Introduction to SQL and Database

What is SQL?

SQL (Structured Query Language) is a standardized programming language used for managing and manipulating relational databases. It allows users to store, retrieve, update, and delete data efficiently.

What is a Database?

A database is a structured collection of data that allows easy access, management, and updating. It organizes data in tables, making it easier to retrieve and manipulate.

Examples of Databases:

- E-commerce Database: Stores customer details, orders, and product inventory.
- Banking Database: Manages accounts, transactions, and customer details.
- Hospital Management Database: Keeps track of patient records, doctor schedules, and medical history.
- Social Media Database: Contains user profiles, posts, comments, and likes.

Popular Database Management Systems (DBMS)

There are several relational database management systems (RDBMS) available, including:

- MySQL: An open-source database widely used in web applications and enterprise solutions.
- MariaDB: A fork of MySQL, offering additional features and improved performance.
- IBM Db2: A powerful RDBMS used in enterprise environments, known for its high performance.
- Oracle Database: A robust, enterprise-grade RDBMS used in large-scale applications.
- Microsoft SQL Server: A proprietary database system developed by Microsoft, mainly used in enterprise applications.
- PostgreSQL: An open-source database known for its extensibility and compliance with SQL standards.

Why Use SQL and MySQL?

- Efficiently manage large datasets
- Perform complex queries and transactions
- Ensure data integrity with constraints like Primary and Foreign Keys
- Support multiple users and concurrent transactions
- Work seamlessly with programming languages like PHP, Python, and Java
- Essential for cloud-based applications, including AWS RDS (Relational Database Service)

SQL Data Types

When designing a database, choosing the correct data type is essential for efficiency and accuracy. Below are common data types used in MySQL:

Numeric Data Types:

- INT: Stores whole numbers (-2,147,483,648 to 2,147,483,647)
- BIGINT: Stores larger whole numbers
- FLOAT: Stores decimal numbers with precision
- DOUBLE: Stores larger decimal numbers
- DECIMAL: Stores exact decimal values (useful for financial data)

String Data Types:

- CHAR(n): Fixed-length string of n characters
- VARCHAR(n): Variable-length string of up to n characters
- TEXT: Large text data

Date & Time Data Types:

- DATE: Stores date (YYYY-MM-DD)
- TIME: Stores time (HH:MM:SS)
- DATETIME: Stores date and time
- TIMESTAMP: Stores timestamp values

Essential MySQL Commands with Explanations

1. Logging into MySQL

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

This command logs in as the root user and prompts for a password.

2. Setting a Password for MySQL

```
ALTER USER 'root'@'localhost' IDENTIFIED BY 'new_password';
```

This updates the password for the root user.

3. Exiting MySQL

```
EXIT;
```

Closes the MySQL shell.

4. Logging in with Authentication

```
mysql -u username -p
```

Logs in as the specified user and prompts for a password.

5. Creating a Database

```
CREATE DATABASE mydatabase;
```

Creates a new database named `mydatabase`.

6. Viewing Available Databases

```
SHOW DATABASES;
```

Lists all existing databases.

7. Selecting a Database

```
USE mydatabase;
```

Switches to the `mydatabase` database.

8. Creating a Table

```
CREATE TABLE users (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    email VARCHAR(100) UNIQUE,
    age INT
);
```

Creates a `users` table with an auto-incrementing `id`, `name`, `email`, and `age` columns.

9. Inserting Values into a Table

```
INSERT INTO users (name, email, age) VALUES ('shreyash',
'shreyash@gmail.com', 30);
```

Inserts a new record into the `users` table.

10. Retrieving Data from a Table

```
SELECT * FROM users;
```

Retrieves all records from the `users` table.

11. Describing a Table Structure

```
DESC users;
```

This command displays the structure of the users table. Example output:

Field	Type	Null	Key	Default	Extra
id	INT	NO	PRI	NULL	auto_increment
name	VARCHAR(100)	YES		NULL	
email	VARCHAR(100)	YES	UNI	NULL	
age	INT	YES		NULL	

Understanding the Columns:

- Field – Column name.
- Type – Data type (e.g., INT, VARCHAR(100), etc.).
- Null – YES means the column can have NULL values, NO means it must have a value.
- Key – Shows constraints (PRI = Primary Key, UNI = Unique Key).
- Default – The default value if none is provided.
- Extra – Additional properties (auto_increment means values increase automatically).

Displays the structure of the `users` table.

12. Finding Maximum and Minimum Values

```
SELECT MAX(age) FROM users;
SELECT MIN(age) FROM users;
```

Finds the highest and lowest values in the `age` column.

13. Updating Data

```
UPDATE users SET age = 31 WHERE name = 'shreyash';
```

Updates `age` to 31 for the user with the name 'shreyash'.

14. Altering a Table (Adding a Column)

```
ALTER TABLE users ADD COLUMN phone VARCHAR(15);
```

Adds a `phone` column to the `users` table.

15. Modifying Column Data Type

```
ALTER TABLE users MODIFY COLUMN phone BIGINT;
```

Changes the data type of the `phone` column.

16. Adding a Primary Key

```
ALTER TABLE users ADD PRIMARY KEY (id);
```

Ensures that `id` is a primary key.

17. Adding a Foreign Key

```
CREATE TABLE orders (
    order_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT,
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

Creates an `orders` table with `user_id` as a foreign key referencing `users`.

18. Creating a Database Backup

```
mysqldump -u root -p mydatabase > mydatabase_backup.sql
```

Creates a backup of `mydatabase`.

19. Restoring a Database from Backup

```
mysql -u root -p mydatabase < mydatabase_backup.sql
```

Restores `mydatabase` from a backup file.

20. Installing PHP-MySQL Extension

```
sudo apt install php8.3-mysqlnd -y
```

Installs the PHP-MySQL extension to enable MySQL support in PHP 8.3.

21. Connecting PHP with MySQL

```
<?php
$servername = "localhost";
$username = "root";
$password = "password";
$dbname = "mydatabase";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

This PHP script establishes a connection to MySQL.

Conclusion

SQL and MySQL are powerful tools for managing databases efficiently. Whether you're a beginner or an experienced developer, mastering these commands will help you build and maintain robust database-driven applications. Additionally, learning how MySQL integrates with AWS will be essential for cloud engineers.