

1. Выполнить проектирование хранимой процедуру (ХП), обеспечивающих начисление заработной платы (ЗП) каждому работнику в указанный месяц заданного года.

**Формулы начисления ЗП**

для начальников и тех рабочих, которые отсутствовали на рабочем месте по уважительной причине.

Для рабочих, которые трудились за станком

Логика работы ХП должна предусматривать:

* организацию корректного выхода из процедуры, предусматривающую освобождение памяти от временной таблицы;
* после обработки всех рабочих смен выполнение в процедуре операции переноса данных в таблицу *t\_salary*.

CREATE OR REPLACE PROCEDURE sp\_calculate\_salary (

p\_month INT, -- Месяц (1-12)

p\_year INT -- Год (например, 2025)

)

LANGUAGE plpgsql

AS $$

DECLARE

v\_shift\_count INT; -- Количество рабочих дней в месяце

BEGIN

-- 1. Подсчет рабочих дней в месяце

SELECT COUNT(\*) INTO v\_shift\_count

FROM t\_date\_work

WHERE date\_part('month', date\_x) = p\_month

AND date\_part('year', date\_x) = p\_year;

-- 2. Создание временной таблицы для расчетов по всем цехам

DROP TABLE IF EXISTS t\_temp\_salary\_detail;

CREATE TEMP TABLE t\_temp\_salary\_detail AS

SELECT

p.id AS id\_people,

p.fam,

p.passport,

po.post,

po.post\_salary,

pl.k\_place,

st.id AS status\_id,

st.k\_status,

w.value\_x,

w.defect\_x,

w.price\_x,

d.date\_x

FROM t\_people p

JOIN t\_ppp pp ON p.id = pp.id\_people

JOIN t\_post po ON pp.id\_post = po.id

JOIN t\_place pl ON pp.id\_place = pl.id

JOIN t\_work w ON p.id = w.id\_people

JOIN t\_status st ON w.id\_status = st.id

JOIN t\_date\_work d ON w.id\_date = d.id

WHERE

pp.date\_decree = (

SELECT MAX(px.date\_decree)

FROM t\_ppp px

WHERE px.id\_people = p.id AND px.date\_decree <= d.date\_x

)

AND date\_part('month', d.date\_x) = p\_month

AND date\_part('year', d.date\_x) = p\_year;

-- 3. Агрегация данных по сотрудникам

DROP TABLE IF EXISTS t\_temp\_salary;

CREATE TEMP TABLE t\_temp\_salary AS

SELECT

id\_people,

fam,

passport,

-- Берем последний оклад

(SELECT post\_salary FROM t\_temp\_salary\_detail d

WHERE d.id\_people = t.id\_people

ORDER BY date\_x DESC LIMIT 1) AS post\_salary,

-- Берем последний коэффициент цеха

(SELECT k\_place FROM t\_temp\_salary\_detail d

WHERE d.id\_people = t.id\_people

ORDER BY date\_x DESC LIMIT 1) AS k\_place,

-- Суммируем дни по всем цехам

SUM(CASE WHEN status\_id = 2 THEN 1 ELSE 0 END) AS day\_bus\_trip,

SUM(CASE WHEN status\_id = 3 THEN 1 ELSE 0 END) AS day\_disease,

SUM(CASE WHEN status\_id = 4 THEN 1 ELSE 0 END) AS day\_vacation,

COUNT(status\_id) AS day\_all,

-- Расчет зарплаты по компонентам

ROUND(SUM(

CASE

WHEN status\_id = 2 THEN (post\_salary \* k\_status \* k\_place) / v\_shift\_count

ELSE 0

END

), 2) AS salary\_bt,

ROUND(SUM(

CASE

WHEN status\_id = 3 THEN (post\_salary \* k\_status \* k\_place) / v\_shift\_count

ELSE 0

END

), 2) AS salary\_d,

ROUND(SUM(

CASE

WHEN status\_id = 4 THEN (post\_salary \* k\_status \* k\_place) / v\_shift\_count

ELSE 0

END

), 2) AS salary\_v,

-- Общая зарплата

ROUND(SUM(

CASE

WHEN post = 'рабочий' AND status\_id > 1 THEN

(post\_salary \* k\_status \* k\_place) / v\_shift\_count

WHEN post = 'рабочий' AND status\_id = 1 THEN

(value\_x - defect\_x) \* price\_x

WHEN post IN ('начальник','заместитель','бригадир') THEN

(post\_salary \* k\_status \* k\_place) / v\_shift\_count

ELSE 0

END

), 2) AS salary\_all

FROM t\_temp\_salary\_detail t

GROUP BY id\_people, fam, passport;

-- 4. Сохранение результатов в основную таблицу

IF NOT EXISTS (SELECT \* FROM information\_schema.tables WHERE table\_name = 't\_salary') THEN

CREATE TABLE t\_salary (

fam character varying(20),

passport character varying(10),

month\_x double precision,

year\_x double precision,

day\_bus\_trip bigint,

day\_disease bigint,

day\_vacation bigint,

day\_all bigint,

salary\_bt numeric,

salary\_d numeric,

salary\_v numeric,

salary\_all numeric

);

END IF;-- Удаляем старые данные за этот месяц

DELETE FROM t\_salary

WHERE month\_x = p\_month AND year\_x = p\_year;

-- Вставляем новые данные

INSERT INTO t\_salary (

fam, passport, month\_x, year\_x,

day\_bus\_trip, day\_disease, day\_vacation, day\_all,

salary\_bt, salary\_d, salary\_v, salary\_all

)

SELECT

fam, passport, p\_month::double precision, p\_year::double precision,

day\_bus\_trip, day\_disease, day\_vacation, day\_all,

salary\_bt, salary\_d, salary\_v, salary\_all

FROM t\_temp\_salary;

RAISE NOTICE 'Зарплата за %-% успешно рассчитана', p\_month, p\_year;

END;

$$;

--Рассчитываем ЗП с января по май 2025 года.

CALL sp\_calculate\_salary(1, 2025);

CALL sp\_calculate\_salary(2, 2025);

CALL sp\_calculate\_salary(3, 2025);

CALL sp\_calculate\_salary(4, 2025);

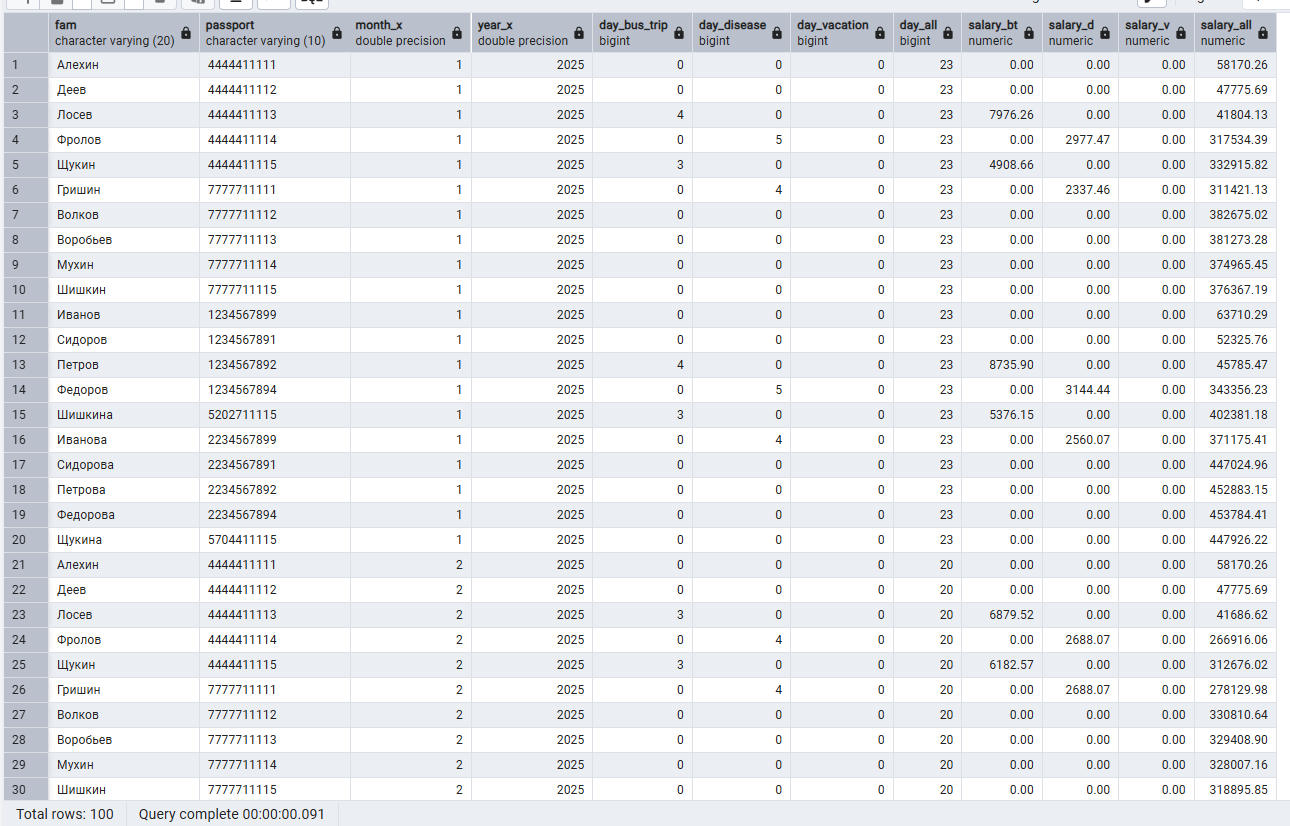
CALL sp\_calculate\_salary(5, 2025);

--Выводим данные. Группируем по месяцу.

SELECT \*

FROM public.t\_salary

order by month\_x;



1. Выполнить проектирование ХП, обеспечивающей вывод результатов работников, которые отсутствовали на рабочем месте в виде:

CREATE OR REPLACE PROCEDURE sp\_employee\_absences\_report(

m\_x INT,

y\_x INT

)

LANGUAGE 'plpgsql'

AS $BODY$

DECLARE

cnt\_shift INT := 1;

BEGIN

-- Расчет количества рабочих дней в месяце

SELECT COUNT(date\_x)

INTO cnt\_shift

FROM t\_date\_work

WHERE date\_part('month', date\_x) = m\_x AND

date\_part('year', date\_x) = y\_x;

DROP TABLE IF EXISTS t\_temp;

-- Создание временной таблицы с данными о работе

CREATE TEMP TABLE t\_temp ON COMMIT DROP AS

SELECT

d.date\_x,

w.id\_people,

w.id\_status,

px.id\_post,

px.id\_place,

w.value\_x,

w.defect\_x,

w.price\_x,

ROUND(po.post\_salary \* pl.k\_place \* st.k\_status / cnt\_shift, 2) AS day\_salary,

date\_part('month', d.date\_x) AS month\_x,

date\_part('year', d.date\_x) AS year\_x,

pl.place AS workshop

FROM

t\_date\_work AS d

JOIN t\_work AS w ON d.id = w.id\_date

JOIN t\_status AS st ON st.id = w.id\_status

JOIN t\_ppp AS px ON w.id\_people = px.id\_people

JOIN t\_post AS po ON px.id\_post = po.id

JOIN t\_place AS pl ON px.id\_place = pl.id

WHERE

px.date\_decree = (

SELECT MAX(q.date\_decree)

FROM t\_ppp AS q

WHERE px.id\_people = q.id\_people AND q.date\_decree <= d.date\_x

) AND

date\_part('month', d.date\_x) = m\_x AND

date\_part('year', d.date\_x) = y\_x;

-- Создание или обновление таблицы отчетности

IF NOT EXISTS (

SELECT \*

FROM information\_schema.tables

WHERE table\_name = 't\_employee\_absences'

AND table\_schema = 'public'

) THEN

-- Создание таблицы отчетности, если она не существует

CREATE TABLE t\_employee\_absences AS

SELECT

t.workshop,

t.month\_x,

t.year\_x AS year\_x,

COUNT(CASE WHEN t.id\_status = 2 THEN 1 END) AS days\_business\_trip,

COUNT(CASE WHEN t.id\_status = 3 THEN 1 END) AS days\_sick\_leave,

COUNT(CASE WHEN t.id\_status = 4 THEN 1 END) AS days\_vacation,

COUNT(CASE WHEN t.id\_status > 1 THEN 1 END) AS total\_days\_absence,

COALESCE(SUM(CASE WHEN t.id\_status = 2 THEN t.day\_salary END), 0) AS salary\_business\_trip,

COALESCE(SUM(CASE WHEN t.id\_status = 3 THEN t.day\_salary END), 0) AS salary\_sick\_leave,

COALESCE(SUM(CASE WHEN t.id\_status = 4 THEN t.day\_salary END), 0) AS salary\_vacation,

COALESCE(SUM(CASE WHEN t.id\_status > 1 THEN t.day\_salary END), 0) AS total\_salary\_absences

FROM

t\_temp t

GROUP BY

t.workshop, t.month\_x, t.year\_x;

ELSE

-- Проверка, существуют ли уже данные за указанный период

IF NOT EXISTS (

SELECT \*

FROM t\_employee\_absences

WHERE month\_x = m\_x AND year\_x = y\_x

) THEN

-- Вставка новых данных

INSERT INTO t\_employee\_absences (

workshop,

month\_x,

year\_x,

days\_business\_trip,

days\_sick\_leave,

days\_vacation,

total\_days\_absence,

salary\_business\_trip,

salary\_sick\_leave,

salary\_vacation,

total\_salary\_absences

)

SELECT

t.workshop,

t.month\_x,

t.year\_x,

COUNT(CASE WHEN t.id\_status = 2 THEN 1 END) AS days\_business\_trip,

COUNT(CASE WHEN t.id\_status = 3 THEN 1 END) AS days\_sick\_leave,

COUNT(CASE WHEN t.id\_status = 4 THEN 1 END) AS days\_vacation,

COUNT(CASE WHEN t.id\_status > 1 THEN 1 END) AS total\_days\_absence,

COALESCE(SUM(CASE WHEN t.id\_status = 2 THEN t.day\_salary END), 0) AS salary\_business\_trip,COALESCE(SUM(CASE WHEN t.id\_status = 3 THEN t.day\_salary END), 0) AS salary\_sick\_leave,

COALESCE(SUM(CASE WHEN t.id\_status = 4 THEN t.day\_salary END), 0) AS salary\_vacation,

COALESCE(SUM(CASE WHEN t.id\_status > 1 THEN t.day\_salary END), 0) AS total\_salary\_absences

FROM

t\_temp t

GROUP BY

t.workshop, t.month\_x, t.year\_x;

ELSE

RAISE NOTICE 'Данные за указанный период уже существуют!';

END IF;

END IF;

-- Вывод результатов

RAISE NOTICE 'Отчет по отсутствиям сотрудников за %/%:', m\_x, y\_x;

END;

$BODY$;

--Расчет и добавление данных

CALL sp\_employee\_absences\_report(1, 2025);

CALL sp\_employee\_absences\_report(2, 2025);

CALL sp\_employee\_absences\_report(3, 2025);

CALL sp\_employee\_absences\_report(4, 2025);

CALL sp\_employee\_absences\_report(5, 2025);

--Просмотр данных

SELECT \*

FROM public.t\_employee\_absences;

