1. Transaction in SQL

What is a Transaction?

A **transaction** is a **logical unit of work** in SQL that consists of one or more operations. Either **all operations succeed**, or **none are applied**.
The main goal is to **ensure data integrity and consistency**.

ACID Properties of Transactions

1. **Atomicity** – All operations succeed as a single unit, or none are applied.
   Example: In a money transfer, if the amount is deducted from one account but not added to the other, the transaction should roll back.
2. **Consistency** – The database moves from one valid state to another.
   Example: Any primary key or unique constraints must still be valid after the transaction.
3. **Isolation** – Transactions do not interfere with each other.
   Example: If one user transfers money while another checks the balance, the second user should not see partial changes until the transaction completes.
4. **Durability** – Once committed, changes are permanent, even if the system crashes.

```sql
BEGIN TRANSACTION;  -- Start a transaction

-- Example of two operations
UPDATE Account
SET Balance = Balance - 500
WHERE AccountID = 101;

UPDATE Account
SET Balance = Balance + 500
WHERE AccountID = 102;

COMMIT;  -- Save changes if all operations succeed

ROLLBACK;  -- Undo all changes if any operation fails
```

Practical Use Cases of Transactions

- **Banking:** Transferring money between accounts.
- **E-commerce:** Adding products to cart and updating inventory during checkout.
- **University systems:** Registering a student for multiple courses at once; all registrations must succeed or fail together.

Why Transactions are Important

- Protects data from human or technical errors.
- Prevents inconsistencies across multiple tables.
- Ensures complex operations complete safely and reliably.

2. Stored Procedure in SQL

What is a Stored Procedure?

A **stored procedure** is a **precompiled collection of SQL statements stored in the database** that can be executed whenever needed.
It can include:

- SELECT, INSERT, UPDATE, DELETE
- Conditional logic (IF / ELSE)
- Loops (FOR, WHILE)
- Input parameters for flexibility

Advantages of Stored Procedures

1. **Reusability:** Write once, execute multiple times.
2. **Performance:** Precompiled, so execution is faster.
3. **Security:** Users can execute procedures without direct access to tables.
4. **Error Reduction:** Centralized logic reduces coding mistakes.

```sql
-- Retrieve customer data by ID
CREATE PROCEDURE GetCustomerByID
    @CustomerID INT
AS
BEGIN
    SELECT FullName, Email, Phone
    FROM Customer
    WHERE CustomerID = @CustomerID;
END;
```

Executing the Stored Procedure:

EXEC GetCustomerByID @CustomerID = 1

Real-Life Uses:

- **Banking:** Safe money transfers.
- **Inventory systems:** Updating stock and order details together.

- **University systems:** Registering a student for multiple courses while checking prerequisites.