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

## **What is TCL in SQL?**

**TCL (Transaction Control Language)** commands are used to manage transactions in a database. A **transaction** is a **sequence of SQL operations** performed as a single logical unit of work.

In PostgreSQL, **transactions** ensure the **ACID properties**:

* **Atomicity**: All or nothing.
* **Consistency**: Data integrity is maintained.
* **Isolation**: Transactions are isolated.
* **Durability**: Once committed, changes are permanent.

## **Common TCL Commands in PostgreSQL**

| **Command** | **Description** |
| --- | --- |
| BEGIN | Starts a transaction block |
| COMMIT | Saves all changes made during the transaction |
| ROLLBACK | Reverts changes made during the transaction |
| SAVEPOINT | Sets a point within a transaction to rollback to |
| RELEASE | Deletes a savepoint |
| ROLLBACK TO | Rollbacks to a specific savepoint |

## 

## **PostgreSQL Syntax**

## **1. BEGIN**

The BEGIN command starts a new transaction block. All subsequent SQL statements will be part of this transaction until a COMMIT or ROLLBACK command is issued.

**BEGIN;**

### **2. COMMIT**

The COMMIT command permanently saves all changes made within the current transaction. Once a transaction is committed, the changes are visible to all other concurrent transactions and cannot be undone with ROLLBACK.

**COMMIT;**

### **3. ROLLBACK**

The ROLLBACK command is used to undo all changes made within the current transaction. This is useful for discarding a series of operations if an error occurs or if the changes are no longer needed. The database state is reverted to what it was before the BEGIN command.

**ROLLBACK;**

### **4. SAVEPOINT savepoint\_name**

The SAVEPOINT command creates a "savepoint" within a transaction. This allows for partial rollbacks to a specific point without rolling back the entire transaction.

**SAVEPOINT savepoint\_name;**

### **5. ROLLBACK TO savepoint\_name**

The ROLLBACK TO savepoint\_name command is used to undo all changes that occurred **after** the specified savepoint was created. Changes made before the savepoint remain intact within the transaction.

**ROLLBACK TO savepoint\_name;**

### 

### 

### **6. RELEASE SAVEPOINT**

The RELEASE SAVEPOINT command is used to remove a savepoint, making it no longer a valid rollback target. This can be done after a part of a transaction is successfully completed.

**RELEASE SAVEPOINT savepoint\_name;**

## **Practical Examples in PostgreSQL**

Let’s work with a simple table students:

CREATE TABLE students (

student\_id SERIAL PRIMARY KEY,

name VARCHAR(50),

marks INT

);

### **Example 1: Basic Transaction**

BEGIN;

INSERT INTO students (name, marks) VALUES ('Ravi', 80);

INSERT INTO students (name, marks) VALUES ('Neha', 85);

COMMIT;

Both rows will be permanently inserted.

### **Example 2: Rolling Back a Transaction**

BEGIN;

INSERT INTO students (name, marks) VALUES ('Ajay', 75);

INSERT INTO students (name, marks) VALUES ('Simran', NULL); -- let's say this violates a NOT NULL constraint

ROLLBACK;

If any error occurs (e.g., null in a NOT NULL column), all changes are rolled back.

### **Example 3: Using SAVEPOINT**

BEGIN;

INSERT INTO students (name, marks) VALUES ('Kiran', 70);

SAVEPOINT sp1;

INSERT INTO students (name, marks) VALUES ('Deepak', NULL); -- violates NOT NULL

-- Error happens

ROLLBACK TO sp1; -- undo only Deepak's row

INSERT INTO students (name, marks) VALUES ('Deepak', 90);

COMMIT;

The transaction only rolls back to the point just before the error, not the entire transaction.

### **Example 4: Using RELEASE SAVEPOINT**

BEGIN;

INSERT INTO students (name, marks) VALUES ('Meena', 78);

SAVEPOINT sp1;

INSERT INTO students (name, marks) VALUES ('Raj', 88);

RELEASE SAVEPOINT sp1;

COMMIT;

Once sp1 is released, you can't rollback to it anymore.

## **Why Use Transactions?**

* To **group multiple operations** into one atomic unit.
* To **maintain data integrity** during errors.
* To **safely rollback** if something goes wrong.

## **Best Practices**

* Always use transactions for **multiple-step DML operations**.
* Use SAVEPOINT for **complex transactions** where partial rollback might be required.
* Always ROLLBACK or COMMIT explicitly after BEGIN.