Relational Databases with MySQL Week 10 Assignment

Points possible: 70

Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete.	25

Instructions: In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document to the repository. Additionally, push an .sql file with all your queries and your Java project code to the same repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

Coding Steps:

In this week's coding activity, you will create a menu driven application backed by a MySQL database.

To start, choose one item that you like. It could be vehicles, sports, foods, etc....

Create a new Java project in Eclipse.

Create a SQL script in the project to create a database with one table. The table should be the item you picked.

Write a Java menu driven application that allows you to perform all four CRUD operations on your table.

Tips:

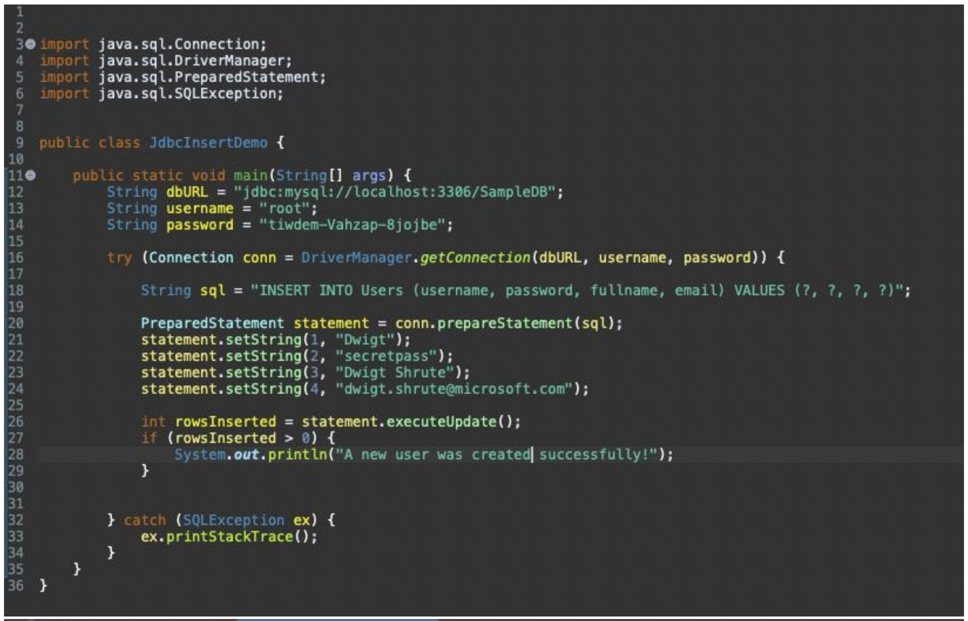
The application does not need to be as complex as the example in the video curriculum.

You need an option for each of the CRUD operations (Create, Read, Update, and Delete).

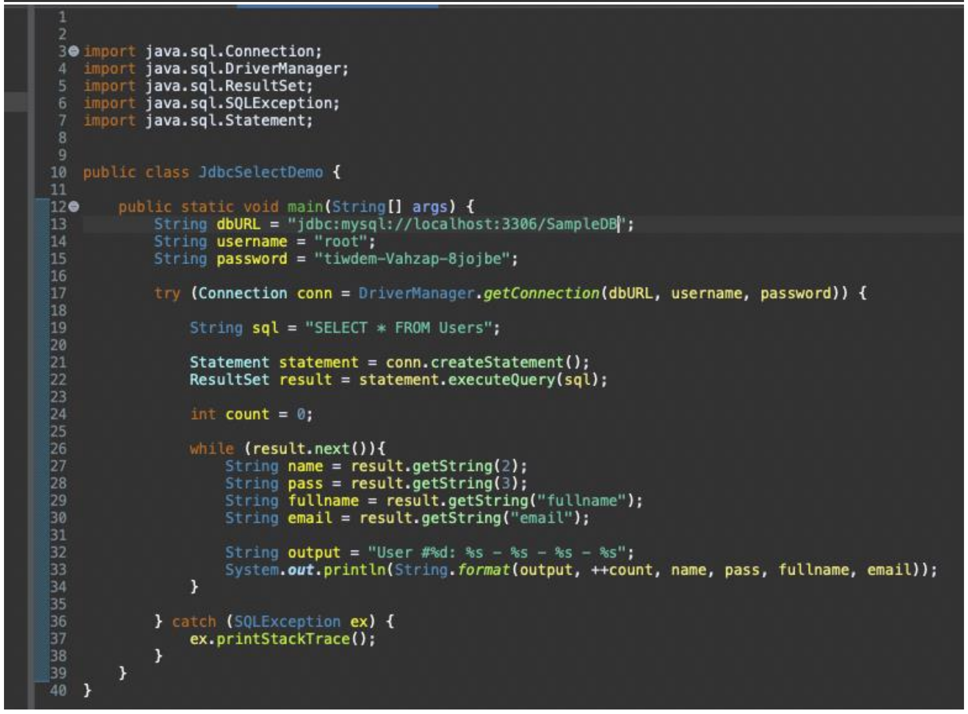
Remember that PreparedStatment.executeQuery() is only for Reading data and .executeUpdate() is used for Creating, Updating, and Deleting data.

Remember that both parameters on PreparedStatements and the ResultSet columns are based on indexes that start with 1, not 0.

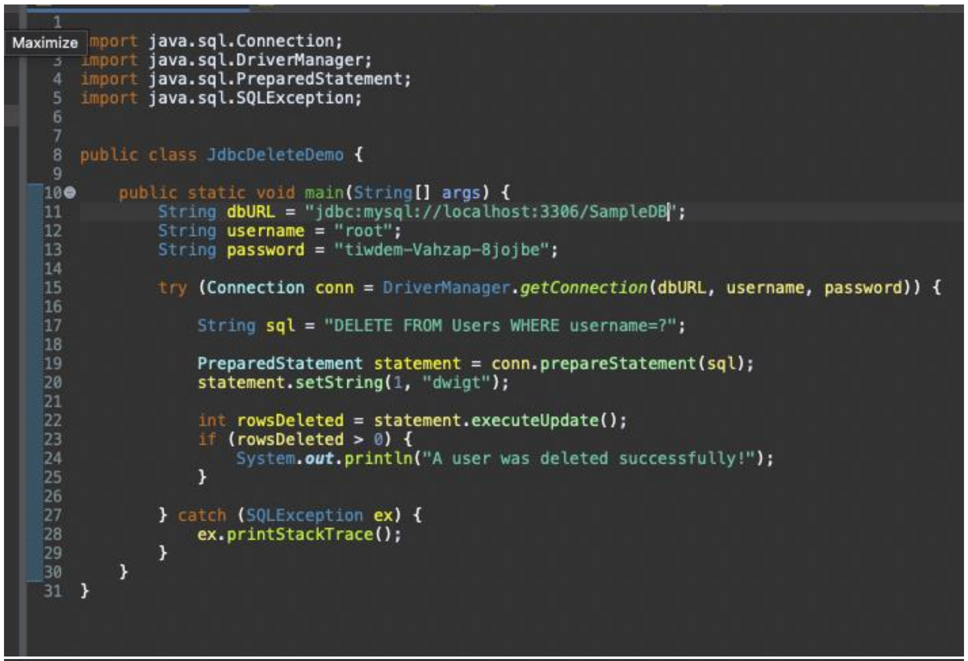
Screenshots of Code:



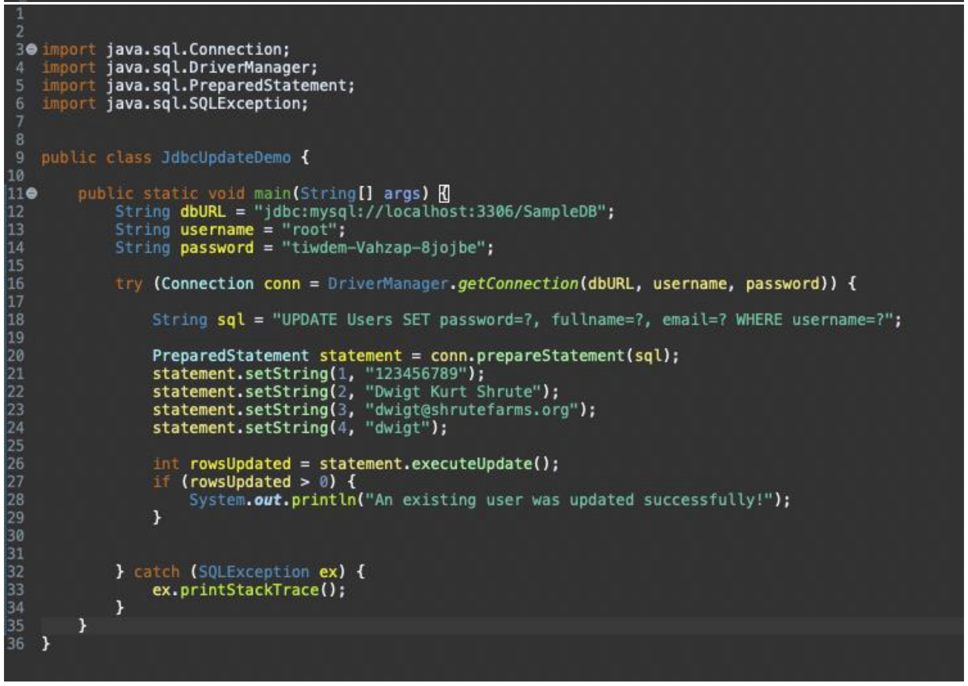
```java


import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;


public class JdbcInsertDemo {

    public static void main(String[] args) {
        String dbURL = "jdbc:mysql://localhost:3306/SampleDB";
        String username = "root";
        String password = "tiwdem-Vahzap-8jojbe";

        try (Connection conn = DriverManager.getConnection(dbURL, username, password)) {

            String sql = "INSERT INTO Users (username, password, fullname, email) VALUES (?, ?, ?, ?)";

            PreparedStatement statement = conn.prepareStatement(sql);
            statement.setString(1, "Dwigt");
            statement.setString(2, "secretpass");
            statement.setString(3, "Dwigt Shrute");
            statement.setString(4, "dwigt.shrute@microsoft.com");

            int rowsInserted = statement.executeUpdate();
            if (rowsInserted > 0) {
                System.out.println("A new user was created successfully!");
            }


        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}
```



```java


import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;


public class JdbcSelectDemo {

    public static void main(String[] args) {
        String dbURL = "jdbc:mysql://localhost:3306/SampleDB";
        String username = "root";
        String password = "tiwdem-Vahzap-8jojbe";

        try (Connection conn = DriverManager.getConnection(dbURL, username, password)) {

            String sql = "SELECT * FROM Users";

            Statement statement = conn.createStatement();
            ResultSet result = statement.executeQuery(sql);

            int count = 0;

            while (result.next()){
                String name = result.getString(2);
                String pass = result.getString(3);
                String fullname = result.getString("fullname");
                String email = result.getString("email");

                String output = "User #%d: %s - %s - %s - %s";
                System.out.println(String.format(output, ++count, name, pass, fullname, email));
            }

        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}
```

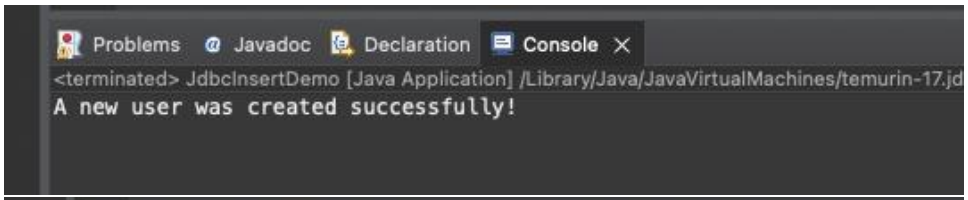


```java

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;


public class JdbcDeleteDemo {

    public static void main(String[] args) {
        String dbURL = "jdbc:mysql://localhost:3306/SampleDB";
        String username = "root";
        String password = "tiwdem-Vahzap-8jojbe";

        try (Connection conn = DriverManager.getConnection(dbURL, username, password)) {

            String sql = "DELETE FROM Users WHERE username=?";

            PreparedStatement statement = conn.prepareStatement(sql);
            statement.setString(1, "dwigt");

            int rowsDeleted = statement.executeUpdate();
            if (rowsDeleted > 0) {
                System.out.println("A user was deleted successfully!");
            }

        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}
```

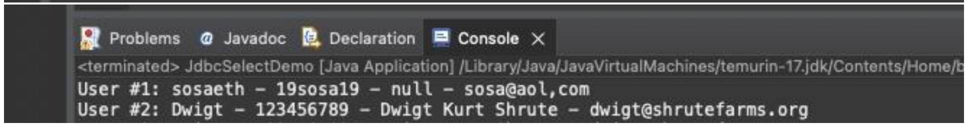


```java


import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;


public class JdbcUpdateDemo {

    public static void main(String[] args) {
        String dbURL = "jdbc:mysql://localhost:3306/SampleDB";
        String username = "root";
        String password = "tiwdem-Vahzap-8jojbe";

        try (Connection conn = DriverManager.getConnection(dbURL, username, password)) {

            String sql = "UPDATE Users SET password=?, fullname=?, email=? WHERE username=?";

            PreparedStatement statement = conn.prepareStatement(sql);
            statement.setString(1, "123456789");
            statement.setString(2, "Dwigt Kurt Shrute");
            statement.setString(3, "dwigt@shrutefarms.org");
            statement.setString(4, "dwigt");

            int rowsUpdated = statement.executeUpdate();
            if (rowsUpdated > 0) {
                System.out.println("An existing user was updated successfully!");
            }


        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}
```

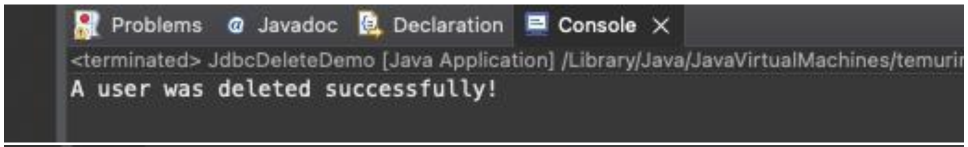
Screenshots of Running Application:



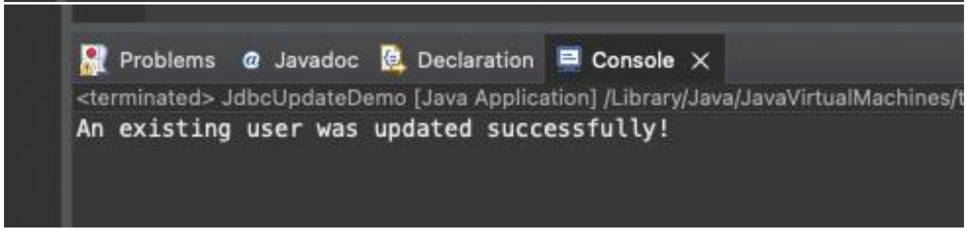
```
<terminated> JdbcInsertDemo [Java Application] /Library/Java/JavaVirtualMachines/temurin-17.jd
A new user was created successfully!
```



```
<terminated> JdbcSelectDemo [Java Application] /Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/b
User #1: sosaeth - 19sosa19 - null - sosa@aol,com
User #2: Dwigt - 123456789 - Dwigt Kurt Shrute - dwigt@shrutefarms.org
```



```
<terminated> JdbcDeleteDemo [Java Application] /Library/Java/JavaVirtualMachines/temuri
A user was deleted successfully!
```



```
<terminated> JdbcUpdateDemo [Java Application] /Library/Java/JavaVirtualMachines/t
An existing user was updated successfully!
```

URL to GitHub Repository:

https://github.com/Asosa809/CRUDsql