

**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»**

Обучающийся Клименков Владислав Максимович
Факультет прикладной информатики
Группа K3241
Направление подготовки 09.03.03 Прикладная информатика
Образовательная программа Мобильные и сетевые технологии 2023
Преподаватель Говорова Марина Михайловна

Санкт-Петербург
2025

1 Цель работы

Овладеть практическими навыками создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

2 Практическое задание

1. Создать 3 процедуры для индивидуальной БД согласно варианту (часть 4 ЛР 2). Допустимо использование IN/OUT параметров. Допустимо создать авторские процедуры. (3 балла)
2. Создать триггеры для индивидуальной БД согласно варианту:

Вариант 2.1. 3 триггера - 3 балла (min). Допустимо использовать триггеры логирования из практического занятия по функциям и триггерам.

Вариант 2.2. 7 оригинальных триггеров - 7 баллов (max).

3 Выполнение

3.1 Вариант и название БД

Вариант 14. «Служба заказа такси»

3.2 Процедуры

№1

Задание:

Создать процедуру для вывода данных о пассажирах, которые заказывали такси в заданном, как параметр, временном интервале.

Код процедуры:

```
-- Создание процедуры
CREATE OR REPLACE FUNCTION get_passengers_by_time_interval(
    start_time_param timestamp without time zone,
    end_time_param timestamp without time zone
)
RETURNS TABLE (
    id integer,
    full_name character varying(100),
    address character varying(255),
    phone character(12),
    passport character varying(20),
```

```

        call_time timestamp without time zone
    )
LANGUAGE SQL
AS $$

SELECT
    p.id,
    p.full_name,
    p.address,
    p.phone,
    p.passport,
    tc.call_time
FROM
    taxi_call tc
JOIN
    person p ON tc.passenger_id = p.id
WHERE
    tc.call_time BETWEEN start_time_param AND end_time_param
ORDER BY
    p.id, tc.call_time;

$$;

-- Вызов процедуры
SELECT * FROM get_passengers_by_time_interval('2024-01-01 00:00:00',
'2025-01-01 00:00:00');

```

Скриншоты выполнения:



```

SQL Shell (psql)
taxi_service=#
taxi_service=#
taxi_service=# CREATE OR REPLACE FUNCTION get_passengers_by_time_interval(
taxi_service(#   start_time_param timestamp without time zone,
taxi_service(#   end_time_param timestamp without time zone
taxi_service(# )
taxi_service=# RETURNS TABLE (
taxi_service(#   id integer,
taxi_service(#   full_name character varying(100),
taxi_service(#   address character varying(255),
taxi_service(#   phone character(12),
taxi_service(#   passport character varying(20),
taxi_service(#   call_time timestamp without time zone
taxi_service(# )
taxi_service=# LANGUAGE SQL
taxi_service=# AS $$
taxi_service$#   SELECT
taxi_service$#       p.id,
taxi_service$#       p.full_name,
taxi_service$#       p.address,
taxi_service$#       p.phone,
taxi_service$#       p.passport,
taxi_service$#       tc.call_time
taxi_service$#   FROM
taxi_service$#       taxi_call tc
taxi_service$#   JOIN
taxi_service$#       person p ON tc.passenger_id = p.id
taxi_service$#   WHERE
taxi_service$#       tc.call_time BETWEEN start_time_param AND end_time_param
taxi_service$#   ORDER BY
taxi_service$#       p.id, tc.call_time;
taxi_service$# $$;
CREATE FUNCTION
taxi_service=#

```

```
SQL Shell (psql)
taxi_service=#
taxi_service=#
taxi_service=# SELECT * FROM get_passengers_by_time_interval('2024-01-01 00:00:00', '2025-01-01 00:00:00');
 id | full_name | address | phone | passport | call_time
-----+-----+-----+-----+-----+-----
201 | Андрон Изотович Михеев | с. Москва, наб. Ватутина, д. 7/7 к. 2/2, 582850 | +74554436816 | 5567 218731 | 2024-01-08 14:48:15
202 | Евфросиния Максимовна Калинина | к. Шенкурск, пр. Демьяна Бедного, д. 480, 248432 | +78861949253 | 1846 652719 | 2024-06-28 16:44:26
207 | Татьяна Макаровна Ларионова | п. Уфа, наб. Восточная, д. 3/2 стр. 5, 674929 | +73803331133 | 8830 112973 | 2024-04-29 16:03:50
207 | Татьяна Макаровна Ларионова | п. Уфа, наб. Восточная, д. 3/2 стр. 5, 674929 | +73803331133 | 8830 112973 | 2024-05-24 09:10:43
209 | Одинцов Дорофей Герасимович | п. Кашира, пер. Высоцкого, д. 2/5 стр. 26, 056539 | +79515892932 | 2505 457575 | 2024-06-13 08:27:01
210 | Сергеев Вадим Георгиевич | г. Киржач, ул. Буденного, д. 32, 191642 | +75388177219 | 2163 163881 | 2024-07-10 00:47:09
213 | Эрнест Ерофеевич Семенов | к. Нижний Тагил, наб. Чехова, д. 6 стр. 8/1, 067247 | +70005310264 | 2805 626496 | 2024-10-04 23:35:14
215 | Соболева Александра Петровна | с. Баксан, алл. Журавлева, д. 5/3 к. 782, 812727 | +74522650493 | 9059 100498 | 2024-02-26 07:21:01
215 | Соболева Александра Петровна | с. Баксан, алл. Журавлева, д. 5/3 к. 782, 812727 | +74522650493 | 9059 100498 | 2024-09-20 00:27:24
222 | Белозерова Виктория Юрьевна | г. Мелеуз, алл. Балтийская, д. 6 стр. 1/2, 309440 | +79985262930 | 5372 425290 | 2024-08-14 16:30:13
225 | Доронина Жанна Станиславовна | д. Салават, алл. Тимирязева, д. 9/7 к. 80, 619086 | +77078571791 | 2566 467874 | 2024-10-03 13:36:20
237 | Ираида Васильевна Тимофеева | п. Новый Уренгой, ул. Нахимова, д. 8 стр. 14, 717362 | +73210296536 | 9416 500671 | 2024-03-06 17:31:22
242 | Милан Марсович Горшков | п. Солнечногорск, алл. Азовская, д. 18, 768431 | +79792460214 | 8271 773495 | 2024-08-09 08:46:46
248 | Г-н Воронов Владимир Чеславович | д. Лагань, ул. Володарского, д. 425, 733877 | +79532133088 | 3212 167111 | 2024-01-09 04:58:45
1258 | 201177 | 2024-01-15 23:02:46 |  |  |  |
273 | Алина Вячеславовна Капустина | клх Владивосток, пер. Флотский, д. 7 к. 5/1, 015603 | +75804345699 | 7036 872951 | 2024-11-04 15:59:40--
274 | Евсеев Гостомысл Виленович | к. Тула, бул. Культуры, д. 6/8, 746497 | +72561229895 | 6715 081187 | 2024-09-07 00:34:15--
284 | Марина Дмитриевна Петрова | д. Сасово, алл. Шмидта, д. 5 к. 2/6, 816013 | +75929531947 | 7688 342507 | 2024-08-25 16:03:17--
284 | Марина Дмитриевна Петрова | д. Сасово, алл. Шмидта, д. 5 к. 2/6, 816013 | +75929531947 | 7688 342507 | 2024-09-25 16:16:11--
293 | Власов Автоном Жанович | п. Токсово, пр. Мусы Джалиля, д. 57 стр. 6/3, 326862 | +76204027946 | 1047 769995 | 2024-01-15 14:07:17--
294 | Леон Бориславович Рыбаков | п. Ханты-Мансийск, пр. Фрунзе, д. 805 стр. 3/3, 179040 | +76092685275 | 7666 989205 | 2024-10-01 22:52:36--
297 | Варвара Кузьминична Носкова | клх Железнодорожск(Курск.), наб. Сиреневая, д. 739 к. 140, 918260 | +71870522196 | 5034 228711 | 2024-06-01 07:46:57--
(22 строки)

taxi_service=#
taxi_service=#
```

№2

Задание:

Создать процедуру, выводящую сведения о том, куда был доставлен пассажир по заданному номеру телефона пассажира.

Код процедуры:

```
-- Создание процедуры
CREATE OR REPLACE FUNCTION get_passenger_end_places(passenger_phone
character(12))
RETURNS TABLE (
    person_id integer,
    full_name character varying(100),
    phone character(12),
    taxi_call_id integer,
    end_place character varying(200),
    end_time timestamp without time zone
)
LANGUAGE SQL
AS $$
SELECT
    p.id AS person_id,
    p.full_name,
    p.phone,
    tc.id AS taxi_call_id,
    tc.end_place,
```

```

        tc.end_time
FROM
    taxi_call tc
JOIN
    person p ON tc.passenger_id = p.id
WHERE
    p.phone = passenger_phone
ORDER BY
    tc.end_time DESC;
$$;

-- Вызов процедуры
SELECT * FROM get_passenger_end_places('+799999162343');

```

Скриншоты выполнения:

```

SQL Shell (psql)
taxi_service=#
taxi_service=#
taxi_service=# CREATE OR REPLACE FUNCTION get_passenger_end_places(passenger_phone character(12))
taxi_service=# RETURNS TABLE (
taxi_service(#      person_id integer,
taxi_service(#      full_name character varying(100),
taxi_service(#      phone character(12),
taxi_service(#      taxi_call_id integer,
taxi_service(#      end_place character varying(200),
taxi_service(#      end_time timestamp without time zone
taxi_service(# )
taxi_service=# LANGUAGE SQL
taxi_service=# AS $$
taxi_service$$      SELECT
taxi_service$$          p.id AS person_id,
taxi_service$$          p.full_name,
taxi_service$$          p.phone,
taxi_service$$          tc.id AS taxi_call_id,
taxi_service$$          tc.end_place,
taxi_service$$          tc.end_time
taxi_service$$      FROM
taxi_service$$          taxi_call tc
taxi_service$$      JOIN
taxi_service$$          person p ON tc.passenger_id = p.id
taxi_service$$      WHERE
taxi_service$$          p.phone = passenger_phone
taxi_service$$      ORDER BY
taxi_service$$          tc.end_time DESC;
taxi_service$$ $$;
CREATE FUNCTION
taxi_service=#

```

```

SQL Shell (psql)
taxi_service=#
taxi_service=#
taxi_service=#
taxi_service=# SELECT * FROM get_passenger_end_places('+799999162343');
 person_id |      full_name      | phone | taxi_call_id |      end_place      |      end_time
-----+-----+-----+-----+-----+-----
      205 | Таисия Евгеньевна Фролова | +799999162343 |      190 | ул. Восточная | 2020-07-30 04:04:29
      205 | Таисия Евгеньевна Фролова | +799999162343 |      305 | ул. Восточная | 2020-03-12 10:54:49
      205 | Таисия Евгеньевна Фролова | +799999162343 |      191 | п. Абинск, ул. Горная, д. 22, 171168 | 2020-01-08 22:42:22
      205 | Таисия Евгеньевна Фролова | +799999162343 |      272 | ст. Льгов, алл. Зеленая, д. 52, 251282 | 2007-05-29 17:20:09
      205 | Таисия Евгеньевна Фролова | +799999162343 |       32 | с. Теберда, пр. Ангарский, д. 7 стр. 21, 675210 | 2006-12-26 16:48:01
(5 строк)

taxi_service=#
taxi_service=#

```

№3

Задание:

Создать процедуру для вычисления суммарного дохода таксопарка за истекший месяц.

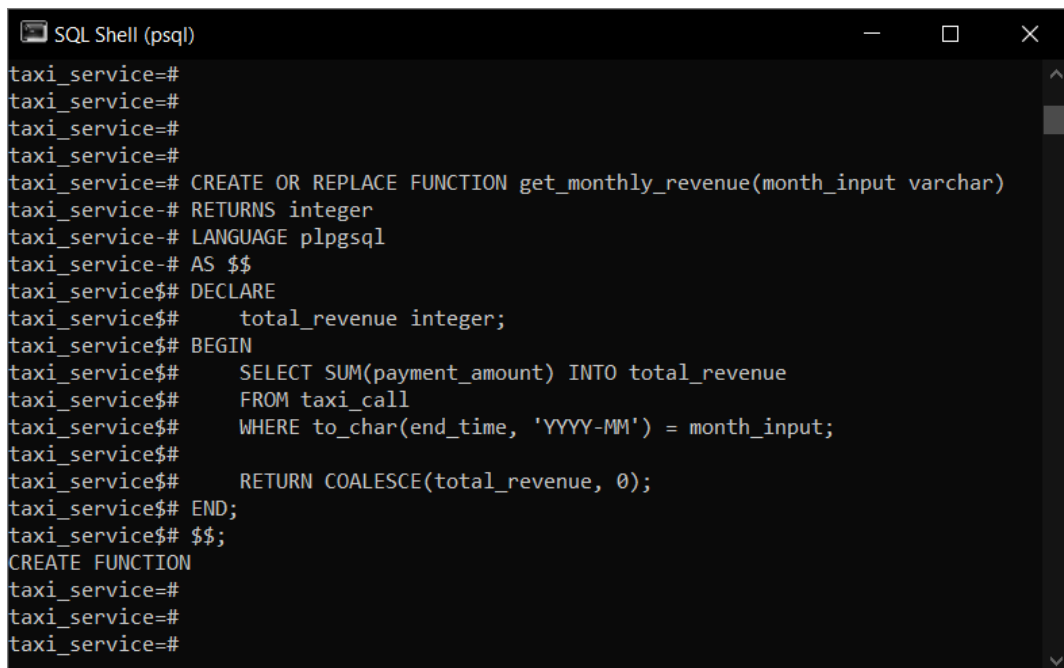
Код процедуры:

```
-- Создание процедуры
CREATE OR REPLACE FUNCTION get_monthly_revenue(month_input varchar)
RETURNS integer
LANGUAGE plpgsql
AS $$
DECLARE
    total_revenue integer;
BEGIN
    SELECT SUM(payment_amount) INTO total_revenue
    FROM taxi_call
    WHERE to_char(end_time, 'YYYY-MM') = month_input;

    RETURN COALESCE(total_revenue, 0);
END;
$$;

-- Вызов процедуры
SELECT get_monthly_revenue('2025-03');
```

Скриншоты выполнения:



```
SQL Shell (psql)
taxi_service=#
taxi_service=#
taxi_service=#
taxi_service=#
taxi_service=# CREATE OR REPLACE FUNCTION get_monthly_revenue(month_input varchar)
taxi_service=# RETURNS integer
taxi_service=# LANGUAGE plpgsql
taxi_service=# AS $$
taxi_service$# DECLARE
taxi_service$#     total_revenue integer;
taxi_service$# BEGIN
taxi_service$#     SELECT SUM(payment_amount) INTO total_revenue
taxi_service$#     FROM taxi_call
taxi_service$#     WHERE to_char(end_time, 'YYYY-MM') = month_input;
taxi_service$#
taxi_service$#     RETURN COALESCE(total_revenue, 0);
taxi_service$# END;
taxi_service$# $$;
CREATE FUNCTION
taxi_service=#
taxi_service=#
taxi_service=#
```

```
SQL Shell (psql)
taxi_service=#
taxi_service=#
taxi_service=#
taxi_service=# SELECT get_monthly_revenue('2025-03');
get_monthly_revenue
-----
              7446
(1 строка)

taxi_service=#
taxi_service=#
```

3.3 Триггеры

№1

Описание триггера:

Триггер для логирования всех изменений в таблице “payment” в отдельную таблицу.

Код триггера:

```
-- Создание таблицы логгирования
CREATE TABLE payment_log (
    id SERIAL PRIMARY KEY,
    operation_type VARCHAR(10) NOT NULL,
    column_name VARCHAR(50) NOT NULL,
    payment_id INTEGER NOT NULL,
    old_value TEXT,
    new_value TEXT,
    change_date TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);

-- Создание триггера
CREATE OR REPLACE FUNCTION log_payment_changes()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'UPDATE' THEN
        IF OLD.payment_type_id IS DISTINCT FROM NEW.payment_type_id THEN
            INSERT INTO payment_log (operation_type, column_name, payment_id,
old_value, new_value)
                VALUES ('UPDATE', 'payment_type_id', OLD.id,
OLD.payment_type_id::TEXT, NEW.payment_type_id::TEXT);
```

```

END IF;

IF OLD.employee_id IS DISTINCT FROM NEW.employee_id THEN
    INSERT INTO payment_log (operation_type, column_name, payment_id,
old_value, new_value)
        VALUES ('UPDATE', 'employee_id', OLD.id, OLD.employee_id::TEXT,
NEW.employee_id::TEXT);
    END IF;

IF OLD.amount IS DISTINCT FROM NEW.amount THEN
    INSERT INTO payment_log (operation_type, column_name, payment_id,
old_value, new_value)
        VALUES ('UPDATE', 'amount', OLD.id, OLD.amount::TEXT,
NEW.amount::TEXT);
    END IF;

IF OLD.date IS DISTINCT FROM NEW.date THEN
    INSERT INTO payment_log (operation_type, column_name, payment_id,
old_value, new_value)
        VALUES ('UPDATE', 'date', OLD.id, OLD.date::TEXT, NEW.date::TEXT);
    END IF;

ELSIF TG_OP = 'DELETE' THEN
    INSERT INTO payment_log (operation_type, column_name, payment_id,
old_value, new_value)
        VALUES
            ('DELETE', 'payment_type_id', OLD.id, OLD.payment_type_id::TEXT,
NULL),
            ('DELETE', 'employee_id', OLD.id, OLD.employee_id::TEXT, NULL),
            ('DELETE', 'amount', OLD.id, OLD.amount::TEXT, NULL),
            ('DELETE', 'date', OLD.id, OLD.date::TEXT, NULL);

ELSIF TG_OP = 'INSERT' THEN
    INSERT INTO payment_log (operation_type, column_name, payment_id,
old_value, new_value)
        VALUES
            ('INSERT', 'payment_type_id', NEW.id, NULL,
NEW.payment_type_id::TEXT),
            ('INSERT', 'employee_id', NEW.id, NULL, NEW.employee_id::TEXT),
            ('INSERT', 'amount', NEW.id, NULL, NEW.amount::TEXT),
            ('INSERT', 'date', NEW.id, NULL, NEW.date::TEXT);
    END IF;

RETURN NEW;

```



```

END;

$$ LANGUAGE plpgsql;

-- Привязка триггера к таблице
CREATE TRIGGER log_payment_changes_trigger
AFTER INSERT OR UPDATE OR DELETE ON payment
FOR EACH ROW EXECUTE FUNCTION log_payment_changes();

-- Проверка триггера
INSERT INTO payment (payment_type_id, employee_id, amount, date)
VALUES (1, 1, 70000, '2025-01-01');

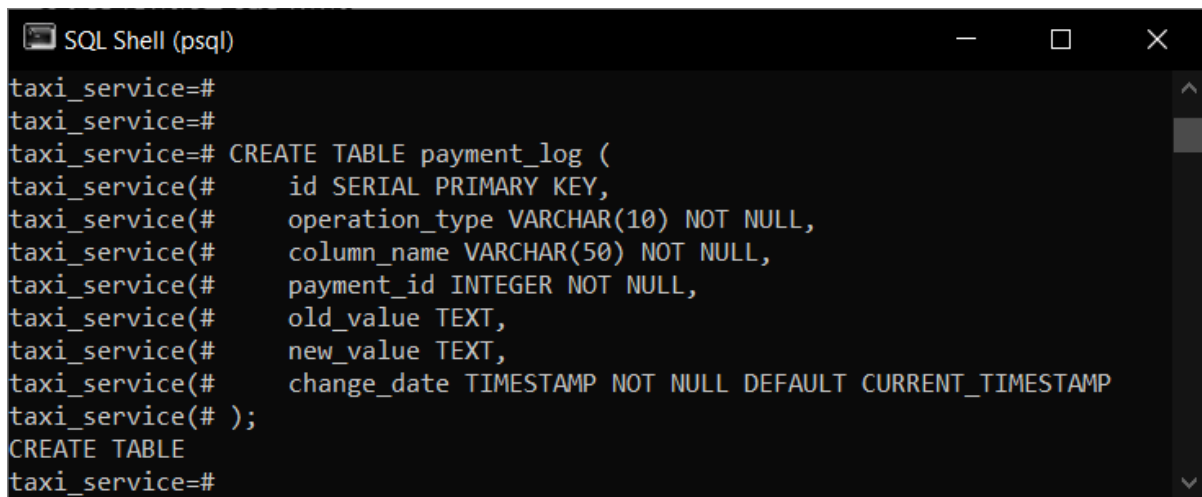
UPDATE payment SET
amount = 80000,
date = '2025-01-02'
WHERE id = 7785629;

DELETE FROM payment
WHERE id = 7785629;

SELECT * FROM payment_log;

```

Скриншоты выполнения:



```

SQL Shell (psql)
taxi_service=#
taxi_service=#
taxi_service=# CREATE TABLE payment_log (
taxi_service(#      id SERIAL PRIMARY KEY,
taxi_service(#      operation_type VARCHAR(10) NOT NULL,
taxi_service(#      column_name VARCHAR(50) NOT NULL,
taxi_service(#      payment_id INTEGER NOT NULL,
taxi_service(#      old_value TEXT,
taxi_service(#      new_value TEXT,
taxi_service(#      change_date TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
taxi_service(# );
CREATE TABLE
taxi_service=#

```

```

SQL Shell (psql)
taxi_service=#
taxi_service=#
taxi_service=# CREATE OR REPLACE FUNCTION log_payment_changes()
taxi_service-# RETURNS TRIGGER AS $$
taxi_service$# BEGIN
taxi_service$#     IF TG_OP = 'UPDATE' THEN
taxi_service$#         IF OLD.payment_type_id IS DISTINCT FROM NEW.payment_type_id THEN
taxi_service$#             INSERT INTO payment_log (operation_type, column_name, payment_id, old_value, new_value)
taxi_service$#                 VALUES ('UPDATE', 'payment_type_id', OLD.id, OLD.payment_type_id::TEXT, NEW.payment_type_id::TEXT);
taxi_service$#         END IF;
taxi_service$#
taxi_service$#         IF OLD.employee_id IS DISTINCT FROM NEW.employee_id THEN
taxi_service$#             INSERT INTO payment_log (operation_type, column_name, payment_id, old_value, new_value)
taxi_service$#                 VALUES ('UPDATE', 'employee_id', OLD.id, OLD.employee_id::TEXT, NEW.employee_id::TEXT);
taxi_service$#         END IF;
taxi_service$#
taxi_service$#         IF OLD.amount IS DISTINCT FROM NEW.amount THEN
taxi_service$#             INSERT INTO payment_log (operation_type, column_name, payment_id, old_value, new_value)
taxi_service$#                 VALUES ('UPDATE', 'amount', OLD.id, OLD.amount::TEXT, NEW.amount::TEXT);
taxi_service$#         END IF;
taxi_service$#
taxi_service$#         IF OLD.date IS DISTINCT FROM NEW.date THEN
taxi_service$#             INSERT INTO payment_log (operation_type, column_name, payment_id, old_value, new_value)
taxi_service$#                 VALUES ('UPDATE', 'date', OLD.id, OLD.date::TEXT, NEW.date::TEXT);
taxi_service$#         END IF;
taxi_service$#
taxi_service$#     ELSIF TG_OP = 'DELETE' THEN
taxi_service$#         INSERT INTO payment_log (operation_type, column_name, payment_id, old_value, new_value)
taxi_service$#             VALUES
taxi_service$#                 ('DELETE', 'payment_type_id', OLD.id, OLD.payment_type_id::TEXT, NULL),
taxi_service$#                 ('DELETE', 'employee_id', OLD.id, OLD.employee_id::TEXT, NULL),
taxi_service$#                 ('DELETE', 'amount', OLD.id, OLD.amount::TEXT, NULL),
taxi_service$#                 ('DELETE', 'date', OLD.id, OLD.date::TEXT, NULL);
taxi_service$#
taxi_service$#     ELSIF TG_OP = 'INSERT' THEN
taxi_service$#         INSERT INTO payment_log (operation_type, column_name, payment_id, old_value, new_value)
taxi_service$#             VALUES
taxi_service$#                 ('INSERT', 'payment_type_id', NEW.id, NULL, NEW.payment_type_id::TEXT),
taxi_service$#                 ('INSERT', 'employee_id', NEW.id, NULL, NEW.employee_id::TEXT),
taxi_service$#                 ('INSERT', 'amount', NEW.id, NULL, NEW.amount::TEXT),
taxi_service$#                 ('INSERT', 'date', NEW.id, NULL, NEW.date::TEXT);
taxi_service$#     END IF;
taxi_service$#
taxi_service$#     RETURN NEW;
taxi_service$# END;
taxi_service$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
taxi_service=#

```

```

SQL Shell (psql)
taxi_service=#
taxi_service=#
taxi_service=#
taxi_service=#
taxi_service=#
taxi_service=#
taxi_service=# CREATE TRIGGER log_payment_changes_trigger
taxi_service-# AFTER INSERT OR UPDATE OR DELETE ON payment
taxi_service-# FOR EACH ROW EXECUTE FUNCTION log_payment_changes();
CREATE TRIGGER
taxi_service=#
taxi_service=#
taxi_service=#
taxi_service=#
taxi_service=#
taxi_service=#

```

```
SQL Shell (psql)
taxi_service=#
taxi_service=#
taxi_service=# INSERT INTO payment (payment_type_id, employee_id, amount, date)
taxi_service=# VALUES (1, 1, 70000, '2025-01-01');
INSERT 0 1
taxi_service=# UPDATE payment SET
taxi_service=# amount = 80000,
taxi_service=# date = '2025-01-02'
taxi_service=# WHERE id = 7785629;
UPDATE 1
taxi_service=# DELETE FROM payment
taxi_service=# WHERE id = 7785629;
DELETE 1
```

```
SQL Shell (psql)
taxi_service=#
taxi_service=#
taxi_service=# SELECT * FROM payment_log;
 id | operation_type | column_name | payment_id | old_value | new_value | change_date
-----+-----+-----+-----+-----+-----+-----
  1 | INSERT        | payment_type_id | 7785629 |          | 1         | 2025-05-14 21:06:23.655198
  2 | INSERT        | employee_id     | 7785629 |          | 1         | 2025-05-14 21:06:23.655198
  3 | INSERT        | amount         | 7785629 |          | 70000     | 2025-05-14 21:06:23.655198
  4 | INSERT        | date           | 7785629 |          | 2025-01-01 | 2025-05-14 21:06:23.655198
  5 | UPDATE        | amount         | 7785629 | 70000    | 80000     | 2025-05-14 21:10:40.990594
  6 | UPDATE        | date           | 7785629 | 2025-01-01 | 2025-01-02 | 2025-05-14 21:10:40.990594
  7 | DELETE        | payment_type_id | 7785629 | 1         |           | 2025-05-14 21:10:56.788537
  8 | DELETE        | employee_id     | 7785629 | 1         |           | 2025-05-14 21:10:56.788537
  9 | DELETE        | amount         | 7785629 | 80000    |           | 2025-05-14 21:10:56.788537
 10 | DELETE        | date           | 7785629 | 2025-01-02 |           | 2025-05-14 21:10:56.788537
(10 строк)

taxi_service=#
taxi_service=#
```

№2

Описание триггера:

Триггер для автоматического расчёта стоимости поездки с учётом всех коэффициентов и скидки пассажира, если данное значение не задано вручную.

Код триггера:

```
-- Создание триггера
CREATE OR REPLACE FUNCTION calculate_payment_amount()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.payment_amount IS NULL THEN
        NEW.payment_amount := (
            SELECT
                NEW.distance *
                tp.price_per_km *
                ca.ratio *
```

```

        (1 - COALESCE(p.discount, 0))
FROM
    tariff_price tp
    JOIN cost_adjustment ca ON ca.id = NEW.cost_adjustment_id
    JOIN passenger p ON p.person_id = NEW.passenger_id
WHERE
    tp.id = NEW.tariff_price_id
);

NEW.payment_amount := ROUND(NEW.payment_amount);
END IF;

RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Привязка триггера к таблице
CREATE TRIGGER calculate_payment_amount_trigger
BEFORE INSERT ON taxi_call
FOR EACH ROW
EXECUTE FUNCTION calculate_payment_amount();

-- Проверка триггера
INSERT INTO taxi_call (driver_id, tariff_price_id, cost_adjustment_id,
passenger_id, administrator_id, start_time, end_time, start_place, end_place,
distance, payment_method, trip_rating, trip_review, call_time)
VALUES (105, 39, 99, 284, 58, '2025-05-14 10:00:00', '2025-05-14 10:30:00',
'Улица Ленина, 1', 'Улица Пушкина, 2', 25, 'карта', 5, 'Отличная поездка!',
'2025-05-14 09:50:00');

SELECT id, payment_amount, start_place, end_place, distance FROM taxi_call
ORDER BY id DESC;

```

Скриншоты выполнения:

```
SQL Shell (psql)
taxi_service=#
taxi_service=# CREATE OR REPLACE FUNCTION calculate_payment_amount()
taxi_service=# RETURNS TRIGGER AS $$
taxi_service$# BEGIN
taxi_service$#     IF NEW.payment_amount IS NULL THEN
taxi_service$#         NEW.payment_amount := (
taxi_service$#             SELECT
taxi_service$#                 NEW.distance *
taxi_service$#                 tp.price_per_km *
taxi_service$#                 ca.ratio *
taxi_service$#                 (1 - COALESCE(p.discount, 0))
taxi_service$#             FROM
taxi_service$#                 tariff_price tp
taxi_service$#                 JOIN cost_adjustment ca ON ca.id = NEW.cost_adjustment_id
taxi_service$#                 JOIN passenger p ON p.person_id = NEW.passenger_id
taxi_service$#             WHERE
taxi_service$#                 tp.id = NEW.tariff_price_id
taxi_service$#         );
taxi_service$#         NEW.payment_amount := ROUND(NEW.payment_amount);
taxi_service$#     END IF;
taxi_service$#     RETURN NEW;
taxi_service$# END;
taxi_service$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
taxi_service=#
```

```
SQL Shell (psql)
taxi_service=#
taxi_service=#
taxi_service=#
taxi_service=# CREATE TRIGGER calculate_payment_amount_trigger
taxi_service=# BEFORE INSERT ON taxi_call
taxi_service=# FOR EACH ROW
taxi_service=# EXECUTE FUNCTION calculate_payment_amount();
CREATE TRIGGER
taxi_service=#
taxi_service=#
taxi_service=#
```

```
SQL Shell (psql)
taxi_service=#
taxi_service=#
taxi_service=#
taxi_service=# INSERT INTO taxi_call (driver_id, tariff_price_id, cost_adjustment_id, passenger_id, administrator_id, start_time, end_time, start_place, end_place, distance, payment_method, trip_rating, trip_review, call_time)
taxi_service=# VALUES (105, 39, 99, 284, 58, '2025-05-14 10:00:00', '2025-05-14 10:30:00', 'Улица Ленина, 1', 'Улица Пушкина, 2', 25, 'карта', 5, 'Отличная поездка!', '2025-05-14 09:50:00');
INSERT 0 1
```

```

Выбрать SQL Shell (psql)
taxi_service=#
taxi_service=#
taxi_service=#
taxi_service=#
taxi_service=# SELECT id, payment_amount, start_place, end_place, distance FROM taxi_call
taxi_service=# ORDER BY id DESC;
 id | payment_amount | start_place | end_place | distance
-----+-----+-----+-----+-----
 414 |         1650 | Улица Ленина, 1 | Улица Пушкина, 2 |         25
 412 |        3300 | ст. Шелехов, бул. Геологический, д. 3 стр. 953, 331122 | ст. Урай, бул. Строительный, д. 9, 892964 |         50
 411 |        1984 | д. Кизилюрт, пр. Курский, д. 7/4 к. 8/3, 471825 | д. |        107
      |              |              | Муром, бул. Нефтяников, д. 9 к. 46, 769290 |
 410 |        2600 | г. Чебоксары, ул. Малая, д. 377, 635180 | ул. Восточная |        164
 409 |         364 | к. Цимлянск, бул. Глинки, д. 37 к. 2/7, 486081 | г. Нефедова, пр. Правды, д. 9/5, 818035 |        169
 408 |       2051 | г. Теберда, алл. Добролюбова, д. 85 стр. 938, 309682 | г. Переславль-Залесский, пр. Макаренко, д. 9/3 к. 2, 628994 |         11
 407 |         951 | к. Белоярский, ул. Просторная, д. 847 стр. 101, 416359 | ст. Октябрьское (Челяб.), алл. Леонова, д. 82 стр. 9, 807417 |         73
 406 |       2478 | клх Кропоткин (Краснод.), пр. Технический, д. 1, 637997 | клх Надым, ул. Маяковского, д. 8/4 стр. 4/8, 795098 |        200
^Сигнал отмены отправлен
taxi_service=#
taxi_service=#
taxi_service=#

```

№3

Описание триггера:

Триггер для блокировки попыток вставить в таблицу “car_usage” запись об использовании водителем автомобиля, если в таблице уже есть запись с тем же водителем и автомобилем, диапазон использования у которой пересекается с таковым у вставляемой записи.

Код триггера:

```

-- Создание триггера
CREATE OR REPLACE FUNCTION check_car_unique_usage()
RETURNS TRIGGER AS $$
BEGIN
    IF EXISTS (
        SELECT 1
        FROM car_usage
        WHERE driver_id = NEW.driver_id
            AND car_id = NEW.car_id
            AND (NEW.start_date <= end_date AND NEW.end_date >= start_date)
    ) THEN
        RAISE EXCEPTION 'Водитель % уже использует автомобиль % в период с % по %',
            NEW.driver_id, NEW.car_id, NEW.start_date, NEW.end_date;
    END IF;

    RETURN NEW;
END;

$$ LANGUAGE plpgsql;

-- Привязка триггера к таблице
CREATE TRIGGER check_car_unique_usage_trigger

```

```

BEFORE INSERT ON car_usage
FOR EACH ROW
EXECUTE FUNCTION check_car_unique_usage();

-- Проверка триггера
INSERT INTO car_usage (driver_id, car_id, start_date, end_date)
VALUES (119, 71, '2025-01-01', '2026-01-01');

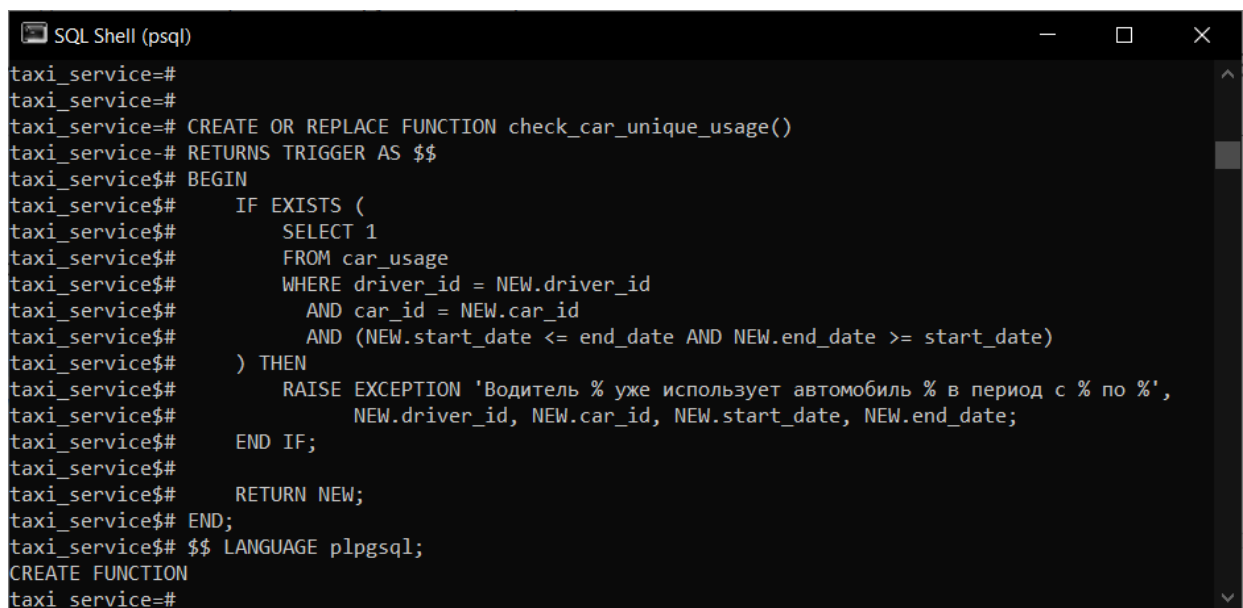
INSERT INTO car_usage (driver_id, car_id, start_date, end_date)
VALUES (119, 71, '2024-02-01', '2025-02-01');

INSERT INTO car_usage (driver_id, car_id, start_date, end_date)
VALUES (119, 71, '2025-02-01', '2027-02-01');

INSERT INTO car_usage (driver_id, car_id, start_date, end_date)
VALUES (119, 71, '2026-02-01', '2027-02-01');

```

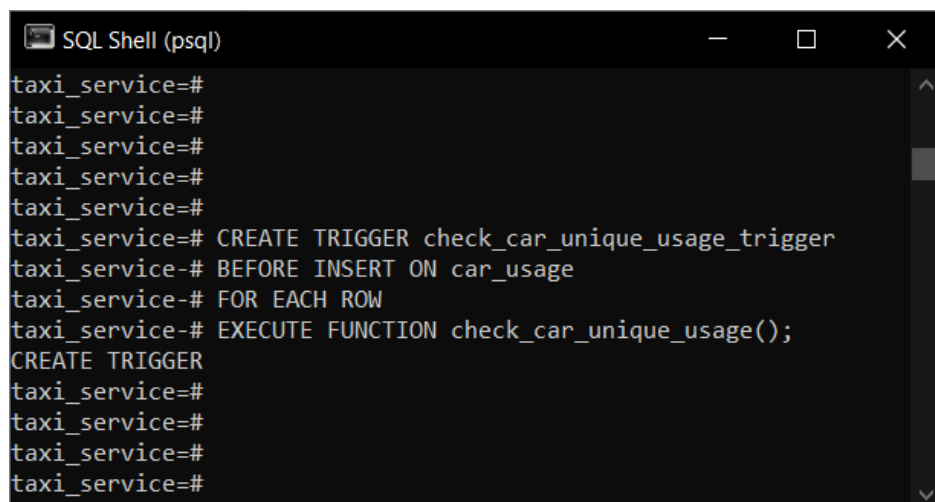
Скриншоты выполнения:



```

SQL Shell (psql)
taxi_service=#
taxi_service=#
taxi_service=# CREATE OR REPLACE FUNCTION check_car_unique_usage()
taxi_service=# RETURNS TRIGGER AS $$
taxi_service$# BEGIN
taxi_service$#     IF EXISTS (
taxi_service$#         SELECT 1
taxi_service$#         FROM car_usage
taxi_service$#         WHERE driver_id = NEW.driver_id
taxi_service$#             AND car_id = NEW.car_id
taxi_service$#             AND (NEW.start_date <= end_date AND NEW.end_date >= start_date)
taxi_service$#     ) THEN
taxi_service$#         RAISE EXCEPTION 'Водитель % уже использует автомобиль % в период с % по %',
taxi_service$#             NEW.driver_id, NEW.car_id, NEW.start_date, NEW.end_date;
taxi_service$#     END IF;
taxi_service$#     RETURN NEW;
taxi_service$# END;
taxi_service$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
taxi_service=#

```



```

SQL Shell (psql)
taxi_service=#
taxi_service=#
taxi_service=#
taxi_service=#
taxi_service=#
taxi_service=# CREATE TRIGGER check_car_unique_usage_trigger
taxi_service=# BEFORE INSERT ON car_usage
taxi_service=# FOR EACH ROW
taxi_service=# EXECUTE FUNCTION check_car_unique_usage();
CREATE TRIGGER
taxi_service=#
taxi_service=#
taxi_service=#
taxi_service=#

```

```
SQL Shell (psql)
taxi_service=#
taxi_service=#
taxi_service=# INSERT INTO car_usage (driver_id, car_id, start_date, end_date)
taxi_service-# VALUES (119, 71, '2025-01-01', '2026-01-01');
INSERT 0 1
taxi_service=# INSERT INTO car_usage (driver_id, car_id, start_date, end_date)
taxi_service-# VALUES (119, 71, '2024-02-01', '2025-02-01');
ОШИБКА: Водитель 119 уже использует автомобиль 71 в период с 2024-02-01 по 2025-02-01
КОНТЕКСТ: функция PL/pgSQL check_car_unique_usage(), строка 10, оператор RAISE
taxi_service=# INSERT INTO car_usage (driver_id, car_id, start_date, end_date)
taxi_service-# VALUES (119, 71, '2025-02-01', '2027-02-01');
ОШИБКА: Водитель 119 уже использует автомобиль 71 в период с 2025-02-01 по 2027-02-01
КОНТЕКСТ: функция PL/pgSQL check_car_unique_usage(), строка 10, оператор RAISE
taxi_service=# INSERT INTO car_usage (driver_id, car_id, start_date, end_date)
taxi_service-# VALUES (119, 71, '2026-02-01', '2027-02-01');
INSERT 0 1
taxi_service=#
```

4 Выводы

В рамках данной работы я получил практические навыки использования процедур, функций и триггеров в базе данных PostgreSQL, создав согласно заданию три процедуры и три триггера для своей базы данных.