**🔹 25 TCL Practice Questions in MySQL**

**Basic Level (Q1–Q8)**

1. Start a transaction and insert a new row into a table. Then **rollback** so the change is undone.
2. Start a transaction, insert a row, and then **commit** it. Verify that the row is permanently saved.
3. Disable **autocommit**, insert a record, then rollback to confirm it doesn’t persist.
4. Re-enable **autocommit** and insert a record. Check if the record is saved without commit.
5. Insert multiple rows in a transaction. Use **commit** to save them.
6. Insert multiple rows in a transaction. Use **rollback** to discard them.
7. Perform an update inside a transaction, rollback it, and check whether the original value remains.
8. Perform a delete inside a transaction, rollback it, and confirm the row still exists.

**Intermediate Level (Q9–Q17)**

1. Update two different rows inside a transaction. Use **commit** to save both changes.
2. Update two rows inside a transaction. Use **rollback** to discard both changes.
3. Use **SAVEPOINT** after an insert, then perform another insert. Rollback to the savepoint. Check which rows remain.
4. Start a transaction, update a salary in an employees table, create a savepoint, update again, then rollback to the savepoint.
5. Insert 3 records in a transaction with 2 savepoints. Rollback only to the second savepoint.
6. Create and release a savepoint. Try rolling back to the released savepoint and note what happens.
7. Disable autocommit, perform multiple inserts, then use **commit**. Verify all are saved.
8. Disable autocommit, perform multiple inserts, then use **rollback**. Verify none are saved.
9. Demonstrate the difference between SET AUTOCOMMIT = 0 and START TRANSACTION.

**Advanced Level (Q18–Q25)**

1. Simulate a bank transfer: Deduct money from one account and add to another inside a transaction. Commit it.
2. Simulate a bank transfer: Deduct money but do not add it. Rollback to ensure no loss occurs.
3. Use multiple savepoints in a single transaction (sp1, sp2, sp3). Rollback to sp2 and commit the rest.
4. Insert rows in a transaction with nested savepoints. Release one savepoint and rollback to another.
5. Perform an update, set a savepoint, perform a delete, then rollback to the savepoint. Confirm only the delete was undone.
6. Perform an update and commit. Then try rollback. Verify that committed changes cannot be undone.
7. Simulate concurrent transactions: Open two sessions, update the same row in each, and observe the effect of commit/rollback.
8. Write a transaction that includes **INSERT, UPDATE, and DELETE** operations. Use savepoints and rollback to undo only specific changes while keeping others.