Introduction to SQL

Part-4

Hitesh Kumar Sharma
Instructor, Pluralsight

Data Querying in PostgreSQL

PLURALSIGHT

Using SELECT Statements to Query Data

- Much of the time, when we interact with SQL databases we will be looking for specific data to read

- In SQL, you do this with a SELECT statement

- The SELECT statement's minimum requirements are:
  - SELECT (a keyword)
  - the list of columns to be displayed (or * for all columns)
  - FROM (a keyword)
  - the table to get data from

Using SELECT Statements to Query Data

Syntax

```
SELECT column1, column2, etc
FROM table-name;
```

Syntax

```
SELECT *
FROM table-name;
```

- **The case of the keywords SELECT and FROM isn't important**

  It can be in uppercase, lowercase or mixed case

- **The statement, and all other SQL statements, must end with a semicolon**

Select All (Select *)

\c dvdrental;

SELECT *
FROM actor;

```
dvdrental=# select * from actor;
 actor_id | first_name  |  last_name   |      last_update
----------+-------------+--------------+------------------------
        1 | Penelope    | Guiness      | 2013-05-26 14:47:57.62
        2 | Nick        | Wahlberg     | 2013-05-26 14:47:57.62
        3 | Ed          | Chase        | 2013-05-26 14:47:57.62
        4 | Jennifer    | Davis        | 2013-05-26 14:47:57.62
        5 | Johnny      | Lollobrigida | 2013-05-26 14:47:57.62
        6 | Bette       | Nicholson    | 2013-05-26 14:47:57.62
        7 | Grace       | Mostel       | 2013-05-26 14:47:57.62
        8 | Matthew     | Johansson    | 2013-05-26 14:47:57.62
        9 | Joe         | Swank        | 2013-05-26 14:47:57.62
       10 | Christian   | Gable        | 2013-05-26 14:47:57.62
       11 | Zero        | Cage         | 2013-05-26 14:47:57.62
       12 | Karl        | Berry        | 2013-05-26 14:47:57.62
```

Clauses in a SELECT Statement

In PostgreSQL, the SELECT statement is used to retrieve data from a database. There are several clauses that can be used in conjunction with the SELECT statement to perform more complex queries and retrieve specific data. Some of the commonly used clauses include:

- **WHERE**: Filters the rows based on specific conditions.
- **ORDER BY:** Sorts the result set in ascending or descending order.
- **LIMIT:** Specifies the maximum number of rows to be returned in the result set.
- **GROUP BY:** Groups the result set based on specific columns.
- **HAVING:** Filters the groups defined by the GROUP BY clause.

Where Clause in SELECT Statement

In PostgreSQL, the WHERE clause is used in a SELECT statement to filter rows based on specific conditions. It allows you to retrieve only the rows that meet the specified criteria. Here's how you can use the WHERE clause in a SELECT statement:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

In this syntax:

- column1, column2, ... are the columns you want to retrieve.
- table_name is the name of the table from which you are retrieving data.
- condition is the filtering criteria that the rows must satisfy.

```
SELECT *
FROM employees
WHERE salary > 50000;
```

This query retrieves all columns from the employees table where the salary is greater than 50000.

Where Clause in SELECT Statement

You can use various comparison operators such as =, >, <, >=, <=, <> (not equal), BETWEEN, LIKE, IN, and IS NULL within the WHERE clause to filter rows based on specific conditions. Additionally, you can combine multiple conditions using logical operators such as AND, OR, and NOT to create more complex filtering conditions.

Where Clause in SELECT Statement

Suppose you have a table named products with the following columns: product_id, product_name, category, and price.

Insert sample data:

```
INSERT INTO products (product_id, product_name, category, price)
VALUES (1, 'Laptop', 'Electronics', 1200),
    (2, 'Headphones', 'Electronics', 100),
    (3, 'T-shirt', 'Clothing', 20),
    (4, 'Shoes', 'Footwear', 50);
```

Where Clause in SELECT Statement

Perform a SELECT query using the WHERE clause with different operators:

-- Equal operator

```
SELECT * FROM products WHERE category = 'Electronics';
```

-- Not equal operator

```
SELECT * FROM products WHERE price <> 100;
```

-- Greater than operator

```
SELECT * FROM products WHERE price > 50;
```

Where Clause in SELECT Statement

```
-- Less than operator

SELECT * FROM products WHERE price < 100;

-- Greater than or equal to operator

SELECT * FROM products WHERE price >= 50;

-- Less than or equal to operator

SELECT * FROM products WHERE price <= 100;

-- BETWEEN operator

SELECT * FROM products WHERE price BETWEEN 100 AND 1200;
```

Where Clause in SELECT Statement

-- LIKE operator

SELECT * FROM products WHERE product_name LIKE 'L%';

-- IN operator

SELECT * FROM products WHERE category IN ('Electronics', 'Clothing');

-- IS NULL operator

SELECT * FROM products WHERE product_name IS NULL;

These queries demonstrate the usage of various operators within the WHERE clause for filtering data based on different conditions.

Hands-On Lab Exercise-10

(Topic: Where Clause)

Like operator in SELECT Statement

In PostgreSQL, the LIKE operator is used in the SELECT statement to search for a specified pattern in a column. The LIKE operator is used with the WHERE clause and is often used with wildcard characters such as % (represents zero, one, or multiple characters) and _ (represents a single character).

```
WHERE title LIKE "Sit%"
    means the value starts with "Sit"

WHERE title LIKE "%sit"
    means the value ends with "sit"

WHERE title LIKE "%sit%"
    means contains "sit" anywhere in the string
```

Between operator in SELECT Statement

In PostgreSQL, the BETWEEN operator is used in the SELECT statement to retrieve values within a specified range. The BETWEEN operator is typically used with the WHERE clause and can be applied to various data types such as numbers, dates, and strings.

Assuming we have a table named Products with a column named UnitPrice, and we want to select all products with a unit price between $10 and $20, the following query can be used:

```
SELECT *
FROM Products
WHERE UnitPrice BETWEEN 10 AND 20;
```

This query will fetch all rows from the Products table where the UnitPrice falls within the range of $10 and $20, inclusive.

DISTINCT in SELECT Statement

In PostgreSQL, the DISTINCT keyword is used in the SELECT statement to eliminate duplicate rows from the query results. It operates on the entire row returned by the SELECT statement, ensuring that only unique rows are displayed in the result set.

Assuming we have a table named Customers with columns CustomerName and City, and we want to retrieve all unique city names from the table, the following query can be used:

```
SELECT DISTINCT City
FROM Customers;
```

This query will fetch all unique values from the City column in the Customers table, ensuring that each city name appears only once in the result set.

Hands-On Lab Exercise-13, 15 & 16

(Topic: Like, Between & DISTINCT Operator)

Order-By Clause in SELECT Statement

In PostgreSQL, the ORDER BY clause is used in a SELECT statement to sort the result set in either ascending or descending order based on specified columns. Here's how you can use the ORDER BY clause:

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1 [ASC | DESC], column2 [ASC | DESC], ...;
```

In this syntax:

- column1, column2, ... are the columns by which you want to sort the result set.
- table_name is the name of the table from which you are retrieving data.
- ASC (ascending) or DESC (descending) specifies the sorting order. If not explicitly specified, ASC is the default sorting order.

```
SELECT *
FROM employees
ORDER BY salary DESC;
```

This query retrieves all columns from the employees table and sorts the result set in descending order based on the salary column.

Order By Clause in SELECT Statement

Suppose you have a table named students with the following columns: student_id, student_name, age, and grade.

Insert sample data:

```
INSERT INTO students (student_id, student_name, age, grade)
VALUES (1, 'John Doe', 20, 'A'),
       (2, 'Jane Smith', 22, 'B'),
       (3, 'Michael Johnson', 21, 'C'),
       (4, 'Emily Williams', 20, 'A');
```

Order By Clause in SELECT Statement

Perform a SELECT query using the ORDER BY clause with different options:

-- Ascending order (default)

SELECT * FROM students ORDER BY student_name;

-- Descending order

SELECT * FROM students ORDER BY age DESC;

-- Order by multiple columns

SELECT * FROM students ORDER BY grade, age DESC;

Order By Clause in SELECT Statement

```
-- NULLS FIRST

SELECT * FROM students ORDER BY student_id NULLS FIRST;

-- NULLS LAST

SELECT * FROM students ORDER BY age NULLS LAST;
```

These queries demonstrate the usage of the ORDER BY clause with various options, including sorting by a single column, multiple columns, and handling NULL values.