

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
УНИВЕРСИТЕТ ИТМО»**

**Факультет инфокоммуникационных технологий**

**Дисциплина:**  
**«Базы данных»**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3  
«ПРОЦЕДУРЫ, ФУНКЦИИ, ТРИГГЕРЫ В PostgreSQL»**

**Выполнил:**  
студент группы К32391  
Петров Андрей Сергеевич

---

(подпись)

**Проверил:**  
Говорова Марина Михайловна

---

(отметка о выполнении)

---

(подпись)

Санкт-Петербург  
2023 г.

**Цель работы:** овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

**Оборудование:** компьютерный класс.

**Программное обеспечение:** СУБД PostgreSQL, SQL Shell (psql).

**Практическое задание:**

**Вариант 1**

1. 2.Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4).
2. Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL (согласно индивидуальному заданию, часть 5). Допустимо создать универсальный триггер или отдельные триггеры на логирование действий.

Выполнение:

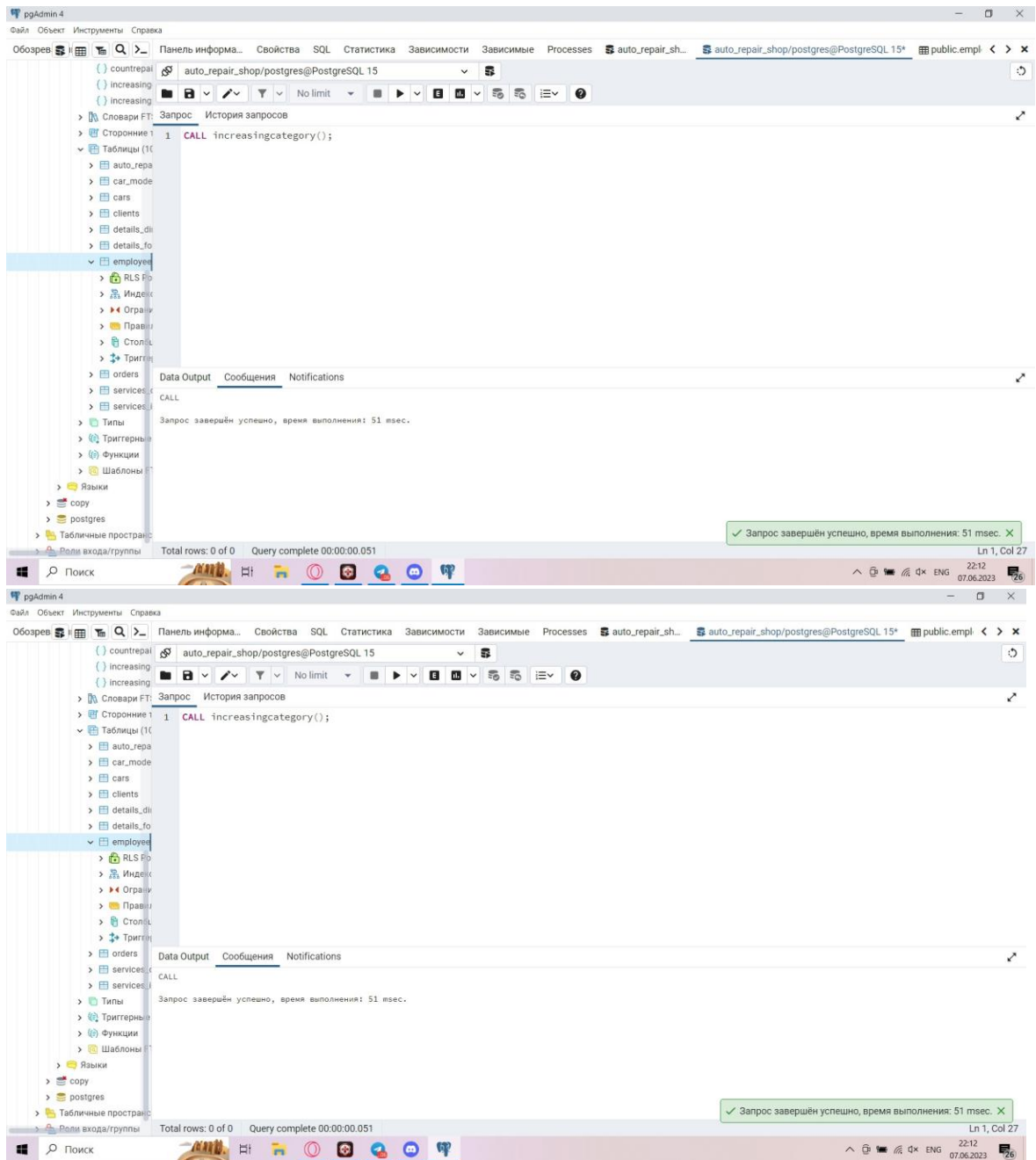
- 1) Процедура для повышения разряда тех мастеров, которые отремонтировали больше 3 автомобилей.

The screenshot displays the pgAdmin 4 interface. The top pane shows the SQL editor with a PL/pgSQL procedure named `IncreasingCategory()`. The procedure iterates through all employees, counts the number of services they have ordered, and updates their category if the count is greater than 3.

```
1 CREATE PROCEDURE IncreasingCategory() LANGUAGE 'plpgsql' AS $$
2 BEGIN
3 UPDATE employees
4 SET category = category + 1
5 WHERE employees.id IN
6 (SELECT employees.id
7 FROM employees,
8 services_in_order
9 WHERE services_in_order.employee_id = employees.id
10 GROUP BY employees.id
11 HAVING count(services_in_order.employee_id));
12 END;
13 $$
```

The bottom pane shows the results of the procedure execution, which was successful. Below the SQL editor, a table view displays the data from the `public.employees` table, sorted by `id` in ascending order.

id	name	post	specialization	category
1	Иванов Сергей Игоревич	Автомеханик	Малар	1
2	Петров Сергей Игоревич	Автомеханик	Автослесарь	2
3	Смирнов Сергей Игоревич	Автомеханик	Автослесарь	1
4	Сидоров Сергей Игоревич	Автомеханик	Жестянщик	3



## 2) Процедура повышения цены деталей для автомобиля “Ford” на 10 %

The image shows two screenshots of the pgAdmin 4 interface. The top screenshot displays the SQL editor with a procedure named `IncreasingPriceFordDatails()` that updates the price of parts for Ford cars by 10%. The bottom screenshot shows the results of a query that lists the parts and their updated prices.

**SQL Procedure:**

```
1 CREATE PROCEDURE IncreasingPriceFordDatails() LANGUAGE 'plpgsql' AS $$
2 BEGIN
3 UPDATE details_directory
4 SET price = price * 1.1
5 WHERE id IN
6 (SELECT details_directory.id
7 FROM details_directory,
8 car_models
9 WHERE car_models.brand = 'Ford' and
10 car_models.id = details_directory.model_id);
11 END;
12 $$
```

**Query Results:**

id	name	price	model_id
5	Воздушный фильтр	795	
6	Маховик коробки передач	50000	
7	Помпа двигателя	3400	
8	Стойка амортизатора	10000	6
9	Свечи зажигания	3000	
10	Тормозная жидкость	350	

pgAdmin 4

Файл Объект Инструменты Справка

Обзор Панель информации Свойства SQL Статистика Зависимости Зависимые Processes auto\_repair\_shop/postgres@PostgreSQL 15\* auto\_repair\_sh... public.empl < >

auto\_repair\_shop/postgres@PostgreSQL 15

Запрос История запросов

```
1 CALL increasingpricefordetails();
```

Процедуры

- countrepair
- increasing
- increasing

Словари РТ

Сторонние

Таблицы

- auto\_repa
- car\_model
- cars
- clients
- details
- details
- employee

Data Output Сообщения Notifications

CALL

Запрос завершен успешно, время выполнения: 66 мсек.

Total rows: 0 of 0 Query complete 00:00:00.066 Ln 1, Col 35

22:17 07.06.2023

pgAdmin 4

Файл Объект Инструменты Справка

Обзор Панель информации Свойства SQL Статистика Зависимости Зависимые Processes auto\_repair\_sh... auto\_repair\_sh... public.employee... public.employee... < >

public.details\_directory/auto\_repair\_shop/postgres@PostgreSQL...

Запрос История запросов

```
1 SELECT * FROM public.details_directory
2 ORDER BY id ASC
```

Процедуры

- countrepair
- increasing
- increasing

Словари РТ

Сторонние

Таблицы

- auto\_repa
- car\_model
- cars
- clients
- details
- details
- employee

Data Output Сообщения Notifications

id	name	price	model_id
5	Воздушный фильтр	795	[null]
6	Маховик коробки передач	50000	[null]
7	Помпа двигателя	3400	[null]
8	Стойка амортизатора	11000	6
9	Свечи зажигания	3000	[null]
10	Тормозная жидкость	350	[null]

Total rows: 10 of 10 Query complete 00:00:00.379

✓ Запрос выполнен успешно. Общее время выполнения: 379 мсек. обработано строк: 10. ✕

22:22 07.06.2023

### 3) Функция, возвращающая сколько автомобилей отремонтировал каждый механик.

The screenshot displays the pgAdmin 4 interface with the 'auto\_repair\_shop/postgres@PostgreSQL 15' database selected. The 'History' tab shows a sequence of queries executed on 08.06.2023 at 03:27:28. The first query is a 'SELECT \* FROM count\_repairs()' which returns three rows of data. The second query is a 'CREATE FUNCTION count\_repairs()' which defines a function that returns a table with two columns: 'name' (character) and 'count' (integer). The function is defined as follows:

```
CREATE FUNCTION count_repairs() RETURNS TABLE(name char, count int) AS $$
SELECT employees.name, count(DISTINCT services_in_order.order_id)
FROM employees, services_in_order
WHERE employees.id = services_in_order.employee_id
GROUP BY employees.id;
$$ LANGUAGE 'sql';
```

The 'Data Output' tab shows the results of the first query, which are three rows of data:

name	count
Иванов Сергей Игоревич	2
Петров Сергей Игоревич	8
Смирнов Сергей Игоревич	1

The 'Messages' tab shows a confirmation message: 'Запрос завершён успешно, время выполнения: 58 мсек.'

The second screenshot shows the same pgAdmin 4 interface, but with the 'History' tab displaying a single query: 'SELECT \* FROM count\_repairs()'. The 'Data Output' tab shows the same three rows of data as the first screenshot.

## Создание триггера:

The screenshot shows the pgAdmin 4 interface with the 'auto\_repair\_shop/postgres@PostgreSQL 15' database selected. The left sidebar shows the 'Таблицы (11)' folder expanded, with 'log\_table' selected. The main pane displays the SQL editor with the following code:

```
1 CREATE FUNCTION log_ac() RETURNS TRIGGER AS $$
2 DECLARE cur_time timestamp;
3 DECLARE cur_user varchar;
4 BEGIN
5 SELECT CURRENT_TIMESTAMP INTO cur_time;
6 SELECT CURRENT_USER INTO cur_user;
7 IF (TG_OP = 'INSERT') THEN
8     INSERT INTO log_table (time_ac, action, username, old_row, new_row)
9     VALUES (cur_time, 'INSERT', cur_user, OLD, NEW);
10 ELSEIF (TG_OP = 'DELETE') THEN
11     INSERT INTO log_table (time_ac, action, username)
12     VALUES (cur_time, 'DELETE', cur_user);
13 ELSEIF (TG_OP = 'UPDATE') THEN
14     INSERT INTO log_table (time_ac, action, username, old_row, new_row)
15     VALUES (cur_time, 'UPDATE', cur_user, OLD, NEW);
16 END IF;
17 END;
18 $$ LANGUAGE plpgsql;
```

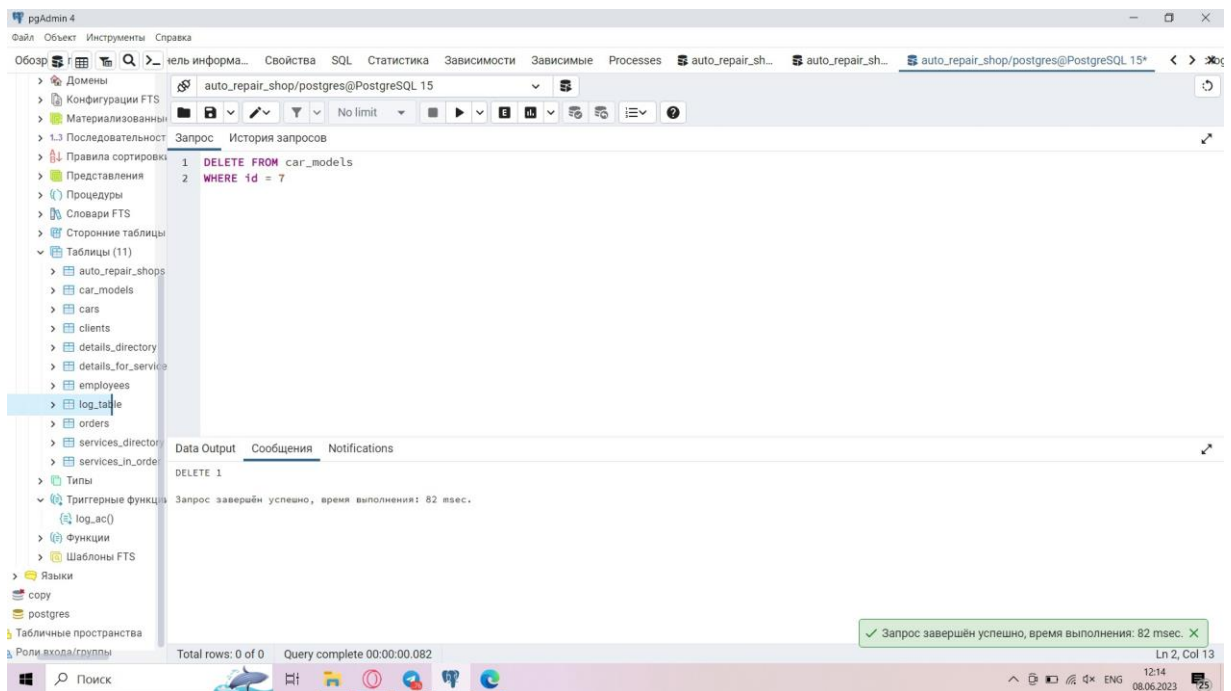
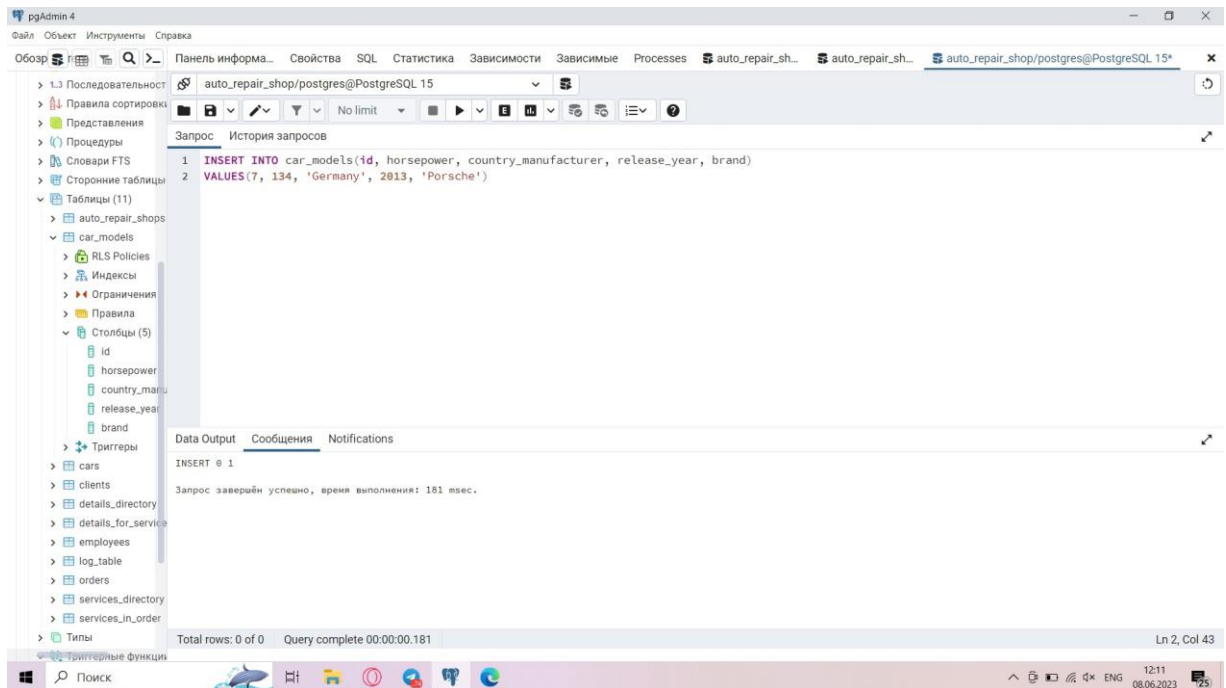
The 'Data Output' tab is active, showing the message: 'Запрос завершен успешно, время выполнения: 158 мсек.' A green status bar at the bottom right confirms: '✓ Запрос завершен успешно, время выполнения: 158 мсек.'

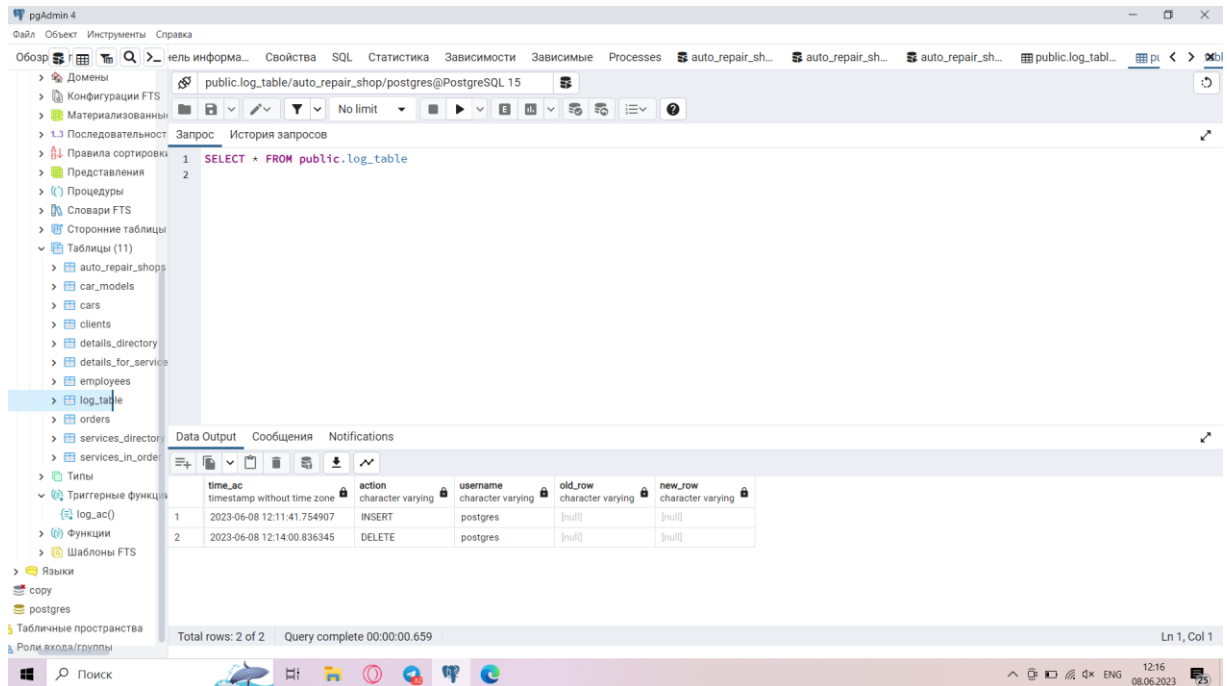
The screenshot shows the pgAdmin 4 interface with the 'auto\_repair\_shop/postgres@PostgreSQL 15' database selected. The left sidebar shows the 'Триггерные функции' folder expanded. The main pane displays the SQL editor with the following code:

```
1 CREATE TRIGGER log_ac
2 AFTER INSERT OR UPDATE OR DELETE
3 ON car_models
4 EXECUTE PROCEDURE log_ac();
```

The 'Data Output' tab is active, showing the message: 'Запрос завершен успешно, время выполнения: 264 мсек.'







Вывод: были созданы функции и процедуры, триггерная функция и логирующий триггер.