Experiment #		Student ID	2410080018
Date		Student Name	yashwanth

Exp 1: Design an entity-relationship (ER) diagram and translate it into a relational database schema.

Aim of the Experiment:

To design an Entity-Relationship (ER) diagram for a given real-world scenario in order to model the data requirements, and then translate the ER diagram into a corresponding relational database schema that can be implemented using a relational database management system (RDBMS).

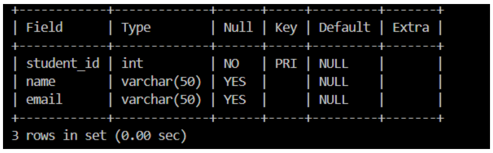
Objectives:

1) Gain insight into how entities, relationships, attributes, and constraints are represented in an ER diagram.
2) Learn how to analyse a real-world scenario and accurately capture its data requirements in an ER model.
3) Develop skills to visually represent the structure of a database system using standard ER diagram conventions.
4) Understand cardinality and participation constraints appropriately- how to define one-to-one, one-to-many, and many-to-many relationships and their implications.
5) Translate the ER model into a set of normalized relational tables using mapping rules.
6) Maintain data integrity and minimize redundancy in the conversion process.
7) Practice the principles of good database design, including primary keys, foreign keys, and referential integrity.

Program :

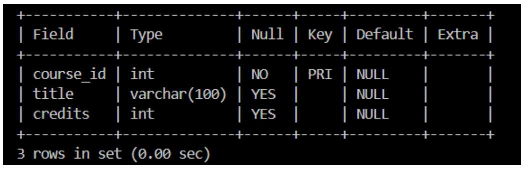
```
CREATE TABLE Student (
    student_id INT PRIMARY KEY,
    name VARCHAR(50),
    email VARCHAR(50)
);

CREATE TABLE Course (
    course_id INT PRIMARY KEY,
    title VARCHAR(100),
    credits INT
);

CREATE TABLE Enrollment (
    student_id INT,
    course_id INT,
    enrollment_date DATE,
    PRIMARY KEY (student_id, course_id),
    FOREIGN KEY (student_id) REFERENCES Student(student_id),
    FOREIGN KEY (course_id) REFERENCES Course(course_id)
);
```

describe student;



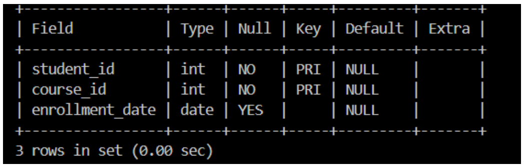
```
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| student_id | int         | NO   | PRI | NULL    |       |
| name       | varchar(50) | YES  |     | NULL    |       |
| email      | varchar(50) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

describe course;

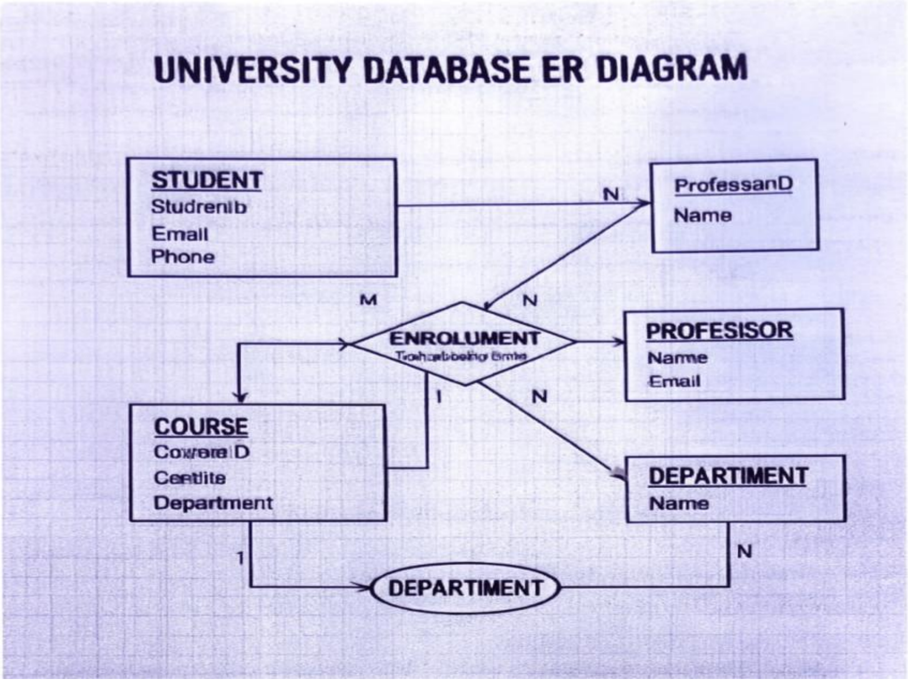


```
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| course_id | int          | NO   | PRI | NULL    |       |
| title     | varchar(100) | YES  |     | NULL    |       |
| credits   | int          | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

describe enrollment;



```
+-----------------+------+------+-----+---------+-------+
| Field           | Type | Null | Key | Default | Extra |
+-----------------+------+------+-----+---------+-------+
| student_id      | int  | NO   | PRI | NULL    |       |
| course_id       | int  | NO   | PRI | NULL    |       |
| enrollment_date | date | YES  |     | NULL    |       |
+-----------------+------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```



Evaluator Remark (if Any):	**Marks Secured:___out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

Exp 2: Apply normalization techniques (1NF, 2NF, 3NF) to optimize an unnormalized database design.

Aim of the Experiment:

To apply normalization techniques—specifically First Normal Form (1NF), Second Normal Form (2NF), and Third Normal Form (3NF)—to an unnormalized database design in order to eliminate data redundancy, improve data integrity, and optimize the overall structure of the database.

Objectives:

1) To gain a clear understanding of database normalization and its role in improving the structure and efficiency of relational database.
2) To recognize redundancy, insertion, update, and deletion anomalies present in an unnormalized database.
3) To remove repeating groups and ensure atomicity by transforming the unnormalized data into 1NF.
4) To eliminate partial dependencies by decomposing the 1NF tables and ensuring that all non-key attributes are fully functionally dependent on the entire primary key.
5) To eliminate transitive dependencies by refining the 2NF design so that all attributes are only dependent on the primary key.
6) To achieve a cleaner, more efficient, and scalable database structure by applying normalization up to 3NF.
7) To ensure that the final design supports consistent, accurate, and non-redundant data storage.

Experiment #		Student ID	2410080018
Date		Student Name	yashwanth

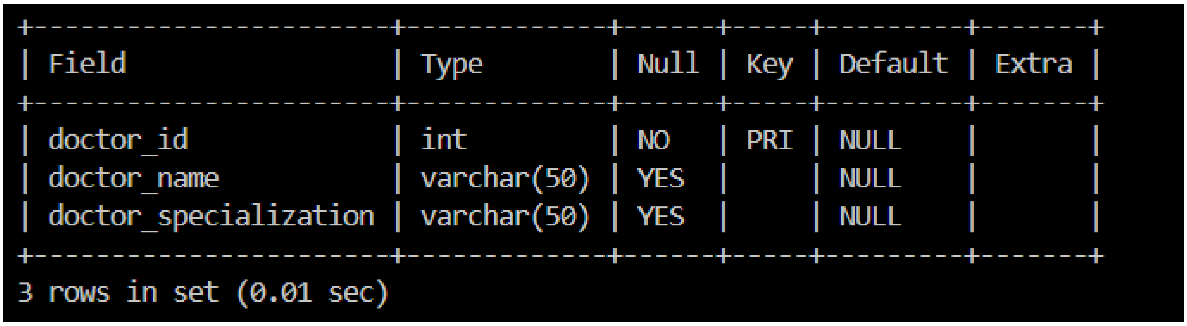
Program :

```
CREATE TABLE Patient (
   patient_id INT PRIMARY KEY,
   patient_name VARCHAR(50)
);

CREATE TABLE Doctor (
   doctor_id INT PRIMARY KEY,
   doctor_name VARCHAR(50),
   doctor_specialization VARCHAR(50)
);

CREATE TABLE Treatment (
   treatment_id INT PRIMARY KEY,
   patient_id INT,
   doctor_id INT,
   treatment_name VARCHAR(50),
   FOREIGN KEY (patient_id) REFERENCES Patient(patient_id),
   FOREIGN KEY (doctor_id) REFERENCES Doctor(doctor_id)
);

CREATE TABLE Medicine (
   med_id INT PRIMARY KEY,
   treatment_id INT,
   med_name VARCHAR(50),
   FOREIGN KEY (treatment_id) REFERENCES Treatment(treatment_id)
);
```

Evaluator Remark (if Any):	**Marks Secured:___out of 50**
	Signature of the Evaluator with Date

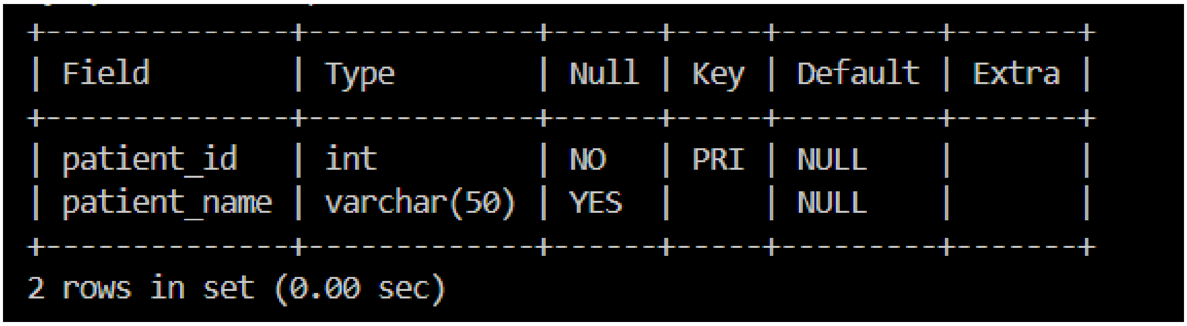
Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

DESCRIBE doctor;



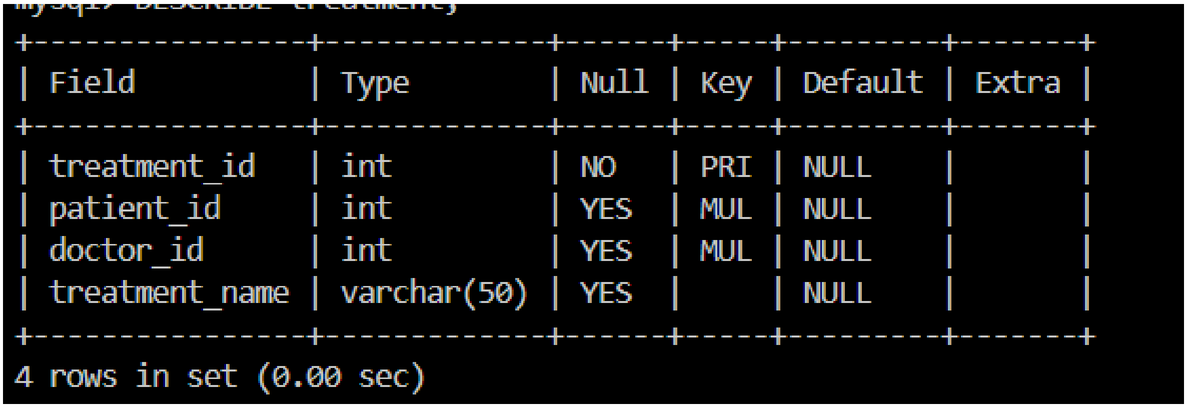
```
+-----------------------+-------------+------+-----+---------+-------+
| Field                 | Type        | Null | Key | Default | Extra |
+-----------------------+-------------+------+-----+---------+-------+
| doctor_id             | int         | NO   | PRI | NULL    |       |
| doctor_name           | varchar(50) | YES  |     | NULL    |       |
| doctor_specialization | varchar(50) | YES  |     | NULL    |       |
+-----------------------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

DESCRIBE patient;



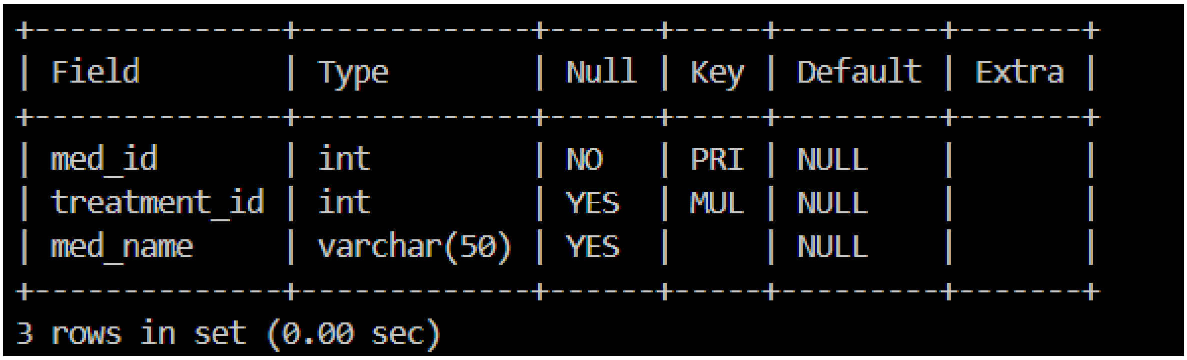
```
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| patient_id   | int         | NO   | PRI | NULL    |       |
| patient_name | varchar(50) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

DESCRIBE treatment;



```
mysql> DESCRIBE treatment;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| treatment_id   | int         | NO   | PRI | NULL    |       |
| patient_id     | int         | YES  | MUL | NULL    |       |
| doctor_id      | int         | YES  | MUL | NULL    |       |
| treatment_name | varchar(50) | YES  |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

DESCRIBE medicine;



```
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| med_id       | int         | NO   | PRI | NULL    |       |
| treatment_id | int         | YES  | MUL | NULL    |       |
| med_name     | varchar(50) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Experiment #		Student ID	2410080018
Date		Student Name	yashwanth

Exp-3: Install a DBMS, create a database define tables and apply constraints using DDL commands.

Aim of the Experiment:

To install a Database Management System (DBMS), create a database, define tables, and apply various constraints using Data Definition Language (DDL) commands in order to establish a structured and well-defined database schema.

Objectives:

1. To understand the process of installing a Database Management System (DBMS).
2. To learn how to create a new database within the installed DBMS.
3. To define and create tables using Data Definition Language (DDL) commands.
4. To apply various constraints (such as **PRIMARY KEY**, **FOREIGN KEY**, **UNIQUE**, **NOT NULL**, and **CHECK**) to enforce data integrity.
5. To gain hands-on experience with the basic structure and syntax of SQL DDL commands.
6. To verify the successful creation of the database and its components through queries and schema inspection.

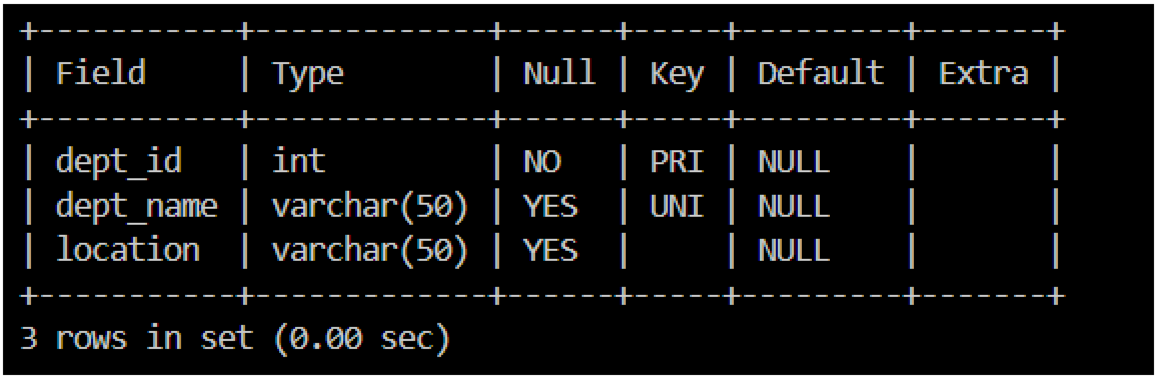
Program :

```
CREATE DATABASE CompanyDB;
USE CompanyDB;

CREATE TABLE Department (
   dept_id INT PRIMARY KEY,
   dept_name VARCHAR(50) UNIQUE,
   location VARCHAR(50)
);

CREATE TABLE Employee (
   emp_id INT PRIMARY KEY,
   emp_name VARCHAR(50) NOT NULL,
   salary DECIMAL(10,2) CHECK (salary > 0),
   dept_id INT,
   FOREIGN KEY (dept_id) REFERENCES Department(dept_id)
);
```

Evaluator Remark (if Any):	**Marks Secured:___out of 50**
	Signature of the Evaluator with Date

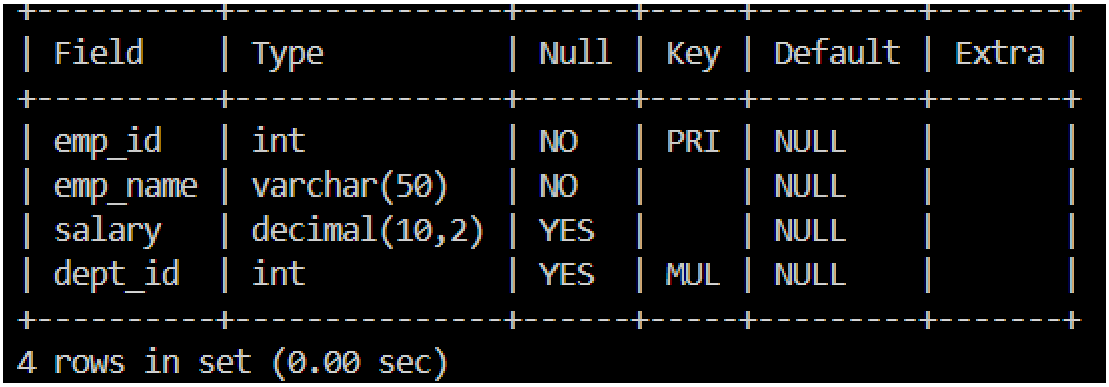
Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

DESCRIBE department;



```
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| dept_id   | int         | NO   | PRI | NULL    |       |
| dept_name | varchar(50) | YES  | UNI | NULL    |       |
| location  | varchar(50) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

DESCRIBE employee;



```
+----------+---------------+------+-----+---------+-------+
| Field    | Type          | Null | Key | Default | Extra |
+----------+---------------+------+-----+---------+-------+
| emp_id   | int           | NO   | PRI | NULL    |       |
| emp_name | varchar(50)   | NO   |     | NULL    |       |
| salary   | decimal(10,2) | YES  |     | NULL    |       |
| dept_id  | int           | YES  | MUL | NULL    |       |
+----------+---------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Experiment #		Student ID	2410080018
Date		Student Name	yashwanth

Exp 4: Perform data insertion, updates and deletion using DML. Create database views to simplify complex queries.

Aim of the Experiment:

To perform data manipulation operations—such as insertion, update, and deletion—using Data Manipulation Language (DML) commands, and to create database views that simplify complex queries, enhance data abstraction, and improve query efficiency.

Objectives:

1.
2. To perform data insertion, updating, and deletion operations using Data Manipulation Language (DML) commands such as INSERT, UPDATE, and DELETE.
3. To understand the syntax and application of DML commands in managing and modifying database records.
4. To create and utilize database views in order to simplify complex SQL queries and enhance query efficiency.
5. To demonstrate how views can provide data abstraction and improve security by restricting access to specific data.
6. To validate the impact of DML operations on base tables and views.

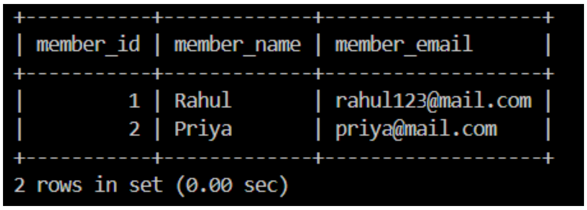
```
CREATE DATABASE LibraryDB;
USE LibraryDB;

CREATE TABLE Member (
   member_id INT PRIMARY KEY,
   member_name VARCHAR(50),
   member_email VARCHAR(50)
);

CREATE TABLE Book (
   book_id INT PRIMARY KEY,
   book_title VARCHAR(100),
   author VARCHAR(50)
);

CREATE TABLE Borrow (
   member_id INT,
   book_id INT,
   borrow_date DATE,
   PRIMARY KEY(member_id, book_id),
   FOREIGN KEY(member_id) REFERENCES Member(member_id),
   FOREIGN KEY(book_id) REFERENCES Book(book_id)
);

-- Insert sample records
INSERT INTO Member VALUES (1, 'Rahul', 'rahul@mail.com');
INSERT INTO Book VALUES (101, 'DBMS Concepts', 'Navathe');
INSERT INTO Borrow VALUES (1, 101, '2025-08-01');

-- Update
UPDATE Member SET member_email='rahul123@mail.com' WHERE member_id=1;

-- Delete
DELETE FROM Borrow WHERE book_id=101;

-- View
CREATE VIEW BorrowedBooks AS
SELECT m.member_name, b.book_title
FROM Member m JOIN Borrow br ON m.member_id=br.member_id
JOIN Book b ON br.book_id=b.book_id;
```

Evaluator Remark (if Any):	**Marks Secured:____out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

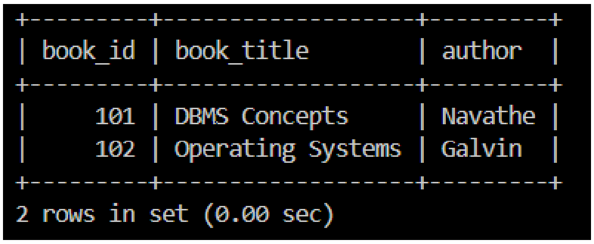
INSERT INTO Member VALUES (2, 'Priya', 'priya@mail.com');
INSERT INTO Book VALUES (102, 'Operating Systems', 'Galvin');
INSERT INTO Borrow VALUES (1, 101, '2025-08-01');

SELECT * FROM Member;



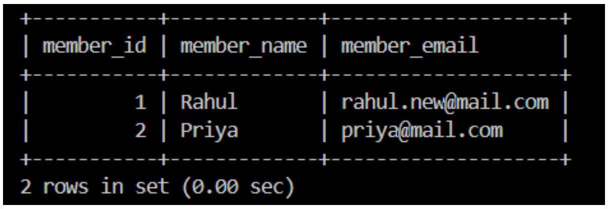
```
+-----------+-------------+------------------+
| member_id | member_name | member_email     |
+-----------+-------------+------------------+
|         1 | Rahul       | rahul123@mail.com |
|         2 | Priya       | priya@mail.com   |
+-----------+-------------+------------------+
2 rows in set (0.00 sec)
```

SELECT * FROM Book;



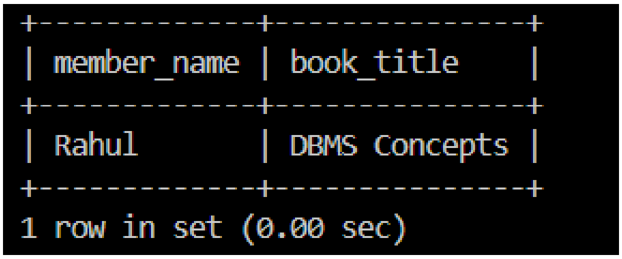
```
+---------+-------------------+---------+
| book_id | book_title        | author  |
+---------+-------------------+---------+
|     101 | DBMS Concepts     | Navathe |
|     102 | Operating Systems | Galvin  |
+---------+-------------------+---------+
2 rows in set (0.00 sec)
```

UPDATE Member
-> SET member_email = 'rahul.new@mail.com'
-> WHERE member_id = 1;
SELECT * FROM Member;



```
+-----------+-------------+--------------------+
| member_id | member_name | member_email       |
+-----------+-------------+--------------------+
|         1 | Rahul       | rahul.new@mail.com |
|         2 | Priya       | priya@mail.com     |
+-----------+-------------+--------------------+
2 rows in set (0.00 sec)
```

SELECT * FROM BorrowedBooks;



```
+-------------+---------------+
| member_name | book_title    |
+-------------+---------------+
| Rahul       | DBMS Concepts |
+-------------+---------------+
1 row in set (0.00 sec)
```

Experiment #		Student ID	2410080018
Date		Student Name	yashwanth

Exp 5: Write select statements with filtering, joins (inner, outer) to retrieve and combine data from multiple tables.

Aim of the Experiment:

To write and execute SQL SELECT statements using filtering conditions and various types of joins (inner and outer) in order to retrieve and combine data from multiple related tables effectively. This helps in understanding how relational data can be queried and manipulated to extract meaningful information.

Objectives:

1. To understand the structure and relationships between multiple tables in a relational database.
2. To write SELECT statements that retrieve specific data using various filtering conditions (e.g., WHERE, LIKE, BETWEEN, IN).
3. To implement INNER JOINs to combine rows from two or more tables based on related columns.
4. To apply OUTER JOINs (LEFT, RIGHT, FULL) to retrieve related and unrelated data from multiple tables.
5. To practice combining multiple filtering and joining techniques in a single query for complex data retrieval.
6. To enhance SQL querying skills for effective data extraction and analysis.

Experiment #		Student ID	2410080018
Date		Student Name	yashwanth

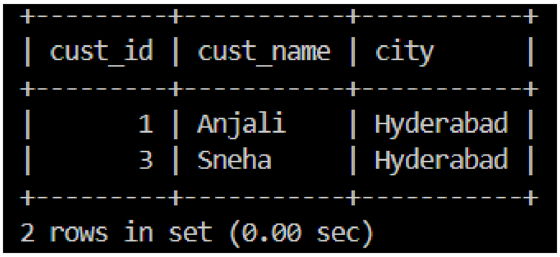
```
CREATE DATABASE SalesDBUSE SalesDB;

CREATE TABLE Customer (
   cust_id INT PRIMARY KEY,
   cust_name VARCHAR(50),
   city VARCHAR(50)
);

CREATE TABLE Orders (
   order_id INT PRIMARY KEY,
   order_date DATE,
   cust_id INT,
   FOREIGN KEY (cust_id) REFERENCES Customer(cust_id)
);

-- Example SELECT with filter
SELECT * FROM Customer WHERE city='Hyderabad';

-- Inner Join
SELECT c.cust_name, o.order_id
FROM Customer c INNER JOIN Orders o ON c.cust_id=o.cust_id;

-- Left Join
SELECT c.cust_name, o.order_id
FROM Customer c LEFT JOIN Orders o ON c.cust_id=o.cust_id;
```

Evaluator Remark (if Any):	**Marks Secured:___out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

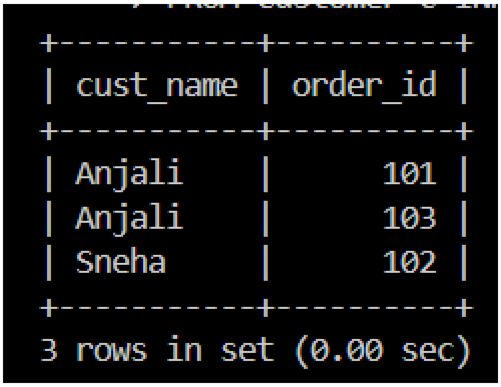
```
INSERT INTO Customer VALUES
```

```
(1, 'Anjali', 'Hyderabad'),
(2, 'Ravi', 'Mumbai'),
(3, 'Sneha', 'Hyderabad');

INSERT INTO Orders VALUES
(101, '2025-09-15', 1),
(102, '2025-09-16', 3),
(103, '2025-09-17', 1);

SELECT * FROM Customer WHERE city='Hyderabad';
```



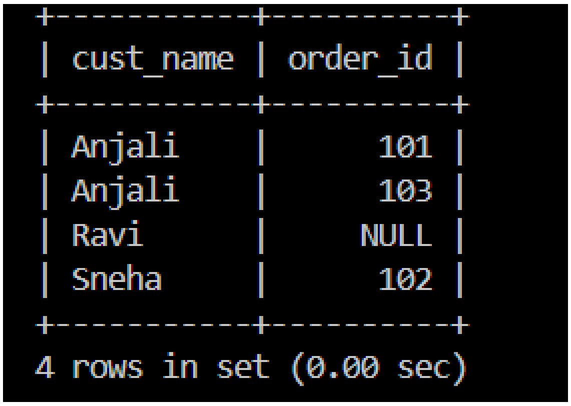
```
+---------+-----------+-----------+
| cust_id | cust_name | city      |
+---------+-----------+-----------+
|       1 | Anjali    | Hyderabad |
|       3 | Sneha     | Hyderabad |
+---------+-----------+-----------+
2 rows in set (0.00 sec)
```

```
SELECT c.cust_name, o.order_id
    -> FROM Customer c INNER JOIN Orders o ON
     c.cust_id=o.cust_id;
```



```
+-----------+----------+
| cust_name | order_id |
+-----------+----------+
| Anjali    |      101 |
| Anjali    |      103 |
| Sneha     |      102 |
+-----------+----------+
3 rows in set (0.00 sec)
```

```
SELECT c.cust_name, o.order_id
    -> FROM Customer c LEFT JOIN Orders o ON
     c.cust_id=o.cust_id;
```



```
+-----------+----------+
| cust_name | order_id |
+-----------+----------+
| Anjali    |      101 |
| Anjali    |      103 |
| Ravi      |     NULL |
| Sneha     |      102 |
+-----------+----------+
4 rows in set (0.00 sec)
```