

# Занятие 4

## Оконные функции

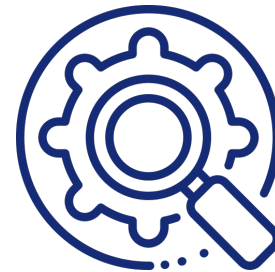
Бояр Владислав

# Занятие состоит из:



## Теория:

- Оконные функции
- Виды оконных функций
- Примеры использования



## Практика:

- Написание запросов с использованием оконных функций

# Оконные функции

# Что такое оконная функция?

**Оконная функция (window function)** – функция, которая проводит операции над выделенными группами строк (окна / партиции) и выводит результат выполнения в отдельной колонке

**Пример партиции оконной функции:**

**Партиции**  
оконной функции  
(в данном примере  
по полю **Имя**)



Имя	Предмет	Оценка
Петя	матем	3
Петя	рус	4
Петя	физ	5
Петя	история	4
Маша	матем	4
Маша	рус	3
Маша	физ	5
Маша	история	3

# Отличие оконной функции от группировки

- ➡ оконные функции в отличие от группировки не уменьшают кол-во строк
- ➡ при группировке для вывода доступны только участвующие в группировке колонки
- ➡ при применении оконных функций для вывода доступны все колонки

# Отличие оконной функции от группировки

Применение функции агрегации и команды GROUP BY

Имя	Предмет	Оценка
Петя	матем	3
Петя	рус	4
Петя	физ	5
Петя	история	4
Маша	матем	4
Маша	рус	3
Маша	физ	5
Маша	история	3



Имя	Средняя оценка
Петя	4
Маша	3,75

# Отличие оконной функции от группировки

Применение Оконной функции

Имя	Предмет	Оценка
Петя	матем	3
Петя	рус	4
Петя	физ	5
Петя	история	4
Маша	матем	4
Маша	рус	3
Маша	физ	5
Маша	история	3




Имя	Предмет	Оценка	Средняя оценка
Петя	матем	3	4
Петя	рус	4	4
Петя	физ	5	4
Петя	история	4	4
Маша	матем	4	3,75
Маша	рус	3	3,75
Маша	физ	5	3,75
Маша	история	3	3,75

# Место оконных функций в запросе

```
SELECT можно_здесь  
  FROM ...  
 WHERE ...  
 GROUP BY ...  
 HAVING ...  
 ORDER BY и_здесь
```



# Оконные функции в порядке очередности выполнения запроса

1. FROM / JOINS – выборка таблиц и их объединения
  2. WHERE – фильтрация значений
  3. GROUP BY / HAVING - группировки
  4. **SELECT**
  5. **ORDER BY**
- 

# Синтаксис оконных функций

```
FUNCTION(...) OVER ([PARTITION BY] ... [ORDER BY] ...)
```

- **FUNCTION** – название оконной функции
  - **PARTITION BY** – условие группировки
  - **ORDER BY** – условие сортировки
- 

# 2 варианта определения оконных функций

дубли определения окна

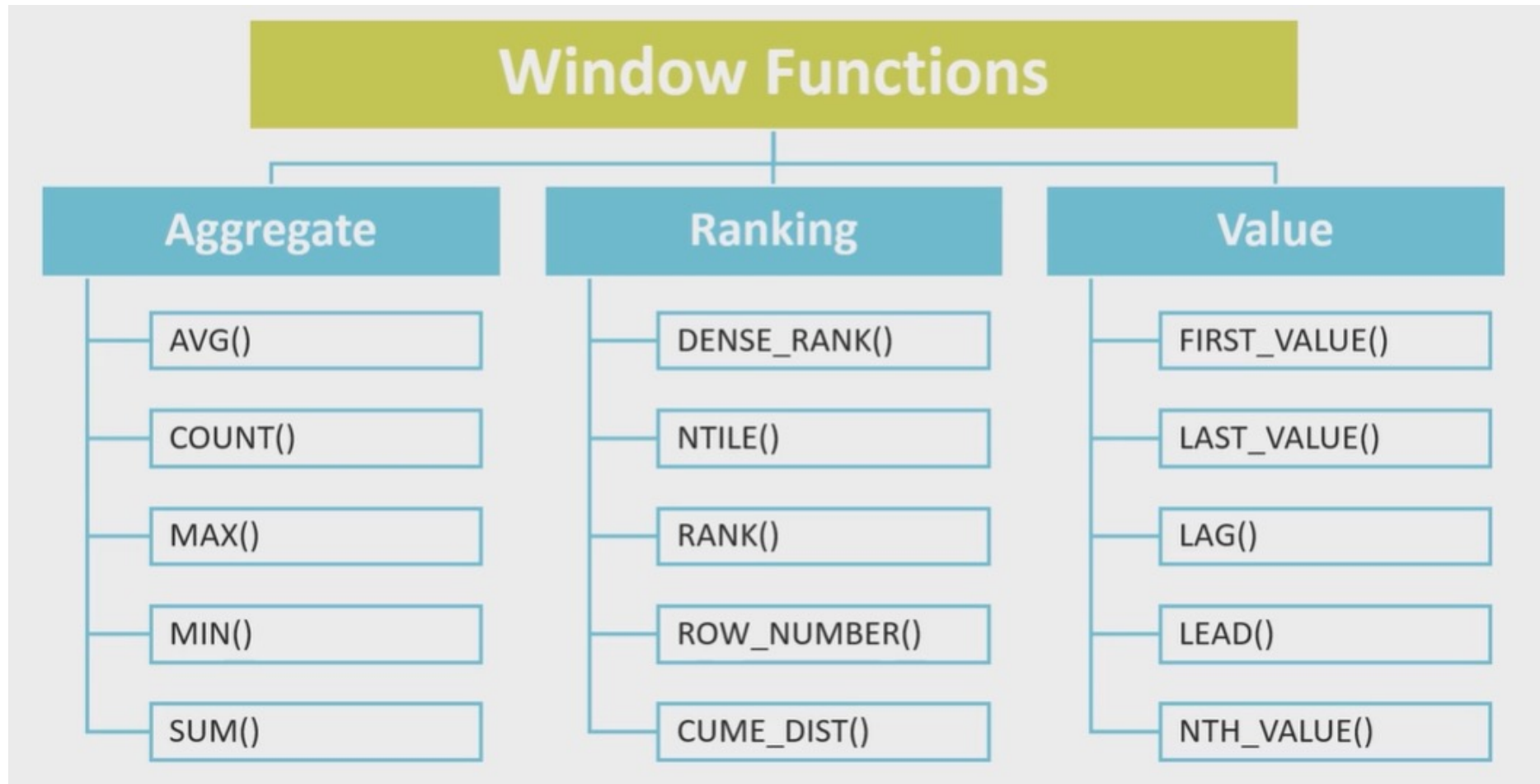
```
select name, subject, grade,  
row_number() over (partition by name order by grade desc),  
rank() over (partition by name order by grade desc),  
dense_rank() over (partition by name order by grade desc)  
from student_grades;
```

=

```
select name, subject, grade,  
row_number() over name_grade,  
rank() over name_grade,  
dense_rank() over name_grade  
from student_grades  
window name_grade as (partition by name order by grade desc);
```

# Виды оконных функций

- Агрегирование
- Ранжирование
- Смещение



# Агрегирующие функции

# Агрегирующие функции

- все агрегатные функции работают по одному принципу: на наборе данных выполняют арифметические вычисления и возвращают итоговое значение;
- оконные агрегатные функции отличаются от обычных агрегатных функций тем, что они не изменяют количество строк в результате запроса
- ключевое преимущество оконных агрегирующих функций: на одни и те же данные можно смотреть в разных разрезах в рамках одного запроса

ABC name ▼	ABC subject ▼	123 grade ▼	123 sum_grade ▼	123 avg_grade ▼	123 count_grade ▼	123 min_grade ▼	123 max_grade ▼
Маша	математика	4	16	4	4	3	5
Маша	физика	5	16	4	4	3	5
Маша	история	3	16	4	4	3	5
Маша	русский	4	16	4	4	3	5
Петя	физика	5	13	4,33	3	4	5
Петя	история	4	13	4,33	3	4	5
Петя	русский	4	13	4,33	3	4	5

# Агрегирующие функции

## Нарастающий итог SUM + ORDER BY

Мы можем применить оконную функцию sum, чтобы посчитать сумму значений нарастающим итогом






```
SELECT "name"  
      ,subject  
      ,grade  
      ,sum(grade) OVER (partition by name ORDER BY subject) AS current_total_grade  
      ,sum(grade) OVER (partition by name) AS total_grade  
FROM student_grades;
```

# Агрегирующие функции

## Нарастающий итог SUM + ORDER BY

Мы можем применить оконную функцию sum, чтобы посчитать сумму значений нарастающим итогом

```
SELECT "name"  
      ,subject  
      ,grade  
      ,sum(grade) OVER (partition by name ORDER BY subject) AS current_total_grade  
      ,sum(grade) OVER (partition by name) AS total_grade  
FROM student_grades;
```

 name ▼	 subject ▼	 grade ▼	 current_total_grade ▼	 total_grade ▼
Маша	история	3	3	16
Маша	математика	4	7	16
Маша	русский	4	11	16
Маша	физика	5	16	16
Петя	история	4	4	13
Петя	русский	4	8	13
Петя	физика	5	13	13



# Агрегирующие функции

## Нарастающий итог **SUM + ORDER BY**

Мы можем применять оконную функцию `sum`, чтобы посчитать сумму значений нарастающим итогом

Вместо `sum` мы можем применить любую другую агрегирующую функцию.

## **NULL**

все агрегатные функции, кроме `count(*)`, игнорируют `NULL` значения .



# Функции ранжирования

# ROW\_NUMBER

- вычисляет порядковый номер строк внутри окна (партиции) в указанном порядке
- порядковый номер не зависит от повторяющихся значений в строках
- порядок сортировки внутри оконной функции не связан с общей сортировкой запроса

ABC name ▼	ABC subject ▼	123 grade ▼	123 row_number ▼
Маша	физика	5	1
Маша	математика	4	2
Маша	русский	4	3
Маша	история	3	4
Петя	физика	5	1
Петя	русский	4	2
Петя	история	4	3

# RANK

- вычисляет ранг каждой строки внутри партии
- при повторяющихся значениях, функция вернет одинаковое значение ранга для них, пропуская при этом числовой ранг

ABC name ▼	ABC subject ▼	123 grade ▼	123 rank ▼
Маша	физика	5	1
Маша	математика	4	2
Маша	русский	4	2
Маша	история	3	4
Петя	физика	5	1
Петя	русский	4	2
Петя	история	4	2

**Пример:** рейтинг студентов

# DENSE\_RANK

- то же самое, что и RANK, только в случае одинаковых значений DENSE\_RANK не пропускает следующий числовой ранг, а идет последовательно

ABC name ▼	ABC subject ▼	123 grade ▼	123 rank ▼	123 dense_rank ▼
Маша	физика	5	1	1
Маша	математика	4	2	2
Маша	русский	4	2	2
Маша	история	3	4	3
Петя	физика	5	1	1
Петя	русский	4	2	2
Петя	история	4	2	2

# NTILE

- разделяет результирующий набор на указанное количество групп

## Как происходит вычисление:

- результат запроса сортируется по условию ORDER BY
- отсортированные строки равномерно делятся на количество указанных групп
- если количество строк не делится целочисленно на количество групп, то в первых группах будет больше на одну строку

ABC name ▼	ABC subject ▼	123 grade ▼	123 ntile ▼
Маша	физика	5	1
Маша	математика	4	1
Маша	русский	4	2
Маша	история	3	2
Петя	физика	5	1
Петя	русский	4	1
Петя	история	4	2

# NULL

## NULL

в функциях ранжирования NULL значениям присваивается одинаковый ранг

# Функции смещения









# Функции смещения

— позволяют обращаться к значениям в рамках партиции

# LAG and LEAD

- LAG() – возвращает предыдущее значение в порядке сортировки в рамках партии
- LEAD() – возвращает следующее значение в порядке сортировки в рамках партии

```
SELECT "name"  
      ,quartal  
      ,subject  
      ,LAG(grade) OVER (ORDER BY quartal) AS previous_grade  
      ,grade  
      ,LEAD(grade) OVER (ORDER BY quartal) AS next_grade  
FROM grades_quartal;
```

ABC name 	ABC quartal 	ABC subject 	123 grade 	123 previous_grade 	123 next_grade 
Петя	1 четверть	физика	4	[NULL]	3
Петя	2 четверть	физика	3	4	4
Петя	3 четверть	физика	4	3	5
Петя	4 четверть	физика	5	4	[NULL]

# FIRST and LAST

- FIRST\_VALUE() – возвращает первое значение в порядке сортировки в рамках партиии
- LAST\_VALUE() – возвращает последнее значение в порядке сортировки в рамках партиии

```
SELECT "name"  
      ,quartal  
      ,subject  
      ,FIRST_VALUE(grade) OVER(ORDER BY quartal) AS first_grade  
      ,grade  
      ,LAST_VALUE(grade) OVER(ORDER BY quartal) AS last_grade  
FROM grades_quartal;
```

ABC name ▼	ABC quartal ▼	ABC subject ▼	123 first_grade ▼	123 grade ▼	123 last_grade ▼
Петя	1 четверть	физика	4	4	4
Петя	2 четверть	физика	4	3	3
Петя	3 четверть	физика	4	4	4
Петя	4 четверть	физика	4	5	5

Практика

