МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

Лекция 4.
Оконные функции для анализа данных

Оконные функции для анализа данных

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Цель:

- Научиться писать оконные функции
- Использовать оконные функции для вычисления статистики
- Проанализировать данные о продажах с помощью **оконных функций** и **оконных фреймов**.

Введение в оконные функции

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Необходимо сохранить отдельные строки, а также получить итоговое значение.

Чтобы сделать это, используем новый набор функций под названием **window functions**, которые **могут вычислять совокупную статистику, сохраняя при этом отдельные строки**.

Эти функции позволяют относительно легко вычислять новые типы статистики в SQL, такие как **ранги** и **скользящие средние.**

Оконные функции

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Агрегатные функции позволяют нам брать много строк и преобразовывать эти строки в одно число. Например, функция COUNT принимает строки таблицы и возвращает количество имеющихся в ней строк.

Однако иногда требуется вычислять несколько строк, но при этом сохранять все строки после вычисления.

Например, ранжировать каждого пользователя в порядке, соответствующем времени, когда он стал клиентом, при этом самый ранний клиент получил 1-е место, второй по рангу клиент получил 2-е место и так далее. Можем получить всех клиентов, используя следующий запрос:

```
SELECT *
FROM customers
ORDER BY date_added;
```

Оконные функции

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Можем упорядочивать клиентов от самых ранних до самых последних, но **не можем присвоить им номер**.

Используем **агрегатную функцию**, чтобы получить даты и упорядочить их таким образом:

```
SELECT date_added, COUNT(*)
FROM customers
GROUP BY date_added
ORDER BY date_added;
```

date_added	count
2010-03-15 00:00:00.000	11
2010-03-16 00:00:00.000	13
2010-03-17 00:00:00.000	12
2010-03-18 00:00:00.000	19
2010-03-19 00:00:00.000	23
2010-03-20 00:00:00.000	16
2010-03-21 00:00:00.000	20
2010-03-22 00:00:00.000	14
2010-03-23 00:00:00.000	11
2010-03-24 00:00:00.000	21
2010-03-25 00:00:00.000	15
2010-03-26 00:00:00.000	17
2010-03-27 00:00:00.000	11
2010-03-28 00:00:00.000	21

Хотя это дает даты, оно избавляется от остальных столбцов и по-прежнему не предоставляет информации о ранге. Что мы можем сделать? Вот тут-то и вступают в игру **оконные функции**.

Основы оконных функций

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Базовый синтаксис оконной функции:

```
SELECT {columns},
{window_func} OVER (PARTITION BY {partition_key}
ORDER BY {order_key})
FROM table1;
```

Основы оконных функций

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Базовый синтаксис оконной функции:

```
SELECT {columns},
{window_func} OVER (PARTITION BY {partition_key}
ORDER BY {order_key})
FROM table1;
```

Где **{columns}** - столбцы для извлечения из таблиц для запроса,
{window_func} – оконная функция, которую хотим использовать,
{partition_key} - столбец или столбцы, которые хотим разбить на разделы,
{order_key} - столбец или столбцы, которые хотим order by,
table1 - это таблица или объединенные таблицы, из которых хотим извлечь данные.
Ключевое слово **OVER** указывает, с чего начинается определение окна.

Основы оконных функций

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Все агрегатные функции могут использоваться в качестве оконных функций.

Используем **COUNT(*)** в следующем запросе:

```
SELECT customer_id, title, first_name, last_name, gender,
COUNT(*) OVER () as total_customers
FROM customers
ORDER BY customer_id;
```

customer_id	title	first_name	last_name	gender	total_customers
1		Arlena	Riveles	F	50000
2	Dr	Ode	Stovin	M	50000
3		Braden	Jordan	M	50000
4		Jessika	Nussen	F	50000
5		Lonnie	Rembaud	F	50000
6		Cortie	Locksley	M	50000
7		Wood	Kennham	M	50000
8		Rutger	Humblestone	M	50000
9		Melantha	Tibb	F	50000
10	Ms	Barbara-anne	Gowlett	F	50000

Основы оконных функций

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Запрос возвращает **title, first_name и last_name**, точно так же, как обычный запрос **SELECT**. Однако теперь появился новый столбец под названием **total_customers**. Этот столбец содержит количество пользователей, которые будут созданы с помощью следующего запроса:

```
SELECT COUNT(*)
FROM customers;
```

ustomer_id	title	first_name	last_name	gender	total_customers
1		Arlena	Riveles	F	50000
2	Dr	Ode	Stovin	M	50000
3		Braden	Jordan	M	50000
4		Jessika	Nussen	F	50000
5		Lonnie	Rembaud	F	50000
6		Cortie	Locksley	M	50000
7		Wood	Kennham	M	50000
8		Rutger	Humblestone	M	50000
9		Melantha	Tibb	F	50000
10	Ms	Barbara-anne	Gowlett	F	50000

запрос вернул как все строки, так и количество (*) в запросе, вместо того, чтобы просто возвращать количество, как это сделала бы обычная агрегатная функция

Основы оконных функций

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

total_customers теперь изменили количество на одно из двух значений, **24 956** или **25 044**. Эти подсчеты являются подсчетами для каждого пола.

customer_id	title	first_name	last_name	gender	total_customers
1		Arlena	Riveles	F	25044
2	Dr	Ode	Stovin	M	24956
3		Braden	Jordan	M	24956
4		Jessika	Nussen	F	25044
5		Lonnie	Rembaud	F	25044

Их можно получить через запрос:

```
SELECT gender, COUNT(*)
FROM customers
GROUP BY 1
```

gender	count
M	24956
F	25044

Основы оконных функций

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Используем **ORDER BY**

```
SELECT customer_id, title, first_name, last_name, gender,
COUNT(*) OVER (ORDER BY customer_id) as total_customers
FROM customers
ORDER BY customer_id;
```

customer_id	title	first_name	last_name	gender	total_customers
1		Arlena	Riveles	F	1
2	Dr	Ode	Stovin	M	2
3		Braden	Jordan	M	3
4		Jessika	Nussen	F	4
5		Lonnie	Rembaud	F	5

Основы оконных функций

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Когда используем **оконную функцию**, запрос создает "окно" над таблицей, в которой проводится подсчет.

PARTITION BY работает аналогично **GROUP BY**, разделяя набор данных на несколько групп.

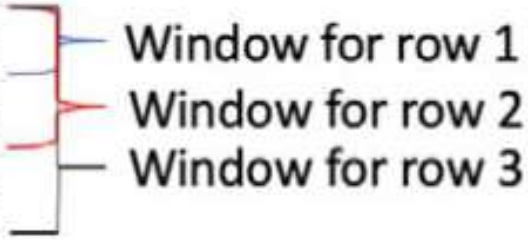
Для каждой группы создается окно. Если **ORDER BY** не указан, предполагается, что окно представляет собой всю группу. Однако, когда указан **ORDER BY**, строки в группе упорядочиваются в соответствии с ним, и для каждой строки создается окно, к которому применяется функция.

Основы оконных функций

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Окна для **customers**, использующих **COUNT(*)**, упорядоченные по запросу окна **customer_id**

customer_id bigint	title text	first_name text	last_name text	gender text	total_customers bigint
1	[null]	Arlena	Riveles	F	1
2	Dr	Ode	Stovin	M	2
3	[null]	Braden	Jordan	M	3



Основы оконных функций

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Что происходит, если объединить PARTITION BY и ORDER BY?

```
SELECT customer_id, title, first_name, last_name, gender,
COUNT(*) OVER (PARTITION BY gender ORDER BY customer_id) as total_customers
FROM customers
ORDER BY customer_id;
```

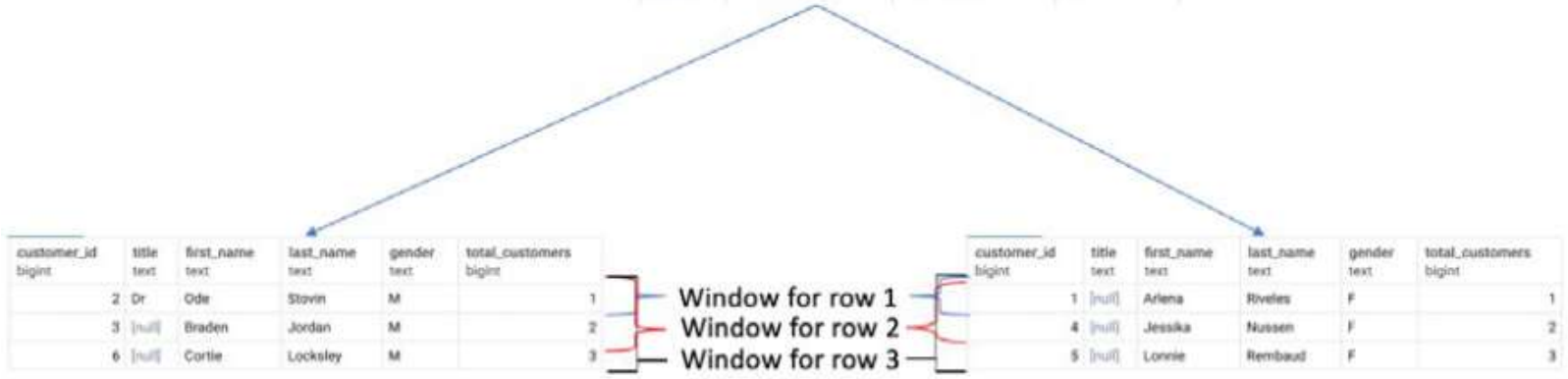
customer_id	title	first_name	last_name	gender	total_customers
1		Arlena	Riveles	F	1
2	Dr	Ode	Stovin	M	1
3		Braden	Jordan	M	2
4		Jessika	Nussen	F	2
5		Lonnie	Rembaud	F	3
6		Cortie	Locksley	M	3
7		Wood	Kennham	M	4
8		Rutger	Humblestone	M	5
9		Melantha	Tibb	F	4
10	Ms	Barbara-anne	Gowlett	F	5
11	Mrs	Urbano	Middlehurst	M	6
12	Mr	Tyne	Duggan	F	6
13		Gannon	Braker	M	7
14		Derry	Lyburn	M	8
15		Nichols	Espinay	M	9
16		Donalt	Huffadine	M	10
17		Terrill	Qualtrough	M	11
18		Mareah	Edgell	F	7

Основы оконных функций

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Три ключевых слова, **OVER()**, **PARTITION BY** и **ORDER BY**, создают предпосылки для пониманию оконных функций

customer_id bigint	title text	first_name text	last_name text	gender text
1	[null]	Arlena	Riveles	F
2	Dr	Ode	Stovin	M
3	[null]	Braden	Jordan	M
4	[null]	Jessika	Nussen	F
5	[null]	Lonnie	Rembaud	F
6	[null]	Cortie	Locksley	M



Упражнение 1. Анализ показателей заполнения клиентских данных с течением времени

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

В течение последних 6 месяцев компания экспериментировала с различными функциями, чтобы побудить людей заполнять все поля в форме клиента, особенно их **улицу**. Чтобы проанализировать эти данные, компания хотела бы получить общее количество пользователей, заполнивших свой адрес за прошедшее время. Напишите запрос для получения этих результатов.

Упражнение 1. Анализ показателей заполнения клиентских данных с течением времени

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Решение

1. Откройте **SQL**-клиент и подключитесь к базе данных **sqlda**

2. Используйте оконные функции и напишите запрос, который вернет информацию о клиенте и о том, сколько человек заполнили свою улицу. Кроме того, упорядочить список в соответствии с датой.

Упражнение 1. Анализ показателей заполнения клиентских данных с течением времени

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Решение

```
SELECT customer_id, street_address, date_added::DATE,
COUNT(CASE WHEN street_address IS NOT NULL THEN customer_id ELSE NULL END)
OVER (ORDER BY date_added::DATE) as total_customers_filled_street
FROM customers
ORDER BY date_added;
```

Теперь у нас есть каждый клиент, найденный по дате регистрации, можем видеть количество людей, заполнивших поле "**Улица**".

customer_id	street_address	date_added	total_customers_filled_street
35683	1 Cordelia Crossing	2010-03-15	10
30046	13961 Steensland Trail	2010-03-15	10
17099	130 Marcy Crossing	2010-03-15	10
2625	0353 Iowa Road	2010-03-15	10
30555	294 Quincy Hill	2010-03-15	10
18685	86 Michigan Junction	2010-03-15	10
13390	38463 Forest Dale Way	2010-03-15	10
7486	61 Village Crossing	2010-03-15	10
6173	79865 Hagan Terrace	2010-03-15	10
12484		2010-03-15	10
48307	8487 Warbler Plaza	2010-03-15	10
48229	943 Cody Trail	2010-03-16	22
42776	6010 Carey Drive	2010-03-16	22
46277	5799 Thackeray Crossing	2010-03-16	22
8571	39223 Lunder Street	2010-03-16	22

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

Ключевые слова

WINDOW Keyword

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Вычисление текущего общего числа клиентов и количества клиентов с названием в каждом поле (нет пустых полей)

```
SELECT customer_id, title, first_name, last_name, gender,
COUNT(*) OVER (PARTITION BY gender ORDER BY customer_id) as total_customers,
SUM(CASE WHEN title IS NOT NULL THEN 1 ELSE 0 END)
OVER (PARTITION BY gender ORDER BY customer_id) as total_customers_title
FROM customers
ORDER BY customer_id;
```

WINDOW Keyword

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Вычисление текущего общего числа клиентов и количества клиентов с названием в каждом поле (нет пустых полей)

customer_id	title	first_name	last_name	gender	total_customers	total_customers_title
1		Arlena	Riveles	F	1	0
2	Dr	Ode	Stovin	M	1	1
3		Braden	Jordan	M	2	1
4		Jessika	Nussen	F	2	0
5		Lonnie	Rembaud	F	3	0
6		Cortie	Locksley	M	3	1
7		Wood	Kennham	M	4	1
8		Rutger	Humblestone	M	5	1
9		Melantha	Tibb	F	4	0
10	Ms	Barbara-anne	Gowlett	F	5	1
11	Mrs	Urbano	Middlehurst	M	6	2
12	Mr	Tyne	Duggan	F	6	2
13		Gannon	Braker	M	7	2

WINDOW Keyword

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Есть ли способ, которым мы можем это упростить?

Ответ - **да**

```
SELECT customer_id, title, first_name, last_name, gender,
COUNT(*) OVER w as total_customers,
SUM(CASE WHEN title IS NOT NULL THEN 1 ELSE 0 END)
OVER w as total_customers_title
FROM customers
WINDOW w AS (PARTITION BY gender ORDER BY customer_id)
ORDER BY customer_id;
```

Не пришлось писать длинный запрос **PARTITION BY** и **ORDER BY** для каждой **оконной функции**. Вместо этого создали псевдоним с определенным окном **w**.

Статистика с оконными функциями

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Функция	Пояснение
ROW_NUMBER()	Задает текущий номер строки в разделе
RANK()	Присваивает рейтинг внутри раздела на основе ORDER BY, создавая пропуски при наличии связей (например, если строка 1 и строка 2 связаны по 1, то строка 3 получит рейтинг 3).
DENSE_RANK()	Присваивает рейтинг внутри раздела на основе ORDER BY, не создавая пропусков при наличии связей (например, если строка 1 и строка 2 оба связаны для 1. тогда строка 3 получит рейтинг 2).
NTILE(num_buckets)	Назначает n-разметку в разделе на основе ORDER BY, где n определяется целым числом num_buckets.
LAG(column1, offset)	Возвращает значение столбца 1, представляющее собой целочисленное смещение строк перед текущей строкой на основе ORDER BY.
LEAD(column1, offset)	Возвращает значение столбца 1, представляющее собой целочисленное смещение строк после текущей строки на основе ORDER BY.

Упражнение 2. Порядок приема на работу

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Задание

Компания хочет повысить по должности продавцов в своих региональных дилерских центрах, для этого необходимо узнать стаж работы сотрудников на занимаемой должности. Написать запрос, который ранжирует продавцов в соответствии с датой их приема для каждого дилерского центра

Упражнение 2. Порядок приема на работу

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Решение

1. Откройте клиент **SQL** и подключитесь к базе данных **sqlda**.
2. Вычислите ранг для каждого продавца, при этом ранг 1 достается первому принятому на работу, 2 — второму и т. д., используя функцию **RANK()**:

```
SELECT *,
RANK() OVER (PARTITION BY dealership_id
ORDER BY hire_date)
FROM salespeople
WHERE termination_date IS NULL;
```

Упражнение 2. Порядок приема на работу

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Решение

1. Откройте клиент **SQL** и подключитесь к базе данных **sqlda**.
2. Вычислите ранг для каждого продавца, при этом ранг 1 достается первому принятому на работу, 2 — второму и т. д., используя функцию **RANK():**

salesperson_id	dealership_id	title	first_name	last_name	suffix	username	gender	hire_date	termination_date	rank
65	1		Dukie	Oxteby		doxteby1s	Male	2015-01-24 00:00:00.000		1
74	1		Marcos	Spong		mspong21	Male	2015-03-18 00:00:00.000		2
60	1		Eveleen	Mace		emace1n	Female	2015-07-15 00:00:00.000		3
87	1		Quent	Wogden		qwogden2e	Male	2015-08-17 00:00:00.000		4
98	1		Englebert	Loraine		eloraine2p	Male	2016-01-23 00:00:00.000		5
31	1		Lelia	Sheriff		lsheriffu	Female	2016-06-18 00:00:00.000		6
168	1		Sheff	McCoughan		smccoughan4n	Male	2016-07-22 00:00:00.000		7
49	1		Nadia	Rennick		nrennick1c	Female	2016-07-24 00:00:00.000		8
10	1		Jereme	Onele		jonele9	Male	2016-08-15 00:00:00.000		9
7	1		Granville	Fidell		gfidell6	Male	2017-06-17 00:00:00.000		10

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

Window Frame

Window Frame

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Запрос **оконной функции** с использованием предложения **оконного фрейма** будет выглядеть следующим образом:

```
SELECT {columns},
{window_func} OVER (PARTITION BY {partition_key}
ORDER BY {order_key} {rangeorrows}
BETWEEN {frame_start} AND {frame_end})
FROM {table1};
```

Window Frame

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

```
SELECT {columns},
{window_func} OVER (PARTITION BY {partition_key}
ORDER BY {order_key} {rangeorrows}
BETWEEN {frame_start} AND {frame_end})
FROM {table1};
```

{columns} — это столбцы, которые нужно извлечь из таблиц для запроса,
{window_func} — это оконная функция, которую хотим использовать,
{partition_key} — это столбец или столбцы, по которым хотим разбить,
{order_key } — столбец или столбцы, по которым необходимо упорядочить,
{rangeorrows} — либо ключевое слово **RANGE**, либо ключевое слово **ROWS**,
{frame_start} — ключевое слово, указывающее, где начинать фрейм,
{frame_end} — ключевое слово, указывающее, где заканчивать оконный фрейм,
{table1} — это таблица или объединенные таблицы, из которых извлекаем данные.

Window Frame

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Одно различие, которое следует учитывать, — это разница между использованием **RANGE** или **ROW** в предложении фрейма.

ROW относятся к фактическим строкам и берут строки до и после текущей строки для вычисления значений.

RANGE отличается, когда две строки имеют одинаковые значения на основе предложения **ORDER BY**, используемого в окне.

Если текущая строка, используемая в оконной функции, имеет то же значение в предложении **ORDER BY**, что и одна или несколько строк, то все эти строки будут добавлены в фрейм.

Window Frame

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Значения, которые могут принимать **{frame_start}** и **{frame_end}.**

UNBOUNDED PRECEDING: ключевое слово, которое при использовании для {frame_start} относится к первой записи раздела, а при использовании для {frame_end} относится к последней записи раздела.

{offset} **PRECEDING**: ключевое слово, относящееся к целочисленным {offset} строкам или диапазонам перед текущей строкой.

CURRENT ROW: текущая строка

{offset} **FOLLOWING**: ключевое слово, относящееся к целочисленным {offset}строкам или диапазонам после текущей строки.

Window Frame

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Используя оконный фрейм, можно рассчитать различную полезную статистику. Одной из таких полезных статистических данных является **скользящее среднее**.

Скользящее среднее — это просто среднее значение статистики в заданном временном окне. Допустим, рассчитать 7-дневное скользящее среднее продаж с течением времени для компании. Этот расчет может быть выполнен с помощью следующего запроса:

Window Frame

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

```
WITH daily_sales as (
SELECT sales_transaction_date::DATE,
SUM(sales_amount) as total_sales
FROM sales
GROUP BY 1
),
moving_average_calculation_7 AS (
SELECT sales_transaction_date, total_sales,
AVG(total_sales) OVER (ORDER BY sales_transaction_date ROWS BETWEEN 7
PRECEDING and CURRENT ROW) AS sales_moving_average_7,
ROW_NUMBER() OVER (ORDER BY sales_transaction_date) as row_number
FROM daily_sales
ORDER BY 1)
SELECT sales_transaction_date,
CASE WHEN row_number>=7 THEN sales_moving_average_7 ELSE NULL END
AS sales_moving_average_7
FROM moving_average_calculation_7;
```

Window Frame

Причина, по которой первые 7 строк пустые, заключается в том, что 7-дневное скользящее среднее определяется только при наличии информации за 7 дней, а расчет окна по-прежнему будет вычислять значения для первых 7 дней, используя первые несколько дней.

sales_transaction_date	sales_moving_average_7
2010-03-10	
2010-03-12	
2010-03-15	
2010-03-17	
2010-03-18	
2010-03-19	
2010-03-21	394.2758571428571
2010-03-23	394.9901249999999
2010-03-24	399.99
2010-03-25	399.99
2010-03-29	449.98875000000004
2010-04-01	544.986375
2010-04-02	594.985125
2010-04-03	594.9851249999999
2010-04-04	589.9852500000001
2010-04-05	589.98525
2010-04-06	639.9839999999999
2010-04-07	689.9827499999999
2010-04-08	634.984125
2010-04-09	584.985375
2010-04-10	579.9855
2010-04-11	749.9812499999999
2010-04-12	799.98

Упражнение 3. Обеденная мотивация команды

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Задание

Чтобы повысить эффективность продаж, отдел продаж решил покупать обед для всех продавцов в компании каждый раз, когда они превышают цифру лучшего ежедневного общего дохода, достигнутого за последние 30 дней. Напишите запрос, который выводит общий объем продаж в долларах за определенный день и цель, которую продавцы должны превзойти на этот день, начиная с 1 января 2019 года:

Упражнение 3. Обеденная мотивация команды

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Решение

1. Откройте клиент **SQL** и подключитесь к базе данных **sqlda**.
2. Рассчитайте общий объем продаж за заданный день и цель, используя следующий запрос:

Упражнение 3. Обеденная мотивация команды

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Решение

```
WITH daily_sales as (
SELECT sales_transaction_date::DATE,
SUM(sales_amount) as total_sales
FROM sales
GROUP BY 1
),
sales_stats_30 AS (
SELECT sales_transaction_date, total_sales,
MAX(total_sales) OVER (ORDER BY sales_transaction_date ROWS BETWEEN 30
PRECEDING and 1 PRECEDING)
AS max_sales_30
FROM daily_sales
ORDER BY 1)
SELECT sales_transaction_date,
total_sales,
max_sales_30
FROM sales_stats_30
WHERE sales_transaction_date>='2019-01-01';
```

Упражнение 3. Обеденная мотивация команды

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Решение

Обратите внимание на использование оконного фрейма от 30 **PRECEDING** до 1 **PRECEDING**, чтобы удалить текущую строку из расчета.

sales_transaction_date	total_sales	max_sales_30
2019-01-01	87694.84400000003	316464.8469999999
2019-01-02	76149.854	316464.8469999999
2019-01-03	161269.80899999998	316464.8469999999
2019-01-04	193209.91199999998	316464.8469999999
2019-01-05	49469.76999999998	316464.8469999999
2019-01-06	96319.835	316464.8469999999
2019-01-07	42239.837	316464.8469999999
2019-01-08	101729.748	316464.8469999999
2019-01-09	118634.90200000003	316464.8469999999
2019-01-10	100089.78000000004	316464.8469999999
2019-01-11	183209.871	283849.83999999997
2019-01-12	125189.827	283849.83999999997
2019-01-13	213809.88299999994	265439.875
2019-01-14	265279.8219999999	265439.875
2019-01-15	130829.84500000004	265439.875
2019-01-16	212354.77899999995	265439.875
2019-01-17	190779.83899999995	265439.875
2019-01-18	77689.858	265439.875
2019-01-19	79629.83	265439.875
2019-01-20	47969.79999999999	265439.875
2019-01-21	9349.858	265439.875
2019-01-22	12789.809	265439.875