



8. Оконные функции



Повторим пройденное

- ▶ Как сделать пересечение/объединение/разность множеств?
- ▶ На что влияет ALL?
- ▶ Чем grouping sets отличается от group by ?
- ▶ Какие комбинации выдаст rollup(c1, c2, c3)? А cube(c1, c2, c3)?

ЧТО БУДЕМ ДЕЛАТЬ СЕГОДНЯ

▶ Основы оконных функций

- `over()`
 - `partition by`
 - `order by`
 - `rows/range between`

▶ Типы оконных функций

- Агрегатные: `sum`, `count`, `min`, `max`
- Ранжирующие: `row_number`, `dense_rank`, `rank`



**Академия
Аналитиков
Авито**

Оконные функции: ОСНОВЫ



Как работают оконные функции?

В чем отличие от агрегатных функций?



Обсуждение

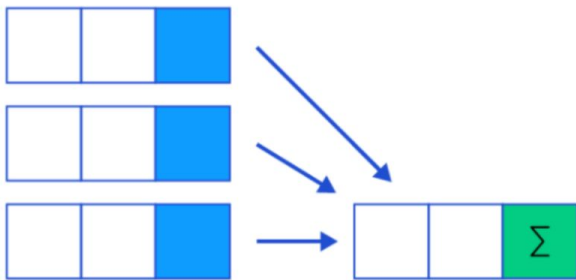


5 минут

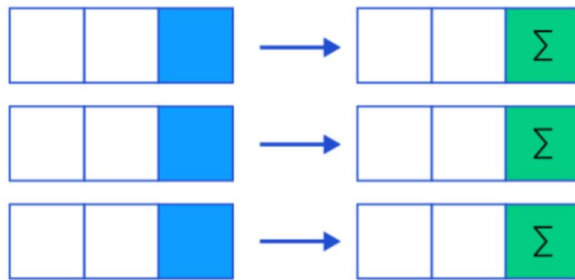
Как работают оконные функции?

Окно – набор строк, в рамках которого происходит вычисление. Оконная функция позволяет разбивать весь набор данных на такие окна. OVER() описывает набор строк, которые будет обрабатывать функция и порядок этой обработки.

Aggregate Functions (SUM, AVG, etc.)



Window Functions (Over, Partition, Order, etc.)



В чем отличие от агрегатных функций?

Оконные функции не приводят к группированию строк в одну строку вывода, строки сохраняют свои отдельные идентификаторы, а агрегированное значение добавляется к каждой строке.

```
select
  user_id,
  amount,
  sum(amount) over (
    partition by user_id
  )
from payments
```

user_id	amount
1	10
1	10
1	10
2	20
2	20
3	30

user_id	amount	sum
1	10	30
1	10	30
1	10	30
2	20	40
2	20	40
3	30	30

```
se
  user_id,
  sum(amount)
from payments
group by user_id
```

user_id	amount
1	10
1	10
1	10
2	20
2	20
3	30

user_id	sum
1	30
2	40
3	30

Зачем это надо?

- Считать накопленную сумму в одну строку без джоина
- Считать накопленную сумму за определенный интервал дат или строк в 1 строку без джоина
- Делать ранжирование строк
- Считать топ N строк по метрике



**Академия
Аналитиков
Авито**

Синтаксис оконных функций. Агрегатные оконные функции.



Вопросы по прериду

- Какое ключевое слово оконной функции?
- Как обозначить весь набор данных для работы оконной функции?
- Что делает **PARTITION BY**?
- Что делает **ORDER BY**?
- Что означает отсутствие конструкции **ORDER BY**?



Обсуждение



3 минуты

Таблицы

d8_training_log

training_date	player_id	training_result
2020-01-11	3	0
2020-01-24	3	1
2020-01-30	4	1

d8_player

player_id	player_name	registration_date
3	Федоров Федор Федорович	2020-01-11
4	Григорьев Григорий Григорьевич	2020-01-25
5	Петров Петр Петрович	2020-01-26
1	Иванов Иван Иванович	2020-01-30
2	Васильев Василий Васильевич	2020-01-11

d8_tournament_log

match_date	player_id	opponent_id	match_result	score
2020-01-13	2	3	1	95
2020-01-13	3	2	-1	15
2020-01-25	2	4	1	55

d8_player_city

player_id	city	actual_date
1	Новосибирск	2020-01-30
1	Владивосток	2020-01-31
3	Новосибирск	2020-01-15

Задача 0

Условие:

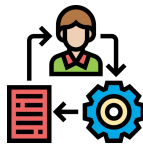
Дана таблица

```
-- d8_training_log (training_date, player_id, training_result);
```

Вопрос:

Вывести топ 3 дня, когда тренировалось больше всего игроков.

🕒 training_date 📈📉	123 🕒 players 📈📉
2020-01-30	4
2020-01-24	2
2020-01-11	2



Практика



5 минут

Задача 1

Условие:

Дана таблица

```
-- d8_training_log (training_date, player_id, training_result);
```

Вопрос:

Для каждого игрока выведите максимальный результат (training_result) за тренировку и минимальную дату, когда этот игрок получил свой максимальный результат без использования join.

123 player_id	123 max_result	min_trainig_date_with
1	1	2020-01-30
2	5	2020-01-11
3	8	2020-01-30
4	3	2020-01-26
5	10	2020-01-27



Практика



8 минут

Задача 2

Условие:

Дана таблица

- d8_tournament_log (match_date, player_id, opponent_id, match_result, score)

Вопрос:

Выведите последнюю даты игры и суммарное кол-во заработанных очков (score) для каждого игрока.

Отсортируйте по айди игрока.



Практика

123 player_id	123 score	🕒 max_date
1	266	2020-02-02
2	249	2020-02-01
3	249	2020-01-31
4	230	2020-02-02
5	138	2020-01-31



8 минуты

Задача 3

Условие:

Дана таблица

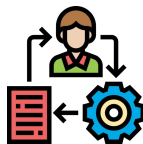
- d8_tournament_log (match_date, player_id, opponent_id, match_result, score);

Вопрос:

Для каждого игрока подсчитайте накопленный score ко дню каждого матча.

Обратите внимание что в один день могло быть две игры.

	🕒 match_date 🔼🔽	123 player_id 🔼🔽	123 sum 🔼🔽
1	2020-01-31	1	164
2	2020-02-01	1	194
3	2020-02-02	1	266
4	2020-01-13	2	95
5	2020-01-25	2	150
6	2020-01-27	2	234
7	2020-02-01	2	249



Практика



8 минут

Задача 4

Условие:

-- d8_player (player_id, player_name, registration_date);

Вопрос:

Каждому игроку сопоставьте число дней, прошедших от регистрации первого игрока до его регистрации без использования join.

abc player_name	123 diff
Васильев Василий Васильевич	0
Григорьев Григорий Григорьевич	14
Иванов Иван Иванович	19
Петров Петр Петрович	15
Федоров Федор Федорович	0



Работаем вместе



8 мин

Задача 5

Условие:

-- d8_training_log (training_date, player_id, training_result);

Вопрос:

На каждую тренировочную игру выведите какую долю составляет ее результат от суммарного этого игрока.

123 player_id T↕	🕒 training_date T↕	123 numeric T↕
1	2020-01-30	1
2	2020-01-11	0,29
2	2020-01-18	0,29
2	2020-01-20	0,24
2	2020-01-24	0,06
2	2020-01-25	0,06
2	2020-01-28	0,06



Работаем вместе



8 мин

Summary по блоку 1

- ▶ Агрегатные функции могут работать с оконками
- ▶ ... `over()` - работа со всем датасетом
- ▶ `distinct` не работает с оконками, нужно быть аккуратней с дублями



Теория



Перерыв

10 минут



rows between ... and ...
range between ... and ...



Вопросы по прериду

- Чем отличаются **rows between** и **range between**?
- Как указать в какую сторону от текущей строки смещать границу окна во время расчета функции?
- Можно ли начать расчет с начала окна и до текущей строки? Если да, то что нужно написать в `over(...)`?



Обсуждение



5 минут

Чем отличаются результаты запросов?

```
select match_date, player_id, sum(score)
over(partition by player_id order by match_date
rows between unbounded preceding and
current row)
from d8_tournament_log
where player_id=1
order by player_id, match_date
```

```
select match_date, player_id, sum(score) over
(partition by player_id order by match_date
range between unbounded preceding and current
row)
from d8_tournament_log
where player_id=1
order by player_id, match_date
```

match_date	player_id	opponent_id	match_result	score
2020-01-31	1	5	-1	98
2020-01-31	1	3	1	66
2020-02-01	1	2	-1	30
2020-02-02	1	4	1	72

```
select *
from d8_tournament_log
where player_id=1
```







Обсуждение







5 минут

Чем отличаются результаты запросов?

```
select match_date, player_id, sum(score)
over(partition by player_id order by match_date
rows between unbounded preceding and
current row)
from d8_tournament_log
where player_id=1
order by player_id, match_date
```

 match_date 	123 player_id 	123 sum 
2020-01-31	1	98
2020-01-31	1	164
2020-02-01	1	194
2020-02-02	1	266

```
select match_date, player_id, sum(score) over
(partition by player_id order by match_date
range between unbounded preceding and current
row)
from d8_tournament_log
where player_id=1
order by player_id, match_date
```

 match_date 	123 player_id 	123 sum 
2020-01-31	1	164
2020-01-31	1	164
2020-02-01	1	194
2020-02-02	1	266

Задача 6

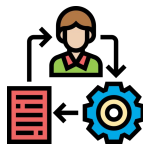
Условие:

-- d8_training_log (training_date, player_id, training_result);

Вопрос:

Для каждого игрока подсчитайте накопленный training result ко дню каждой тренировки. Используйте **rows between**.

🕒 match_date 📈	123 player_id 📈	123 sum 📈
2020-01-31	1	164
2020-02-01	1	194
2020-02-02	1	266
2020-01-13	2	95
2020-01-25	2	150
2020-01-27	2	234
2020-02-01	2	249



Практика



8 минут





Задача 7

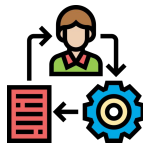
Условие:

-- d8_training_log (training_date, player_id, training_result);

Вопрос:

Для каждого игрока выведите сумму очков, которые он получил с текущей даты включительно и за следующие 3 дня.

 match_date 	123 player_id 	123 score 
2020-01-31	1	266
2020-02-01	1	102
2020-02-02	1	72
2020-01-13	2	95
2020-01-25	2	139
2020-01-27	2	84
2020-02-01	2	15



Практика



8 минут

Задача 8

Условие:

-- d8_training_log (training_date, player_id, training_result);

Вопрос:

Для каждого игрока выведите сумму очков, которые он получил с текущей даты включительно и за следующие 3 тренировки.

	🕒 match_date 📈📉	123 player_id 📈📉	123 sum 📈📉
1	2020-01-31	1	266
2	2020-02-01	1	102
3	2020-02-02	1	72
4	2020-01-13	2	249
5	2020-01-25	2	154
6	2020-01-27	2	99
7	2020-02-01	2	15



Практика



3 минут

Summary по блоку 2

- ▶ **rows between ... and ...** задает смещение в количестве строк
- ▶ **range between ... and ...** задает смещение как дельту от значения текущей строки в колонке из сортировки
- ▶ **preceding** и **following** указывают в какую сторону от текущей строки смещать границу
- ▶ начало и конец окна задаются ключевым словом **unbounded**, а текущая строка - ключевым словом **current row**



Теория



```
row_number  
rank  
dense_rank
```



Вопросы по прериду

- Чем отличаются **RANK** и **DENSE RANK**?
- Какая функция возвращает для всех строк в выбранном разрезе(окне) разные номера при ранжировании?



Обсуждение



5 минут

Задача 9

Условие:

```
-- d8_player (player_id int, player_name text, registration_date date);
```

```
-- d8_player_city (player_id int, city text, actual_date date);
```

Вопрос:

Для каждого игрока найдите последний актуальный город.

Какой последний город у игрока с id 3?

123 player_id	↑↓	ABC city	↑↓
	1	Владивосток	
	2	[NULL]	
	3	Москва	
	4	[NULL]	
	5	Владивосток	



Работаем вместе



12 мин

Задача 10

Условие:

-- d8_tournament_log (match_date, player_id, opponent_id, match_result, score);

Вопрос:

Найдите игроков с наибольшим количеством побед.
выведите игроков занявших призовые 1, 2 и 3 место.
Несколько игроков могут занимать одно место.

123 player_id	123 total_result	123 total_score	123 dense_rank
2	4	249	1
1	2	266	2
3	2	249	2
5	2	138	2
4	0	230	3



Практика



12 минут

Summary по блоку 3

- ▶ **ROW_NUMBER** — Дубликаты сортируются произвольным образом (1, 2, 3, 4, 5, ...)
- ▶ **RANK** — Дубликаты получают одинаковый номер. Строка следующая за дубликатом получает тот же номер что и в случае ROW_NUMBER (1, 2, 2, 2, 5, ...)
- ▶ **DENSE_RANK** — Дубликаты получают одинаковый номер. Строка следующая за дубликатом - номер на единицу больше (1, 2, 2, 2, 3, ...)



Теория

Вопросы

Что осталось непонятым?

Фидбек

Что нового вы узнали на этом занятии?

Что показалось самым важным?

Что будете применять и в каких ситуациях?

Что хочется изучить подробнее?

Ссылка на Обратную связь.

Что делаем в следующий раз?

- ▶ Находить следующее значение в том же разрезе по заданному порядку
- ▶ Еще раз вспомним про `cte` и поговорим про рекурсию

Обязательно ознакомьтесь с преридом =)

Домашнее задание

Мягкий дедлайн: 10:00 BC

Жесткий дедлайн: 10:00 BT

ОБСУЖДЕНИЕ ДЗ

01.

Найдите фамилию сотрудника с максимальной датой последнего действия

```
with users as (  
    select surname, last_action_date from d7_buyer  
    union all  
    select surname, last_action_date from d7_seller  
    union all  
    select surname, last_action_date from d7_manager  
)  
select distinct surname  
from users  
where last_action_date = (select max(last_action_date) from users)  
order by 1
```

ОБСУЖДЕНИЕ ДЗ

02. Найдите месяцы в которых было более трех последних действий покупателей и более двух последних действий продавцов

```
with
sq1 as (
    select DATE_FORMAT(last_action_date, '%Y-%m-01') as month from d7_buyer group by 1
having count(*) > 3),
sq2 as (
    select DATE_FORMAT(last_action_date, '%Y-%m-01') as month from d7_seller group by 1
having count(*) > 2)
select sq1.month
from sq1
join sq2 using(month)
```

ОБСУЖДЕНИЕ ДЗ

03. Для каждой роли подсчитайте максимальную дату последнего действия пользователей.

```
select * from (  
  select 'b' as role, max(last_action_date) last_action_date from d7_buyer  
  union all  
  select 's' as role, max(last_action_date) last_action_date from d7_seller  
  union all  
  select 'm' as role, max(last_action_date) last_action_date from d7_manager  
) sq
```

ОБСУЖДЕНИЕ ДЗ

- 04.** Посчитайте количество пользователей в разрезе квартала регистрации и месяца.
Подведите итоги по каждому разрезу и общий итог, отсортируйте по первой и второй колонке.
Формат ответа 'q', 'm', 'cntd_users'

```
select quarter(registration_date) as q,  
       month(registration_date) as m,  
       count(distinct id) cntd_users  
from d7_user  
group by 1,2 with rollup  
order by 1,2
```


ОБСУЖДЕНИЕ ДЗ

05. Реализуйте аналог d7_set3 INTERSECT ALL d7_set4 через другие операции (UNION ALL можно использовать). Результат отсортируйте по возрастанию.

```
with s1 as (select 's1' s, n from d7_set3)
,s2 as (select 's2' s, n from d7_set4)
,u as (select * from s1 union all select * from s2)
,n_cnt as (select n, least(sum(case when s = 's1' then 1 else 0 end), sum(case when s = 's2' then 1 else 0 end)) cnt
           from u
           group by n)
select n_cnt.n
from n_cnt
join numbers on numbers.n <= n_cnt.cnt
where cnt > 0
order by 1
```

ОБСУЖДЕНИЕ ДЗ

06. Реализуйте аналог d7_set3 INTERSECT ALL d7_set4 через другие операции.
UNION использовать нельзя.

```
with s1 as (select n, count(*) as cnt from d7_set3 group by 1)
, s2 as (select n, count(*) as cnt from d7_set4 group by 1)
, n_cnt as (
    select s1.n, least(s1.cnt, s2.cnt) as cnt
    from s1
    join s2 on s1.n = s2.n
)
select n_cnt.n
from n_cnt
join numbers on numbers.n <= n_cnt.cnt
where cnt > 0
order by 1
```

ОБСУЖДЕНИЕ ДЗ

07. Реализуйте аналог d7_set3 EXCEPT ALL d7_set4 через другие операции (UNION ALL можно использовать).

```
with s1 as (select 's1' s, n from d7_set3)
     ,s2 as (select 's2' s, n from d7_set4)
     ,u  as (select * from s1 union all select * from s2)
     ,n_cnt as ( select n, greatest(sum(case when s = 's1' then 1 else 0 end) - sum(case when s = 's2' then 1 else 0 end), 0) cnt
from u
group by n
)
select n_cnt.n
from n_cnt
join numbers on numbers.n <= n_cnt.cnt
where cnt > 0
order by 1
```

ОБСУЖДЕНИЕ ДЗ

08. Реализуйте аналог d7_set3 EXCEPT ALL d7_set4 через другие операции

UNION использовать нельзя.

```
with s1 as (select n, count(*) cnt from d7_set3 group by 1)
,s2 as (select n, count(*) cnt from d7_set4 group by 1)
,n_cnt as (select n, greatest(s1.cnt - coalesce(s2.cnt,0), 0) as cnt
            from s1
            left join s2 using (n))
select n_cnt.n
from n_cnt
join numbers on numbers.n <= n_cnt.cnt
where cnt > 0
order by 1
```