

# Задание 1

В данном задании вам предлагается написать несколько SQL-запросов для определенных таблиц. В каждом случае вам НЕ будет предоставлена таблица в явном виде. Однако, вам будет дан пример таблицы с данными.

**КОММЕНТАРИЙ:** для демонстрации работы запросов и их проверки создадим произвольные таблицы с аналогичными данными в СУБД pgAdmin 4.

## ЗАДАНИЕ 1.1

Посчитать % изменение количества клиентов, совершивших покупку, месяц-к-месяцу.

**КОММЕНТАРИЙ:** Запрос считает показатель количество уникальных пользователей, совершивших покупку, месяц-к-месяцу. При этом учитывается, что если в предыдущем месяце не было заказов, то прирост составляет 100%.

### ЗАПРОС:

```
SELECT
    EXTRACT(YEAR FROM sq.year_month) AS year,
    EXTRACT(MONTH FROM sq.year_month) AS month,
    sq.unique_clnt AS count_unique_clients,
    ROUND(CASE
        WHEN sq.priv_clnt IS NULL THEN 100.00 -- В первый месяц прирост клиентов месяц-к-месяцу был 100%
        WHEN sq.orders_delta_t > '31 days' THEN 100.00 --Учитываем, что пользователи могли ничего не
покупать больше месяца
        ELSE (sq.unique_clnt - sq.priv_clnt)/CAST(sq.priv_clnt AS numeric) * 100.00
    END, 2) growth_month_to_month
FROM (
    SELECT -- Таблица с информацией по уникальным клиентам за этот и предыдущий месяц
        date_trunc('MONTH', order_date) AS year_month,
        date_trunc('MONTH', order_date) - (LAG(date_trunc('MONTH', order_date)) OVER w)
        AS orders_delta_t, -- Оценочное время отсутствия заказов
        COUNT(DISTINCT client_id) AS unique_clnt,
        LAG(COUNT(DISTINCT client_id)) OVER w AS priv_clnt
    FROM clients_orders
    GROUP BY date_trunc('MONTH', order_date)
    WINDOW w AS ()
) AS sq
```

### ДЕМОНСТРАЦИЯ РАБОТЫ:

таблица **clients\_orders** и запрос:

	client_id integer	order_id integer	order_date date
1	4	14486	2021-02-05
2	2	14634	2021-02-23
3	6	14825	2021-02-12
4	5	14232	2021-03-09
5	9	14957	2021-03-21
6	4	14486	2021-05-02
7	2	14734	2021-06-19
8	1	14826	2021-10-02
9	5	14233	2021-10-09
10	9	1952	2021-12-04

```
1 SELECT
2   EXTRACT(YEAR FROM sq.year_month) AS year,
3   EXTRACT(MONTH FROM sq.year_month) AS month,
4   sq.unique_clnt AS count_unique_clients,
5   ROUND(CASE
6       WHEN sq.priv_clnt IS NULL THEN 100.00 -- В первый месяц прирост клиентов месяц-к-месяцу был 100%
7       WHEN sq.orders_delta_t > '31 days' THEN 100.00 --Учитываем, что пользователи могли ничего не покупать больше месяца
8       ELSE (sq.unique_clnt - sq.priv_clnt)/CAST(sq.priv_clnt AS numeric) * 100.00
9   END, 2) growth_month_to_month
10  FROM (
11      SELECT -- Таблица с информацией по уникальным клиентам за этот и предыдущий месяц
12          date_trunc('MONTH', order_date) AS year_month,
13          date_trunc('MONTH', order_date) - (LAG(date_trunc('MONTH', order_date)) OVER w)
14          AS orders_delta_t, -- Оценочное время отсутствия заказов
15          COUNT(DISTINCT client_id) AS unique_clnt,
16          LAG(COUNT(DISTINCT client_id)) OVER w AS priv_clnt
17      FROM clients_orders
18      GROUP BY date_trunc('MONTH', order_date)
19      WINDOW w AS ()
20  ) AS sq
```

Результирующая таблица:

	year numeric	month numeric	count_unique_clients bigint	growth_month_to_month numeric
1	2016	2	1	100.00
2	2017	7	1	100.00
3	2019	3	1	100.00
4	2020	1	1	100.00
5	2020	2	3	200.00
6	2020	3	2	-33.33
7	2020	4	1	-50.00
8	2020	5	1	0.00
9	2020	11	2	100.00
10	2020	12	2	0.00

## ЗАДАНИЕ 1.2

Вывести сумму GMV (Gross Merchandise Value) с нарастающим итогом по дням.

### ЗАПРОС:

```
SELECT
    *,
    SUM(gmv) OVER w AS accumulative_sum_gmv
FROM gmv_info
WINDOW w AS (
    ORDER BY fact_date
    ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
)
ORDER BY fact_date
```

### ДЕМОНСТРАЦИЯ РАБОТЫ:

таблица **gmv\_info** и запрос:

	fact_date date	gmv integer	
1	2021-01-01	4372888	1
2	2021-01-02	3282842	2
3	2021-01-03	3242547	3
4	2021-01-04	3462848	4
5	2021-01-05	4322589	5
6	2021-01-06	4642443	6
7	2021-01-07	9452641	7
8	2021-01-08	8782546	8
9	2021-01-09	5292243	9
10	2021-01-10	2212341	

```
1 SELECT
2     *,
3     SUM(gmv) OVER w AS accumulative_sum_gmv
4 FROM gmv_info
5 WINDOW w AS (
6     ORDER BY fact_date
7     ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
8 )
9 ORDER BY fact_date
```

Результирующая таблица:

	fact_date date	gmv integer	accumulative_sum_gmv bigint
1	2021-01-01	4372888	4372888
2	2021-01-02	3282842	7655730
3	2021-01-03	3242547	10898277
4	2021-01-04	3462848	14361125
5	2021-01-05	4322589	18683714
6	2021-01-06	4642443	23326157
7	2021-01-07	9452641	32778798
8	2021-01-08	8782546	41561344
9	2021-01-09	5292243	46853587
10	2021-01-10	2212341	49065928

## ЗАДАНИЕ 1.3

Получить время отклика на каждое письмо (письмо идентифицируется по полю mail\_id), отправленное пользователем mr\_employee@ozon.ru.

Дана таблица с логом электронных писем пользователя mr\_employee@ozon.ru (т.е. письма, отправленные с этой электронной почты и полученные на нее). У всех цепочек сообщений уникальная тема. В одной цепочке может быть несколько писем.

**КОММЕНТАРИЙ:** Подзапрос sq считает время отклика для всех сообщений в каждой линейке писем. Внешнем запросом мы выбираем исходящие сообщения от [mr\\_employee@ozon.ru](mailto:mr_employee@ozon.ru). Если на сообщение не пришло ответа то значение отклика будет Null.

### ЗАПРОС:

--Таблица отправленных сообщений пользователем с временем отклика

```
SELECT *
FROM (
    SELECT
        *,
        LEAD(timestamp) OVER w - timestamp AS response_time
    FROM mails
    WINDOW w AS (
        PARTITION BY mail_subject
        ORDER BY timestamp
    )
) AS sq
WHERE mail_from = 'mr_employee@ozon.ru'
```

### ДЕМОНСТРАЦИЯ РАБОТЫ:

таблица **mails** и запрос:

	mail_id [PK] integer	mail_from text	mail_to text	mail_subject text	timestamp timestamp without time zone
1	1	mr_employee@ozon.ru	mr_intern@ozon.ru	Задание для практики	2021-01-08 12:00:03
2	2	mr_intern@ozon.ru	mr_employee@ozon.ru	Задание для практики	2021-01-10 13:41:34
3	3	mr_employee@ozon.ru	mr_boss@ozon.ru	Отчет по продажам 2021-01-10	2021-01-11 15:02:57
4	4	mr_boss@ozon.ru	mr_employee@ozon.ru	Отчет по продажам 2021-01-10	2021-01-18 11:03:08
5	5	stranger@ozon.ru	mr_employee@ozon.ru	Хотите заработать?	2021-01-18 12:16:44
6	6	mr_employee@ozon.ru	mr_intern@ozon.ru	Задание для практики	2021-01-20 19:48:54
7	7	mr_employee@ozon.ru	mr_intern2@ozon.ru	Новое задание	2021-01-21 16:41:32
8	8	mr_intern2@ozon.ru	mr_employee@ozon.ru	Новое задание	2021-01-21 17:29:04
9	9	mr_employee@ozon.ru	mrs_boss@ozon.ru	Отчет по продажам 2021-01-20	2021-01-22 15:02:57
10	10	stranger2@ozon.ru	mr_employee@ozon.ru	Выучить английский за 1 день	2021-01-23 17:16:44

```

1  --Таблица отправленных сообщений пользователем с временем отклика
2  SELECT *
3  FROM (
4      SELECT
5          *,
6          LEAD(timestamp) OVER w - timestamp AS response_time
7      FROM mails
8      WINDOW w AS (
9          PARTITION BY mail_subject
10         ORDER BY timestamp
11     )
12 ) AS sq
13 WHERE mail_from = 'mr_employee@ozon.ru'

```

Результирующая таблица:

	mail_id [PK] integer	mail_from text	mail_to text	mail_subject text	timestamp timestamp without time zone	response_time interval
1	1	mr_employee@ozon.ru	mr_intern@ozon.ru	Задание для практики	2021-01-08 12:00:03	2 days 01:41:31
2	6	mr_employee@ozon.ru	mr_intern@ozon.ru	Задание для практики	2021-01-20 19:48:54	[null]
3	7	mr_employee@ozon.ru	mr_intern2@ozon.ru	Новое задание	2021-01-21 16:41:32	00:47:32
4	3	mr_employee@ozon.ru	mr_boss@ozon.ru	Отчет по продажам 2021-01-10	2021-01-11 15:02:57	6 days 20:00:11
5	9	mr_employee@ozon.ru	mrs_boss@ozon.ru	Отчет по продажам 2021-01-20	2021-01-22 15:02:57	[null]
6	11	mr_employee@ozon.ru	mr_boss@ozon.ru	Повышение	2021-02-01 09:02:27	02:00:41
7	13	mr_employee@ozon.ru	mr_boss@ozon.ru	Повышение	2021-02-01 14:02:52	21:30:06
8	16	mr_employee@ozon.ru	mr_boss@ozon.ru	Повышение	2021-02-02 12:03:16	[null]
9	17	mr_employee@ozon.ru	mr_intern@ozon.ru	Ревью задания	2021-02-02 13:41:34	20:00:00

## ЗАДАНИЕ 1.4

Вывести id сотрудников с разницей в заработной плате в пределах 5000 рублей.

**КОММЕНТАРИЙ:** Запрос сделан исходя из его применимости на практике. Он выводит группу сотрудников исходя из номера id интересующего нас сотрудника или исходя из интересующей зарплаты.

### ЗАПРОС:

```
DROP FUNCTION IF EXISTS salary_by_id, others_employees, near_salary;
```

```
--функция находит зарплату сотрудника с указанным индексом
```

```
CREATE FUNCTION salary_by_id(id integer) RETURNS integer AS $$
```

```
    SELECT employees_salary.salary_rub FROM employees_salary WHERE employees_salary.employee_id = id ;
```

```
$$ LANGUAGE SQL;
```

```
--функция возвращает id сотрудника с ближайшей большей зарплатой среди всех сотрудников
```

```
CREATE FUNCTION near_salary(id integer) RETURNS integer AS $$
```

```
    SELECT employee_id FROM employees_salary WHERE employees_salary.salary_rub IN (
```

```
        SELECT employees_salary.salary_rub FROM employees_salary WHERE employees_salary.salary_rub >= $1) ;
```

```
$$ LANGUAGE SQL;
```

```
--функция возвращает таблицу людей у которых зарплата больше (не более чем на 5000)
```

```
--или равна зарплате сотрудника с введенным id
```

```
CREATE FUNCTION others_employees(employee_id integer) RETURNS SETOF employees_salary AS $$
```

```
    SELECT *
```

```
    FROM employees_salary
```

```
    WHERE (employees_salary.salary_rub >= salary_by_id($1))
```

```
        AND (employees_salary.salary_rub <= salary_by_id($1)+5000) ;
```

```
$$ LANGUAGE SQL;
```

```
--если нас интересует информация по пользователю, то аргумент функции others_employees - id сотрудника
```




```
--если интересует информация по определенному диапазону, то используем функцию near_salary({зарплата})
```

```
SELECT * FROM others_employees(near_salary(100000))
```

```
ORDER BY employee_id
```

### ДЕМОНСТРАЦИЯ РАБОТЫ:

таблица **employees\_salary** и запрос:

	 employee_id [PK] integer 	salary_rub integer 
1	1	100000
2	2	104000
3	3	68000
4	4	72000
5	5	101000
6	6	56000
7	7	102000
8	8	98000
9	9	25000
10	10	74000

```

1 DROP FUNCTION IF EXISTS salary_by_id, others_employees, near_salary;
2
3 --функция находит зарплату сотрудника с указанным индексом
4 CREATE FUNCTION salary_by_id(id integer) RETURNS integer AS $$
5     SELECT employees_salary.salary_rub FROM employees_salary WHERE employees_salary.employee_id = id ;
6 $$ LANGUAGE SQL;
7
8 --функция возвращает id сотрудника с ближайшей большей зарплатой среди всех сотрудников
9 CREATE FUNCTION near_salary(id integer) RETURNS integer AS $$
10     SELECT employee_id FROM employees_salary WHERE employees_salary.salary_rub IN (
11         SELECT employees_salary.salary_rub FROM employees_salary WHERE employees_salary.salary_rub >= $1) ;
12 $$ LANGUAGE SQL;
13
14 --функция возвращает таблицу людей у которых зарплата больше (не более чем на 5000)
15 --или равна зарплате сотрудника с введенным id
16 CREATE FUNCTION others_employees(employee_id integer) RETURNS SETOF employees_salary AS $$
17     SELECT *
18     FROM employees_salary
19     WHERE (employees_salary.salary_rub >= salary_by_id($1))
20         AND (employees_salary.salary_rub <= salary_by_id($1)+5000) ;
21 $$ LANGUAGE SQL;
22
23 --если нас интересует информация по пользователю, то аргумент функции others_employees - id сотрудника
24 --если интересует информация по определенному диапазону, то используем функцию near_salary({зарплата})
25 SELECT * FROM others_employees(near_salary(100000))
26 ORDER BY employee_id

```

Результирующая таблица:

	employee_id integer	salary_rub integer
1	1	100000
2	2	104000
3	5	101000
4	7	102000
5	11	105000
6	13	100000
7	18	101000
8	19	100500
9	22	101000
10	33	103000
11	44	104000