**Prompt: Give me a detailed explanation of TCL in SQL for learner with syntax and 2 examples using COMMIT,ROLLBACK and SAVEPOINT.**

**TCL in SQL: A Beginner's Guide**

TCL, which stands for **Transaction Control Language**, is a set of SQL commands used to manage transactions within a database. A **transaction** is a single logical unit of work, which consists of one or more SQL statements. Think of a transaction like a single operation that must either completely succeed or completely fail. For example, transferring money from one bank account to another involves two steps: withdrawing from the first account and depositing into the second. Both steps must succeed for the transaction to be considered complete. If one step fails, both must be undone. TCL commands allow you to control this all-or-nothing behavior.

The three main TCL commands are:

* **COMMIT**: Saves all changes made in a transaction to the database permanently.
* **ROLLBACK**: Undoes all changes made in a transaction since the last COMMIT or ROLLBACK.
* **SAVEPOINT**: Creates a temporary marker within a transaction, allowing you to partially roll back a transaction to that specific point.

**Understanding the Syntax**

A transaction starts implicitly with the first SQL statement (like INSERT, UPDATE, or DELETE) after a COMMIT or ROLLBACK. You can also start one explicitly with BEGIN TRANSACTION;.

* **COMMIT;:** Commits the entire transaction. All changes become permanent and visible to other users.
* **ROLLBACK;**: Reverts the entire transaction. The database state is restored to what it was before the transaction began.
* **SAVEPOINT savepoint\_name;**: Creates a named savepoint.
* **ROLLBACK TO savepoint\_name;**: Reverts the transaction only up to the specified SAVEPOINT, leaving subsequent changes untouched.

**Example 1: COMMIT and ROLLBACK**

Imagine we have a products table. We'll start a transaction to update some records.

SQL

-- Assume the "products" table exists with some data

BEGIN TRANSACTION;

-- SQL statement 1: Update the price of a product

UPDATE products

SET price = 12.50

WHERE product\_id = 101;

-- SQL statement 2: Delete a product

DELETE FROM products

WHERE product\_id = 102;

At this point, the changes are only temporary and are not visible to other database users.

**Scenario A: Everything looks good.**

If we are happy with the changes, we can save them permanently using COMMIT.

SQL

COMMIT;

Now the changes are permanent. The new price for product 101 and the deletion of product 102 are saved in the database.

**Scenario B: Something went wrong.**

If we realize we made a mistake (e.g., we deleted the wrong product), we can use ROLLBACK to undo all the changes since the BEGIN TRANSACTION statement.

SQL

ROLLBACK;

After this command, the database will be in the exact same state as it was before we ran the UPDATE and DELETE statements. The price of product 101 will be unchanged, and product 102 will still exist.

**Example 2: Using SAVEPOINT for Partial Rollback**

SAVEPOINT is useful for complex transactions where you might need to undo only a part of the work without affecting other changes.

Let's continue with our products table and perform several operations.

SQL

BEGIN TRANSACTION;

-- Operation 1: Insert a new product

INSERT INTO products (product\_name, price) VALUES ('Monitor', 150.00);

-- Create a savepoint here

SAVEPOINT after\_insert;

-- Operation 2: Update the price of an existing product

UPDATE products

SET price = 8.75

WHERE product\_id = 103;

-- Operation 3: Another update that we might need to undo

UPDATE products

SET price = 250.00

WHERE product\_id = 104;

**Scenario C: A partial rollback.**

Suppose we realize that the second UPDATE (for product\_id = 103) was an error, but the INSERT and the third UPDATE are correct. We don't want to undo the entire transaction.

SQL

ROLLBACK TO after\_insert;

After this command, the database will revert only the changes made **after** the after\_insert savepoint. This means:

* The INSERT of the 'Monitor' is still in the transaction and will be committed later.
* The UPDATE of product 103 is undone.
* The UPDATE of product 104 is also undone because it happened after the SAVEPOINT.

Finally, to make the remaining changes permanent, we would run:

SQL

COMMIT;

This will save the insertion of the 'Monitor' to the database, while the two update operations are discarded.