5.

Рассчитайте количество зарегистрированных пользователей по месяцам с накоплением.

Выгрузите два поля:

* месяц регистрации, приведённый к типу date;
* общее количество зарегистрированных пользователей на текущий месяц.

WITH sql1 AS (

SELECT CAST(DATE\_TRUNC('month',created\_at) as date) as date\_crt,

COUNT(user\_id) OVER (ORDER BY DATE\_TRUNC('month',CAST(created\_at AS date))) as sum\_us

FROM tools\_shop.users

)

SELECT distinct(date\_crt),

sum\_us

FROM sql1

ORDER BY sum\_us

6.

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

Выгрузите два поля:

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

WITH sql1 AS (

SELECT CAST(DATE\_TRUNC('month',created\_at) as date) as date\_crt,

SUM(costs) OVER (ORDER BY DATE\_TRUNC('month',CAST(created\_at AS date))) as sum\_us

FROM tools\_shop.costs

WHERE CAST(DATE\_TRUNC('month',created\_at) as date) BETWEEN '2017-01-01-' AND '2018-12-31'

)

SELECT distinct(date\_crt),

sum\_us

FROM sql1

ORDER BY sum\_us

7.

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 sql1 AS (

SELECT CAST(DATE\_TRUNC('month',event\_time) as date) as month\_event,

COUNT(e.user\_id) as count\_ev

FROM tools\_shop.events as e

INNER JOIN

(SELECT DISTINCT (user\_id) as user\_id

FROM tools\_shop.orders) as o

ON e.user\_id = o.user\_id

WHERE e.event\_name = 'view\_item'

GROUP BY month\_event

)

SELECT month\_event,

count\_ev,

SUM(count\_ev) OVER (ORDER BY month\_event)

FROM sql1

**Особенности оконных функций**

В этом уроке расскажем, как упростить запрос с несколькими оконными функциями, а также о том, с чем оконные функции сочетать нельзя.

**Окно как переменная**

Определение окна в выражении OVER может встречаться в запросе несколько раз. Например, в запросе ниже используются разные функции ранжирования записей:

Скопировать кодSQL

SELECT \*,

ROW\_NUMBER() OVER (ORDER BY revenue),

RANK() OVER (ORDER BY revenue),

DENSE\_RANK() OVER (ORDER BY revenue)

FROM online\_store.orders;

| **user\_id** | **event\_dt** | **revenue** | **row\_number** | **rank** | **dense\_rank** |
| --- | --- | --- | --- | --- | --- |
| 117600800430 | 2020-06-23 | 1 | 1 | 1 | 1 |
| 13263143928 | 2020-06-18 | 1 | 2 | 1 | 1 |
| 318506600467 | 2020-06-21 | 1 | 3 | 1 | 1 |
| 461670247155 | 2020-06-17 | 1 | 4 | 1 | 1 |
| 896743136756 | 2020-06-19 | 1 | 5 | 1 | 1 |

Вместо того чтобы повторять определение окна трижды, его можно вынести в переменную. Для этого используют конструкцию WINDOW.

Скопировать кодSQL

SELECT \*,

ROW\_NUMBER() OVER my\_window,

RANK() OVER my\_window,

DENSE\_RANK() OVER my\_window

FROM online\_store.orders

WINDOW my\_window AS (ORDER BY revenue);

Теперь, чтобы исправить определение окна, понадобится поменять всего одну строчку кода вместо трёх.

В основном запросе конструкцию WINDOW указывают после оператора WHERE и до оператора ORDER BY. Синтаксис у конструкции такой: WINDOW <название окна> AS <определение окна>. В одном запросе можно записать сразу несколько определений окна. В таком случае их нужно указать через запятую, как в случае временных таблиц.

Скопировать кодSQL

WHERE ...

WINDOW my\_window AS (ORDER BY revenue),

one\_more\_window AS (PARTITION BY user\_id)

ORDER BY ...

8.

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 creat\_at,

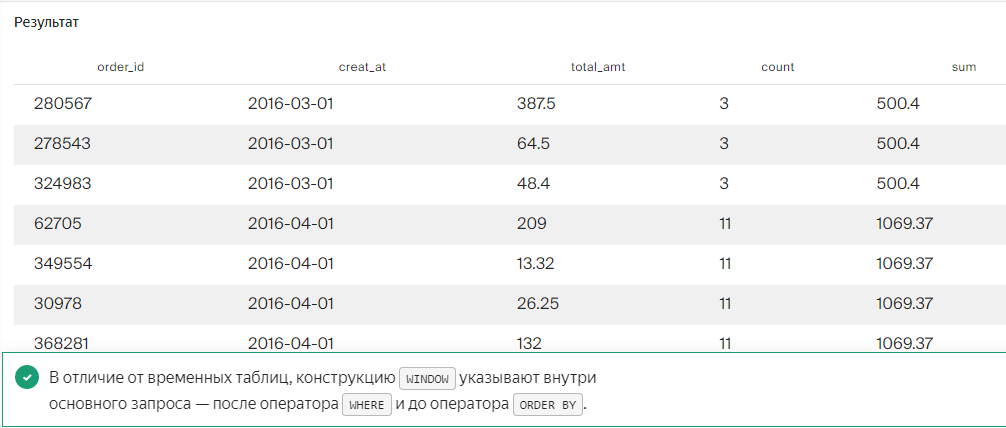
total\_amt,

COUNT(\*) OVER my\_window,

SUM(total\_amt) OVER my\_window

FROM tools\_shop.orders

WINDOW my\_window AS (ORDER BY DATE\_TRUNC('month',created\_at)::date)



9.

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 sql1 AS(

SELECT DATE\_TRUNC('month',created\_at) ::date as cr\_at,

SUM (costs) as month\_cost

FROM tools\_shop.costs

GROUP BY cr\_at

ORDER BY cr\_at

)

SELECT

cr\_at,

month\_cost ,

month\_cost - LAG(month\_cost,1,month\_cost) OVER (ORDER BY cr\_at) as diff\_lag\_costs

FROM sql1

