

# SQL Clauses

## 1. WHERE Clause

### Purpose:

Used to filter records that fulfill a specific condition.

### Used With:

- SELECT
- UPDATE
- DELETE

### Syntax:

SELECT column1, column2, ...

FROM table\_name

WHERE condition;

### Examples:

```
SELECT * FROM customer WHERE city = 'Seattle';
```

Object Explorer

- Extensions
- Foreign Data Wrappers
- Languages
- Publications
- Schemas (1)
  - public
    - Aggregates
    - Collations
    - Domains
    - FTS Configurations
    - FTS Dictionaries
    - FTS Parsers
    - FTS Templates
    - Foreign Tables
    - Functions
    - Materialized Views
    - Operators
    - Procedures
    - Sequences
    - Tables (5)
      - customer
      - employee\_backup
      - product
      - sales
      - students
    - Trigger Functions
    - Types
    - Views
  - Subscriptions
  - Login/Group Roles
  - Tablespaces

www/postgres@P... x www/postgres@PostgreSQL 17\* x

www/postgres@PostgreSQL 17

Query Query History

```
1 SELECT * FROM customer WHERE city = 'Seattle';
```

Scratch Pad x

Data Output Messages Notifications

Showing rows: 1 to 31 Page No: 1 of 1

	customer_id [PK] character varying (255)	customer_name character varying (255)	segment character varying (255)	age integer	country character varying (255)	city character varying (255)	state character varying (255)	postal_code bigint	region character
1	IM-15070	Irene Maddox	Consumer	66	United States	Seattle	Washington	98103	West
2	DK-13090	Dave Kipp	Consumer	24	United States	Seattle	Washington	98103	West
3	NK-18490	Neil Knudson	Home Office	61	United States	Seattle	Washington	98105	West
4	DB-13060	Dave Brooks	Consumer	50	United States	Seattle	Washington	98115	West
5	MJ-17740	Max Jones	Consumer	40	United States	Seattle	Washington	98115	West
6	KC-16675	Kimberly Carter	Corporate	31	United States	Seattle	Washington	98105	West
7	SS-20590	Sonia Sunley	Consumer	51	United States	Seattle	Washington	98115	West
8	JF-15490	Jeremy Farry	Consumer	38	United States	Seattle	Washington	98105	West
9	EB-13840	Ellis Ballard	Corporate	25	United States	Seattle	Washington	98105	West
10	AT-10735	Annie Thurman	Consumer	30	United States	Seattle	Washington	98115	West
11	DS-13030	Darrin Sayre	Home Office	62	United States	Seattle	Washington	98115	West
12	JS-15880	John Stevenson	Consumer	60	United States	Seattle	Washington	98103	West
13	DW-13540	Don Weiss	Consumer	60	United States	Seattle	Washington	98105	West

**SELECT \* FROM customer WHERE state = 'California';**

www/postgres@P... x www/postgres@PostgreSQL 17\* x

www/postgres@PostgreSQL 17

Query Query History

```
1 SELECT * FROM customer WHERE state = 'California';
```

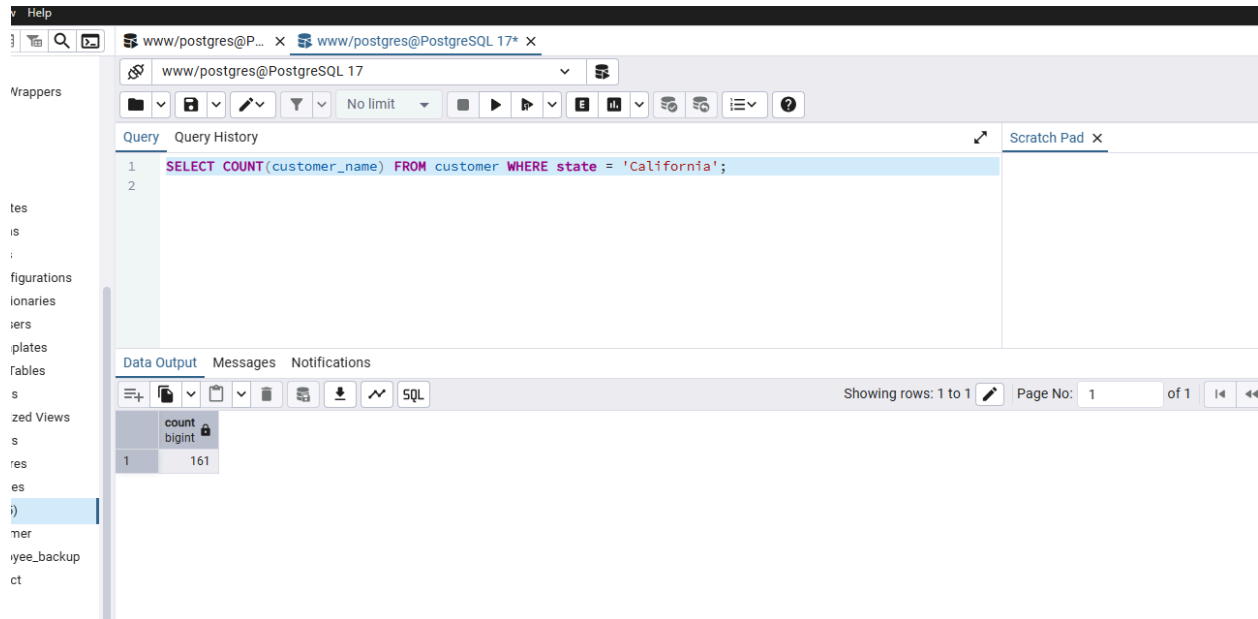
Scratch Pad x

Data Output Messages Notifications

Showing rows: 1 to 161 Page No: 1 of 1

	customer_id [PK] character varying (255)	customer_name character varying (255)	segment character varying (255)	age integer	country character varying (255)	city character varying (255)	state character varying (255)	postal_code bigint	region character
1		Darrin Van Huff	Corporate	31	United States	Los Angeles	California	90036	West
2		Brosina Hoffman	Consumer	20	United States	Los Angeles	California	90032	West
3		Zuschuss Donatelli	Consumer	66	United States	San Francisco	California	94109	West
4		Eric Hoffmann	Consumer	21	United States	Los Angeles	California	90049	West
5		Ruben Ausman	Corporate	51	United States	Los Angeles	California	90049	West
6		Kunst Miller	Consumer	69	United States	Los Angeles	California	90004	West
7		Duane Noonan	Consumer	42	United States	San Francisco	California	94122	West
8		Jim Sink	Corporate	33	United States	Los Angeles	California	90036	West
9		Katherine Ducich	Consumer	33	United States	San Francisco	California	94122	West
10		Lindsay Shagiani	Home Office	66	United States	Los Angeles	California	90004	West

**SELECT COUNT(customer\_name) FROM customers WHERE state = 'California';**



## 2. ORDER BY Clause

### Purpose:

Used to sort the result set in **ascending (ASC)** or **descending (DESC)** order.

### Syntax:

```
SELECT column1, column2, ...
```

```
FROM table_name
```

```
[WHERE condition]
```

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

### Notes:

- Default is ASC (ascending).

- You can sort by one or more columns.

## Examples:

```
SELECT customer_id, customer_name FROM customer ORDER BY customer_name;
```

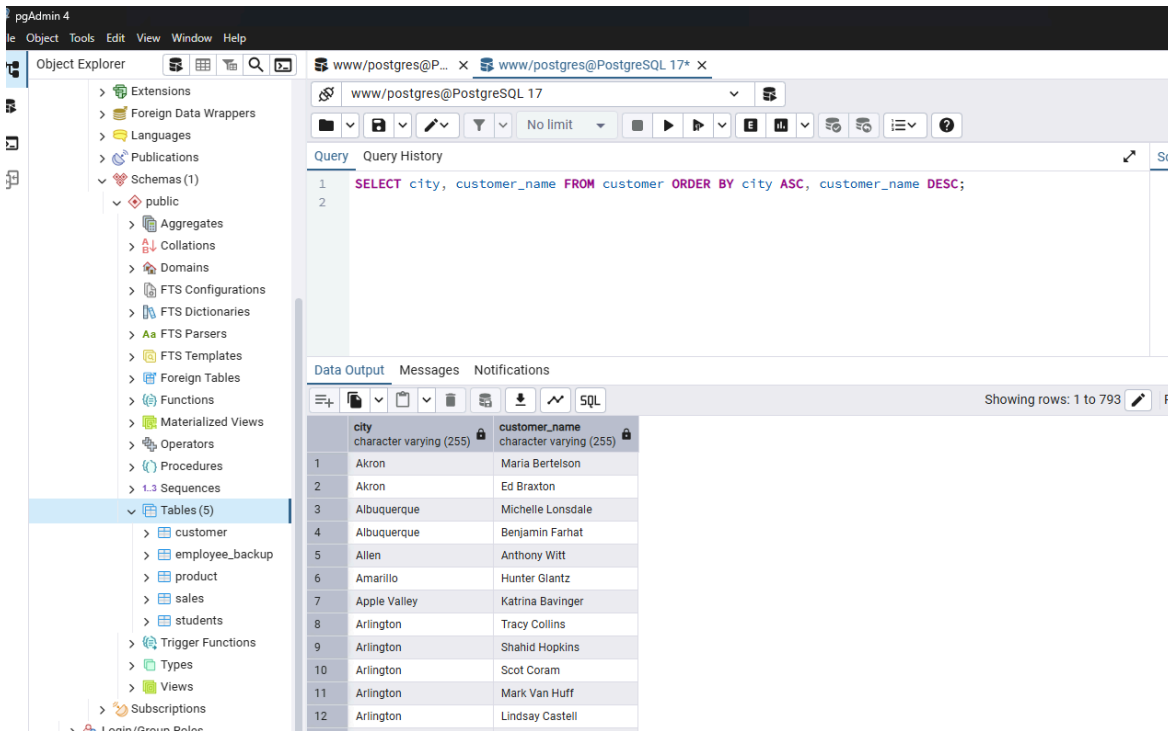
The screenshot shows a PostgreSQL client window with a query editor and a results pane. The query editor contains the following SQL statement:

```
1 SELECT customer_id, customer_name FROM customer ORDER BY customer_name;
```

The results pane displays the output of the query, showing 11 rows of data. The columns are labeled `customer_id` and `customer_name`. The data is sorted by `customer_name` in ascending order.

customer_id	customer_name
1	AB-10015
2	AH-10030
3	AS-10045
4	AB-10060
5	AH-10075
6	AS-10090
7	AB-10105
8	AH-10120
9	AS-10135
10	AB-10150
11	AB-10165

```
SELECT city, customer_name FROM customer ORDER BY city ASC, customer_name DESC;
```



This query **retrieves a list of cities and customer names** from the customer table, and **sorts** the results:

- First by city in **ascending** ( $A \rightarrow Z$ ) order
- Then by customer\_name in **descending** ( $Z \rightarrow A$ ) order **within each city**

### 3. GROUP BY Clause

#### Purpose:

The GROUP BY clause is used to **split rows into groups** based on one or more columns. It is commonly used with **aggregate functions** such as:

- COUNT() – counts number of rows
- SUM() – adds values

- AVG() – calculates average
- MIN() / MAX() – finds minimum or maximum value

Each group returns **one result row**.

### **Syntax:**

SELECT aggregate\_function(column), column2, ...

FROM table\_name

WHERE condition

GROUP BY column2, column3, ...

ORDER BY column2;

### **Example:**

```
SELECT COUNT(customer_id), state FROM Customer GROUP BY state;
```

The screenshot shows a PostgreSQL client interface with a sidebar on the left containing a tree view of database objects. The main area displays a query and its results. The query is:

```
1 SELECT COUNT(customer_id), state FROM Customer GROUP BY state;
```

The results are shown in a table with two columns: 'count' and 'state'. The data is as follows:

count	state
3	Oklahoma
20	Colorado
30	North Carolina
5	Mississippi
24	Florida
6	Delaware
2	Nevada
6	Louisiana
87	New York
10	New Jersey
2	Arkansas
4	New Mexico
6	Missouri

## 4. SELECT DISTINCT Clause

### Purpose:

Used to return **only unique (non-duplicate)** values.

### Syntax:

```
SELECT DISTINCT column1, column2, ...
```

```
FROM table_name;
```

### Example:

```
SELECT DISTINCT state FROM customer;
```

The screenshot shows a PostgreSQL client interface with a sidebar explorer on the left and a main query editor on the right. The sidebar lists various database objects, with 'Tables (5)' selected under the 'public' schema. The main editor displays a query: `SELECT DISTINCT state FROM customer;`. Below the query editor, the 'Data Output' tab shows the results of the query as a table with 13 rows, each containing a state name. The interface also includes a top toolbar with various icons and a bottom status bar indicating 'Showing rows: 1 to 4'.

	state
1	Oklahoma
2	Colorado
3	North Carolina
4	Mississippi
5	Florida
6	Delaware
7	Nevada
8	Louisiana
9	New York
10	New Jersey
11	Arkansas
12	New Mexico
13	Missouri

## 5. LIMIT Clause (*PostgreSQL-specific*)

### Purpose:

Limits the number of rows returned by the query.

### Syntax:

SELECT column1, column2, ...

FROM table\_name

ORDER BY column

LIMIT number;



## Example:

```
SELECT customer_name, customer_id, age
```

```
FROM customer
```

```
ORDER BY customer_name
```

```
LIMIT 6;
```

The screenshot shows the Admin 4 PostgreSQL interface. On the left, the Object Explorer displays the database schema, with the 'public' schema expanded and 'Tables (5)' selected. The main query editor shows the following SQL query:

```
1 SELECT customer_name, customer_id, age
2 FROM customer
3 ORDER BY customer_name
4 LIMIT 6;
5
```

Below the query editor, the Data Output tab is active, displaying the results of the query in a table format. The table has three columns: customer\_name, customer\_id, and age. The results are as follows:

	customer_name	customer_id	age
1	Aaron Bergman	AB-10015	66
2	Aaron Hawkins	AH-10030	60
3	Aaron Smayling	AS-10045	42
4	Adam Bellavance	AB-10060	25
5	Adam Hart	AH-10075	21
6	Adam Shillingsburg	AS-10090	46

## Summary Table

Clause	Use Case
WHERE	Filter records based on conditions
ORDER BY	Sort results in ASC or DESC order
GROUP BY	Group data for aggregate functions
DISTINCT	Select only unique values
LIMIT	Restrict number of records returned (PostgreSQL)

## Practice SQL Questions

### GROUP BY Clause

1. Count the number of customers in each state.
2. Show the average discount given per product category.

### WHERE + GROUP BY

3. Count the number of customers per state in the "West" region.
4. Get the total sales for products where quantity is more than 5.

### ORDER BY Clause

5. List all orders sorted by sales in descending order.
6. Display cities and customer names, sorted by city (A-Z) and customer name (Z-A).

## **DISTINCT Clause**

7. List all unique product categories.
8. Get all unique combinations of segment and region from orders.

## **LIMIT Clause**

9. Show the top 10 customers with the highest total sales.
10. Display the first 5 orders sorted by order date.

You can download the complete set of SQL notes and practice files from this GitHub repository:

👉 [SQL-resources-and-tutorials by akshay-dhage](#)