5.Create a database for Banking and Write SQL queries for group functions.

Create Accounts Table

```
CREATE TABLE Accounts (
    AccountID INT PRIMARY KEY,
    Name VARCHAR(100),
    Balance INT
);
-- Create Transactions Table
CREATE TABLE Transactions (
    TransactionID INT PRIMARY KEY,
    AccountID INT,
    Amount INT,
    Type VARCHAR(10), -- 'Deposit' or 'Withdraw'
    FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)
);
```

Insert Data

```
-- Insert Accounts
INSERT INTO Accounts (AccountID, Name, Balance) VALUES
(1, 'Alice', 5000),
(2, 'Bob', 7000),
(3, 'Charlie', 3000);
--  Insert Transactions
INSERT INTO Transactions (TransactionID, AccountID, Amount, Type) VALUES
(101, 1, 2000, 'Deposit')
(102, 1, 500, 'Withdraw'),
(103, 2, 1500, 'Deposit'),
(104, 3, 1000, 'Withdraw');
```

Apply Group Functions

1] SUM() - Total Balance in All Accounts

SELECT SUM(Balance) AS TotalBalance FROM Accounts;

Output:

```
+--------------+
| TotalBalance |
+--------------+
| 15000        |
+--------------+
```

2] AVG() - Average Account Balance

SELECT AVG(Balance) AS AvgBalance FROM Accounts;

Output:

```
+------------+
| AvgBalance |
+------------+
| 5000       |
+------------+
```

3] COUNT() - Total Transactions

SELECT COUNT(TransactionID) AS TotalTransactions FROM Transactions;

Output:

```
+------------------+
| TotalTransactions |
+------------------+
| 4                |
+------------------+
```

4] MAX() - Maximum Deposit

SELECT MAX(Amount) AS MaxDeposit FROM Transactions WHERE Type = 'Deposit';

Output:

```
+------------+
| MaxDeposit |
+------------+
| 2000       |
+------------+
```

5] MIN() - Minimum Withdrawal

SELECT MIN(Amount) AS MinWithdrawal FROM Transactions WHERE Type = 'Withdraw';

Output:

```
+---------------+
| MinWithdrawal |
+---------------+
| 500           |
+---------------+
```

6. Create a database for Library and Write SQL queries for sub queries, nested queries.

Create Table

```
CREATE TABLE Books (
    BookID INT PRIMARY KEY,
    Title VARCHAR(100),
    Author VARCHAR(100)
);
-- Create Members Table
CREATE TABLE Members (
    MemberID INT PRIMARY KEY,
    Name VARCHAR(100)
);
-- Create BorrowedBooks Table
CREATE TABLE BorrowedBooks (
    BorrowID INT PRIMARY KEY,
    MemberID INT,
    BookID INT,
    FOREIGN KEY (MemberID) REFERENCES Members(MemberID),
    FOREIGN KEY (BookID) REFERENCES Books(BookID)
);
```

Insert Data

```
-- Insert Books
INSERT INTO Books (BookID, Title, Author) VALUES
(1, 'SQL Basics', 'John Smith'),
(2, 'Python Programming', 'Alice Brown'),
(3, 'C Programming', 'David Lee');
```

```
-- Insert Members
INSERT INTO Members (MemberID, Name) VALUES
(101, 'Robert'),
(102, 'Sophia'),
(103, 'Michael');
-- Insert Borrowed Books
INSERT INTO BorrowedBooks (BorrowID, MemberID, BookID) VALUES
(1001, 101, 1),
(1002, 102, 2),
(1003, 103, 3);
```

Simple SQL Queries Using Subqueries & Nested Queries

1] Find Members Who Borrowed (Subquery in WHERE)

```
SELECT Name FROM Members
WHERE MemberID IN (SELECT MemberID FROM BorrowedBooks WHERE BookID =
(SELECT BookID FROM Books WHERE Title = 'SQL Basics'));
```

Output:

```
+--------+
| Name   |
+--------+
| Robert |
+--------+
```

2] Find Books Borrowed by Sophia (Nested Query in WHERE)

```
SELECT Title FROM Books
WHERE BookID IN (SELECT BookID FROM BorrowedBooks WHERE MemberID =
(SELECT MemberID FROM Members WHERE Name = 'Sophia'));
```

Output:

```
+----------------------+
| Title                |
+----------------------+
| Python Programming   |
+----------------------+
```

3] Find Books That Have Been Borrowed (Using IN)

```
SELECT Title FROM Books
WHERE BookID IN (SELECT BookID FROM BorrowedBooks);
```

Output:

```
+----------------------+
| Title                |
+----------------------+
| SQL Basics           |
| Python Programming   |
| C Programming        |
+----------------------+
```

8. Create a database for Orders and Write SQL queries to create views.

Create Orders Table

```
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerName VARCHAR(100),
    OrderDate DATE,
    Amount INT
);
```

Insert Orders

```
INSERT INTO Orders (OrderID, CustomerName, OrderDate, Amount) VALUES
(1, 'Alice', '2025-03-01', 500),
(2, 'Bob', '2025-03-02', 700),
(3, 'Charlie', '2025-03-03', 300),
(4, 'David', '2025-03-04', 1000);
```

Create Views for Orders Data

1] Create a View for All Orders

```
CREATE VIEW AllOrders AS
SELECT * FROM Orders;
```

View the Data

```
SELECT * FROM AllOrders;
```

Output: *(Same as Orders Table)*

2] Create a View for Orders Above 500

```
CREATE VIEW HighValueOrders AS
SELECT * FROM Orders WHERE Amount > 500;
```

View the Data

```
SELECT * FROM HighValueOrders;
```

Output:

```
+---------+-------------+------------+--------+
| OrderID | CustomerName | OrderDate | Amount |
+---------+-------------+------------+--------+
| 2       | Bob         | 2025-03-02 | 700    |
| 4       | David       | 2025-03-04 | 1000   |
+---------+-------------+------------+--------+
```

3] Create a View for Orders Placed by 'Alice'

```
CREATE VIEW AliceOrders AS
SELECT * FROM Orders WHERE CustomerName = 'Alice';
```

View the Data

```
SELECT * FROM AliceOrders;
```

Output:

```
+---------+-------------+------------+--------+
| OrderID | CustomerName | OrderDate | Amount |
+---------+-------------+------------+--------+
| 1       | Alice       | 2025-03-01 | 500    |
+---------+-------------+------------+--------+
```