**Assignment 8 : Begin a transaction, perform a series of INSERTs into 'orders', setting a SAVEPOINT after each, rollback to the second SAVEPOINT, and COMMIT the overall transaction.**

The SQL code demonstrating a transaction with SAVEPOINTs and selective rollback:

SQL

BEGIN TRANSACTION;

-- First INSERT

INSERT INTO orders (customer\_id, order\_date, total\_amount)

VALUES (123, CURDATE(), 100.00);

SAVEPOINT first\_order; -- Set a SAVEPOINT after the first insert

-- Second INSERT

INSERT INTO orders (customer\_id, order\_date, total\_amount)

VALUES (456, CURDATE(), 200.00);

SAVEPOINT second\_order; -- Set a SAVEPOINT after the second insert

-- Third INSERT (optional, for demonstration)

INSERT INTO orders (customer\_id, order\_date, total\_amount)

VALUES (789, CURDATE(), 300.00);

-- Rollback to the second SAVEPOINT (keeps first insert)

ROLLBACK TRANSACTION TO SAVEPOINT second\_order;

COMMIT TRANSACTION;

**Explanation:**

1. BEGIN TRANSACTION: Initiates the transaction.
2. **First INSERT:** Inserts the first order record.
3. SAVEPOINT first\_order: Creates a SAVEPOINT named first\_order at this point.
4. **Second INSERT:** Inserts the second order record.
5. SAVEPOINT second\_order: Creates another SAVEPOINT named second\_order.
6. **Optional Third INSERT:** This line demonstrates inserting a third order but is commented out. You can uncomment it if you want to include it and then rollback to keep only the first order.
7. ROLLBACK TRANSACTION TO SAVEPOINT second\_order: This statement rolls back the transaction to the second\_order SAVEPOINT. This undoes the third insert (if uncommented) and keeps the first two inserts.
8. COMMIT TRANSACTION: Finalizes the transaction, making the first two inserted orders permanent.

**Points to Remember:**

* This example demonstrates selective rollback using SAVEPOINTs. You can adjust the SAVEPOINT names and rollback points based on your specific needs.
* Ensure your database system supports SAVEPOINT functionality.