Продвинутый SQL

Задача 1

Рассчитайте LTV для схемы `tools_shop` за весь период.

```
SELECT SUM(total_amt) / COUNT(DISTINCT user_id) AS ltv
FROM tools_shop.orders
;
```

Задача 2

Добавьте группировку по месяцу создания аккаунта клиента, используя функцию `DATE_TRUNC()`.

Дополните предыдущий запрос:

- Присоедините к `orders` таблицу с данными пользователей.
- Усеките дату создания аккаунта до месяца и сгруппируйте таблицу по этому полю.

В итоговую таблицу должны войти два поля: месяц создания аккаунта и значение LTV за текущий месяц.

```
SELECT CAST(DATE_TRUNC('month', u.created_at) AS date),
       SUM(o.total_amt) / COUNT(DISTINCT u.user_id) AS ltv
FROM tools_shop.orders AS o
LEFT OUTER JOIN tools_shop.users AS u ON o.user_id = u.user_id
GROUP BY CAST(DATE_TRUNC('month', u.created_at) AS date)
;
```

Задача 3

Напишите запрос, который выведет общую конверсию из пользователя в клиента для схемы `tools_shop`. Результат округлите до одного знака после запятой.

```
SELECT ROUND(COUNT(DISTINCT o.user_id) * 100.0 / COUNT(DISTINCT u.user_id), 2)
FROM tools_shop.users AS u
LEFT JOIN tools_shop.orders o ON u.user_id = o.user_id
;
```

Задача 4

Рассчитайте метрику ARPU для схемы `tools_shop`

```
SELECT SUM(o.total_amt) / COUNT(DISTINCT u.user_id) AS arpu
FROM tools_shop.users AS u
LEFT JOIN tools_shop.orders o ON o.user_id = u.user_id
;
```

Задача 5

Расчёт средних значений. ARPU и ARPPU

```
SELECT CAST(DATE_TRUNC('day', o.event_dt) AS date),
SUM(o.revenue) / COUNT(DISTINCT p.user_id) AS arppu
FROM online_store.orders o
JOIN online_store.profiles p ON o.user_id = p.user_id
GROUP BY CAST(DATE_TRUNC('day', o.event_dt) AS date)
;
```

Задача 6

Используя функцию `DATE_TRUNC()` , рассчитайте ARPPU в разрезе года оформления заказа. Поле с датой приведите к типу данных `date`, а значение ARPPU округлите до двух знаков после запятой.

```
SELECT CAST(DATE_TRUNC('year', o.created_at) AS date) AS date,
       ROUND((SUM(o.total_amt) / COUNT(DISTINCT (u.user_id))), 2) AS arppu
FROM tools_shop.orders AS o
JOIN tools_shop.users AS u ON o.user_id = u.user_id
GROUP BY CAST(DATE_TRUNC('year', o.created_at) AS date)
ORDER BY CAST(DATE_TRUNC('year', o.created_at) AS date) -- Не обязательно для
                                                        --правильного решения задачи
;
```

Задача 7

Посчитать конверсию пользователей из первого заказа во второй

```
SELECT COUNT(so.user_id) * 100.0 / COUNT(p.user_id) AS cr
FROM online_store.profiles p
LEFT JOIN
-- пользователи, которые оформили два заказа и более
  (SELECT user_id,
          COUNT(*) AS orders_cnt
   FROM online_store.orders
   GROUP BY user_id
   HAVING COUNT(*) >= 2) so ON p.user_id = so.user_id
WHERE payer IS TRUE; -- пользователи, которые совершали заказ
```

Задача 8

Рассчитать среднее от средней длительности сессий на пользователя.

```
SELECT AVG(avg_duration)
FROM
  (SELECT user_id,
          AVG(session_duration) AS avg_duration
   FROM online_store.sessions
   GROUP BY user_id) adur;
```

Задача 9

ROI. Динамика возврата инвестиций

```
SELECT dc.dt,
       revenue * 100 / costs AS ROI
FROM
  (SELECT dt,
          SUM(costs) AS costs
   FROM online_store.costs
   GROUP BY dt) dc
JOIN
  (SELECT event_dt AS dt,
          SUM(revenue) AS revenue
   FROM online_store.orders
   GROUP BY event_dt) dr ON dc.dt = dr.dt;
```

Задача 10

Рассчитайте ROI в динамике по месяцам для схемы `tools_shop`

```
SELECT dc.dt,
       total_amt * 100 / costs AS ROI
FROM
```

```
(SELECT DATE_TRUNC('month', created_at)::date AS dt,
       SUM(costs) AS costs
FROM tools_shop.costs
GROUP BY dt) AS dc
JOIN
(SELECT DATE_TRUNC('month', created_at)::date AS dt,
       SUM(total_amt) AS total_amt
FROM tools_shop.orders
GROUP BY dt) AS dr ON dc.dt = dr.dt
;
```

Задача 11

Напишите запрос, который выведет все поля таблицы `tools_shop.orders` и отдельным полем суммарную стоимость заказов за каждый месяц.

```
SELECT * ,
SUM(total_amt) OVER (PARTITION BY DATE_TRUNC('month', created_at))
FROM tools_shop.orders
;
```

Задача 12

Проранжируйте записи в таблице `tools_shop.users` по дате регистрации — от меньшей к большей. Напишите запрос, который выведет идентификатор пользователя с рангом 2021.

```
WITH i AS
(SELECT *,
ROW_NUMBER() OVER (ORDER BY created_at) AS rn
FROM tools_shop.users)

SELECT user_id
FROM i
WHERE rn = 2021
;
```

Задача 13

Проранжируйте записи в таблице `tools_shop.orders` по дате оплаты заказа — от большей к меньшей. Напишите запрос, который выведет стоимость заказа с рангом 50.

```
WITH i AS
(SELECT *,
ROW_NUMBER() OVER (ORDER BY paid_at DESC) AS rn
FROM tools_shop.orders)

SELECT total_amt
FROM i
WHERE rn = 50
;
```

Задача 14

Проранжируйте записи в таблице `tools_shop.users` в зависимости от значения в поле `created_at` — от большего к меньшему. Записи с одинаковым значением `created_at` должны получить один ранг. Ранги должны быть указаны последовательно.

```
SELECT *,
        DENSE_RANK() OVER (ORDER BY created_at DESC)
FROM tools_shop.users
```

Задача 15

Разбейте заказы из таблицы `tools_shop.orders` на десять групп, отсортировав их по значению суммы заказа по возрастанию.

Выведите поля:

- идентификатор заказа;
- сумма заказа;
- ранг группы.

```
SELECT order_id,
       total_amt,
       NTILE(10) OVER (ORDER BY total_amt)
FROM tools_shop.orders
;
```

Задача 16

Разбейте пользователей в таблице `tools_shop.users` на пять групп так, чтобы в первую группу попали пользователи, которые недавно зарегистрировались.

Выгрузите поля:

- идентификатор пользователя;
- дата регистрации;
- ранг группы.

```
SELECT user_id,
       created_at,
       NTILE(5) OVER (ORDER BY created_at DESC)
FROM tools_shop.users
;
```

Задача 17

Выведите все поля таблицы `tools_shop.orders` и проранжируйте заказы для каждого клиента в зависимости от даты оплаты заказа — от меньшей к большей.

```
SELECT *,
       ROW_NUMBER() OVER (PARTITION BY user_id ORDER BY paid_at)
FROM tools_shop.orders
;
```

Задача 18

Рассчитайте общее количество заказов в таблице `tools_shop.orders` по дням. Выведите все поля таблицы и новое поле с количеством заказов.

```
SELECT *,
      COUNT(*) OVER (PARTITION BY created_at::date)
FROM tools_shop.orders
;
```

Задача 19

Рассчитайте общую выручку в таблице `tools_shop.orders` по месяцам. Выведите все поля таблицы и новое поле с суммой выручки.

```
SELECT *,
      ROUND(SUM(total_amt) OVER (PARTITION BY DATE_TRUNC('month', created_at)::date ), 2)
```

```
FROM tools_shop.orders
;
```

Задача 20

Рассчитайте общую выручку в таблице `tools_shop.orders`
по месяцам. Выведите все поля таблицы и новое поле с суммой выручки.

Напишите запрос к таблице `tools_shop.orders`, который выведет:

- идентификатор пользователя `user_id`;
- дату и время заказа `created_at`;
- сумму заказа `total_amt`;
- сумму заказа с накоплением для каждого пользователя, отсортированную по возрастанию даты заказа.

```
SELECT user_id,
       created_at,
       total_amt,
       SUM(total_amt) OVER (PARTITION BY user_id ORDER BY created_at)
FROM tools_shop.orders
;
```

Задача 21

Напишите запрос к таблице `tools_shop.orders`, который выведет:

- месяц заказа, приведённый к типу `date`;
- сумму заказа `total_amt`;
- сумму заказа с накоплением, отсортированную по возрастанию месяца оформления заказа.

```
SELECT DATE_TRUNC('month', created_at)::date,
       total_amt,
       SUM(total_amt) OVER (ORDER BY DATE_TRUNC('month', created_at)::date)
FROM tools_shop.orders
;
```

Задача 22

Из таблицы `tools_shop.orders` выведите поля `order_id`, `user_id`, `paid_at` и к ним добавьте поле `paid_at` с датой предыдущего заказа для каждого пользователя. Если предыдущего заказа нет, выведите дату 1 января 1980 года.

```
SELECT order_id,
       user_id,
       paid_at,
       LAG(paid_at, 1, '1980-01-01') OVER (PARTITION BY user_id ORDER BY paid_at)
FROM tools_shop.orders
;
```

Задача 23

Выведите поля `event_id`, `event_time`, `user_id` из таблицы `tools_shop.events`
и к ним добавьте поле с датой и временем следующего события для каждого пользователя. Если события нет, оставьте значение `NULL`

```
SELECT event_id,
       event_time,
       user_id,
       LEAD(event_time, 1, NULL) OVER (PARTITION BY user_id ORDER BY event_time)
FROM tools_shop.events
```

Задача 24

Исправьте предыдущий запрос: замените дату следующего события на интервал между текущим и следующим событием. Значение интервала должно быть положительным. Если события нет, оставьте значение `NULL`.

```
SELECT event_id,
       event_time,
       user_id,
       LEAD(event_time, 1, NULL) OVER (PARTITION BY user_id ORDER BY event_time) - event_time
FROM tools_shop.events
```

Задача 25

Напишите запрос, который проранжирует расходы на привлечение пользователей за каждый день по убыванию.

Выгрузите три поля:

- дата, которую нужно привести к типу `date`;
- расходы на привлечение;
- ранг строки.

```
SELECT created_at::date,
       costs,
       ROW_NUMBER() OVER (ORDER BY costs DESC)
FROM tools_shop.costs
;
```

Задача 26

Измените предыдущий запрос: записям с одинаковыми значениями расходов назначьте одинаковый ранг. Ранги не должны прерываться.

```
SELECT created_at::date,
       costs,
       DENSE_RANK() OVER (ORDER BY costs DESC)
FROM tools_shop.costs
;
```

Задача 27

Выведите список уникальных `user_id` пользователей, которые совершили три заказа и более.

```
WITH i AS
(SELECT
       user_id,
       COUNT(user_id) AS r
FROM tools_shop.orders
GROUP BY user_id)

SELECT DISTINCT user_id
FROM i
WHERE r >= 3
```

Задача 28

Выведите количество заказов, в которых было четыре товара и более

```
WITH i AS
(SELECT order_id,
        COUNT(item_id) AS r
FROM tools_shop.order_x_item
GROUP BY order_id)
SELECT COUNT(DISTINCT(order_id))
```

```
FROM i
WHERE r >= 4
```

Задача 29

Рассчитайте количество зарегистрированных пользователей по месяцам с накоплением.

Выгрузите два поля:

- месяц заказа, приведённый к типу `date`;
- общее количество зарегистрированных пользователей на текущий месяц.

```
WITH i AS(
SELECT DATE_TRUNC('month', created_at)::date AS date,
       COUNT(DISTINCT(user_id))
FROM tools_shop.users
GROUP BY date)

SELECT date,
       SUM(count) OVER (ORDER BY date)
FROM i
;
```

Задача 30

Рассчитайте сумму трат на привлечение пользователей с накоплением по месяцам с 2017 по 2018 год включительно.

Выгрузите два поля:

- месяц, приведённый к типу `date`;
- сумма трат на текущий месяц с накоплением.

```
WITH i AS(
SELECT DATE_TRUNC('month', created_at)::date AS date,
       SUM(costs) OVER (ORDER BY DATE_TRUNC('month', created_at)::date) AS costs
FROM tools_shop.costs
WHERE EXTRACT(YEAR FROM created_at) IN (2017, 2018))

SELECT DISTINCT date,
       costs
FROM i
ORDER BY date
;
```

Задача 31

Посчитайте события с названием `view_item` по месяцам с накоплением. Рассчитайте количество событий только для тех пользователей, которые совершили хотя бы одну покупку.

Выгрузите поля:

- месяц события, приведённый к типу `date`;
- количество событий за текущий месяц;
- количество событий за текущий месяц с накоплением

```
WITH i AS
(SELECT DATE_TRUNC('month', event_time)::date AS date,
       COUNT(event_id) AS count
FROM tools_shop.events
WHERE event_name = 'view_item'
AND user_id IN (
                SELECT user_id
                FROM tools_shop.orders
                WHERE paid_at IS NOT NULL)
                GROUP BY DATE_TRUNC('month', event_time)::date)
```

```
SELECT date,
       count,
       SUM(count) OVER (ORDER BY date)
FROM i
;
```

Задача 32

Используя конструкцию `WINDOW`, рассчитайте суммарную стоимость и количество заказов с накоплением от месяца к месяцу.

Выгрузите поля:

- идентификатор заказа;
- месяц оформления заказа, приведённый к типу `date`;
- сумма заказа;
- количество заказов с накоплением;
- суммарная стоимость заказов с накоплением.

```
SELECT order_id,
       DATE_TRUNC('month', created_at)::date AS date,
       total_amt,
       COUNT(order_id) OVER (ORDER BY DATE_TRUNC('month', created_at)::date),
       SUM(total_amt) OVER (ORDER BY DATE_TRUNC('month', created_at)::date)
FROM tools_shop.orders
```

Задача 33

Напишите запрос, который выведет сумму трат на привлечение пользователей по месяцам, а также разницу в тратах между текущим и предыдущим месяцами. Разница должна показывать, на сколько траты текущего месяца отличаются от предыдущего. В случае, если данных по предыдущему месяцу нет, укажите ноль.

Выгрузите поля:

- месяц, приведённый к типу `date`;
- траты на привлечение пользователей в текущем месяце;
- разница в тратах между текущим и предыдущим месяцами.

```
WITH i AS(
SELECT DATE_TRUNC('month', created_at)::date AS date,
       SUM(costs) AS costs
FROM tools_shop.costs
GROUP BY DATE_TRUNC('month', created_at)::date
ORDER BY DATE_TRUNC('month', created_at)::date)

SELECT *,
costs - LAG(costs, 1, costs) OVER (ORDER BY date) AS delta_previous_costs
FROM i
;
```

Задача 34

Напишите запрос, который выведет сумму выручки по годам и разницу выручки между текущим и следующим годом. Разница должна показывать, на сколько траты следующего года отличаются от текущего. В случае, если данных по следующему году нет, укажите ноль.

Выгрузите поля:

- год, приведённый к типу `date`;
- выручка за текущий год;

- разница в выручке между текущим и следующим годом.

```
WITH i AS(
SELECT DATE_TRUNC('year', created_at)::date AS date,
       SUM(total_amt) AS yr_revenue
FROM tools_shop.orders
GROUP BY DATE_TRUNC('year', created_at)::date
ORDER BY DATE_TRUNC('year', created_at)::date)

SELECT *,
(LEAD(yr_revenue, 1, yr_revenue) OVER (ORDER BY date) - yr_revenue) AS delta
FROM i
;
```

Задача 35

Когортный анализ Retention Rate

```
WITH profile AS
  (SELECT user_id,
          dt,
          COUNT(*) OVER (PARTITION BY dt) AS cohort_users_cnt
   FROM online_store.profiles
   WHERE channel = 'Organic'),
sessions AS
(SELECT user_id,
        session_start::date AS session_date
FROM online_store.sessions
GROUP BY 1,
         2)
SELECT p.dt cohort_dt,
       session_date,
       COUNT(p.user_id) AS users_cnt,
       cohort_users_cnt,
       ROUND(COUNT(p.user_id) * 100.0 / cohort_users_cnt, 2) AS retention_rate
FROM profile p
JOIN sessions s ON p.user_id = s.user_id
GROUP BY 1,
         2,
         4;
```

Задача 36

Добавьте к предыдущему запросу месяц первого события, используя функцию `DATE_TRUNC()` . Приведите поле к типу `date` . Эта дата будет началом когорты.

```
SELECT o.user_id,
       MIN(DATE_TRUNC('month', event_time)::date) AS date_1st
FROM tools_shop.orders AS o
LEFT OUTER JOIN tools_shop.users AS u ON o.user_id = u.user_id
LEFT OUTER JOIN tools_shop.events AS e ON o.user_id = e.user_id
GROUP BY o.user_id
;
```

Задача 37

Определить, было ли у пользователей хотя бы одно событие в каждом месяце. Напишите запрос, который выведет все уникальные комбинации `user_id` и значения месяца события для выбранных пользователей. Не забудьте добавить запрос из предыдущего задания во временную таблицу.

```
WITH i AS(
SELECT o.user_id,
       MIN(DATE_TRUNC('month', event_time)::date) AS date_1st
FROM tools_shop.orders AS o
LEFT OUTER JOIN tools_shop.users AS u ON o.user_id = u.user_id
LEFT OUTER JOIN tools_shop.events AS e ON o.user_id = e.user_id
GROUP BY o.user_id)
```

```
SELECT DISTINCT user_id,
       DATE_TRUNC('month', event_time)::date AS dates
FROM tools_shop.events
WHERE user_id IN (SELECT user_id
                  FROM i)
GROUP BY user_id,
         dates
ORDER BY user_id
```

Задача 38

Найти общее количество пользователей для каждой когорты. Используйте временную таблицу из прошлых заданий. Напишите запрос, который выведет дату старта когорты и количество пользователей в ней. Оставьте в таблице только уникальные значения.

```
WITH i AS(
SELECT o.user_id,
       MIN(DATE_TRUNC('month', event_time)::date) AS date_1st
FROM tools_shop.orders AS o
LEFT OUTER JOIN tools_shop.users AS u ON o.user_id = u.user_id
LEFT OUTER JOIN tools_shop.events AS e ON o.user_id = e.user_id
GROUP BY o.user_id)

SELECT date_1st,
       COUNT(user_id)

FROM i
GROUP BY date_1st
ORDER BY date_1st
;
```

Задача 39

Остаётся их правильно объединить и рассчитать Retention Rate.

Выведите в запросе несколько полей:

1. месяц старта когорты,
2. месяц события,
3. общее количество пользователей когорты,
4. количество пользователей в этот месяц,
5. Retention Rate.

Значение Retention Rate округлите до двух знаков после запятой.

```
WITH profile AS
  (SELECT u.user_id,
          DATE_TRUNC('month', MIN(event_time))::date AS dt
   FROM tools_shop.users u
   JOIN tools_shop.orders o ON u.user_id = o.user_id
   JOIN tools_shop.events e ON u.user_id = e.user_id
   GROUP BY 1),

sessions AS
  (SELECT p.user_id,
          DATE_TRUNC('month', event_time)::date AS session_dt
   FROM tools_shop.events e
   JOIN profile p ON p.user_id = e.user_id
   GROUP BY 1,
            2),

cohort_users_cnt AS
  (SELECT dt,
          COUNT(user_id) AS cohort_users_cnt
   FROM profile
   GROUP BY 1)
```

```
SELECT p.dt,
       s.session_dt,
       COUNT(p.user_id) AS users_cnt,
       cohort_users_cnt,
       ROUND(COUNT(p.user_id) * 100.0 / cohort_users_cnt, 2) AS retention_rate
FROM profile AS p
JOIN sessions AS s ON p.user_id = s.user_id
JOIN cohort_users_cnt AS c ON s.session_dt = c.dt
GROUP BY 1,
         2,
         4
;
```

Задача 40

Когортный анализ LTV: практика

Сначала определите профиль пользователя. Напишите запрос, который выведет пользователей, зарегистрировавшихся в 2019 году и совершивших хотя бы одну покупку.

Выведите два поля:

- идентификатор пользователя;
- дата регистрации пользователя.

```
WITH profile AS(
SELECT u.user_id,
       DATE_TRUNC('month', u.created_at):: date AS date,
       EXTRACT(MONTH FROM AGE(o.created_at, u.created_at)) AS lifetime,
       SUM(total_amt) OVER (PARTITION BY u.user_id ORDER BY (o.created_at)) AS ltv

FROM tools_shop.users AS u
INNER JOIN tools_shop.orders AS o ON u.user_id = o.user_id
WHERE EXTRACT (YEAR FROM u.created_at) = '2019')

SELECT date,
       lifetime,
       ROUND(AVG(ltv),2) AS avg_ltv
FROM profile
WHERE lifetime <= 5
GROUP BY 1,
         2
;
```

Задача 41

Когортный анализ Churn Rate: практика

```
WITH
profile AS(
SELECT u.user_id,
       MIN(DATE_TRUNC('month', e.event_time):: date) AS min_date
FROM tools_shop.users AS u
JOIN tools_shop.orders AS o ON u.user_id = o.user_id
JOIN tools_shop.events AS e ON u.user_id = e.user_id
GROUP BY u.user_id),

cohort AS(
SELECT min_date,
       DATE_TRUNC('month', e.event_time):: date AS date,
       COUNT(DISTINCT(p.user_id)) AS users_cnt
FROM profile AS p
JOIN tools_shop.events AS e ON p.user_id = e.user_id
GROUP BY 1,
         2)
SELECT *,
       LAG(users_cnt) OVER (PARTITION BY min_date ORDER BY date) AS previous_users_cnt,
       ROUND((1 - (users_cnt::numeric/ LAG(users_cnt) OVER (PARTITION BY min_date ORDER BY date)))*100, 2) AS churn_rate
FROM cohort
;
```

Задача 42

Определение рамки. Режим ROWS

Напишите запрос, который выведет поля `order_id`, `created_at` и минимальное значение поля `created_at` среди текущей и десяти следующих записей для таблицы `tools_shop.orders`
.

```
SELECT order_id,
       created_at,
       MIN(created_at) OVER (ROWS BETWEEN CURRENT ROW AND 10 FOLLOWING) AS min_created_at
FROM tools_shop.orders
;
```

Задача 43

Напишите запрос, который выведет поля `order_id` , `total_amt` и среднее значение `total_amt` на основе трёх предыдущих записей вместе с текущей.

```
SELECT order_id,
       total_amt,
       AVG(total_amt) OVER (ROWS BETWEEN 3 PRECEDING AND CURRENT ROW)
FROM tools_shop.orders
;
```

Задача 44

Напишите запрос, который выведет поля `order_id`
, `user_id`
, `items_cnt`
и максимальное количество товаров в заказе среди предыдущего, текущего и следующего заказов для каждого пользователя.

```
SELECT order_id,
user_id,
items_cnt,
MAX(items_cnt) OVER (PARTITION BY user_id ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING)
FROM tools_shop.orders
;
```

Задача 45

Для каждого заказа в таблице `tools_shop.orders` посчитайте товары, которые оплатили в последние 24 часа (англ. 24 hours) относительно даты оплаты текущего заказа.

Выведите поля:

- идентификатор заказа;
- количество товаров в заказе;
- время оплаты;
- количество товаров, оплаченных в последние 24 часа.

```
SELECT order_id,
      items_cnt,
      paid_at,
      COUNT(items_cnt) OVER (ORDER BY paid_at RANGE BETWEEN '24 hours' PRECEDING AND CURRENT ROW)
FROM tools_shop.orders
;
```

Задача 46

Для каждого пользователя в таблице `tools_shop.users` посчитайте, сколько регистраций оформили за промежуток между последними 30 днями от текущего дня и следующими 30 днями.

Выведите поля:

- идентификатор пользователя;
- дата регистрации;
- количество регистраций за указанный период.

```
SELECT user_id,
       created_at,
       COUNT(user_id) OVER (ORDER BY created_at RANGE BETWEEN '30 day' PRECEDING AND '30 day' FOLLOWING) AS reg_cnt
FROM tools_shop.users
;
```

Задача 47

Оконные функции выбора. FIRST_VALUE(), LAST_VALUE(), NTH_VALUE()

Напишите запрос, который выведет все поля таблицы `tools_shop.orders` и отдельным полем сумму первого заказа для каждого пользователя.

```
SELECT *,
      FIRST_VALUE(total_amt) OVER (PARTITION BY user_id ORDER BY created_at)
FROM tools_shop.orders
;
```

Задача 48

Напишите запрос, который выведет все поля таблицы `tools_shop.costs` и отдельным полем сумму трат в последний день каждого месяца.

```
SELECT *,
LAST_VALUE(costs) OVER (PARTITION BY DATE_TRUNC('month',created_at )::date)
FROM tools_shop.costs
;
```

Задача 49

Магазин решил устроить розыгрыш и вручить подарок пользователю, который оформил десятитысячный по счёту заказ. Выведите идентификатор такого пользователя.

```
SELECT DISTINCT NTH_VALUE(user_id, 10000) OVER (ORDER BY created_at ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS user_id
FROM tools_shop.orders
;
```

Задача 50

Исключение записей. Оператор EXCLUDE

Для каждого заказа из таблицы `tools_shop.orders`, оплаченного в 2020 году, рассчитайте количество заказанных товаров с накоплением без учёта текущего заказа. Записи отсортируйте по дате оплаты заказа.

Выведите поля:

- идентификатор заказа;
- сумма заказа;
- дата оплаты заказа;
- количество заказанных товаров с накоплением.

```
SELECT order_id,
total_amt,
paid_at,
SUM(total_amt) OVER (ORDER BY paid_at ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW EXCLUDE CURRENT ROW)
FROM tools_shop.orders
;
```

Задача 51

Для каждой записи в таблице `tools_shop.costs` рассчитайте средние траты на привлечение пользователей за последние 30 дней.

```
SELECT OVER (ORDER BY created_at RANGE BETWEEN '30 days' PRECEDING AND CURRENT ROW EXCLUDE CURRENT ROW)
FROM tools_shop.costs
;
```

Задача 52

Для каждой записи в таблице `tools_shop.orders` рассчитайте суммарную стоимость заказов за предыдущий и следующий дни.

Выгрузите поля:

- идентификатор заказа;
- сумма заказа;
- дата оформления заказа до оплаты;
- суммарная стоимость заказов за предыдущий и следующий дни.

```
SELECT OVER (ORDER BY created_at::date RANGE BETWEEN '1 day' PRECEDING AND '1 day' FOLLOWING EXCLUDE GROUP)
FROM tools_shop.orders
;
```

Задача 53

Расчёт скользящих значений

Напишите запрос, который рассчитает среднюю стоимость привлечения пользователей за последние 30 дней для таблицы `tools_shop.costs`.

Выгрузите поля:

- дата, приведённая к типу `date`;
- стоимость привлечения пользователей в текущий день;
- средняя стоимость привлечения пользователей за последние 30 дней, не включая текущий.

```
SELECT created_at:: date,
       costs,
        AVG(costs) OVER (ORDER BY created_at RANGE BETWEEN '30 days' PRECEDING AND CURRENT ROW EXCLUDE CURRENT ROW)

FROM tools_shop.costs
;
```

Задача 54

Напишите запрос, который рассчитает скользящее среднее время между событиями пользователей за последние семь дней. Для расчёта среднего оставьте только те события, разница между которыми составляет не больше 30 дней.

Выгрузите поля:

- дата события, приведённая к типу `date`;

- среднее время между событиями пользователей за последние 30 дней.

Выведите первые 5000 записей.

```
WITH event_lag AS(
SELECT *
FROM (SELECT user_id,
      event_time:: date,
      LAG(event_time) OVER (PARTITION BY user_id ORDER BY event_time)::date AS previous_event_time,
      event_time - LAG(event_time) OVER (PARTITION BY user_id ORDER BY event_time ) AS event_lag
FROM tools_shop.events) AS i
WHERE event_lag <= '30 days')

SELECT event_time,
AVG(event_lag) OVER (ORDER BY event_time RANGE BETWEEN '7 days' PRECEDING AND CURRENT ROW EXCLUDE GROUP)
FROM event_lag
LIMIT 5000
;
```

Задача 55

Напишите запрос, который для каждой записи таблицы `tools_shop.orders` рассчитает средний чек заказов, оформленных за последние 90 дней.

Выгрузите поля:

- идентификатор заказа;
- сумма заказа;
- средняя сумма заказа за последние 90 дней, не включая текущий.

```
SELECT order_id,
       total_amt,
       AVG(total_amt) OVER (ORDER BY created_at RANGE BETWEEN '90 days' PRECEDING AND CURRENT ROW EXCLUDE GROUP)
FROM tools_shop.orders
;
```