Green University of Bangladesh
Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Fall, Year:2024), B.Sc. in CSE (Day)

Lab Report NO: 03

Course Title: Database System Lab

Course Code: CSE 210   Section:231(D1)

Lab Experiment Name: Modifying MySQL databases and Updating Data in MySQL Table, Implementation of Integrity Constraints in MySQL, Modifying MySQL databases and Updating Data in MySQL Table,Querying and Filtering data in MySQL Table.

Student Details

Name		ID
1.	Promod Chandra Das	231002005

Lab Date : 14-10-2024

Submission Date : 20-10-2024

Course Teacher's Name : Fatema-Tuj- Johora

Lab Report Status

Marks: ………………………………… **Signature:....................**

Comments:... **Date:.............................**

❖ **TITLE OF THE LAB REPORT EXPERIMENT**

Modifying MySQL databases and Updating
Data in MySQL Table, Implementation of Integrity Constraints in
MySQL, Modifying MySQL databases, and Updating
Data in MySQL Table,Querying and Filtering data in MySQL Table
Create a 5-column database by yourself.

❖ **OBJECTIVES**

1. **Create a 5-column database in MySQL:**
   - Understand how to define and structure a MySQL database.
   - Practice creating tables with multiple columns in MySQL.
2. **Modifying MySQL databases and Updating Data in MySQL Tables:**
   - Learn how to alter the structure of existing tables.
   - Perform updates on data within tables (e.g., modifying values).
3. **Implementing Integrity Constraints in MySQL:**
   - Learn how to apply integrity constraints (e.g., PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL) to ensure data integrity.
4. **Querying and Filtering Data in MySQL Table:**
   - Practice writing queries to retrieve specific data.
   - Filter data based on conditions using WHERE, LIKE, and other clauses.

Steps to Create a 5-Column MySQL Database and Implement the Objectives:

1. Create a Database and Table:

Start by creating a MySQL database and a table with 5 columns. Example:

sql

Copy code

```
CREATE DATABASE my_database;

USE my_database;

CREATE TABLE employees (
   id INT AUTO_INCREMENT PRIMARY KEY,
   first_name VARCHAR(50) NOT NULL,
   last_name VARCHAR(50) NOT NULL,
   email VARCHAR(100) UNIQUE,
   hire_date DATE NOT NULL
);
```

- **Columns:** id, first_name, last_name, email, hire_date.

2. Modifying the Database and Table Structure:

You may need to modify the table structure after creation. For example, adding a column:

sql

Copy code

```
ALTER TABLE employees ADD COLUMN phone_number VARCHAR(15);
```

Or changing an existing column:

sql

Copy code

```
ALTER TABLE employees MODIFY COLUMN email VARCHAR(150);
```

3. Updating Data in MySQL Table:

Updating data in the table can be done using the UPDATE statement. For example:

sql

Copy code

```
UPDATE employees
SET email = 'newemail@example.com'
WHERE id = 1;
```

4. Implement Integrity Constraints:

Add integrity constraints to ensure data consistency:

- **Primary Key:** Ensures each row has a unique identifier.
- **Unique Constraint:** Ensures the email field contains unique values.
- **Foreign Key:** Can be used to reference other tables (not shown in the current table).

Example:

sql

Copy code

```
ALTER TABLE employees
ADD CONSTRAINT unique_email UNIQUE (email);
```

5. Querying and Filtering Data:

Retrieve specific data using queries. For example, to filter employees hired after a certain date:

sql

Copy code

```
SELECT * FROM employees
WHERE hire_date > '2020-01-01';
```

Or search for employees by last name using LIKE:

sql

Copy code

```
SELECT * FROM employees
WHERE last_name LIKE 'S%';
```

Final Notes:

- The 5-column table can be modified and manipulated in various ways to meet different objectives.
- These queries demonstrate how to create, modify, update, and query data in a MySQL database efficiently.

❖ IMPLEMENTATION

Step 1: Create a 5-Column Database and Table

First, create a database and a table with 5 columns.

```
SQL


CREATE DATABASE company_db;

USE company_db;

CREATE TABLE employees (
   emp_id INT AUTO_INCREMENT PRIMARY KEY,  -- Unique employee ID
   first_name VARCHAR(50) NOT NULL,        -- Employee's first name
   last_name VARCHAR(50) NOT NULL,         -- Employee's last name
   email VARCHAR(100) UNIQUE,              -- Unique email address
   hire_date DATE NOT NULL                 -- Date the employee was hired
);
```

In this table:

- emp_id is the primary key and will auto-increment.
- email has a UNIQUE constraint to ensure no duplicate emails.
- NOT NULL ensures required fields.

Step 2: Modifying the Database Structure

If you need to modify the database (for example, adding new columns), use ALTER TABLE. Let's add a new column for phone_number.

```
Sql


ALTER TABLE employees
ADD COLUMN phone_number VARCHAR(15);
```

You can also modify existing columns. For instance, increasing the length of email:

```
SQL

ALTER TABLE employees
MODIFY COLUMN email VARCHAR(150);
```

Step 3: Updating Data in the Table
Once data is inserted into the table, you may need to update it. The UPDATE statement allows for modifying data. For example, updating the email of an employee with emp_id = 1:

```
SQL

UPDATE employees
SET email = 'updatedemail@example.com'
WHERE emp_id = 1;
```

You can also update multiple columns at once:

```
SQL

UPDATE employees
SET first_name = 'John', last_name = 'Doe'
WHERE emp_id = 2;
```

Step 4: Implementing Integrity Constraints
Integrity constraints are important to ensure data validity. Here are a few constraints you can add:

1. **Primary Key** (already defined as emp_id).
2. **Unique Constraint** on the email column ensures no duplicate emails.
3. **NOT NULL Constraint** ensures important fields like hire_date are always provided.
4. **Foreign Key** can be added if the table references another table, for example:

```
SQL

ALTER TABLE employees
ADD CONSTRAINT fk_department
FOREIGN KEY (dept_id)
REFERENCES departments(dept_id);
```

Step 5: Querying and Filtering Data
To query or filter specific data, use SELECT with conditions.

- To retrieve all employees hired after a specific date:

```
SQL

SELECT * FROM employees
WHERE hire_date > '2023-01-01';
```

- To filter based on a pattern (e.g., search for employees whose last name starts with "S"):

```
SQL

SELECT * FROM employees
WHERE last_name LIKE 'S%';
```

- To retrieve specific columns (e.g., first name and email):

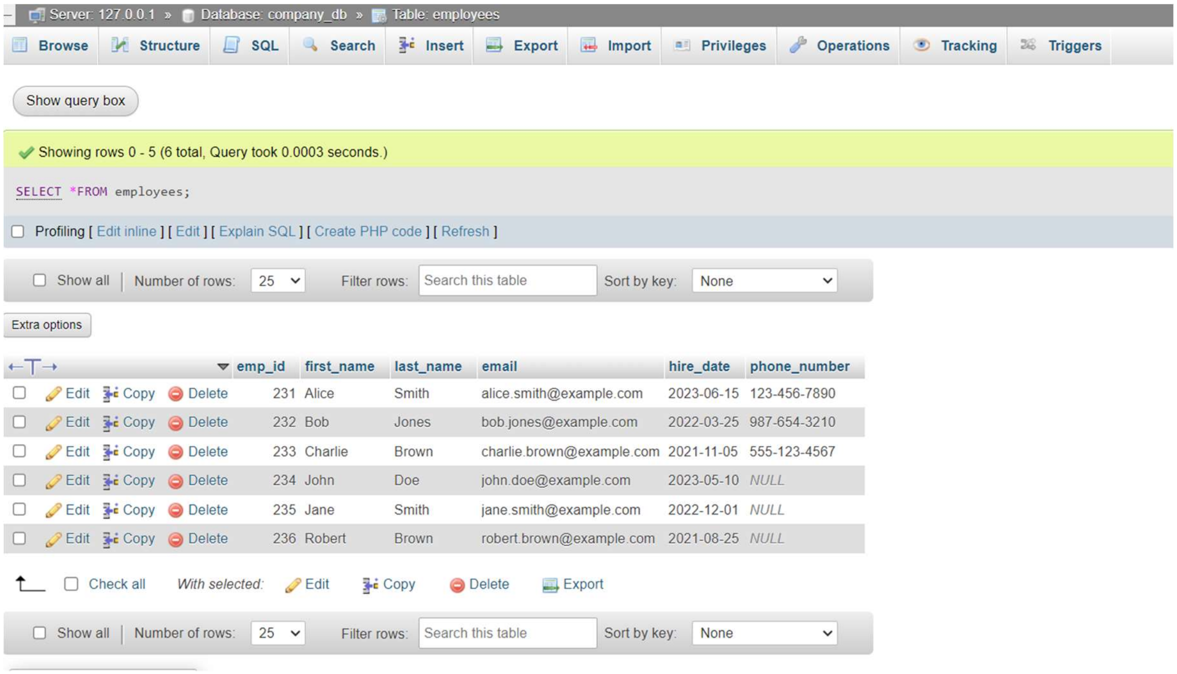
```
SQL


SELECT first_name, email FROM employees;
```

- To order the results by hire_date:

```
SQL

SELECT * FROM employees
ORDER BY hire_date DESC;
```



✓ Showing rows 0 - 5 (6 total, Query took 0.0003 seconds.)

```
SELECT *FROM employees;
```

emp_id	first_name	last_name	email	hire_date	phone_number
231	Alice	Smith	alice.smith@example.com	2023-06-15	123-456-7890
232	Bob	Jones	bob.jones@example.com	2022-03-25	987-654-3210
233	Charlie	Brown	charlie.brown@example.com	2021-11-05	555-123-4567
234	John	Doe	john.doe@example.com	2023-05-10	NULL
235	Jane	Smith	jane.smith@example.com	2022-12-01	NULL
236	Robert	Brown	robert.brown@example.com	2021-08-25	NULL

DISCUSSION:

In MySQL, modifying databases and updating tables involves altering the structure and data. You can update data using the UPDATE statement, modify columns with ALTER TABLE, and enforce integrity constraints like PRIMARY KEY, FOREIGN KEY, NOT NULL, and UNIQUE for consistency. Querying data can be done with SELECT, and filtering data is achieved using conditions like WHERE