# #PostgreSQL#

Section 2. Filtering Data
Where – filter rows based on a specified condition.
AND operator – combine two boolean expressions and return true if both expressions evaluate to true.
OR operator – combine two boolean expressions and return false if either expression evaluates to false.
Limit – retrieve a subset of rows generated by a query.
Fetch– limit the number of rows returned by a query.
In – select data that matches any value in a list of values.
Between – select data that is a range of values.
Like – filter data based on pattern matching.
Is Null – check if a value is null or not.

Can You Teach Me above all the topics with Realtime  example (create a simple table and insert some values)  and show output in Table Format and Also Explain Each topic why this needed, what happen, like syntax and so on.

Certainly! Let's explore each of these PostgreSQL data filtering topics with a real-time example, using a simple table called `employees`. We'll create this table, insert some values, and then demonstrate how each filtering mechanism works.

## Table Creation

First, we'll create a table called `employees` to hold some data. The table will have columns for `id`, `name`, `age`, `salary`, and `department`.

```sql


CREATE TABLE employees (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50),
    age INTEGER,
    salary NUMERIC(10, 2),
    department VARCHAR(50)
);
```

## Inserting Data

Next, we'll insert some example data into the `employees` table.

```sql
INSERT INTO employees (name, age, salary, department)
VALUES
('Alice', 30, 50000, 'HR'),
('Bob', 25, 40000, 'Finance'),
('Charlie', 35, 60000, 'IT'),
('David', 40, 70000, 'Marketing'),
('Eve', 28, 55000, 'HR');
```

### Data in the `employees` Table

Here's what our `employees` table looks like after inserting the data:

| id | name | age | salary | department |
|---|---|---|---|---|
| 1 | Alice | 30 | 50000 | HR |
| 2 | Bob | 25 | 40000 | Finance |
| 3 | Charlie | 35 | 60000 | IT |
| 4 | David | 40 | 70000 | Marketing |
| 5 | Eve | 28 | 55000 | HR |

### Example 1: WHERE Clause - Filter Rows Based on a Condition

The `WHERE` clause is used to filter rows that meet a specific condition.

**Syntax:**

```sql
SELECT * FROM employees WHERE condition;
```

**Example:** To retrieve employees who are older than 30 years:

```sql
SELECT * FROM employees WHERE age > 30;
```

| id | name | age | salary | department |
|---|---|---|---|---|
| 3 | Charlie | 35 | 60000 | IT |
| 4 | David | 40 | 70000 | Marketing |

### Example 2: AND Operator - Combine Multiple Conditions

The `AND` operator combines two boolean expressions, returning `true` if both conditions are true.

**Syntax:**

```sql
SELECT * FROM employees WHERE condition1 AND condition2;
```

**Example:** To retrieve employees who are older than 30 and have a salary greater than 50000:

```sql
SELECT * FROM employees WHERE age > 30 AND salary > 50000;
```

| id | name | age | salary | department |
|---|---|---|---|---|
| 3 | Charlie | 35 | 60000 | IT |
| 4 | David | 40 | 70000 | Marketing |

## Example 3: OR Operator - Combine Multiple Conditions

The `OR` operator combines two boolean expressions, returning `true` if at least one condition is true.

**Syntax:**

```sql
SELECT * FROM employees WHERE condition1 OR condition2;
```

**Example:** To retrieve employees who are either in HR or have a salary greater than 60000:

```sql
SELECT * FROM employees WHERE department = 'HR' OR salary > 60000;
```

| id | name | age | salary | department |
|---|---|---|---|---|
| 3 | Charlie | 35 | 60000 | IT |
| 4 | David | 40 | 70000 | Marketing |
| 5 | Eve | 28 | 55000 | HR |

## Example 4: LIMIT Clause - Retrieve a Subset of Rows

The `LIMIT` clause is used to restrict the number of rows returned by a query.

**Syntax:**

```sql
SELECT * FROM employees LIMIT number;
```

**Example:** To retrieve only the first 3 employees:

```sql
SELECT * FROM employees LIMIT 3;
```

| id | name | age | salary | department |
|---|---|---|---|---|
| 1 | Alice | 30 | 50000 | HR |
| 2 | Bob | 25 | 40000 | Finance |
| 3 | Charlie | 35 | 60000 | IT |

## Example 5: FETCH Clause - Limit the Number of Rows Retrieved

The `FETCH` clause is a synonym for `LIMIT` in PostgreSQL, used to restrict the number of rows returned.

**Syntax:**

```sql
SELECT * FROM employees FETCH FIRST number ROWS ONLY;
```

**Example:** To retrieve only the first 2 employees:

```sql
SELECT * FROM employees FETCH FIRST 2 ROWS ONLY;
```

| id | name | age | salary | department |
|---|---|---|---|---|
| 1 | Alice | 30 | 50000 | HR |
| 2 | Bob | 25 | 40000 | Finance |

## Example 6: IN Operator - Select Data Matching Any Value in a List

The `IN` operator is used to select data that matches any value from a list.

**Syntax:**

```sql
SELECT * FROM employees WHERE column_name IN (value1, value2, ...);
```

**Example:** To retrieve employees who belong to the 'HR' or 'IT' departments:

```sql
SELECT * FROM employees WHERE department IN ('HR', 'IT');
```

| id | name | age | salary | department |
|---|---|---|---|---|
| 1 | Alice | 30 | 50000 | HR |
| 3 | Charlie | 35 | 60000 | IT |
| 5 | Eve | 28 | 55000 | HR |

## Example 7: BETWEEN Clause - Select Data in a Range

The `BETWEEN` clause is used to select data within a range of values.

**Syntax:**

```sql
SELECT * FROM employees WHERE column_name BETWEEN value1 AND value2;
```

**Example:** To retrieve employees with ages between 25 and 35:

```sql
SELECT * FROM employees WHERE age BETWEEN 25 AND 35;
```

| id | name | age | salary | department |
|---|---|---|---|---|
| 1 | Alice | 30 | 50000 | HR |
| 2 | Bob | 25 | 40000 | Finance |
| 3 | Charlie | 35 | 60000 | IT |

## Example 8: LIKE Clause - Filter Data Based on Pattern Matching

The `LIKE` clause is used to filter data based on pattern matching.

**Syntax:**

```sql
SELECT * FROM employees WHERE column_name LIKE pattern;
```

**Example:** To retrieve employees whose names start with 'A':

```sql
SELECT * FROM employees WHERE name LIKE 'A%';
```

| id | name | age | salary | department |
|---|---|---|---|---|
| 1 | Alice | 30 | 50000 | HR |

## Example 9: IS NULL - Check if a Value is NULL

The `IS NULL` clause is used to check if a value is null or not.

**Syntax:**

```sql
SELECT * FROM employees WHERE column_name IS NULL;
```

**Example:** Assuming we have a column `address` that might be NULL:

```sql
SELECT * FROM employees WHERE address IS NULL;
```

Since no employee has an `address` column in our example, it will return no rows.

## Explanation:

Each of these filtering mechanisms allows you to extract specific data from your table based on predefined criteria. They are essential for refining your data analysis and ensuring that the information you retrieve is accurate and relevant to your needs. Understanding these clauses helps in writing efficient and effective SQL queries, which is crucial for database management and reporting tasks.