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

**What is TRUNCATE in PostgreSQL?**

In PostgreSQL, the **TRUNCATE** command is a **Data Definition Language (DDL)** operation used to **remove all rows** from a table **very quickly** without scanning them row-by-row.

It is:

* **Faster** than DELETE (no WHERE clause) because it bypasses row-by-row deletion.
* **Minimal logging** in the Write-Ahead Log (WAL).
* **Cannot be rolled back partially** if executed outside a transaction (unless explicitly wrapped in one).
* **Resets** identity/serial columns by default (if specified).

## **Syntax**

TRUNCATE [ TABLE ] table\_name [, ...]

[ RESTART IDENTITY | CONTINUE IDENTITY ]

[ CASCADE | RESTRICT ];

**Parameters:**

* **TABLE** → Optional keyword for readability.
* **table\_name** → Name(s) of the table(s) to truncate. Multiple tables can be given, separated by commas.
* **RESTART IDENTITY** → Resets any SERIAL/IDENTITY columns to start from their initial value.
* **CONTINUE IDENTITY** → Keeps the sequence values as they are (default).
* **CASCADE** → Also truncates tables that have foreign-key references to the table(s) being truncated.
* **RESTRICT** → Refuses to truncate if there are any foreign-key dependencies (default).

**Key Points to Remember**

1. **Faster than DELETE** because it does not scan each row or fire individual DELETE triggers.
2. **Cannot have a WHERE clause** — it always removes **all rows**.
3. **Privilege required**: You must own the table or have the TRUNCATE privilege.
4. **Foreign keys**:  
   * Use CASCADE to truncate dependent tables automatically.
   * Without CASCADE, it will fail if foreign key constraints exist.
5. **Transactions**:  
   * If run inside a transaction, you can ROLLBACK to undo it.
   * If run outside, changes are permanent immediately.
6. **Sequence handling**: Use RESTART IDENTITY if you want primary key counters reset.

**Examples**

### **1. Basic TRUNCATE**

TRUNCATE TABLE employees;

Removes all rows from employees instantly.

### **2. Truncate Multiple Tables**

TRUNCATE TABLE employees, departments;

Both tables are emptied in a single command.

### **3. Restart Serial/Identity Columns**

TRUNCATE TABLE employees RESTART IDENTITY;

* Deletes all rows from employees.
* Resets the auto-increment primary key counter to **start from 1**.

### **4. Keep Serial/Identity Values**

TRUNCATE TABLE employees CONTINUE IDENTITY;

* Deletes all rows from employees.
* Keeps the existing sequence values (new rows will continue from the last number).

### **5. With CASCADE**

TRUNCATE TABLE orders CASCADE;

If orders have child tables (via foreign keys), they will also be truncated automatically.

### **6. Inside a Transaction (Rollback)**

BEGIN;

TRUNCATE TABLE employees;

-- Oops! We didn’t mean to do that

ROLLBACK;

Here, the table remains unchanged because we rolled back the transaction.

### **7. Prevent Truncate if Foreign Keys Exist**

TRUNCATE TABLE orders RESTRICT;

* If another table references orders, this will fail with an error.
* This is the default behavior if no CASCADE is specified.

## **Performance Notes**

* **TRUNCATE is almost instantaneous** because it only marks data pages as reusable.
* It does not fire DELETE triggers.
* Use with caution — **you cannot recover data unless you have backups**.

## **When to Use TRUNCATE**

When you need to **empty a table quickly**.  
When you don’t need to log each deleted row.  
When you want to reset sequences (with RESTART IDENTITY).  
When you can afford to lose **all rows** without filtering.

Do **not** use if:

* You need to remove **only some rows** (use DELETE instead).
* You want DELETE triggers to execute.
* You have complex foreign key relationships you don’t want to break.