Introduction to SQL

Part-6

Hitesh Kumar Sharma
Instructor, Pluralsight

Proprietary and confidential

Aggregate Functions in PostgreSQL

PLURALSIGHT

Aggregate Functions

In PostgreSQL, aggregate functions perform calculations on a set of values and return a single value. These functions are often used with the SELECT statement to compute aggregated values from a table. Here are some commonly used aggregate functions in PostgreSQL:

COUNT( ): This function counts the number of rows that match a specific condition.

```
SELECT COUNT(column_name) FROM table_name;
```

SUM( ): This function calculates the sum of values in a column.

```
SELECT SUM(column_name) FROM table_name;
```

Proprietary and confidential

Aggregate Functions - Example: COUNT()

QUERY: How many films are in the films table?

APPROACH: Use the COUNT() function to count the number of rows in the film table.

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

RESULTS:

```
+----------+
| COUNT(*) |
+----------+
| 1000     |
+----------+
```

Proprietary and confidential

Aggregate Functions - Example: COUNT()

QUERY: How many distinct ratings are represented in the films table?

APPROACH: Use the COUNT() function combined with DISTINCT to count the number of ratings in the film table.

```
SELECT COUNT(DISTINCT(rating))
FROM film;
```

RESULTS:

```
+-------------------------+
| COUNT(DISTINCT(rating)) |
+-------------------------+
| 5                       |
+-------------------------+
```

Proprietary and confidential

Aggregate Functions - Example: SUM()

QUERY: If I wanted to watch all of the movies in the film catalog, how long would it take?

APPROACH: Use the SUM() function to add up all the length values in the films table.

```
SELECT SUM(length)
FROM film;
```

RESULTS:

```
+---------------+
| SUM(length)   |
+---------------+
| 115272        |
+---------------+
```

Proprietary and confidential

Aggregate Functions

AVG( ): This function calculates the average value of a set of values in a column.

SELECT AVG(column_name) FROM table_name;

MAX( ): This function returns the maximum value from a set of values.

SELECT MAX(column_name) FROM table_name;

MIN( ): This function returns the minimum value from a set of values.

SELECT MIN(column_name) FROM table_name;

Proprietary and confidential

Aggregate Functions - Example: AVG()

QUERY: What is the average cost to rent a "G"-rated film?

APPROACH: Use the AVG() function to find the average value in the rental_rate column of all films whose rating is "G".

```
SELECT AVG(rental_rate)
FRPM film
WHERE rating = "G";
```

RESULTS:

```
+------------------+
| AVG(rental_rate) |
+------------------+
| 2.888876         |
+------------------+
```

Proprietary and confidential

Aggregate Functions - Example: MIN()

QUERY: How short is the shortest film? What about the longest?

APPROACH: Use the MIN() and MAX() function to examine the length.

```
SELECT MIN(length)
FROM film;
```

RESULTS:

```
+-------------+
| MIN(length) |
+-------------+
| 46          |
+-------------+
```

Proprietary and confidential

Aggregate Functions - Example: MIN()

```
SELECT MAX(length)
FROM film;
```

RESULTS:

```
+-------------+
| MAX(length) |
+-------------+
| 185         |
+-------------+
```

Proprietary and confidential

Character Functions

PostgreSQL provides various character functions that can be used to manipulate and process strings. These functions help with tasks such as extracting substrings, converting case, and searching within strings. Here are some commonly used character functions in PostgreSQL:

- UPPER: Converts a string to uppercase.

```
SELECT UPPER(column_name) FROM table_name;
```

- LOWER: Converts a string to lowercase.

```
SELECT LOWER(column_name) FROM table_name;
```

- INITCAP: Converts the first letter of each word to uppercase.

```
SELECT INITCAP(column_name) FROM table_name;
```

Proprietary and confidential

Character Functions

- CONCAT: Concatenates two or more strings.

```
SELECT CONCAT(string1, string2) FROM table_name;
```

- LENGTH: Returns the length of a string.

```
SELECT LENGTH(column_name) FROM table_name;
```

- TRIM: Removes specified characters from the beginning and end of a string.

```
SELECT TRIM(leading/trailing/both characters FROM column_name) FROM
table_name;
```

Proprietary and confidential

Character Functions

- POSITION: Returns the position of a substring within a string.

```
SELECT POSITION(substring IN string) FROM table_name;
```

- REPLACE: Replaces all occurrences of a substring within a string with a new substring.

```
SELECT REPLACE(string, old_substring, new_substring) FROM table_name;
```

These character functions can be used to modify and process strings in PostgreSQL to suit your data manipulation needs.

Proprietary and confidential

Hands-On Lab Exercise-17

(Topic: Character Functions)

PLURALSIGHT

Simple Mathematical Functions in PostgreSQL

In PostgreSQL, you can use the following basic math operators for performing arithmetic operations:

Here's an example of how you can use these operators in SQL queries:

```
 -- Addition
SELECT 10 + 5 AS sum;

-- Subtraction
SELECT 10 - 5 AS difference;
```

Proprietary and confidential

Simple Mathematical Functions in PostgreSQL

```
-- Multiplication
SELECT 10 * 5 AS product;

-- Division
SELECT 10 / 5 AS quotient;

-- Modulus
SELECT 10 % 3 AS remainder;
```

You can use these operators in combination with numeric values or columns in the SELECT statements in PostgreSQL to perform various mathematical calculations.

Proprietary and confidential

Hands-On Lab Exercise-18

(Topic: Simple Mathematical Functions)

PLURALSIGHT

GROUP BY Clause

- **The GROUP BY clause allows you to execute aggregate functions on "groups" of data created by the query**
  - **GROUP BY specifies the column(s) used to create the groups**
  - **The aggregate functions returns a value for EACH group created**

Syntax

```
SELECT column1, column2, etc
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

Proprietary and confidential

GROUP BY Clause Example

QUERY: How many movies are available broken down by rating (G, PG, PG-13, etc)?

APPROACH: Use the GROUP BY clause to create groups of films by rating and then use the COUNT() function to count the number of rows in each group.

```
SELECT rating, COUNT(*)
FROM film
GROUP BY rating;
```

RESULTS:

```
+----------+----------+
| rating   | COUNT(*) |
+----------+----------+
| PG       | 194      |
| G        | 178      |
| NC-17    | 210      |
| PG-13    | 223      |
| R        | 195      |
+----------+----------+
```

Proprietary and confidential

GROUP BY Clause Example

QUERY: What is the average price to rent a movie broken down by rating (G, PG, PG-13, etc)??

APPROACH: Use the GROUP BY clause to create groups of films by rating and then use the AVG() function to calculate the average rental_rate of rows in each group.

```
SELECT rating, avg(rental_rate)
FROM film
GROUP BY rating;
```

RESULTS:

```
+----------+------------------+
| rating   | AVG(rental_rate) |
+----------+------------------+
| PG       | 3.051856         |
| G        | 2.888876         |
| NC-17    | 2.970952         |
| PG-13    | 3.034843         |
| R        | 2.9387818        |
+----------+------------------+
```

Proprietary and confidential

HAVING Clause

HAVING: This clause filters records that work on summarized GROUP BY results.

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

These aggregate functions are powerful tools in PostgreSQL that allow you to perform various calculations and analysis on your data. Use them to derive meaningful insights and metrics from your database.

Proprietary and confidential

HAVING Clause

- **The HAVING clause is used with the GROUP BY clause**

- **It allows you to include only those groups that meet a specified condition**

Syntax

```
SELECT column1, columns2, etc
FROM table_name
WHERE row-condition
GROUP BY column_name(s)
HAVING group-condition
ORDER BY column_name(s);
```

Proprietary and confidential

HAVING Clause Example

QUERY: What is the average rating for movies broken down by rating (G, PG, PG-13, etc)? NOTE: I'm not interested in the rating if there are less than 200 films in the group.

APPROACH: Use the GROUP BY clause to create groups of films by rating and then use the COUNT() function to count the number rows in each group. Only display the groups that have at least 200 rows.

```
SELECT rating, COUNT(*)
FROM film
GROUP BY rating
HAVING COUNT(*) >= 200
ORDER BY rating;
```

RESULTS:

```
+----------+-----------+
| rating   | COUNT(*)  |
+----------+-----------+
| NC-17    | 210       |
| PG-13    | 223       |
+----------+-----------+
```

Proprietary and confidential

HAVING Clause Example

QUERY: What is the average rating for movies broken down by rating (G, PG, PG-13, etc)? NOTE: I'm not interested in the rating if there are less than 200 films in the group.

APPROACH: Use the GROUP BY clause to create groups of films by rating and then use the COUNT() function to count the number rows in each group. Only display the groups that have at least 200 rows.

```
SELECT rating, COUNT(*)
FROM film
GROUP BY rating
HAVING COUNT(*) >= 200
ORDER BY rating;
```

RESULTS:

```
+----------+-----------+
| rating   | COUNT(*)  |
+----------+-----------+
| NC-17    | 210       |
| PG-13    | 223       |
+----------+-----------+
```

Proprietary and confidential

Renaming Columns (Alias) using AS keyword

- **Computed fields don't have an official name in a SQL query**

Example

```
SELECT rental_id, SUM(amount)
FROM sakila.payment
GROUP BY rental_id
ORDER BY rental_id;
```

rental_id	SUM(amount)
NULL	9.95
1	2.99
2	2.99
3	3.99
4	4.99
5	6.99

- **This can be a problem if you want to use it to order the results**
- **SQL provides the AS keyword to create an alias for the column name**
- You can use it for ordering or other purposes

Example

```
SELECT rental_id, SUM(amount) AS total_amount
FROM sakila.payment
GROUP BY rental_id
ORDER BY rental_id;
```

- AS can also create an alias for a table name; we will see this in the next module

Proprietary and confidential

AS keyword- Example

QUERY: What is the average price to rent a movie broken down by rating (G, PG, PG-13, etc) and displayed in ascending order by average price?

APPROACH: Use the GROUP BY clause to create groups of films by rating and then use the `AVG()` function to calculate the average rental_rate of rows in each group. Make sure to name the value returned by the `AVG()` function so that we can use it in the ORDER BY clause.

```
SELECT rating, AVG(rental_rate) AS avg_rate
FROM film
GROUP BY rating
ORDER BY avg_rating;
```

RESULTS:

```
+----------+------------+
| rating   | avg_rate   |
+----------+------------+
| G        | 2.888876   |
| R        | 2.938781   |
| NC-17    | 2.970952   |
| PG-13    | 3.034843   |
| PG       | 3.051856   |
+----------+------------+
```

Proprietary and confidential

Hands-On

(Topic: Aggregate Functions and GROUP BY & HAVING Clause)

PLURALSIGHT