1) Statement to create the Contact table.

→ CREATE TABLE Contact (

ContactID INT,

CompanyID INT,

Firstname VARCHAR(50),

Lastname VARCHAR(50),

Street VARCHAR(50),

City VARCHAR(50),

State VARCHAR(50),

Zip VARCHAR(10),

IsMain Boolean,

Email VARCHAR(45) UNIQUE,

Phone VARCHAR(15));

2) Statement to create the Employee table.

→ CREATE TABLE Employee(

EmployeeID INT,

Firstname VARCHAR(40),

Lastname VARCHAR(40),

Salary INT,

HireDate DATE,

JobTitle VARCHAR(30),

Email VARCHAR(50),

Phone VARCHAR(12));

3) Statement to create the ContactEmployee table.

→ Create Table ContactEmployee(

ContactEmployeeID INT PRIMARY KEY,

ContactID INT,

```
  ContactDate DATE,

  Description VARCHAR(100));
```

4) **In the Employee table, the statement that changes Lesley Bland's phone number to 215-555-8800.**

```
→ UPDATE Employee

  SET PHONE= ‘215-555-8800’

 WHERE Firstname= ‘Lesley’;
```

5) **In the Company table, the statement that changes the name of "Urban Outfitters,Inc." to "Urban Outfitters".**

```
→ UPDATE Company

  SET Companyname = 'Urban Outfitters'

  WHERE Companyname = 'Urban Outfitters, Inc.';
```

6) **In ContactEmployee table, the statement that removes Dianne Connor's contact even with Jack Lee (one statement).**

```
→ DELETE FROM ContactEmployee

  Where  contactemployeeid =(

  SELECT  contactemployeeid

  FROM contactemployee  ce

  JOIN Contact c  ON ce.ContactID = c.ContactID

  JOIN Employee e ON ce.ContactID = e.EmployeeID

  WHERE c.FirstName = 'Dianne' AND c.LastName = 'Connor'

  AND e.FirstName = 'Jack' AND e.LastName = 'Lee'

  );
```

7) **Write the SQL SELECT query that displays the names of the employees that have contacted Toll Brothers (one statement). Run the SQL SELECT query in MySQL Workbench. Copy the results below as well.**

→ **SELECT DISTINCT e.FirstName, e.LastName**

FROM Employee e

JOIN Contact c ON e.EmployeeID = c.EmployeeID

JOIN Company co ON c.CompanyID = co.CompanyID

WHERE co.CompanyName = 'Toll Brothers';

8) What is the significance of "%" and "_" operators in the LIKE statement?

→ **In SQL, the LIKE statement is used to search for a specified pattern in a column. The % and _ operators are wildcards that play a key role in defining these patterns.**

" %"(Percent Sign): Represents zero or more characters. It matches any sequence of characters (including none) in the specified position.

- **Example: WHERE name LIKE 'A%' matches any string starting with 'A', such as 'Apple', 'Ant', or 'A'.**
- **Example: WHERE name LIKE '%ing' matches any string ending with 'ing', like 'Spring' or 'King'.**
- **Example: WHERE name LIKE '%an%' matches any string containing 'an', such as 'Banana', 'Mango', or 'Hand'.**

" _ "(Underscore): Represents exactly one character. It matches any single character in the specified position.

- **Example: WHERE name LIKE 'A_ple' matches strings like 'Apple' or 'Ample', where the second character can be anything, but the string must be exactly five characters long.**
- **Example: WHERE name LIKE '_o_' matches three-character strings where the second character is 'o', like 'Fox' or 'Dog'.**
- They are case-sensitive in most SQL databases, unless the database is configured otherwise.
- To search for literal % or _ characters, they must be escaped (e.g., LIKE '%\%%' to find strings containing a percent sign).

9) Explain normalization in the context of databases.

→ **Normalization in the context of databases is the process of organizing data in a relational database to eliminate redundancy, ensure data integrity, and optimize storage efficiency. It involves structuring tables and their relationships to adhere to specific rules, called normal forms, which reduce anomalies during data operations like insertion, deletion, and updates.**

Key Objectives of Normalization

1. **Eliminate Data Redundancy**

2. **Ensure Data Integrity**
3. **Simplify Data Management**
4. **Optimize Storage**

10)What does a join in MySQL mean?

→ In MySQL, a JOIN is a SQL operation that combines rows from two or more tables based on a related column between them, allowing you to retrieve data from multiple tables as if they were a single table. The JOIN operation is fundamental in relational databases to query related data stored across normalized tables.

11) What do you understand about DDL, DCL, and DML in MySQL?

→ In MySQL, DDL, DCL, and DML are categories of SQL (Structured Query Language) commands used to interact with a database. Each serves a distinct purpose in managing and manipulating database structure, access, and data.

1. DDL (Data Definition Language)

- **Purpose: DDL commands are used to define, modify, or delete the structure of database objects like tables, schemas, or indexes.**

2. DML (Data Manipulation Language)

- **Purpose: DML commands are used to manipulate the data within database tables (e.g., inserting, updating, or deleting records).**

3. DCL (Data Control Language)

- **Purpose: DCL commands are used to define access controls and permissions for database users or roles.**

- **DDL**: Create a table for employees (CREATE TABLE employees).
- **DML**: Add employee data (INSERT INTO employees) or retrieve it (SELECT * FROM employees).
- **DCL**: Grant a user permission to view the table (GRANT SELECT ON employees TO 'user1').

12) What is the role of the MySQL JOIN clause in a query, and what are some common types of joins?

→ The JOIN clause in MySQL is used to combine rows from two or more tables based on a related column, enabling you to retrieve data from multiple tables in a single query. It is essential in relational databases where data is normalized across multiple tables, and you need to link them to produce meaningful results.

Role of the JOIN Clause

- **Combine Data: JOIN links tables based on a specified condition, typically involving primary and foreign keys, to fetch related data.**
- **Enable Relational Queries: It allows you to query data spread across normalized tables, such as combining customer and order information.**
- **Flexible Data Retrieval: By specifying the type of JOIN, you control which rows are included in the result set, even when matches are not found in one of the tables.**

Common Types of JOINs in MySQL

1. **INNER JOIN (or simply JOIN):**
    - **Description: Returns only the rows where there is a match in both tables based on the ON condition.**
    - **Use: When you want data that exists in both tables.**
2. **LEFT JOIN (or LEFT OUTER JOIN):**

- **Description: Returns all rows from the left table (first table) and the matched rows from the right table. If no match is found, NULL values are returned for columns from the right table.**
- **Use: When you want all records from the left table, even if there are no corresponding matches in the right table.**

3. **RIGHT JOIN (or RIGHT OUTER JOIN):**

- **Description: Returns all rows from the right table and the matched rows from the left table. If no match is found, NULL values are returned for columns from the left table.**
- **Use: When you want all records from the right table, even if there are no matches in the left table.**

4. **FULL JOIN (or FULL OUTER JOIN):**

- **Description: Returns all rows from both tables, with NULL values in places where there is no match in one of the tables.**
- **Note: MySQL does not natively support FULL OUTER JOIN, but it can be achieved using a combination of LEFT JOIN and RIGHT JOIN with UNION.**
- **Use: When you want all records from both tables, regardless of matches.**