**What is a Database Transaction?**

A **transaction** is a group of one or more SQL operations (such as INSERT, UPDATE, or DELETE) that are treated as a **single logical unit of work**.

You can think of it as:

“Either all steps of the transaction are completed successfully, or none of them are applied at all.”

**Simple Real-life Example**

Imagine you’re transferring Rs.500 from **Account A** to **Account B**.

This requires two steps:

1. Deduct Rs.500 from Account A.
2. Add Rs.500 to Account B.

If the first step succeeds but the second one fails (say, due to a network problem), money could disappear! To avoid that, both steps are grouped into a **transaction** — meaning either both steps are done or both are cancelled.

**How It Looks in a Database**

START TRANSACTION;

UPDATE accounts SET balance = balance - 500 WHERE name = 'A';

UPDATE accounts SET balance = balance + 500 WHERE name = 'B';

COMMIT;

If something goes wrong during the process, you can use:

ROLLBACK;

The **COMMIT** command makes all the changes permanent, while **ROLLBACK** cancels everything done in that transaction and brings the database back to its previous state.

**In SQLAlchemy or Any Programming Language**

When you call session.commit() in SQLAlchemy, all pending operations are permanently saved to the database.

If any error occurs and you call session.rollback(), all the changes made since the last commit are undone — as if they never happened.

**ACID Properties of a Transaction**

Transactions follow four key properties known as **ACID**:

1. **Atomicity** means that the entire transaction is treated as a single unit — either all operations happen or none happen. There’s no halfway result.
2. **Consistency** ensures that the database remains in a valid state before and after the transaction. All rules, constraints, and relationships remain correct (for example, no negative balance).
3. **Isolation** means that multiple transactions happening at the same time don’t affect each other. For instance, if two users are transferring money simultaneously, one transaction doesn’t see the incomplete results of another.
4. **Durability** guarantees that once a transaction is committed, its result is permanently stored in the database, even if the system crashes or restarts afterward.

**Summary**

A **transaction** is like performing several related operations inside a “safety box.”  
If everything goes right, we **commit** and make the changes permanent.  
If something goes wrong, we **rollback** and undo the partial work.

The **ACID** principles — Atomicity, Consistency, Isolation, and Durability — ensure that the database remains accurate, reliable, and safe even in the face of errors or failures.