ПРАКТИЧЕСКАЯ РАБОТА №4. ОКОННЫЕ ФУНКЦИИ

Оконные функции в MySQL — это специальный тип функции, который позволяет выполнять агрегатные и аналитические операции над группами строк, которые определены внутри отдельного окна (или оконного фрейма). Оконные функции представляют собой удобный инструмент по работы с данными и предоставляют более гибкий способ обращения с ними, чем традиционные агрегатные функции за счёт того, что они могут учитывать порядок сортировки данных и разбивать их на группы без фактической группировки.

Сферы применения оконных функций:

1. Анализ данных внутри групп: Оконные функции позволяют проводить анализ данных внутри определённых групп, например, внутри каждой конкретной категории товара или для каждого отдельного клиента. Это может быть полезно для получения оперативной аналитики по каждой определённой группе.
2. Анализ временных рядов:

Оконные функции широко используются при работе с временными рядами, например, для сравнения текущих значений с предыдущими периодами, выявления сезонных изменений и т.д.

1. Ранжирование и сортировка:

Оконные функции помогают в определении ранга элемента(-ов) в наборе данных и в выполнении сложных операций сортировки, например, вычислении позиции строки в отдельно взятой группе.

1. Анализ статистических данных:

Оконные функции полезны для статистического анализа данных, например, для вычисления средних и медианных значений, стандартного отклонения и дисперсии (аналитические функции не будут рассмотрены в рамках данного курса).

1. Сравнение данных внутри групп:

Оконные функции могут использоваться для сравнения значений внутри группы, что позволяет выявить аномалии или выполнить дополнительный анализ в контексте группы данных.

1. Создание отчётов и аналитика:

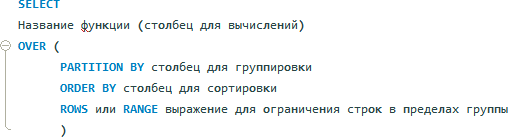
Оконные функции часто применяются в бизнес-аналитике и используются при создании отчётов для вычисления сложных агрегированных показателей или результатов анализа данных. (аналитические функции не будут рассмотрены в рамках данного курса).

# Синтаксис оконных функций

Возьмём пример простейшей оконной функции и разберём подробно синтаксис. Это поможет понять, для чего используется те или иные инструкции и операторы.

*OVER()*

Окно для функции определяется посредством обязательной инструкции OVER(). Синтаксис этой инструкции выглядит так:



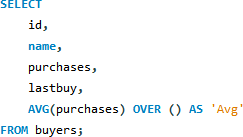
*Рисунок 1. Синтаксис инструкции OVER()*

Теперь рассмотрим, как будет изменяться таблица при использовании тех или иных ключевых слов функций. Для тренировки возьмём простую табличку. Она содержит стандартные данные: id покупателя, имя, количество покупок и дату последней покупки.



*Рисунок 2. Тестовая таблица*

Откроем окно при помощи OVER() и найдём среднее значение покупок среди всех покупателей:



*Рисунок 3. Открытие окна при помощи OVER()*



*Рисунок 4. Результат открытия окна*

Так как мы использовали инструкцию OVER() без предложений, то окном является весь набор данных, без сортировок и группировок. В результате мы видим новый столбец Avg, который для каждой строки выводит одно и то же значение — 2, что является средним значением покупок по всем покупателям.

*PARTITION BY()*

Теперь применим инструкцию PARTITION BY, которая используется для разделения таблицы на логические подгруппы данных, называемые разделами (partitions). Разделение происходит на основе одного или нескольких столбцов.



*Рисунок 5. Применение инструкции PARTITION BY()*

В результате получаем таблицу, сгруппированную по полю «lastbuy». Теперь для каждой из групп (коих тут 3) рассчитывается своё среднее значение.

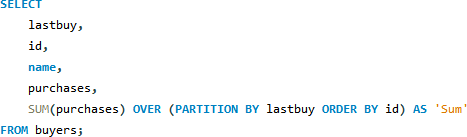


*Рисунок 6. Результат применения PARTITION BY()*

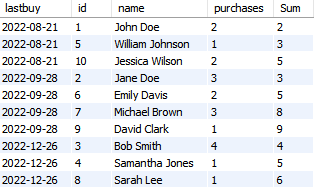
*ORDER BY*

Хоть вам и знаком ORDER BY, в рамках оконной функции оно пусть и выполняет сортировку, но имеет несколько иное назначение. Для более

наглядной демонстрации заменим функцию AVG на SUM. Попробуем применить ORDER BY для разделов, определённых ранее:



*Рисунок 7. Применение ORDER BY к разделам*



*Рисунок 8. Результат применения сортировки по разделам*

Как мы видим, в результате сортировки помимо упорядочивания значений id внутри разделов, у нас изменились и значения столбца Sum. Дело в том, что при помощи ORDER BY мы указали, что хотим видеть не просто сумму всех значений в окне, а для каждого значения покупок сумму с предыдущими

согласно сортировке. Таким образом, последнее значение столбца Sum является итоговым значением суммы покупок в данном разделе, а каждое предшествующее ему – значение данной строки и предыдущих, относящихся к этому разделу.

*ROWS или RANGE*

Инструкция ROWS позволяет ограничить строки в окне, указывая фиксированное количество строк, предшествующих или следующих за текущей. Инструкция RANGE же, в свою очередь, работает не со строками, а с диапазоном строк в инструкции ORDER BY. То есть в RANGE одной строкой

могут считаться несколько реальных строк, одинаковых по рангу.

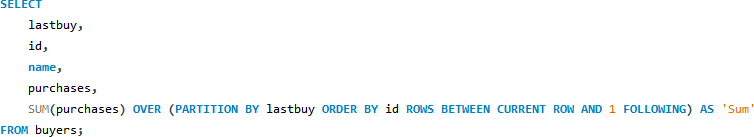
Обе инструкции всегда используются только вместе с ORDER BY.

Так же для ограничения строк посредством ROWS или RANGE можно использовать следующие ключевые слова:

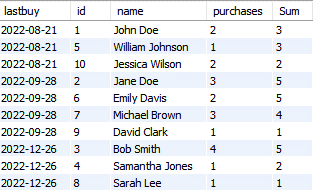
* UNBOUNDED PRECEDING — указывает, что окно начинается с первой строки группы;
* UNBOUNDED FOLLOWING – с помощью данной инструкции можно указать, что окно заканчивается на последней строке группы;
* CURRENT ROW – инструкция указывает, что окно начинается или заканчивается на текущей строке;
* BETWEEN «граница окна» AND «граница окна» — указывает нижнюю и верхнюю границу окна;
* «Значение» PRECEDING – определяет число строк перед текущей строкой (не допускается в предложении RANGE)\*;
* «Значение» FOLLOWING — определяет число строк после текущей строки (не допускается в предложении RANGE)\*.

\* В некоторых СУБД допустимо использование двух последних ключевых слов и вместе с RANGE, но способ их применения может немного отличаться от описанного здесь.

Рассмотрим пример применения нескольких из описанных выше инструкций:



*Рисунок 9. Применение ROWS с инструкциями CURRENT ROW и FOLLOWING*



*Рисунок 10. Результат выполнения с использованием инструкции ROWS*

В данном случае сумма будет рассчитываться по текущей и следующей ячейке в окне. Последнее значение окна будет иметь то же значение, что и столбец покупок, так как больше не с чем складывать.

В итоге, при правильной комбинации данных ключевых слов, можно подогнать диапазон работы вашей оконной функции под любую возникшую задачу.

# Виды оконных функций

В рамках данной работы разделим оконные функции на следующие 3 группы:

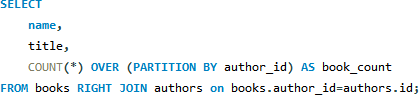
## Агрегатные функции:

Агрегатными функциями называются функции, которые выполняют арифметические вычисления на наборе данных и возвращают итоговое значение. К агрегатным функциям относятся:

* + COUNT(\*) — вычисляет количество значений в столбце (не учитывает значения NULL);
  + SUM(\*) — возвращает сумму значений в столбце;
  + AVG(\*) — определяет среднее значение в столбце;
  + MAX(\*) — определяет максимальное значение в столбце;
  + MIN(\*) — определяет минимальное значение в столбце.
    - — в качестве параметра принимает столбец, к которому будут применяться вычисления.

Теперь рассмотрим примеры использования каждой из оконных агрегатных функций:

### COUNT()



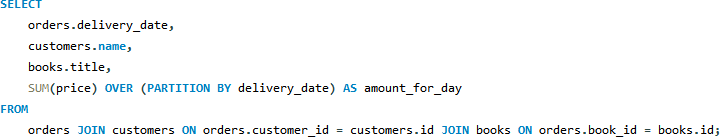
*Рисунок 11. Пример использования оконной агрегатной функции COUNT*



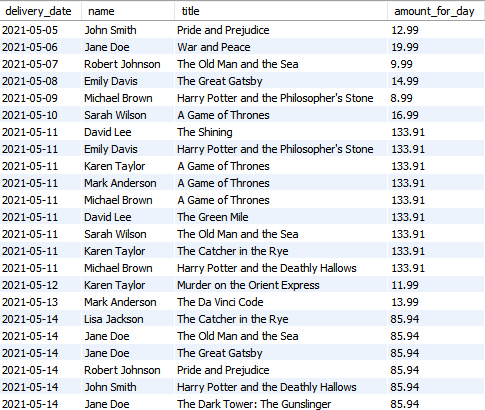
*Рисунок 12. Результат использования*

Результатом использования функции в данном случае является имя автора, название книги и количество книг в базе данных, написанных данным автором.

### SUM()



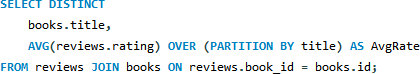
*Рисунок 13. Пример использования оконной агрегатной функции SUM*



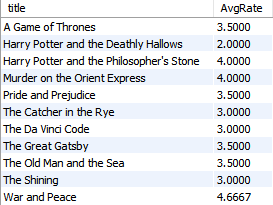
*Рисунок 14. Результат использования*

Результатом использования функции SUM в данном примере является выручка за день, определённая по разделам по дате доставки, а также добавлены имена заказчиков и название заказанных ими книг.

### AVG()



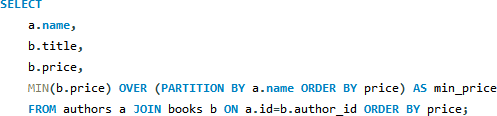
*Рисунок 15. Пример использования оконной агрегатной функции AVG*



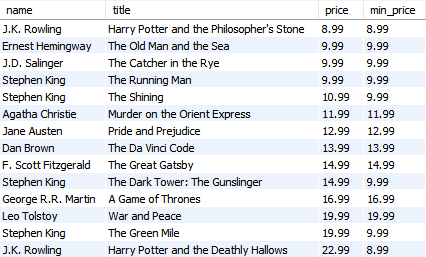
*Рисунок 16. Результат использования*

Результатом использования функции AVG в данном случае является значение рейтинга по всем книгам, находящимся в базе данных.

### MIN()



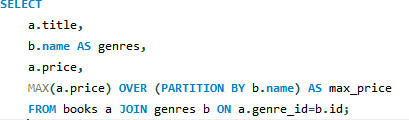
*Рисунок 17. Пример использования оконной агрегатной функции MIN*



*Рисунок 18. Результат использования*

В результате использования функции MIN выводятся строки, в которых для каждой книги будет указано имя автора, цена и минимальная цена среди книг этого автора.

### MAX()



*Рисунок 19. Пример использования оконной агрегатной функции MAX*



*Рисунок 20. Результат использования*

Результатом запроса является список книг и их жанры, а также цена книги и максимальная цена книги в этом жанре.

## Ранжирующие функции:

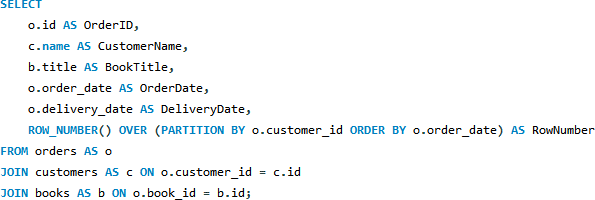
Ранжирующие функции — это функции, которые определяют ранг для каждой строки в окне. Например, их можно использовать для присвоения порядковых номеров или для составления рейтинга. К ранжирующим функциям относятся:

* ROW\_NUMBER() — функция возвращает номер строки и используется для нумерации;
* RANK() — функция возвращает ранг каждой строки. Данная функция в том числе анализирует данные и, в случае нахождения одинаковых – возвращает одинаковый ранг с пропуском следующего значения (например, два различных товара были проданы на одинаковую сумму по итогам месяца. При использовании данной функции для оценки ранга продаж за месяц обоим товарам будет выставлен ранг 1, а следующий за ними товар получит ранг 3);
* DENSE\_RANK() — так же, как и прошлая функция, возвращает ранг каждой строчки, но в отличие от функции RANK, следующий ранг пропускаться не будет;
* NTILE(\*) — это функция, которая позволяет определить, к какой группе относится текущая строка. Количество групп задаётся в скобках.

\* — в качестве параметра принимает количество групп, на которое будет разделено текущее окно.

Разберём использование ранжирующих оконных функций на примерах:

### ROW\_NUMBER()



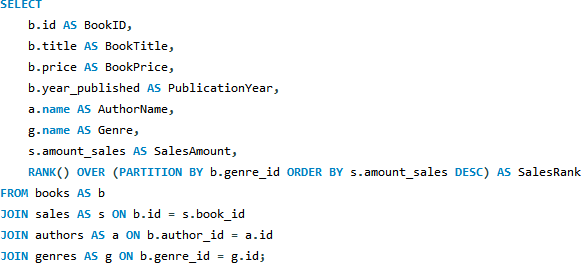
*Рисунок 21. Пример использования ранжирующей функции RANK*



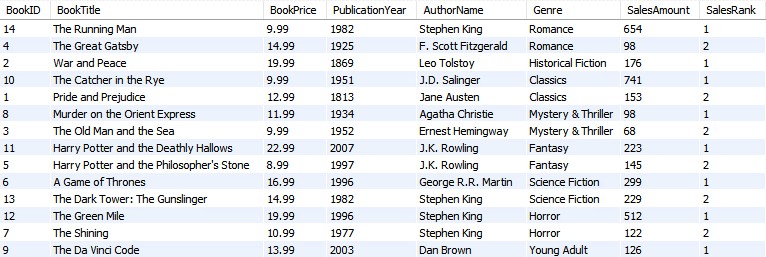
*Рисунок 22. Результат использования*

В этом запросе оконная функция ROW\_NUMBER() используется для присвоения номера строки каждой записи в таблице orders для каждого клиента. В результате получаем таблицу, которая включает в себя информацию о заказах, включая номер строки, присвоенный каждому заказу для каждого клиента.

### RANK()



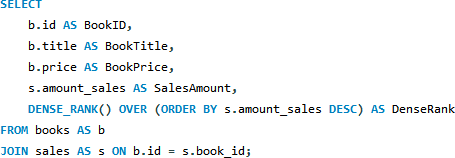
*Рисунок 23. Пример использования ранжирующей функции RANK*



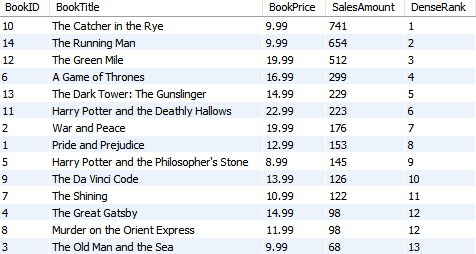
*Рисунок 24. Результат использования*

В данном запросе оконная функция RANK() используется для определения ранга продаж каждой книги внутри своего жанра.

### DENSE\_RANK()



*Рисунок 25. Пример использования ранжирующей функции DENSE\_RANK*

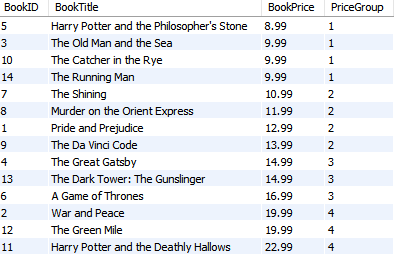


*Рисунок 26. Результат использования*

В данном запросе оконную функцию DENSE\_RANK() мы используем для определения ранга продаж каждой из книг в магазине. PARTITION BY в этом примере не используется, так как целью является вычисление общего ранга для всех книг на основе их продаж.

### NTILE(\*)

*Рисунок 27. Пример использования ранжирующей функции NTILE*



*Рисунок 28. Результат использования*

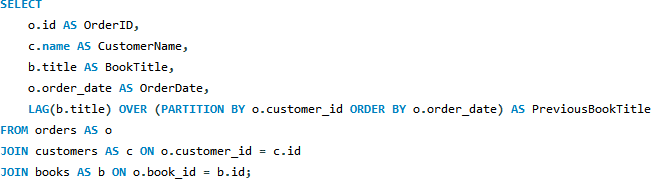
В результате запроса мы получаем отсортированную таблицу, которая разделена на 4 примерно равные ценовые группы.

## Функции смещения

Функции смещения — это функции, которые позволяют перемещаться и обращаться к разным строкам в окне относительно текущей строки, а также обращаться к значениям в начале или в конце окна. К функциям смещения относятся:

* + LAG(\*) и LEAD(\*) — функция LAG обращается к данным из предыдущей строки окна, а LEAD к данным из следующей строки. Функцию можно использовать для сравнения текущего значения строки с предыдущим или следующим. Имеет три параметра: столбец, значение которого необходимо вернуть, количество строк для смещения (по умолчанию это 1) и значение, которое необходимо вернуть, если после смещения возвращается значение NULL;
  + FIRST\_VALUE(\*) и LAST\_VALUE(\*) — функция FIRST\_VALUE позволяет получить первое значение в окне, а LAST\_VALUE, соответственно, последнее значение.
    - — в качестве параметра принимает столбец, по которому происходит смещение и значение которого необходимо вернуть. Рассмотрим использование оконных функций смещения на примерах. Так как функции LAG и LEAD, равно как FIRST\_VALUE и LAST\_VALUE являются обратными друг другу, то на примере будет разобрано лишь по одной из пары функций.

### LAG(\*) и LEAD(\*)



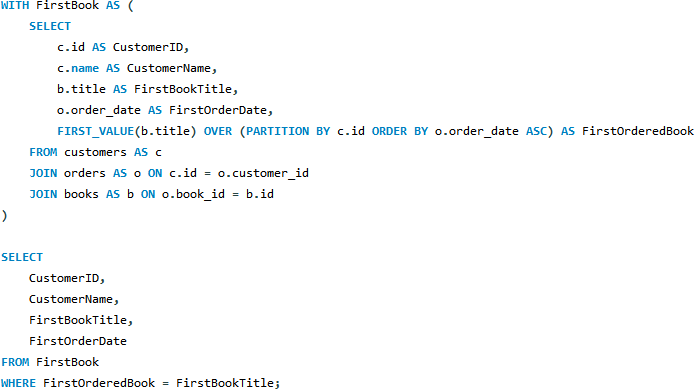
*Рисунок 29. Пример использования функции смещения LAG*



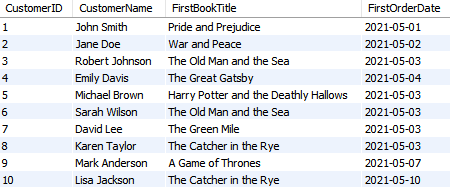
*Рисунок 30. Результат использования*

Функция LAG() в данном запросе используется для получения предыдущей книги, которая была заказана этим же клиентом.

### FIRST\_VALUE() и LAST\_VALUE()



*Рисунок 31. Пример использования функции смещения FIRST\_VALUE*



*Рисунок 32. Результат использования*

В данном случае при использовании функции FIRST\_VALUE() мы получаем имя клиента, а также его первую заказанную в магазине книгу и дату первого заказа.