📘 STUDENT PRACTICE EXERCISES

Part A — Basic to Intermediate

1. List all branches located in Mumbai.
2. Find customers whose email ends with `example.com`.
3. Display all accounts with a balance between 10,000 and 60,000.
4. Show top 2 highest balance accounts.
5. Write a query to find customers who do **not** have an account.
6. Get list of all accounts and their branch names (JOIN).
7. Display customers and their total balance (subquery).
8. Show accounts categorized as "Low", "Medium", "High" using CASE.
9. Write a CTE to show customers older than 30 years.
10. Create a view that shows AccountID, Customer FullName, and BranchName.

Part B — Advanced / Joins / Functions

11. Find customers having more than one account.
12. Show all accounts with at least one transaction.
13. Display customers with account balance greater than branch average.
14. Write a TVF (table-valued function) to return all accounts for a branch.
15. Create a stored procedure to withdraw money securely using TRY…CATCH.

🟢 ANSWER KEY (INSTRUCTOR)

1.

```
SELECT * FROM dbo.Branches WHERE City='Mumbai';
```

2.

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

3.

```
SELECT * FROM dbo.Accounts WHERE Balance BETWEEN 10000 AND 60000;
```

4.

```
SELECT TOP 2 * FROM dbo.Accounts ORDER BY Balance DESC;
```

5.

```
SELECT c.*
FROM dbo.Customers c
LEFT JOIN dbo.Accounts a ON c.CustomerID=a.CustomerID
WHERE a.CustomerID IS NULL;
```

6.

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

7.

```
SELECT CustomerID,
       (SELECT SUM(Balance) FROM dbo.Accounts a WHERE a.CustomerID=c.CustomerID)
AS TotalBalance
FROM dbo.Customers c;
```

8.

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

9.

```
WITH CTE AS (
    SELECT *, DATEDIFF(year,DateOfBirth,GETDATE()) AS Age
    FROM dbo.Customers
)
SELECT * FROM CTE WHERE Age > 30;
```

10.

```
CREATE VIEW v_AccountDetails AS
SELECT a.AccountID,
       CONCAT(c.FirstName,' ',c.LastName) AS FullName,
       b.BranchName
FROM dbo.Accounts a
JOIN dbo.Customers c ON a.CustomerID=c.CustomerID
JOIN dbo.Branches b ON a.BranchID=b.BranchID;
```

11.

```
SELECT CustomerID, COUNT(*) AS AccountCount
FROM dbo.Accounts
GROUP BY CustomerID
HAVING COUNT(*) > 1;
```

12.

```
SELECT DISTINCT a.*
FROM dbo.Accounts a
JOIN dbo.Transactions t ON a.AccountID=t.AccountID;
```

13.

```
SELECT a.*
FROM dbo.Accounts a
JOIN (
      SELECT BranchID, AVG(Balance) AS AvgBal FROM dbo.Accounts GROUP BY BranchID
     ) b ON a.BranchID=b.BranchID
WHERE a.Balance > b.AvgBal;
```

14. (TVF)

```
CREATE FUNCTION dbo.ufn_GetAccountsByBranch(@BranchID INT)
RETURNS TABLE
AS RETURN (
    SELECT * FROM dbo.Accounts WHERE BranchID=@BranchID
);
```

15. (Withdrawal SP)

```
CREATE PROCEDURE dbo.usp_Withdraw
    @AccountID BIGINT,
    @Amount MONEY
AS
BEGIN
    BEGIN TRY
        BEGIN TRAN;

        UPDATE dbo.Accounts
        SET Balance = Balance - @Amount
        WHERE AccountID=@AccountID AND Balance >= @Amount;

        IF @@ROWCOUNT = 0
            THROW 50001, 'Insufficient Funds', 1;

        INSERT INTO dbo.Transactions(AccountID,Amount,TranType,Description)
        VALUES(@AccountID,-@Amount,'W','Withdrawal');

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