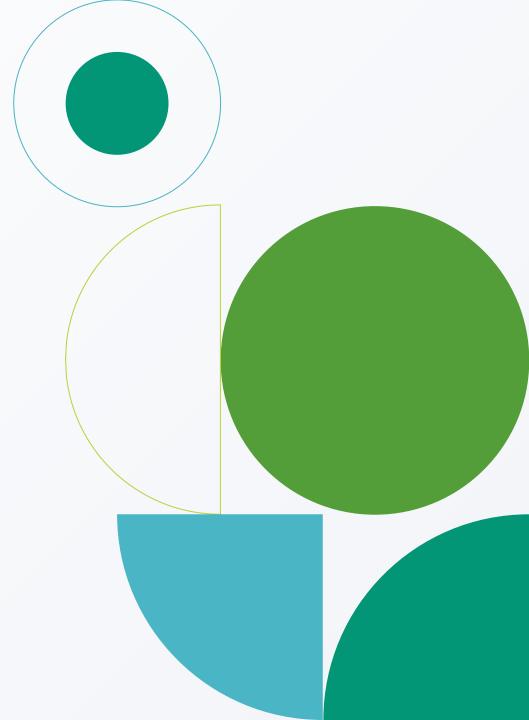
JIEKUNA

Организация обработки больших дау дах их



Agenda (повестка дня)

- >
- Агрегатные функции
- Фильтрация данных
- Условные выражения







Агрегатные функции SQL, фильтрация

Count	The number of rows containing non-null values
Max	The maximum attribute value encountered in a given column
Min	The minimum attribute value encountered in a given column
SUM	The sum of all values for a given column
AVG	The arithmetic mean (average) for a specified column.

Нафатные функции выполняют вычисления над набором строк и возвращают одну строку. PostgreSQL предоставляет все стандартные агрегатные функции SQL следующим образом:

Примеры агрегатных функций PostgreSQL

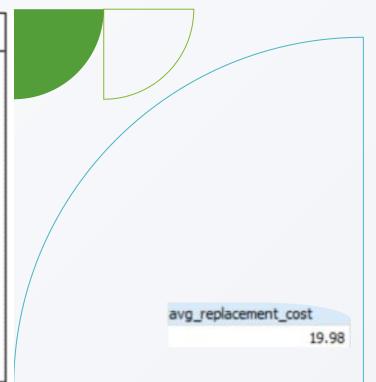
Давайте для демонстрации воспользуемся таблицей фильмов в образце базы данных. Примеры функций AVG() В следующем операторе функция AVG() используется для расчета средней стоимости замены всех пленок:

Обратите внимание, что функция ROUND() использовалась для округления результата до двух знаков после запятой.



film

* film_id
title
description
release_year
language_id
rental_duration
rental_rate
length
replacement_cost
rating
last_update
special_features
fulltext



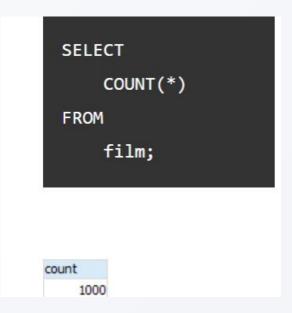


примеры функции COUNT ()

Лекция 4

Чтобы получить количество фильмов, вы используете функцию COUNT(*) следующим образом:

* film * film_id title description release_year language_id rental_duration rental_rate length replacement_cost rating last_update special_features fulltext



```
SELECT
   column_1,
   column_2,
   ...,
   aggregate_function(column_3)
FROM
   table_name
GROUP BY
   column_1,
   column_2,
   ...;
```

Предложение GROUP BY

- Предложение GROUP BY делит строки, возвращаемые инструкцией SELECT, на группы.
- Для каждой группы вы можете применить агрегатную функцию, например SUM(), чтобы вычислить сумму элементов, или COUNT(), чтобы получить количество элементов в группах.
- Следующий оператор иллюстрирует основной синтаксис предложения GROUP BY:

SELECT

customer id

FROM

payment

GROUP BY

customer_id;

4	customer_id smallint
1	184
2	87
3	477
4	273
5	550
6	51
7	394
8	272
9	70

Использование GROUP ВУ без агрегатной

СВы можете использовать предложение GROUP BY без применения агрегатной функции. Следующий запрос получает данные из таблицы платежей и группирует результат по идентификатору клиента.

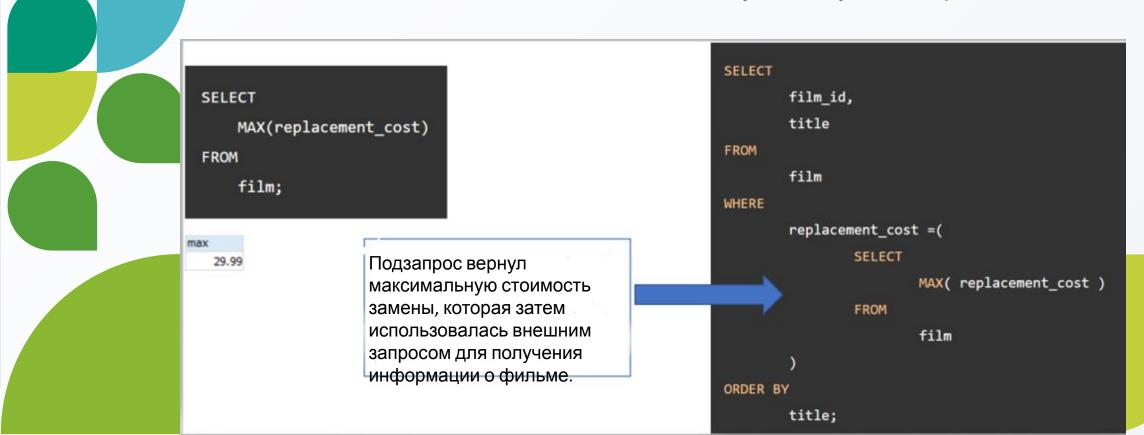
В этом случае GROUP BY работает как предложение DISTINCT, которое удаляет повторяющиеся строки из набора результатов.



Примеры функции МАКС()

Следующий оператор возвращает максимальную стоимость замены пленок.

Чтобы получить пленки с максимальной стоимостью замены, используйте следующий запрос:





Пример использования GROUP BY с функцией

S фрумение GROUP BY полезно, когда оно используется вместе с агрегатной функцией.

- Например, чтобы выбрать общую сумму, которую заплатил каждый клиент, вы используете предложение GROUP BY, чтобы разделить строки в таблице платежей на группы, сгруппированные по идентификатору клиента. Для каждой группы вы рассчитываете общие суммы с помощью функции SUM().
- В следующем запросе используется предложение GROUP BY для получения общей суммы, выплаченной каждому клиенту:

customer_id,
SUM (amount)
FROM
payment
GROUP BY
customer_id;

4	customer_id smallint	sum numeric
1	184	80.80
2	87	137.72
3	477	106.79
4	273	130.72
5	550	151.69
6	51	123.70
7	394	77.80

Примеры функции MIN()

• В следующем примере функция MIN() используется для возврата минимальной стоимости замены пленок:

```
SELECT

MIN(replacement_cost)

FROM

film;
```

9.99

Чтобы получить пленки с минимальной стоимостью замены, используйте следующий запрос:

```
SELECT
        film id,
        title
FROM
        film
WHERE
        replacement_cost =(
                 SELECT
                         MIN( replacement cost )
                 FROM
                         film
ORDER BY
        title;
```

Примеры функции

В следующем операторе функция SUM() используется для расчета общей продолжительности фильмов,

сгруппированных по рейтингу:

```
rating,
        SUM( rental_duration )
FROM
        film
GROUP BY
        rating
ORDER BY
        rating;
```

rating	sum
G	861
PG	986
PG-13	1127
R	931
NC-17	1080

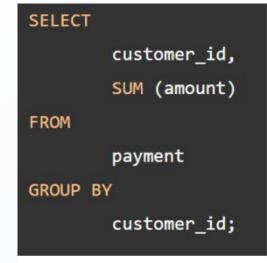
Предложение HAVING

- Предложение HAVING задает условие поиска для группы или агрегата.
- Предложение HAVING часто используется вместе с предложением GROUP BY для фильтрации групп или агрегатов на основе заданного условия.
- Следующий оператор иллюстрирует основной синтаксис предложения HAVING:

```
SELECT
        column1,
        aggregate_function (column2)
FROM
        table_name
GROUP BY
        column1
HAVING
        condition;
```

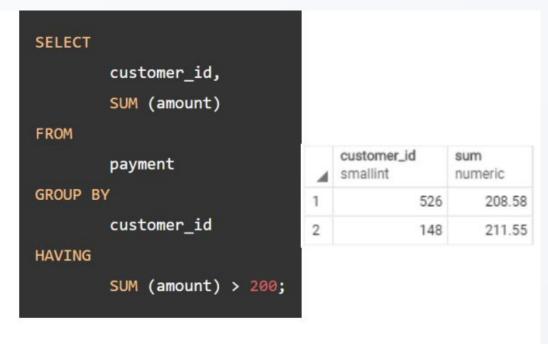
Использование предложения HAVING с примером функции SUM

В следующем запросе используется предложение GROUP BY с функцией SUM(), чтобы найти общую сумму каждого клиента:



4	customer_id smallint	sum numeric
1	184	80.80
2	87	137.72
3	477	106.79
4	273	130.72

Следующий оператор добавляет предложение HAVING для выбора единственных клиентов, которые потратили более 200:



Выражение CASE

- Выражение CASE такое же, как выражение IF/ELSE в других языках программирования. Это позволяет вам добавлять к запросу логику if-else, чтобы сформировать мощный запрос.
- Поскольку CASE это выражение, вы можете использовать его в любых местах, где может использоваться выражение, например, SELECT, WHERE, GROUP BY и предложение HAVING.
- Выражение CASE имеет две формы: общую и простую.

Общее выражение CASE

• Следующее иллюстрирует общую форму оператора CASE:

```
CASE

WHEN condition_1 THEN result_1

WHEN condition_2 THEN result_2

[WHEN ...]

[ELSE else_result]

END
```

• В этом синтаксисе каждое условие (условие_1, условие_2...) представляет собой логическое выражение, которое возвращает либо истину, либо ложь. Когда условие оценивается как ложное, выражение CASE оценивает следующее условие сверху вниз, пока не найдет условие, которое оценивается как истинное. Если условие оценивается как истинное, выражение CASE возвращает соответствующий результат, следующий за условием. Например, если условие_2 имеет значение true, выражение CASE возвращает результат_2. Кроме того, он немедленно прекращает вычисление следующего выражения. Если все условия оцениваются как ложные, выражение CASE возвращает результат (else_result), следующий за ключевым словом ELSE. Если вы опустите предложение ELSE, выражение CASE вернет NULL.

Общий пример CASE

Предположим, вы хотите маркировать фильмы по их продолжительности, основываясь на следующей логике:

- Если продолжительность менее 50 минут, фильм короткий.
- Если продолжительность больше 50 минут и меньше или равна 120 минутам, фильм считается средним.
- Если продолжительность превышает 120 минут, фильм длинный.

Чтобы применить эту логику, вы можете использовать выражение CASE в инструкции SELECT следующим образом:

```
* film

* film_id
title
description
release_year
language_id
rental_duration
rental_rate
length
replacement_cost
rating
last_update
special_features
fulltext
```

```
SELECT title,
length,
CASE

WHEN length> 0

AND length <= 50 THEN 'Short'
WHEN length > 50

AND length <= 120 THEN 'Medium'
WHEN length> 120 THEN 'Long'
END duration
FROM film
ORDER BY title;
```

4	title character varying (255)	length smallint	duration text
1	Academy Dinosaur	86	Medium
2	Ace Goldfinger	48	Short
3	Adaptation Holes	50	Short
4	Affair Prejudice	117	Medium
5	African Egg	130	Long
6	Agent Truman	169	Long
7	Airplane Sierra	62	Medium
8	Airport Pollock	54	Medium
9	Alabama Devil	114	Medium
10	Aladdin Calendar	63	Medium
11	Alamo Videotape	126	Long
12	Alaska Phantom	136	Long
13	Ali Forever	150	Long
14	Alice Fantasia	94	Medium

Простое выражение CASE

• PostgreSQL предоставляет другую форму выражения CASE, называемую простой формой, следующим образом:

```
CASE expression

WHEN value_1 THEN result_1

WHEN value_2 THEN result_2

[WHEN ...]

ELSE

else_result

END
```

- CASE сначала оценивает выражение и последовательно сравнивает результат с каждым значением (value_1, value_2, ...) в предложениях WHEN, пока не найдет совпадение.
- Как только результат выражения равен значению (значение1, значение2 и т. д.) в предложении WHEN, CASE возвращает соответствующий результат в предложении THEN.
- Если CASE не находит совпадений, он возвращает else_result, следующий за ELSE, или значение NULL, если ELSE недоступно.

Простой пример выражения CASE

• В следующем операторе выражение CASE используется для добавления описания рейтинга к выходным данным:

```
SELECT title,
       rating,
       CASE rating
           WHEN 'G' THEN 'General Audiences'
           WHEN 'PG' THEN 'Parental Guidance Suggested'
           WHEN 'PG-13' THEN 'Parents Strongly Cautioned'
           WHEN 'R' THEN 'Restricted'
           WHEN 'NC-17' THEN 'Adults Only'
       END rating description
FROM film
ORDER BY title;
```

4	title character varying (255)	rating mpaa_rating	rating_description text
1	Academy Dinosaur	PG	Parental Guidance Suggested
2	Ace Goldfinger	G	General Audiences
3	Adaptation Holes	NC-17	Adults Only
4	Affair Prejudice	G	General Audiences
5	African Egg	G	General Audiences
6	Agent Truman	PG	Parental Guidance Suggested
7	Airplane Sierra	PG-13	Parents Strongly Cautioned
8	Airport Pollock	R	Restricted
9	Alabama Devil	PG-13	Parents Strongly Cautioned
10	Aladdin Calendar	NC-17	Adults Only
11	Alamo Videotape	G	General Audiences
12	Alaska Phantom	PG	Parental Guidance Suggested

Спасибо за внимани

Лекция 4