NAME: Shikha Singh
Roll no: 25

Index

Kindly note:

Every Practical need to have Query statement and its output for that respective query (Screenshot required)

Every practical need to have order as follow:

Name of Practical :
Description of practical:
Questions :Take practice questions which we conduct in our sessions.
Output:screenshot required

Practical 1: Entity-Relationship (E-R) Diagram for a Delivery Service System

Name of Practical:
E-R Diagram and Conversion of Entities and Relationships into Tables

Description of Practical:
The task involves creating an Entity-Relationship diagram for a delivery service system, then converting the identified entities and their relationships into database tables. Afterward, queries will be written to interact with the database, and screenshots of the query results will be taken.

Questions:

1. Draw the E-R diagram for the Delivery Service System.

The entities could be as follows:

- **Customer** (Customer_ID, Name, Address, Phone_Number)
- **Order** (Order_ID, Order_Date, Delivery_Status, Customer_ID)
- **Delivery_Staff** (Staff_ID, Name, Vehicle_Number, Phone_Number)
- **Product** (Product_ID, Name, Price)
- **Order_Details** (Order_Detail_ID, Order_ID, Product_ID, Quantity)
- **Delivery** (Delivery_ID, Order_ID, Staff_ID, Delivery_Date)

Relationships:

- Customer places Orders.
- Order contains Products (through Order_Details).
- Order is delivered by Delivery Staff.
- Delivery relates to Order and Delivery Staff.

2. Convert the entities and relationships into database tables.

Output:

```
Last login: Sat Sep 21 20:54:22 on ttys001
/Users/shikhasingh/.zshrc:1: command not found: His
shikhasingh@Shikhas-MacBook-Air ~ % mysql -u root -p

Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 9.0.1 Homebrew

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE DeliveryServiceSystem;
Query OK, 1 row affected (0.00 sec)

mysql> USE DeliveryServiceSystem;
Database changed
mysql> CREATE TABLE Customer (
    ->     Customer_ID INT PRIMARY KEY AUTO_INCREMENT,
    ->     Name VARCHAR(100),
    ->     Address VARCHAR(255),
    ->     Phone_Number VARCHAR(15)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE Orders (
    ->     Order_ID INT PRIMARY KEY AUTO_INCREMENT,
    ->     Order_Date DATE,
    ->     Delivery_Status VARCHAR(50),
    ->     Customer_ID INT,
    ->     FOREIGN KEY (Customer_ID) REFERENCES Customer(Customer_ID)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE Delivery_Staff (
    ->     Staff_ID INT PRIMARY KEY AUTO_INCREMENT,
    ->     Name VARCHAR(100),
    ->     Vehicle_Number VARCHAR(20),
    ->     Phone_Number VARCHAR(15)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE Product (
    ->     Product_ID INT PRIMARY KEY AUTO_INCREMENT,
    ->     Name VARCHAR(100),
    ->     Price DECIMAL(10, 2)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE Order_Details (
    ->     Order_Detail_ID INT PRIMARY KEY AUTO_INCREMENT,
    ->     Order_ID INT,
    ->     Product_ID INT,
    ->     Quantity INT,
    ->     FOREIGN KEY (Order_ID) REFERENCES Orders(Order_ID),
    ->     FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE Delivery (
```

```
mysql> CREATE TABLE Delivery (
    ->     Delivery_ID INT PRIMARY KEY AUTO_INCREMENT,
    ->     Order_ID INT,
    ->     Staff_ID INT,
    ->     Delivery_Date DATE,
    ->     FOREIGN KEY (Order_ID) REFERENCES Orders(Order_ID),
    ->     FOREIGN KEY (Staff_ID) REFERENCES Delivery_Staff(Staff_ID)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO Customer (Name, Address, Phone_Number)
    -> VALUES
    -> ('John Doe', '123 Main St, City', '9876543210'),
    -> ('Jane Smith', '456 Elm St, City', '8765432109');
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Product (Name, Price)
    -> VALUES
    -> ('Laptop', 800.00),
    -> ('Smartphone', 500.00),
    -> ('Headphones', 50.00);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Orders (Order_Date, Delivery_Status, Customer_ID)
    -> VALUES
    -> ('2024-09-01', 'Pending', 1),
    -> ('2024-09-02', 'Delivered', 2);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Order_Details (Order_ID, Product_ID, Quantity)
    -> VALUES
    -> (1, 1, 1),  -- Laptop in order 1
    -> (2, 2, 2);  -- Smartphone in order 2
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Delivery_Staff (Name, Vehicle_Number, Phone_Number)
    -> VALUES
    -> ('Mike Johnson', 'ABC123', '9876543211'),
    -> ('Emma Williams', 'XYZ987', '8765432110');
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Delivery (Order_ID, Staff_ID, Delivery_Date)
    -> VALUES
    -> (2, 1, '2024-09-03');  -- Order 2 delivered by Staff 1
Query OK, 1 row affected (0.00 sec)

mysql> SELECT Order_ID, Order_Date, Delivery_Status
    -> FROM Orders;
+----------+------------+-----------------+
| Order_ID | Order_Date | Delivery_Status |
+----------+------------+-----------------+
|        1 | 2024-09-01 | Pending         |
|        2 | 2024-09-02 | Delivered       |
+----------+------------+-----------------+
2 rows in set (0.00 sec)

mysql> SELECT Product.Name, Order_Details.Quantity
    -> FROM Order_Details
    -> JOIN Product ON Order_Details.Product_ID = Product.Product_ID
```

```
mysql> SELECT Product.Name, Order_Details.Quantity
    -> FROM Order_Details
    -> JOIN Product ON Order_Details.Product_ID = Product.Product_ID
    -> WHERE Order_ID = 1;
+--------+----------+
| Name   | Quantity |
+--------+----------+
| Laptop |        1 |
+--------+----------+
1 row in set (0.00 sec)

mysql> SELECT Delivery.Delivery_ID, Delivery.Delivery_Date, Delivery_Staff.Name
    -> FROM Delivery
    -> JOIN Delivery_Staff ON Delivery.Staff_ID = Delivery_Staff.Staff_ID
    -> WHERE Delivery.Order_ID = 2;
+-------------+---------------+--------------+
| Delivery_ID | Delivery_Date | Name         |
+-------------+---------------+--------------+
|           1 | 2024-09-03    | Mike Johnson |
+-------------+---------------+--------------+
1 row in set (0.00 sec)

mysql> SELECT Customer.Name, Customer.Address, Orders.Order_ID
    -> FROM Orders
    -> JOIN Customer ON Orders.Customer_ID = Customer.Customer_ID
    -> WHERE Orders.Delivery_Status = 'Delivered';
+------------+------------------+----------+
| Name       | Address          | Order_ID |
+------------+------------------+----------+
| Jane Smith | 456 Elm St, City |        2 |
+------------+------------------+----------+
1 row in set (0.00 sec)

mysql> SELECT * FROM Customer;
+-------------+------------+------------------+--------------+
| Customer_ID | Name       | Address          | Phone_Number |
+-------------+------------+------------------+--------------+
|           1 | John Doe   | 123 Main St, City | 9876543210   |
|           2 | Jane Smith | 456 Elm St, City  | 8765432109   |
+-------------+------------+------------------+--------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM Orders;
+----------+------------+-----------------+-------------+
| Order_ID | Order_Date | Delivery_Status | Customer_ID |
+----------+------------+-----------------+-------------+
|        1 | 2024-09-01 | Pending         |           1 |
|        2 | 2024-09-02 | Delivered       |           2 |
+----------+------------+-----------------+-------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM Delivery_Staff;
+----------+---------------+----------------+--------------+
| Staff_ID | Name          | Vehicle_Number | Phone_Number |
+----------+---------------+----------------+--------------+
|        1 | Mike Johnson  | ABC123         | 9876543211   |
|        2 | Emma Williams | XYZ987         | 8765432110   |
+----------+---------------+----------------+--------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM Product;
+------------+------------+--------+
| Product_ID | Name       | Price  |
+------------+------------+--------+
```

```
mysql> SELECT * FROM Delivery_Staff;
+----------+---------------+----------------+--------------+
| Staff_ID | Name          | Vehicle_Number | Phone_Number |
+----------+---------------+----------------+--------------+
|        1 | Mike Johnson  | ABC123         | 9876543211   |
|        2 | Emma Williams | XYZ987         | 8765432110   |
+----------+---------------+----------------+--------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM Product;
+------------+------------+--------+
| Product_ID | Name       | Price  |
+------------+------------+--------+
|          1 | Laptop     | 800.00 |
|          2 | Smartphone | 500.00 |
|          3 | Headphones |  50.00 |
+------------+------------+--------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM Order_Details;
+-----------------+----------+------------+----------+
| Order_Detail_ID | Order_ID | Product_ID | Quantity |
+-----------------+----------+------------+----------+
|               1 |        1 |          1 |        1 |
|               2 |        2 |          2 |        2 |
+-----------------+----------+------------+----------+
2 rows in set (0.01 sec)

mysql> SELECT * FROM Delivery;
+-------------+----------+----------+---------------+
| Delivery_ID | Order_ID | Staff_ID | Delivery_Date |
+-------------+----------+----------+---------------+
|           1 |        2 |        1 | 2024-09-03    |
+-------------+----------+----------+---------------+
1 row in set (0.00 sec)

mysql>
```

Practical: 2

Name of Practical:

Performing DDL Commands in SQL

Description of Practical:

This practical involves demonstrating the use of Data Definition Language (DDL) commands in SQL, which include `CREATE`, `ALTER`, `DROP`, and `TRUNCATE`. These commands are used to define and modify the structure of database objects like tables.

Question:

Create a table named `Employees` with the following columns:

- `EmployeeID` (INT, Primary Key)
- `FirstName` (VARCHAR(50))
- `LastName` (VARCHAR(50))
- `Email` (VARCHAR(100))
- `PhoneNumber` (VARCHAR(15))
- `HireDate` (DATE)

After creating the table, perform the following operations:

1. Add a new column `Department` (VARCHAR(50)).
2. Drop the column `PhoneNumber`.
3. Drop the `Employees` table.

Output:

```
    -> VALUES (1, 'John', 'Doe', 'john.doe@example.com', '2023-01-15'),
    ->        (2, 'Jane', 'Smith', 'jane.smith@example.com', '2023-03-22');
ERROR 1046 (3D000): No database selected
mysql> CREATE DATABASE CompanyDB;
Query OK, 1 row affected (0.00 sec)

mysql> USE CompanyDB;
Database changed
mysql> CREATE TABLE Employees (
    ->     EmployeeID INT PRIMARY KEY,
    ->     FirstName VARCHAR(50),
    ->     LastName VARCHAR(50),
    ->     Email VARCHAR(100),
    ->     HireDate DATE
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> DESCRIBE Employees;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| EmployeeID | int         | NO   | PRI | NULL    |       |
| FirstName  | varchar(50) | YES  |     | NULL    |       |
| LastName   | varchar(50) | YES  |     | NULL    |       |
| Email      | varchar(100)| YES  |     | NULL    |       |
| HireDate   | date        | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)

mysql> INSERT INTO Employees (EmployeeID, FirstName, LastName, Email, HireDate)
    -> VALUES (1, 'John', 'Doe', 'john.doe@example.com', '2023-01-15'),
    ->        (2, 'Jane', 'Smith', 'jane.smith@example.com', '2023-03-22');
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM Employees;
+------------+-----------+----------+-----------------------+------------+
| EmployeeID | FirstName | LastName | Email                 | HireDate   |
+------------+-----------+----------+-----------------------+------------+
|          1 | John      | Doe      | john.doe@example.com  | 2023-01-15 |
|          2 | Jane      | Smith    | jane.smith@example.com | 2023-03-22 |
+------------+-----------+----------+-----------------------+------------+
2 rows in set (0.00 sec)

mysql> ALTER TABLE Employees
    -> ADD Department VARCHAR(50);
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE Employees
    -> DROP COLUMN HireDate;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DROP TABLE Employees;
Query OK, 0 rows affected (0.01 sec)

mysql> DROP DATABASE CompanyDB;
Query OK, 0 rows affected (0.01 sec)

mysql>
```

Practical :3

Name of Practical:

Performing DML Commands in SQL

Description of Practical:

This practical focuses on Data Manipulation Language (DML) commands such as `INSERT`, `UPDATE`, `DELETE`, and `SELECT`. These commands are used to modify data within database tables.

Question:

1. Create a database named `SchoolDB`.
2. Create a table named `Students` with the following columns:
    - `StudentID` (INT, Primary Key)
    - `FirstName` (VARCHAR(50))
    - `LastName` (VARCHAR(50))
    - `Email` (VARCHAR(100))
    - `EnrollmentDate` (DATE)
3. Insert data into the `Students` table.
4. Update the `Email` of a student.
5. Delete a student record.
6. Retrieve all data from the `Students` table.

Output:

```
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE DATABASE SchoolDB;
Query OK, 1 row affected (0.01 sec)

mysql> USE SchoolDB;
Database changed
mysql> CREATE TABLE Students (
    ->     StudentID INT PRIMARY KEY,
    ->     FirstName VARCHAR(50),
    ->     LastName VARCHAR(50),
    ->     Email VARCHAR(100),
    ->     EnrollmentDate DATE
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO Students (StudentID, FirstName, LastName, Email, EnrollmentDate)
    -> VALUES (1, 'Alice', 'Johnson', 'alice.johnson@example.com', '2023-08-01'),
    ->        (2, 'Bob', 'Williams', 'bob.williams@example.com', '2023-09-10'),
    ->        (3, 'Charlie', 'Brown', 'charlie.brown@example.com', '2023-07-15');
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> UPDATE Students
    -> SET Email = 'alice.newemail@example.com'
    -> WHERE StudentID = 1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> DELETE FROM Students
    -> WHERE StudentID = 3;
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM Students;
+-----------+-----------+----------+---------------------------+----------------+
| StudentID | FirstName | LastName | Email                     | EnrollmentDate |
+-----------+-----------+----------+---------------------------+----------------+
|         1 | Alice     | Johnson  | alice.newemail@example.com | 2023-08-01     |
|         2 | Bob       | Williams | bob.williams@example.com   | 2023-09-10     |
+-----------+-----------+----------+---------------------------+----------------+
2 rows in set (0.00 sec)

mysql>
```

Practical: 4

Name of Practical:

Performing DQL Commands in SQL

Description of Practical:

This practical involves using Data Query Language (DQL) commands, focusing on the `SELECT` statement and various operations like filtering, sorting, grouping, and joining data from tables. These queries are used to retrieve data and display it in a meaningful way.

Question:

1. Create a database named `LibraryDB`.
2. Create two tables: `Books` and `Authors`.
    - The `Books` table should have the following columns:
        - `BookID` (INT, Primary Key)
        - `Title` (VARCHAR(100))
        - `AuthorID` (INT, Foreign Key references `Authors`)
        - `PublishedYear` (YEAR)
    - The `Authors` table should have the following columns:
        - `AuthorID` (INT, Primary Key)
        - `AuthorName` (VARCHAR(100))
3. Insert sample data into the `Books` and `Authors` tables.
4. Perform various DQL operations:
    - Retrieve all books written by a specific author.
    - Retrieve books published after a certain year.
    - Show a list of authors along with the number of books they've written.
5. Show all tables in the database.

Output:

```
mysql> USE LibraryDB;
Database changed
mysql> CREATE TABLE Books (
    ->     BookID INT PRIMARY KEY,
    ->     Title VARCHAR(100),
    ->     AuthorID INT,
    ->     PublishedYear YEAR,
    ->     FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID)
    -> );
ERROR 1824 (HY000): Failed to open the referenced table 'Authors'
mysql> CREATE TABLE Authors (
    ->     AuthorID INT PRIMARY KEY,
    ->     AuthorName VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO Authors (AuthorID, AuthorName)
    -> VALUES (1, 'J.K. Rowling'),
    ->        (2, 'George Orwell'),
    ->        (3, 'Jane Austen');
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Books (BookID, Title, AuthorID, PublishedYear)
    -> VALUES (1, 'Harry Potter and the Sorcerer\'s Stone', 1, 1997),
    ->        (2, '1984', 2, 1949),
    ->        (3, 'Pride and Prejudice', 3, 1813),
    ->        (4, 'Harry Potter and the Chamber of Secrets', 1, 1998);
ERROR 1146 (42S02): Table 'librarydb.books' doesn't exist
mysql> SELECT Title, PublishedYear
    -> FROM Books
    -> JOIN Authors ON Books.AuthorID = Authors.AuthorID
    -> WHERE AuthorName = 'J.K. Rowling';
ERROR 1146 (42S02): Table 'librarydb.books' doesn't exist
mysql> SELECT Title, PublishedYear
    -> FROM Books
    -> WHERE PublishedYear > 1950;
ERROR 1146 (42S02): Table 'librarydb.books' doesn't exist
mysql> SELECT Authors.AuthorName, COUNT(Books.BookID) AS NumberOfBooks
    -> FROM Authors
    -> LEFT JOIN Books ON Authors.AuthorID = Books.AuthorID
    -> GROUP BY Authors.AuthorName;
ERROR 1146 (42S02): Table 'librarydb.books' doesn't exist
mysql> SHOW TABLES;
+---------------------+
| Tables_in_librarydb |
+---------------------+
| Authors             |
+---------------------+
1 row in set (0.00 sec)

mysql> CREATE DATABASE LibraryDB;
ERROR 1007 (HY000): Can't create database 'LibraryDB'; database exists
mysql> USE LibraryDB;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+---------------------+
| Tables_in_librarydb |
+---------------------+
| Authors             |
+---------------------+
```

```
mysql> DROP DATABASE LibraryDB;
Query OK, 1 row affected (0.01 sec)

mysql> CREATE DATABASE LibraryDB;
Query OK, 1 row affected (0.01 sec)

mysql> USE LibraryDB;
Database changed
mysql> CREATE TABLE Authors (
    ->     AuthorID INT PRIMARY KEY,
    ->     AuthorName VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE Books (
    ->     BookID INT PRIMARY KEY,
    ->     Title VARCHAR(100),
    ->     AuthorID INT,
    ->     PublishedYear YEAR,
    ->     FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO Authors (AuthorID, AuthorName)
    -> VALUES (1, 'J.K. Rowling'),
    ->        (2, 'George Orwell'),
    ->        (3, 'Jane Austen');
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Books (BookID, Title, AuthorID, PublishedYear)
    -> VALUES (1, 'Harry Potter and the Sorcerer\'s Stone', 1, 1997),
    ->        (2, '1984', 2, 1949),
    ->        (3, 'Pride and Prejudice', 3, 1813),
    ->        (4, 'Harry Potter and the Chamber of Secrets', 1, 1998);
ERROR 1264 (22003): Out of range value for column 'PublishedYear' at row 3
mysql> SELECT Title, PublishedYear
    -> FROM Books
    -> WHERE PublishedYear > 1950;
Empty set (0.00 sec)

mysql> SELECT Authors.AuthorName, COUNT(Books.BookID) AS NumberOfBooks
    -> FROM Authors
    -> LEFT JOIN Books ON Authors.AuthorID = Books.AuthorID
    -> GROUP BY Authors.AuthorName;
+---------------+---------------+
| AuthorName    | NumberOfBooks |
+---------------+---------------+
| J.K. Rowling  |             0 |
| George Orwell |             0 |
| Jane Austen   |             0 |
+---------------+---------------+
3 rows in set (0.00 sec)

mysql> SELECT Authors.AuthorName, COUNT(Books.BookID) AS NumberOfBooks
    -> FROM Authors
    -> LEFT JOIN Books ON Authors.AuthorID = Books.AuthorID
    -> GROUP BY Authors.AuthorName;
+---------------+---------------+
| AuthorName    | NumberOfBooks |
+---------------+---------------+
| J.K. Rowling  |             0 |
| George Orwell |             0 |
| Jane Austen   |             0 |
+---------------+---------------+
```

Practical 5:

Name of Practical:

Performing SQL Joins in a Healthcare Database

Description of Practical:

This practical demonstrates how to use SQL joins to retrieve related information from multiple tables within a healthcare database. It covers the use of `INNER JOIN`, `LEFT JOIN`, and `RIGHT JOIN` to retrieve data about patients and their assigned doctors.

Question:

1. Create a database named `HealthcareDB`.
2. Create two tables: `Patients` and `Doctors`.
    - The `Patients` table should have the following columns:
        - `PatientID` (INT, Primary Key)
        - `PatientName` (VARCHAR(100))
        - `DoctorID` (INT, Foreign Key references `Doctors`)
    - The `Doctors` table should have the following columns:
        - `DoctorID` (INT, Primary Key)
        - `DoctorName` (VARCHAR(100))
3. Insert sample data into the `Patients` and `Doctors` tables.
4. Perform various join operations:

- Use `INNER JOIN` to retrieve patient names along with their assigned doctors.
- Use `LEFT JOIN` to show all doctors and the patients assigned to them, including doctors with no patients.
- Use `RIGHT JOIN` to show all patients, including those not assigned to any doctor.

```
mysql> CREATE DATABASE HealthcareDB;
Query OK, 1 row affected (0.01 sec)

mysql> USE HealthcareDB;
Database changed
mysql> CREATE TABLE Doctors (
    ->     DoctorID INT PRIMARY KEY,
    ->     DoctorName VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE Patients (
    ->     PatientID INT PRIMARY KEY,
    ->     PatientName VARCHAR(100),
    ->     DoctorID INT,
    ->     FOREIGN KEY (DoctorID) REFERENCES Doctors(DoctorID)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO Doctors (DoctorID, DoctorName)
    -> VALUES (1, 'Dr. Smith'),
    ->        (2, 'Dr. Brown'),
    ->        (3, 'Dr. Johnson');
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Patients (PatientID, PatientName, DoctorID)
    -> VALUES (1, 'John Doe', 1),
    ->        (2, 'Jane Smith', 2),
    ->        (3, 'Bob Johnson', NULL);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT Patients.PatientName, Doctors.DoctorName
    -> FROM Patients
    -> INNER JOIN Doctors ON Patients.DoctorID = Doctors.DoctorID;
+-------------+------------+
| PatientName | DoctorName |
+-------------+------------+
| John Doe    | Dr. Smith  |
| Jane Smith  | Dr. Brown  |
+-------------+------------+
2 rows in set (0.01 sec)

mysql> SELECT Doctors.DoctorName, Patients.PatientName
    -> FROM Doctors
    -> LEFT JOIN Patients ON Doctors.DoctorID = Patients.DoctorID;
+-------------+-------------+
| DoctorName  | PatientName |
+-------------+-------------+
| Dr. Smith   | John Doe    |
| Dr. Brown   | Jane Smith  |
| Dr. Johnson | NULL        |
+-------------+-------------+
3 rows in set (0.00 sec)
```

```
mysql> SELECT Doctors.DoctorName, Patients.PatientName
    -> FROM Doctors
    -> LEFT JOIN Patients ON Doctors.DoctorID = Patients.DoctorID;
+-------------+-------------+
| DoctorName  | PatientName |
+-------------+-------------+
| Dr. Smith   | John Doe    |
| Dr. Brown   | Jane Smith  |
| Dr. Johnson | NULL        |
+-------------+-------------+
3 rows in set (0.00 sec)

mysql> SELECT Patients.PatientName, Doctors.DoctorName
    -> FROM Patients
    -> RIGHT JOIN Doctors ON Patients.DoctorID = Doctors.DoctorID;
+-------------+-------------+
| PatientName | DoctorName  |
+-------------+-------------+
| John Doe    | Dr. Smith   |
| Jane Smith  | Dr. Brown   |
| NULL        | Dr. Johnson |
+-------------+-------------+
3 rows in set (0.00 sec)

mysql> SHOW TABLES;
+------------------------+
| Tables_in_healthcaredb |
+------------------------+
| Doctors                |
| Patients               |
+------------------------+
2 rows in set (0.00 sec)

mysql>
```

Practical 6:

Name of Practical:

Performing SQL Subqueries in a University Database

Description of Practical:

This practical covers the use of SQL subqueries (queries within queries) to retrieve data from a table based on the results of another query. Subqueries are often used for complex filtering and aggregation.

Question:

1. Create a database named `UniversityDB`.
2. Create two tables: `Students` and `Courses`.
    - The `Students` table should have the following columns:
        - `StudentID` (INT, Primary Key)
        - `StudentName` (VARCHAR(100))
        - `CourseID` (INT, Foreign Key references `Courses`)
        - `Score` (INT)
    - The `Courses` table should have the following columns:
        - `CourseID` (INT, Primary Key)
        - `CourseName` (VARCHAR(100))
3. Insert sample data into the `Students` and `Courses` tables.
4. Perform various subqueries:
    - Retrieve the names of students who have scored above the average score.
    - Use a subquery to retrieve the names of students who are enrolled in the same course as 'Alice'.
    - Retrieve the details of the student(s) with the highest score using a subquery.

```
mysql> CREATE DATABASE UniversityDB;
Query OK, 1 row affected (0.00 sec)

mysql> USE UniversityDB;
Database changed
mysql> CREATE TABLE Courses (
    ->     CourseID INT PRIMARY KEY,
    ->     CourseName VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE Students (
    ->     StudentID INT PRIMARY KEY,
    ->     StudentName VARCHAR(100),
    ->     CourseID INT,
    ->     Score INT,
    ->     FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO Courses (CourseID, CourseName)
    -> VALUES (1, 'Mathematics'),
    ->        (2, 'Physics'),
    ->        (3, 'Chemistry');
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Students (StudentID, StudentName, CourseID, Score)
    -> VALUES (1, 'Alice', 1, 85),
    ->        (2, 'Bob', 2, 90),
    ->        (3, 'Charlie', 1, 75),
    ->        (4, 'David', 3, 95),
    ->        (5, 'Eve', 2, 88);
Query OK, 5 rows affected (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT StudentName
    -> FROM Students
    -> WHERE Score > (SELECT AVG(Score) FROM Students);
+-------------+
| StudentName |
+-------------+
| Bob         |
| David       |
| Eve         |
+-------------+
3 rows in set (0.00 sec)

mysql> SELECT StudentName
    -> FROM Students
    -> WHERE CourseID = (SELECT CourseID FROM Students WHERE StudentName = 'Alice');
+-------------+
| StudentName |
+-------------+
| Alice       |
| Charlie     |
+-------------+
2 rows in set (0.01 sec)
```

```
mysql> SELECT StudentID, StudentName, Score
    -> FROM Students
    -> WHERE Score = (SELECT MAX(Score) FROM Students);
+-----------+-------------+-------+
| StudentID | StudentName | Score |
+-----------+-------------+-------+
|         4 | David       |    95 |
+-----------+-------------+-------+
1 row in set (0.00 sec)

mysql> SHOW TABLES;
+------------------------+
| Tables_in_universitydb |
+------------------------+
| Courses                |
| Students               |
+------------------------+
2 rows in set (0.01 sec)

mysql>
```

Practical 7:

Name of Practical:

Performing SQL Views in a Retail Store Database

Description of Practical:

This practical covers the use of SQL views, which are virtual tables representing the result of a query. Views are useful for simplifying complex queries, providing a specific dataset to users, and enhancing security by limiting access to certain columns of a table.

Question:

1. Create a database named `RetailStoreDB`.
2. Create two tables: `Products` and `Orders`.
    - The `Products` table should have the following columns:
        - `ProductID` (INT, Primary Key)
        - `ProductName` (VARCHAR(100))
        - `Price` (DECIMAL(10, 2))
    - The `Orders` table should have the following columns:
        - `OrderID` (INT, Primary Key)
        - `ProductID` (INT, Foreign Key references `Products`)
        - `Quantity` (INT)
3. Insert sample data into the `Products` and `Orders` tables.
4. Perform the following tasks:
    - Create a view called `OrderSummary` that lists the product name, quantity, and total price for each order.
    - Query the `OrderSummary` view to display all orders.
    - Create a view called `ExpensiveProducts` that lists all products with a price above $50.
    - Query the `ExpensiveProducts` view to display products.

```
2 rows in set (0.01 sec)

mysql> CREATE DATABASE RetailStoreDB;
Query OK, 1 row affected (0.00 sec)

mysql> USE RetailStoreDB;
Database changed
mysql> CREATE TABLE Products (
    ->     ProductID INT PRIMARY KEY,
    ->     ProductName VARCHAR(100),
    ->     Price DECIMAL(10, 2)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE Orders (
    ->     OrderID INT PRIMARY KEY,
    ->     ProductID INT,
    ->     Quantity INT,
    ->     FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO Products (ProductID, ProductName, Price)
    -> VALUES (1, 'Laptop', 1200.00),
    ->        (2, 'Phone', 800.00),
    ->        (3, 'Tablet', 300.00),
    ->        (4, 'Headphones', 50.00),
    ->        (5, 'Mouse', 20.00);
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Orders (OrderID, ProductID, Quantity)
    -> VALUES (1, 1, 2),
    ->        (2, 2, 1),
    ->        (3, 3, 3),
    ->        (4, 4, 5),
    ->        (5, 5, 10);
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> CREATE VIEW OrderSummary AS
    -> SELECT Products.ProductName, Orders.Quantity, (Products.Price * Orders.Quantity) AS TotalPrice
    -> FROM Orders
    -> JOIN Products ON Orders.ProductID = Products.ProductID;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM OrderSummary;
+-------------+----------+------------+
| ProductName | Quantity | TotalPrice |
+-------------+----------+------------+
| Laptop      |        2 |    2400.00 |
| Phone       |        1 |     800.00 |
| Tablet      |        3 |     900.00 |
| Headphones  |        5 |     250.00 |
| Mouse       |       10 |     200.00 |
+-------------+----------+------------+
5 rows in set (0.01 sec)

mysql> CREATE VIEW ExpensiveProducts AS
    -> SELECT ProductName, Price
    -> FROM Products
    -> WHERE Price > 50;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM ExpensiveProducts;
```

```
mysql> SELECT * FROM ExpensiveProducts;
+-------------+---------+
| ProductName | Price   |
+-------------+---------+
| Laptop      | 1200.00 |
| Phone       |  800.00 |
| Tablet      |  300.00 |
+-------------+---------+
3 rows in set (0.00 sec)

mysql> SHOW FULL TABLES WHERE Table_Type = 'VIEW';
+--------------------------+------------+
| Tables_in_retailstoredb | Table_type |
+--------------------------+------------+
| expensiveproducts       | VIEW       |
| ordersummary            | VIEW       |
+--------------------------+------------+
2 rows in set (0.01 sec)

mysql>
```

Practical 8:

Name of Practical:

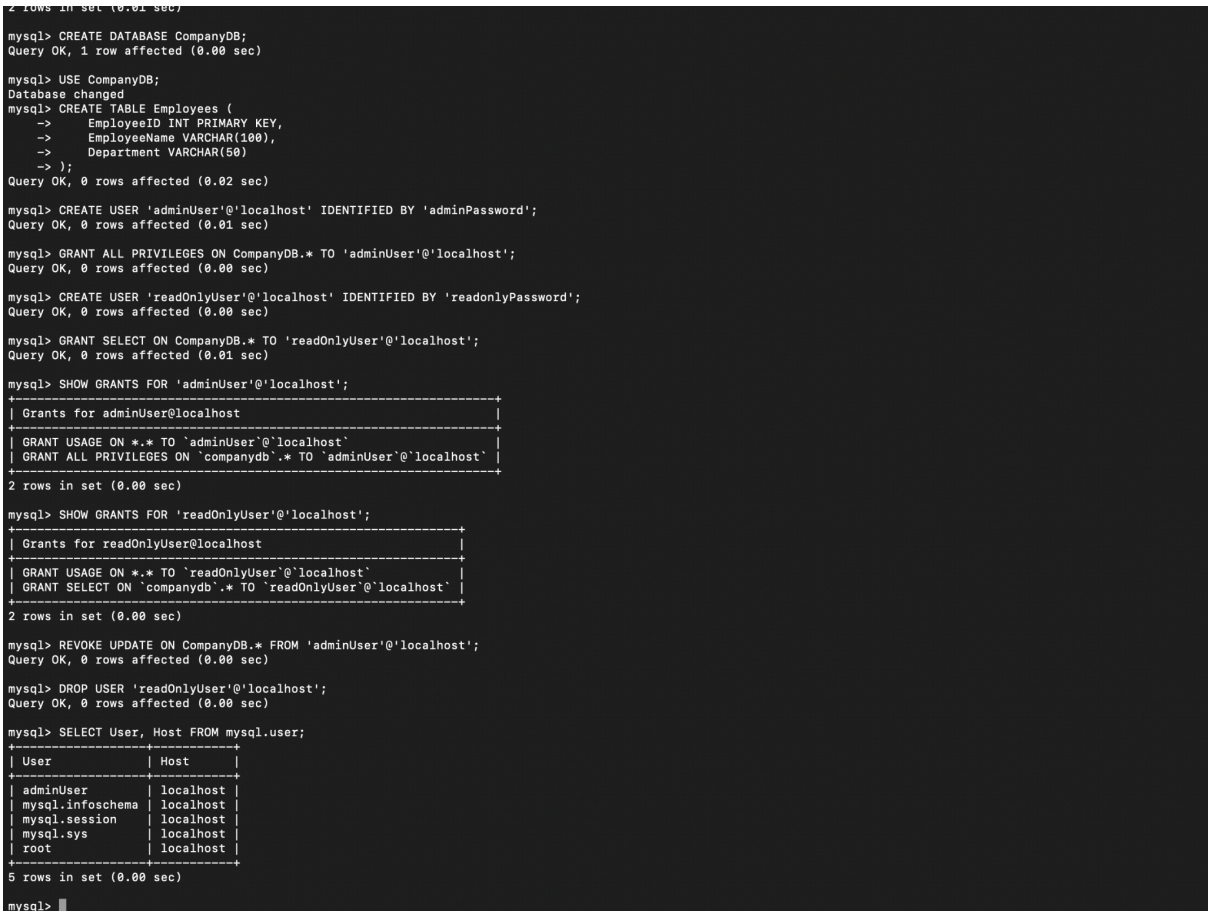
Performing User Management in a Company Database

Description of Practical:

This practical covers the basic steps of creating users, granting them specific privileges, and managing roles in a MySQL database. It demonstrates how to set up users with different levels of access, revoke permissions, and remove users when necessary.

Question:

1. Create a database named `CompanyDB`.
2. Create users with different access levels:
    - Create an admin user with full privileges on the `CompanyDB`.
    - Create a read-only user who can only select data from tables in `CompanyDB`.
3. Grant and revoke permissions as required.
4. Drop the users when they are no longer needed.



```
2 rows in set (0.01 sec)

mysql> CREATE DATABASE CompanyDB;
Query OK, 1 row affected (0.00 sec)

mysql> USE CompanyDB;
Database changed
mysql> CREATE TABLE Employees (
    ->     EmployeeID INT PRIMARY KEY,
    ->     EmployeeName VARCHAR(100),
    ->     Department VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE USER 'adminUser'@'localhost' IDENTIFIED BY 'adminPassword';
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT ALL PRIVILEGES ON CompanyDB.* TO 'adminUser'@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE USER 'readOnlyUser'@'localhost' IDENTIFIED BY 'readonlyPassword';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT SELECT ON CompanyDB.* TO 'readOnlyUser'@'localhost';
Query OK, 0 rows affected (0.01 sec)

mysql> SHOW GRANTS FOR 'adminUser'@'localhost';
+-----------------------------------------------------------------------+
| Grants for adminUser@localhost                                        |
+-----------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `adminUser`@`localhost`                         |
| GRANT ALL PRIVILEGES ON `companydb`.* TO `adminUser`@`localhost` |
+-----------------------------------------------------------------------+
2 rows in set (0.00 sec)

mysql> SHOW GRANTS FOR 'readOnlyUser'@'localhost';
+-----------------------------------------------------------------+
| Grants for readOnlyUser@localhost                               |
+-----------------------------------------------------------------+
| GRANT USAGE ON *.* TO `readOnlyUser`@`localhost`                |
| GRANT SELECT ON `companydb`.* TO `readOnlyUser`@`localhost` |
+-----------------------------------------------------------------+
2 rows in set (0.00 sec)

mysql> REVOKE UPDATE ON CompanyDB.* FROM 'adminUser'@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> DROP USER 'readOnlyUser'@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT User, Host FROM mysql.user;
+------------------+-----------+
| User             | Host      |
+------------------+-----------+
| adminUser        | localhost |
| mysql.infoschema | localhost |
| mysql.session    | localhost |
| mysql.sys        | localhost |
| root             | localhost |
+------------------+-----------+
5 rows in set (0.00 sec)

mysql>
```

Practical Conclusion:

In this practical, you learned how to create and manage users in MySQL, grant and revoke privileges, and drop users when no longer necessary. This setup is important for managing access control in any database environment.