ASSIGNMENT

SQL (module – 4)

Working with database using SQL

- For this assignment, you will finish building the contact management database for MarketCo :

```
create database MarketCo;
use MarketCo;

create table company (
    companyid INT primary key,
       comp_name varchar(45),
    street varchar(45),
    city varchar(45),
    state varchar(2),
    zip varchar(10)
);

create table contact (
    contactid INt primary key,
    companyid INT,
    first_name varchar(45),
    last_name varchar(45),
    street varchar(45),
    city varchar(45),
    state varchar (2),
    zip varchar(10),
    Ismain boolean,
    Email varchar(45),
    phone varchar(12)
);

create table contactemployee (
        contactempid INT primary key ,
    contactid INT ,
    empid INt ,
    contactdate DATE ,
    description varchar(100)
);

create table employee (
    empid INT primary key,
    first_name varchar(45),
```

```sql
    last_name varchar(45),
    salary decimal(10,2),
    hiredate DATE ,
    job_title varchar(25),
    Email varchar(45),
    phone varchar(12)
);

INSERT INTO company (companyid, comp_name, street, city, state, zip)
VALUES
(1, 'TechVision Pvt Ltd', '12 Nehru Street', 'Ahmedabad', 'GJ', '380015'),
(2, 'GreenSoft Solutions', '89 Park Avenue', 'Mumbai', 'MH', '400001'),
(3, 'Skyline Industries', '45 MG Road', 'Bangalore', 'KA', '560001'),
(4, 'BrightFuture Ltd', '78 Civil Lines', 'Delhi', 'DL', '110001'),
(5, 'DataCore Systems', '56 Jubilee Hills', 'Hyderabad', 'TS', '500033'),
(6, 'Urban Outfitters, Inc.', '500 Fashion Ave', 'New York', 'NY', '10018'),
(7, 'Toll Brothers', '1200 Liberty St', 'Philadelphia', 'PA', '19107'),
(8, 'BlueTech Solutions', '33 Tech Park', 'San Jose', 'CA',
'95112');


INSERT INTO contact (contactid, companyid, first_name,
last_name, street, city, state, zip, Ismain, Email, phone)
VALUES
(1, 1, 'Ravi', 'Patel', '12 Nehru Street', 'Ahmedabad', 'GJ',
'380015', TRUE, 'ravi.patel@techvision.com', '9876543210'),
(2, 2, 'Priya', 'Sharma', '89 Park Avenue', 'Mumbai', 'MH',
'400001', TRUE, 'priya.sharma@greensoft.com',
'9988776655'),
(3, 3, 'Amit', 'Verma', '45 MG Road', 'Bangalore', 'KA',
'560001', TRUE, 'amit.verma@skyline.com', '9123456780'),
(4, 4, 'Kiran', 'Mehta', '78 Civil Lines', 'Delhi', 'DL', '110001',
TRUE, 'kiran.mehta@brightfuture.com', '9090909090'),
(5, 5, 'Sneha', 'Rao', '56 Jubilee Hills', 'Hyderabad', 'TS',
'500033', TRUE, 'sneha.rao@datacore.com', '9876501234'),
(6, 1, 'Vikas', 'Jain', '13 Nehru Street', 'Ahmedabad', 'GJ',
'380015', FALSE, 'vikas.jain@techvision.com', '9822334455'),
(7, 2, 'Dianne', 'Connor', '500 Fashion Ave', 'New York', 'NY',
'10018', TRUE, 'dianne.connor@urban.com', '212-555-
1234'),
(8, 3, 'Mark', 'Stone', '1200 Liberty St', 'Philadelphia', 'PA',
'19107', TRUE, 'mark.stone@tollbrothers.com', '215-555-
3333');
```

```sql
INSERT INTO contactemployee (contactempid, contactid, empid, contactdate, description)
VALUES
(1, 1, 101, '2023-06-10', 'Meeting about data migration project'),
(2, 2, 102, '2023-07-12', 'Discussion about software development contract'),
(3, 3, 103, '2023-08-25', 'HR policy collaboration meeting'),
(4, 4, 104, '2023-09-05', 'Project timeline and delivery discussion'),
(5, 5, 105, '2023-10-01', 'Marketing campaign strategy call'),
(6, 6, 101, '2023-10-15', 'Follow-up meeting with TechVision secondary contact'),
(7, 7, 101, '2023-09-05', 'Sales meeting between Dianne Connor and Lesley Bland'),
(8, 8, 102, '2023-09-20', 'Technical review between Dianne Connor and Jack Lee'),
(9, 9, 102, '2023-10-10', 'Project meeting with Toll Brothers'),
(10, 10, 103, '2023-10-25', 'Marketing partnership discussion with Toll Brothers');

INSERT INTO employee (empid, first_name, last_name, salary, hiredate, job_title, Email, phone)
```

VALUES

(101, 'Neha', 'Kapoor', 75000.00, '2020-03-15', 'Data Analyst', 'neha.kapoor@techvision.com', '9001122233'),

(102, 'Rohit', 'Singh', 95000.00, '2019-08-20', 'Software Engineer', 'rohit.singh@greensoft.com', '9112233445'),

(103, 'Anjali', 'Nair', 65000.00, '2021-01-10', 'HR Manager', 'anjali.nair@skyline.com', '9223344556'),

(104, 'Suresh', 'Gupta', 85000.00, '2018-06-01', 'Project Manager', 'suresh.gupta@brightfuture.com', '9334455667'),

(105, 'Pooja', 'Iyer', 72000.00, '2022-05-12', 'Marketing Lead', 'pooja.iyer@datacore.com', '9445566778'),

(106, 'Lesley', 'Bland', 65000.00, '2020-06-10', 'Sales Manager', 'lesley.bland@urban.com', '215-555-7700'),

(107, 'Jack', 'Lee', 72000.00, '2019-09-15', 'Project Engineer', 'jack.lee@tollbrothers.com', '215-555-9911'),

(108, 'Ava', 'Reed', 85000.00, '2018-05-01', 'Marketing Lead', 'ava.reed@urban.com', '212-555-6622');

Q1) Statement to create the Contact table ?

ANS: create table contact (

    contactid INt primary key,

    companyid INT,

    first_name varchar(45),

    last_name varchar(45),

```
    street varchar(45),
    city varchar(45),
    state varchar (2),
    zip varchar(10),
    Ismain boolean,
    Email varchar(45),
    phone varchar(12)
);
```

Q2) Statement to create the Employee table ?

ANS: create table employee (

```
    empid INT primary key,
    first_name varchar(45),
    last_name varchar(45),
    salary decimal(10,2),
    hiredate DATE ,
    job_title varchar(25),
    Email varchar(45),
    phone varchar(12)
);
```

Q3) Statement to create the ContactEmployee table ?

ANS: CREATE TABLE ContactEmployee (

```
    ContactEmpID INT PRIMARY KEY,
    ContactID INT,
    EmpID INT,
    ContactDate DATE,
    Description VARCHAR(100),
    FOREIGN KEY (ContactID) REFERENCES Contact(ContactID),
    FOREIGN KEY (EmpID) REFERENCES Employee(EmpID)
);
```

Q4) In the Employee table, the statement that changes Lesley Bland's phone number to 215-555-8800 ?

ANS: UPDATE Employee

```
SET Phone = '215-555-8800'
WHERE First_name = 'Lesley' AND Last_name = 'Bland';
```

Q5) In the Company table, the statement that changes the name of "Urban Outfitters, Inc." to "Urban Outfitters" ?

ANS: update company

```
set comp_name = 'Urban Outfitters'
where companyid = 6 ;
```

Q6) In ContactEmployee table, the statement that removes Dianne Connor’s contact event with Jack Lee (one statement). HINT: Use the primary key of the ContactEmployee table to specify the correct record to remove.

ANS: DELETE FROM Contactemployee

```
        WHERE Contactid = 8;
```

select * from Contactemployee;

Q7) 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.

ANS: SELECT CONCAT(e.first_name, ' ', e.last_name) AS EmployeeName, c.Comp_name, ce.Contactdate

```
FROM ContactEmployee ce, Employee e, Company c

WHERE ce.empid = e.empid

  AND c.companyid = c.companyid

  AND c.comp_name = 'Toll Brothers';
```

Q8) What is the significance of “%” and “_” operators in the LIKE statement?

ANS: LIKE is used in SQL to search for a specific pattern in a column — often with wildcard characters.

The two most common wildcards are **%** and **_**.

1) The % (percent) wildcard

- Represents zero, one, or many characters.
- It’s like saying “anything can go here.”

Examples:

Query	**Meaning**
WHERE Name LIKE 'A%'	Finds all names starting with **A** (e.g., *Anna, Ajay, Amit*)
WHERE Name LIKE '%n'	Finds all names ending with **n** (e.g., *John, Rohan*)
WHERE Name LIKE '%an%'	Finds all names that **contain “an”** anywhere (e.g., *Anita, Ranjan*)

---

2) **The _ (underscore) wildcard**

- Represents exactly one character.

Examples:

Query	**Meaning**
WHERE Name LIKE '_im'	Finds all 3-letter names with **‘im’** at the end (e.g., *Kim, Tim*)
WHERE Name LIKE 'A_'	Finds all 2-letter names starting with **A** (e.g., *An, Al*)

Query	Meaning
WHERE Name LIKE '_a_'	Finds all 3-letter names with **'a'** in the middle (e.g., *Sam, Raj*)

Q9) Explain normalization in the context of databases ?

ANS: Normalization is the process of organizing data in a database to reduce redundancy (duplicate data) and improve data integrity (accuracy and consistency).

Why we need normalization:

Without normalization:

- The same data may be repeated in many places
- Updates become hard (you must change it everywhere)
- The database becomes large and inconsistent

With normalization:

Less duplication
Easier updates
Better accuracy
Efficient storage

Normalization Forms (Levels):

Normal Form	**Rule**	**Example**
1NF	Each cell should contain only one value	No repeating groups

Normal Form	**Rule**	**Example**
2NF	Must be in 1NF + No partial dependency	Non-key columns depend on the whole key
3NF	Must be in 2NF + No transitive dependency	Non-key columns depend only on the primary key

Q10) What does a join in MySQL mean?

ANS: A JOIN in MySQL is used to combine data from two or more tables based on a related column between them.

Why we use JOIN:

In a normalized database, data is split into multiple tables (like Employee, Department, etc.).
To see complete information, you must join those tables.

Q11) What do you understand about DDL, DCL, and DML in MySQL?

ANS: **DDL — Data Definition Language**
Commands that define or change the structure of a database (tables, columns, etc.)

Used for: Creating or modifying database objects.

DDL Commands:

Command	**Description**
CREATE	To create a new table or database
ALTER	To modify an existing table (add, delete, or change columns)
DROP	To delete a table or database permanently
TRUNCATE	To remove all records from a table (structure remains

DML — Data Manipulation Language

Commands that handle the data stored inside tables — inserting, updating, or deleting records.

DML Commands:

Command	**Description**
INSERT	Add new records
UPDATE	Modify existing records
DELETE	Remove records
SELECT	Retrieve records (for viewing)

DCL — Data Control Language

Meaning:

👉 Commands that **control access and permissions** in the database (who can do what).

Common DCL Commands:

Command	Description
GRANT	Give permission to users
REVOKE	Take back permission from users

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

ANS : the JOIN clause in MySQL is used to combine data from two or more tables based on a related column between them.

Common Types of JOINs in MySQL

Type	Description	Example Use
INNER JOIN	Returns only matching rows from both tables	Employees who belong to a department
LEFT JOIN	Returns all rows from the left table and matching rows from the right	All employees, even if some have no department
RIGHT JOIN	Returns all rows from the right table and matching from the left	All departments, even if no employee is assigned

Type	Description	Example Use
CROSS JOIN	Returns the Cartesian product (all combinations of rows)	Rarely used, gives every possible pair