**I MCA-D RDBMS Lab(M23DE0108)**

**Prof.Nagaraj C**

**Example:**

**1.**Write a query to create a table Employees with columns **EmpID (Primary Key), FirstName, LastName, and Salary** and describe the table structure.

2. Insert a Row into the Table

3. Insert Multiple Rows

4. Display all rows from the Employees table.

5. Select only FirstName and Salary from the Employees table.

6. Add a new column Email to the Employees table.

7. Update Email for the employee with EmpID = 1 to [alice.johnson@example.com](mailto:alice.johnson@example.com).

8. Delete the employee with EmpID = 4.

9. Remove all rows from the Employees table.

10. Remove the Email column from the Employees table.

11. Delete the entire Employees table from the database.

12. Select all employees with a salary greater than 60000.

13. Update the Salary of all employees to 80000 where LastName = 'Smith'.

14. Count the total number of employees in the Employees table.

15. To apply a 10% salary hike only for employees whose current salary is below 70,000

1. CREATE TABLE Employees (

EmpID NUMBER PRIMARY KEY,

FirstName VARCHAR2(50),

LastName VARCHAR2(50),

Salary NUMBER(10, 2)

);

Desc Employees;

2. INSERT INTO Employees (EmpID, FirstName, LastName, Salary) VALUES (1, 'Alice', 'Johnson', 55000.50);

3. INSERT INTO Employees (EmpID, FirstName, LastName, Salary) VALUES (2, 'Bob', 'Smith', 60000),(3, 'Charlie', 'Brown', 70000),(4, 'Diana', 'Miller', 75000);

4. SELECT \* FROM Employees;

5. SELECT FirstName, Salary FROM Employees;

6. ALTER TABLE Employees ADD Email VARCHAR(100);

7. UPDATE Employees SET Email = 'alice.johnson@example.com' WHERE EmpID = 1;

8. DELETE FROM Employees WHERE EmpID = 4;

9. TRUNCATE TABLE Employees;

10. ALTER TABLE Employees DROP COLUMN Email;

11. DROP TABLE Employees;

12. SELECT \* FROM Employees WHERE Salary > 60000;

13. UPDATE Employees SET Salary = 80000 WHERE LastName = 'Smith';

14. SELECT COUNT(\*) AS TotalEmployees FROM Employees;

15.UPDATE Employees

SET Salary = Salary + (Salary \* 0.10)

WHERE Salary < 70000;

**Sum,avg-(works on numerical data)**

**Count,min and max- (works on numerical and non numerical data.)**

**Example:**

The SALARY database of an organization has a table with the following attributes.

EMPSALARY (empcod:int, emp\_name: string, dob: date, department: string, salary: real)

1. Create the above table.
2. Enter the five tuples into the table
3. Find the sum of the salaries of all employees.
4. Find the sum and average of the salaries of employees of a particular department.
5. Find the least and highest salaries that an employee draws.
6. CREATE TABLE EMPSALARY (

empcod INT PRIMARY KEY,

emp\_name VARCHAR(100),

dob DATE,

department VARCHAR(100),

salary DECIMAL(10, 2)

);

c)SELECT SUM(salary) AS total\_salary

FROM EMPSALARY;

d.SELECT department, SUM(salary) AS total\_salary, AVG(salary) AS average\_salary

FROM EMPSALARY

WHERE department = 'IT';

e)…SELECT MIN(salary) AS least\_salary, MAX(salary) AS highest\_salary

FROM EMPSALARY;

EX.NO:1

DEPARTMENT (dept\_no, dept\_name, location)

1. Create the Simple DEPARTMENT Table.

2. Display structure of department table.

3. Insert below records into Department Table

|  |  |  |
| --- | --- | --- |
| Dept\_no | Dept\_name | Location |
| 10 | Account | NY |
| 20 | HR | NY |
| 30 | Production | DL |
| 40 | Sales | NY |
| 50 | EDP | MU |
| 60 | TRG | DL |
| 110 | RND | AH |

4. Display all records of Department table

5. Display all department belonging to location 'NY'

6. Display details of Department 10

7. List all department names starting with 'A'

8. List all departments whose number is between 1 and 100

9. Delete 'TRG' department

10. Change department name 'EDP' to 'IT

**Queries:**

**1. Create the Simple DEPARTMENT Table**

CREATE TABLE DEPARTMENT (

dept\_no INT PRIMARY KEY,

dept\_name VARCHAR(50) NOT NULL,

location VARCHAR(50));

**2. Display the structure of the DEPARTMENT table**

(In MySQL, use the `DESCRIBE` statement to check the structure)

DESCRIBE DEPARTMENT;

**3. Insert the given records into the DEPARTMENT table**

INSERT INTO DEPARTMENT (dept\_no, dept\_name, location) VALUES

(10, 'Account', 'NY'),(20, 'HR', 'NY'),(30, 'Production', 'DL'),(40, 'Sales', 'NY'),(50, 'EDP', 'MU'),(60, 'TRG', NULL),(110, 'RND', 'AH');

**4. Display all records of the DEPARTMENT table**

SELECT \* FROM DEPARTMENT;

**5. Display all departments belonging to location 'NY'**

SELECT \* FROM DEPARTMENT WHERE location = 'NY';

**6. Display details of Department 10**

SELECT \* FROM DEPARTMENT WHERE dept\_no = 10;

**7. List all department names starting with 'A'**

SELECT \* FROM DEPARTMENT WHERE dept\_name LIKE 'A%';

**8. List all departments whose number is between 1 and 100**

SELECT \* FROM DEPARTMENT WHERE dept\_no BETWEEN 1 AND 100;

**9. Delete 'TRG' department**

DELETE FROM DEPARTMENT WHERE dept\_name = 'TRG';

**10. Change department name 'EDP' to 'IT'**

UPDATE DEPARTMENT SET dept\_name = 'IT' WHERE dept\_name = 'EDP';

ENUM is short for **enumeration**, meaning it consists of a predefined list of values that a variable or column can take.

Using ENUM ensures data integrity by restricting the allowed values to the specified list. It uses numeric indexes (starting from 1) for representing string values.

**CREATE TABLE** Student\_grade(  
 id INT **PRIMARY KEY** AUTO\_INCREMENT,  
 Grade VARCHAR(250) NOT NULL,  
 priority **ENUM**('Low', 'Medium', 'High') NOT NULL  
);

The prioritized column will accept only three columns. Here, the order of numbering **Low->1, Medium->2, High->3.**

**INSERT INTO** Student\_grade(Grade, priority)  
**VALUES**('Good grades', 'High');

Instead of using the enumeration values, you can also use the numerical indexes too, in order to insert the values into the **ENUM**column of the table

**INSERT INTO**  
 Student\_grade(Grade, priority),  
**VALUES**  
('Poor grades', 1);

**Retrieve Value of ENUM column**

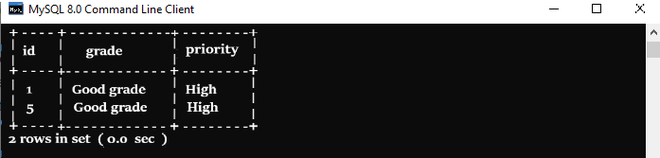
**Query:**

The following statement brought all the high grades student results

**SELECT** \* **FROM** Student\_grade  
**WHERE** priority = 'High';

The same result you can get through this My SQL query

**SELECT** \* **FROM** Student\_grade  
**WHERE** priority = 3;



In SQL, DATE\_SUB(CURDATE(), INTERVAL X UNIT) is used to subtract a specific time interval from the current date.

**Breaking it Down:**

* CURDATE(): Returns the current date (without time).
* DATE\_SUB(): Subtracts a given interval from a date.
* INTERVAL X UNIT: Specifies how much to subtract (e.g., days, months, years).

**Example Usages:**

1. **Subtract 18 years from the current date (for age validation)**

SELECT DATE\_SUB(CURDATE(), INTERVAL 18 YEAR);

**Output (if today is 2025-02-10):**

2007-02-10

This is useful for ensuring that a user's birthdate makes them at least 18 years old.

1. **Find all employees who joined more than 5 years ago**

SELECT \* FROM EMPLOYEE WHERE join\_date < DATE\_SUB(CURDATE(), INTERVAL 5 YEAR);

1. **Get the date 30 days before today**

SELECT DATE\_SUB(CURDATE(), INTERVAL 30 DAY);

1. **Get the date 3 months ago**

SELECT DATE\_SUB(CURDATE(), INTERVAL 3 MONTH);

**EX.NO:2**

EMPLOYEE (emp\_id, emp\_name, birth\_date, gender, dept\_no, address, designation, salary, experience, email)

DEPARTMENT (dept\_no, dept\_name, location)

Create the EMP Table with all necessary constraints such as In EMP TABLE: Employee id should be primary key, Department no should be Foreign key, employee age (birth\_date) should be greater than 18 years, salary should be greater than zero, email should have (@ and dot) sign in address, designation of employee can be “manager”, “clerk”, “leader”, “analyst”, “designer”, “coder”, “tester”.

**1. Create DEPARTMENT Table with Constraints**

CREATE TABLE DEPARTMENT (

dept\_no INT PRIMARY KEY,

dept\_name VARCHAR(50) UNIQUE,

location VARCHAR(100)

);

Here

dept\_no is the primary key.

dept\_name is unique.

**2. Create EMPLOYEE Table with Constraints**

CREATE TABLE EMPLOYEE (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(100) NOT NULL,

birth\_date DATE NOT NULL CHECK (birth\_date <= DATE\_SUB(CURDATE(), INTERVAL 18 YEAR)),

gender ENUM('Male', 'Female') NOT NULL,

dept\_no INT,

address VARCHAR(255),

designation ENUM('manager', 'clerk', 'leader', 'analyst', 'designer', 'coder', 'tester') NOT NULL,

salary DECIMAL(10,2) CHECK (salary > 0),

experience INT CHECK (experience >= 0),

email VARCHAR(100) NOT NULL CHECK (email LIKE '%@%.%'),

FOREIGN KEY (dept\_no) REFERENCES DEPARTMENT(dept\_no)

);

**Constraints Applied**:

* emp\_id is the **Primary Key**.
* dept\_no is a **Foreign Key** referencing DEPARTMENT(dept\_no).
* **Employee Age** must be **greater than 18 years**.
* **Salary must be greater than 0**.
* **Email must contain '@' and '.'**.
* **Designation is restricted** to specific roles.
* **Gender is restricted** to Male or Female.

**3. Insert Data into DEPARTMENT Table**

INSERT INTO DEPARTMENT VALUES

(1, 'IT', 'New York'),

(2, 'HR', 'Los Angeles'),

(3, 'Finance', 'Chicago'),

(4, 'Marketing', 'Houston'),

(5, 'Operations', 'Ahmedabad');

**4. Insert Data into EMPLOYEE Table**

INSERT INTO EMPLOYEE VALUES

(101, 'Alice', '1995-06-15', 'Female', 1, '123 Street, NY', 'manager', 7000, 5, 'alice@example.com'),

(102, 'Bob', '2000-03-10', 'Male', 2, '456 Road, LA', 'clerk', 3000, 1, 'bob@example.com'),

(103, 'Charlie', '1988-09-25', 'Male', 3, '789 Lane, CHI', 'analyst', 9000, 10, 'charlie@example.com'),

(104, 'Diana', '1997-12-20', 'Female', 1, '234 Ave, NY', 'coder', 4000, 2, 'diana@example.com'),

(105, 'Eva', '1996-07-08', 'Female', 4, '567 Blvd, HOU', 'tester', 6000, 4, 'eva@example.com');

**5. Display Structure of Tables**

DESC EMPLOYEE;

DESC DEPARTMENT;

**6. List All Records in Ascending Order**

SELECT \* FROM EMPLOYEE ORDER BY emp\_name ASC;

SELECT \* FROM DEPARTMENT ORDER BY dept\_name ASC;

**7. Delete Department Located in Ahmedabad**

DELETE FROM DEPARTMENT WHERE location = 'Ahmedabad';

**8. Display Female Employees**

SELECT \* FROM EMPLOYEE WHERE gender = 'Female';

**9. Display Employees Grouped by Department**

SELECT D.dept\_name, E.emp\_name

FROM EMPLOYEE E

JOIN DEPARTMENT D ON E.dept\_no = D.dept\_no

ORDER BY D.dept\_name;

**10. Employees with Salary Between 2000 and 5000**

SELECT emp\_name FROM EMPLOYEE WHERE salary BETWEEN 2000 AND 5000;

**11. Display Female Employees in Descending Order**

SELECT emp\_name, designation

FROM EMPLOYEE

WHERE gender = 'Female'

ORDER BY emp\_name DESC;

**12. Employees Whose Names Start and End with 'A'**

SELECT emp\_name FROM EMPLOYEE WHERE emp\_name LIKE 'A%A';

**13. Employees with Minimum Salary**

SELECT emp\_name, salary

FROM EMPLOYEE

WHERE salary = (SELECT MIN(salary) FROM EMPLOYEE);

**14. Increase Salary by 10% for IT Department Employees**

UPDATE EMPLOYEE

SET salary = salary \* 1.10

WHERE dept\_no = (SELECT dept\_no FROM DEPARTMENT WHERE dept\_name = 'IT');

**15. Count Employees in IT Department**

SELECT COUNT(\*) AS total\_employees

FROM EMPLOYEE

WHERE dept\_no = (SELECT dept\_no FROM DEPARTMENT WHERE dept\_name = 'IT');

**16. Employees Born in the Current Month**

SELECT \* FROM EMPLOYEE WHERE MONTH(birth\_date) = MONTH(CURDATE());

**17. Print Employee with Department Name**

SELECT CONCAT(emp\_name, ' works in department ', dept\_name) AS Employee\_Department

FROM EMPLOYEE

JOIN DEPARTMENT ON EMPLOYEE.dept\_no = DEPARTMENT.dept\_no;

**18. Employees with Less Than 1 Year Experience (Freshers)**

SELECT emp\_name FROM EMPLOYEE WHERE experience < 1;

**19. Employees with More Than 5 Years of Experience (Department Wise)**

SELECT D.dept\_name, E.emp\_name

FROM EMPLOYEE E

JOIN DEPARTMENT D ON E.dept\_no = D.dept\_no

WHERE E.experience > 5;

**20. Create a Sequence for Department ID (if using Oracle)**

CREATE SEQUENCE dept\_seq START WITH 6 INCREMENT BY 1;

**21. List Departments with No Employees**

SELECT dept\_name

FROM DEPARTMENT

WHERE dept\_no NOT IN (SELECT DISTINCT dept\_no FROM EMPLOYEE);

**Output Example**

**1. Employees Table (Sample)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **emp\_id** | **emp\_name** | **birth\_date** | **gender** | **dept\_no** | **address** | **designation** | **salary** | **experience** | **email** |
| 101 | Alice | 15-06-1995 | Female | 1 | 123 Street, NY | manager | 7000 | 5 | alice@example.com |
| 102 | Bob | 10-03-2000 | Male | 2 | 456 Road, LA | clerk | 3000 | 1 | bob@example.com |
| 103 | Charlie | 25-09-1988 | Male | 3 | 789 Lane, CHI | analyst | 9000 | 10 | charlie@example.com |

**2. Departments Table (Sample)**

| **dept\_no** | **dept\_name** | **location** |
| --- | --- | --- |
| 1 | IT | New York |
| 2 | HR | Los Angeles |

EX:NO:4

Create the database COMPANY and create given tables with all necessary constraints such as primary key, foreign key, unique key, not null and check constraints.

EMPLOYEE (emp\_id, emp\_name, birth\_date, gender, dept\_no, address, designation, salary, experience, email)

DEPART (dept\_no, dept\_name, total\_employees, location)

PROJECT (proj\_id, type\_of\_project, status, start\_date, emp\_id)

Insert proper data (at least 5 appropriate records) in all the tables.

1. Delete the department whose total number of employees less than 1.

2. Display the names and the designation of all female employee in descending order. 3. Display the names of all the employees who names starts with ‘A’ ends with ‘A’.

4. Find the name of employee and salary for those who had obtain minimum salary.

5. Add 10% raise in salary of all employees whose department is ‘CIVIL’.

6. Count total number of employees of ‘MCA’ department.

7. List all employees who born in the current month.

8. Print the record of employee and dept table as “Employee works in department ‘CE’. 9. List names of employees who are fresher’s(less than 1 year of experience).

10. List department wise names of employees who has more than 5 years of experience.

**Step 1: Create the COMPANY Database and Tables**

CREATE DATABASE COMPANY;

USE COMPANY;

1. **Create the DEPART table**

CREATE TABLE DEPART (

dept\_no INT PRIMARY KEY,

dept\_name VARCHAR(100) NOT NULL UNIQUE,

total\_employees INT CHECK (total\_employees >= 0),

location VARCHAR(100)

);

**2.Create the EMPLOYEE table**

CREATE TABLE EMPLOYEE (

emp\_id INT PRIMARY KEY,

emp\_name VARCHAR(100) NOT NULL,

birth\_date DATE NOT NULL,

gender VARCHAR(10) CHECK (gender IN ('Male', 'Female')),

dept\_no INT NOT NULL,

address VARCHAR(255),

designation VARCHAR(50),

salary DECIMAL(10, 2) CHECK (salary > 0),

experience DECIMAL(3, 1),

email VARCHAR(100) UNIQUE NOT NULL CHECK (email LIKE '%@%.%')

);

**3.Create the PROJECT table**

CREATE TABLE PROJECT (

proj\_id INT PRIMARY KEY,

type\_of\_project VARCHAR(100),

status VARCHAR(50),

start\_date DATE,

emp\_id INT,

FOREIGN KEY (emp\_id) REFERENCES EMPLOYEE(emp\_id)

);

**Step 2: Insert Sample Data into the Tables**

-- Insert data into DEPART table

INSERT INTO DEPART (dept\_no, dept\_name, total\_employees, location) VALUES

(1, 'CIVIL', 5, 'Delhi'),

(2, 'MCA', 3, 'Mumbai'),

(3, 'IT', 6, 'Bangalore'),

(4, 'MECH', 0, 'Chennai');

**Insert data into EMPLOYEE table**

INSERT INTO EMPLOYEE (emp\_id, emp\_name, birth\_date, gender, dept\_no, address, designation, salary, experience, email) VALUES

(101, 'Alice', '1995-03-15', 'Female', 1, 'Delhi', 'Engineer', 50000.00, 3, 'alice@company.com'),

(102, 'Bob', '1987-05-20', 'Male', 2, 'Mumbai', 'Manager', 75000.00, 10, 'bob@company.com'),

(103, 'Anna', '1998-08-10', 'Female', 2, 'Mumbai', 'Clerk', 30000.00, 1, 'anna@company.com'),

(104, 'David', '2000-12-01', 'Male', 3, 'Bangalore', 'Designer', 25000.00, 0.5, 'david@company.com'),

(105, 'Eve', '1992-07-30', 'Female', 1, 'Delhi', 'Analyst', 45000.00, 5, 'eve@company.com');

**Insert data into PROJECT table**

INSERT INTO PROJECT (proj\_id, type\_of\_project, status, start\_date, emp\_id) VALUES

(201, 'Construction', 'Ongoing', '2023-01-01', 101),

(202, 'Software', 'Completed', '2022-12-15', 104),

(203, 'Networking', 'Ongoing', '2023-05-01', 102);

**Step 3: Queries to Fulfill the Requirements**

1. **Delete the department whose total number of employees is less than 1**

DELETE FROM DEPART WHERE total\_employees < 1;

1. **Display the names and designations of all female employees in descending order**

SELECT emp\_name, designation

FROM EMPLOYEE

WHERE gender = 'Female'

ORDER BY emp\_name DESC;

1. **Display the names of all employees whose names start with 'A' and end with 'A'**

SELECT emp\_name

FROM EMPLOYEE

WHERE emp\_name LIKE 'A%A';

1. **Find the name of the employee and salary for those who obtained the minimum salary**

SELECT emp\_name, salary

FROM EMPLOYEE

WHERE salary = (SELECT MIN(salary) FROM EMPLOYEE);

Select emp\_name,min(salary) from employee;

1. **Add a 10% raise in the salary of all employees whose department is ‘CIVIL’**

UPDATE EMPLOYEE SET salary = salary \* 0.10 WHERE dept\_name = 'CIVIL';

UPDATE EMPLOYEE

SET salary = salary \* 0.10

WHERE dept\_no = (SELECT dept\_no FROM DEPART WHERE dept\_name = 'CIVIL');

1. **Count the total number of employees in the ‘MCA’ department**

SELECT COUNT(\*) AS Total\_Employees

FROM EMPLOYEE WHERE dept\_name = 'MCA';

WHERE dept\_no = (SELECT dept\_no FROM DEPART WHERE dept\_name = 'MCA');

1. **List all employees who were born in the current month**

SELECT \*

FROM EMPLOYEE

WHERE MONTH(birth\_date) = MONTH(CURDATE());

1. **Print the record of employees and their department as “Employee works in department ‘CE’”**

SELECT CONCAT(E.emp\_name, D.dept\_name) AS Employee\_Department

FROM EMPLOYEE E

JOIN DEPART D ON E.dept\_no = D.dept\_no;

1. **List the names of employees who are fresher’s (less than 1 year of experience)**

SELECT emp\_name

FROM EMPLOYEE

WHERE experience < 1;

1. **List department-wise names of employees who have more than 5 years of experience**

SELECT D.dept\_name, E.emp\_name

FROM DEPART D

JOIN EMPLOYEE E ON D.dept\_no = E.dept\_no

WHERE E.experience > 5

ORDER BY D.dept\_name;

**Explanation of Queries**

1. **Delete Department**: Deletes rows from DEPART where total\_employees is less than 1.
2. **Descending Order**: Filters female employees and orders them by name descending.
3. **Pattern Matching**: Finds employees whose names start and end with 'A' using LIKE.
4. **Minimum Salary**: Identifies the employee(s) with the lowest salary.
5. **Raise Salary**: Updates salaries with a 10% increase for employees in the CIVIL department.
6. **Count Employees**: Counts the employees associated with the MCA department.
7. **Current Month Birthdays**: Uses MONTH function to filter employees born this month.
8. **Employee-Department Statement**: Creates a formatted string combining employee and department names.
9. **Fresher Employees**: Filters employees with less than 1 year of experience.
10. **Department-Wise Experienced Employees**: Joins tables to group employees by department who have over 5 years of experience.