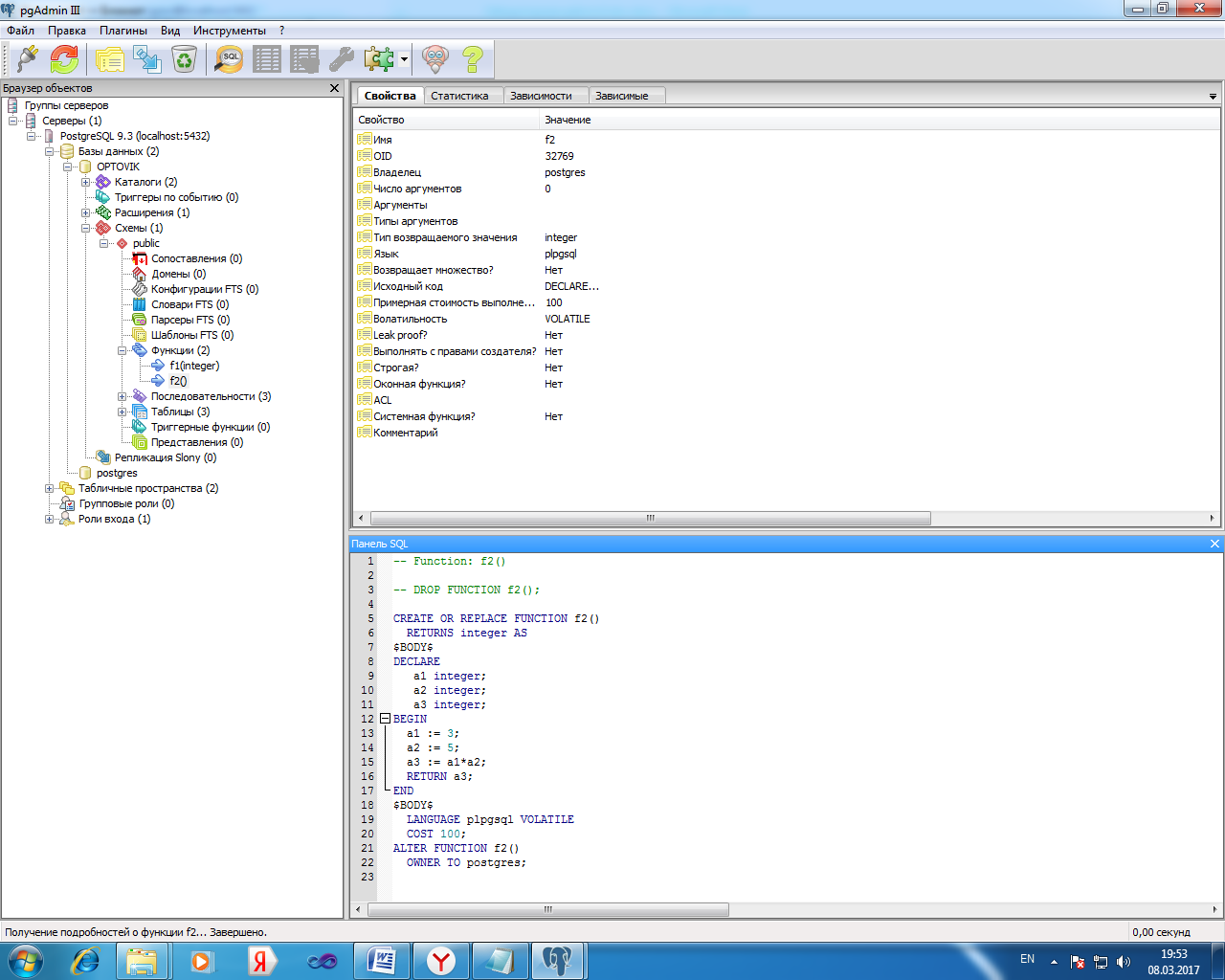


**Часть 2. Переменные и запросы в пользовательских функциях**

**Объявления переменных в функциях.**

Все переменные, используемые в блоке, должны быть объявлены в разделе объявлений блока (исключение составляют: переменная цикла FOR, выполняющая функцию итерации диапазона целых чисел, которая автоматически объявляется как целочисленная переменная, а также переменная цикла FOR, автоматически объявляющаяся как переменная записи, которая перебирает результаты выполнения курсора).

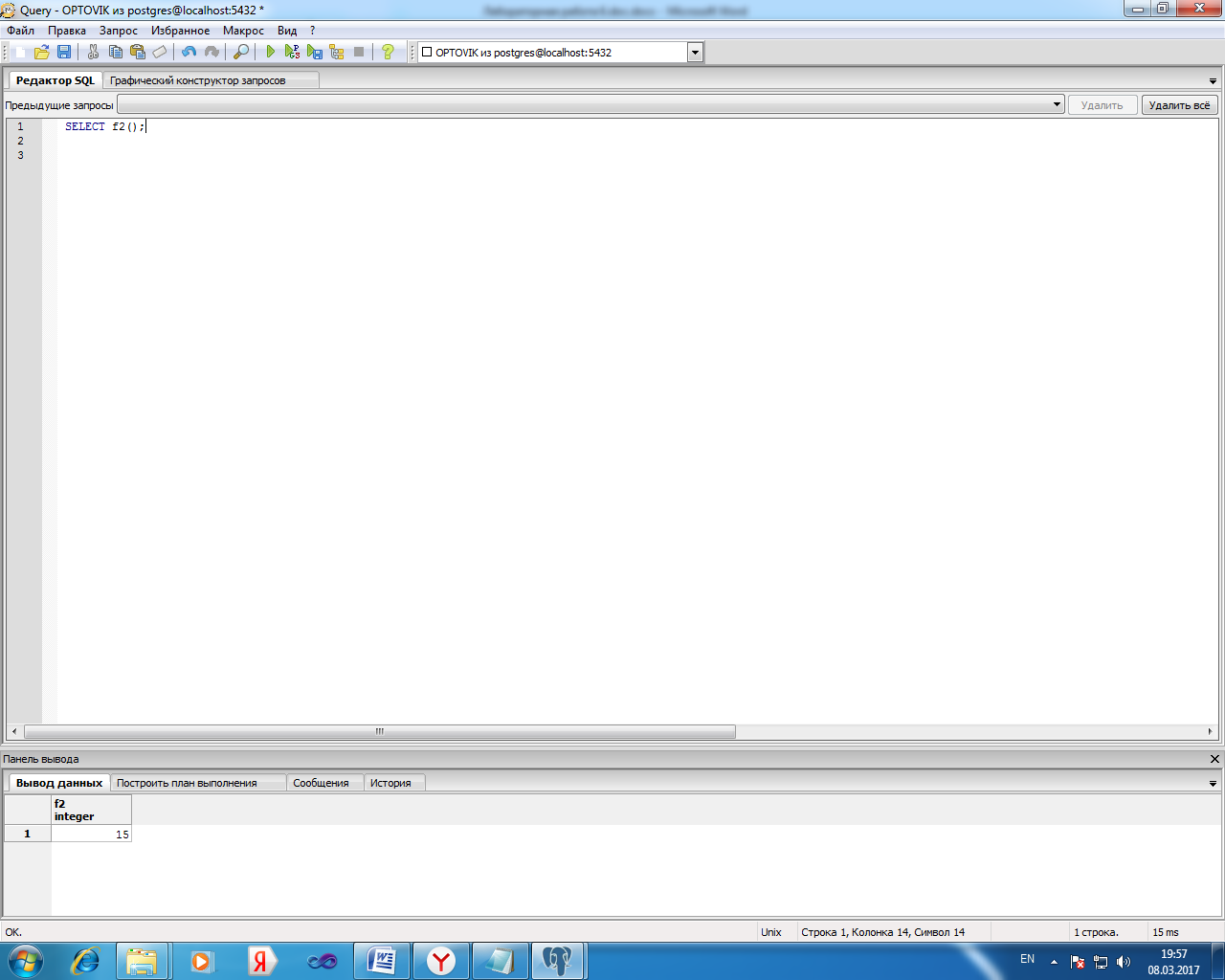
***Пример № 2 (функция без параметров).***

******

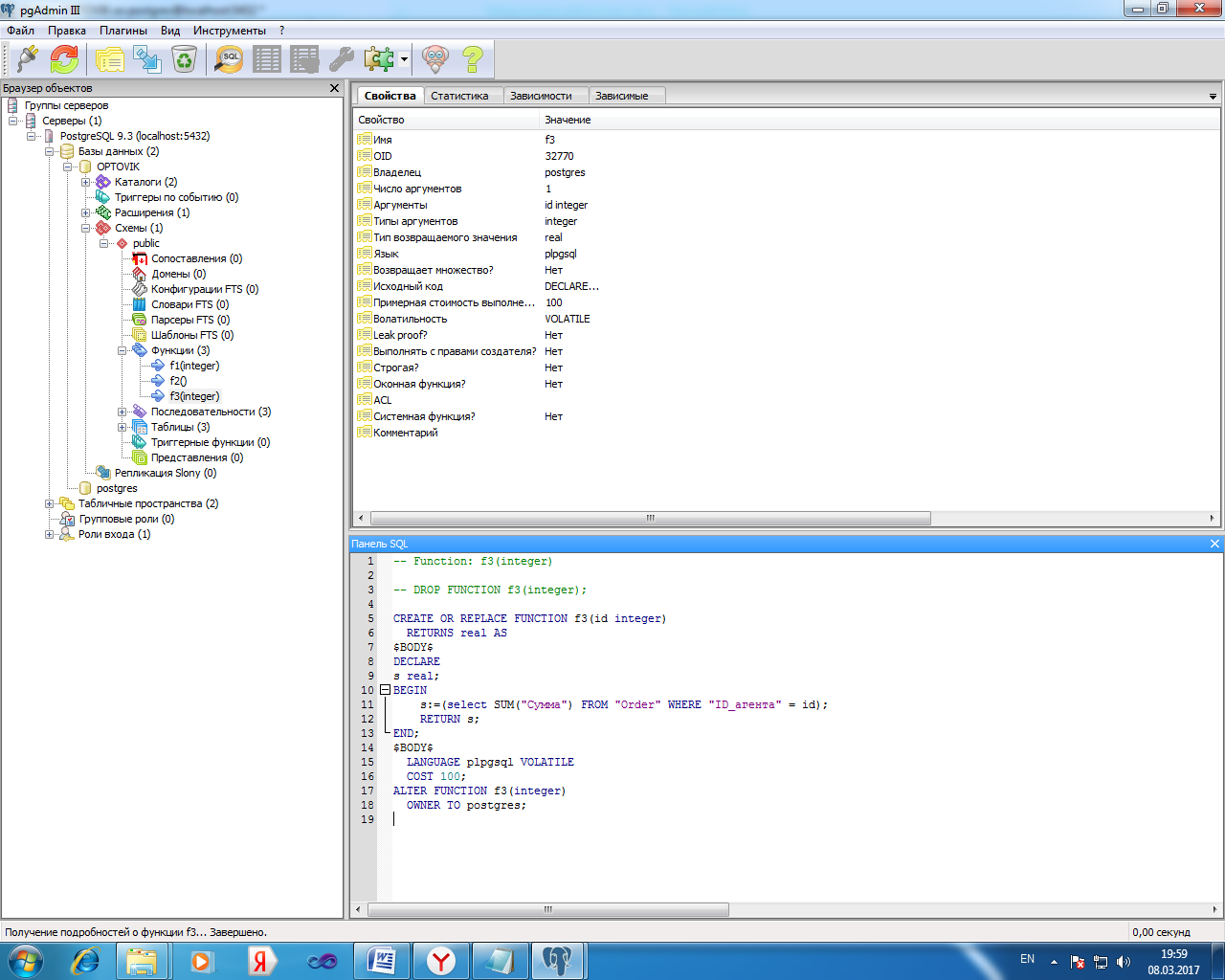
Вызов функции f2:

SELECT f2();

Результат вызова функции f2:



***Пример № 3. Запрос SQL и возврат скалярного значения через переменную.***

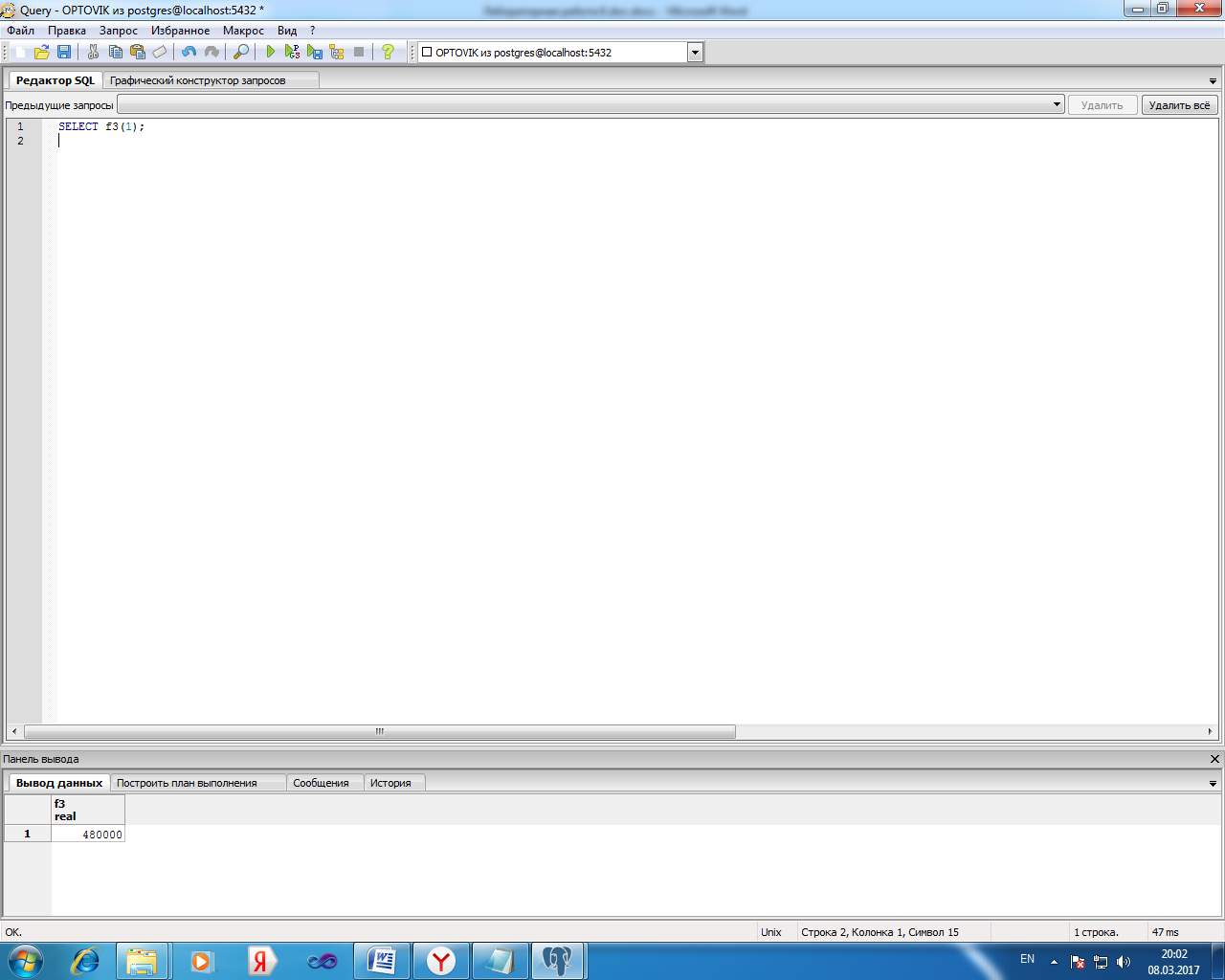


Результат выполнения запроса помещается в переменную s. Затем значение этой переменной возвращается функцией.

Вызов функции f3:

SELECT f3(1);

Результат вызова функции f3:

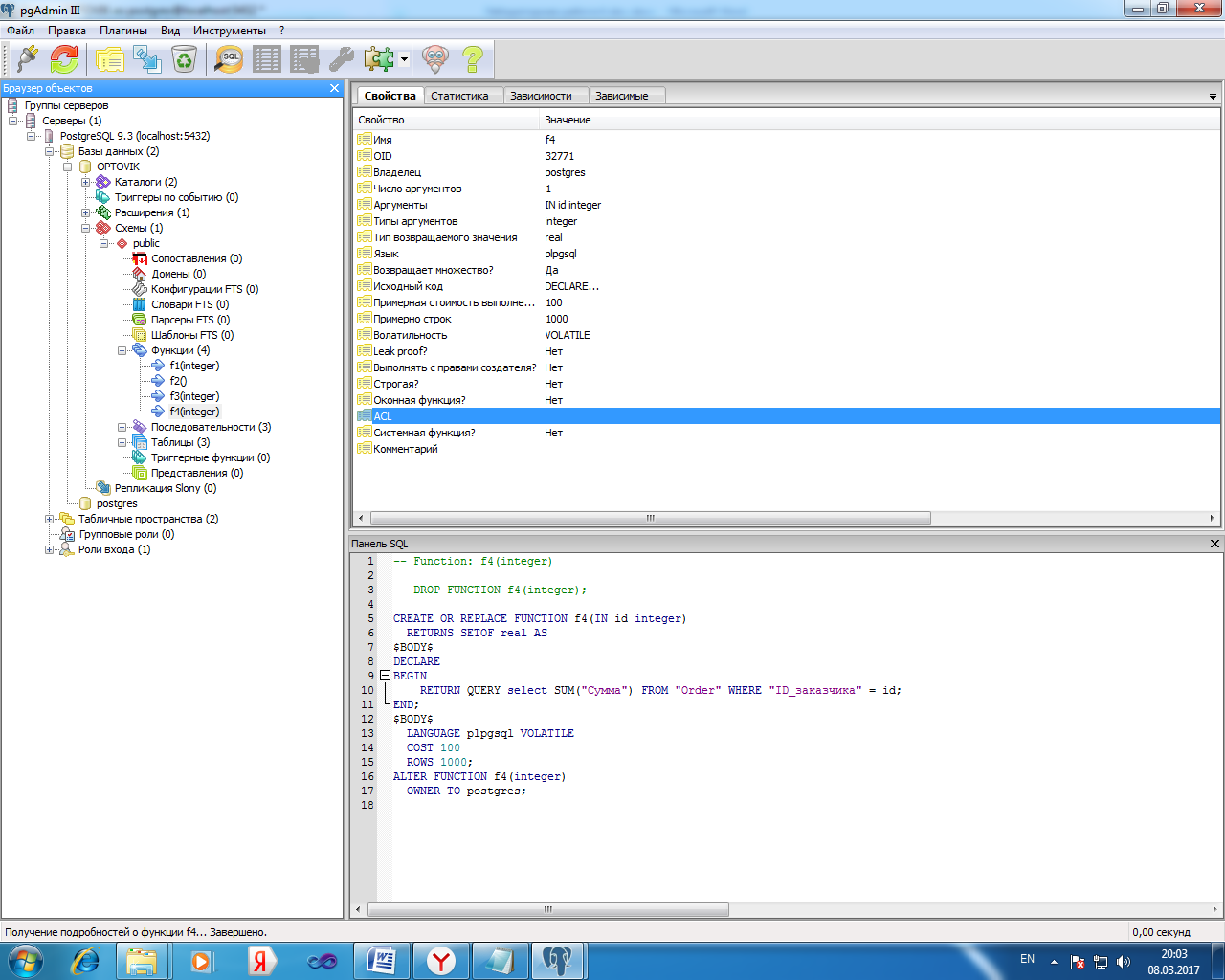


***Пример № 4.***

В данном примере возврат осуществляется с помощью оператора SELECT. Возвращаемое значение скалярное, и помещается в таблицу с одной колонкой.

Операторы **SETOF** и **QUERY** обязательны, если используется возврат с помощью оператора SELECT (возврат одного значения!).

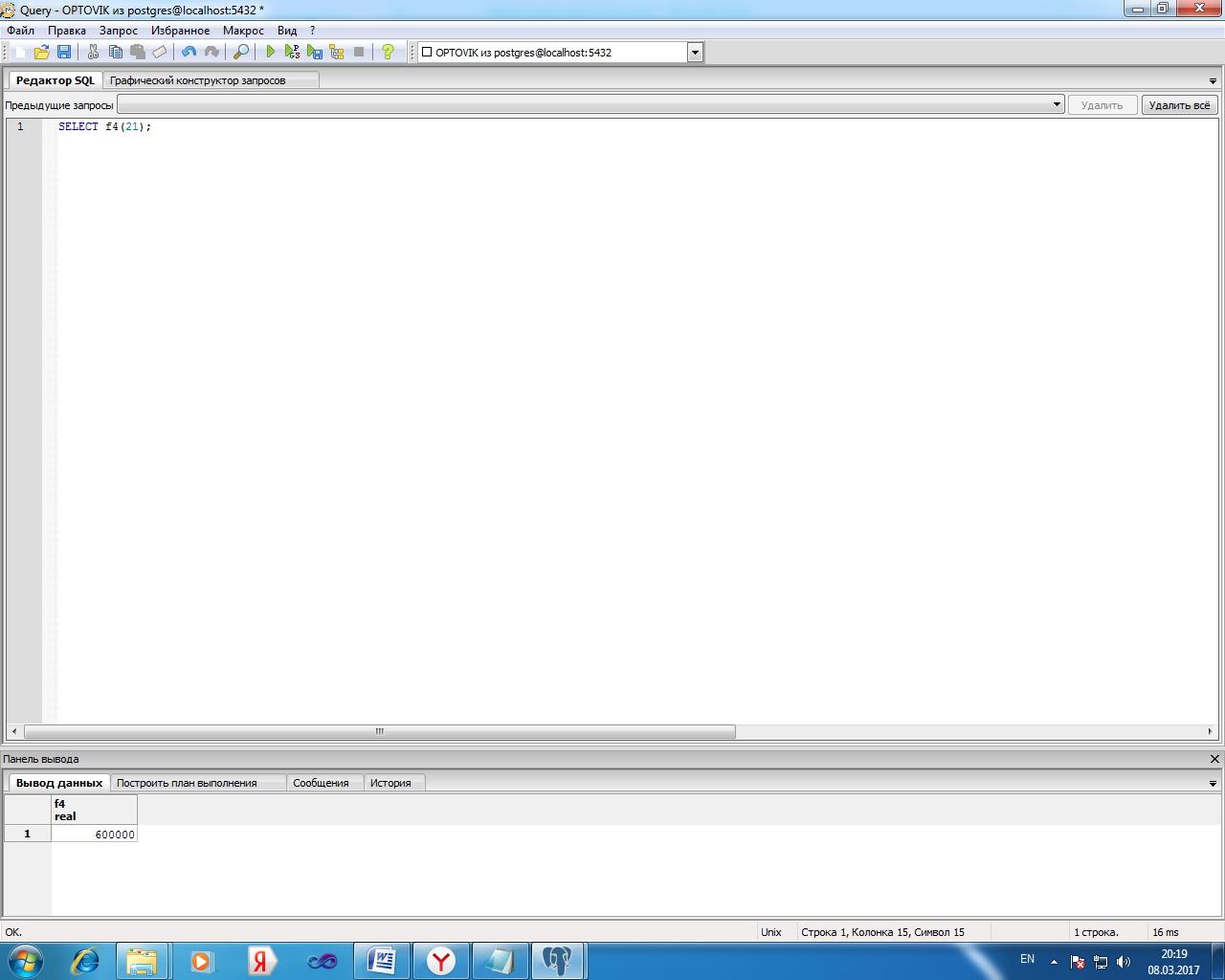
В примере возвращается общая сумма контрактов из таблицы **Order** для заказчика с ID, равным 1.



Вызов функции f4:

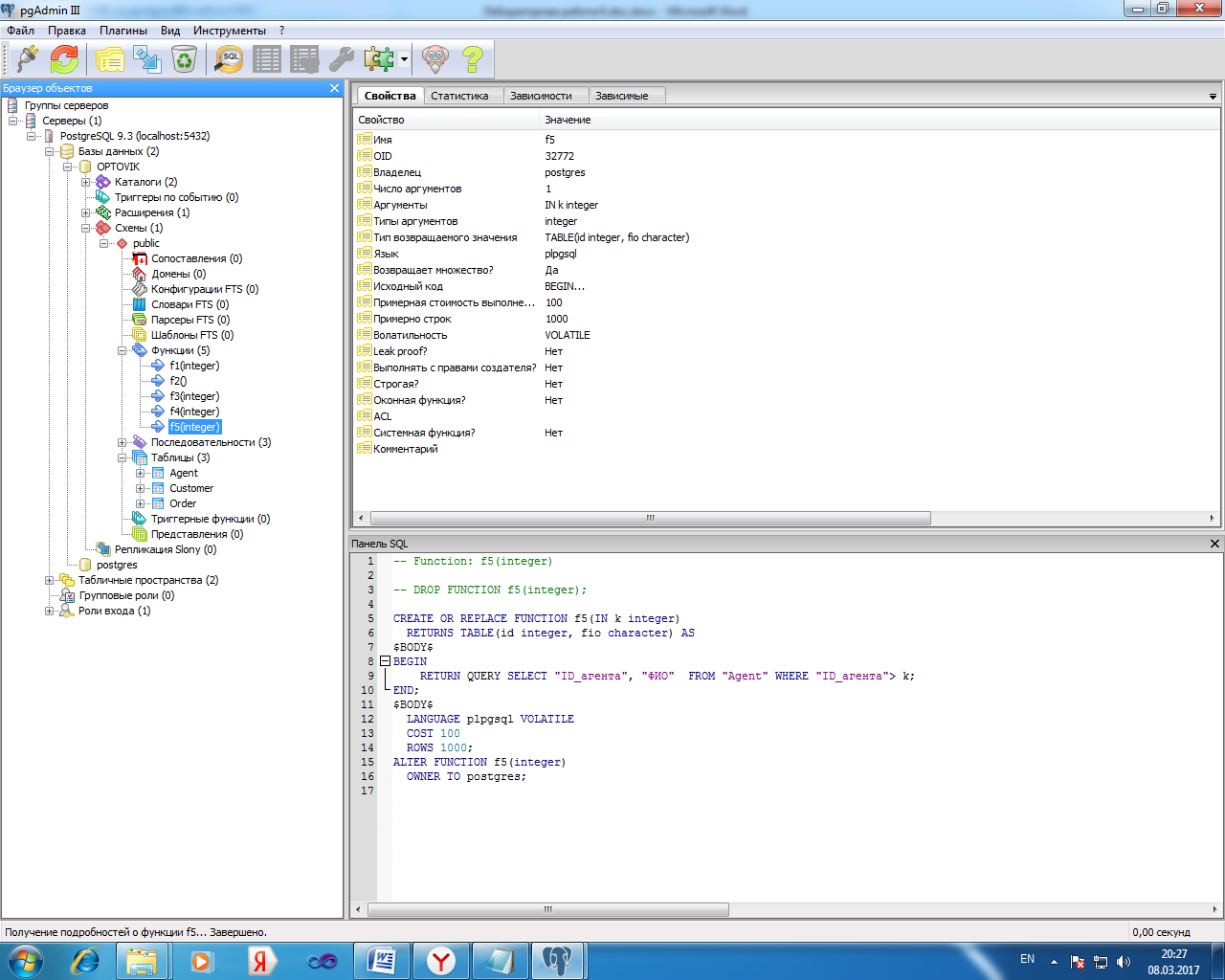
SELECT f4(19);

Результат вызова функции f4:



Пример № 5. Возврат несколько значений во временную таблицу.

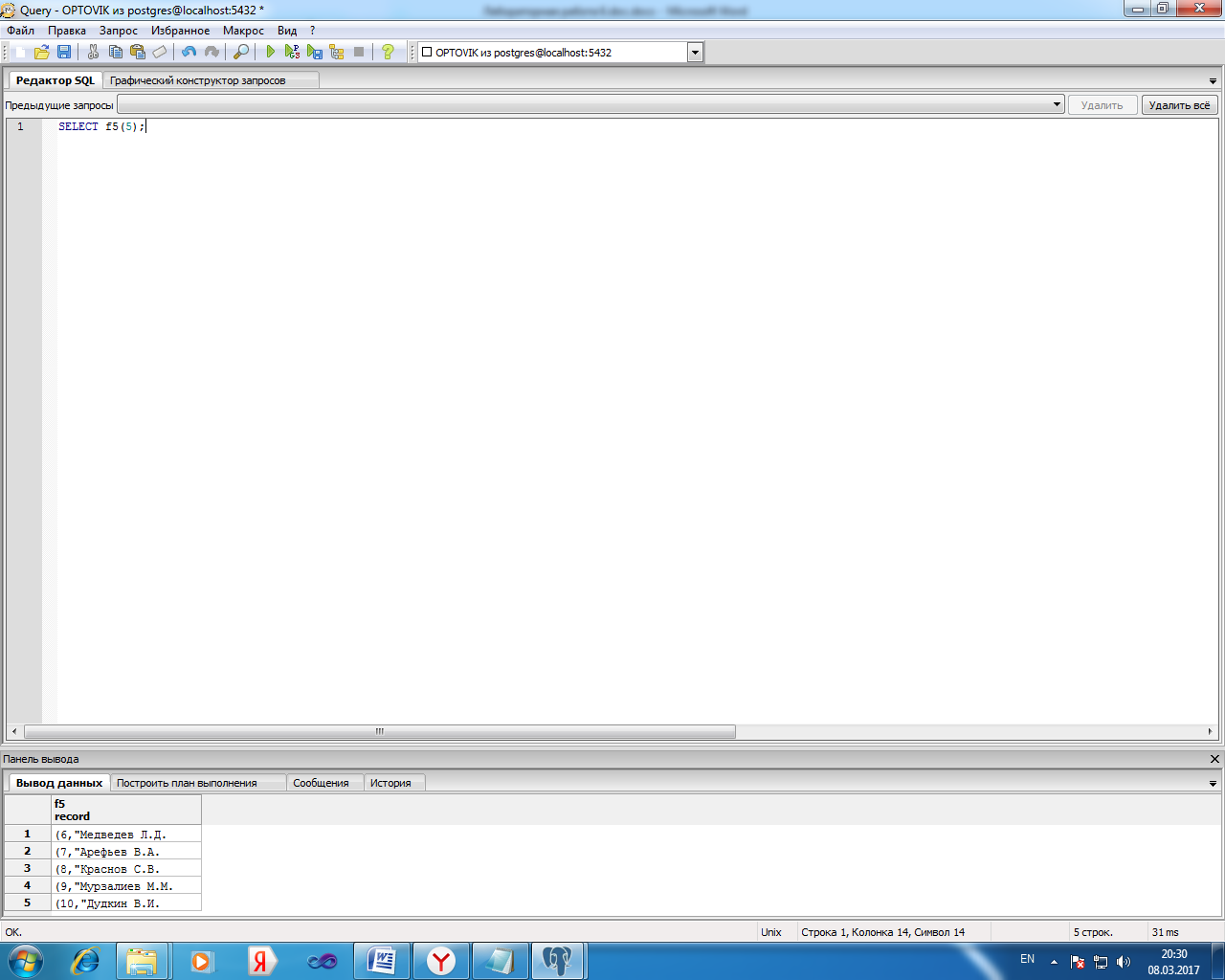
Запрос формирует таблицу из двух столбцов (ID\_агента и ФИО) для агентов с ID > 5.



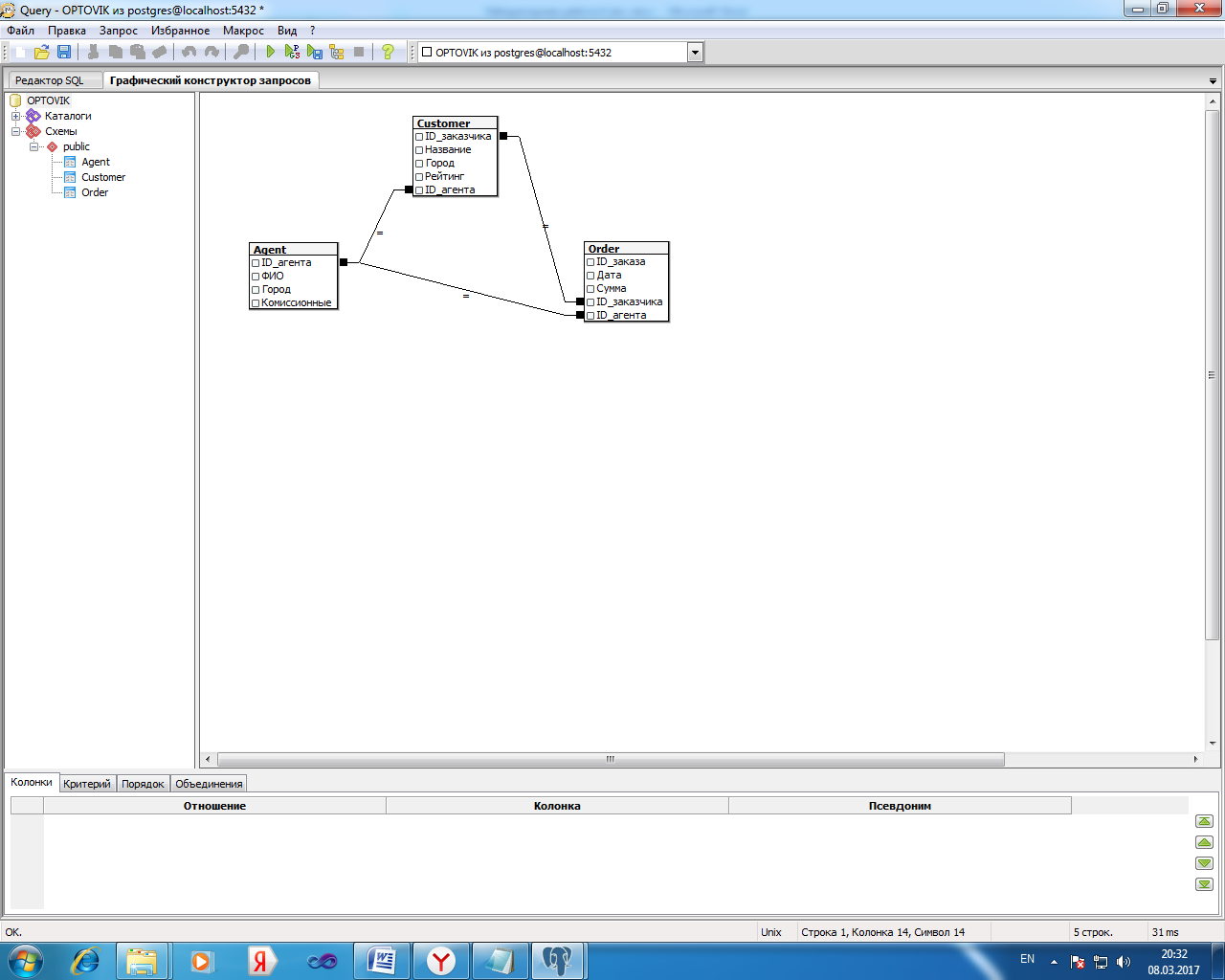
Вызов функции f5:

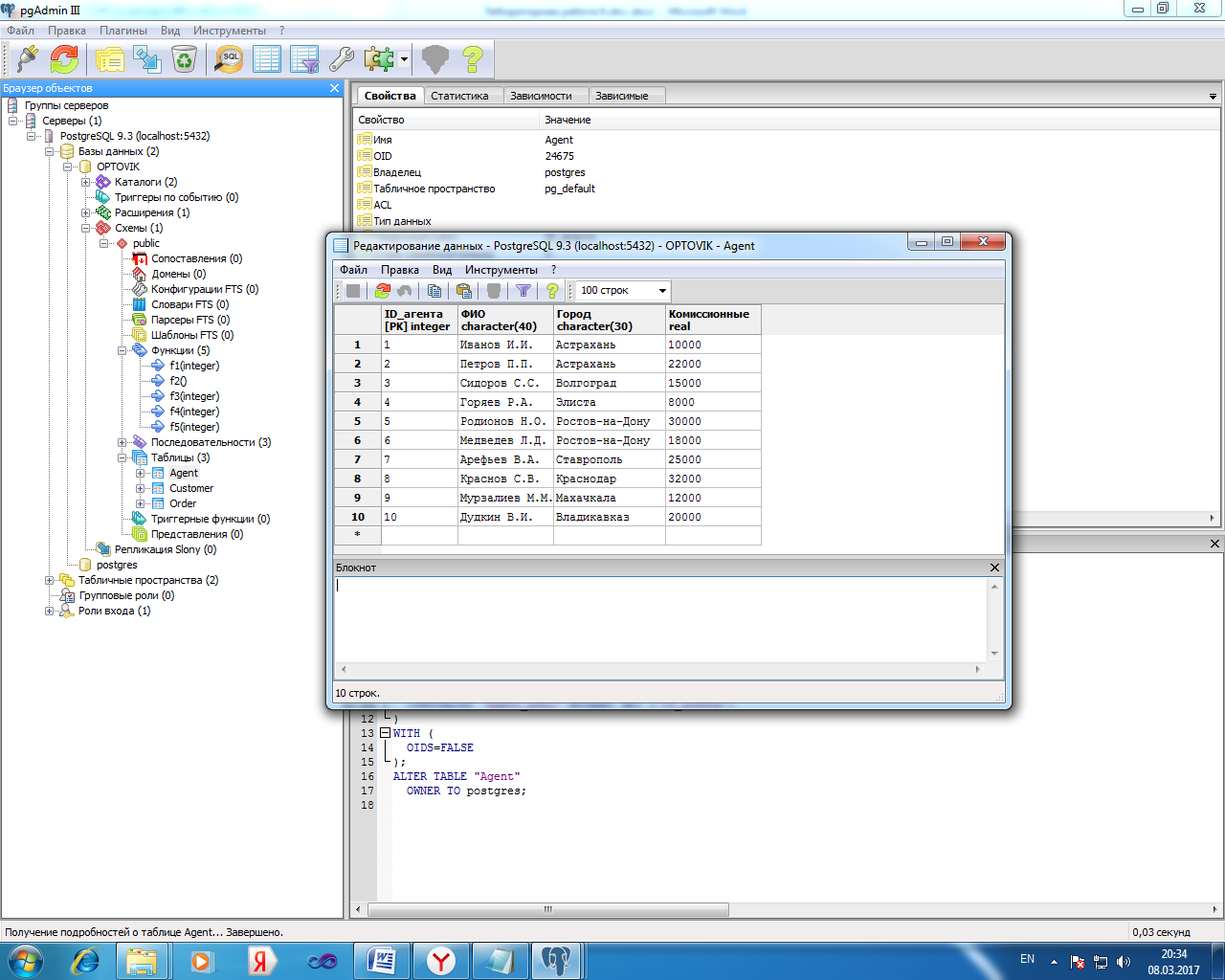
SELECT f5(5);

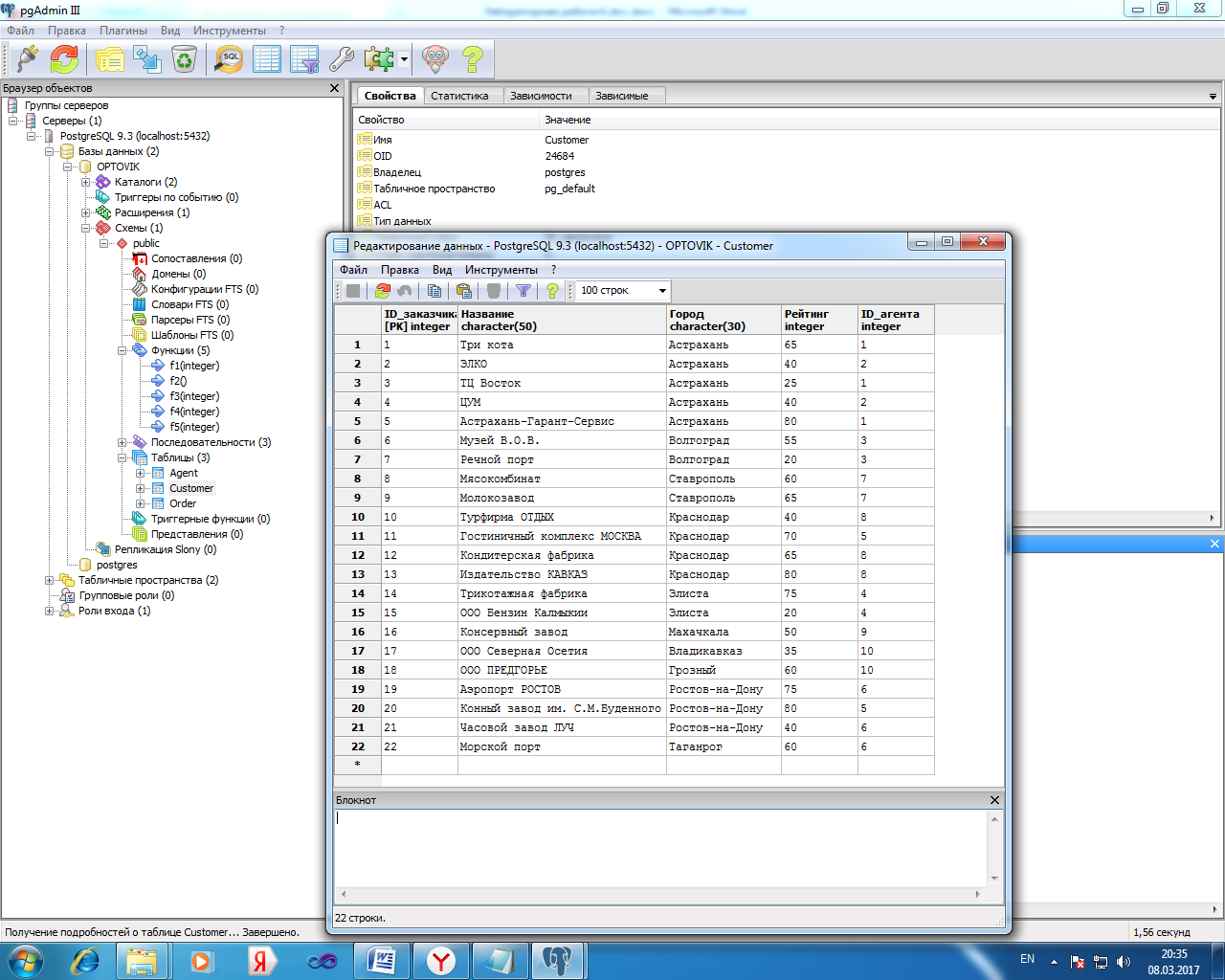
Результат вызова функции f5:

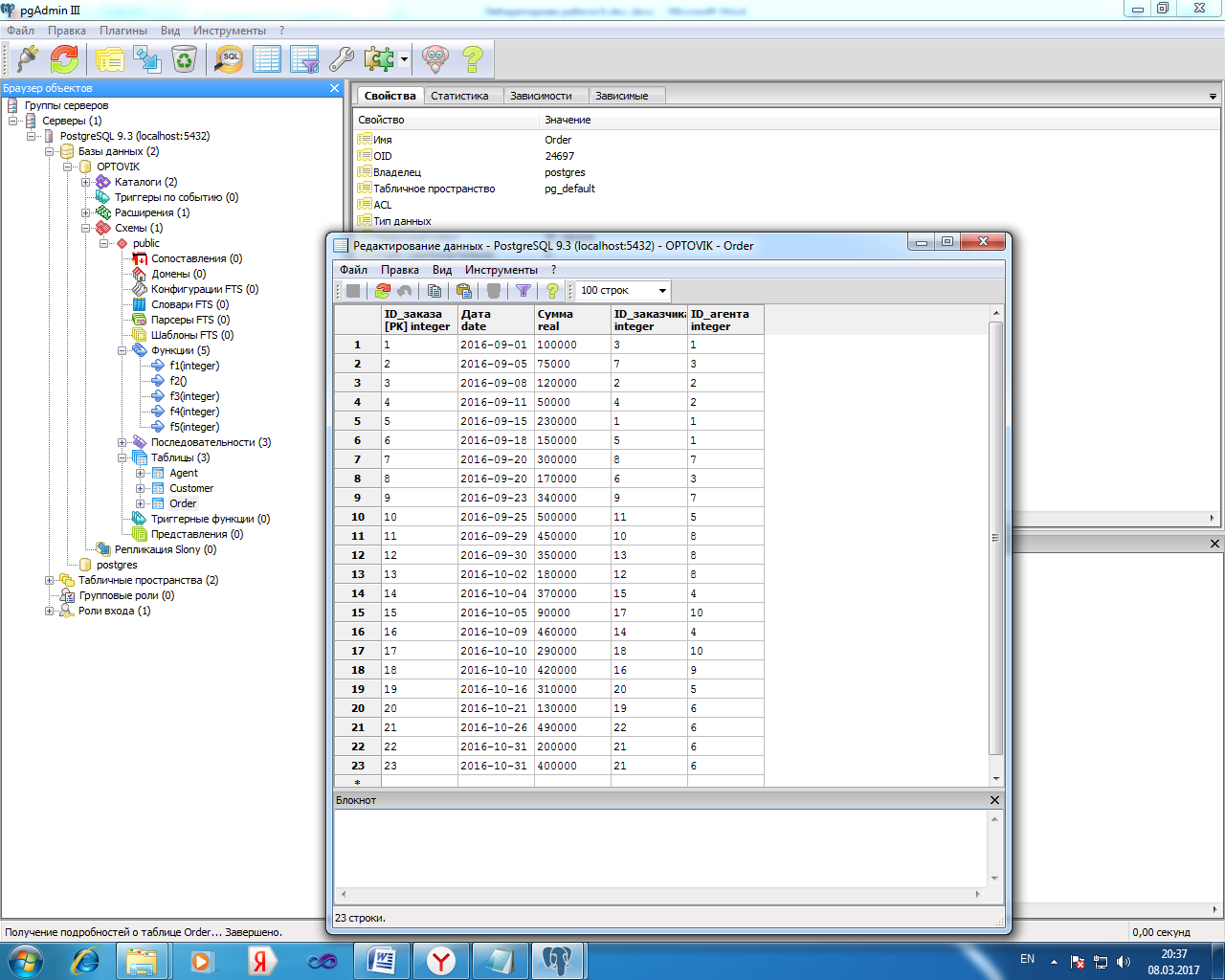


**УЧЕБНАЯ БАЗА ДАННЫХ “OPTOVIK”**

****







**ЗАДАНИЯ К ЛАБОРАТОРОНОЙ РАБОТЕ № 11**

**Выберите предметную область (из лабораторной работы №2).**

Разработайте пользовательские функции для обработки данных предметной области (достаточно три функции).

К обязательным требованиям относятся:

1. Использование одной из агрегатных функций (COUNT, SUM, AVG, MAX, MIN),
2. Выборка данных по нескольким критериям (критерии поиска задаются в параметрах вызова функции).
3. Получение отчета о деятельности организации (из предметной области) за некоторый период (период задается в параметрах вызова функции).

**Код создания таблиц:**

**CREATE TABLE Groups (**

**GroupNumber SERIAL PRIMARY KEY,**

**Specialty VARCHAR(50),**

**Department VARCHAR(50),**

**NumberOfStudents INTEGER**

**);**

**CREATE TABLE Teachers (**

**TeacherId SERIAL PRIMARY KEY,**

**LastName VARCHAR(50),**

**FirstName VARCHAR(50),**

**MiddleName VARCHAR(50),**

**Phone VARCHAR(15),**

**Experience INTEGER**

**);**

**CREATE TABLE Workload (**

**TeacherId INTEGER REFERENCES Teachers(TeacherId),**

**GroupNumber INTEGER REFERENCES Groups(GroupNumber),**

**Hours INTEGER,**

**Subject VARCHAR(50),**

**LessonType VARCHAR(20),**

**Payment DECIMAL,**

**PRIMARY KEY (TeacherId, GroupNumber)**

**);  
INSERT INTO Groups (Specialty, Department, NumberOfStudents) VALUES ('Computer Science', 'IT Department', 30);**

**INSERT INTO Groups (Specialty, Department, NumberOfStudents) VALUES ('Mathematics', 'Science Department', 25);**

**INSERT INTO Groups (Specialty, Department, NumberOfStudents) VALUES ('Business Administration', 'Management Department', 35);**

**INSERT INTO Teachers (LastName, FirstName, MiddleName, Phone, Experience) VALUES ('Smith', 'John', 'Michael', '123-456-7890', 10);**

**INSERT INTO Teachers (LastName, FirstName, MiddleName, Phone, Experience) VALUES ('Johnson', 'Emily', 'Anne', '987-654-3210', 5);**

**INSERT INTO Teachers (LastName, FirstName, MiddleName, Phone, Experience) VALUES ('Williams', 'David', 'Robert', '555-555-5555', 15);**

**INSERT INTO Workload (TeacherId, GroupNumber, Hours, Subject, LessonType, Payment) VALUES (1, 1, 10, 'Database Management', 'Lecture', 500.00);**

**INSERT INTO Workload (TeacherId, GroupNumber, Hours, Subject, LessonType, Payment) VALUES (2, 2, 8, 'Calculus', 'Lab', 400.00);**

**INSERT INTO Workload (TeacherId, GroupNumber, Hours, Subject, LessonType, Payment) VALUES (3, 3, 12, 'Marketing Strategies', 'Seminar', 600.00);**Функция для подсчета общего количества студентов в группах по определенной специальности и отделению:

CREATE OR REPLACE FUNCTION count\_students\_in\_groups(spec VARCHAR, dept VARCHAR)

RETURNS INTEGER AS $$

DECLARE

total\_students INTEGER;

BEGIN

SELECT SUM(NumberOfStudents) INTO total\_students

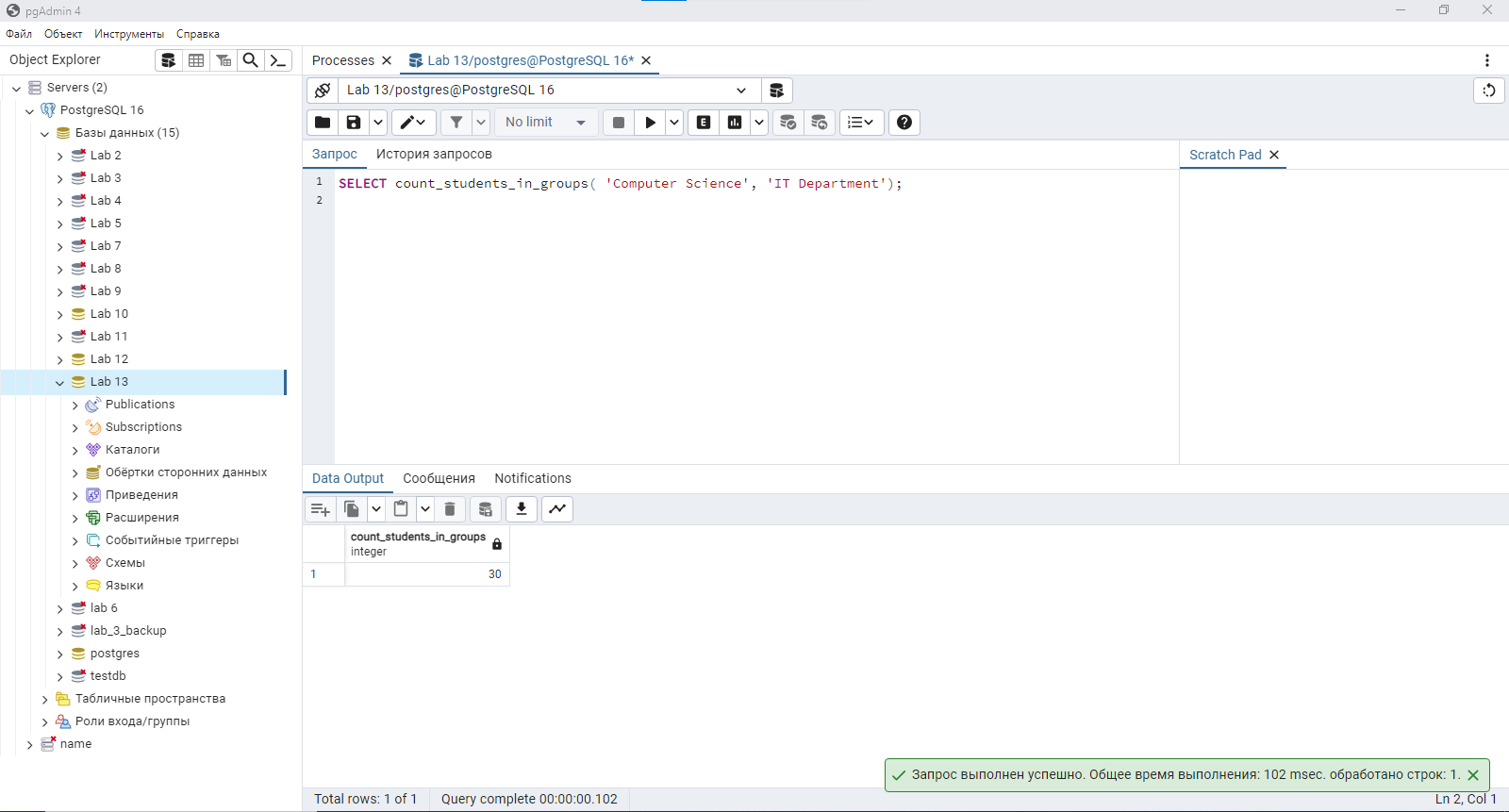
FROM Groups

WHERE Specialty = spec AND Department = dept;

RETURN total\_students;

END;

$$ LANGUAGE plpgsql;



2. Функция для выборки информации о преподавателе по его фамилии и номеру телефона:

CREATE OR REPLACE FUNCTION getTeacherInfoByLastNameAndPhone(last\_name VARCHAR, phone\_num VARCHAR) RETURNS TABLE (

TeacherId2 INTEGER,

FirstName VARCHAR,

MiddleName VARCHAR,

Experience INTEGER

) AS $$

BEGIN

RETURN QUERY

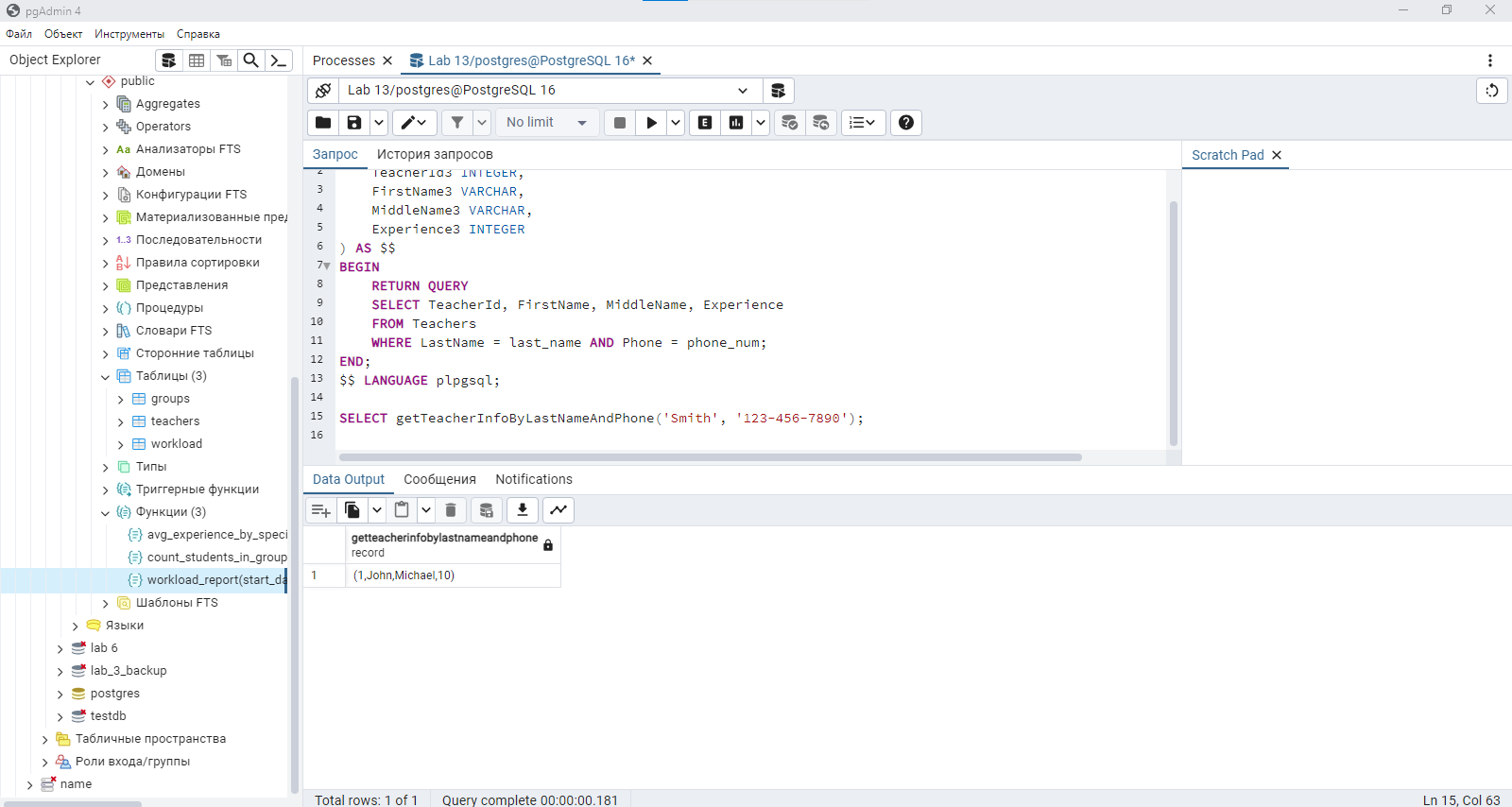
SELECT TeacherId, FirstName, MiddleName, Experience

FROM Teachers

WHERE LastName = last\_name AND Phone = phone\_num;

END;

$$ LANGUAGE plpgsql;



3. Функция для получения отчета о нагрузке преподавателя за определенный период времени:

CREATE OR REPLACE FUNCTION getWorkloadReportForTeacher(teacher\_id INTEGER, start\_date1 DATE, end\_date1 DATE) RETURNS TABLE (

GroupNumber INTEGER,

Hours INTEGER,

Subject VARCHAR,

LessonType VARCHAR,

Payment DECIMAL

) AS $$

BEGIN

RETURN QUERY

SELECT w.GroupNumber, w.Hours, w.Subject, w.LessonType, w.Payment

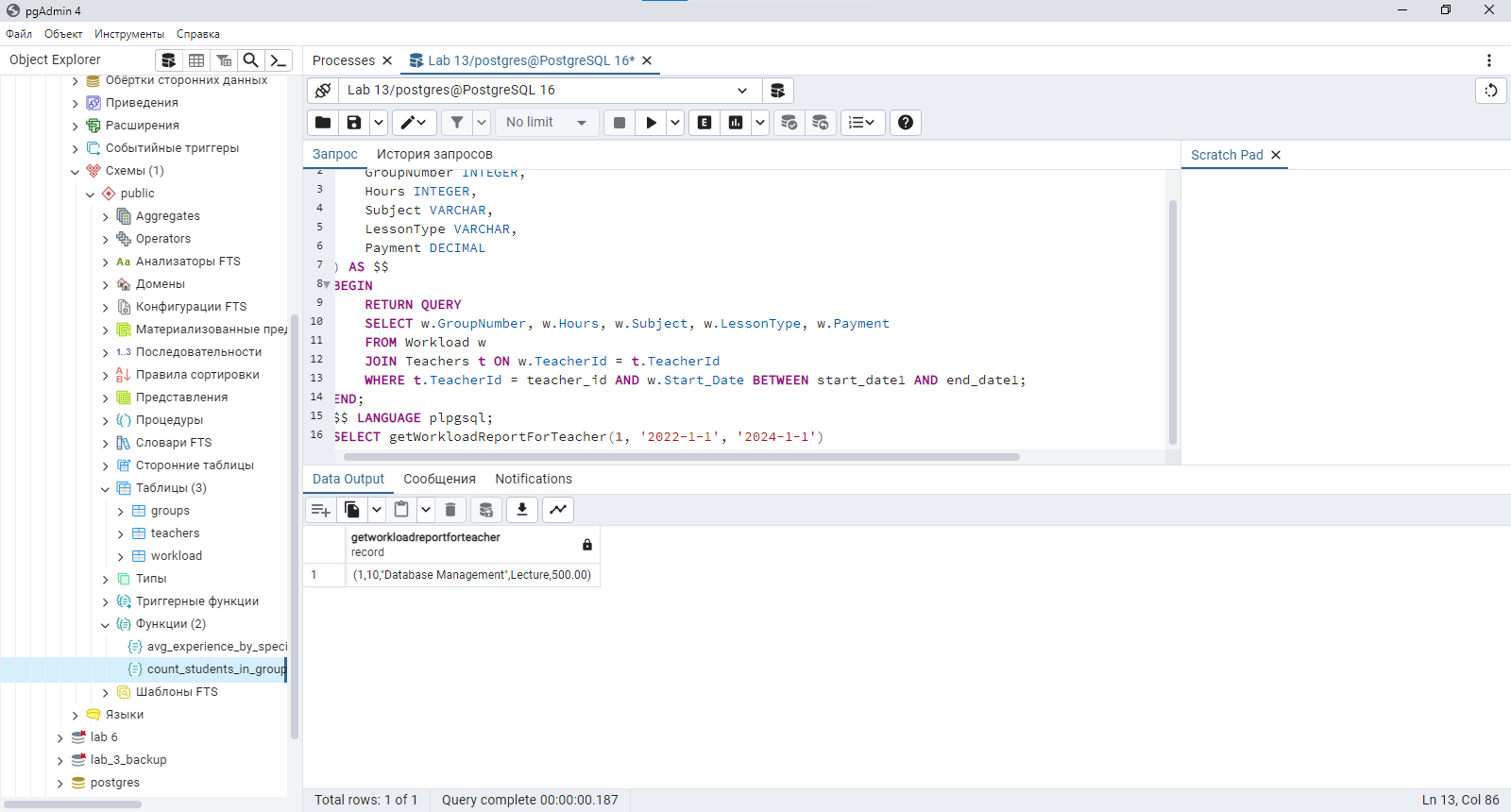
FROM Workload w

JOIN Teachers t ON w.TeacherId = t.TeacherId

WHERE t.TeacherId = teacher\_id AND w.Start\_Date BETWEEN start\_date1 AND end\_date1;

END;

$$ LANGUAGE plpgsql;

SELECT getWorkloadReportForTeacher(1, '2022-1-1', '2024-1-1') ****

**ДОПОЛНИТЕЛЬНЫЙ МАТЕРИАЛ:**

DROP TABLE "Test1";

CREATE TABLE "Test1"("A" integer, "B" integer, "C" integer);

INSERT INTO "Test1" ("A", "B", "C") VALUES (1,2,3);

INSERT INTO "Test1" ("A", "B", "C") VALUES (4,5,6);

INSERT INTO "Test1" ("A", "B", "C") VALUES (7,8,9);

INSERT INTO "Test1" ("A", "B", "C") VALUES (10,11,12);

INSERT INTO "Test1" ("A", "B", "C") VALUES (13,14,15);

INSERT INTO "Test1" ("A", "B", "C") VALUES (15,17,18);

INSERT INTO "Test1" ("A", "B", "C") VALUES (19,20,21);

INSERT INTO "Test1" ("A", "B", "C") VALUES (22,23,24);

INSERT INTO "Test1" ("A", "B", "C") VALUES (25,26,27);

INSERT INTO "Test1" ("A", "B", "C") VALUES (28,29,30);

------------------------------------------------

DROP FUNCTION temp\_func();

CREATE OR REPLACE FUNCTION temp\_func()

RETURNS void AS

$BODY$

begin

update "Test1" set "C"="C"\*2;

end

$BODY$

LANGUAGE plpgsql;

-------------------------------------------------

DROP FUNCTION f();

CREATE OR REPLACE FUNCTION f()

RETURNS void AS

$BODY$

begin

**perform** temp\_func();

end

$BODY$

LANGUAGE plpgsql;

-------------------------------------------------

select f();

**Вывод:** *:*в ходе лабораторной работы я изучил правила создания функций, приобрести практические навыки создания функций в среде PostgreSQL, а также научился проектировать БД в целом на основе поставленного задания с использованием всех полученных ранее базовых знаний и навыков.