TRANSACTION

What is a Transaction in SQL?

A **transaction** is a group of one or more SQL statements that are executed as a **single unit of work**.

- Either **all statements succeed**
- Or **all statements fail** (no partial changes)

Transactions help keep data **safe and consistent**.

When Do We Use Transactions?

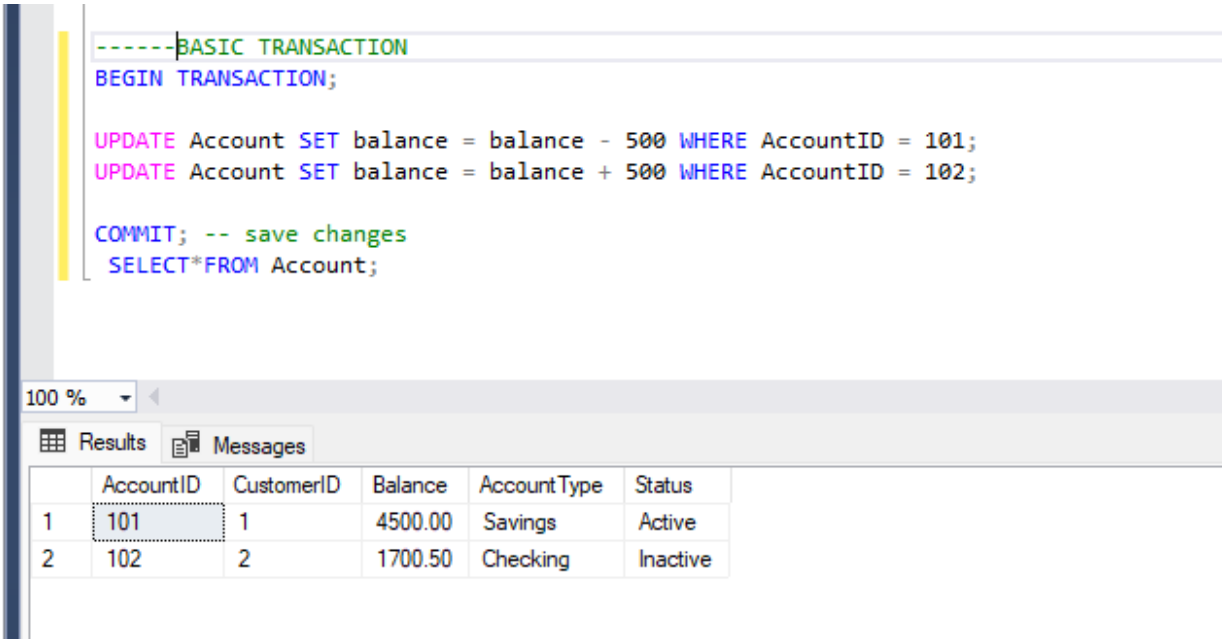
- Money transfers
- Reservations
- Multiple related updates
- Any operation that must be **all-or-nothing**

1 Transaction Types (Order / Categories

Type	Description	Example
Basic / Simple Transaction	A single transaction with commit or rollback.	`BEGIN TRANSACTION; UPDATE Accounts SET balance=balance-100 WHERE id=1; COMMIT;`

Read-Only Transaction	Only reads data, does not modify.	`SELECT * FROM Accounts;`
Read-Write Transaction	Reads and updates data.	`UPDATE Accounts SET balance=balance+100 WHERE id=2;`
Batch / Multi-Statement Transaction	Multiple operations executed as a unit.	See example below
Distributed Transaction	Transaction spans multiple databases.	Transfer between DB1.Accounts → DB2.Accounts
Online Transaction (OLTP)	Short, fast transactions for real-time operations.	ATM withdrawal, booking tickets

A) Basic Transaction Flow



- ✅ `COMMIT` → all changes saved
- ❌ If error → can use `ROLLBACK` to undo all changes

B) ROLLBACK

-Used to **undo the transaction** if something goes wrong.

```
---- ROLLBACK

 BEGIN TRANSACTION;
UPDATE Account SET balance = balance - 500 WHERE AccountID = 101;
ROLLBACK;-- Something goes wrong
 -- undo changes

SELECT*FROM Account;
```

100 %

Results | Messages

	AccountID	CustomerID	Balance	AccountType	Status
1	101	1	4500.00	Savings	Active
2	102	2	1700.50	Checking	Inactive

C)TRY…CATCH with Transaction

Handles errors and safely rolls back if needed.

```
--------TRY…CATCH with Transaction
BEGIN TRY
    BEGIN TRANSACTION;

    UPDATE Account SET balance = balance - 500 WHERE AccountID = 101;
    UPDATE Account SET balance = balance + 500 WHERE AccountID = 102;

    COMMIT; -- success
END TRY
BEGIN CATCH
    ROLLBACK; -- undo all changes
    PRINT 'Error: ' + ERROR_MESSAGE();
END CATCH;

SELECT*FROM Account
```

100 %

Results | Messages

	AccountID	CustomerID	Balance	AccountType	Status
1	101	1	4000.00	Savings	Active
2	102	2	2200.50	Checking	Inactive

✅ Benefits:

- Prevents partial updates
- Handles errors automatically
- Keeps data consistent (ACID)

Stored Procedures

1) What is a Stored Procedure?

A **Stored Procedure** is a **precompiled set of SQL statements** stored in the database.

- You can **execute it multiple times** without rewriting the SQL code.
- Improves performance and **reusability**.
- Can include **logic**, **parameters**, and **transactions**.

2) Benefits

A) **Reusability** → Write once, use many times.
B) **Faster execution** → Precompiled in the database.
C) **Security** → Users can execute without direct access to tables.
D) **Easier maintenance** → Change code in one place only.