MySQL on Amazon Linux: A Quick Start Guide

This document outlines essential steps and commands for installing, configuring, and basic usage of MySQL (MariaDB) on Amazon Linux.

1. Installation

MariaDB is the default relational database in Amazon Linux and is largely compatible with MySQL.

- **Download and Install MariaDB Server:**

```
sudo yum install mariadb105-server -y
```

This command uses `yum` to install `mariadb105-server` (version 10.5) and the `-y` flag automatically confirms the installation.

2. Service Management

- **Start the MariaDB Service:**

```
sudo service mariadb start
```

This command initiates the MariaDB database service.

3. Initial MySQL Access and Setup

- **Access MySQL (as root without password initially):**

```
sudo mysql
```

This command logs you into the MySQL shell as the `root` user, which initially has no password.

- **Set Password for the `root@localhost` User:**

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

Replace `'your_strong_password'` with your desired strong password. It's crucial to set a password for security reasons.

- **Exit MySQL Shell:**

```
EXIT;
```

This command exits the MySQL command-line interface.

- **Verify Password Setup (Attempting to log in without a password should now fail):**

```
sudo mysql
```

If you get an "Access denied" error, the password has been successfully set.

- **Log in to MySQL with Password (as root):**

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

After running this command, you will be prompted to enter the password you just set.

4. Basic Database Operations

- **Create a New Database:**

```
CREATE DATABASE your_database_name;
```

Replace `your_database_name` with the desired name for your database (e.g., `test`).

- **Verify Database Creation:**

```
SHOW DATABASES;
```

This command lists all databases on the server.

- **Select a Database to Work With:**

```
USE your_database_name;
```

Before creating tables or inserting data, you must select the database you want to work in.

To interrupt or cancel a long-running query or if you're stuck:*Press `Ctrl + C`*

5. Table Operations

- **Create a Table:**

```
CREATE TABLE table_name (
column1_name DATATYPE [CONSTRAINTS],
column2_name DATATYPE [CONSTRAINTS],
-- ... more columns
);
```

Example:

```
CREATE TABLE testuser(
id INT,
name VARCHAR(50),
phone VARCHAR(20), -- Added a length for VARCHAR
age INT,
city VARCHAR(50)    -- Added a length for VARCHAR
);
```

Describe Table Structure:

```
DESCRIBE table_name;
```

or

```
DESC table_name;
```

These commands display the structure (columns, data types, constraints) of a specified table.

- **Alter Table (Modify Column Type/Size):**

```
ALTER TABLE table_name MODIFY column_name NEW_DATATYPE;
```

Example:

```
ALTER TABLE payments MODIFY city VARCHAR(20);
```

- Insert Data into a Table:

```
INSERT INTO table_name (column1, column2, ...)
VALUES (value1, value2, ...);
```

- Select Data from a Table:

```
SELECT * FROM table_name;
```

Useful SQL Examples

Here are some practical SQL examples covering common database operations.

Example 1: Customer Payments

-
  ```
  - Create Database
  CREATE DATABASE customer;
  ```

-
  ```
  - Use the Database
  USE customer;
  ```

-
  ```
  - Create Table 'payments'
  CREATE TABLE payments
  (
  cust_id INT PRIMARY KEY,
  cust_name VARCHAR(50),
  pay_mode VARCHAR(20),
  city VARCHAR(15)
  );
  ```

-
  ```
  - Modify 'city' column to VARCHAR(20)
  ALTER TABLE payments MODIFY city VARCHAR(20);
  ```

-
  ```
  - Insert Data into 'payments'
  INSERT INTO payments
  (cust_id, cust_name, pay_mode, city)
  VALUES
  (101, "Olivia Barrett", "Netbanking", "Portland"),
  (102, "Ethen Sinclair", "Credit Card", "Miami"),
  (103, "Maya Hernandez", "Credit Card", "Seattle"),
  (104, "Liam Donovan", "Netbanking", "Denver"),
  (105, "Sophia Nguyen", "Credit Card", "NewOrleans"),
  (106, "Caleb Foster", "Debit Card", "Minneapolis"),
  (107, "Ava Patel", "Debit Card", "Phoenix"),
  (108, "Lucas Carter", "Netbanking", "Boston"),
  (109, "Isabella Martinez", "Netbanking", "Nashville"),
  (110, "Jackson Brooks", "Credit Card", "Boston");
  ```

-
  ```
  - Aggregate Query: Count customers by payment mode
  SELECT pay_mode, COUNT(cust_name)
  FROM payments
  GROUP BY pay_mode;
  ```

Example 2: College Students

-
  ```
  - Create Database
  ```

  ```
  CREATE DATABASE collagestudents;
  ```

-
  ```
  - Use the Database
  USE collagestudents;
  ```

- `- Create Table 'students'`
  ```
  CREATE TABLE students
  (
  rollno INT PRIMARY KEY,
  name VARCHAR(50),
  marks INT NOT NULL,
  grade VARCHAR(1),
  city VARCHAR(20)
  );
  ```
- `- Insert Data into 'students'`
  ```
  INSERT INTO students
  (rollno, name, marks, grade, city)
  VALUES
  (101, "tejas", 78, "C", "Pune"),
  (102, "samarth", 93, "A", "Mumbai"),
  (103, "atharva", 85, "B", "Mumbai"),
  (104, "om", 96, "F", "Delhi"),
  (105, "don", 12, "F", "Delhi"),
  (106, "tom", 82, "B", "Pune");
  ```
- `- Aggregate Query: Average marks by city, ordered by average marks descending`
  ```
  SELECT city, AVG(marks)
  FROM students
  GROUP BY city
  ORDER BY AVG(marks) DESC;
  ```

Example 3: Simple Student Table (Student)

- `- Create Table 'Sstudent'`
  ```
  CREATE TABLE Sstudent(
  id INT PRIMARY KEY,
  name VARCHAR(50),
  age INT NOT NULL
  );
  ```
- `- Insert Data into 'Sstudent'`
  ```
  INSERT INTO Sstudent
  (id, name, age)
  VALUES
  (5, "rony", 35),
  (6, "jon", 24),
  (7, "rohan", 40);
  ```
- `- Select all data from 'Sstudent'`
  ```
  SELECT * FROM Sstudent;
  ```