**Аналитические функции в SQL**

**Аналитический SQL**демонстрирует исключительную производительность, запросы с использованием Аналитического SQL потребляют меньше ресурсов и выполняются быстрее.  
  
Применяется в основном для отчетности, для следующих типов задач

1. Запросы рейтингов, первых N
2. Запросы с накопительным итогом
3. Запросы с конструкцией окна
4. Может применятся и для оптимизации запросов

Какие бывают функции в аналитическом, разберём основные  
**ROW\_NUMBER() – номер строки в группе  
RANK(f) – f имя поля, относительный ранг записи в группе**.

**LAG(f, n) –f имя поля, n предыдущее значение в группе, m – знач по умолчанию. Значение поля в группе, которое находится на n-ом месте с конца.  
LEAD(f, n) - f имя поля, n последующее значение в группе. Значение поля в группе, которое находится на n-ом месте с конца.  
FIRST\_VALUE(f) – f имя поля, первое значение в группе ,  
LAST\_VALUE(f) –f имя поля, последнее значение в группе  
STD\_DEV(f) – f имя поля, значение стандартного распределения в группе  
SUM(f) – f имя поля, накопительная сумма по группе  
AVG (f)– f имя поля, среднее по группе заданной групп**  
Синтаксис

SELECT аналитическая функция OVER([PARTITION партицирование…]

ORDER BY (упорядочивание выражение 2 [,…] [{ASC/DESC}] [{NULLS FIRST/NULLS LAST}]) а

где Партишен - это некоторое количество записей с общими ключами на которую будет распространятся действие Аналитической функции , сортировка внутри партишена осуществляется с помощью **order by**  
  
  
Проще всего разобраться с аналитическими функциями на примерах.  
Подготовим необходимые данные для демонстрации

**1. Список заказов с их кумулятивной стоимостью (нарастающим итогом) клиентам и датам.**

**По таблице Orders получить на каждый день суммарный расход за этот день и все предыдущие дни.**

|  |  |  |  |
| --- | --- | --- | --- |
| **OrderId** | [**ClientId**](http://www.sql-tutorial.ru/ru/book_running_totals.html) | [**date**](http://www.sql-tutorial.ru/ru/book_running_totals.html) | **sum** |
| 1 | 2 | 2001-03-22 00:00:00.000 | 1440.00 |
| 2 | 2 | 2001-03-29 00:00:00.000 | 7848.00 |
| 3 | 2 | 2001-04-02 00:00:00.000 | 2040.00 |

Фактически, чтобы решить задачу нам нужно добавить еще один столбец, содержащий накопительный итог (run\_tot). В соответствии с темой, этот столбец будет представлять собой коррелирующий подзапрос, в котором для ТОГО ЖЕ пункта, что и у ТЕКУЩЕЙ строки включающего запроса, и для всех дат, меньших либо равных дате ТЕКУЩЕЙ строки включающего запроса, будет подсчитываться сумма значений столбца out:

1. SELECT ClientId, date, sum,
2. (SELECT SUM(sum)
3. FROM Orders o2
4. WHERE point = o1.point AND date <= o1.date) run\_total
5. FROM Orders o1
6. WHERE ClientId IN (SELECT DISTINCT ClientId FROM Orders)
7. ORDER BY ClientId, date;

SELECT ClientId, date, SUM(sum) over (partition by ClientId order by date) as CumSum  
FROM Orders  
ORDER BY ClientId, date

**Другой пример**:

Create table PersonA(Tbn number primary key, name varchar2(20), otd number, sal number);

*--Табельный номер , имя, отдел , зарплата*

**2. Запросы списка лидеров**  
Первые три сотрудника с самой высокой зарплатой по отделам (партишен по отделу)

select \* from ( select name , otd , sal , row\_number() over (partition by otd order by sal desc)

as num from personA) where num<4

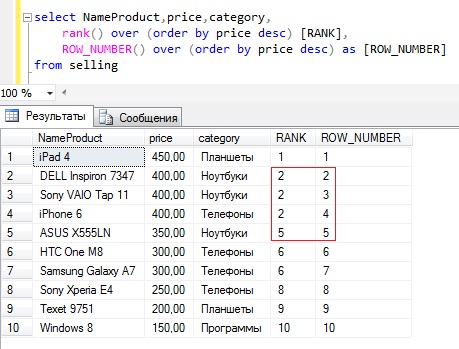
Более корректно 

select \* from ( select name , otd , sal

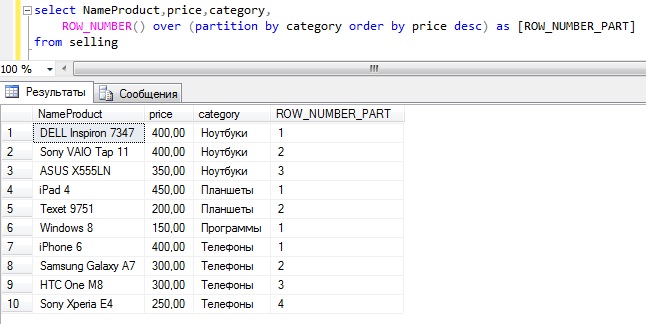
, rank() over (partition by otd order by sal desc) as num from personA) where num<4

**Внимание ! Пример демонстрирует отличие rank() от row\_number()**

**2.1. *Пример без группировки с сортировкой по цене и отличие от row\_number()***



***2.2. Пример с группировкой по категории и с сортировкой по цене***



**3. По наименованию (партишн по отделу) сортировка по name**

select \* from ( select name , otd , sal , row\_number() over (partition by otd order by name) as num

from personA) where num<4

4. Накопительный итог по зарплате 

select name , otd , sal , sum(sal) over (partition by otd order by sal) as num from personA

5. Среднее по зарплате в рамках отдела

select name , otd , sal , avg(sal) over (partition by otd order by sal) as num from personA

6. Демонстрация работы **lag, leed** - сотрудник , отдел , зарплата , сотрудник с более большей заплатой (maxsl),  
, сотрудник с менее меньшей заплатой чем данный(minsl) в рамках отдела

select name , otd , sal , lead(name,1) over (partition by otd order by sal) as maxsl

, lag(name,1) over (partition by otd order by sal) as minsl

from personA

запрос демонстрирует конструкцию окна в рамках отдела , среднее по зарплате, вычисляется, не только в рамках отдела , но так же и в рамках окна из 3х строк

select name , otd , sal

, avg(sal) over (partition by otd order by sal ROWS BETWEEN 3 PRECEDING AND CURRENT ROW) as num from personA

в запросе используется конструкция окна ROWS BETWEEN N PRECEDING AND CURRENT ROW  
то есть, для вычисления среднего avg(sal), считаются 3 предыдущие строки перед текущей строкой 

select name , otd , sal ,

first\_value(name) over (partition by otd) as maxsl, last\_value(name) over (partition by otd) as minsl

from personA

этот запрос демонстрирует применение аналитических функций **first\_value last\_value**