|  |  |
| --- | --- |
| sait_icon_wordmark_horiz_text_black | **CPRG 251**  **Assignment 4 (Module 12)**  **Winter 2021** |

**Name:**

**Marks:**  / 41

## **D2L Submission Instructions**

1. One ZIP file needs to submit to D2L with the following naming convention **CPRG251\_A4\_Firstname\_Lastname.zip** using your first and last name.

If working in a group of two (2) or more, only one team member needs to submit to D2L (both can if you so wish). Both members will receive the same feedback. The file should have the following naming convention: **CPRG251\_A4\_Lastname of member 1\_Lastname of member 2.zip**

1. The ZIP file must contain the following:   
   1. The following directory structure:

* bin/ – Compiled Java files.
* src/ – Java source code files:
  + sait/mms/application/
  + sait/mms/contracts/
  + sait/mms/drivers/
  + sait/mms/managers/
  + sait/mms/problemdomain/
* doc/ – Generated Javadoc files.
  + Ensure the *private* option is checked and everything is included in the generated documentation.
* lib/ – Any third-party libraries. This folder can be empty.
* res/ – Any resource or data files.
* test/ – Unit test cases (if requested).

1. A text file named **Readme.txt** in the root folder of the ZIP archive and contain:

* A project title.
* What the program does.
* The date.
* The author
* How to run the program.

1. A runnable JAR file in the root folder of the ZIP archive.
   1. Use the naming convention: **FirstInitial.Lastname1.jar** (i.e.: J.Blow1.jar).
   2. It is to be built using only Eclipse IDE and JDK 1.8x.

## **Assignment Instructions**

1. You will use only Eclipse IDE.
2. The due date for this assignment is posted in D2L in the assignment submission area and in the provided calendar located in the *Course Information* area**.** Any assignment submitted after the due date will receive a mark of zero, but feedback maybe given.
3. Submissions must be student’s original work. Refer to the Academic Misconduct (AC.3.4) policies and procedures.

## **Problem**

You are tasked with modifying the movie management system to work with a relational database system. You’re required to connect to and perform queries on a MariaDB database running on your computer.

Attached is a SQL script that creates and populates a table. Below are instructions on how to import the SQL script into your installation of MariaDB. The “movies” table has the following structure:

|  |  |
| --- | --- |
| Column Name | Data Type |
| id | **INT** |
| duration | **INT** |
| title | **VARCHAR(255)** |
| year | **YEAR** |

The appropriate SQL statements will be sent to and executed on the database. Depending on the query being performed, the proper method is to be used. If records are retrieved, they will be displayed to the user. Any exceptions that occur because of queries will need to be handled.

Movies will be retrieved and persisted to the database on-demand. This means they will not be loaded into memory when the program starts and saved from memory to the database when the program ends. You will not be using any problem domain classes.

Your project will contain SQL statements that:

1. Creates a new record representing a movie.
2. Retrieves records with movies released in a specific year.
3. Retrieves records with a list of random movies.
4. Delete a movie using its id

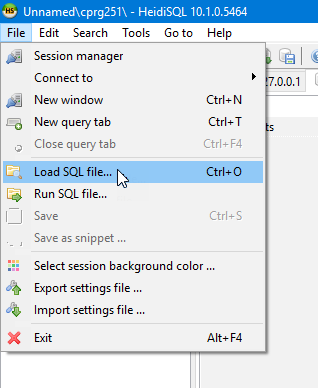
Notes:

* You can use **ORDER BY RAND()** to randomize the order of records.
* There is no need to specify a value for the ID column of a new movie record. This will be automatically set to the next available ID by the database.
* Ensure any results, queries, and connections are closed when they are no longer needed.

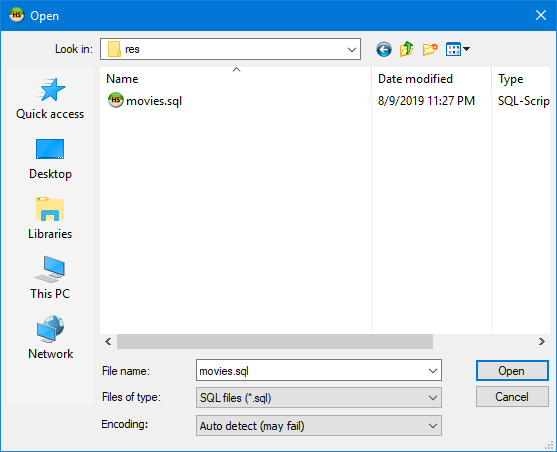
## Database Setup

Before continuing, ensure that you have followed the instructions included in the “MySQL Install Instructions” document.

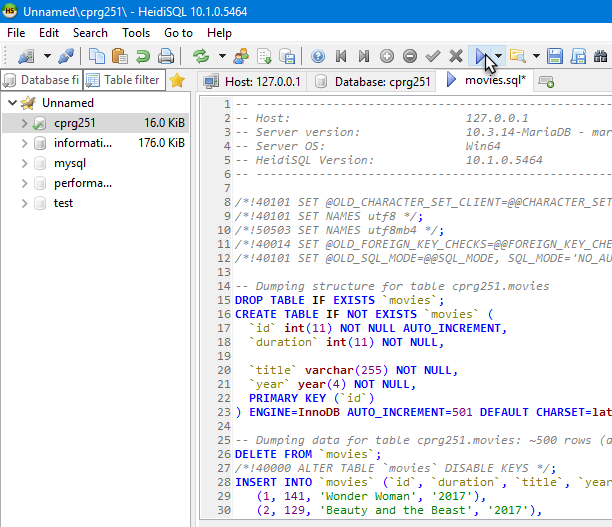
With HeidiSQL open, go to the “File” menu and click on “Load SQL File…”.



Navigate to the location of the “movies.sql” file. This should be in the “res” folder of your Eclipse project.

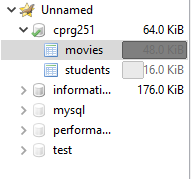


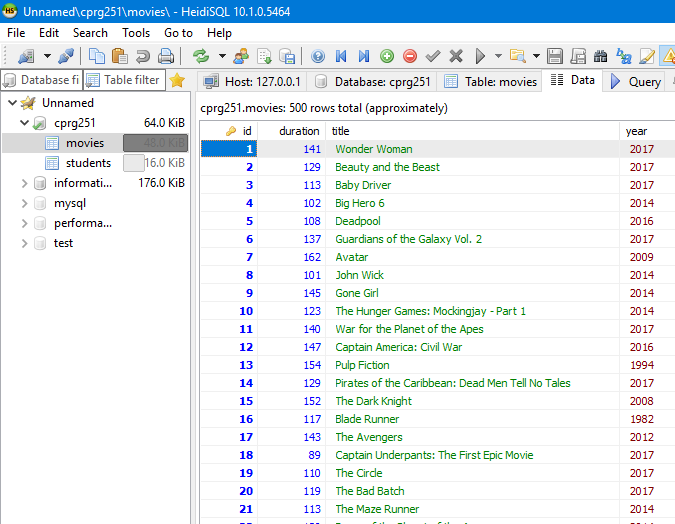
A tab will be created in HeidiSQL containing the contents of the “movies.sql” file. Make sure the “cprg251” database is selected on the left-hand side.



Finally, click the play button on the toolbar (or press F9) to run the SQL script. You may receive some warnings, and this is fine.

To make sure the table was created and populated, right click the “cprg251” database on the left-hand side and click “Refresh”. A new table called “movies” should be displayed and if not, check that the “cprg251” database is expanded.



To make sure the table is populated, select the table on the left side and then open the “Data” tab.

**Notes:**

## To follow the object-oriented principles, your project should contain ONLY the following classes and methods in their respective package.

|  |  |  |
| --- | --- | --- |
| **Package** | **Class** | **Methods** |
| sait.mms.application | appDriver | Main method |
| sait.mms.managers | MovieManagementSystem | displayMenu, addMovie, printMoviesInYear, printRandomMovies, deleteMovie |
| sait.mms.problemdomain | Movie | Set(s) and get(s) for duration, title, year  toString |
| Sait.mms.contracts | DatabaseDriver (Interface) | Connect, get(query), update, disconnect |
| Sait.mms.drivers | MariaDBDriver | Implementation of the interface methods (above methods) |
|  |  |  |
|  |  |  |

## **Marking Guide**

|  |  |  |  |
| --- | --- | --- | --- |
| **Follows submission guidelines** | | | |
| 1. | Correct files |  |  |
| 2. | File naming conventions followed |  |  |
| 3. | NO extra files |  |  |
| **Subtotal** | |  | **/3** |
| **Movie Management System using JDBC** | | | |
|  | **Movie Management System using JDBC** |  |  |
| 4. | Uses driver to connect to database |  |  |
| 5. | Compiles and runs |  |  |
| 6. | Runnable JAR file |  |  |
| 7. | Properly generated documentation |  |  |
| 8. | Clear instructions |  |  |
| 9. | Adds new movie. |  |  |
| 10. | Retrieves movies released in a year. |  |  |
| 11. | Deletes movie |  |  |
| 12. | Retrieves random movies. |  |  |
| 13. | Handles exceptions appropriately. |  |  |
| 14. | Determines total movie duration |  |  |
| 15. | Uses statements to accept user input. |  |  |
| **Subtotal** | |  | **/38** |
| **Total** | |  | **/41** |