

**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: Хисаметдинова Д.Н.

Факультет: ИКТ

Группа: К3241

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

## Оглавление

Цель работы .....	3
Практическое задание .....	3
Вариант 17. БД «Телефонный провайдер» .....	3
Выполнение ЛР .....	<b>Error! Bookmark not defined.</b>
Вывод по работе .....	<b>Error! Bookmark not defined.</b>

## Цель работы

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

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

## Практическое задание

### Вариант 2 (max - 8 баллов)

1. Создать процедуры/функции согласно индивидуальному заданию (часть 4).
2. Создать авторский триггер по варианту индивидуального задания.

### Вариант 17. БД «Телефонный провайдер»

**Описание предметной области:** Информационная система служит для хранения информации об абонентах телефонной компании и для учета оплаты всех видов услуг абонентами.

Каждый абонент подключен к определенному тарифу. Тариф определяет базовое количество минут, ГБт, смс. Кроме того, он может подключить дополнительные услуги за отдельную плату. Необходимо знать текущий баланс клиента. У клиента могут быть подключены сторонние ресурсы, требующие оплаты, не зависящие от текущего тарифа.

Клиент может менять тариф.

В системе должны храниться сведения о продолжительности разговоров каждого абонента, о стоимости внутренних и междугородных переговоров, о задолженности абонента.

БД должна содержать следующий минимальный набор сведений: ФИО абонента. Номер телефона. Адрес абонента. Город. Зона (город, республика, СНГ, дальнее зарубежье). Страна. Стоимость тарифа. Сроки действия тарифа. Продолжительность разговора в минутах. Дата звонка. Время звонка. Код зоны. Цена минуты. Сумма оплаты. Дата оплаты. Статус оплаты. Дата фактической оплаты.

**Задание 4.** Создать хранимые процедуры:

- вывести список всех звонков заданного абонента.
- вывести задолженность по оплате для заданного абонента.
- рассчитать общую стоимость звонков по каждой зоне за истекшую неделю.

**Задание 5.** Создать необходимые триггеры.

## Выполнение практического задания

### Создайте хранимые процедуры:

- вывести список всех звонков заданного абонента.

```
CREATE OR REPLACE PROCEDURE get_client_calls(client_id_param INTEGER, OUT calls_cursor
REFCURSOR)
LANGUAGE plpgsql
AS $$
BEGIN
    OPEN calls_cursor FOR
        SELECT 'domestic' AS call_type, dc.domestic_call_id, dc.phone_number,
dc.call_start_time, dc.call_end_time, dc.domestic_callee_number, NULL AS call_zone_id
        FROM domestic_call dc
        JOIN phone_on_tariff pot ON dc.phone_number = pot.phone_on_tariff_number
        JOIN contract c ON pot.contract_id = c.contract_id
        WHERE c.client_id = client_id_param
        UNION
        SELECT 'international' AS call_type, ic.international_call_id, ic.phone_number,
ic.international_call_start_time, ic.international_call_end_time,
ic.international_callee_number, ic.call_zone_id
        FROM international_call ic
        JOIN phone_on_tariff pot ON ic.phone_number = pot.phone_on_tariff_number
        JOIN contract c ON pot.contract_id = c.contract_id
        WHERE c.client_id = client_id_param;
END;
$$;
```

```
DROP PROCEDURE
phone_provider=# CREATE OR REPLACE PROCEDURE get_client_calls(client_id_param INTEGER, OUT calls_cursor REFCURSOR) LA
NGUAGE plpgsql AS $$ BEGIN OPEN calls_cursor FOR SELECT 'domestic' AS call_type, dc.domestic_call_id, dc.phone_number
, dc.call_start_time, dc.call_end_time, dc.domestic_callee_number FROM domestic_call dc JOIN phone_on_tariff pot ON d
c.phone_number = pot.phone_on_tariff_number JOIN contract c ON pot.contract_id = c.contract_id WHERE c.client_id = cl
ient_id_param UNION SELECT 'international' AS call_type, ic.international_call_id, ic.phone_number, ic.international_
call_start_time, ic.international_call_end_time, ic.international_callee_number FROM international_call ic JOIN phone_
on_tariff pot ON ic.phone_number = pot.phone_on_tariff_number JOIN contract c ON pot.contract_id = c.contract_id WHE
RE c.client_id = client_id_param; END; $$;
CREATE PROCEDURE
```

Теперь выведу все звонки для абонента с client\_id = 5:

(Так как у абонента могут быть и домашние, и международные звонки, задам переменную v\_call\_type, отвечающую за тип звонка, и, так как если звонок международный, у домашних звонков сделаю v\_call\_zone\_id по умолчанию NULL)

```
DO $$
DECLARE
    my_cursor REFCURSOR;
    v_call_type VARCHAR;
    v_call_id INTEGER;
    v_phone_number VARCHAR;
```

```

v_start_time TIMESTAMP;
v_end_time TIMESTAMP;
v_callee_number VARCHAR;
v_call_zone_id INTEGER;
BEGIN
    CALL get_client_calls(5, my_cursor);

    LOOP
        FETCH my_cursor INTO v_call_type, v_call_id, v_phone_number, v_start_time, v_end_time,
v_callee_number, v_call_zone_id;
        EXIT WHEN NOT FOUND;

        RAISE NOTICE 'Type: %, ID: %, Phone Number: %, Start Time: %, End Time: %, Callee
Number: %, Zone ID: %',
            v_call_type, v_call_id, v_phone_number, v_start_time, v_end_time,
v_callee_number, v_call_zone_id;
    END LOOP;

    CLOSE my_cursor;
END;
$$;

```

```

CREATE OR REPLACE PROCEDURE
phone_provider=# DO $$ DECLARE my_cursor REFCURSOR; v_call_type VARCHAR; v_call_id INTEGER; v_phone_number VARCHAR; v
_start_time TIMESTAMP; v_end_time TIMESTAMP; v_callee_number VARCHAR; v_call_zone_id INTEGER; BEGIN CALL get_client_c
alls(5, my_cursor); LOOP FETCH my_cursor INTO v_call_type, v_call_id, v_phone_number, v_start_time, v_end_time, v_cal
lee_number, v_call_zone_id; EXIT WHEN NOT FOUND; RAISE NOTICE 'Type: %, ID: %, Phone Number: %, Start Time: %, End Ti
me: %, Callee Number: %, Zone ID: %', v_call_type, v_call_id, v_phone_number, v_start_time, v_end_time, v_callee_numb
er, v_call_zone_id; END LOOP; CLOSE my_cursor; END; $$;
ЗАМЕЧАНИЕ: Type: domestic, ID: 18, Phone Number: 82453780011, Start Time: 2003-12-17 10:03:10, End Time: 2018-05-17
21:11:09, Callee Number: 86251707383, Zone ID: <NULL>
ЗАМЕЧАНИЕ: Type: domestic, ID: 19, Phone Number: 82453780011, Start Time: 2004-01-19 03:33:11, End Time: 2011-09-08
12:53:37, Callee Number: 88136644056, Zone ID: <NULL>
DO

```

- вывести задолженность по оплате для заданного абонента.

```

CREATE OR REPLACE PROCEDURE get_total_debt(client_id_param INT, OUT total_debt NUMERIC)
LANGUAGE plpgsql AS $$
DECLARE
    service_record RECORD;
    tariff_record RECORD;
    additional_service_record RECORD;
    domestic_call_record RECORD;
    international_call_record RECORD;
    payment_record RECORD;
    service_cost NUMERIC := 0;
    tariff_cost NUMERIC := 0;
    additional_service_cost NUMERIC := 0;
    domestic_call_cost NUMERIC := 0;
    international_call_cost NUMERIC := 0;
    payment_total NUMERIC := 0;
    period_diff NUMERIC;
BEGIN
    -- Расчет задолженности по внутренним услугам
    FOR service_record IN

```

```

        SELECT
            isi.date_of_connection,
            COALESCE(isi.date_of_disconnection, CURRENT_TIMESTAMP) AS
date_of_disconnection,
            internal_service.internal_service_price,
            internal_service.internal_service_periodicity
        FROM
            internal_service_inclusion isi
            JOIN internal_service ON isi.internal_service_id =
internal_service.internal_service_id
            JOIN phone_on_tariff pot ON isi.phone_number = pot.phone_on_tariff_number
            JOIN contract ON pot.contract_id = contract.contract_id
        WHERE
            contract.client_id = client_id_param
    LOOP
        IF service_record.internal_service_periodicity = 'yearly' THEN
            service_cost := service_cost + DATE_PART('year',
AGE(service_record.date_of_disconnection, service_record.date_of_connection)) *
service_record.internal_service_price;
        ELSIF service_record.internal_service_periodicity = 'monthly' THEN
            service_cost := service_cost + DATE_PART('month',
AGE(service_record.date_of_disconnection, service_record.date_of_connection)) *
service_record.internal_service_price;
        ELSIF service_record.internal_service_periodicity = 'weekly' THEN
            service_cost := service_cost + (DATE_PART('day',
AGE(service_record.date_of_disconnection, service_record.date_of_connection)) / 7) *
service_record.internal_service_price;
        ELSIF service_record.internal_service_periodicity = 'daily' THEN
            service_cost := service_cost + DATE_PART('day',
AGE(service_record.date_of_disconnection, service_record.date_of_connection)) *
service_record.internal_service_price;
        ELSIF service_record.internal_service_periodicity = 'hourly' THEN
            service_cost := service_cost + (DATE_PART('day',
AGE(service_record.date_of_disconnection, service_record.date_of_connection)) * 24 +
DATE_PART('hour', AGE(service_record.date_of_disconnection,
service_record.date_of_connection))) * service_record.internal_service_price;
        END IF;
    END LOOP;

-- Расчет задолженности по основному тарифу
FOR tariff_record IN
    SELECT
        pot.activation_date,
        COALESCE(pot.deletion_date, CURRENT_TIMESTAMP) AS deletion_date,
        basic_tariff.basic_tariff_price
    FROM
        phone_on_tariff pot
        JOIN basic_tariff ON pot.tariff_id = basic_tariff.tariff_id
        JOIN contract ON pot.contract_id = contract.contract_id
    WHERE
        contract.client_id = client_id
LOOP

```

```

        tariff_cost := tariff_cost + EXTRACT(epoch FROM (tariff_record.deletion_date -
tariff_record.activation_date)) / 2628000 * tariff_record.basic_tariff_price;
    END LOOP;

    -- Расчет задолженности по дополнительным услугам тарифа
    FOR additional_service_record IN
        SELECT
            tsi.tariff_service_date_of_connection,
            COALESCE(tsi.tariff_service_date_of_disconnection, CURRENT_TIMESTAMP) AS
date_of_disconnection,
            internal_service.internal_service_price,
            internal_service.internal_service_periodicity
        FROM
            tariff_service_inclusion tsi
            JOIN internal_service ON tsi.internal_service_id =
internal_service.internal_service_id
            JOIN basic_tariff bt ON tsi.tariff_id = bt.tariff_id
            JOIN phone_on_tariff pot ON bt.tariff_id = pot.tariff_id
            JOIN contract ON pot.contract_id = contract.contract_id
        WHERE
            contract.client_id = client_id_param
    LOOP
        -- Расчет периода и стоимости дополнительной услуги
        IF service_record.internal_service_periodicity = 'yearly' THEN
            period_diff := DATE_PART('year', AGE(service_record.date_of_disconnection,
service_record.date_of_connection));
        ELSIF service_record.internal_service_periodicity = 'monthly' THEN
            period_diff := DATE_PART('month', AGE(service_record.date_of_disconnection,
service_record.date_of_connection));
        ELSIF service_record.internal_service_periodicity = 'weekly' THEN
            period_diff := (DATE_PART('day', AGE(service_record.date_of_disconnection,
service_record.date_of_connection)) / 7);
        ELSIF service_record.internal_service_periodicity = 'daily' THEN
            period_diff := DATE_PART('day', AGE(service_record.date_of_disconnection,
service_record.date_of_connection));
        ELSIF service_record.internal_service_periodicity = 'hourly' THEN
            period_diff := (DATE_PART('day', AGE(service_record.date_of_disconnection,
service_record.date_of_connection)) * 24 + DATE_PART('hour',
AGE(service_record.date_of_disconnection, service_record.date_of_connection)));
        ELSE
            period_diff := 0;
        END IF;

        additional_service_cost := additional_service_cost + (period_diff *
service_record.internal_service_price);
    END LOOP;

    -- Расчет стоимости звонков
    FOR domestic_call_record IN
        SELECT call_end_time, call_start_time, phone_number
        FROM domestic_call

```

```

        WHERE phone_number IN (SELECT phone_on_tariff_number FROM phone_on_tariff WHERE
contract_id IN (SELECT contract_id FROM contract WHERE client_id = client_id_param))
    LOOP
        SELECT domestic_call_cost_per_minute INTO domestic_call_cost
        FROM basic_tariff
        WHERE tariff_id = (SELECT tariff_id FROM phone_on_tariff WHERE
phone_on_tariff_number = domestic_call_record.phone_number);

        domestic_call_cost := domestic_call_cost + EXTRACT(EPOCH FROM
(domestic_call_record.call_end_time - domestic_call_record.call_start_time))/60 *
domestic_call_cost;
    END LOOP;

FOR international_call_record IN
    SELECT international_call_end_time, international_call_start_time, phone_number
    FROM international_call
    WHERE phone_number IN (SELECT phone_on_tariff_number FROM phone_on_tariff WHERE
contract_id IN (SELECT contract_id FROM contract WHERE client_id = client_id_param))
    LOOP
        SELECT international_call_cost_per_minute INTO international_call_cost
        FROM basic_tariff
        WHERE tariff_id = (SELECT tariff_id FROM phone_on_tariff WHERE
phone_on_tariff_number = international_call_record.phone_number);

        international_call_cost := international_call_cost + EXTRACT(EPOCH FROM
(international_call_record.international_call_end_time -
international_call_record.international_call_start_time))/60 * international_call_cost;
    END LOOP;

-- Суммирование платежей клиента
FOR payment_record IN
    SELECT payment_amount
    FROM payment
    JOIN phone_on_tariff pot ON payment.phone_number = pot.phone_on_tariff_number
    JOIN contract ON pot.contract_id = contract.contract_id
    WHERE contract.client_id = client_id_param AND payment.payment_status
    LOOP
        payment_total := payment_total + payment_record.payment_amount;
    END LOOP;

-- Итоговая задолженность
total_debt := service_cost + tariff_cost + additional_service_cost +
domestic_call_cost + international_call_cost - payment_total;
END;
$$;

```



```
DO $$
DECLARE
    client_debt NUMERIC;
BEGIN
    CALL get_total_debt(5, client_debt);
    RAISE NOTICE 'Total debt for client ID 5: %', client_debt;
END;
$$;
```

```
phone_provider=# CREATE OR REPLACE PROCEDURE get_total_debt(client_id_param INT, OUT total_debt NUMERIC)
```

```
phone_provider=# LANGUAGE plpgsql AS $$
```

```
phone_provider## DECLARE
```

```
phone_provider##     service_record RECORD;
```

```
phone_provider##     tariff_record RECORD;
```

```
phone_provider##     additional_service_record RECORD;
```

```
phone_provider##     domestic_call_record RECORD;
```

```
phone_provider##     international_call_record RECORD;
```

```
phone_provider##     payment_record RECORD;
```

```
phone_provider##     service_cost NUMERIC := 0;
```

```
phone_provider##     tariff_cost NUMERIC := 0;
```

```
phone_provider##     additional_service_cost NUMERIC := 0;
```

```
phone_provider##     domestic_call_cost NUMERIC := 0;
```

```
phone_provider##     international_call_cost NUMERIC := 0;
```

```
phone_provider##     payment_total NUMERIC := 0;
```

```
phone_provider##     period_diff NUMERIC;
```

```
phone_provider## BEGIN
```

```
phone_provider##     -- Расчет задолженности по внутренним услугам
```

```
phone_provider##     FOR service_record IN
```

```
phone_provider##         SELECT
```

```
phone_provider##             isi.date_of_connection,
```

```
phone_provider##             COALESCE(isi.date_of_disconnection, CURRENT_TIMESTAMP) AS date_of_disconnec
```

```
tion,
```

```
phone_provider##             internal_service.internal_service_price,
```

```
phone_provider##             internal_service.internal_service_periodicity
```

```
phone_provider##         FROM
```

```
phone_provider##             internal_service_inclusion isi
```

```
phone_provider##             JOIN internal_service ON isi.internal_service_id = internal_service.interna
```

```
l_service_id
```

```
phone_provider## JOIN phone_on_tariff pot ON isi.phone_number = pot.phone_on_tariff_number
```

```
phone_provider##             JOIN contract ON pot.contract_id = contract.contract_id
```

```
phone_provider##         WHERE
```

```
phone_provider##             contract.client_id = client_id_param
```

```
phone_provider##     LOOP
```

```
phone_provider## IF service_record.internal_service_periodicity = 'yearly' THEN
```

```
phone_provider##     service_cost := service_cost + DATE_PART('year', AGE(service_record.date_of
```

```
_disconnection, service_record.date_of_connection)) * service_record.internal_service_price;
```

```
phone_provider##     ELSIF service_record.internal_service_periodicity = 'monthly' THEN
```

```
phone_provider##     service_cost := service_cost + DATE_PART('month', AGE(service_record.date_o
```

```
f_disconnection, service_record.date_of_connection)) * service_record.internal_service_price;
```

```
phone_provider##     ELSIF service_record.internal_service_periodicity = 'weekly' THEN
```

```
phone_provider##     service_cost := service_cost + (DATE_PART('day', AGE(service_record.date_of
```

```
_disconnection, service_record.date_of_connection)) / 7) * service_record.internal_service_price;
```

```
phone_provider##     ELSIF service_record.internal_service_periodicity = 'daily' THEN
```

```
phone_provider##     service_cost := service_cost + DATE_PART('day', AGE(service_record.date_of
```

```
disconnection, service_record.date_of_connection)) * service_record.internal_service_price;
```

```
phone_provider##     ELSIF service_record.internal_service_periodicity = 'hourly' THEN
```

```
phone_provider##     service_cost := service_cost + (DATE_PART('day', AGE(service_record.date_of
```

```
_disconnection, service_record.date_of_connection)) * 24 + DATE_PART('hour', AGE(service_record.date_of_
```

```

phone_provider## IF service_record.internal_service_periodicity = 'yearly' THEN
phone_provider##     service_cost := service_cost + DATE_PART('year', AGE(service_record.date_of
phone_provider## _disconnection, service_record.date_of_connection)) * service_record.internal_service_price;
phone_provider##     ELIF service_record.internal_service_periodicity = 'monthly' THEN
phone_provider##     service_cost := service_cost + DATE_PART('month', AGE(service_record.date_o
phone_provider## f_disconnection, service_record.date_of_connection)) * service_record.internal_service_price;
phone_provider##     ELIF service_record.internal_service_periodicity = 'weekly' THEN
phone_provider##     service_cost := service_cost + (DATE_PART('day', AGE(service_record.date_of
phone_provider## _disconnection, service_record.date_of_connection)) / 7) * service_record.internal_service_price;
phone_provider##     ELIF service_record.internal_service_periodicity = 'daily' THEN
phone_provider##     service_cost := service_cost + DATE_PART('day', AGE(service_record.date_of_
phone_provider## _disconnection, service_record.date_of_connection)) * service_record.internal_service_price;
phone_provider##     ELIF service_record.internal_service_periodicity = 'hourly' THEN
phone_provider##     service_cost := service_cost + (DATE_PART('day', AGE(service_record.date_of
phone_provider## _disconnection, service_record.date_of_connection)) * 24 + DATE_PART('hour', AGE(service_record.date_of_
phone_provider## _disconnection, service_record.date_of_connection))) * service_record.internal_service_price;
phone_provider##     END IF;
phone_provider## END LOOP;
phone_provider##
phone_provider## -- Расчет задолженности по основному тарифу
phone_provider## FOR tariff_record IN
phone_provider##     SELECT
phone_provider##         pot.activation_date,
phone_provider##         COALESCE(pot.deletion_date, CURRENT_TIMESTAMP) AS deletion_date,
phone_provider##         basic_tariff.basic_tariff_price
phone_provider##     FROM
phone_provider##         phone_on_tariff pot
phone_provider##         JOIN basic_tariff ON pot.tariff_id = basic_tariff.tariff_id
phone_provider##         JOIN contract ON pot.contract_id = contract.contract_id
phone_provider##     WHERE
phone_provider##         contract.client_id = client_id
phone_provider## LOOP
phone_provider##     tariff_cost := tariff_cost + EXTRACT(epoch FROM (tariff_record.deletion_date -
phone_provider## tariff_record.activation_date)) / 2628000 * tariff_record.basic_tariff_price;
phone_provider## END LOOP;
phone_provider##
phone_provider## -- Расчет задолженности по дополнительным услугам тарифа
phone_provider## FOR additional_service_record IN
phone_provider##     SELECT
phone_provider##         tsi.tariff_service_date_of_connection,
phone_provider##         COALESCE(tsi.tariff_service_date_of_disconnection, CURRENT_TIMESTAMP) AS da
phone_provider## te_of_disconnection,
phone_provider##         internal_service.internal_service_price,
phone_provider##         internal_service.internal_service_periodicity
phone_provider##     FROM
phone_provider##         tariff_service_inclusion tsi
phone_provider##         JOIN internal_service ON tsi.internal_service_id = internal_service.interna
phone_provider## _service_id
phone_provider## JOIN basic_tariff bt ON tsi.tariff_id = bt.tariff_id
phone_provider## JOIN phone_on_tariff pot ON bt.tariff_id = pot.tariff_id
phone_provider## JOIN contract ON pot.contract_id = contract.contract_id

```

```

date_of_disconnection,
phone_provider##          internal_service.internal_service_price,
phone_provider##          internal_service.internal_service_periodicity
phone_provider##          FROM
phone_provider##          tariff_service_inclusion tsi
phone_provider##          JOIN internal_service ON tsi.internal_service_id = internal_service.interna
l_service_id
phone_provider## JOIN basic_tariff bt ON tsi.tariff_id = bt.tariff_id
phone_provider## JOIN phone_on_tariff pot ON bt.tariff_id = pot.tariff_id
phone_provider##          JOIN contract ON pot.contract_id = contract.contract_id
phone_provider##          WHERE
phone_provider##          contract.client_id = client_id_param
phone_provider##          LOOP
phone_provider##          -- Расчет периода и стоимости дополнительной услуги
phone_provider## IF service_record.internal_service_periodicity = 'yearly' THEN
phone_provider##          period_diff := DATE_PART('year', AGE(service_record.date_of_disconnection,
service_record.date_of_connection));
phone_provider##          ELSIF service_record.internal_service_periodicity = 'monthly' THEN
phone_provider##          period_diff := DATE_PART('month', AGE(service_record.date_of_disconnection,
service_record.date_of_connection));
phone_provider##          ELSIF service_record.internal_service_periodicity = 'weekly' THEN
phone_provider##          period_diff := (DATE_PART('day', AGE(service_record.date_of_disconnection,
service_record.date_of_connection)) / 7);
phone_provider##          ELSIF service_record.internal_service_periodicity = 'daily' THEN
phone_provider##          period_diff := DATE_PART('day', AGE(service_record.date_of_disconnection, s
ervice_record.date_of_connection));
phone_provider##          ELSIF service_record.internal_service_periodicity = 'hourly' THEN
phone_provider##          period_diff := (DATE_PART('day', AGE(service_record.date_of_disconnection,
service_record.date_of_connection)) * 24 + DATE_PART('hour', AGE(service_record.date_of_disconnection, s
ervice_record.date_of_connection)));
phone_provider##          ELSE
phone_provider##          period_diff := 0;
phone_provider##          END IF;
phone_provider##          additional_service_cost := additional_service_cost + (period_diff * service_rec
ord.internal_service_price);
phone_provider##          END LOOP;
phone_provider## -- Расчет стоимости звонков
phone_provider##          FOR domestic_call_record IN
phone_provider##          SELECT call_end_time, call_start_time, phone_number
phone_provider##          FROM domestic_call
phone_provider##          WHERE phone_number IN (SELECT phone_on_tariff_number FROM phone_on_tariff WHERE
contract_id IN (SELECT contract_id FROM contract WHERE client_id = client_id_param))
phone_provider##          LOOP
phone_provider##          SELECT domestic_call_cost_per_minute INTO domestic_call_cost
phone_provider##          FROM basic_tariff
phone_provider##          WHERE tariff_id = (SELECT tariff_id FROM phone_on_tariff WHERE phone_on_tariff_
number = domestic_call_record.phone_number);
phone_provider##          domestic_call_cost := domestic_call_cost + EXTRACT(EPOCH FROM (domestic_call_re

```

```

hone_provider$#      FOR domestic_call_record IN
hone_provider$#      SELECT call_end_time, call_start_time, phone_number
hone_provider$#      FROM domestic_call
hone_provider$#      WHERE phone_number IN (SELECT phone_on_tariff_number FROM phone_on_tariff WHERE
contract_id IN (SELECT contract_id FROM contract WHERE client_id = client_id_param))
hone_provider$#      LOOP
hone_provider$#      SELECT domestic_call_cost_per_minute INTO domestic_call_cost
hone_provider$#      FROM basic_tariff
hone_provider$#      WHERE tariff_id = (SELECT tariff_id FROM phone_on_tariff WHERE phone_on_tariff_
umber = domestic_call_record.phone_number);
hone_provider$#      domestic_call_cost := domestic_call_cost + EXTRACT(EPOCH FROM (domestic_call_re
ord.call_end_time - domestic_call_record.call_start_time))/60 * domestic_call_cost;
hone_provider$#      END LOOP;
hone_provider$#
hone_provider$#      FOR international_call_record IN
hone_provider$#      SELECT international_call_end_time, international_call_start_time, phone_number
hone_provider$#      FROM international_call
hone_provider$#      WHERE phone_number IN (SELECT phone_on_tariff_number FROM phone_on_tariff WHERE
contract_id IN (SELECT contract_id FROM contract WHERE client_id = client_id_param))
hone_provider$#      LOOP
hone_provider$#      SELECT international_call_cost_per_minute INTO international_call_cost
hone_provider$#      FROM basic_tariff
hone_provider$#      WHERE tariff_id = (SELECT tariff_id FROM phone_on_tariff WHERE phone_on_tariff_
umber = international_call_record.phone_number);
hone_provider$#      international_call_cost := international_call_cost + EXTRACT(EPOCH FROM (intern
tional_call_record.international_call_end_time - international_call_record.international_call_start_tim
))/60 * international_call_cost;
hone_provider$#      END LOOP;
hone_provider$#
hone_provider$#      -- Суммирование платежей клиента
hone_provider$#      FOR payment_record IN
hone_provider$#      SELECT payment_amount
hone_provider$#      FROM payment
hone_provider$#      JOIN phone_on_tariff pot ON payment.phone_number = pot.phone_on_tariff_number
hone_provider$#      JOIN contract ON pot.contract_id = contract.contract_id
hone_provider$#      WHERE contract.client_id = client_id_param AND payment.payment_status
hone_provider$#      LOOP
hone_provider$#      payment_total := payment_total + payment_record.payment_amount;
hone_provider$#      END LOOP;
hone_provider$#
hone_provider$#      -- Итоговая задолженность
hone_provider$#      total_debt := service_cost + tariff_cost + additional_service_cost + domestic_call_
ost + international_call_cost - payment_total;
hone_provider$# END;
hone_provider$# $$;
REATE PROCEDURE

```

```

phone_provider=# DO $$
phone_provider$# DECLARE
phone_provider$#     client_debt NUMERIC;
phone_provider$# BEGIN
phone_provider$#     CALL get_total_debt(5, client_debt);
phone_provider$#     RAISE NOTICE 'Total debt for client ID 5: %', client_debt;
phone_provider$# END;
phone_provider$# $$;
ЗАМЕЧАНИЕ: Total debt for client ID 5: 93812502.71461187214355223772
DO
phone_provider=#

```

Чтобы вывести результат в виде таблицы, напишу функцию, которая оборачивает вызов этой процедуры:

```
CREATE OR REPLACE FUNCTION get_client_debt(client_id INT) RETURNS NUMERIC AS $$
DECLARE
    debt NUMERIC;
BEGIN
    CALL get_total_debt(client_id, debt);
    RETURN debt;
END;
$$ LANGUAGE plpgsql;
```

The screenshot shows a SQL IDE with a dark theme. The top pane, titled 'Query', contains the SQL code for creating the function. The bottom pane, titled 'Data Output', shows the message 'Query returned successfully in 62 msec.'.

```
Query    Query History
1 CREATE OR REPLACE FUNCTION get_client_debt(client_id INT) RETURNS NUMERIC AS $$
2 DECLARE
3     debt NUMERIC;
4 BEGIN
5     CALL get_total_debt(client_id, debt);
6     RETURN debt;
7 END;
8 $$ LANGUAGE plpgsql;
9

Data Output    Messages    Notifications
CREATE FUNCTION
Query returned successfully in 62 msec.
```

А затем напишу простой селект:

```
SELECT client_id, get_client_debt(client_id) AS client_debt
FROM client
WHERE client_id = 55;
```

```
phone_provider=# SELECT client_id, get_client_debt(client_id) AS client_debt
phone_provider=# FROM client
phone_provider=# WHERE client_id = 55;
 client_id |          client_debt
-----+-----
       55 | 61675388.24794520547955223772
(1 строка)
```



- рассчитать общую стоимость звонков по каждой зоне за истекшую неделю.

Напишу INSERT-запросы для international\_call, чтобы появились записи на истекшую неделю:

```
phone_provider=# INSERT INTO "international_call" ("international_call_id", "phone_number", "international_call_start_time", "international_call_end_time", "international_callee_number", "call_zone_id"
VALUES (60, '81539108668', '2023-12-02 10:00:00', '2023-12-02 10:30:00', '86300643270', 4);
INSERT 0 1
phone_provider=# INSERT INTO "international_call" ("international_call_id", "phone_number", "international_call_start_time", "international_call_end_time", "international_callee_number", "call_zone_id"
VALUES (61, '88686637097', '2023-12-03 11:00:00', '2023-12-03 12:00:00', '81517335515', 15);
INSERT 0 1
phone_provider=# INSERT INTO "international_call" ("international_call_id", "phone_number", "international_call_start_time", "international_call_end_time", "international_callee_number", "call_zone_id"
VALUES (62, '83164665019', '2023-12-04 09:00:00', '2023-12-04 11:00:00', '88963845491', 2);
INSERT 0 1
phone_provider=# INSERT INTO "international_call" ("international_call_id", "phone_number", "international_call_start_time", "international_call_end_time", "international_callee_number", "call_zone_id"
VALUES (63, '84157873461', '2023-12-05 14:00:00', '2023-12-05 16:00:00', '89610856021', 10);
INSERT 0 1
phone_provider=# INSERT INTO "international_call" ("international_call_id", "phone_number", "international_call_start_time", "international_call_end_time", "international_callee_number", "call_zone_id"
VALUES (64, '88686637097', '2023-12-06 15:00:00', '2023-12-06 20:00:00', '88427575477', 1);
INSERT 0 1
phone_provider=# INSERT INTO "international_call" ("international_call_id", "phone_number", "international_call_start_time", "international_call_end_time", "international_callee_number", "call_zone_id"
VALUES (65, '83939283170', '2023-12-07 08:00:00', '2023-12-07 08:45:00', '88007835173', 9);
INSERT 0 1
```

```
phone_provider=# INSERT INTO "international_call" ("international_call_id", "phone_number", "international_call_start_time", "international_call_end_time", "international_callee_number", "call_zone_id"
VALUES (66, '84548540986', '2023-12-04 17:00:00', '2023-12-04 19:30:00', '83950048013', 1);
INSERT 0 1
phone_provider=# INSERT INTO "international_call" ("international_call_id", "phone_number", "international_call_start_time", "international_call_end_time", "international_callee_number", "call_zone_id"
VALUES (67, '81539108668', '2023-12-05 10:15:00', '2023-12-05 13:15:00', '86300643270', 4);
INSERT 0 1
phone_provider=# INSERT INTO "international_call" ("international_call_id", "phone_number", "international_call_start_time", "international_call_end_time", "international_callee_number", "call_zone_id"
VALUES (68, '88686637097', '2023-12-05 12:30:00', '2023-12-05 15:30:00', '81517335515', 15);
INSERT 0 1
phone_provider=# INSERT INTO "international_call" ("international_call_id", "phone_number", "international_call_start_time", "international_call_end_time", "international_callee_number", "call_zone_id"
VALUES (69, '83164665019', '2023-12-06 14:45:00', '2023-12-06 17:45:00', '88963845491', 2);
INSERT 0 1
phone_provider=# INSERT INTO "international_call" ("international_call_id", "phone_number", "international_call_start_time", "international_call_end_time", "international_callee_number", "call_zone_id"
VALUES (70, '84157873461', '2023-12-06 16:00:00', '2023-12-06 18:00:00', '89610856021', 10);
INSERT 0 1
phone_provider=# INSERT INTO "international_call" ("international_call_id", "phone_number", "international_call_start_time", "international_call_end_time", "international_callee_number", "call_zone_id"
VALUES (71, '88686637097', '2023-12-07 19:00:00', '2023-12-07 23:00:00', '88427575477', 1);
INSERT 0 1
phone_provider=# INSERT INTO "international_call" ("international_call_id", "phone_number", "international_call_start_time", "international_call_end_time", "international_callee_number", "call_zone_id"
VALUES (72, '83939283170', '2023-12-07 07:30:00', '2023-12-07 09:30:00', '88007835173', 9);
INSERT 0 1
```

Процедура

```

phone_provider=# CREATE OR REPLACE PROCEDURE calculate_weekly_call_costs()
phone_provider=# LANGUAGE plpgsql AS $$
phone_provider$# DECLARE
phone_provider$#     zone_record RECORD;
phone_provider$#     total_cost NUMERIC;
phone_provider$# BEGIN
phone_provider$#     CREATE TEMP TABLE IF NOT EXISTS weekly_call_costs (call_zone_id INT, total_cost NUMERIC);
phone_provider$#
phone_provider$#     TRUNCATE weekly_call_costs;
phone_provider$#
phone_provider$#     FOR zone_record IN SELECT call_zone_id, cost_per_minute_call_zone FROM call_zone LOOP
phone_provider$#
phone_provider$#         SELECT INTO total_cost SUM(EXTRACT(EPOCH FROM (ic.international_call_end_time - ic.international_call_start_time)) / 60 * zone_record.cost_per_minute_call_zone)
phone_provider$#         FROM international_call ic
phone_provider$#         WHERE ic.call_zone_id = zone_record.call_zone_id AND ic.international_call_start_time >= NOW() - INTERVAL '1 week';
phone_provider$#
phone_provider$#         INSERT INTO weekly_call_costs VALUES (zone_record.call_zone_id, COALESCE(total_cost, 0));
phone_provider$#     END LOOP;
phone_provider$#
phone_provider$#     FOR zone_record IN SELECT * FROM weekly_call_costs LOOP
phone_provider$#         RAISE NOTICE 'Zone ID: %, Total Cost: %', zone_record.call_zone_id, zone_record.total_cost;
phone_provider$#
phone_provider$#     END LOOP;
phone_provider$#
phone_provider$# END;
phone_provider$# $$;
CREATE PROCEDURE
phone_provider=# CALL calculate_weekly_call_costs();
ЗАМЕЧАНИЕ: Zone ID: 1, Total Cost: 33120.000000000000000000
ЗАМЕЧАНИЕ: Zone ID: 2, Total Cost: 24900.000000000000000000
ЗАМЕЧАНИЕ: Zone ID: 3, Total Cost: 0
ЗАМЕЧАНИЕ: Zone ID: 4, Total Cost: 9870.000000000000000000
ЗАМЕЧАНИЕ: Zone ID: 5, Total Cost: 0
ЗАМЕЧАНИЕ: Zone ID: 6, Total Cost: 0
ЗАМЕЧАНИЕ: Zone ID: 7, Total Cost: 0
ЗАМЕЧАНИЕ: Zone ID: 8, Total Cost: 0
ЗАМЕЧАНИЕ: Zone ID: 9, Total Cost: 8085.000000000000000000

```

```

phone_provider=# SELECT * FROM weekly_call_costs;
call_zone_id |      total_cost

```

call_zone_id	total_cost
1	33120.000000000000000000
2	24900.000000000000000000
3	0
4	9870.000000000000000000
5	0
6	0
7	0
8	0
9	8085.000000000000000000
10	2160.000000000000000000
11	0
12	0
13	0
14	0
15	13440.000000000000000000
16	0
17	0
18	0

(18 строк)

```

CREATE OR REPLACE PROCEDURE calculate_weekly_call_costs()
LANGUAGE plpgsql AS $$
DECLARE
    zone_record RECORD;
    total_cost NUMERIC;
BEGIN
    CREATE TEMP TABLE IF NOT EXISTS weekly_call_costs (call_zone_id INT, total_cost
    NUMERIC);

    TRUNCATE weekly_call_costs;

    FOR zone_record IN SELECT call_zone_id, cost_per_minute_call_zone FROM call_zone LOOP

        SELECT INTO total_cost SUM(EXTRACT(EPOCH FROM (ic.international_call_end_time -
ic.international_call_start_time)) / 60 * zone_record.cost_per_minute_call_zone)
        FROM international_call ic
        WHERE ic.call_zone_id = zone_record.call_zone_id AND
ic.international_call_start_time >= NOW() - INTERVAL '1 week';

        INSERT INTO weekly_call_costs VALUES (zone_record.call_zone_id,
COALESCE(total_cost, 0));
    END LOOP;

    FOR zone_record IN SELECT * FROM weekly_call_costs LOOP
        RAISE NOTICE 'Zone ID: %, Total Cost: %', zone_record.call_zone_id,
zone_record.total_cost;
    END LOOP;

END;
$$;
CALL calculate_weekly_call_costs();
SELECT * FROM weekly_call_costs;

```

Функция:

```

phone_provider=# CREATE OR REPLACE FUNCTION calculate_weekly_call_costs_func()
phone_provider=# RETURNS TABLE(call_zone_id INT, total_cost NUMERIC) AS $$
phone_provider$# BEGIN
phone_provider$#     RETURN QUERY
phone_provider$#     SELECT
phone_provider$#         ic.call_zone_id,
phone_provider$#         SUM(EXTRACT(EPOCH FROM (ic.international_call_end_time - ic.international_call_
start_time)) / 60 * cz.cost_per_minute_call_zone) AS total_cost
phone_provider$#     FROM
phone_provider$#         international_call ic
phone_provider$#         JOIN call_zone cz ON ic.call_zone_id = cz.call_zone_id
phone_provider$#     WHERE
phone_provider$#         ic.international_call_start_time >= NOW() - INTERVAL '1 week'
phone_provider$#     GROUP BY
phone_provider$#         ic.call_zone_id;
phone_provider$# END;
phone_provider$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
phone_provider=# SELECT * FROM calculate_weekly_call_costs_func();

```



```

CREATE FUNCTION
phone_provider=# SELECT * FROM calculate_weekly_call_costs_func();
call_zone_id |      total_cost
-----+-----
          1 | 33120.000000000000000000
          2 | 24900.000000000000000000
          4 |  9870.000000000000000000
          9 |  8085.000000000000000000
         10 |  2160.000000000000000000
         15 | 13440.000000000000000000
(6 строк)

```

```

CREATE OR REPLACE FUNCTION calculate_weekly_call_costs_func()
RETURNS TABLE(call_zone_id INT, total_cost NUMERIC) AS $$
BEGIN
    RETURN QUERY
    SELECT
        ic.call_zone_id,
        SUM(EXTRACT(EPOCH FROM (ic.international_call_end_time -
ic.international_call_start_time)) / 60 * cz.cost_per_minute_call_zone) AS total_cost
    FROM
        international_call ic
    JOIN call_zone cz ON ic.call_zone_id = cz.call_zone_id
    WHERE
        ic.international_call_start_time >= NOW() - INTERVAL '1 week'
    GROUP BY
        ic.call_zone_id;
END;
$$ LANGUAGE plpgsql;
SELECT * FROM calculate_weekly_call_costs_func();

```

### Создайте авторские триггеры:

- 1) Триггер, который предотвратит вставку записи о звонке, если время начала звонка больше времени его окончания, выполним запрос

```

phone_provider=# CREATE OR REPLACE FUNCTION before_insert_call()
phone_provider=# RETURNS TRIGGER AS $$
phone_provider$# BEGIN
phone_provider$#
phone_provider$#     IF NEW.international_call_start_time >= NEW.international_call_end_time THEN
phone_provider$#         RAISE EXCEPTION 'Время начала звонка не может быть позже времени окончания звонка.';
phone_provider$#     END IF;
phone_provider$#     RETURN NEW;
phone_provider$# END;
phone_provider$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
phone_provider=#
phone_provider=# CREATE TRIGGER before_insert_call_trigger
phone_provider=# BEFORE INSERT ON international_call
phone_provider=# FOR EACH ROW EXECUTE FUNCTION before_insert_call();

```

```

CREATE OR REPLACE FUNCTION before_insert_call()
RETURNS TRIGGER AS $$
BEGIN

    IF NEW.international_call_start_time >= NEW.international_call_end_time THEN
        RAISE EXCEPTION 'Время начала звонка не может быть позже времени окончания звонка.';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER before_insert_call_trigger
BEFORE INSERT ON international_call
FOR EACH ROW EXECUTE FUNCTION before_insert_call();

```

Проверка работы триггера:

```

phone_provider=# INSERT INTO "international_call" ("international_call_id", "phone_number", "international_call_start_time", "international_call_end_time", "international_callee_number", "call_zone_id") VALUES (79, '83939283170', '2023-12-05 18:00:00', '2023-12-05 12:00:00', '88007835173', 9);
ОШИБКА:  Время начала звонка не может быть позже времени окончания звонка.
КОНТЕКСТ:  функция PL/pgSQL before_insert_call(), строка 5, оператор RAISE

```

2) Триггер, проверяющий, достаточно ли баланса на телефоне для совершения звонка:

```

CREATE OR REPLACE FUNCTION check_balance_before_call()
RETURNS TRIGGER AS $$
DECLARE
    expected_cost NUMERIC;
    client_balance NUMERIC;
BEGIN
    expected_cost := EXTRACT(EPOCH FROM (NEW.international_call_end_time -
NEW.international_call_start_time)) / 60
        * (SELECT cost_per_minute_call_zone FROM call_zone WHERE call_zone_id
= NEW.call_zone_id);

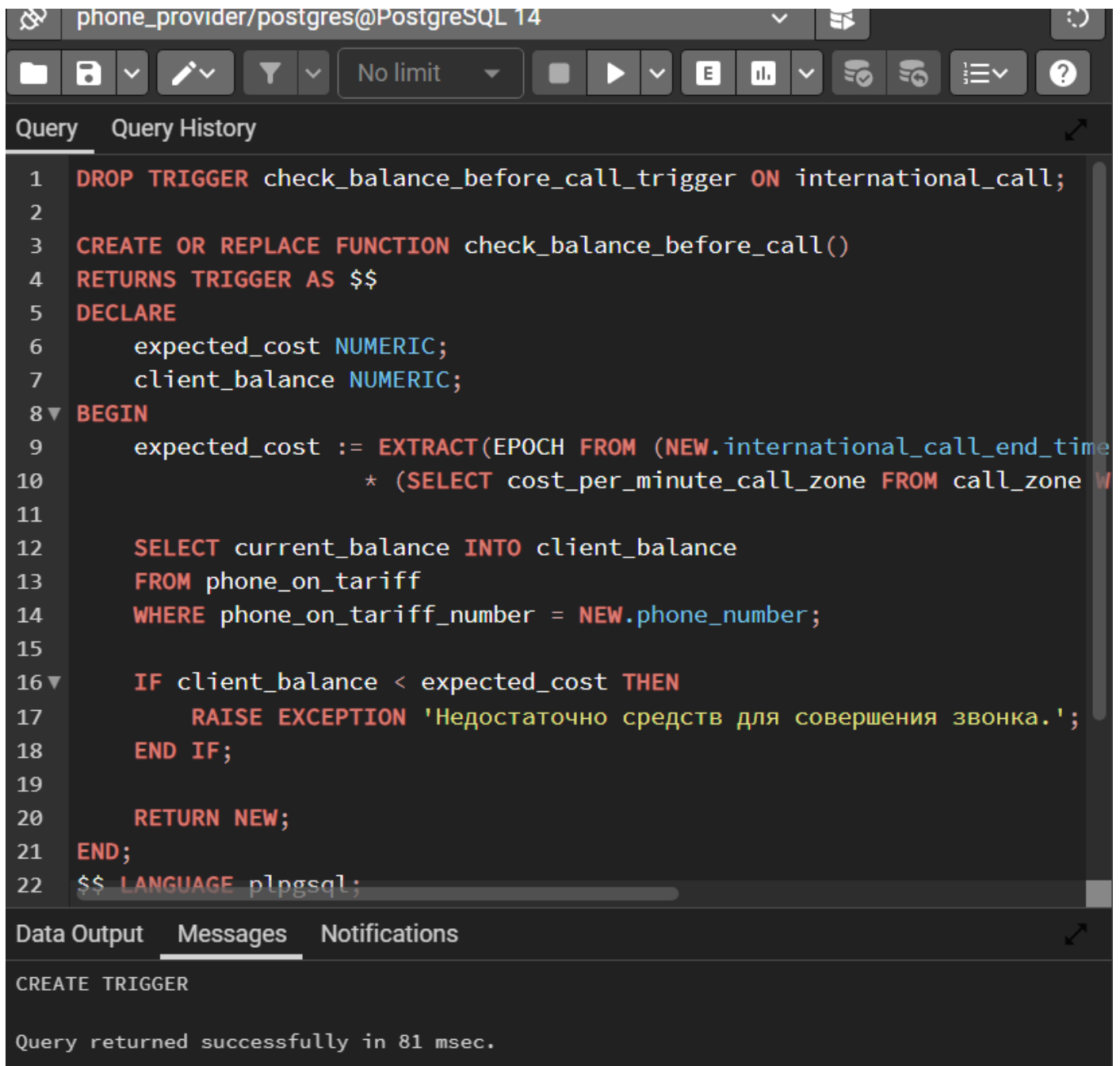
    SELECT current_balance INTO client_balance
    FROM phone_on_tariff
    WHERE phone_on_tariff_number = NEW.phone_number;

    IF client_balance < expected_cost THEN
        RAISE EXCEPTION 'Недостаточно средств для совершения звонка.';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_balance_before_call_trigger
BEFORE INSERT ON international_call
FOR EACH ROW EXECUTE FUNCTION check_balance_before_call();

```



The screenshot shows a PostgreSQL IDE interface. The top bar indicates the connection is to 'phone\_provider/postgres@PostgreSQL 14'. Below the toolbar, the 'Query' tab is active, displaying a PL/pgSQL function definition. The function, named 'check\_balance\_before\_call', is designed to be triggered before an international call. It declares two numeric variables, 'expected\_cost' and 'client\_balance'. The logic involves calculating the expected cost by extracting the epoch from the call's end time and multiplying it by the cost per minute from a 'call\_zone' table. It then checks the current balance against this expected cost. If the balance is insufficient, it raises an exception with the Russian message 'Недостаточно средств для совершения звонка.' (Insufficient funds for making a call). If the balance is sufficient, it returns the new record. The function is created using the 'LANGUAGE plpgsql' syntax. Below the query editor, the 'Messages' tab shows the execution output, confirming that the trigger was created successfully in 81 milliseconds.

```
1 DROP TRIGGER check_balance_before_call_trigger ON international_call;
2
3 CREATE OR REPLACE FUNCTION check_balance_before_call()
4 RETURNS TRIGGER AS $$
5 DECLARE
6     expected_cost NUMERIC;
7     client_balance NUMERIC;
8 BEGIN
9     expected_cost := EXTRACT(EPOCH FROM (NEW.international_call_end_time
10                                     * (SELECT cost_per_minute_call_zone FROM call_zone W
11
12     SELECT current_balance INTO client_balance
13     FROM phone_on_tariff
14     WHERE phone_on_tariff_number = NEW.phone_number;
15
16     IF client_balance < expected_cost THEN
17         RAISE EXCEPTION 'Недостаточно средств для совершения звонка.';
18     END IF;
19
20     RETURN NEW;
21 END;
22 $$ LANGUAGE plpgsql;
```

CREATE TRIGGER

Query returned successfully in 81 msec.

```
INSERT INTO "international_call" ("international_call_id", "phone_number",
"international_call_start_time", "international_call_end_time",
"international_callee_number", "call_zone_id") VALUES (18, '81341681439', '2001-02-10
04:05:38', '2022-04-01 07:29:08', '84662995076', 12);
```

The screenshot shows a PostgreSQL IDE interface. At the top, there are two tabs: "Query" and "Query History". The "Query" tab is active, displaying a single SQL statement: `1 INSERT INTO "international_call" ("international_call_id", "phone_number`. Below the query editor, there are three tabs: "Data Output", "Messages", and "Notifications". The "Messages" tab is active, showing an error message in Russian: `ERROR: Недостаточно средств для совершения звонка.` The context provided is: `CONTEXT: функция PL/pgSQL check_balance_before_call(), строка 14, оператор RAISE`. Below this, the error is repeated in Russian: `ОШИБКА: Недостаточно средств для совершения звонка.` followed by `SQL state: P0001`. A "Maximise" button is visible on the right side of the Messages tab.

```
Query  Query History
1  INSERT INTO "international_call" ("international_call_id", "phone_number

Data Output  Messages  Notifications
ERROR: Недостаточно средств для совершения звонка.
CONTEXT:  функция PL/pgSQL check_balance_before_call(), строка 14, оператор RAISE
ОШИБКА: Недостаточно средств для совершения звонка.
SQL state: P0001
```

### Вывод:

В ходе лабораторной работы были написаны процедуры и созданы авторские триггеры, на практике применены знания, полученные на лекциях и практическом занятии по триггерам.