SQL Guided Lab 1

Note: MySQL must be installed on your WSL or Linux system before starting this lab.

MySQL Basics

Step 1: Access MySQL from the Command Line

1. Open your terminal.
2. Start MySQL by Typing the following command and press Enter:

mysql -u root -p

Explanation:

- `-u root`: Specifies that you want to log in as the root user.
- `-p`: Tells MySQL to prompt you for your password.
- You will be prompted to enter your MySQL root password. Enter it and press Enter.

Step 2: Creating a Database(How To Create a new Database)

1. Create a new database:
   - Type the following command and press Enter:

CREATE DATABASE testdb;

Explanation:

- `CREATE DATABASE testdb;`: Instructs MySQL to create a new database named`testdb`.
- You should see a message indicating that the query was successful.

Step 3: Viewing Databases

1. How do you View existing databases:
   - Type the following command and press Enter:

SHOW DATABASES;

- Explanation:
- `SHOW DATABASES;`: Lists all databases on your MySQL server both existing ones and the one that you have just created.
- This command helps verify that your new database has been created.

Step 4: Switching Between Databases

1. Select a database to use:
   - Type the following command and press Enter:

USE testdb;

- Explanation:
- `USE testdb;`: Sets `testdb` as the active database for your session. Any commands done from now will apply to this database.

Understanding Data Types

Common Data Types:

- **INT:** Integer numbers (e.g., `123`, `456`)
- **VARCHAR(size):** Variable length string (e.g., `VARCHAR(50)` can store strings up to 50 characters)
- **TEXT:** Large text strings
- **DATE:** Date values (e.g., `2023-01-01`)
- **FLOAT:** Floating point numbers (e.g., `123.45`)

- Data types define the kind of data that can be stored in each column of a table. Choosing the correct data type ensures data integrity and optimizes storage.

Step 5: Creating a Table(Creating and Managing Tables)

1. Create a new table:
   - Type the following command and press Enter:

```
CREATE TABLE employees (
id INT AUTO_INCREMENT PRIMARY KEY,
first_name VARCHAR(50),
last_name VARCHAR(50),
hire_date DATE,
salary FLOAT
);
```

- Explanation:
- **`CREATE TABLE employees`:** Instructs MySQL to create a new table named `employees`.
- **`id INT AUTO_INCREMENT PRIMARY KEY`**: Creates a column named `id` that stores integers.
`AUTO_INCREMENT` automatically assigns a unique value to each new row.
`PRIMARY KEY` ensures each `id` is unique.
- **`first_name VARCHAR(50)`:** Creates a column for first names with a maximum length of 50 characters.
- **`last_name VARCHAR(50)`:** Creates a column for last names with a maximum length of 50 characters.
- **`hire_date DATE`:** Creates a column to store date values.
- **`salary FLOAT`:** Creates a column to store floating point numbers.

Step 6: Viewing Tables in a Database

1. View all tables:

- Type the following command and press Enter:

SHOW TABLES;

- Explanation:

- `SHOW TABLES;`: Lists all tables in the active database.

Step 7: Describing a Table

1. View table structure:

- Type the following command and press Enter:

DESCRIBE employees;

- Explanation:

- `DESCRIBE employees;`: Shows details about each column in the `employees` table, including the column name, data type, and any constraints.

Step 8: Deleting a Table

1. Delete a table:

- Type the following command and press Enter:

DROP TABLE employees;

- Explanation:

- `DROP TABLE employees;`: Permanently deletes the `employees` table and all its data.

Step 9: Inserting Data(How to Insert Data into Tables)

1. Insert a row into the table:

- Type the following command and press Enter:

```
sql
INSERT INTO employees (first_name, last_name, hire_date, salary)
VALUES ('John', 'Doe', '2023-01-15', 50000);
```

- Explanation:

- `INSERT INTO employees (first_name, last_name, hire_date, salary)`: Specifies the table and columns to insert data into.

- `VALUES ('John', 'Doe', '2023-01-15', 50000)`: Specifies the values to insert into the columns, in the same order.

Step 10: Querying Data From Tables in the Database

1. Select all columns from a table:
   - Type the following command and press Enter:

SELECT * FROM employees;

- Explanation:
- `SELECT *`: Selects all columns.
- `FROM employees`: Specifies the table to select data from. `FROM` is used to specify the source table.

2. Select specific columns:
   - Type the following command and press Enter:

SELECT first_name, last_name FROM employees;

- Explanation:
- `SELECT first_name, last_name`: Selects only the `first_name` and `last_name` columns.

3. Filter records with a condition:
   - Type the following command and press Enter:

SELECT * FROM employees WHERE salary > 40000;

- Explanation:
- `WHERE salary > 40000`: Filters the results to include only rows where the `salary` is greater than `40000`. **`WHERE` is used to specify conditions.**

4. Sort records:
   - Type the following command and press Enter:

SELECT * FROM employees ORDER BY hire_date DESC;

- Explanation:
- `ORDER BY hire_date DESC`: Sorts the results by `hire_date` in descending order. **`ORDER BY` is used to sort the result set.**

Step 11: Updating and Deleting Data

1. Update records:
   - Type the following command and press Enter:

UPDATE employees SET salary = 55000 WHERE first_name = 'John' AND last_name = 'Doe';

- Explanation:
- `UPDATE employees`: Specifies the table to update.
- `SET salary = 55000`: Sets the `salary` to `55000`.
- `WHERE first_name = 'John' AND last_name = 'Doe'`: Filters the rows to update.

Step 12: Deleting Data

1. Delete records:
   - Type the following command and press Enter:

DELETE FROM employees WHERE first_name = 'John' AND last_name = 'Doe';

- Explanation:
- `DELETE FROM employees`: Specifies the table to delete from.
- `WHERE first_name = 'John' AND last_name = 'Doe'`: Filters the rows to delete.

Now, it's time to test your understanding with a challenge.
Challenge Steps:

1. Create a new database: `companydb
2. Switch to the new database:
3. Create a table: `projects` with columns `id`, `project_name`, `start_date`, `end_date`, `budget`
4. Insert sample data into the `projects` table you have just created.
5. Write SELECT queries for the following :
   - Retrieve all projects:
   - Retrieve only the `project_name` and `budget` for all projects:
   - Retrieve projects with a budget greater than 60000:
   - Sort projects by `start_date` in ascending order:
6. Update the budget for `Project Alpha` to 120000:
7. Delete `Project Gamma`:

To save the queries for each step of the challenge, you can use a text editor to create a script file, often with a `.sql` extension.:

Step 1: Open a Text Editor
Open your preferred text editor. This could be Visual Studio Code, Sublime Text, Notepad++, or any other text editor available on your system.

Step 2: Create a New File
Create a new file and save it with a `.sql` extension, for example, `SQL_Lab1_queries.sql`.

Step 3: Write the Queries
In the newly created `.sql` file, write the SQL queries for each step of the challenge. Make sure to add comments to separate each step and explain what each query does. Comments in SQL are written with `--` for single-line comments or `/* ... */` for multi-line comments.

Step 4: Save the File
After writing all the queries, save the file. Make sure it is saved with the `.sql` extension.

Step 5: Executing the Script (Optional)
1. Open the terminal.

2. Log in to MySQL with the command : ***mysql -u root -p***
(If you created another user , replace root with the appropriate username)

3. Execute the script with the command : ***SOURCE /path/to/SQL_Lab1_queries.sql;***

Replace `***SOURCE /path/to/SQL_Lab1_queries.sql;***` with the actual path to your `.sql` file.

Step 6: Submitting the File
Create a repo with the naming convention ***firstname-MySQL_Training.git*** replace firstname accordingly.
Push the file ***SQL_Lab1_queries.sql*** to your repo.