# STEP BY STEP PROCESS TO GET STARTED WITH JDBC

# 1. MySQL Connector/J:

* Download the MySQL Connector/J JAR file from the official MySQL website, the link was given in classroom.

# 2. MySQL Server and MySQL Workbench should be installed:

* The link was given in classroom.

# 3. Create a Database and User in MySQL Workbench:

- Launch MySQL Workbench and connect to your MySQL Server.

- Create a new database and necessary tables within that database using SQL queries.

- Create a new user with appropriate privileges on the database.

# 4. Configure MySQL Server Connection:

- Open MySQL Workbench.

- Click on the "+" icon next to "MySQL Connections" to create a new connection.

- Enter the connection details such as Connection Name, Hostname (usually "localhost"), Port (usually "3306"), Username, and Password.

- Click "Test Connection" to ensure the connection is successful.

# 5. Write Java Code using JDBC:

## import java.sql.Connection;

## import java.sql.DriverManager;

## import java.sql.SQLException;

## public class JdbcConnection {

## public static void main(String[] args) {

## String jdbcUrl = "jdbc:mysql://localhost:3306/<database\_name>?useSSL=false";

## String username = "<username>";

## String password = "<password>";

## try (Connection connection = DriverManager.getConnection(jdbcUrl, username, password)) {

## if (connection != null) {

## System.out.println("Connected to the database!");

## // Perform database operations here

## }

## } catch (SQLException e) {

## System.err.println("Connection error: " + e.getMessage());

## }

## }

## }

Replace **`<database\_name>`, `<username>`,** and **`<password>`** with your actual database name, username, and password respectively.

# 6. Add MySQL Connector/J JAR to Your Project:

- Add the downloaded MySQL Connector/J JAR file to your Java project's build path.

# 7. Run Your Java Application:

# Running Queries

# Establish a Database Connection:

Done previously.

# 2. Create a Statement:

After establishing the connection, you need to create a `**Statement**` object.

import java.sql.Statement;

import java.sql.ResultSet;

// Inside the try block after connection is established

**Statement statement = connection.createStatement();**

# 3. Execute a Query:

You can execute SQL queries using the `**executeQuery**` method of the `Statement` class. This method is used for SELECT queries that return a `**ResultSet**`.

**String selectQuery = "SELECT \* FROM your\_table\_name";**

**ResultSet resultSet = statement.executeQuery(selectQuery);**

## For non-SELECT queries (INSERT, UPDATE, DELETE), you can use the `executeUpdate` method:

**String updateQuery = "UPDATE your\_table\_name SET column\_name = 'new\_value' WHERE condition";**

**int rowsAffected = statement.executeUpdate(updateQuery);**

# 4. Process the Results:

## For SELECT Queries:

while (resultSet.next()) {

// Retrieve data from the result set

int id = resultSet.getInt("id");

String name = resultSet.getString("name");

// Process retrieved data

System.out.println("ID: " + id + ", Name: " + name);

}

For Non-SELECT Queries:

The `**executeUpdate**` method returns the number of rows affected:

**System.out.println("Rows affected: " + rowsAffected);**

# 5. Close Resources:

Always close the objects to release database resources:

resultSet.close();

statement.close();

connection.close();