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.

3. 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.

4. 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).

5. 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.

6. 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:

```
SELECT
Название функции (столбец для вычислений)
OVER (
      PARTITION BY столбец для группировки
      ORDER BY столбец для сортировки
      ROWS или RANGE выражение для ограничения строк в пределах группы
      )
```

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.

id	name	purchases	lastbuy
1	John Doe	2	2022-08-21
2	Jane Doe	3	2022-09-28
3	Bob Smith	4	2022-12-26
4	Samantha Jones	1	2022-12-26
5	William Johnson	1	2022-08-21
6	Emily Davis	2	2022-09-28
7	Michael Brown	3	2022-09-28
8	Sarah Lee	1	2022-12-26
9	David Clark	1	2022-09-28
10	Jessica Wilson	2	2022-08-21

Рисунок 2. Тестовая таблица

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

```
SELECT
    id,
    name,
    purchases,
    lastbuy,
    AVG(purchases) OVER () AS 'Avg'
FROM buyers;
```

Рисунок 3. Открытие окна при помощи OVER()

id	name	purchases	lastbuy	Avg
1	John Doe	2	2022-08-21	2.0000
2	Jane Doe	3	2022-09-28	2.0000
3	Bob Smith	4	2022-12-26	2.0000
4	Samantha Jones	1	2022-12-26	2.0000
5	William Johnson	1	2022-08-21	2.0000
6	Emily Davis	2	2022-09-28	2.0000
7	Michael Brown	3	2022-09-28	2.0000
8	Sarah Lee	1	2022-12-26	2.0000
9	David Clark	1	2022-09-28	2.0000
10	Jessica Wilson	2	2022-08-21	2.0000

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.

```
SELECT
    lastbuy,
    id,
    name,
    purchases,
    AVG(purchases) OVER (PARTITION BY lastbuy) AS 'Avg'
FROM buyers;
```

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.

lastbuy	id	name	purchases	Avg
2022-08-21	1	John Doe	2	1.6667
2022-08-21	5	William Johnson	1	1.6667
2022-08-21	10	Jessica Wilson	2	1.6667
2022-09-28	2	Jane Doe	3	2.2500
2022-09-28	6	Emily Davis	2	2.2500
2022-09-28	7	Michael Brown	3	2.2500
2022-09-28	9	David Clark	1	2.2500
2022-12-26	3	Bob Smith	4	2.0000
2022-12-26	4	Samantha Jones	1	2.0000
2022-12-26	8	Sarah Lee	1	2.0000

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:

```
SELECT
    lastbuy,
    id,
    name,
    purchases,
    SUM(purchases) OVER (PARTITION BY lastbuy ORDER BY id) AS 'Sum'
FROM buyers;
```

Рисунок 7. Применение ORDER BY к разделам

lastbuy	id	name	purchases	Sum
2022-08-21	1	John Doe	2	2
2022-08-21	5	William Johnson	1	3
2022-08-21	10	Jessica Wilson	2	5
2022-09-28	2	Jane Doe	3	3
2022-09-28	6	Emily Davis	2	5
2022-09-28	7	Michael Brown	3	8
2022-09-28	9	David Clark	1	9
2022-12-26	3	Bob Smith	4	4
2022-12-26	4	Samantha Jones	1	5
2022-12-26	8	Sarah Lee	1	6

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:

```
SELECT
    lastbuy,
    id,
    name,
    purchases,
    SUM(purchases) OVER (PARTITION BY lastbuy ORDER BY id ROWS BETWEEN CURRENT ROW AND 1 FOLLOWING) AS 'Sum'
FROM buyers;
```

Рисунок 9. Применение ROWS с инструкциями CURRENT ROW и FOLLOWING

lastbuy	id	name	purchases	Sum
2022-08-21	1	John Doe	2	3
2022-08-21	5	William Johnson	1	3
2022-08-21	10	Jessica Wilson	2	2
2022-09-28	2	Jane Doe	3	5
2022-09-28	6	Emily Davis	2	5
2022-09-28	7	Michael Brown	3	4
2022-09-28	9	David Clark	1	1
2022-12-26	3	Bob Smith	4	5
2022-12-26	4	Samantha Jones	1	2
2022-12-26	8	Sarah Lee	1	1

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:

1. 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:

1.1) COUNT()

```
SELECT
    name,
    title,
    COUNT(*) OVER (PARTITION BY author_id) AS book_count
FROM books RIGHT JOIN authors on books.author_id=authors.id;
```

Рисунок 11. Пример использования оконной агрегатной функции COUNT

name	title	book_count
Jane Austen	Pride and Prejudice	1
Leo Tolstoy	War and Peace	1
Ernest Hemingway	The Old Man and the Sea	1
F. Scott Fitzgerald	The Great Gatsby	1
J.K. Rowling	Harry Potter and the Philosopher's Stone	2
J.K. Rowling	Harry Potter and the Deathly Hallows	2
George R.R. Martin	A Game of Thrones	1
Stephen King	The Shining	4
Stephen King	The Green Mile	4
Stephen King	The Dark Tower: The Gunslinger	4
Stephen King	The Running Man	4
Agatha Christie	Murder on the Orient Express	1
Dan Brown	The Da Vinci Code	1
J.D. Salinger	The Catcher in the Rye	1

Рисунок 12. Результат использования

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

1.2) SUM()

```
SELECT
    orders.delivery_date,
    customers.name,
    books.title,
    SUM(price) OVER (PARTITION BY delivery_date) AS amount_for_day
FROM
    orders JOIN customers ON orders.customer_id = customers.id JOIN books ON orders.book_id = books.id;
```

Рисунок 13. Пример использования оконной агрегатной функции SUM

delivery_date	name	title	amount_for_day
2021-05-05	John Smith	Pride and Prejudice	12.99
2021-05-06	Jane Doe	War and Peace	19.99
2021-05-07	Robert Johnson	The Old Man and the Sea	9.99
2021-05-08	Emily Davis	The Great Gatsby	14.99
2021-05-09	Michael Brown	Harry Potter and the Philosopher's Stone	8.99
2021-05-10	Sarah Wilson	A Game of Thrones	16.99
2021-05-11	David Lee	The Shining	133.91
2021-05-11	Emily Davis	Harry Potter and the Philosopher's Stone	133.91
2021-05-11	Karen Taylor	A Game of Thrones	133.91
2021-05-11	Mark Anderson	A Game of Thrones	133.91
2021-05-11	Michael Brown	A Game of Thrones	133.91
2021-05-11	David Lee	The Green Mile	133.91
2021-05-11	Sarah Wilson	The Old Man and the Sea	133.91
2021-05-11	Karen Taylor	The Catcher in the Rye	133.91
2021-05-11	Michael Brown	Harry Potter and the Deathly Hallows	133.91
2021-05-12	Karen Taylor	Murder on the Orient Express	11.99
2021-05-13	Mark Anderson	The Da Vinci Code	13.99
2021-05-14	Lisa Jackson	The Catcher in the Rye	85.94
2021-05-14	Jane Doe	The Old Man and the Sea	85.94
2021-05-14	Jane Doe	The Great Gatsby	85.94
2021-05-14	Robert Johnson	Pride and Prejudice	85.94
2021-05-14	John Smith	Harry Potter and the Deathly Hallows	85.94
2021-05-14	Jane Doe	The Dark Tower: The Gunslinger	85.94

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.

1.3) AVG()

```
SELECT DISTINCT
    books.title,
    AVG(reviews.rating) OVER (PARTITION BY title) AS AvgRate
FROM reviews JOIN books ON reviews.book_id = books.id;
```

Рисунок 15. Пример использования оконной агрегатной функции AVG

title	AvgRate
A Game of Thrones	3.5000
Harry Potter and the Deathly Hallows	2.0000
Harry Potter and the Philosopher's Stone	4.0000
Murder on the Orient Express	4.0000
Pride and Prejudice	3.5000
The Catcher in the Rye	3.0000
The Da Vinci Code	3.0000
The Great Gatsby	3.5000
The Old Man and the Sea	3.5000
The Shining	3.0000
War and Peace	4.6667

Рисунок 16. Результат использования

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

1.4) MIN()

```
SELECT
    a.name,
    b.title,
    b.price,
    MIN(b.price) OVER (PARTITION BY a.name ORDER BY price) AS min_price
    FROM authors a JOIN books b ON a.id=b.author_id ORDER BY price;
```

Рисунок 17. Пример использования оконной агрегатной функции MIN

name	title	price	min_price
J.K. Rowling	Harry Potter and the Philosopher's Stone	8.99	8.99
Ernest Hemingway	The Old Man and the Sea	9.99	9.99
J.D. Salinger	The Catcher in the Rye	9.99	9.99
Stephen King	The Running Man	9.99	9.99
Stephen King	The Shining	10.99	9.99
Agatha Christie	Murder on the Orient Express	11.99	11.99
Jane Austen	Pride and Prejudice	12.99	12.99
Dan Brown	The Da Vinci Code	13.99	13.99
F. Scott Fitzgerald	The Great Gatsby	14.99	14.99
Stephen King	The Dark Tower: The Gunslinger	14.99	9.99
George R.R. Martin	A Game of Thrones	16.99	16.99
Leo Tolstoy	War and Peace	19.99	19.99
Stephen King	The Green Mile	19.99	9.99
J.K. Rowling	Harry Potter and the Deathly Hallows	22.99	8.99

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.

1.5) MAX()

```
SELECT
    a.title,
    b.name AS genres,
    a.price,
    MAX(a.price) OVER (PARTITION BY b.name) AS max_price
    FROM books a JOIN genres b ON a.genre_id=b.id;
```

Рисунок 19. Пример использования оконной агрегатной функции MAX

title	genres	price	max_price
Pride and Prejudice	Classics	12.99	12.99
The Catcher in the Rye	Classics	9.99	12.99
Harry Potter and the Philosopher's Stone	Fantasy	8.99	22.99
Harry Potter and the Deathly Hallows	Fantasy	22.99	22.99
War and Peace	Historical Fiction	19.99	19.99
The Shining	Horror	10.99	19.99
The Green Mile	Horror	19.99	19.99
The Old Man and the Sea	Mystery & Thriller	9.99	11.99
Murder on the Orient Express	Mystery & Thriller	11.99	11.99
The Great Gatsby	Romance	14.99	14.99
The Running Man	Romance	9.99	14.99
A Game of Thrones	Science Fiction	16.99	16.99
The Dark Tower: The Gunslinger	Science Fiction	14.99	16.99
The Da Vinci Code	Young Adult	13.99	13.99

Рисунок 20. Результат использования

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

2. 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:

2.1) ROW_NUMBER()

```
SELECT
    o.id AS OrderID,
    c.name AS CustomerName,
    b.title AS BookTitle,
    o.order_date AS OrderDate,
    o.delivery_date AS DeliveryDate,
    ROW_NUMBER() OVER (PARTITION BY o.customer_id ORDER BY o.order_date) AS RowNumber
FROM orders AS o
JOIN customers AS c ON o.customer_id = c.id
JOIN books AS b ON o.book_id = b.id;
```

Рисунок 21. Пример использования ранжирующей функции RANK

OrderID	CustomerName	BookTitle	OrderDate	DeliveryDate	RowNumber
1	John Smith	Pride and Prejudice	2021-05-01	2021-05-05	1
22	John Smith	Harry Potter and the Deathly Hallows	2021-05-03	2021-05-14	2
2	Jane Doe	War and Peace	2021-05-02	2021-05-06	1
23	Jane Doe	The Dark Tower: The Gunslinger	2021-05-03	2021-05-14	2
11	Jane Doe	The Old Man and the Sea	2021-05-10	2021-05-14	3
12	Jane Doe	The Great Gatsby	2021-05-10	2021-05-14	4
3	Robert Johnson	The Old Man and the Sea	2021-05-03	2021-05-07	1
13	Robert Johnson	Pride and Prejudice	2021-05-10	2021-05-14	2
4	Emily Davis	The Great Gatsby	2021-05-04	2021-05-08	1
14	Emily Davis	Harry Potter and the Philosopher's Stone	2021-05-07	2021-05-11	2
21	Michael Brown	Harry Potter and the Deathly Hallows	2021-05-03	2021-05-11	1
5	Michael Brown	Harry Potter and the Philosopher's Stone	2021-05-05	2021-05-09	2
17	Michael Brown	A Game of Thrones	2021-05-07	2021-05-11	3
19	Sarah Wilson	The Old Man and the Sea	2021-05-03	2021-05-11	1
6	Sarah Wilson	A Game of Thrones	2021-05-06	2021-05-10	2
18	David Lee	The Green Mile	2021-05-03	2021-05-11	1
7	David Lee	The Shining	2021-05-07	2021-05-11	2
20	Karen Taylor	The Catcher in the Rye	2021-05-03	2021-05-11	1
15	Karen Taylor	A Game of Thrones	2021-05-07	2021-05-11	2
8	Karen Taylor	Murder on the Orient Express	2021-05-08	2021-05-12	3
16	Mark Anderson	A Game of Thrones	2021-05-07	2021-05-11	1
9	Mark Anderson	The Da Vinci Code	2021-05-09	2021-05-13	2
10	Lisa Jackson	The Catcher in the Rye	2021-05-10	2021-05-14	1

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.

2.2) RANK()

```
SELECT
    b.id AS BookID,
    b.title AS BookTitle,
    b.price AS BookPrice,
    b.year_published AS PublicationYear,
    a.name AS AuthorName,
    g.name AS Genre,
    s.amount_sales AS SalesAmount,
    RANK() OVER (PARTITION BY b.genre_id ORDER BY s.amount_sales DESC) AS SalesRank
FROM books AS b
JOIN sales AS s ON b.id = s.book_id
JOIN authors AS a ON b.author_id = a.id
JOIN genres AS g ON b.genre_id = g.id;
```

Рисунок 23. Пример использования ранжирующей функции RANK

BookID	BookTitle	BookPrice	PublicationYear	AuthorName	Genre	SalesAmount	SalesRank
14	The Running Man	9.99	1982	Stephen King	Romance	654	1
4	The Great Gatsby	14.99	1925	F. Scott Fitzgerald	Romance	98	2
2	War and Peace	19.99	1869	Leo Tolstoy	Historical Fiction	176	1
10	The Catcher in the Rye	9.99	1951	J.D. Salinger	Classics	741	1
1	Pride and Prejudice	12.99	1813	Jane Austen	Classics	153	2
8	Murder on the Orient Express	11.99	1934	Agatha Christie	Mystery & Thriller	98	1
3	The Old Man and the Sea	9.99	1952	Ernest Hemingway	Mystery & Thriller	68	2
11	Harry Potter and the Deathly Hallows	22.99	2007	J.K. Rowling	Fantasy	223	1
5	Harry Potter and the Philosopher's Stone	8.99	1997	J.K. Rowling	Fantasy	145	2
6	A Game of Thrones	16.99	1996	George R.R. Martin	Science Fiction	299	1
13	The Dark Tower: The Gunslinger	14.99	1982	Stephen King	Science Fiction	229	2
12	The Green Mile	19.99	1996	Stephen King	Horror	512	1
7	The Shining	10.99	1977	Stephen King	Horror	122	2
9	The Da Vinci Code	13.99	2003	Dan Brown	Young Adult	126	1

Рисунок 24. Результат использования

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

2.3) DENSE_RANK()

```
SELECT
    b.id AS BookID,
    b.title AS BookTitle,
    b.price AS BookPrice,
    s.amount_sales AS SalesAmount,
    DENSE_RANK() OVER (ORDER BY s.amount_sales DESC) AS DenseRank
FROM books AS b
JOIN sales AS s ON b.id = s.book_id;
```

Рисунок 25. Пример использования ранжирующей функции DENSE_RANK

BookID	BookTitle	BookPrice	SalesAmount	DenseRank
10	The Catcher in the Rye	9.99	741	1
14	The Running Man	9.99	654	2
12	The Green Mile	19.99	512	3
6	A Game of Thrones	16.99	299	4
13	The Dark Tower: The Gunslinger	14.99	229	5
11	Harry Potter and the Deathly Hallows	22.99	223	6
2	War and Peace	19.99	176	7
1	Pride and Prejudice	12.99	153	8
5	Harry Potter and the Philosopher's Stone	8.99	145	9
9	The Da Vinci Code	13.99	126	10
7	The Shining	10.99	122	11
4	The Great Gatsby	14.99	98	12
8	Murder on the Orient Express	11.99	98	12
3	The Old Man and the Sea	9.99	68	13

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.

2.4) NTILE(*)

```
SELECT
    b.id AS BookID,
    b.title AS BookTitle,
    b.price AS BookPrice,
    NTILE(4) OVER (ORDER BY b.price) AS PriceGroup
FROM books AS b;
```

Рисунок 27. Пример использования ранжирующей функции NTILE

BookID	BookTitle	BookPrice	PriceGroup
5	Harry Potter and the Philosopher's Stone	8.99	1
3	The Old Man and the Sea	9.99	1
10	The Catcher in the Rye	9.99	1
14	The Running Man	9.99	1
7	The Shining	10.99	2
8	Murder on the Orient Express	11.99	2
1	Pride and Prejudice	12.99	2
9	The Da Vinci Code	13.99	2
4	The Great Gatsby	14.99	3
13	The Dark Tower: The Gunslinger	14.99	3
6	A Game of Thrones	16.99	3
2	War and Peace	19.99	4
12	The Green Mile	19.99	4
11	Harry Potter and the Deathly Hallows	22.99	4

Рисунок 28. Результат использования

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

3. 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.

3.1) LAG(*) и LEAD(*)

```
SELECT
    o.id AS OrderID,
    c.name AS CustomerName,
    b.title AS BookTitle,
    o.order_date AS OrderDate,
    LAG(b.title) OVER (PARTITION BY o.customer_id ORDER BY o.order_date) AS PreviousBookTitle
FROM orders AS o
JOIN customers AS c ON o.customer_id = c.id
JOIN books AS b ON o.book_id = b.id;
```

Рисунок 29. Пример использования функции смещения LAG

OrderID	CustomerName	BookTitle	OrderDate	PreviousBookTitle
1	John Smith	Pride and Prejudice	2021-05-01	NULL
22	John Smith	Harry Potter and the Deathly Hallows	2021-05-03	Pride and Prejudice
2	Jane Doe	War and Peace	2021-05-02	NULL
23	Jane Doe	The Dark Tower: The Gunslinger	2021-05-03	War and Peace
11	Jane Doe	The Old Man and the Sea	2021-05-10	The Dark Tower: The Gunslinger
12	Jane Doe	The Great Gatsby	2021-05-10	The Old Man and the Sea
3	Robert Johnson	The Old Man and the Sea	2021-05-03	NULL
13	Robert Johnson	Pride and Prejudice	2021-05-10	The Old Man and the Sea
4	Emily Davis	The Great Gatsby	2021-05-04	NULL
14	Emily Davis	Harry Potter and the Philosopher's Stone	2021-05-07	The Great Gatsby
21	Michael Brown	Harry Potter and the Deathly Hallows	2021-05-03	NULL
5	Michael Brown	Harry Potter and the Philosopher's Stone	2021-05-05	Harry Potter and the Deathly H...
17	Michael Brown	A Game of Thrones	2021-05-07	Harry Potter and the Philosoph...
19	Sarah Wilson	The Old Man and the Sea	2021-05-03	NULL
6	Sarah Wilson	A Game of Thrones	2021-05-06	The Old Man and the Sea
18	David Lee	The Green Mile	2021-05-03	NULL
7	David Lee	The Shining	2021-05-07	The Green Mile
20	Karen Taylor	The Catcher in the Rye	2021-05-03	NULL
15	Karen Taylor	A Game of Thrones	2021-05-07	The Catcher in the Rye
8	Karen Taylor	Murder on the Orient Express	2021-05-08	A Game of Thrones
16	Mark Anderson	A Game of Thrones	2021-05-07	NULL
9	Mark Anderson	The Da Vinci Code	2021-05-09	A Game of Thrones
10	Lisa Jackson	The Catcher in the Rye	2021-05-10	NULL

Рисунок 30. Результат использования

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

3.2) FIRST_VALUE() и LAST_VALUE()

```
WITH FirstBook AS (
    SELECT
        c.id AS CustomerID,
        c.name AS CustomerName,
        b.title AS FirstBookTitle,
        o.order_date AS FirstOrderDate,
        FIRST_VALUE(b.title) OVER (PARTITION BY c.id ORDER BY o.order_date ASC) AS FirstOrderedBook
    FROM customers AS c
    JOIN orders AS o ON c.id = o.customer_id
    JOIN books AS b ON o.book_id = b.id
)

SELECT
    CustomerID,
    CustomerName,
    FirstBookTitle,
    FirstOrderDate
FROM FirstBook
WHERE FirstOrderedBook = FirstBookTitle;
```

Рисунок 31. Пример использования функции смещения FIRST_VALUE

CustomerID	CustomerName	FirstBookTitle	FirstOrderDate
1	John Smith	Pride and Prejudice	2021-05-01
2	Jane Doe	War and Peace	2021-05-02
3	Robert Johnson	The Old Man and the Sea	2021-05-03
4	Emily Davis	The Great Gatsby	2021-05-04
5	Michael Brown	Harry Potter and the Deathly Hallows	2021-05-03
6	Sarah Wilson	The Old Man and the Sea	2021-05-03
7	David Lee	The Green Mile	2021-05-03
8	Karen Taylor	The Catcher in the Rye	2021-05-03
9	Mark Anderson	A Game of Thrones	2021-05-07
10	Lisa Jackson	The Catcher in the Rye	2021-05-10

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.