**FSD-II**

**Module 7: Introduction to My SQL**

**AIM: a.** Write a program to create a Database and table inside that database using My SQL Command line client

**Description:**

**In this program, you will use MySQL commands to:**

* **Create a new database.**
* **Switch to that database.**
* **Create a table with appropriate columns.**

**These steps are executed using the MySQL Command Line Client.**

**Procedure:**

1. **Open MySQL Command Line Client and enter your password.**
2. **Create a new database using the CREATE DATABASE command.**
3. **Select the database using the USE command.**
4. **Create a table using the CREATE TABLE command with appropriate columns and data types.**

**MySQL Code:**

**-- Step 1: Create a Database**

**CREATE DATABASE StudentDB;**

**-- Step 2: Use the Database**

**USE StudentDB;**

**-- Step 3: Create a Table inside StudentDB**

**CREATE TABLE Students (**

**student\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**name VARCHAR(100),**

**department VARCHAR(50),**

**age INT,**

**email VARCHAR(100)**

**);**

**Output:**

MySQL Command Execution and Output

🔹 Step 1: Create a Database

mysql> CREATE DATABASE StudentDB;

Output:

Query OK, 1 row affected

🔹 Step 2: Use the Database

mysql> USE StudentDB;

✅ Output:

Database changed

🔹 Step 3: Create a Table

mysql> CREATE TABLE Students (

-> student\_id INT AUTO\_INCREMENT PRIMARY KEY,

-> name VARCHAR(100),

-> department VARCHAR(50),

-> age INT,

-> email VARCHAR(100)

-> );

✅ Output:

Query OK, 0 rows affected

🔹 Step 4: Verify the Table

mysql> DESCRIBE Students;

✅ Output:

+------------+--------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+------------+--------------+------+-----+---------+----------------+

| student\_id | int | NO | PRI | NULL | auto\_increment |

| name | varchar(100) | YES | | NULL | |

| department | varchar(50) | YES | | NULL | |

| age | int | YES | | NULL | |

| email | varchar(100) | YES | | NULL | |

+------------+--------------+------+-----+---------+----------------+

5 rows in set (0.00 sec)

**AIM: b. Write a My SQL queries to create table, and insert the data, update the data in the table**

**Description:**

These are basic MySQL operations to:

* Create a table named Employees
* Insert some records into the table
* Update specific data in that table

**Procedure:**

Step 1: Open MySQL Command Line Client

* Login using your password.

Step 2: Create a Database (if not already created)

CREATE DATABASE CompanyDB;

USE CompanyDB;

Step 3: Create a Table

CREATE TABLE Employees (

emp\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100),

position VARCHAR(100),

salary DECIMAL(10, 2)

);

Step 4: Insert Data into the Table

INSERT INTO Employees (name, position, salary)

VALUES

('Alice Johnson', 'Manager', 60000.00),

('Bob Smith', 'Developer', 50000.00),

('Charlie Lee', 'Designer', 45000.00);

Step 5: View the Table Data

SELECT \* FROM Employees;

Step 6: Update Data in the Table

UPDATE Employees

SET salary = 55000.00

WHERE name = 'Bob Smith';

Step 7: View Updated Data

SELECT \* FROM Employees;

**Expected Output:**

After Creating the Table:

Query OK, 0 rows affected (0.02 sec)

After Inserting Data:

Query OK, 3 rows affected (0.01 sec)

SELECT \* FROM Employees;

+--------+---------------+-----------+----------+

| emp\_id | name | position | salary |

+--------+---------------+-----------+----------+

| 1 | Alice Johnson | Manager | 60000.00 |

| 2 | Bob Smith | Developer | 50000.00 |

| 3 | Charlie Lee | Designer | 45000.00 |

+--------+---------------+-----------+----------+

After Updating Bob Smith’s Salary:

Query OK, 1 row affected (0.01 sec)

Rows matched: 1 Changed: 1 Warnings: 0

SELECT \* FROM Employees; (After Update)

+--------+---------------+-----------+----------+

| emp\_id | name | position | salary |

+--------+---------------+-----------+----------+

| 1 | Alice Johnson | Manager | 60000.00 |

| 2 | Bob Smith | Developer | 55000.00 |

| 3 | Charlie Lee | Designer | 45000.00 |

+--------+---------------+-----------+----------+

**AIM : c. Write a My SQL queries to implement the subqueries in the My SQL command line client**

**Description:**

**A subquery is a query nested inside another query. It is used to:**

* **Filter results based on another query.**
* **Perform operations like comparing values, calculating aggregates, or selecting related data.**

**Types:**

* **Scalar subquery (returns a single value)**
* **Row subquery**
* **Table subquery**

**Procedure:**

1. **Open the MySQL command line client and login.**
2. **Select or create a database:**
3. **CREATE DATABASE SubqueryDemo;**
4. **USE SubqueryDemo;**
5. **Create tables and insert data for demonstration.**

**Sample Tables:**

**🔹 Table 1: Employees**

**CREATE TABLE Employees (**

**emp\_id INT PRIMARY KEY AUTO\_INCREMENT,**

**name VARCHAR(50),**

**department VARCHAR(50),**

**salary INT**

**);**

**INSERT INTO Employees (name, department, salary) VALUES**

**('Alice', 'HR', 40000),**

**('Bob', 'IT', 50000),**

**('Charlie', 'IT', 55000),**

**('Diana', 'HR', 60000),**

**('Evan', 'Finance', 45000);**

**Subquery Examples**

**1️⃣ Subquery to find employees with salary greater than the average salary:**

**SELECT name, salary**

**FROM Employees**

**WHERE salary > (SELECT AVG(salary) FROM Employees);**

**Explanation: The subquery calculates the average salary, and the main query selects those with a higher salary.**

**Subquery used in FROM clause (table subquery):**

**SELECT department, avg\_salary**

**FROM (**

**SELECT department, AVG(salary) AS avg\_salary**

**FROM Employees**

**GROUP BY department**

**) AS dept\_avg;**

**Explanation: This subquery calculates average salary per department and gives a result set as a temporary table.**

**Subquery in SELECT clause (scalar subquery):**

**SELECT name,**

**(SELECT MAX(salary) FROM Employees) AS highest\_salary**

**FROM Employees;**

**Explanation: Displays each employee’s name and the highest salary in the company.**

**Subquery using IN to get employees from departments having more than one employee**

**SELECT name, department:**

**FROM Employees**

**WHERE department IN (**

**SELECT department**

**FROM Employees**

**GROUP BY department**

**HAVING COUNT(\*) > 1**

**);**

**Explanation: Finds departments with more than one employee and lists the employees in those departments.**

**.Expected Output (Example):**

**Output for Query 1:**

**+--------+--------+**

**| name | salary |**

**+--------+--------+**

**| Charlie| 55000 |**

**| Diana | 60000 |**

**+--------+--------+**

**AIM: d. Write a My SQL program to create the script files in the My SQL workbench**

**Description:**

**A MySQL script file is a .sql file that contains SQL commands like CREATE, INSERT, UPDATE, etc. You can write and save SQL commands in this file and then run it in MySQL Workbench or the command line.**

**Procedure (Using MySQL Workbench):**

**Step 1: Open MySQL Workbench**

**Launch MySQL Workbench and connect to your MySQL Server.**

**Step 2: Create a New SQL Script**

* **Click on File > New Query Tab or Ctrl+T.**
* **An editor window opens where you can write SQL code.**

**Step 3: Write SQL Commands**

**Paste the following SQL code (or write your own):**

**-- Creating a database**

**CREATE DATABASE IF NOT EXISTS SchoolDB;**

**USE SchoolDB;**

**-- Creating a table**

**CREATE TABLE IF NOT EXISTS Students (**

**student\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**name VARCHAR(100),**

**grade INT,**

**age INT**

**);**

**-- Inserting data**

**INSERT INTO Students (name, grade, age) VALUES**

**('Alice', 10, 15),**

**('Bob', 9, 14),**

**('Charlie', 10, 16);**

**-- Updating data**

**UPDATE Students SET grade = 11 WHERE name = 'Alice';**

**Step 4: Save the Script File**

* **Go to File > Save Script As...**
* **Name your file (e.g., school\_script.sql)**
* **Save it on your computer.**

**Step 5: Execute the Script**

* **Click the lightning bolt (⚡) icon or press Ctrl + Shift + Enter to run all the commands.**
* **Check the Output tab to verify that the commands were executed successfully.**

**Output:**

**Once executed, the output messages at the bottom should look like:**

**1 statement executed successfully (0.01 sec)**

**3 rows inserted. (0.00 sec)**

**1 row updated. (0.00 sec)**

**You can confirm the data by running:**

**SELECT \* FROM Students;**

**Output:**

**+------------+---------+-------+-----+**

**| student\_id | name | grade | age |**

**+------------+---------+-------+-----+**

**| 1 | Alice | 11 | 15 |**

**| 2 | Bob | 9 | 14 |**

**| 3 | Charlie | 10 | 16 |**

**+------------+---------+-------+-----+**

**AIM: e. Write a My SQL program to create a database directory in Project and initialize a database. sql file to integrate the database into API**

**Here’s a complete MySQL + Node.js (API) integration setup, including how to:**

1. **Create a MySQL database and table**
2. **Save them in a .sql file**
3. **Create a project directory for API**
4. **Initialize database.sql in that project**
5. **Connect the database to your Express.js API**

**Description**

**This setup will:**

* **Create a MySQL database and table.**
* **Save SQL statements in a database.sql file.**
* **Use a Node.js + Express API.**
* **Load the SQL into MySQL and integrate it with your API.**

**Procedure (Step-by-Step):**

**Step 1: Create a Project Directory**

**Open terminal (CMD or Bash) and run:**

**mkdir student-api**

**cd student-api**

**Step 2: Create the database.sql File**

**Create a file named database.sql with the following content:**

**-- Create database and table**

**CREATE DATABASE IF NOT EXISTS StudentDB;**

**USE StudentDB;**

**CREATE TABLE IF NOT EXISTS Students (**

**id INT AUTO\_INCREMENT PRIMARY KEY,**

**name VARCHAR(100),**

**grade INT,**

**age INT**

**);**

**Save this file inside the student-api folder.**

**Step 3: Install Node.js Packages**

**Initialize Node.js and install packages:**

**npm init -y**

**npm install express mysql2**

**Step 4: Create db.js to Connect MySQL with Node.js**

**Create a file db.js in the same folder:**

**const mysql = require('mysql2');**

**const connection = mysql.createConnection({**

**host: 'localhost',**

**user: 'root', // use your MySQL username**

**password: '', // use your MySQL password**

**database: 'StudentDB' // database created in database.sql**

**});**

**connection.connect((err) => {**

**if (err) throw err;**

**console.log('Connected to MySQL database!');**

**});**

**module.exports = connection;**

**Step 5: Create Express API File (app.js)**

**const express = require('express');**

**const app = express();**

**const db = require('./db');**

**app.use(express.json());**

**app.get('/students', (req, res) => {**

**db.query('SELECT \* FROM Students', (err, results) => {**

**if (err) throw err;**

**res.json(results);**

**});**

**});**

**const PORT = 3000;**

**app.listen(PORT, () => {**

**console.log(`Server running on http://localhost:${PORT}`);**

**});**

**Step 6: Load the SQL File into MySQL**

**Use MySQL CLI to run the script:**

**mysql -u root -p < database.sql**

**This creates the database and the table.**

**Step 7: Start the Express Server**

**node app.js**

**Open your browser and go to:**

**http://localhost:3000/students**

**You’ll get an empty array [] (since no data inserted yet).**

**Output:**

**Once setup is done:**

* **Terminal Output:**

**Connected to MySQL database!**

**Server running on http://localhost:3000**

* **Browser Output (GET /students):**

**[]**

# Module 8: Team Collaboration Using GitHub

o Form groups of 2–3 students

o Create a shared GitHub repo

o Assign tasks and work in branches

o Use Issues, Pull Requests, and Code Reviews

o Document code with README.md

**“ It was perform by the students “**