# PROMPT-Give the detailed documentation of tcl command with its syntax and 2 examples in postgresql as a fresher

**🔍 What is TCL in SQL?**

**TCL** stands for **Transaction Control Language**. It is used to manage the **changes made by DML (Data Manipulation Language)** statements like INSERT, UPDATE, and DELETE.

🔁 A **transaction** is a group of SQL statements that execute together as a unit.  
A transaction should either **complete fully** or **not happen at all** (all-or-nothing).

**🧠 Why Use TCL Commands?**

TCL ensures:

* **Data consistency** and **integrity**
* **Safe execution** of multiple SQL statements
* You can **undo mistakes** before saving

**🔧 Common TCL Commands in PostgreSQL**

| **Command** | **Purpose** |
| --- | --- |
| BEGIN | Starts a new transaction |
| COMMIT | Saves all changes made in the transaction |
| ROLLBACK | Undoes all changes since BEGIN |
| SAVEPOINT | Creates a point in a transaction you can roll back to |
| ROLLBACK TO SAVEPOINT | Undoes part of a transaction up to a savepoint |
| RELEASE SAVEPOINT | Deletes a savepoint |

**📌 Syntax of TCL Commands**

**✅ Start a transaction**

sql

CopyEdit

BEGIN;

**✅ Commit a transaction (save changes)**

sql

CopyEdit

COMMIT;

**✅ Rollback a transaction (undo changes)**

sql

CopyEdit

ROLLBACK;

**✅ Create a savepoint**

sql

CopyEdit

SAVEPOINT savepoint\_name;

**✅ Rollback to a savepoint**

sql

CopyEdit

ROLLBACK TO SAVEPOINT savepoint\_name;

**✅ Release a savepoint**

sql

CopyEdit

RELEASE SAVEPOINT savepoint\_name;

**✅ Example 1: COMMIT and ROLLBACK**

**🔴 Scenario:**

Insert a new customer and a transaction. If everything is successful, save it; otherwise, cancel all.

sql

CopyEdit

BEGIN;

-- Step 1: Insert into customers

INSERT INTO customers (customer\_id, full\_name, balance, bank\_id)

VALUES ('C005', 'Priya Sen', 3000.00, 'BK001');

-- Step 2: Insert into transactions

INSERT INTO transactions (customer\_id, transaction\_date, amount, transaction\_type, description)

VALUES ('C005', CURRENT\_DATE, 3000.00, 'Deposit', 'Opening deposit');

-- Step 3: Save all changes

COMMIT;

**✅ Result:**

If both queries are correct, data is saved permanently.  
If there’s an error before COMMIT, no changes will be saved (use ROLLBACK).

**✅ Example 2: Using SAVEPOINT and ROLLBACK TO SAVEPOINT**

**🔴 Scenario:**

You want to deduct an amount from a customer’s balance, but may cancel the operation if a condition fails.

sql

CopyEdit

BEGIN;

-- Step 1: Start with an update

UPDATE customers

SET balance = balance - 500

WHERE customer\_id = 'C001';

-- Step 2: Set a savepoint

SAVEPOINT before\_bonus;

-- Step 3: Try to give bonus, but maybe it's not allowed

UPDATE customers

SET balance = balance + 1000

WHERE customer\_id = 'C001';

-- Oops! Bonus shouldn't be added. Let's undo it.

ROLLBACK TO SAVEPOINT before\_bonus;

-- Step 4: Finalize transaction

COMMIT;

**✅ Result:**

* ₹500 is deducted.
* ₹1000 bonus is **not** added (rolled back).
* Only the valid deduction is saved.

**⚠️ Tips for Freshers**

| **Tip** | **Explanation** |
| --- | --- |
| Always use BEGIN for grouped operations | Ensures all steps succeed or fail together |
| Use ROLLBACK before COMMIT if there's a mistake | Prevents bad data from being saved |
| Use SAVEPOINT in large transactions | Helps undo just part of the changes |
| Use SELECT \* FROM table before and after | To verify the changes before committing |

**🧾 Summary**

| **Command** | **Description** |
| --- | --- |
| BEGIN | Starts a transaction |
| COMMIT | Saves all changes since BEGIN |
| ROLLBACK | Cancels all changes since BEGIN |
| SAVEPOINT | Creates a rollback point |
| ROLLBACK TO SAVEPOINT | Reverts to a specific point |
| RELEASE SAVEPOINT | Deletes a savepoint |

**✅ Real-Life Use Cases of TCL**

* **Banking**: Transferring money between accounts (debit + credit)
* **E-commerce**: Order placement and inventory update
* **HR system**: Add employee and assign department
* **School database**: Enroll student + assign to classes