1. CREATE DATABASE & USE IT

```
IF DB_ID('BankDemo') IS NOT NULL
    DROP DATABASE BankDemo;
GO

CREATE DATABASE BankDemo;
GO

USE BankDemo;
GO
```

---

2. CREATE TABLES (BANKING DOMAIN)

Branches

```
CREATE TABLE dbo.Branches (
    BranchID INT IDENTITY(1,1) PRIMARY KEY,
    BranchName NVARCHAR(100) NOT NULL,
    City NVARCHAR(50) NOT NULL,
    IFSC CHAR(11) NOT NULL UNIQUE
);
```

Customers

```
CREATE TABLE dbo.Customers (
    CustomerID INT IDENTITY(1,1) PRIMARY KEY,
    FirstName NVARCHAR(50) NOT NULL,
    LastName NVARCHAR(50) NOT NULL,
    Email NVARCHAR(255) NULL UNIQUE,
    DateOfBirth DATE NULL,
    CreatedAt DATETIME2 NOT NULL DEFAULT SYSUTCDATETIME()
);
```

Accounts

```
CREATE TABLE dbo.Accounts (
    AccountID BIGINT IDENTITY(1000000000,1) PRIMARY KEY,
    CustomerID INT NOT NULL,
    BranchID INT NOT NULL,
    AccountType CHAR(1) NOT NULL CHECK (AccountType IN ('S','C')),
    Balance MONEY NOT NULL DEFAULT (0),
    IsActive BIT NOT NULL DEFAULT(1),
    OpenedOn DATE NOT NULL DEFAULT (GETDATE()),
    CONSTRAINT FK_Accounts_Customers FOREIGN KEY (CustomerID) REFERENCES
dbo.Customers(CustomerID),
    CONSTRAINT FK_Accounts_Branches FOREIGN KEY (BranchID) REFERENCES
dbo.Branches(BranchID)
);
```

Transactions

```
CREATE TABLE dbo.Transactions (
    TransactionID BIGINT IDENTITY(1,1) PRIMARY KEY,
    AccountID BIGINT NOT NULL,
    TranDate DATETIME2 NOT NULL DEFAULT SYSUTCDATETIME(),
    Amount MONEY NOT NULL CHECK (Amount <> 0),
    TranType CHAR(1) NOT NULL CHECK (TranType IN ('D','W','T')),
    Description NVARCHAR(250) NULL,
    RelatedAccountID BIGINT NULL,
    CONSTRAINT FK_Transactions_Accounts FOREIGN KEY (AccountID) REFERENCES
dbo.Accounts(AccountID)
);
```

---

3. INSERT SAMPLE DATA

Insert Branches

```
INSERT INTO dbo.Branches (BranchName, City, IFSC)
VALUES
('Central', 'Mumbai', 'BKID0000001'),
('Andheri', 'Mumbai', 'BKID0000002'),
('Bandra', 'Mumbai', 'BKID0000003');
```

Insert Customers

```
INSERT INTO dbo.Customers (FirstName, LastName, Email, DateOfBirth)
VALUES
('Amit','Kumar','amit.k@example.com','1985-06-15'),
('Neha','Sharma','neha.s@example.com','1990-02-21'),
('Ravi','Patel', NULL, '1979-11-03'),
('Sara','Khan','sara.k@example.com','1992-09-10');
```

Insert Accounts

```
INSERT INTO dbo.Accounts (CustomerID, BranchID, AccountType, Balance)
VALUES
(1,1,'S',15000),
(1,2,'C',50000),
(2,2,'S',8000),
(3,1,'S',2500),
(4,3,'C',90000);
```

Insert Transactions

```
INSERT INTO dbo.Transactions (AccountID, Amount, TranType, Description)
VALUES
(1000000000, 5000,'D','Salary'),
(1000000001,-2000,'W','ATM Withdrawal'),
(1000000002, 3000,'D','Gift'),
```

```
(1000000003,-1000,'W','Bill Payment'),
(1000000004, 15000,'D','Online Transfer');
```

🔵 4. SELECT, FILTERING, ORDERING

Basic SELECT

```
SELECT * FROM dbo.Customers;
```

DISTINCT

```
SELECT DISTINCT City FROM dbo.Branches;
```

WHERE + ORDER BY

```
SELECT FirstName, LastName, DateOfBirth
FROM dbo.Customers
WHERE DateOfBirth > '1988-01-01'
ORDER BY LastName;
```

TOP / OFFSET-FETCH

```
SELECT TOP 3 * FROM dbo.Accounts ORDER BY Balance DESC;

SELECT AccountID, Balance
FROM dbo.Accounts
ORDER BY AccountID
OFFSET 2 ROWS FETCH NEXT 2 ROWS ONLY;
```

Filtering Operators

```
SELECT * FROM dbo.Accounts WHERE AccountType IN ('S','C');

SELECT * FROM dbo.Accounts WHERE Balance BETWEEN 5000 AND 60000;

SELECT * FROM dbo.Customers WHERE Email LIKE '%@example.com';
```

```
SELECT * FROM dbo.Customers WHERE (FirstName='Amit' OR FirstName='Sara') AND
Email IS NOT NULL;
```

🔵 5. JOINS (BANKING EXAMPLES)

INNER JOIN

```
SELECT a.AccountID, a.Balance, c.FirstName, b.BranchName
FROM dbo.Accounts a
JOIN dbo.Customers c ON a.CustomerID=c.CustomerID
JOIN dbo.Branches b ON a.BranchID=b.BranchID;
```

LEFT JOIN

```
SELECT c.FirstName, a.AccountID, a.Balance
FROM dbo.Customers c
LEFT JOIN dbo.Accounts a ON c.CustomerID=a.CustomerID;
```

FULL JOIN

```
SELECT c.CustomerID, a.AccountID
FROM dbo.Customers c
FULL JOIN dbo.Accounts a ON c.CustomerID=a.CustomerID;
```

CROSS JOIN

```
SELECT BranchName, Rate
FROM dbo.Branches
CROSS JOIN (VALUES (4.5),(6.0)) AS r(Rate);
```

SELF JOIN

```
SELECT c1.FirstName, c2.FirstName
FROM dbo.Customers c1
JOIN dbo.Customers c2 ON c1.LastName=c2.LastName AND
c1.CustomerID<>c2.CustomerID;
```

🔵 6. FUNCTIONS (STRING/DATE/NUMERIC + CASE)

String Functions

```
SELECT FirstName,
       LEFT(FirstName,1) AS Initial,
       LEN(Email) AS EmailLength,
       CONCAT(FirstName,' ',LastName) AS FullName
FROM dbo.Customers;
```

Date/Time Functions

```
SELECT GETDATE() AS Today,
       DATEADD(day,30,GETDATE()) AS After30Days,
       DATEDIFF(year,DateOfBirth,GETDATE()) AS Age
FROM dbo.Customers;
```

Numeric Functions

```
SELECT Balance,
       ROUND(Balance,0) AS Rounded,
       CEILING(Balance) AS CeilVal,
       FLOOR(Balance) AS FloorVal
FROM dbo.Accounts;
```

CASE Expression

```
SELECT AccountID, Balance,
       CASE
         WHEN Balance < 5000 THEN 'Low'
         WHEN Balance < 50000 THEN 'Medium'
         ELSE 'High'
       END AS Category
FROM dbo.Accounts;
```

🔵 7. SUBQUERIES

Single-row subquery

```
SELECT *
FROM dbo.Accounts
WHERE BranchID = (SELECT BranchID FROM dbo.Branches WHERE
BranchName='Central');
```

Multi-row subquery

```
SELECT *
FROM dbo.Accounts
WHERE CustomerID IN (SELECT CustomerID FROM dbo.Customers WHERE Email LIKE
'%example.com');
```

🔵 8. VIEWS

Simple View

```
CREATE VIEW dbo.v_CustomerAccount AS
SELECT c.FirstName, c.LastName, a.AccountID, a.Balance
FROM dbo.Customers c
JOIN dbo.Accounts a ON c.CustomerID=a.CustomerID;
GO
```

```
SELECT * FROM dbo.v_CustomerAccount;
```

Aggregated View

```
CREATE VIEW dbo.v_BranchSummary AS
SELECT b.BranchName, COUNT(a.AccountID) AS TotalAcc, SUM(a.Balance) AS
TotalBalance
FROM dbo.Branches b
LEFT JOIN dbo.Accounts a ON b.BranchID=a.BranchID
GROUP BY b.BranchName;
GO
```

🔵 9. STORED PROCEDURES

Procedure: Get Accounts by Customer

```
CREATE PROCEDURE dbo.usp_GetAccountsByCustomer
    @CustomerID INT
AS
BEGIN
    SELECT * FROM dbo.Accounts WHERE CustomerID=@CustomerID;
END;
GO
```

Procedure: Deposit

```
CREATE PROCEDURE dbo.usp_Deposit
    @AccountID BIGINT,
    @Amount MONEY
AS
BEGIN
    BEGIN TRY
        BEGIN TRAN;
        UPDATE dbo.Accounts SET Balance = Balance + @Amount WHERE
AccountID=@AccountID;

        INSERT INTO dbo.Transactions(AccountID,Amount,TranType,Description)
        VALUES(@AccountID,@Amount,'D','Deposit');
```

```
        COMMIT TRAN;
    END TRY
    BEGIN CATCH
        ROLLBACK TRAN;
        THROW;
    END CATCH
END;
GO
```

---

10. INDEXES

Non-Clustered Index

```
CREATE NONCLUSTERED INDEX IX_Accounts_CustomerID
ON dbo.Accounts(CustomerID)
INCLUDE (Balance);
```

Filtered Index (Active Accounts Only)

```
CREATE NONCLUSTERED INDEX IX_Accounts_Active
ON dbo.Accounts(IsActive)
WHERE IsActive = 1;
```