**Asutosh Dash-Daily Assignment**

**Sprint 1 Day 5**

1. Connect to a Database

Write a Java program to connect to a MySQL (or any other) database using JDBC and print a success or failure message.

Solution:

package com.nisum;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
import java.sql.Statement;  
  
public class Example {  
 public static void main(String[] args) throws SQLException {  
 String url="jdbc:mysql://localhost:3306/jdbc";  
 String username="root";  
 String password="root";  
 Connection connection = DriverManager.*getConnection*(url,username,password);  
 Statement statement=connection.createStatement();  
 statement.executeUpdate("INSERT INTO PRODUCT VALUES (NULL ,'CHOCO',1000)");  
 System.*out*.println(connection);  
 if(connection){  
 System.*out*.println("Success");  
 }else{  
 System.*out*.println("Failure");  
 }  
 }  
}

1. Insert Data Using Statement

Create a table `students(id INT, name VARCHAR, age INT)`. Write a program to insert a new student record using Statement.

Solution:

package com.nisum;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
import java.sql.Statement;  
  
public class Example {  
 public static void main(String[] args) throws SQLException {  
 String url="jdbc:mysql://localhost:3306/jdbc";  
 String username="root";  
 String password="root";  
 Connection connection = DriverManager.*getConnection*(url,username,password);  
 Statement statement=connection.createStatement();  
 statement.executeUpdate("INSERT INTO PRODUCT VALUES (NULL ,'Asutosh',22)");  
 System.*out*.println(connection);  
// if(connection){  
// System.out.println("Success");  
// }else{  
// System.out.println("Failure");  
// }  
 }  
}

1. Retrieve and Display Records

Write a program to retrieve all records from the `students` table and display them using ResultSet

Solution:

package com.nisum;  
  
import java.sql.\*;  
import java.util.HashMap;  
  
public class Example1 {  
 public static void main(String[] args) throws SQLException {  
 String url = "jdbc:mysql://localhost:3306/jdbc";  
 String username = "root";  
 String password = "root";  
  
 try (  
 Connection connection = DriverManager.*getConnection*(url, username, password);  
 Statement statement = connection.createStatement()  
 ) {  
  
// int rowsUpdated = statement.executeUpdate("UPDATE product SET price=10 WHERE name='CHOCO'");  
// System.out.println("Rows updated: " + rowsUpdated);  
  
  
 ResultSet res = statement.executeQuery("SELECT \* FROM student");  
  
  
 while (res.next()) {  
 int id = res.getInt("id");  
 String name = res.getString("name");  
 int age = res.getInt("age");  
  
 System.*out*.println("ID: " + id + ", Name: " + name + ", Age: " + age);  
 }  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
}

1. Insert Using PreparedStatement

Modify your earlier insert program to use PreparedStatement instead of Statement to prevent SQL injection

Solution:

package com.nisum;  
  
import java.sql.\*;  
import java.util.HashMap;  
  
public class Example1 {  
 public static void main(String[] args) throws SQLException {  
 String url = "jdbc:mysql://localhost:3306/jdbc";  
 String username = "root";  
 String password = "root";  
  
 try (  
 Connection connection = DriverManager.*getConnection*(url, username, password);  
 PreparedStatement preparedStatement = connection.prepareStatement("SELECT \* FROM student");  
 ) {  
  
// int rowsUpdated = statement.executeUpdate("UPDATE product SET price=10 WHERE name='CHOCO'");  
// System.out.println("Rows updated: " + rowsUpdated);  
  
  
 ResultSet res = preparedStatement.executeQuery();  
  
  
 while (res.next()) {  
 int id = res.getInt("id");  
 String name = res.getString("name");  
 int age = res.getInt("age");  
  
 System.*out*.println("ID: " + id + ", Name: " + name + ", Age: " + age);  
 }  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
}

1. Update a Record

Write a program to update a student’s name using their ID via PreparedStatement

Solution:

package com.nisum;  
  
import java.sql.\*;  
import java.util.HashMap;  
  
public class Example6 {  
 public static void main(String[] args) throws SQLException {  
 String url = "jdbc:mysql://localhost:3306/jdbc";  
 String username = "root";  
 String password = "root";  
 int newID=1;  
 String newName="Anti";  
  
 try (  
 Connection connection = DriverManager.*getConnection*(url, username, password);  
 PreparedStatement preparedStatement = connection.prepareStatement("SELECT \* FROM product");  
 PreparedStatement updateStatement = connection.prepareStatement("UPDATE product SET name=? WHERE id=?");  
 ) {  
  
 updateStatement.setString(1,newName);  
 updateStatement.setInt(2,newID);  
  
 int rowsupdated = updateStatement.executeUpdate();  
 System.*out*.println(rowsupdated);  
  
  
 ResultSet res = preparedStatement.executeQuery();  
  
  
 while (res.next()) {  
 int id = res.getInt("id");  
 String name = res.getString("name");  
 int age = res.getInt("price");  
  
 System.*out*.println("ID: " + id + ", Name: " + name + ", Age: " + age);  
 }  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
}

1. Delete a Record by ID

Implement a program that takes a student ID as input and deletes the record

Solution:

package com.nisum;  
  
import java.sql.\*;  
import java.util.HashMap;  
import java.util.Scanner;  
  
public class Example6 {  
 public static void main(String[] args) throws SQLException {  
 String url = "jdbc:mysql://localhost:3306/jdbc";  
 String username = "root";  
 String password = "root";  
// int newID=1;  
// String newName="Anti";  
 Scanner scanner=new Scanner(System.*in*);  
 System.*out*.println("Enter the id to delete it : ");  
 int studentID= scanner.nextInt();  
  
 try (  
 Connection connection = DriverManager.*getConnection*(url, username, password);  
 PreparedStatement preparedStatement = connection.prepareStatement("SELECT \* FROM product");  
// PreparedStatement updateStatement = connection.prepareStatement("UPDATE product SET name=? WHERE id=?");  
 PreparedStatement deleteStatement = connection.prepareStatement("Delete FROM product2 where id =?");  
 ) {  
  
// updateStatement.setString(1,newName);  
// updateStatement.setInt(2,newID);  
 deleteStatement.setInt(1,studentID);  
  
 int rowsupdated = deleteStatement.executeUpdate();  
 System.*out*.println(rowsupdated);  
  
  
 ResultSet res = preparedStatement.executeQuery();  
  
  
 while (res.next()) {  
 int id = res.getInt("id");  
 String name = res.getString("name");  
 int age = res.getInt("price");  
  
 System.*out*.println("ID: " + id + ", Name: " + name + ", Age: " + age);  
 }  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
}

1. Search with Wildcards

Use LIKE in SQL to search students by partial name (e.g., input: "Ra" returns "Ravi", "Ramesh")

Solution:

package com.nisum;  
  
import java.sql.\*;  
import java.util.HashMap;  
import java.util.Scanner;  
  
public class Example6 {  
 public static void main(String[] args) throws SQLException {  
 String url = "jdbc:mysql://localhost:3306/jdbc";  
 String username = "root";  
 String password = "root";  
// int newID=1;  
// String newName="Anti";  
// Scanner scanner=new Scanner(System.in);  
// System.out.println("Enter the id to delete it : ");  
// int studentID= scanner.nextInt();  
  
 try (  
 Connection connection = DriverManager.*getConnection*(url, username, password);  
 PreparedStatement preparedStatement = connection.prepareStatement("SELECT \* FROM product where name LIKE 'C%'");  
// PreparedStatement updateStatement = connection.prepareStatement("UPDATE product SET name=? WHERE id=?");  
// PreparedStatement deleteStatement = connection.prepareStatement("Delete FROM product2 where id =?");  
  
 ) {  
  
// updateStatement.setString(1,newName);  
// updateStatement.setInt(2,newID);  
// deleteStatement.setInt(1,studentID);  
//  
// int rowsupdated = deleteStatement.executeUpdate();  
// System.out.println(rowsupdated);  
  
  
 ResultSet res = preparedStatement.executeQuery();  
  
  
 while (res.next()) {  
 int id = res.getInt("id");  
 String name = res.getString("name");  
 int age = res.getInt("price");  
  
 System.*out*.println("ID: " + id + ", Name: " + name + ", Age: " + age);  
 }  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
}

1. Transaction Management

Write a program that inserts into two tables in a single transaction. Roll back if any insert fails

Solution:

package com.nisum;  
  
import java.sql.\*;  
  
public class Example5 {  
 public static void main(String[] args) throws SQLException {  
 String url="jdbc:mysql://localhost:3306/jdbc";  
 String username="root";  
 String password="root";  
 Connection connection = DriverManager.*getConnection*(url,username,password);  
 Statement statement=connection.createStatement();  
 ResultSet res=statement.executeQuery("Select \* from accounts");  
 while(res.next()){  
 System.*out*.println(res.getInt("id")+" "+res.getString("name")+" "+res.getString("balance"));  
 }  
 connection.setAutoCommit(false);  
 try{  
 statement.executeUpdate("update accounts set balance = balance-100 where name='John'");  
 System.*out*.println("Transaction commited");  
 System.*out*.println("------------------");  
 if(true)throw new RuntimeException("error");  
 statement.executeUpdate("update accounts set balance = balance+100 where name='Jane'");  
 connection.commit();  
 System.*out*.println("Transaction commited");  
  
 }catch (Exception e){  
 System.*out*.println("rolled back");  
 connection.rollback();  
 }  
  
 }  
}

1. Call Stored Procedure

Create a stored procedure in DB (e.g., getStudentById) and write a Java program to call it using CallableStatement

This topic was not covered int the sessions.

1. Handle Exceptions Gracefully

Enhance one of your earlier programs to:

* 1. Use try-with-resources
  2. Catch SQLException and print detailed error info (SQLState, error code, message)

Solution:

package com.nisum;  
  
import java.sql.\*;  
import java.util.HashMap;  
import java.util.Scanner;  
  
public class Example6 {  
 public static void main(String[] args) throws SQLException {  
 String url = "jdbc:mysql://localhost:3306/jdbc";  
 String username = "root";  
 String password = "root";  
 int newID=1;  
 String newName="Anti";  
 Scanner scanner=new Scanner(System.*in*);  
 System.*out*.println("Enter the id to delete it : ");  
 int studentID= scanner.nextInt();  
  
 try (  
 Connection connection = DriverManager.*getConnection*(url, username, password);  
 PreparedStatement preparedStatement = connection.prepareStatement("SELECT \* FROM product where name LIKE 'C%'");  
 PreparedStatement updateStatement = connection.prepareStatement("UPDATE product SET name=? WHERE id=?");  
 PreparedStatement deleteStatement = connection.prepareStatement("Delete FROM product2 where id =?");  
  
 ) {  
  
 updateStatement.setString(1,newName);  
 updateStatement.setInt(2,newID);  
 deleteStatement.setInt(1,studentID);  
  
 int rowsupdated = deleteStatement.executeUpdate();  
 System.*out*.println(rowsupdated);  
  
  
 ResultSet res = preparedStatement.executeQuery();  
  
  
 while (res.next()) {  
 int id = res.getInt("id");  
 String name = res.getString("name");  
 int age = res.getInt("price");  
  
 System.*out*.println("ID: " + id + ", Name: " + name + ", Age: " + age);  
 }  
  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
}