Scenario No. 1: Imagine a Library Management System which manages member information, library resources, physical loans, digital access, and membership tiers.
Members: Stores personal details of each member (MemberID, Name, Address, Phone, Email, TierID).
Resources: Lists the different resources available in the library, including books and digital content (ResourceID, Title, Author, Type, Format, AvailabilityStatus).
Loans: Records the physical loans of resources issued to members (LoanID, MemberID, ResourceID, LoanDate, ReturnDate).
DigitalAccess: Tracks the digital access of members to online resources (AccessID, MemberID, ResourceID, AccessDate, ExpiryDate).
MembershipTiers: Defines the privileges of different types of memberships (TierID, TierName, MaxLoans, MaxDigitalAccess).

Solve the following statements with the help of SQL Queries
1.Retrieve all members who belong to TierID 2.
2.List members who have borrowed more than 5 resources.
3.What is the total of all MaxLoans defined in the Membership Tiers?
4. List member names along with the date they borrowed each resource.
5.Create a trigger to automatically mark a resource as 'Not Available' when it's loaned
6.Add a new column DateOfBirth to the Members table.

Ans :

CREATE TABLE MembershipTiers (TierID INT PRIMARY KEY, TierName VARCHAR(50), MaxLoans INT, MaxDigitalAccess INT);

CREATE TABLE Members (MemberID INT PRIMARY KEY, Name VARCHAR(100), Address VARCHAR(200), Phone VARCHAR(15), Email VARCHAR(100), TierID INT, DateOfBirth DATE, FOREIGN KEY (TierID) REFERENCES MembershipTiers(TierID));

CREATE TABLE Resources (ResourceID INT PRIMARY KEY, Title VARCHAR(100), Author VARCHAR(100), Type VARCHAR(50), Format VARCHAR(50), AvailabilityStatus VARCHAR(20));

CREATE TABLE Loans (LoanID INT PRIMARY KEY, MemberID INT, ResourceID INT, LoanDate DATE, ReturnDate DATE, FOREIGN KEY (MemberID) REFERENCES Members(MemberID), FOREIGN KEY (ResourceID) REFERENCES Resources(ResourceID));

CREATE TABLE DigitalAccess (AccessID INT PRIMARY KEY, MemberID INT, ResourceID INT, AccessDate DATE, ExpiryDate DATE, FOREIGN KEY (MemberID) REFERENCES Members(MemberID), FOREIGN KEY (ResourceID) REFERENCES Resources(ResourceID));

INSERT INTO MembershipTiers VALUES (1, 'Basic', 3, 2), (2, 'Premium', 7, 5), (3, 'Elite', 10, 10);

INSERT INTO Members (MemberID, Name, Address, Phone, Email, TierID) VALUES (101, 'Alice', '123 Maple St', '1234567890', 'alice@mail.com', 2), (102, 'Bob', '456 Oak St', '2345678901', 'bob@mail.com', 1), (103, 'Charlie', '789 Pine St', '3456789012', 'charlie@mail.com', 2);

INSERT INTO Resources VALUES (201, 'Data Structures', 'Mark Allen', 'Book', 'Hardcover', 'Available'), (202, 'Machine Learning', 'Andrew Ng', 'Book', 'Paperback', 'Available'), (203, 'Database Systems', 'Elmasri', 'Book', 'PDF', 'Available'), (204, 'Operating Systems', 'Galvin', 'Book', 'eBook', 'Available');

INSERT INTO Loans VALUES (1, 101, 201, '2025-04-01', '2025-04-10'), (2, 101, 202, '2025-04-02', '2025-04-11'), (3, 101, 203, '2025-04-03', '2025-04-12'), (4, 101, 204, '2025-04-04', '2025-04-13'), (5, 101, 202, '2025-04-05', '2025-04-14'), (6, 101, 203, '2025-04-06', '2025-04-15');

INSERT INTO DigitalAccess VALUES (1, 102, 203, '2025-04-01', '2025-05-01'), (2, 103, 204, '2025-04-01', '2025-05-01');

Query 1:
SELECT * FROM Members WHERE TierID = 2;

Query 2:
SELECT MemberID, COUNT(*) AS TotalLoans FROM Loans GROUP BY MemberID HAVING COUNT(*) > 5;

Query 3:
SELECT SUM(MaxLoans) AS TotalMaxLoans FROM MembershipTiers;

Query 4:
SELECT Members.Name, Loans.LoanDate FROM Members JOIN Loans ON Members.MemberID = Loans.MemberID;

Query 5:

```
DELIMITER //
CREATE TRIGGER before_loan_insert
BEFORE INSERT ON Loans
FOR EACH ROW
BEGIN
   UPDATE Resources
   SET AvailabilityStatus = 'Not Available'
   WHERE ResourceID = NEW.ResourceID;
```

```
END;
//
DELIMITER ;
```

Query 6:

```
ALTER TABLE Members ADD COLUMN DateOfBirth date ;
```

Scenario 2: Imagine a Library Management System which manages member information, library resources, physical loans, digital access, and membership tiers.
Members: Stores personal details of each member (MemberID, Name, Address, Phone, Email, TierID).
Resources: Lists the different resources available in the library, including books and digital content (ResourceID, Title, Author, Type, Format, AvailabilityStatus).
Loans: Records the physical loans of resources issued to members (LoanID, MemberID, ResourceID, LoanDate, ReturnDate).
DigitalAccess: Tracks the digital access of members to online resources (AccessID, MemberID, ResourceID, AccessDate, ExpiryDate).
MembershipTiers: Defines the privileges of different types of memberships (TierID, TierName, MaxLoans, MaxDigitalAccess).

Solve the following statements with the help of SQL Queries
1. Get all resources that are either Books or Magazines.
2.List all members sorted alphabetically by their names.
3.Count the number of members in each membership tier.
4.Create a procedure to retrieve all loans of a given member.
5.Update the phone number of the member with MemberID = 1.
6. Show a list of all loans along with member names and emails. Include loans even if the member details are missing.

Ans:

CREATE TABLE MembershipTiers (TierID INT PRIMARY KEY, TierName VARCHAR(50), MaxLoans INT, MaxDigitalAccess INT);

CREATE TABLE Members (MemberID INT PRIMARY KEY, Name VARCHAR(100), Address VARCHAR(200), Phone VARCHAR(20), Email VARCHAR(100), TierID INT, FOREIGN KEY (TierID) REFERENCES MembershipTiers(TierID));

CREATE TABLE Resources (ResourceID INT PRIMARY KEY, Title VARCHAR(100), Author VARCHAR(100), Type VARCHAR(50), Format VARCHAR(50), AvailabilityStatus VARCHAR(20));

CREATE TABLE Loans (LoanID INT PRIMARY KEY, MemberID INT, ResourceID INT, LoanDate DATE, ReturnDate DATE, FOREIGN KEY (MemberID) REFERENCES Members(MemberID), FOREIGN KEY (ResourceID) REFERENCES Resources(ResourceID));

CREATE TABLE DigitalAccess (AccessID INT PRIMARY KEY, MemberID INT, ResourceID INT, AccessDate DATE, ExpiryDate DATE, FOREIGN KEY (MemberID) REFERENCES Members(MemberID), FOREIGN KEY (ResourceID) REFERENCES Resources(ResourceID));

INSERT INTO MembershipTiers VALUES (1, 'Basic', 3, 2), (2, 'Premium', 5, 5);

INSERT INTO Members VALUES (1, 'Alice', '123 Apple St', '1234567890', 'alice@example.com', 1), (2, 'Bob', '456 Banana Ave', '2345678901', 'bob@example.com', 2), (3, 'Charlie', '789 Cherry Blvd', '3456789012', 'charlie@example.com', 1);

INSERT INTO Resources VALUES (101, 'SQL for Beginners', 'John Doe', 'Book', 'Physical', 'Available'), (102, 'Advanced SQL', 'Jane Smith', 'Book', 'PDF', 'Borrowed'), (103, 'Tech Monthly', 'Tech Group', 'Magazine', 'Digital', 'Available'), (104, 'Science Today', 'Science Org', 'Journal', 'Physical', 'Available');

INSERT INTO Loans VALUES (1, 1, 101, '2024-03-01', '2024-03-15'), (2, 2, 102, '2024-04-01', NULL);

INSERT INTO DigitalAccess VALUES (1, 1, 103, '2024-04-01', '2024-04-30'), (2, 3, 104, '2024-04-05', '2024-05-05');

Query 1:
SELECT * FROM Resources WHERE Type IN ('Book', 'Magazine');

Query 2:
SELECT * FROM Members ORDER BY Name ASC;

Query 3:
SELECT TierID, COUNT(*) AS TotalMembers FROM Members GROUP BY TierID;

Query 4:
```
DELIMITER //
CREATE PROCEDURE GetMemberLoans(IN memID INT)
BEGIN
   SELECT * FROM Loans WHERE MemberID = memID;
END;
//
DELIMITER ;
```

CALL GetMemberLoans(1);

Query 5:
UPDATE Members SET Phone = '9876543210' WHERE MemberID = 1;

Query 6:
SELECT Loans.*, Members.Name, Members.Email FROM Loans LEFT JOIN Members ON Loans.MemberID = Members.MemberID;

Scenario 3: Imagine a Library Management System which manages member information, library resources, physical loans, digital access, and membership tiers.
Members: Stores personal details of each member (MemberID, Name, Address, Phone, Email, TierID).
Resources: Lists the different resources available in the library, including books and digital content (ResourceID, Title, Author, Type, Format, AvailabilityStatus).
Loans: Records the physical loans of resources issued to members (LoanID, MemberID, ResourceID, LoanDate, ReturnDate).
DigitalAccess: Tracks the digital access of members to online resources (AccessID, MemberID, ResourceID, AccessDate, ExpiryDate).
MembershipTiers: Defines the privileges of different types of memberships (TierID, TierName, MaxLoans, MaxDigitalAccess).

Solve the following statements with the help of SQL Queries
1.Find all digital resources that are currently accessible (i.e., not expired yet).
2.Find members who accessed more than 2 digital resources.
3.Find the average number of digital accesses allowed across all tiers.
4.Show each resource and who currently has it on loan (if anyone).
5.Automatically update the availability status to 'Available' when a loan is deleted (resource returned).
6.Create a procedure to list all resources based on **availability status** (e.g., "Available", "Not Available").

Ans:

CREATE TABLE MembershipTiers (TierID INT PRIMARY KEY, TierName VARCHAR(50), MaxLoans INT, MaxDigitalAccess INT);

CREATE TABLE Members (MemberID INT PRIMARY KEY, Name VARCHAR(100), Address VARCHAR(255), Phone VARCHAR(15), Email VARCHAR(100), TierID INT, FOREIGN KEY (TierID) REFERENCES MembershipTiers(TierID));

CREATE TABLE Resources (ResourceID INT PRIMARY KEY, Title VARCHAR(200), Author VARCHAR(100), Type VARCHAR(50), Format VARCHAR(50), AvailabilityStatus VARCHAR(20));

CREATE TABLE Loans (LoanID INT PRIMARY KEY, MemberID INT, ResourceID INT, LoanDate DATE, ReturnDate DATE, FOREIGN KEY (MemberID) REFERENCES Members(MemberID), FOREIGN KEY (ResourceID) REFERENCES Resources(ResourceID));

CREATE TABLE DigitalAccess (AccessID INT PRIMARY KEY, MemberID INT, ResourceID INT, AccessDate DATE, ExpiryDate DATE, FOREIGN KEY (MemberID)

REFERENCES Members(MemberID), FOREIGN KEY (ResourceID) REFERENCES Resources(ResourceID));

INSERT INTO MembershipTiers VALUES (1, 'Basic', 2, 3), (2, 'Premium', 5, 7);

INSERT INTO Members VALUES (101, 'Alice', '123 Street', '1111111111', alice@example.com', 1), (102, 'Bob', '456 Avenue', '2222222222', 'bob@example.com', 2), (103, 'Carol', '789 Road', '3333333333', 'carol@example.com', 2);

INSERT INTO Resources VALUES (1, 'Database Systems', 'Elmasri', 'Book', 'PDF', 'Not Available'), (2, 'Computer Networks', 'Tanenbaum', 'Book', 'ePub', 'Available'), (3, 'AI Basics', 'Russell', 'Book', 'PDF', 'Not Available'), (4, 'Cloud Computing', 'Armbrust', 'Book', 'PDF', 'Available');

INSERT INTO Loans VALUES (201, 101, 1, '2025-04-01', NULL), (202, 102, 3, '2025-04-05', NULL);

INSERT INTO DigitalAccess VALUES (301, 101, 2, '2025-03-10', '2025-04-30'), (302, 102, 4, '2025-04-01', '2025-04-10'), (303, 103, 2, '2025-03-15', '2025-04-15'), (304, 103, 4, '2025-03-15', '2025-03-25'), (305, 103, 1, '2025-03-15', '2025-04-20');

Query 1:
SELECT * FROM DigitalAccess WHERE ExpiryDate >= CURDATE();

Query 2:
SELECT MemberID, COUNT(*) AS AccessCount FROM DigitalAccess GROUP BY MemberID HAVING COUNT(*) > 2;

Query 3:
SELECT AVG(MaxDigitalAccess) AS AvgDigitalLimit FROM MembershipTiers;

Query 4:
SELECT Resources.Title, Loans.MemberID, Loans.LoanDate FROM Resources LEFT JOIN Loans ON Resources.ResourceID = Loans.ResourceID;

Query 5:

```
DELIMITER //
CREATE TRIGGER after_loan_delete
AFTER DELETE ON Loans
FOR EACH ROW
BEGIN
   UPDATE Resources
   SET AvailabilityStatus = 'Available'
   WHERE ResourceID = OLD.ResourceID;
```

```
END;
//
DELIMITER ;
```

Query 6:

```
DELIMITER //
CREATE PROCEDURE GetResourcesByAvailability(IN status VARCHAR(20))
BEGIN
   SELECT ResourceID, Title, Author, Type, Format FROM Resources WHERE AvailabilityStatus = status;
END;
//
DELIMITER ;

CALL GetResourcesByAvailability('Available');
```

Scenario 4: Imagine a Library Management System which manages member information, library resources, physical loans, digital access, and membership tiers.
Members: Stores personal details of each member (MemberID, Name, Address, Phone, Email, TierID).
Resources: Lists the different resources available in the library, including books and digital content (ResourceID, Title, Author, Type, Format, AvailabilityStatus).
Loans: Records the physical loans of resources issued to members (LoanID, MemberID, ResourceID, LoanDate, ReturnDate).
DigitalAccess: Tracks the digital access of members to online resources (AccessID, MemberID, ResourceID, AccessDate, ExpiryDate).
MembershipTiers: Defines the privileges of different types of memberships (TierID, TierName, MaxLoans, MaxDigitalAccess).

Solve the following statements with the help of SQL Queries

1.Find the number of resources grouped by type and format
2.Retrieve members and their membership tier name (LEFT JOIN)
3.Get list of resources that are currently unavailable
4.Create a function that returns the total number of resources available in a given format (e.g., 'Digital' or 'Physical').
5.Create a procedure to list all resources based on availability status
6.Find total digital accesses per resource

Ans:

CREATE TABLE MembershipTiers (TierID INT PRIMARY KEY, TierName VARCHAR(50), MaxLoans INT, MaxDigitalAccess INT);

CREATE TABLE Members (MemberID INT PRIMARY KEY, Name VARCHAR(100), Address VARCHAR(255), Phone VARCHAR(15), Email VARCHAR(100), TierID INT, FOREIGN KEY (TierID) REFERENCES MembershipTiers(TierID));

CREATE TABLE Resources (ResourceID INT PRIMARY KEY, Title VARCHAR(100), Author VARCHAR(100), Type VARCHAR(50), Format VARCHAR(50), AvailabilityStatus VARCHAR(20));

CREATE TABLE Loans (LoanID INT PRIMARY KEY, MemberID INT, ResourceID INT, LoanDate DATE, ReturnDate DATE, FOREIGN KEY (MemberID) REFERENCES Members(MemberID), FOREIGN KEY (ResourceID) REFERENCES Resources(ResourceID));

CREATE TABLE DigitalAccess (AccessID INT PRIMARY KEY, MemberID INT, ResourceID INT, AccessDate DATE, ExpiryDate DATE, FOREIGN KEY (MemberID) REFERENCES Members(MemberID), FOREIGN KEY (ResourceID) REFERENCES Resources(ResourceID));

INSERT INTO MembershipTiers VALUES (1, 'Basic', 2, 5), (2, 'Premium', 5, 10);

INSERT INTO Members VALUES (101, 'Alice', '123 Street', '1234567890', 'alice@example.com', 1), (102, 'Bob', '456 Avenue', '0987654321', 'bob@example.com', 2);

INSERT INTO Resources VALUES (201, 'SQL Basics', 'John Doe', 'Book', 'Physical', 'Available'), (202, 'Advanced SQL', 'Jane Roe', 'Book', 'Digital', 'Unavailable'), (203, 'AI Journal', 'Dr. Smith', 'Journal', 'Digital', 'Available');

INSERT INTO Loans VALUES (301, 101, 201, '2025-04-01', '2025-04-10');

INSERT INTO DigitalAccess VALUES (401, 101, 202, '2025-04-02', '2025-05-02'), (402, 102, 203, '2025-04-03', '2025-05-03');

Query 1:

SELECT Type, Format, COUNT(*) AS TotalResources FROM Resources GROUP BY Type, Format;

Query 2:

SELECT m.Name, t.TierName FROM Members m LEFT JOIN MembershipTiers t ON m.TierID = t.TierID;

Query 3:

SELECT * FROM Resources WHERE AvailabilityStatus = 'Unavailable';

Query 4:

```
DELIMITER //

CREATE FUNCTION CountAvailableResourcesByFormat(resFormat VARCHAR(20))

RETURNS INT

DETERMINISTIC

BEGIN

    DECLARE total INT;

    SELECT COUNT(*) INTO total FROM Resources WHERE Format = resFormat AND AvailabilityStatus = 'Available';

    RETURN total;

END; //
```

```
DELIMITER ;

SELECT CountAvailableResourcesByFormat('Digital') AS DigitalAvailableResources;
```

Query 5:

```
DELIMITER //

CREATE PROCEDURE ListResourcesByAvailability(IN status VARCHAR(20))

BEGIN

   SELECT * FROM Resources WHERE AvailabilityStatus = status;

END;

//

DELIMITER ;

CALL ListResourcesByAvailability('Available');
```

Query 6:

```
SELECT ResourceID, COUNT(*) AS TotalAccesses FROM DigitalAccess GROUP BY ResourceID;
```