

# 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.

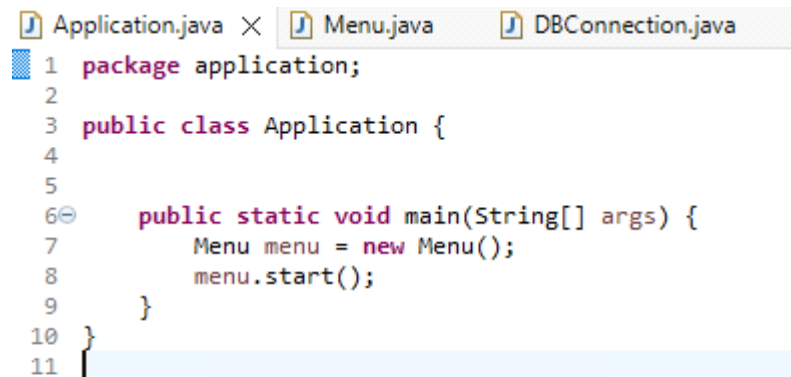
You need an option for each of the CRUD operations (Create, Read, Update, and Delete).

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:

### Application



```
1 package application;
2
3 public class Application {
4
5
6     public static void main(String[] args) {
7         Menu menu = new Menu();
8         menu.start();
9     }
10 }
11
```

### Menu:



```
1 package application;
2
3 import java.sql.SQLException;
4
5 public class Menu {
6     private ArtistDao artistDao = new ArtistDao();
7     private GenreDao genreDao = new GenreDao();
8     private Scanner scanner = new Scanner(System.in);
9     private List<String> options = Arrays.asList("Display Artist", "Display a Artist", "Create Artist", "Delete Artist", "Create Genre", "Destroy Genre");
10
11     public void start() {
12         String selection = "";
13         do {
14             printMenu();
15             selection = scanner.nextLine();
16
17             try {
18                 if (selection.equals("1")) {
19                     displayArtists();
20                 } else if (selection.equals("2")) {
21                     displayArtist();
22                 } else if (selection.equals("3")) {
23                     createArtist();
24                 } else if (selection.equals("4")) {
25                     deleteArtist();
26                 } else if (selection.equals("5")) {
27                     createGenre();
28                 } else if (selection.equals("6")) {
29                     deleteGenre();
30                 }
31             } catch (SQLException e) {
32                 e.printStackTrace();
33             }
34
35             System.out.println("Press Enter to continue: ");
36             scanner.nextLine();
37         } while (!selection.equals("-1"));
38     }
39
40     private void printMenu() {
41         System.out.println("Select an Option: \n");
42         for(int i = 0; i < options.size(); i++) {
43             System.out.println(i + 1 + " " + options.get(i));
44         }
45     }
46
47     private void displayArtists() throws SQLException {
48         List<Artist> artist = artistDao.getArtist();
49         for (Artist artist1 : artist) {
50             System.out.println(artist1.getArtistID() + " : " + artist1.getName());
51         }
52     }
53
54     private void displayArtist() throws SQLException {
55         System.out.print("Artist id: ");
56         int id = Integer.parseInt(scanner.nextLine());
57     }
58 }
59
```

```

65     Artist artist = artistDao.getArtistById(id);
66     System.out.println(artist.getArtistID() + ":" + artist.getName());
67     for (Genre genre : artist.getGenre()) {
68         System.out.println("\tArtistID: " + genre.getGenreID() + " Name: " + genre.getName());
69     }
70 }
71 }
72
73 private void createArtist() throws SQLException {
74     System.out.print("Enter Artist name:");
75     String artistName = scanner.nextLine();
76     artistDao.createNewArtist(artistName);
77 }
78
79 private void deleteArtist() throws SQLException {
80     System.out.print("Enter Artist ID: ");
81     int id = Integer.parseInt(scanner.nextLine());
82     artistDao.deleteArtistById(id);
83 }
84 private void createGenre() throws SQLException {
85     System.out.println("Enter name of Genre:");
86     String name = scanner.nextLine();
87     System.out.println("Enter Genre ID: ");
88     int genreID = Integer.parseInt(scanner.nextLine());
89     genreDao.createNewGenre(name, genreID);
90 }
91 private void deleteGenre() throws SQLException {
92     System.out.println("Enter Genre ID: ");
93     int id = Integer.parseInt(scanner.nextLine());
94     genreDao.deleteGenreById(id);
95 }
96 }
97

```

## DBConnection:

```

1 package dao;
2
3 import java.sql.Connection;
4
5
6
7 public class DBConnection {
8
9
10     private final static String con = "jdbc:mysql://localhost:3306/music";
11     private final static String Username = "root";
12     private final static String password = "dedira003";
13     private static Connection connection;
14     private static DBConnection instance;
15     private DBConnection(Connection connection) {
16         this.connection = connection;
17     }
18
19     public static Connection getConnection() {
20         if (instance == null) {
21             try {
22                 connection = DriverManager.getConnection(con, Username, password);
23                 instance = new DBConnection(connection);
24                 System.out.println("Connection Successfull");
25             } catch (SQLException e) {
26                 e.printStackTrace();
27             }
28         }
29         return DBConnection.connection;
30     }
31 }
32

```

## Artist:

```

1 package entity;
2
3 import java.util.ArrayList;
4
5
6 public class Artist {
7
8     private int artistID;
9     private String name;
10    private List<Genre> genre = new ArrayList<Genre>();
11
12    public Artist(int artistID, String name, List<Genre> genre) {
13        this.setArtistID(artistID);
14        this.setName(name);
15    }
16
17    public int getArtistID() {
18        return artistID;
19    }
20
21    public void setArtistID(int artistID) {
22        this.artistID = artistID;
23    }
24
25    public String getName() {
26        return name;
27    }
28
29    public void setName(String name) {
30        this.name = name;
31    }
32
33    public List<Genre> getGenre() {
34        return genre;
35    }
36
37    public void setGenre(List<Genre> genre) {
38        this.genre = genre;
39    }
40 }
41

```

**ArtistDao:**

```

1 package dao;
2
3⊕ import java.sql.Connection;
11
12 public class ArtistDao {
13     private Connection connection;
14     private GenreDao genreDao;
15     private final String GET_ARTIST_QUERY = "SELECT * FROM artist";
16     private final String GET_ARTIST_BY_ID_QUERY = "SELECT * FROM artist WHERE id = ?";
17     private final String CREATE_NEW_ARTIST_QUERY = "INSERT INTO artist(name) VALUES(?)";
18     private final String DELETE_ARTIST_BY_ID_QUERY = "DELETE FROM artist WHERE id = ?";
19⊖ public ArtistDao() {
20         connection = DBConnection.getConnection();
21         genreDao = new GenreDao();
22     }
23
24⊖ public List<Artist> getArtist() throws SQLException {
25     ResultSet rs = connection.prepareStatement(GET_ARTIST_QUERY).executeQuery();
26     List<Artist> artist = new ArrayList<Artist>();
27
28     while (rs.next()) {
29         artist.add(populateArtist(rs.getInt(1), rs.getString(2)));
30     }
31
32     return artist;
33 }
34⊖ public Artist getArtistById(int id) throws SQLException {
35     PreparedStatement ps = connection.prepareStatement(GET_ARTIST_BY_ID_QUERY);
36     ps.setInt(1, id);
37     ResultSet rs = ps.executeQuery();
38     rs.next();
39     return populateArtist(rs.getInt(1), rs.getString(2));
40 }
41
42⊖ public void createNewArtist(String artistName) throws SQLException {
43     PreparedStatement ps = connection.prepareStatement(CREATE_NEW_ARTIST_QUERY);
44     ps.setString(1, artistName);
45     ps.executeUpdate();
46 }
47
48⊖ public void deleteArtistById(int id) throws SQLException {
49     genreDao.deleteGenreByGenreId(id);
50     PreparedStatement ps = connection.prepareStatement(DELETE_ARTIST_BY_ID_QUERY);
51     ps.setInt(1, id);
52     ps.executeUpdate();
53 }
54
55⊖ private Artist populateArtist(int id, String name) throws SQLException {
56     return new Artist(id, name, genreDao.getGenreByArtistId(id));
57 }
58 }
59 {

```

**GenreDao:**

```

1 package dao;
2
3⊕ import java.sql.Connection;
11
12 public class GenreDao {
13
14     private Connection connection;
15     private final String GET_GENRE_BY_ARTIST_ID_QUERY = "SELECT * FROM genre WHERE genre_id = ?";
16     private final String DELETE_GENRE_BY_GENRE_ID_QUERY = "DELETE FROM genre WHERE genre_id = ?";
17     private final String CREATE_NEW_GENRE_QUERY = "INSERT INTO genre(name, genre_id) VALUES(?,?)";
18     private final String DELETE_GENRE_BY_ID_QUERY = "DELETE FROM genre WHERE id =?";
19⊖ public GenreDao() {
20         connection = DBConnection.getConnection();
21     }
22
23⊖ public List<Genre> getGenreByArtistId(int id) throws SQLException {
24     PreparedStatement ps = connection.prepareStatement(GET_GENRE_BY_ARTIST_ID_QUERY);
25     ps.setInt(1, id);
26     ResultSet rs = ps.executeQuery();
27     List<Genre> genre = new ArrayList<Genre>();
28
29     while (rs.next()) {
30         genre.add(new Genre(rs.getInt(1), rs.getString(2), rs.getInt(3)));
31     }
32
33     return genre;
34 }
35
36⊖ public void createNewGenre(String name, int genreID) throws SQLException {
37     PreparedStatement ps = connection.prepareStatement(CREATE_NEW_GENRE_QUERY);
38     ps.setString(1, name);
39     ps.setInt(2, genreID);
40     ps.executeUpdate();
41 }
42
43⊖ public void deleteGenreByGenreId(int genreID) throws SQLException {
44     PreparedStatement ps = connection.prepareStatement(DELETE_GENRE_BY_GENRE_ID_QUERY);
45     ps.setInt(1, genreID);
46     ps.executeUpdate();
47 }
48
49⊖ public void deleteGenreById(int id) throws SQLException {
50     PreparedStatement ps = connection.prepareStatement(DELETE_GENRE_BY_ID_QUERY);
51     ps.setInt(1, id);
52     ps.executeUpdate();
53 }
54
55 }

```

**Genre:**

```

1 package entity;
2
3 public class Genre {
4
5     private int genreID;
6     private String name;
7
8     public Genre(int genreID, String name, int artist) {
9         this.setGenreID(genreID);
10        this.setName(name);
11    }
12
13
14    public int getGenreID() {
15        return genreID;
16    }
17
18    public void setGenreID(int genreID) {
19        this.genreID = genreID;
20    }
21
22    public String getName() {
23        return name;
24    }
25
26    public void setName(String name) {
27        this.name = name;
28    }
29

```

**Screenshots of Running Application:**

**Show method:**

```

Application (2) [Java Application] C:\Program Files\Java\
Connection Successfull
Select an Option:

1  Display Artist
2  Display a Artist
3  Create Artist
4  Delete Artist
5  Create Genre
6  Destroy Genre
1
2: Jeff
Press Enter to continue:

```

**Show methodId:**

```
Connection Successfull
Select an Option:

1  Display Artist
2  Display a Artist
3  Create Artist
4  Delete Artist
5  Create Genre
6  Destroy Genre
2
Artist id: 2
p: Jeff
Press Enter to continue:
```

### Create method:

```
Select an Option:

1  Display Artist
2  Display a Artist
3  Create Artist
4  Delete Artist
5  Create Genre
6  Destroy Genre
3
Enter Artist name: Billy Joel
Press Enter to continue:
```

### Delete method:

```
Select an Option:

1  Display Artist
2  Display a Artist
3  Create Artist
4  Delete Artist
5  Create Genre
6  Destroy Genre
4
Enter Artist ID: 3
Press Enter to continue:
```

```
Select an Option:

1  Display Artist
2  Display a Artist
3  Create Artist
4  Delete Artist
5  Create Genre
6  Destroy Genre
1
p: Jeff
Press Enter to continue:
```

### Create Genre:



Select an Option:

- 1 Display Artist
- 2 Display a Artist
- 3 Create Artist
- 4 Delete Artist
- 5 Create Genre
- 6 Destroy Genre

5

Enter name of Genre:

Rap

Enter Genre ID:

4

Press Enter to continue:

Select an Option:

- 1 Display Artist
- 2 Display a Artist
- 3 Create Artist
- 4 Delete Artist
- 5 Create Genre
- 6 Destroy Genre

1

2: Jeff

4: Snoop Dogg

### Delete Genre:

Connection Successfull

Select an Option:

- 1 Display Artist
- 2 Display a Artist
- 3 Create Artist
- 4 Delete Artist
- 5 Create Genre
- 6 Destroy Genre

6

Enter Genre ID:

4

Press Enter to continue:

URL to GitHub Repository: <https://github.com/Starssk1ttles/Week10Sql>