

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:

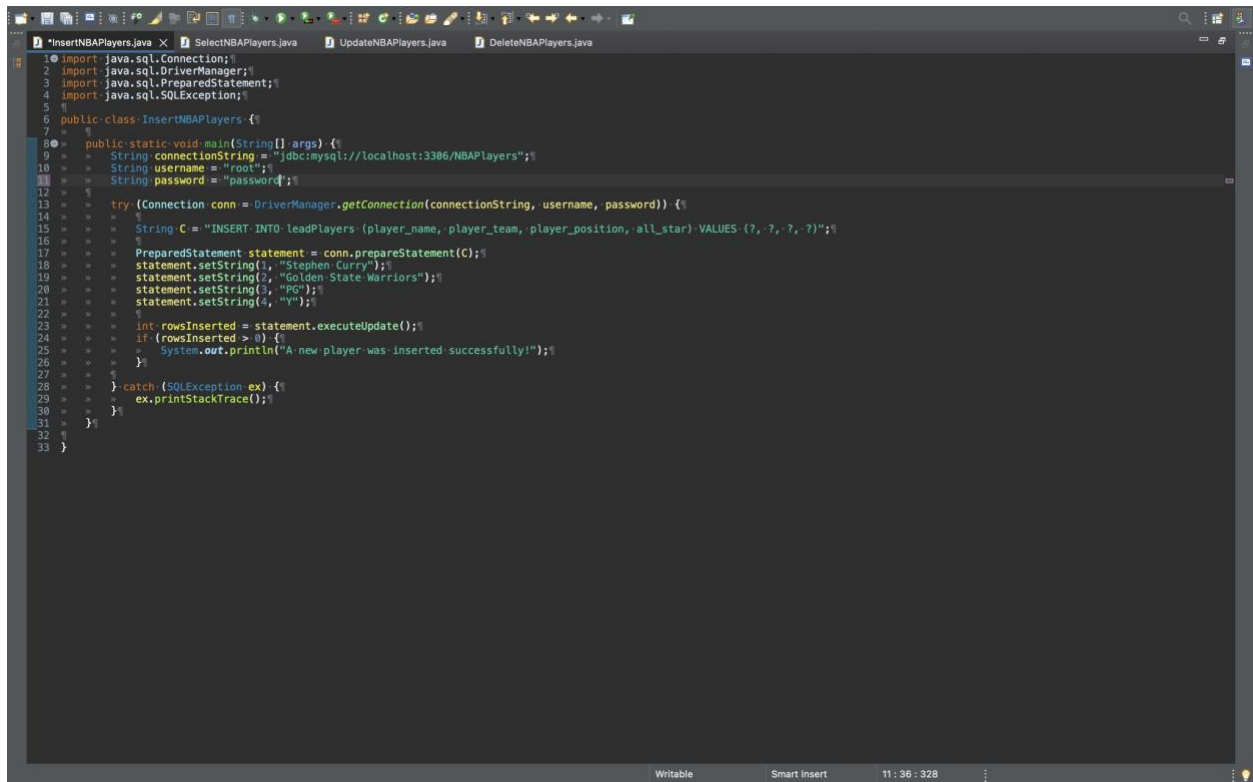
The application does not need to be as complex as the example in the video curriculum.

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

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

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

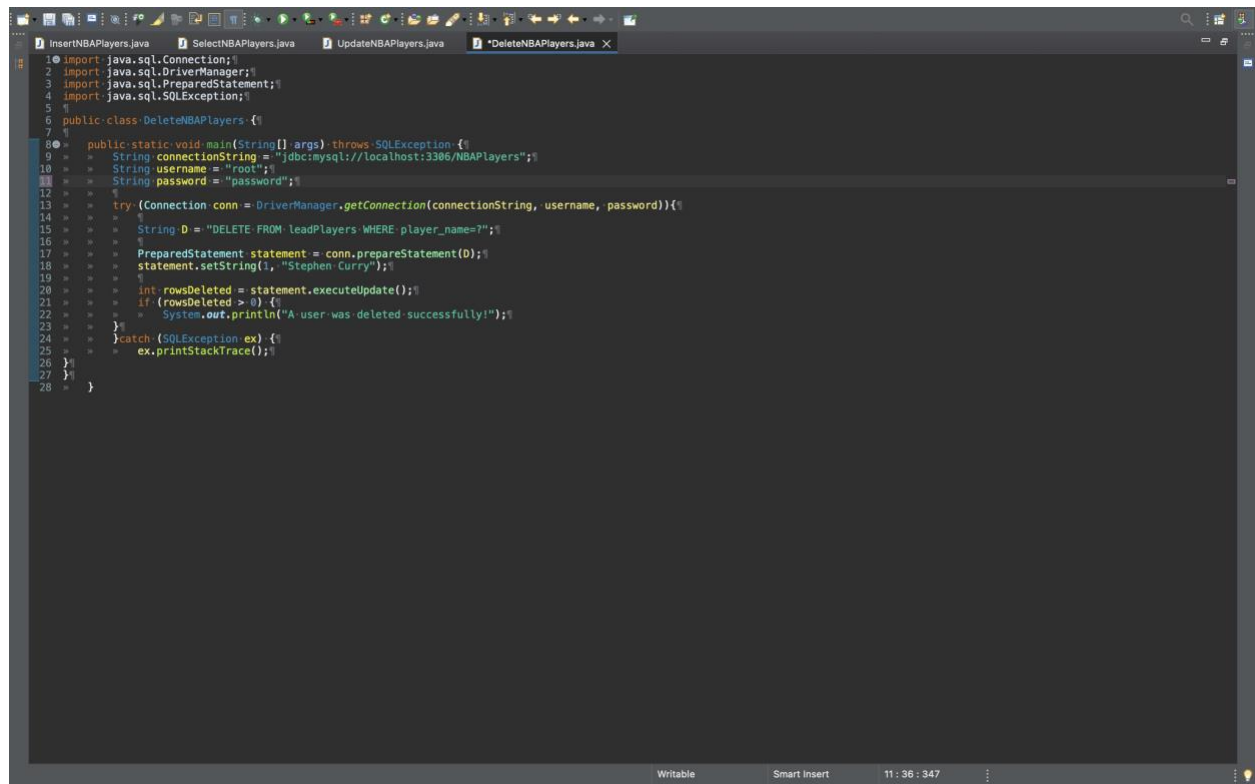
Screenshots of Code:

A screenshot of an IDE window showing the code for `InsertNBAPlayers.java`. The code imports `java.sql.Connection`, `java.sql.DriverManager`, `java.sql.PreparedStatement`, and `java.sql.SQLException`. It defines a public class `InsertNBAPlayers` with a `main` method. Inside `main`, it sets up a JDBC connection string, username, and password. It then attempts to get a connection from `DriverManager`. A SQL statement is prepared: `INSERT INTO leadPlayers (player_name, player_team, player_position, all_star) VALUES (?, ?, ?, ?)`. The statement is executed using `executeUpdate()`. If the update is successful, it prints a message. A try-catch block handles `SQLException` by printing the stack trace.

```
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.PreparedStatement;
4 import java.sql.SQLException;
5
6 public class InsertNBAPlayers {
7     public static void main(String[] args) {
8         String connectionString = "jdbc:mysql://localhost:3306/NBAPlayers";
9         String username = "root";
10        String password = "password";
11
12        try {
13            Connection conn = DriverManager.getConnection(connectionString, username, password);
14
15            String C = "INSERT INTO leadPlayers (player_name, player_team, player_position, all_star) VALUES (?, ?, ?, ?)";
16
17            PreparedStatement statement = conn.prepareStatement(C);
18            statement.setString(1, "Stephen Curry");
19            statement.setString(2, "Golden State Warriors");
20            statement.setString(3, "PG");
21            statement.setString(4, "Y");
22
23            int rowsInserted = statement.executeUpdate();
24            if (rowsInserted > 0) {
25                System.out.println("A new player was inserted successfully!");
26            }
27        } catch (SQLException ex) {
28            ex.printStackTrace();
29        }
30    }
31 }
32
33 }
```

```
InsertNBAPlayers.java  *SelectNBAPlayers.java  UpdateNBAPlayers.java  DeleteNBAPlayers.java
1 import java.sql.Connection;
2
3 public class SelectNBAPlayers {
4
5     public static void main(String[] args) {
6         String connectionString = "jdbc:mysql://localhost:3306/NBAPlayers";
7         String username = "root";
8         String password = "password";
9
10        try {
11            Connection conn = DriverManager.getConnection(connectionString, username, password);
12
13            String R = "SELECT * FROM leadPlayers";
14            Statement sm = conn.createStatement();
15            ResultSet rs = sm.executeQuery(R);
16
17            int count = 0;
18
19            while (rs.next()) {
20                String player_name = rs.getString("player_name");
21                String player_team = rs.getString("player_team");
22                String player_position = rs.getString("player_position");
23                String all_star = rs.getString("all_star");
24
25                String output = "User #id: %s -- %s -- %s -- %s";
26                System.out.println(String.format(output, ++count, player_name, player_team, player_position, all_star));
27            }
28        } catch (SQLException ex) {
29            ex.printStackTrace();
30        }
31    }
32}
33
34
35
36
37
38
39
40
41
42
```

```
InsertNBAPlayers.java  SelectNBAPlayers.java  *UpdateNBAPlayers.java  DeleteNBAPlayers.java
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.PreparedStatement;
4 import java.sql.SQLException;
5
6 public class UpdateNBAPlayers {
7
8     public static void main(String[] args) throws SQLException {
9         String connectionString = "jdbc:mysql://localhost:3306/NBAPlayers";
10        String username = "root";
11        String password = "password";
12
13        try {
14            Connection conn = DriverManager.getConnection(connectionString, username, password);
15
16            String U = "UPDATE leadPlayers SET player_team=?, player_position=?, all_star=? WHERE player_name=?";
17            PreparedStatement ps = conn.prepareStatement(U);
18            ps.setString(1, "Golden State Warriors");
19            ps.setString(2, "PG");
20            ps.setString(3, "");
21            ps.setString(4, "Stephen Curry");
22
23            int rowsUpdated = ps.executeUpdate();
24            if (rowsUpdated > 0) {
25                System.out.println("An existing player was updated successfully!");
26            }
27        } catch (SQLException ex) {
28            ex.printStackTrace();
29        }
30    }
31}
32
33
```



```
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.PreparedStatement;
4 import java.sql.SQLException;
5
6 public class DeleteNBAPlayers {
7
8     public static void main(String[] args) throws SQLException {
9         String connectionString = "jdbc:mysql://localhost:3306/NBAPlayers";
10        String username = "root";
11        String password = "password";
12
13        try {
14            Connection conn = DriverManager.getConnection(connectionString, username, password);
15            String D = "DELETE FROM leadPlayers WHERE player_name=?";
16            PreparedStatement statement = conn.prepareStatement(D);
17            statement.setString(1, "Stephen Curry");
18            int rowsDeleted = statement.executeUpdate();
19            if (rowsDeleted > 0) {
20                System.out.println("A user was deleted successfully!");
21            }
22        } catch (SQLException ex) {
23            ex.printStackTrace();
24        }
25    }
26 }
27
28 }
```

Writtable Smart Insert 11:36:347

Screenshots of Running Application:

```
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.PreparedStatement;
4 import java.sql.SQLException;
5
6 public class InsertNBAPlayers {
7     public static void main(String[] args) {
8         String connectionString = "jdbc:mysql://localhost:3306/NBAPlayers";
9         String username = "root";
10        String password = "password";
11
12        try {
13            Connection conn = DriverManager.getConnection(connectionString, username, password);
14
15            String C = "INSERT INTO leadPlayers (player_name, player_team, player_position, all_star) VALUES (?, ?, ?, ?)";
16
17            PreparedStatement statement = conn.prepareStatement(C);
18            statement.setString(1, "Stephen Curry");
19            statement.setString(2, "Golden State Warriors");
20            statement.setString(3, "PG");
21            statement.setString(4, "Y");
22
23            int rowsInserted = statement.executeUpdate();
24            if (rowsInserted > 0) {
25                System.out.println("A new player was inserted successfully!");
26            }
27        } catch (SQLException ex) {
28
29        }
```

Console

<terminated>: InsertNBAPlayers [Java Application] /Library/Java/JavaVirtualMachines/jdk-11.0.14.jdk/Contents/Home/bin/java (Jul 12, 2022, 9:33:39 PM - 9:33:40 PM) [pid: 30654]

A new player was inserted successfully!

```
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.PreparedStatement;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6
7 public class SelectNBAPlayers {
8     public static void main(String[] args) {
9         String connectionString = "jdbc:mysql://localhost:3306/NBAPlayers";
10        String username = "root";
11        String password = "password";
12
13        try {
14            Connection conn = DriverManager.getConnection(connectionString, username, password);
15
16            String R = "SELECT * FROM leadPlayers";
17
18            Statement sm = conn.createStatement();
19            ResultSet rs = sm.executeQuery(R);
20
21            int count = 0;
22
23            while (rs.next()) {
24                String player_name = rs.getString("player_name");
25                String player_team = rs.getString("player_team");
26                String player_position = rs.getString("player_position");
27                String all_star = rs.getString("all_star");
28
29                String output = "User #d: %s - %s - %s - %s";
30                System.out.println(String.format(output, ++count, player_name, player_team, player_position, all_star));
31            }
32        } catch (SQLException ex) {
33
34        }
```

Console

<terminated>: InsertNBAPlayers [Java Application] /Library/Java/JavaVirtualMachines/jdk-11.0.14.jdk/Contents/Home/bin/java (Jul 12, 2022, 9:39:06 PM - 9:39:08 PM) [pid: 30697]

User #1: Stephen Curry - Golden State Warriors - PG - Y

```
InsertNBAPlayers.java  SelectNBAPlayers.java  *UpdateNBAPlayers.java  DeleteNBAPlayers.java
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.PreparedStatement;
4 import java.sql.SQLException;
5
6 public class UpdateNBAPlayers {
7
8     public static void main(String[] args) throws SQLException {
9         String connectionString = "jdbc:mysql://localhost:3306/NBAPlayers";
10        String username = "root";
11        String password = "password";
12
13        try {
14            Connection conn = DriverManager.getConnection(connectionString, username, password);
15            String U = "UPDATE leadPlayers SET player_team=?, player_position=?, all_star=? WHERE player_name=?";
16            PreparedStatement ps = conn.prepareStatement(U);
17            ps.setString(1, "Golden State Warriors");
18            ps.setString(2, "PG");
19            ps.setString(3, "Y");
20            ps.setString(4, "Stephen Curry");
21            int rowsUpdated = ps.executeUpdate();
22            if (rowsUpdated > 0) {
23                System.out.println("An existing player was updated successfully!");
24            }
25        } catch (SQLException ex) {
26            ex.printStackTrace();
27        }
28    }
29 }
```

Console

<terminated> InsertNBAPlayers [Java Application] /Library/Java/JavaVirtualMachines/jdk-11.0.14.jdk/Contents/Home/bin/java (Jul 12, 2022, 9:42:57 PM - 9:43:00 PM) [pid: 30808]
An existing player was updated successfully!

```
InsertNBAPlayers.java  SelectNBAPlayers.java  UpdateNBAPlayers.java  DeleteNBAPlayers.java
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.PreparedStatement;
4 import java.sql.SQLException;
5
6 public class DeleteNBAPlayers {
7
8     public static void main(String[] args) throws SQLException {
9         String connectionString = "jdbc:mysql://localhost:3306/NBAPlayers";
10        String username = "root";
11        String password = "password";
12
13        try {
14            Connection conn = DriverManager.getConnection(connectionString, username, password);
15            String D = "DELETE FROM leadPlayers WHERE player_name=?";
16            PreparedStatement statement = conn.prepareStatement(D);
17            statement.setString(1, "Stephen Curry");
18            int rowsDeleted = statement.executeUpdate();
19            if (rowsDeleted > 0) {
20                System.out.println("A user was deleted successfully!");
21            }
22        } catch (SQLException ex) {
23            ex.printStackTrace();
24        }
25    }
26 }
```

Console

<terminated> InsertNBAPlayers [Java Application] /Library/Java/JavaVirtualMachines/jdk-11.0.14.jdk/Contents/Home/bin/java (Jul 12, 2022, 9:45:38 PM - 9:45:39 PM) [pid: 30859]
A user was deleted successfully!

Writable Smart Insert 25 : 34 : 813

URL to GitHub Repository: