МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»
РТУ МИРЭА

Практическое занятие № 1

Тема занятия: 1) Создание пользовательских функций
2) Создание пользовательских функций. Язык pl/pgSQL
3) Создание пользовательских функции. Работа с датой и временем

Выполнил студент 3 курса группы
ЭФБО-10-23 Ефремов А.И.

Проверил доцент Бочаров М.И.

Москва
2025 г.

Ефремов ЭФБО-10-23 Практическая 1

Практическая 1.1

1. Конвертирует стоимость объекта недвижимости в другую валюту

```
CREATE OR REPLACE FUNCTION convert_property_currency_yefremov(exchange_rate DOUBLE
PRECISION, property_id INTEGER)
RETURNS DOUBLE PRECISION AS $$
BEGIN
    RETURN (SELECT price / exchange_rate FROM property WHERE id = property_id);
END;
```

```
SELECT convert_property_currency_yefremov(100, 1);
```

```
mirea=# SELECT convert_property_currency_yefremov(100, 1);
 convert_property_currency_yefremov
------------------------------------
                              50000
(1 row)
```

2. Рассчитывает заработную плату риэлтора за месяц по формуле: `N*S+R`, где N – общая сумма проданных объектов недвижимости в месяц (подсчет осуществляется автоматически по данным таблицы «Продажи», S –коэффициент, R – премия.

```
CREATE OR REPLACE FUNCTION calculate_realtor_salary_yefremov(coefficient DOUBLE PRECISION,
premium DOUBLE PRECISION, target_month INTEGER, target_year INTEGER, realtor_last_name
VARCHAR)
RETURNS DOUBLE PRECISION AS $$
DECLARE
    total_sales DOUBLE PRECISION;
    realtor_id_val INTEGER;
BEGIN
    SELECT id INTO realtor_id_val FROM realtor WHERE last_name = realtor_last_name;
    IF realtor_id_val IS NULL THEN
        RAISE EXCEPTION 'Риэлтор с фамилией % не найден', realtor_last_name;
    END IF;

    SELECT COALESCE(SUM(price), 0) INTO total_sales
    FROM sale
    WHERE realtor_id = realtor_id_val
      AND EXTRACT(MONTH FROM sale_date) = target_month
      AND EXTRACT(YEAR FROM sale_date) = target_year;

    RETURN total_sales * coefficient + premium;
END;
```

```
SELECT calculate_realtor_salary_yefremov(0.01, 10000, 1, 2023, 'Иванов');
```

```
mirea=# SELECT calculate_realtor_salary_yefremov(0.01, 10000, 1, 2023, 'Иванов');
 calculate_realtor_salary_yefremov
-----------------------------------
                             10000
(1 row)
```

3. Рассчитывает сумму ежемесячного платежа по ипотеке. Входные параметры: код объекта недвижимости, процентная ставка, срок (в годах), первоначальный взнос.

```
CREATE OR REPLACE FUNCTION calculate_mortgage_payment_yefremov(property_id INTEGER,
annual_rate DOUBLE PRECISION, term_years INTEGER, initial_payment DOUBLE PRECISION)
RETURNS DOUBLE PRECISION AS $$
DECLARE
    property_price DOUBLE PRECISION;
    monthly_rate DOUBLE PRECISION;
    months INTEGER;
    pv DOUBLE PRECISION;
BEGIN
    SELECT price INTO property_price FROM property WHERE id = property_id;
    IF property_price IS NULL THEN
        RAISE EXCEPTION 'Объект недвижимости с ID % не найден', property_id;
    END IF;

    IF initial_payment >= property_price THEN
        RAISE EXCEPTION 'Первоначальный взнос не может быть больше или равен стоимости';
    END IF;

    pv := property_price - initial_payment;
    monthly_rate := annual_rate / 100 / 12;
    months := term_years * 12;

    IF monthly_rate = 0 THEN
        RETURN pv / months;
    ELSE
        RETURN (monthly_rate * pv) / (1 - POWER(1 + monthly_rate, -months));
    END IF;
END;
```

```
SELECT calculate_mortgage_payment_yefremov(1, 10, 20, 1000000);
```

```
CREATE FUNCTION
mirea=# SELECT calculate_mortgage_payment_yefremov(1, 10, 20, 1000000);
 calculate_mortgage_payment_yefremov
-------------------------------------
                 38600.86580296036
(1 row)
```

4. Формирует список объектов недвижимости, стоимость 1м^2 которых меньше средней по району

```
CREATE OR REPLACE FUNCTION list_cheap_properties_yefremov(district_name VARCHAR)
RETURNS SETOF property AS $$
DECLARE
    avg_price_per_m2 DOUBLE PRECISION;
    district_id_val INTEGER;
BEGIN
```

```
    SELECT id INTO district_id_val FROM district WHERE name = district_name;
    IF district_id_val IS NULL THEN
        RAISE EXCEPTION 'Район % не найден', district_name;
    END IF;

    SELECT AVG(price::DOUBLE PRECISION / area) INTO avg_price_per_m2
    FROM property
    WHERE district_id = district_id_val;

    RETURN QUERY
    SELECT * FROM property p
    WHERE p.district_id = district_id_val
      AND (p.price::DOUBLE PRECISION / p.area) < avg_price_per_m2;
END;
```

```
SELECT * FROM list_cheap_properties_yefremov('Центральный'::VARCHAR);
```

```
mirea=# SELECT * FROM list_cheap_properties_yefremov('Центральный'::VARCHAR);
 id | district_id |      address      | floor | room_count | type_id | status |  price  |           description           | building_material_id | area | listing_date
----+-------------+-------------------+-------+------------+---------+--------+---------+--------------------------------+----------------------+------+--------------
 17 |           1 | пр. Победы, 17    |     7 |          4 |       1 | t      | 9000000 | Квартира с дизайнерским ремонтом |                    2 |   85 | 2023-02-28
 21 |           1 | ул. Новая, 21     |     5 |          3 |       1 | t      | 7200000 | Квартира в новостройке         |                    2 |   68 | 2023-03-05
 25 |           1 | ул. Торговая, 26  |     6 |          3 |       1 | t      | 7800000 | Квартира рядом с торговым центром |                 1 |   72 | 2023-01-30
 29 |           1 | ул. Береговая, 28 |     5 |          4 |       1 | t      | 8500000 | Квартира с видом на реку       |                    2 |   80 | 2023-02-05
(4 rows)
```

5. Формирует статистику по продажам за указанный год

```
CREATE OR REPLACE FUNCTION sales_statistics_yefremov(target_year INTEGER)
RETURNS TABLE(type_name VARCHAR, sales_count BIGINT, percentage DOUBLE PRECISION,
total_sum DOUBLE PRECISION) AS $$
DECLARE
    total_sales_count BIGINT;
BEGIN
    SELECT COUNT(*) INTO total_sales_count
    FROM sale s
    JOIN property p ON s.property_id = p.id
    WHERE EXTRACT(YEAR FROM s.sale_date) = target_year;

    RETURN QUERY
    SELECT t.name AS type_name,
           COUNT(s.id) AS sales_count,
           (COUNT(s.id)::DOUBLE PRECISION / total_sales_count * 100) AS percentage,
           SUM(s.price) AS total_sum
    FROM sale s
    JOIN property p ON s.property_id = p.id
    JOIN type t ON p.type_id = t.id
    WHERE EXTRACT(YEAR FROM s.sale_date) = target_year
    GROUP BY t.name;
END;
```

```
SELECT * FROM sales_statistics_yefremov(2023);
```

```
mirea=# SELECT * FROM sales_statistics_yefremov(2023);
  type_name  | sales_count |     percentage     | total_sum
-------------+-------------+--------------------+-----------
 Апартаменты |           2 | 10.526315789473683 |   9800000
 Дом         |           4 | 21.052631578947366 |  40500000
 Квартира    |          13 |  68.42105263157895 |  79750000
(3 rows)
```

Практическая 1.2

1. Написать функцию, которая уменьшает стоимость объектов недвижимости

Если объект недвижимости был добавлен более 6 месяцев назад и средняя оценка ниже 6 баллов (из 10) на 5%
Если объект недвижимости был добавлен более 9 месяцев назад и средняя оценка ниже 5 баллов (из 10) на 10%
Если объект недвижимости был добавлен более 12 месяцев назад и средняя оценка ниже 4 баллов (из 10) на 20%

```
CREATE OR REPLACE FUNCTION decrease_property_prices_yefremov()
RETURNS VOID AS $$
DECLARE
    prop RECORD;
    avg_score DOUBLE PRECISION;
    months_since_listing INTEGER;
BEGIN
    FOR prop IN SELECT * FROM property LOOP
        months_since_listing := (CURRENT_DATE - prop.listing_date) / 30;

        SELECT AVG(score) INTO avg_score
        FROM evaluation
        WHERE property_id = prop.id;

        IF avg_score IS NULL THEN
            CONTINUE;
        END IF;

        IF months_since_listing > 12 AND avg_score < 4 THEN
            UPDATE property SET price = price * 0.8 WHERE id = prop.id;
        ELSIF months_since_listing > 9 AND avg_score < 5 THEN
            UPDATE property SET price = price * 0.9 WHERE id = prop.id;
        ELSIF months_since_listing > 6 AND avg_score < 6 THEN
            UPDATE property SET price = price * 0.95 WHERE id = prop.id;
        END IF;
    END LOOP;
END;
```

```
SELECT decrease_property_prices_yefremov();
```

```
mirea=# SELECT decrease_property_prices_yefremov();
 decrease_property_prices_yefremov
-----------------------------------

(1 row)
```

2. Написать функцию, которая проверяет, что дата продажи позже, чем дата добавления объявления. При обнаружении аномалии (дата продажи раньше даты размещения объявления) необходимо вывести адрес такого объекта. Если таких объектов нет, вывести соответствующее сообщение.

```
CREATE OR REPLACE FUNCTION check_sale_dates_yefremov()
RETURNS SETOF VARCHAR AS $$
DECLARE
    anomaly_count INTEGER := 0;
    addr RECORD;
BEGIN
    CREATE TEMP TABLE anomalies (address VARCHAR);

    FOR addr IN
        SELECT p.address
        FROM sale s
        JOIN property p ON s.property_id = p.id
        WHERE s.sale_date < p.listing_date
    LOOP
        INSERT INTO anomalies VALUES (addr.address);
        anomaly_count := anomaly_count + 1;
    END LOOP;

    IF anomaly_count = 0 THEN
        RETURN NEXT 'Аномалий не обнаружено';
    ELSE
        RETURN QUERY SELECT * FROM anomalies;
    END IF;

    DROP TABLE anomalies;
END;
```

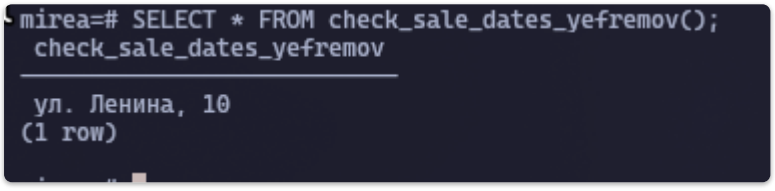
```
SELECT * FROM check_sale_dates_yefremov();
```

```
    check_sale_dates_yefremov
--------------------------------
 Аномалий не обнаружено
(1 row)
```

Добавляем аномалию:

```
UPDATE sale SET sale_date = '2022-01-01' WHERE id=1;
```

```
SELECT * FROM check_sale_dates_yefremov();
```



3Добавить таблицу «Динамика цен», где будет хранится изменения стоимости (для одного объекта недвижимости может быть несколько изменений стоимости в разные даты). Таблица будет содержать следующие колонки: код объекта недвижимости, новая стоимость, дата изменения. Написать функцию, которая будет возвращаться следующий результат по указанному объекту недвижимости:

```
CREATE TABLE price_dynamics (
    id SERIAL PRIMARY KEY,
    property_id INTEGER NOT NULL REFERENCES property(id),
    new_price DOUBLE PRECISION NOT NULL,
    change_date DATE NOT NULL
);
```

```
CREATE OR REPLACE FUNCTION property_price_history_yefremov(prop_id INTEGER)
RETURNS TABLE(change_date DATE, new_price DOUBLE PRECISION, price_change DOUBLE
PRECISION, percent_change DOUBLE PRECISION, warning VARCHAR) AS $$
DECLARE
    initial_price DOUBLE PRECISION;
    prev_price DOUBLE PRECISION;
    rec RECORD;
BEGIN
    SELECT price INTO initial_price FROM property WHERE id = prop_id;
    change_date := (SELECT listing_date FROM property WHERE id = prop_id);
    new_price := initial_price;
    price_change := 0;
    percent_change := 0;
    warning := '';
    RETURN NEXT;

    prev_price := initial_price;

    FOR rec IN
        SELECT * FROM price_dynamics
        WHERE property_id = prop_id
        ORDER BY change_date
    LOOP
        change_date := rec.change_date;
        new_price := rec.new_price;
        price_change := rec.new_price - prev_price;
        percent_change := (price_change / prev_price) * 100;
        IF ABS(percent_change) > 20 THEN
            warning := 'Больше 20%';
        ELSE
            warning := '';
        END IF;
        RETURN NEXT;
        prev_price := rec.new_price;
    END LOOP;
END;
```

```
INSERT INTO price_dynamics (property_id, new_price, change_date) VALUES (1, 6000000,
'2023-02-01');
```

```
INSERT INTO price_dynamics (property_id, new_price, change_date) VALUES (1, 4000000,
'2023-03-01');
```

```
SELECT * FROM property_price_history_yefremov(1);
```

```
mirea=# SELECT * FROM property_price_history_yefremov(1);
 change_date | new_price | price_change |   percent_change   |  warning
-------------+-----------+--------------+--------------------+-----------
 2023-01-15  |   5000000 |            0 |                  0 |
 2023-02-01  |   6000000 |      1000000 |                 20 |
 2023-03-01  |   4000000 |     -2000000 | -33.33333333333333 | Больше 20%
(3 rows)
```

4. Написать функцию, которая проверяет, что общая площадь объекта недвижимости соответствует сумме площадей всех комнат (таблица Структура объекта недвижимости). Если площади всех объектов соответствуют, вывести соответствующее сообщение, иначе – адрес объекта и размер расхождения

```
CREATE TABLE property_structure (
    id SERIAL PRIMARY KEY,
    property_id INTEGER NOT NULL REFERENCES property(id),
    room_name VARCHAR(255),
    area SMALLINT NOT NULL
);
```

```
CREATE OR REPLACE FUNCTION check_property_areas_yefremov()
RETURNS SETOF RECORD AS $$
DECLARE
    prop RECORD;
    total_room_area SMALLINT;
    discrepancy SMALLINT;
    result RECORD;
BEGIN
    FOR prop IN SELECT * FROM property LOOP
        SELECT SUM(area) INTO total_room_area FROM property_structure WHERE property_id
= prop.id;
        IF total_room_area IS NULL THEN
            total_room_area := 0;
        END IF;

        discrepancy := prop.area - total_room_area;

        IF discrepancy <> 0 THEN
            SELECT prop.address, discrepancy INTO result;
            RETURN NEXT result;
        END IF;
    END LOOP;

    IF NOT FOUND THEN
        SELECT 'Все площади соответствуют'::VARCHAR AS message, 0::SMALLINT AS
discrepancy INTO result;
        RETURN NEXT result;
    END IF;
END;
```

```
INSERT INTO property_structure (property_id, room_name, area) VALUES (1, 'Гостиная',
20), (1, 'Спальня', 30);
```

```
SELECT * FROM check_property_areas_yefremov() AS (address VARCHAR, discrepancy
SMALLINT);
```

```
mirea=# SELECT * FROM check_property_areas_yefremov() AS (address VARCHAR, discrepancy SMALLINT);
       address        | discrepancy
----------------------+-------------
 ул. Ленина, 10       |          -5
 ул. Советская, 25    |          65
 ул. Северная, 5      |          35
 ул. Лесная, 15       |          50
 ул. Южная, 30        |         120
 ул. Садовая, 8       |          55
 ул. Западная, 12     |          48
 ул. Речная, 7        |          40
 пр. Мира, 20         |          70
 ул. Сосновая, 3      |          42
 ул. Полевая, 18      |          90
 ул. Горная, 22       |          60
 ул. Центральная, 11  |          46
 ул. Берёзовая, 9     |          33
 ул. Дачная, 14       |          80
 ул. Водная, 6        |          50
 пр. Победы, 17       |          85
 ул. Снежная, 4       |          47
 ул. Солнечная, 19    |         100
 ул. Луговая, 13      |          52
 ул. Новая, 21        |          68
 ул. Старая, 2        |          30
 ул. Зелёная, 16      |         110
 ул. Каменная, 23     |          49
 ул. Торговая, 26     |          72
 ул. Школьная, 1      |          44
 ул. Парковая, 24     |          95
 ул. Моховая, 27      |          51
 ул. Береговая, 28    |          80
 ул. Хвойная, 29      |          32
(30 rows)
```

5. Написать функцию, которая рассчитывает комиссию риэлтора

Стоимость продажи: до 1 млн. руб. – 2 %
Стоимость продажи: от 1 млн. руб. до 3 млн. руб. – 1,9 %
Стоимость продажи: от 3 млн. руб.– 1,7 %
Рассчитанную премию, необходимо записать в столбец «Комиссия риэлтору» таблицы «Продажа».

```
ALTER TABLE sale ADD COLUMN realtor_commission DOUBLE PRECISION;
```

```
CREATE OR REPLACE FUNCTION calculate_realtor_commissions_yefremov()
RETURNS VOID AS $$
DECLARE
    s RECORD;
BEGIN
    FOR s IN SELECT * FROM sale LOOP
        IF s.price < 1000000 THEN
            UPDATE sale SET realtor_commission = s.price * 0.02 WHERE id = s.id;
        ELSIF s.price < 3000000 THEN
            UPDATE sale SET realtor_commission = s.price * 0.019 WHERE id = s.id;
        ELSE
            UPDATE sale SET realtor_commission = s.price * 0.017 WHERE id = s.id;
        END IF;
```

```
    END LOOP;
END;
```

```
SELECT calculate_realtor_commissions_yefremov();
```

```
SELECT realtor_commission FROM sale;
```

```
mirea=# SELECT calculate_realtor_commissions_yefremov();
 calculate_realtor_commissions_yefremov
----------------------------------------

(1 row)

mirea=#
SELECT realtor_commission FROM sale;
 realtor_commission
--------------------
 125800.00000000001
             100300
             200600
              91800
              66300
             133450
              74800
 108800.00000000001
              86700
             141100
              83300
             150450
             183600
              96900
              53200
              85000
 129200.00000000001
             163200
             141950
              84150
(20 rows)
```

Практическая 1.3

1. Создайте таблицу РАБОЧИЙ ДЕНЬ,

в которой будет храниться информация о входе/выходе сотрудника через систему контроля доступа посредством электронной карты. Считается, что в таблице хранятся данные за одну неделю. Для каждого сотрудника фиксируется время входа (значение = 1), время выхода (значение = 2). Рабочий день начинается в 9:00 часов и заканчивается в 18:00 часов. Время обеда: с 13:00 до 14:00 часов. Опоздание – это вход позднее 9:00 часов, уход на обед до 13:00, возвращение с обеда после 14:00, уход в конце рабочего дня до 18:00 часов. Заполните таблицу тестовыми данными (указать для нескольких сотрудников время входа/выхода для 1-2 рабочих дней). Фрагмент тестовых данных представлен ниже.

```
CREATE TABLE work_day (
    employee_code INTEGER NOT NULL,
    entry_time TIMESTAMP NOT NULL,
    reader_value SMALLINT NOT NULL  -- 1: вход, 2: выход
);

-- Тестовые данные (как в примере + дополнительные для 2 дней)
INSERT INTO work_day (employee_code, entry_time, reader_value) VALUES
```

```
(100, '2023-03-13 09:00:00', 1),
(100, '2023-03-13 13:00:00', 2),
(100, '2023-03-13 14:00:00', 1),
(100, '2023-03-13 18:00:00', 2),
(101, '2023-03-13 09:10:00', 1),
(101, '2023-03-13 12:42:12', 2),
(101, '2023-03-13 13:08:00', 1),
(101, '2023-03-13 18:00:00', 2),
(102, '2023-03-13 09:00:14', 1),
(102, '2023-03-13 13:00:00', 2),
(102, '2023-03-13 14:00:00', 1),
(102, '2023-03-13 19:30:10', 2),
-- День 2 для тех же сотрудников
(100, '2023-03-14 09:05:00', 1),
(100, '2023-03-14 13:10:00', 2),
(100, '2023-03-14 14:15:00', 1),
(100, '2023-03-14 18:10:00', 2),
(101, '2023-03-14 08:55:00', 1),
(101, '2023-03-14 12:50:00', 2),
(101, '2023-03-14 14:05:00', 1),
(101, '2023-03-14 17:50:00', 2),
(102, '2023-03-14 09:15:00', 1),
(102, '2023-03-14 12:55:00', 2),
(102, '2023-03-14 14:10:00', 1),
(102, '2023-03-14 18:05:00', 2);
```

```
SELECT * FROM work_day WHERE employee_code=100;
```

```
mirea=# SELECT * FROM work_day WHERE employee_code=100;
 employee_code |     entry_time      | reader_value
---------------+---------------------+--------------
           100 | 2023-03-13 09:00:00 |            1
           100 | 2023-03-13 13:00:00 |            2
           100 | 2023-03-13 14:00:00 |            1
           100 | 2023-03-13 18:00:00 |            2
           100 | 2023-03-14 09:05:00 |            1
           100 | 2023-03-14 13:10:00 |            2
           100 | 2023-03-14 14:15:00 |            1
           100 | 2023-03-14 18:10:00 |            2
(8 rows)
```

2. Создайте функцию, которая выводит общее количество часов, отработанное сотрудником за прошедшую неделю. Если сотрудник отработал меньше 40 часов, сообщить «Меньше нормы», если 40 часов – «Норма», если больше 40 часов – «Больше нормы».

```
CREATE OR REPLACE FUNCTION employee_weekly_hours_yefremov(emp_code INTEGER)
RETURNS VARCHAR AS $$
DECLARE
    total_hours DOUBLE PRECISION := 0;
    day_date DATE;
    morning_in TIMESTAMP;
    lunch_out TIMESTAMP;
    lunch_in TIMESTAMP;
    evening_out TIMESTAMP;
    rec RECORD;
BEGIN
```

```
    FOR day_date IN SELECT DISTINCT DATE(entry_time) FROM work_day WHERE employee_code =
emp_code ORDER BY DATE(entry_time) LOOP
        morning_in := NULL; lunch_out := NULL; lunch_in := NULL; evening_out := NULL;

        FOR rec IN SELECT * FROM work_day WHERE employee_code = emp_code AND
DATE(entry_time) = day_date ORDER BY entry_time LOOP
            IF rec.reader_value = 1 AND morning_in IS NULL THEN
                morning_in := rec.entry_time;
            ELSIF rec.reader_value = 2 AND lunch_out IS NULL THEN
                lunch_out := rec.entry_time;
            ELSIF rec.reader_value = 1 AND lunch_in IS NULL THEN
                lunch_in := rec.entry_time;
            ELSIF rec.reader_value = 2 AND evening_out IS NULL THEN
                evening_out := rec.entry_time;
            END IF;
        END LOOP;

        IF morning_in IS NOT NULL AND lunch_out IS NOT NULL AND lunch_in IS NOT NULL AND
evening_out IS NOT NULL THEN
            total_hours := total_hours + EXTRACT(EPOCH FROM (lunch_out - morning_in)) /
3600 + EXTRACT(EPOCH FROM (evening_out - lunch_in)) / 3600;
        END IF;
    END LOOP;

    IF total_hours < 40 THEN
        RETURN total_hours || ' часов: Меньше нормы';
    ELSIF total_hours = 40 THEN
        RETURN total_hours || ' часов: Норма';
    ELSE
        RETURN total_hours || ' часов: Больше нормы';
    END IF;
END;
```

```
SELECT employee_weekly_hours_yefremov(100);
```

```
mirea=# SELECT employee_weekly_hours_yefremov(100);
     employee_weekly_hours_yefremov
------------------------------------------
 15.999999999999998 часов: Меньше нормы
(1 row)

mirea=#
```

3. Создайте функцию, которая производит расчет заработной платы сотрудников по формуле:

`ЗАРПЛАТА=ОКЛАД + ОКЛАД*А`

ОКЛАД – базовый оклад предприятия

А - при отсутствии опоздания у сотрудника А=1, за каждые 10 мин. опоздания А уменьшается на 0,05 (А >= 0).

```
CREATE OR REPLACE FUNCTION calculate_employee_salary_yefremov(base_salary DOUBLE
PRECISION, emp_code INTEGER)
RETURNS DOUBLE PRECISION AS $$
DECLARE
    total_late_minutes DOUBLE PRECISION := 0;
```

```
    a_coefficient DOUBLE PRECISION := 1.0;
    rec RECORD;
    day_date DATE;
    morning_in TIME;
    lunch_out TIME;
    lunch_in TIME;
    evening_out TIME;
BEGIN
    FOR day_date IN SELECT DISTINCT DATE(entry_time) FROM work_day WHERE employee_code =
emp_code ORDER BY DATE(entry_time) LOOP
        morning_in := NULL; lunch_out := NULL; lunch_in := NULL; evening_out := NULL;

        FOR rec IN SELECT * FROM work_day WHERE employee_code = emp_code AND
DATE(entry_time) = day_date ORDER BY entry_time LOOP
            IF rec.reader_value = 1 AND morning_in IS NULL THEN
                morning_in := (rec.entry_time)::TIME;
            ELSIF rec.reader_value = 2 AND lunch_out IS NULL THEN
                lunch_out := (rec.entry_time)::TIME;
            ELSIF rec.reader_value = 1 AND lunch_in IS NULL THEN
                lunch_in := (rec.entry_time)::TIME;
            ELSIF rec.reader_value = 2 AND evening_out IS NULL THEN
                evening_out := (rec.entry_time)::TIME;
            END IF;
        END LOOP;

        -- Опоздания
        IF morning_in > '09:00:00' THEN
            total_late_minutes := total_late_minutes + EXTRACT(EPOCH FROM (morning_in -
'09:00:00'::TIME)) / 60;
        END IF;
        IF lunch_out < '13:00:00' THEN
            total_late_minutes := total_late_minutes + EXTRACT(EPOCH FROM
('13:00:00'::TIME - lunch_out)) / 60;
        END IF;
        IF lunch_in > '14:00:00' THEN
            total_late_minutes := total_late_minutes + EXTRACT(EPOCH FROM (lunch_in -
'14:00:00'::TIME)) / 60;
        END IF;
        IF evening_out < '18:00:00' THEN
            total_late_minutes := total_late_minutes + EXTRACT(EPOCH FROM
('18:00:00'::TIME - evening_out)) / 60;
        END IF;
    END LOOP;

    a_coefficient := a_coefficient - (FLOOR(total_late_minutes / 10) * 0.05);
    IF a_coefficient < 0 THEN
        a_coefficient := 0;
    END IF;

    RETURN base_salary + base_salary * a_coefficient;
END;
```

```
SELECT calculate_employee_salary_yefremov(100000, 101);
```

```
CREATE FUNCTION
mirea=# SELECT calculate_employee_salary_yefremov(100000, 101);
 calculate_employee_salary_yefremov
------------------------------------
                             175000
(1 row)

mirea=#
```