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

Институт перспективных технологий и индустриального программирования
(ИПТИП)

ОТЧЕТ ПО ПРАКТИЧЕСКИМ РАБОТАМ

по дисциплине

«Базы данных и анализ промышленных данных»

Практическая работа №7

Выполнил студент группы ЭФМО-02-23 Мурадов Н.Н.

Москва 2023

СОДЕРЖАНИЕ

Задача

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 13:30 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.

Код сотрудника	**Дата/время входа/выхода**	**Значение считывателя карт**
100	13.03.2023,09:00:00	1
100	13.03.2023,13:00:00	2
100	13.03.2023,14:00:00	1
100	13.03.2023,18:00:00	2
101	13.03.2023,09:10:00	1
101	13.03.2023,12:42:12	2
101	13.03.2023,13:08:00	1
101	13.03.2023,18:00:00	2
102	13.03.2023,09:00:14	1
102	13.03.2023,13:00:00	2
102	13.03.2023,14:00:00	1
102	13.03.2023,19:30:10	2
…	…	

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».
3. Создайте функцию, которая выводит общее количество часов, проведенное сотрудником за обедом. Время обеда: выход строго с 13:00-13:30.
4. Создайте функцию, которая выводит ТОП-5 самых опоздавших сотрудников за прошедшую неделю.
5. 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.

6. На указанный день произвести расчет количества работающих сотрудников (отработавших ровно 60 минут). Результаты разбить по каждому часу.

9:00:00-9:59:59	5

10:00:00-10:59:59	4

Решение

Листинг кода:

Создание БД:

```
CREATE DATABASE department;

\c department;

1:

CREATE TABLE WORKING_DAY (
	ID BIGINT PRIMARY KEY,
	EMPLOYEE_ID BIGINT,
	TIME_OF_TRANSFER TIMESTAMP,
	VALUE BIGINT
);

CREATE OR REPLACE FUNCTION getDopTime7_1(ID BIGINT, day interval = interval '1 days')
RETURNS interval AS $$
DECLARE
	hr interval := interval '1 hours';
	other interval := interval '1 minutes' * rand(0, 59) + interval '1 seconds' * rand(0, 59);
BEGIN
	hr := (SELECT CASE
		WHEN MOD(ID, 4) = 0 THEN hr * rand(6, 10)
		WHEN MOD(ID, 4) = 1 THEN hr * rand(11, 13)
		WHEN MOD(ID, 4) = 2 THEN hr * rand(13, 15)
		WHEN MOD(ID, 4) = 3 THEN hr * rand(16, 20)
	END);
	RETURN day + hr + other;
END $$ language plpgsql;

CREATE OR REPLACE FUNCTION getTable7_1(countEmp INTEGER, countTransfer BIGINT)
RETURNS TABLE(ID BIGINT, EMPLOYEE_ID BIGINT, TIME_OF_TRANSFER TIMESTAMP, VALUE BIGINT) AS $$
DECLARE
	startData TIMESTAMP;
	maxId BIGINT;
	id BIGINT := 0;
	day interval;
	empId BIGINT;
BEGIN
	delete FROM WORKING_DAY WHERE TRUE;
	startData := (select date_trunc('week', now()) - interval '7 days' * rand(1,30));
	for i in 0..countTransfer loop
		day := interval '1 days' * rand(0,6);
		empId := rand(50, 50 + countEmp);
		for i1 in 0..3 loop
```

```
                                INSERT INTO WORKING_DAY SELECT
                                        id, empId, startData + getDopTime7_1(i1, day), CASE
WHEN MOD(id, 2) = 0 THEN 1 ELSE 2 END;
                                id := id + 1;
                        end loop;
                end loop;
                RETURN query(SELECT * FROM WORKING_DAY);
        END $$ language plpgsql;

        SELECT * FROM getTable7_1(15, 100) limit 15;

        SELECT * FROM getDopTime7_1();

        DROP FUNCTION getTable7_1(countEmp BIGINT, countTransfer BIGINT);

        DROP FUNCTION getDopTime7_1(ID BIGINT, day interval);

        2:

        WITH x AS (SELECT *, (CASE
                        WHEN VALUE = 2 THEN TIME_OF_TRANSFER::TIME::INTERVAL
- (LAG(TIME_OF_TRANSFER) OVER())::TIME
                        ELSE 'allballs'::TIME::INTERVAL
                END) hR FROM WORKING_DAY WHERE EMPLOYEE_ID = 62 ORDER BY
id DESC)
                SELECT EMPLOYEE_ID, extract(hour from SUM(hr)) FROM x GROUP BY
EMPLOYEE_ID;

        SELECT EMPLOYEE_ID, COUNT(*) FROM WORKING_DAY GROUP BY
EMPLOYEE_ID;

        CREATE OR REPLACE FUNCTION getInfo7_2(idEMP BIGINT)
        RETURNS TABLE(hrINT BIGINT, message TEXT) AS $$
        DECLARE
                hrINT BIGINT;
                message TEXT;
        BEGIN
                DROP TABLE IF EXISTS x;
                create temporary table x as SELECT *, (CASE
                        WHEN VALUE = 2 THEN TIME_OF_TRANSFER::TIME::INTERVAL
- (LAG(TIME_OF_TRANSFER) OVER())::TIME
                        ELSE 'allballs'::TIME::INTERVAL
                END) hR FROM WORKING_DAY WHERE EMPLOYEE_ID = idEMP
ORDER BY id DESC;
                hrINT := (SELECT extract(hour from SUM(hr)) FROM x);
                message := (SELECT CASE
                        WHEN hrINT < 40 THEN 'Меньше нормы'
                        WHEN hrINT > 40 THEN 'Больше нормы'
                        ELSE 'Норма'
                END);
                RETURN query (SELECT hrINT, message);
        END $$ language plpgsql;
```

```
	SELECT * FROM getInfo7_2(62);

	DROP FUNCTION getInfo7_2(idEMP BIGINT);

	3:

	SELECT *, (CASE
			WHEN MOD(ID, 4) = 2 THEN
TIME_OF_TRANSFER::TIME::INTERVAL - (LAG(TIME_OF_TRANSFER)
OVER())::TIME
			ELSE 'allballs'::TIME::INTERVAL
		END) hR FROM WORKING_DAY WHERE EMPLOYEE_ID = 62 ORDER BY
id DESC;

	CREATE OR REPLACE FUNCTION getInfo7_3(idEMP BIGINT)
	RETURNS TIME AS $$
	BEGIN
		DROP TABLE IF EXISTS x;
		create temporary table x as SELECT *, (CASE
			WHEN MOD(ID, 4) = 2 THEN
TIME_OF_TRANSFER::TIME::INTERVAL - (LAG(TIME_OF_TRANSFER)
OVER())::TIME
			ELSE 'allballs'::TIME::INTERVAL
		END) hR FROM WORKING_DAY WHERE EMPLOYEE_ID = idEMP
ORDER BY id DESC;
		RETURN (SELECT SUM(hr) FROM x);
	END $$ language plpgsql;

	SELECT * FROM getInfo7_3(62);

	DROP FUNCTION getInfo7_3(idEMP BIGINT);

	4:

	SELECT *, (CASE
			WHEN MOD(ID, 4) = 0 THEN TIME_OF_TRANSFER::TIME -
INTERVAL '9 hours'
			WHEN MOD(ID, 4) = 1 THEN '13 hours'::INTERVAL::TIME -
TIME_OF_TRANSFER::TIME::INTERVAL
			WHEN MOD(ID, 4) = 2 THEN TIME_OF_TRANSFER::TIME -
INTERVAL '14 hours'
			WHEN MOD(ID, 4) = 3 THEN '18 hours'::INTERVAL::TIME -
TIME_OF_TRANSFER::TIME::INTERVAL
		END) hR FROM WORKING_DAY WHERE EMPLOYEE_ID = 62;

	CREATE OR REPLACE FUNCTION getEmp7_4(idEMP BIGINT)
	RETURNS TIME AS $$
	BEGIN
		DROP TABLE IF EXISTS x, x1;
		create temporary table x as SELECT *, (CASE
```

```
                        WHEN MOD(ID, 4) = 0 THEN TIME_OF_TRANSFER::TIME -
INTERVAL '9 hours'
                        WHEN MOD(ID, 4) = 1 THEN '13 hours'::INTERVAL::TIME -
TIME_OF_TRANSFER::TIME::INTERVAL
                        WHEN MOD(ID, 4) = 2 THEN TIME_OF_TRANSFER::TIME -
INTERVAL '14 hours'
                        WHEN MOD(ID, 4) = 3 THEN '18 hours'::INTERVAL::TIME -
TIME_OF_TRANSFER::TIME::INTERVAL
                END) hR FROM WORKING_DAY WHERE EMPLOYEE_ID = idEMP;
                create temporary table x1 as SELECT x.id, x.EMPLOYEE_ID,
x.TIME_OF_TRANSFER, x.VALUE, (CASE
                        WHEN x.hR > (interval '15 hours') THEN 'allballs'::TIME
                        ELSE x.hR
                END) hR FROM x;
                RETURN (SELECT SUM(hr) FROM x1);
        END $$ language plpgsql;

        CREATE OR REPLACE FUNCTION getInfo7_4()
        RETURNS TABLE(EMPLOYEE_ID BIGINT, tim TIME) AS $$
        BEGIN
                RETURN query(SELECT w.EMPLOYEE_ID, getEmp7_4(w.EMPLOYEE_ID)
tim FROM WORKING_DAY w GROUP BY w.EMPLOYEE_ID ORDER BY tim DESC limit
5);
        END $$ language plpgsql;

        SELECT * FROM getInfo7_4();

        DROP FUNCTION getInfo7_4();

        5:

        CREATE OR REPLACE FUNCTION interval_div(TIME, interval) RETURNS BIGINT
AS $$
          SELECT EXTRACT(EPOCH FROM $1) / EXTRACT(EPOCH FROM $2)
        $$ LANGUAGE SQL;

        CREATE OPERATOR / (
          FUNCTION = interval_div,
          LEFTARG = TIME,
          RIGHTARG = interval);

        CREATE OR REPLACE FUNCTION getInfo7_5(SALARY BIGINT)
        RETURNS TABLE(EMPLOYEE_ID BIGINT, tim TIME, pay FLOAT) AS $$
        BEGIN
                RETURN query(SELECT w.EMPLOYEE_ID, getEmp7_4(w.EMPLOYEE_ID)
tim, (SALARY + SALARY * (1 - (getEmp7_4(w.EMPLOYEE_ID) / INTERVAL '10 minutes')
* 0.005))::FLOAT FROM WORKING_DAY w GROUP BY w.EMPLOYEE_ID ORDER BY
tim);
        END $$ language plpgsql;

        SELECT * FROM getInfo7_5(25000);
```

```
DROP FUNCTION getInfo7_5(SALARY BIGINT);

DROP FUNCTION interval_div(TIME, interval) CASCADE;

6:

SELECT EXTRACT(DAY FROM TIME_OF_TRANSFER) d, COUNT(*) FROM
WORKING_DAY GROUP BY d ORDER BY d;

SELECT FORMAT('%1$s:00:00-%1$s:59:59', EXTRACT(HOUR FROM
TIME_OF_TRANSFER)) h, COUNT(*) FROM WORKING_DAY WHERE EXTRACT(DAY
FROM TIME_OF_TRANSFER) = 21 GROUP BY h;

CREATE OR REPLACE FUNCTION getInfo7_6(DAY TIMESTAMP)
RETURNS TABLE(h TEXT, COUNT BIGINT) AS $$
BEGIN
    DROP TABLE IF EXISTS x;
    create temporary table x as SELECT EXTRACT(HOUR FROM
TIME_OF_TRANSFER) hINT, COUNT(*) cnt FROM WORKING_DAY WHERE
EXTRACT(DAY FROM TIME_OF_TRANSFER) = EXTRACT(DAY FROM DAY) GROUP
BY hINT;
    RETURN query(SELECT FORMAT('%1$s:00:00-%1$s:59:59', x.hINT) h, x.cnt
FROM x ORDER BY x.hINT);
END $$ language plpgsql;

SELECT * FROM getInfo7_6('2023-11-21');

DROP FUNCTION getInfo7_6(DAY TIMESTAMP);
```

Результаты

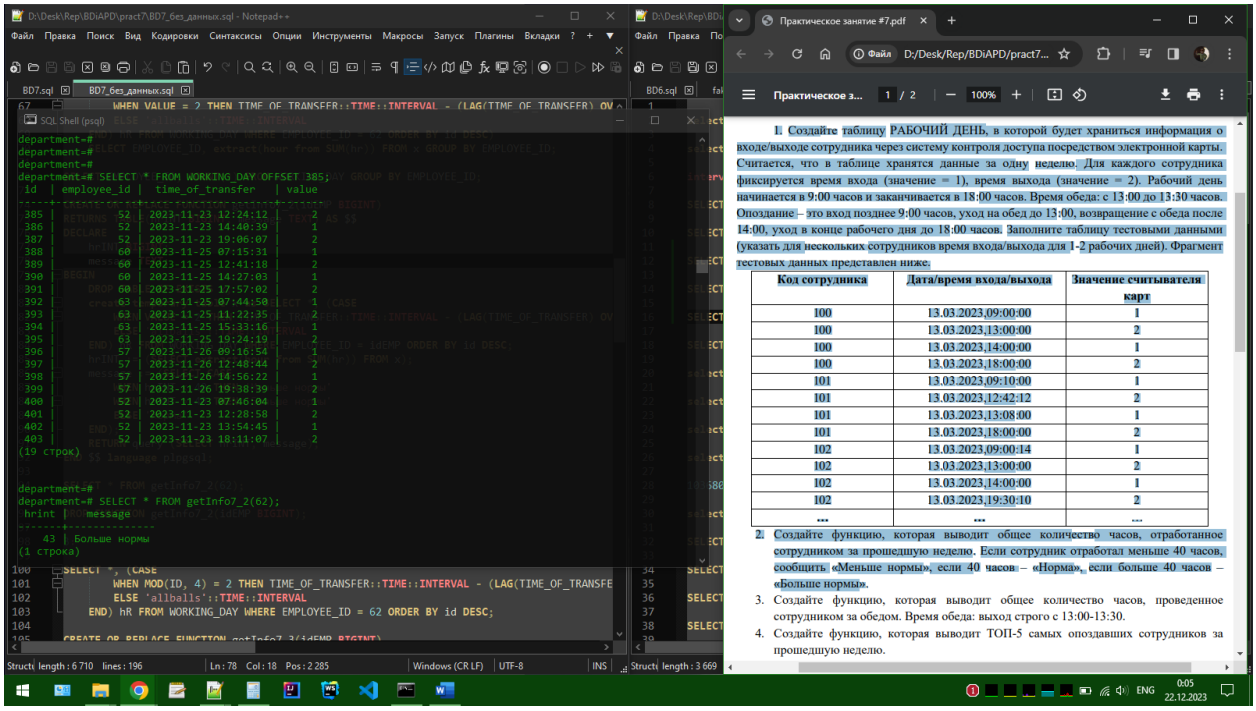

Рисунок 1 – Результат к 1 и 2 задаче

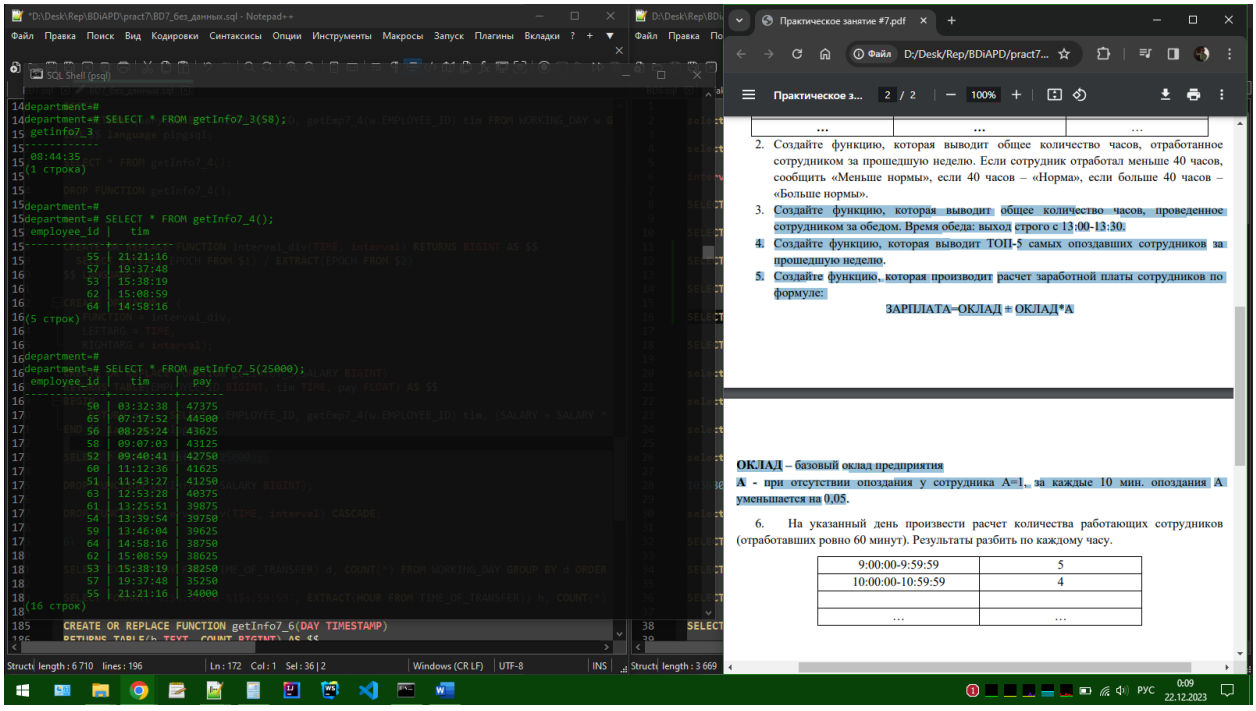

Рисунок 2 – Результат к 3, 4 и 5 функции

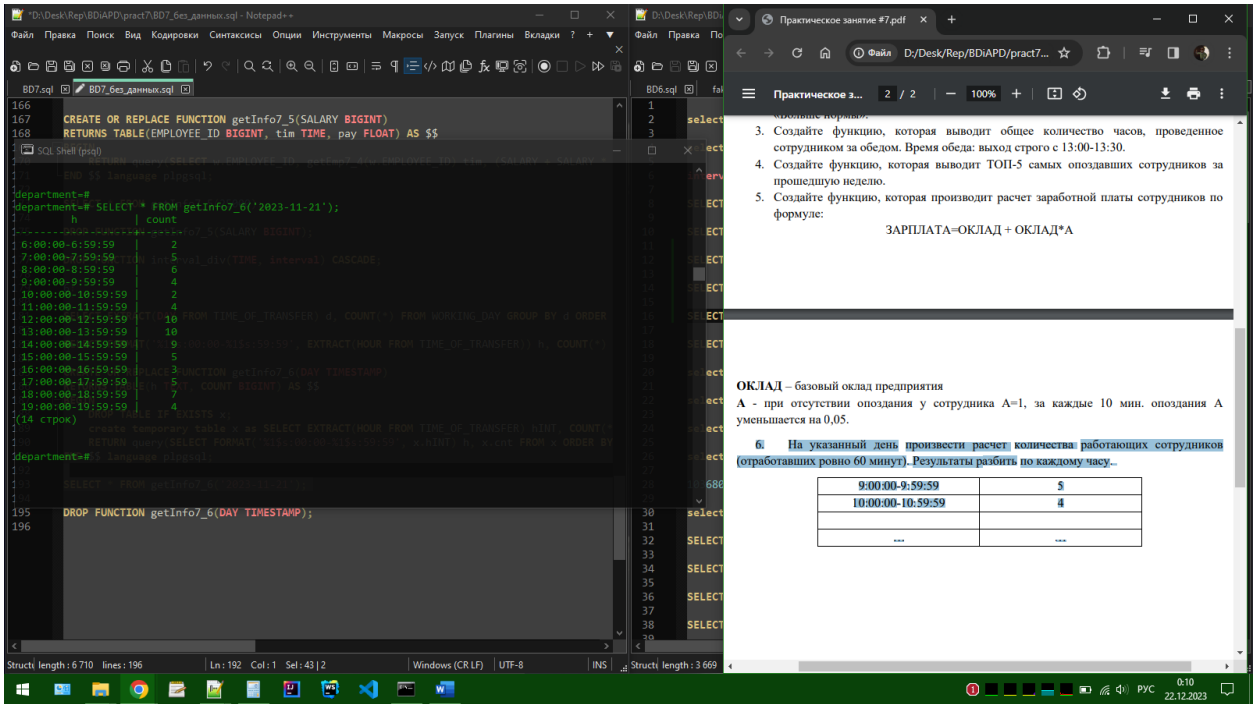

Рисунок 3 – Результат к 6 функции