**PROMPT:** give me the detailed documentation of delete command with examples in PostgreSQL

## **What is the DELETE Command in PostgreSQL?**

The **DELETE** command is a **Data Manipulation Language (DML)** statement in SQL used to remove one or more rows from an existing table.

* It **does not remove the table structure** (for that, we use DROP TABLE).
* You can delete **specific rows** using a condition (WHERE) or **all rows** without a condition.
* Once committed, **deleted data cannot be recovered** unless you have a backup or use transaction rollback before committing.

**Syntax**

DELETE FROM table\_name

[WHERE condition]

[RETURNING \* | column\_list];

### **Parameters**

| **Parameter** | **Description** |
| --- | --- |
| table\_name | The name of the table from which rows will be deleted. |
| WHERE condition | Optional. Specifies which rows to delete. Without it, **all rows** will be deleted. |
| RETURNING | Optional. Returns the deleted rows (can be \* for all columns or a specific list of columns). Useful for logging or checking what was deleted. |

## **Key Points to Remember**

* Without WHERE, **all rows** are deleted (table becomes empty but still exists).
* Use transactions (BEGIN...ROLLBACK) for safety when testing.
* Foreign keys with ON DELETE CASCADE will automatically delete related rows in child tables.
* Deletion is **permanent after a commit**.
* In large tables, deletion can be slower than truncation (TRUNCATE), because DELETE logs each row change individually in the **Write-Ahead Log (WAL)**.

**Examples**

### **Delete Specific Rows**

DELETE FROM employees

WHERE department = 'HR';

Deletes all employees from the HR department.

### **Delete All Rows from a Table**

DELETE FROM employees;

Deletes all rows in employees.  
 Be careful — this is irreversible without backup.

### **Delete Using Multiple Conditions**

DELETE FROM employees

WHERE department = 'Finance'

AND salary < 40000;

Deletes finance department employees earning less than ₹40,000.

### **Delete Using a Subquery**

DELETE FROM employees

WHERE department\_id IN (

SELECT department\_id

FROM departments

WHERE location = 'Mumbai'

);

Deletes employees whose department is located in Mumbai.

### **Delete and Return Deleted Data**

DELETE FROM employees

WHERE employee\_id = 101

RETURNING \*;

Deletes the employee with ID 101 and **returns the deleted row’s data**.

### **Delete with JOIN (Using USING clause in PostgreSQL)**

DELETE FROM employees e

USING departments d

WHERE e.department\_id = d.department\_id

AND d.department\_name = 'IT';

Deletes employees who work in the IT department.

### **Delete in a Transaction (Safe Mode)**

BEGIN;

DELETE FROM employees

WHERE employee\_id = 105;

ROLLBACK; -- If you change your mind:

COMMIT; -- If you are sure:

Allows you to **undo** the deletion before committing.

## **DELETE vs TRUNCATE vs DROP**

| **Command** | **Deletes Rows?** | **Removes Structure?** | **Logging** | **Speed** | **Can Rollback?** |
| --- | --- | --- | --- | --- | --- |
| **DELETE** | Yes | No | Logs each row | Slower | Yes |
| **TRUNCATE** | Yes (all rows) | No | Minimal logging | Faster | Yes (if in transaction) |
| **DROP** | Yes (and table) | Yes | Logs table removal | N/A | Yes (if in transaction) |

## **Best Practices**

* Always use WHERE unless you **really** want to empty the table.
* For large deletions, consider batching with LIMIT to avoid long locks:

DELETE FROM logs

WHERE log\_date < '2024-01-01'

LIMIT 5000;

* Use RETURNING for audit logs.
* If you only want to reset a table, prefer TRUNCATE for speed.