GETTING STARTED WITH MYSQL AND SQL

FULL STACK SKILLS BOOTCAMP

ONLINE KNOWLEDGE BASE PLATFORM

- **Lesson Overview:**
- In this lesson, we will be introduced to:

1. Different Types of Database
2. SQL and CRUD operations
3. Table joins and keys
4. Aggregate Queries

TYPES OF DATABASES

- **Different Types of Databases:**

1. **File-Based Databases**
    1. Data is stored in files on disk (e.g., CSV, JSON, XML).
2. **Document-Based Databases**
    1. Data is stored as documents, suitable for flexible schemas.
    2. Examples: MongoDB, CouchDB.
3. **Relational Databases**
    1. Data is organized into tables with rows and columns, using SQL for queries.
    2. Examples: MySQL, PostgreSQL, SQLite.

ADVANTAGES OF RELATIONAL DATABASES

- **Why Choose Relational Databases?**

1. **Structured Data**: Consistent schema for organized storage.
2. **Powerful Querying**: SQL supports complex queries and data manipulation.
3. **Data Integrity**: Enforces constraints and maintains relationships.
4. **Scalability**: Handles large datasets efficiently.

- **When to Use Document Databases:**

- For flexible or hierarchical data structures (e.g., content management systems, catalogs).

WHAT IS MYSQL?

- **Introduction to MySQL**
- A popular open-source relational database management system (RDBMS).
- Uses SQL (Structured Query Language) to manage data.
- **Key Features**:
  - High performance and reliability.
  - Suitable for small to large-scale databases.
  - Widely supported by development tools and programming languages.

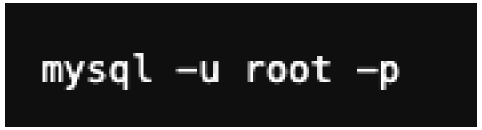
WHAT IS SQL?

- **Introduction to SQL**
- SQL (Structured Query Language) is a standardized language for managing relational databases.
- **Key Capabilities**:
    - Create, modify, and delete database structures (DDL).
    - Retrieve and manipulate data effectively (DML).
    - Perform aggregation and data analysis.

LOGGING IN TO MYSQL

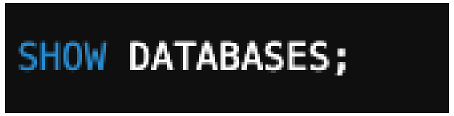
- **Steps to Log In:**

1. Open the terminal or MySQL client.



```
mysql -u root -p
```

- Enter your MySQL root password.
- Verify your connection by running:

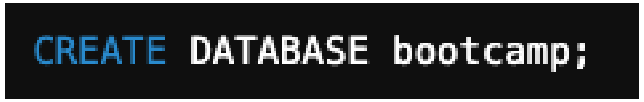


```
SHOW DATABASES;
```

CREATING A DATABASE

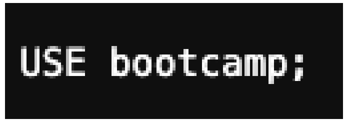
- **Steps to Create a Database:**

1. Create a database:



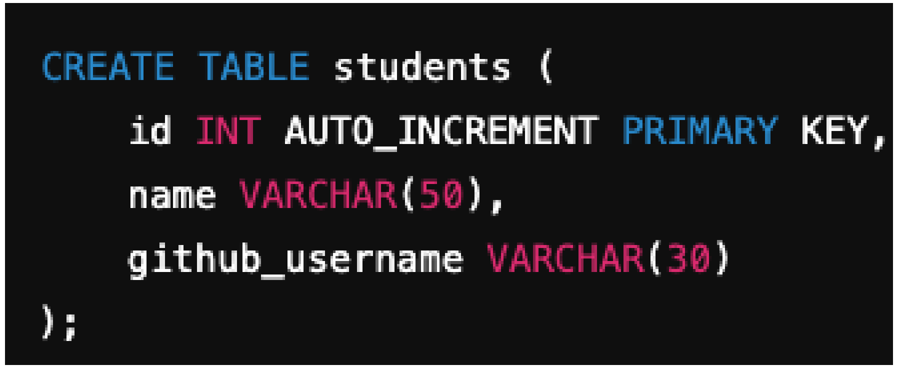
```
CREATE DATABASE bootcamp;
```

2. Switch to the database:



```
USE bootcamp;
```

3. Create a table:

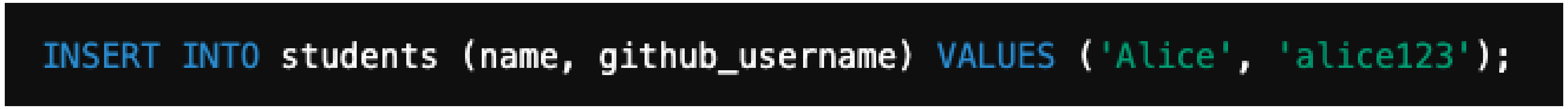


```
CREATE TABLE students (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50),
    github_username VARCHAR(30)
);
```

CRUD QUERIES

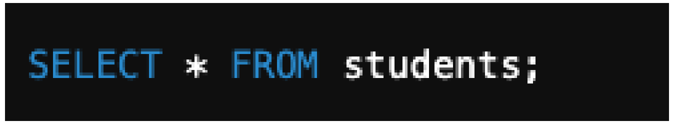
- **CRUD Operations Overview:**

1. **Create**: Insert new records



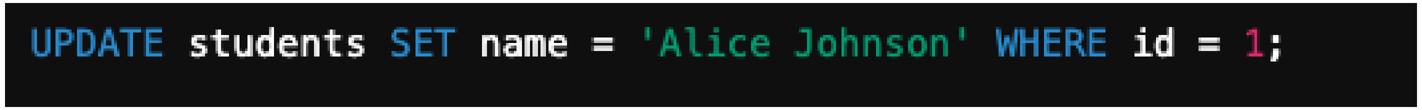
```
INSERT INTO students (name, github_username) VALUES ('Alice', 'alice123');
```

2. **Read**: Retrieve data



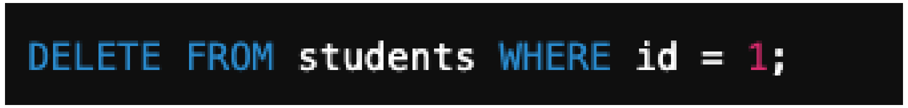
```
SELECT * FROM students;
```

3. **Update**: Modify existing data



```
UPDATE students SET name = 'Alice Johnson' WHERE id = 1;
```

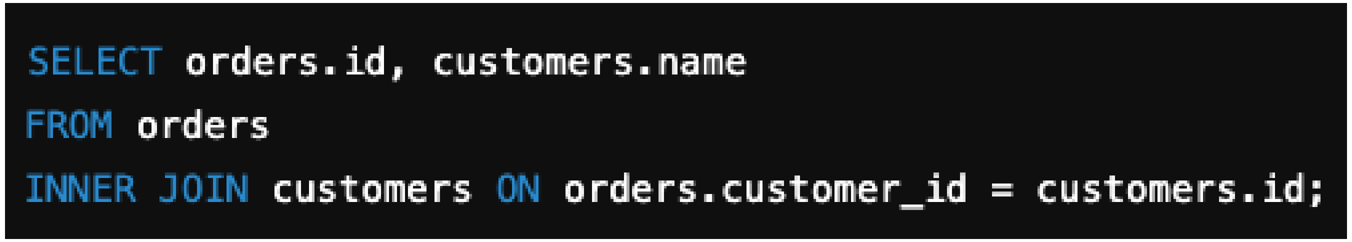
4. **Delete**: Remove records.



```
DELETE FROM students WHERE id = 1;
```

TABLE JOINS EXPLAINED

- **Understanding Table Joins:**



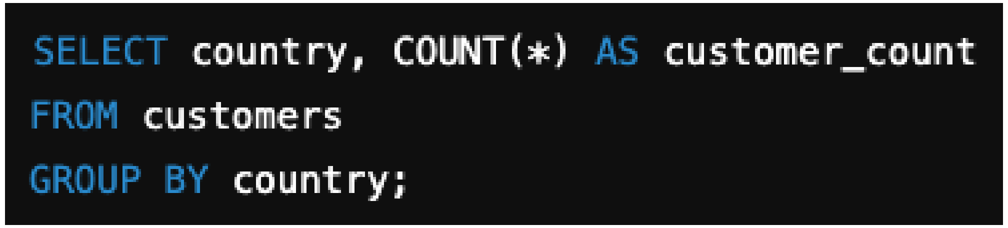
```
SELECT orders.id, customers.name
FROM orders
INNER JOIN customers ON orders.customer_id = customers.id;
```

1. **INNER JOIN**: Matches rows in both tables.
2. **LEFT JOIN**: Includes unmatched rows from the left table.
3. **RIGHT JOIN**: Includes unmatched rows from the right table.

GROUPING AND AGGREGATE FUNCTIONS

- **Using Grouping and Aggregations:**

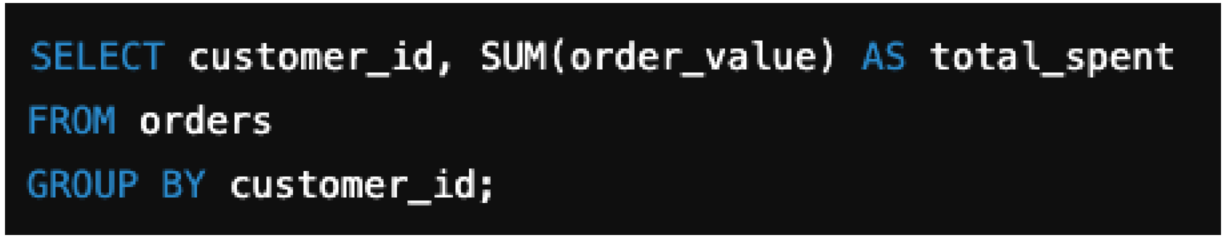
1. **Group By**: Organizes data into groups.



```
SELECT country, COUNT(*) AS customer_count
FROM customers
GROUP BY country;
```

AGGREGATE FUNCTIONS

- SUM(), AVG(), COUNT(), MAX(), MIN().Example



```
SELECT customer_id, SUM(order_value) AS total_spent
FROM orders
GROUP BY customer_id;
```

QUESTIONS?