




# SQL

Оконные функции



# Что позволяет делать оконная функция?

Оконные (аналитические) функции позволяют делать агрегацию в запросе не накладывая ограничения, которые есть при обычной группировке. Такой подход позволяет сильно сократить запрос.

Помимо этого аналитические функции позволяют использовать в вычислениях соседние строки, что дает возможность решать более сложные аналитические задачи.



# Задача

Сформируйте поле `delta` которое содержит разницу между максимальной зарплатой в отделе и зарплатой сотрудника.



## Решение с аналитическими ф-ми

```
select
    FIRST_NAME,
    LAST_NAME,
    max(salary) over(partition by department_id) - salary as delta from
hr.employees;
```

**max(salary) over(partition by department\_id)**

**over()** - функция, позволяющая указать параметры расчета (окна).

**partition by** - поле группировки



## Использование order by

```
select
    CUSTOMER_ID,
    ORDER_DATE,
    sum(ORDER_TOTAL) over(partition by CUSTOMER_ID
                           order by ORDER_DATE) as sum_orders
from oe.orders;
```

Данный запрос позволит определить сумму текущего заказа и предыдущих (предыдущий определяется по порядку значений в поле **ORDER\_DATE**)



# Задание

У нас есть таблица проводок по счетам и мы хотим получить баланс после совершения каждой проводки.



# Скрипт

```
create table tBalance(  
  id number,  
  account varchar2(20),  
  value number  
);
```

```
insert into tBalance values (1,'01',100);  
insert into tBalance values (2,'01',200);  
insert into tBalance values (3,'01',-100);  
insert into tBalance values (4,'01',200);  
insert into tBalance values (5,'01',100);  
insert into tBalance values (6,'01',-100);  
insert into tBalance values (7,'01',100);  
insert into tBalance values (8,'02',10);  
insert into tBalance values (9,'02',20);  
insert into tBalance values (10,'02',-10);  
insert into tBalance values (11,'02',-20);  
insert into tBalance values (12,'02',10);  
insert into tBalance values (13,'02',-10);  
insert into tBalance values (14,'02',10);
```



## row\_number() vs rank() vs dense\_rank()

1 "value1"	1 "value1"	1 "value1"
2 "value1"	1 "value1"	1 "value1"
3 "value1"	1 "value1"	1 "value1"
4 "value1"	1 "value1"	1 "value1"
5 "value2"	5 "value2"	2 "value2"
6 "value2"	5 "value2"	2 "value2"
7 "value2"	5 "value2"	2 "value2"
8 "value2"	5 "value2"	2 "value2"





# Задача

Определите id департамента, который стоит на втором месте (в списке по убыванию) по кол-ву сотрудников.



# Задача

Определите id департаментов, которые занимают третье место по кол-ву сотрудников.



# lag() lead()

```
select
  a.*,
  lag(pay,1) over (partition by name order by id) as lastPay
from Payouts a
```



# Задача

Определить разницу зарплаты сотрудника по отношению к зарплате предыдущего и последующего сотрудника по размеру зарплаты.



# ROWS BETWEEN

```
select a.*,  
sum(pay) over (  
  partition by name  
  order by id ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) as num  
from payouts a
```



# Задача

Необходимо рассчитать поле `coef`, которое хранит среднее значение среди текущего значения заказа, предыдущего и следующего у клиента (ранжировать по дате).