The `WITH` clause in PostgreSQL (also known as \*\*Common Table Expressions\*\* or CTEs) is used to define temporary result sets that you can reference within a `SELECT`, `INSERT`, `UPDATE`, or `DELETE` query. It helps make complex queries more readable and efficient by breaking them down into smaller parts.

### Basic Syntax of `WITH`

```sql

WITH cte\_name AS (

-- Some query

SELECT ...

)

SELECT ...

FROM cte\_name;

```

- `cte\_name`: The name of the common table expression (CTE), which acts like a temporary result set.

- The query inside the CTE can be referenced in the subsequent main query.

### Simple Example

Let's look at a basic example involving an employee table to calculate and list high-salary employees.

#### Step 1: Create `employees` Table

```sql

CREATE TABLE employees (

employee\_id SERIAL PRIMARY KEY,

name VARCHAR(100),

department VARCHAR(50),

salary NUMERIC(10, 2)

);

```

#### Step 2: Insert Sample Data

```sql

INSERT INTO employees (name, department, salary)

VALUES

('John Doe', 'HR', 50000),

('Jane Smith', 'Engineering', 70000),

('Alice Johnson', 'Engineering', 80000),

('Bob Williams', 'HR', 45000),

('Sara Lee', 'Marketing', 60000);

```

#### Step 3: Use `WITH` Clause to Find Employees With High Salary

We can define a CTE to filter employees with a salary greater than or equal to 60,000, and then use this temporary result to perform the main query.

```sql

WITH high\_salary\_employees AS (

SELECT employee\_id, name, salary

FROM employees

WHERE salary >= 60000

)

SELECT \*

FROM high\_salary\_employees

ORDER BY salary DESC;

```

### Explanation:

- The \*\*CTE\*\* (`high\_salary\_employees`) filters all employees with a salary of 60,000 or higher.

- In the \*\*main query\*\*, we simply select all columns from the CTE and order the results by salary in descending order.

### Output:

| employee\_id | name | salary |

|-------------|----------------|---------|

| 3 | Alice Johnson | 80000.00|

| 2 | Jane Smith | 70000.00|

| 5 | Sara Lee | 60000.00|

### More Complex Example with Multiple CTEs

You can also define multiple CTEs inside a single `WITH` clause.

```sql

WITH high\_salary AS (

SELECT employee\_id, name, salary

FROM employees

WHERE salary >= 60000

),

engineering\_department AS (

SELECT employee\_id, name

FROM employees

WHERE department = 'Engineering'

)

SELECT hse.name AS high\_earner, ed.name AS engineer

FROM high\_salary hse

JOIN engineering\_department ed ON hse.employee\_id = ed.employee\_id;

```

In this case:

- One CTE (`high\_salary`) gets all employees with a high salary.

- Another CTE (`engineering\_department`) filters employees in the Engineering department.

- The main query joins these two CTEs to find employees in the Engineering department who also have a high salary.

### Conclusion

The `WITH` clause simplifies queries by allowing you to build temporary tables for better readability and organization. This is especially useful when dealing with complex joins or subqueries.