PostgreSQL SQL Clauses Guide

I Have to Learn Postgresql can You teach me following Topics with Example and Output and Explanation its Very Help full for me!
PostgreSQL Clause
PostgreSQL Where
PostgreSQL Order By
PostgreSQL Group By
PostgreSQL Having
PostgreSQL Distinct
PostgreSQL Limit
PostgreSQL Fetch

Certainly! Let's dive into each topic one by one with examples, output, and clear explanations. I will create a sample table named **employees** to demonstrate these concepts.

Sample Table: `employees`

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

INSERT INTO employees (name, department, salary, joining_date) VALUES
('Alice', 'HR', 60000, '2022-01-15'),
('Bob', 'IT', 75000, '2021-06-20'),
('Charlie', 'Finance', 50000, '2020-09-10'),
('Diana', 'IT', 90000, '2023-03-12'),
('Eve', 'HR', 55000, '2022-07-25');
```

---

1. PostgreSQL Clause

A clause in SQL represents specific keywords used to define the purpose of a query. Common clauses include `SELECT`, `FROM`, and `WHERE`.

Example:

```sql
SELECT name, department, salary
FROM employees
```

```
  WHERE department = 'IT';
```

Output:

name	department	salary
Bob	IT	75000.00
Diana	IT	90000.00

Explanation:

- `SELECT` specifies which columns to retrieve.
- `FROM` specifies the table.
- `WHERE` filters rows based on a condition.

---

2. PostgreSQL WHERE

The `WHERE` clause filters rows based on a condition.

Example:

```sql
SELECT name, salary
FROM employees
WHERE salary > 60000;
```

Output:

name	salary
Bob	75000.00
Diana	90000.00

Explanation:

- Filters employees with a salary greater than 60000.

---

3. PostgreSQL ORDER BY

The `ORDER BY` clause sorts the result set in ascending (`ASC`) or descending (`DESC`) order.

Example:

```sql
SELECT name, department, salary
FROM employees
ORDER BY salary DESC;
```

Output:

name	department	salary
Diana	IT	90000.00
Bob	IT	75000.00
Alice	HR	60000.00
Eve	HR	55000.00
Charlie	Finance	50000.00

Explanation:

- Sorts employees by salary in descending order.

---

4. PostgreSQL GROUP BY

The `**GROUP BY**` clause groups rows sharing the same values in specified columns and allows aggregate functions like `**SUM**`, `**COUNT**`, etc.

Example:

```sql
SELECT department, COUNT(*) AS total_employees
FROM employees
GROUP BY department;
```

Output:

department	total_employees
HR	2
IT	2
Finance	1

Explanation:

- Groups employees by department and counts the number of employees in each.

---

5. PostgreSQL HAVING

The `**HAVING**` clause filters groups based on a condition, often used with `**GROUP BY**`.

Example:

```sql
SELECT department, AVG(salary) AS avg_salary
FROM employees
GROUP BY department
HAVING AVG(salary) > 60000;
```

Output:

department	avg_salary
IT	82500.00

Explanation:

- Groups employees by department, calculates the average salary, and filters departments where the average salary is greater than 60000.

---

6. PostgreSQL DISTINCT

The `DISTINCT` clause removes duplicate values.

Example:

```
sql


  SELECT DISTINCT department
  FROM employees;
```

Output:

department
HR
IT
Finance

Explanation:

- Retrieves unique department names from the table.

---

7. PostgreSQL LIMIT

The `LIMIT` clause restricts the number of rows returned.

Example:

```
sql


  SELECT name, salary
  FROM employees
  ORDER BY salary DESC
  LIMIT 3;
```

Output:

name	salary
Diana	90000.00
Bob	75000.00
Alice	60000.00

Explanation:

- Retrieves the top 3 highest-paid employees.

---

8. PostgreSQL FETCH

The `FETCH` clause is similar to `LIMIT` but offers better compatibility with SQL standards.

Example:

```sql
SELECT name, salary
FROM employees
ORDER BY salary DESC
FETCH FIRST 2 ROWS ONLY;
```

Output:

name	salary
Diana	90000.00
Bob	75000.00

Explanation:

- Retrieves the top 2 highest-paid employees using SQL-standard syntax.

Would you like to explore any of these topics further or move to advanced PostgreSQL concepts?

Also provide If postgresql have Multiple clause if i use these clause Which clause is First Execute or order of the Clause Execution

In PostgreSQL (and SQL in general), the execution of a query follows a specific logical order of operations, which may differ from the order in which the clauses are written. Here's the typical order of execution for a query with multiple clauses:

Order of Execution of SQL Clauses

1. **`FROM` Clause**
   The database first identifies the source tables and joins them if necessary.
2. **`WHERE` Clause**
   Filters rows based on the specified condition.
3. **`GROUP BY` Clause**
   Groups the filtered rows into sets based on the specified columns.
4. **`HAVING` Clause**
   Filters the grouped data based on a condition (applies after grouping).
5. **`SELECT` Clause**
   Retrieves the specified columns or expressions.

6. **`DISTINCT` Clause**
   Removes duplicate rows from the results.
7. **`ORDER BY` Clause**
   Sorts the rows based on the specified columns or expressions.
8. **`LIMIT`/`FETCH` Clauses**
   Restricts the number of rows returned.

Example with Multiple Clauses

```sql
SELECT department, AVG(salary) AS avg_salary
FROM employees
WHERE salary > 50000
GROUP BY department
HAVING AVG(salary) > 60000
ORDER BY avg_salary DESC
LIMIT 2;
```

Explanation of Execution Steps

1. **`FROM employees`:**
   The query starts with the **`FROM`** clause to determine the table being queried.
2. **`WHERE salary > 50000`:**
   Filters rows where the salary is greater than 50,000.
   Remaining rows:

   ```
   Alice, Bob, Diana, Eve
   ```

3. **`GROUP BY department`:**
   Groups the filtered rows by **`department`**.
   Grouped result:

   ```makefile
   HR: Alice, Eve
   IT: Bob, Diana
   ```

4. **`HAVING AVG(salary) > 60000`:**
   Filters groups where the average salary exceeds 60,000.
   Remaining groups:

   ```
   IT
   ```

5. **`SELECT department, AVG(salary) AS avg_salary`:**
   Computes the average salary for the remaining groups.
   Result:

```
IT, 82500.00
```

6. **`ORDER BY avg_salary DESC`:**
   Sorts the results by average salary in descending order.
   (In this case, there's only one row, so no sorting happens.)
7. **`LIMIT 2`:**
   Limits the output to the first 2 rows (not needed here since only one row remains).

Key Points to Remember

- Even though the **`SELECT`** clause appears first in the query, it is logically executed later.
- **`WHERE`** filters rows before grouping, while **`HAVING`** filters after grouping.
- **`ORDER BY`** and **`LIMIT`** are applied after all the data processing (filtering, grouping, etc.).

Would you like more examples or further clarification?