**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Отчет**

по лабораторной работе №5 «Процедуры, функции, триггеры в PostgreSQL**»**

по дисциплине **«Проектирование и реализация баз данных»**

Автор: Бунос М.В.

Факультет: ИКТ

Группа: К3241

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

**Оглавление**

[1. Создать процедуры согласно индивидуальному заданию 3](#_Toc152879395)

[2. Создать необходимый триггер 5](#_Toc152879396)

[Вывод 6](#_Toc152879397)

**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:**

1. Создать процедуры/функции 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.

# Создать процедуры согласно индивидуальному заданию

1. Выполнить списание автомобилей, выпущенных ранее заданного года.

**CREATE** **OR** **REPLACE** **PROCEDURE** WriteOffCars(before\_year **INTEGER**)

**LANGUAGE** plpgsql

**AS** $$

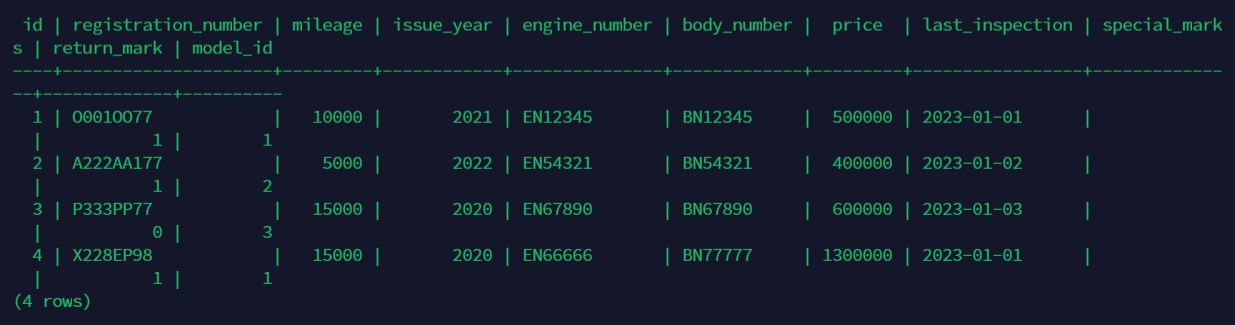
**BEGIN**

**DELETE** **FROM** cars **WHERE** issue\_year < before\_year;

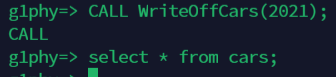
**END**;

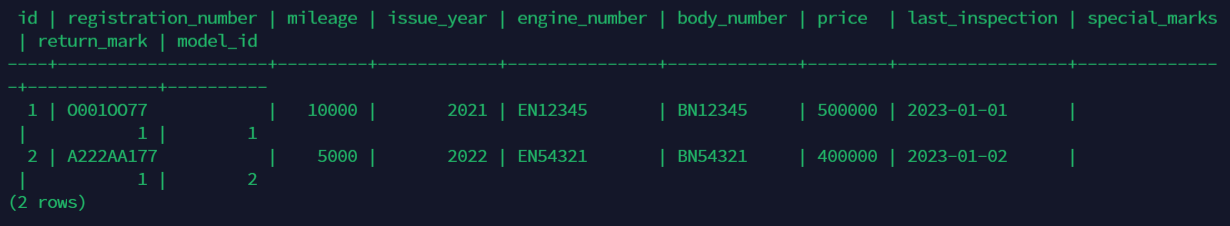
$$;

До:



После:





1. Выдачи автомобиля и расчета стоимости с учетом скидки постоянным клиентам.

**CREATE** **OR** **REPLACE** **PROCEDURE** RentCarWithDiscount(

client\_id **INT**,

car\_id **INT**,

rental\_period **INT**,

deposit **DECIMAL**,

insurance\_id **INT**,

employee\_id **INT**

)

**LANGUAGE** plpgsql

**AS** $$

**DECLARE**

base\_price **DECIMAL**;

discount\_rate **DECIMAL**;

final\_price **DECIMAL**;

**BEGIN**

**SELECT** price **INTO** base\_price

**FROM** rent\_prices

**WHERE** model\_id = (**SELECT** model\_id **FROM** cars **WHERE** id = car\_id)

**AND** start\_datetime <= **CURRENT\_TIMESTAMP**

**AND** (end\_datetime **IS** **NULL** **OR** end\_datetime >= **CURRENT\_TIMESTAMP**)

**LIMIT** 1;

**SELECT** discount **INTO** discount\_rate

**FROM** clients

**WHERE** id = client\_id;

final\_price := base\_price \* rental\_period \* (1 - discount\_rate / 100);

**INSERT** **INTO** contracts (

given\_datetime,

result\_price,

deposit,

returned\_deposit,

payment\_status,

common\_status,

car\_id,

insurance\_id,

employee\_id,

client\_id

) **VALUES** (

**CURRENT\_TIMESTAMP**,

final\_price,

deposit,

**FALSE**,

**FALSE**,

1,

car\_id,

insurance\_id,

employee\_id,

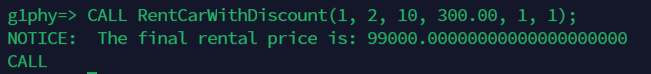
client\_id

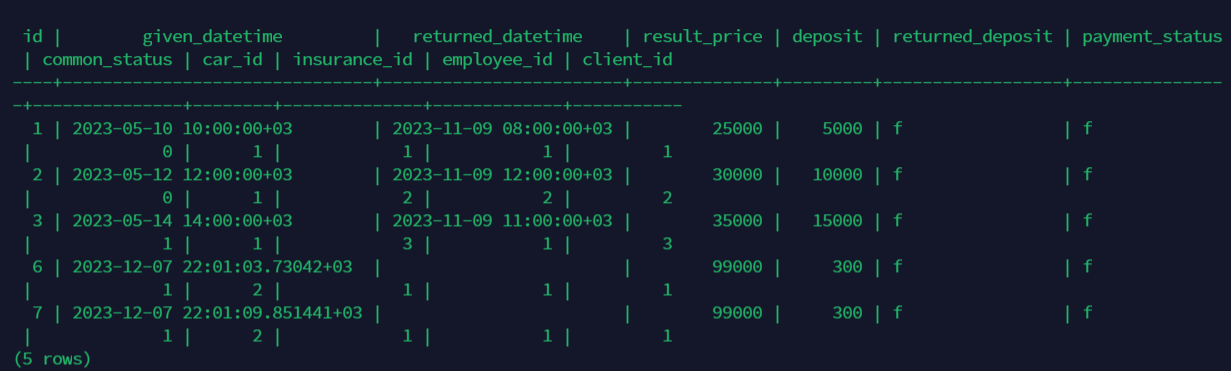
);

RAISE NOTICE 'The final rental price is: %', final\_price;

**END**;

$$;





1. Для вычисления количества автомобилей заданной марки.

**CREATE** **OR** **REPLACE** **PROCEDURE** CountCarsByModel(model\_name TEXT)

**LANGUAGE** plpgsql

**AS** $$

**DECLARE**

car\_count **INT**;

**BEGIN**

**SELECT** **COUNT**(\*)

**INTO** car\_count

**FROM** cars

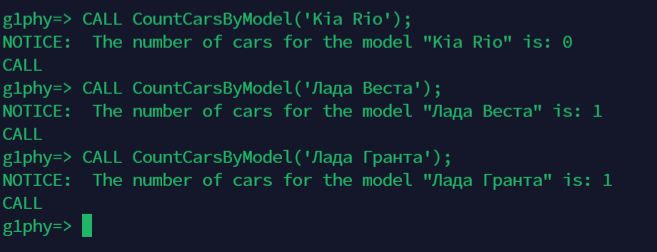
**JOIN** car\_models **ON** cars.model\_id = car\_models.id

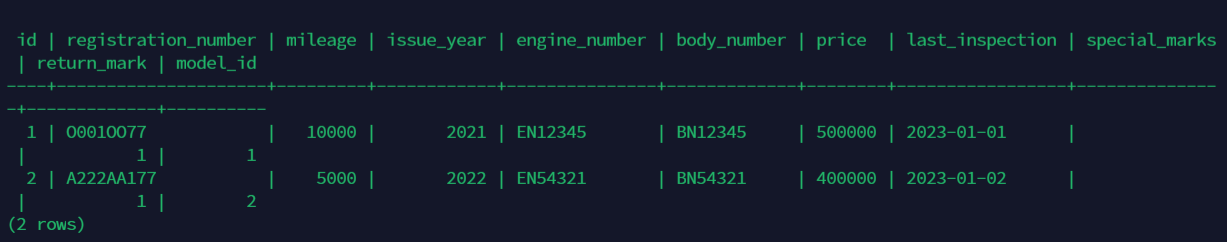
**WHERE** car\_models.name = model\_name;

RAISE NOTICE 'The number of cars for the model "%" is: %', model\_name, car\_count;

**END**;

$$;





# Создать необходимый триггер

Триггер будет автоматически обновлять статус автомобиля на "недоступен" после того, как контракт на его аренду будет создан. Это логичное действие, поскольку автомобиль, который уже арендован, не должен быть доступен для новых аренд до возврата.

**CREATE** **OR** **REPLACE** **FUNCTION** update\_car\_status()

**RETURNS** **TRIGGER** **AS** $$

**BEGIN**

**UPDATE** cars

**SET** return\_mark = 0

**WHERE** id = **NEW**.car\_id;

**RETURN** **NEW**;

**END**;

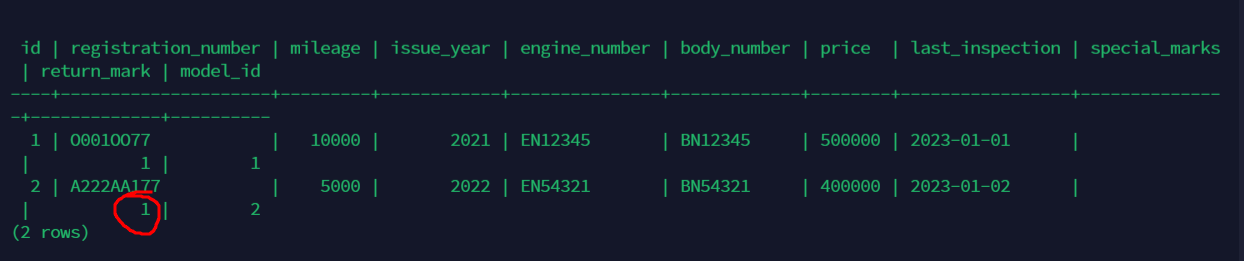
$$ **LANGUAGE** plpgsql;

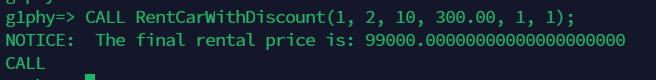
**CREATE** **TRIGGER** trigger\_update\_car\_status

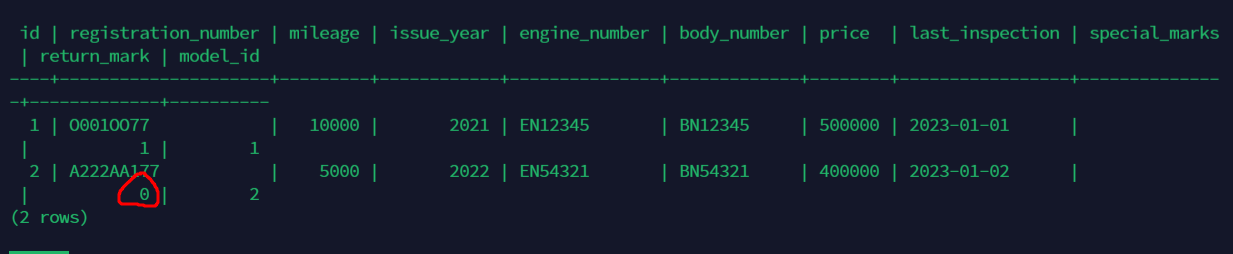
AFTER **INSERT** **ON** contracts

**FOR** EACH **ROW**

**EXECUTE** **FUNCTION** update\_car\_status();







# Вывод

В ходе лабораторной работы были успешно созданы и протестированы процедуры и триггеры в системе управления базами данных PostgreSQL. Работа позволила овладеть практическими навыками программирования в SQL и пониманием механизма триггеров.