**CSE 3241 Project Checkpoint 05**

**SQL and More SQL**

Names Date

**Submitted to the Carmen Dropbox**

1. Provide a current version of your ER Diagram and Relational Model as per Project Checkpoint 04. **If you were instructed to change the model for Project Checkpoint 04, make sure you use the revised versions of your models.** You must highlight and indicate the corrections/modifications.
2. Use the database populated with the data provided for Checkpoint 4 (with at least 20 sample records for each table and including data provided in the original project documents).
3. For Project Checkpoint 04, you were asked to come up with a set of queries and three additional interesting queries that your database can provide, outlined in Worksheet 02. If you were instructed to fix the queries in Checkpoint 04, make sure you include the fixed queries here. These queries should be provided in a plain text file named “ExtraQueries.txt”. Include also an updated version of the “Views.txt” file if you were instructed to fix them.
4. Given your relational schema, provide the SQL for the following more advanced queries. These queries may require you to use techniques such as nesting, aggregation using having clauses, and other techniques. If your database schema does not contain the information to answer to these queries, revise your ER Model and your relational schema to contain the appropriate information for these queries. **Note that if your database does contain the information but in non-aggregated form, you should NOT revise your model but instead figure out how to aggregate it for the query!** These queries should be provided in a plain text file named “AdvancedQueries.txt”.
   1. Provide a list of patron names, along with the total combined running time of all the movies they have checked out.
   2. Provide a list of patron names and email addresses for patrons who have checked out more albums than the average patron.
   3. Provide a list of the movies in the database and associated total copies lent to patrons, sorted from the movie that has been lent the most to the movies that has been lent the least.
   4. Provide a list of the albums in the database and associated totals for copies checked out to customers, sorted from the ones that have been checked out the highest amount to the ones checked out the lowest.

* 1. Find the most popular actor in the database (i.e. the one who has had the most lent movies)
  2. Find the most listened to artist in the database (use the running time of the album and number of times the album has been lent out to calculate)
  3. Provide a list of customer information for patrons who have checked out anything by the most watched actors in the database.
  4. Provide a list of artists who authored the albums checked out by customers who have checked out more albums than the average customer.

1. Develop a Java application for the Library, that integrates with the Media database your are implementing, and provides the functionality described below. Use as starting point the application developed for Checkpoint #3 and follow the instructions for importing the SQL library API in to your java project (the Embedded\_SQL lab will help to practice the connection to a SQLite database).

* The Java app will present a staff user with a main menu to select one of the following options:
  1. **Search**
  2. **Add new records**
  3. **Order items**
  4. **Edit records**
  5. **Useful reports**
* Each option in the main menu will provide the following options:
  1. **Search**
     1. Artist
     2. Tracks (optional)
  2. **Add new records**
     1. Add an Artist
     2. Add an AudioBook (optional)
  3. **Order items**
     1. Order a Movie
     2. Activate item received
  4. **Edit records**
     1. Edit an Artist
  5. **Useful reports**
     1. Tracks by ARTIST released before YEAR
     2. Number of albums checked out by a single patron
     3. Most popular actor in the database
     4. Most listened to artist in the database
     5. Patron who has checked out the most videos

Notes about the options:

**Search:**

* 1a and 1b. The user provides an artist, the program retrieves the information available in the database
* 1b. Optional. The user provides a track name to search information about (this one and any other search criteria are optional to be implemented)

**Add new records:**

* 2a and 2b. The user provides all the info needed to enter a new artist or a new Audiobook into the database
* 2b. Optional. The user provides all the info needed to enter a new Audiobook into the database. (this one and any other search criteria are optional to be implemented).

**Order items:**

* 3a. The user enter the information to order a new movie
* 3b. Activate item received: when an ordered item arrives, the Library transfer this item from ordered items to the current circulation database, activating the item. The item must be marked (or remove according to your design) from the ordered items and must be inserted (or activated according to your design in the current media database), so patrons can search and checkout the item.

**Edit records**

* 4a. The staff user selects an artist (provide the name), edit any field of the artist and then save it, updating the database.

**Useful reports:**

* 5a. This query comes from [Checkpoint #4]: Find the titles of all tracks by ARTIST released before YEAR.
* 5b. [Checkpoint #4]: Find the total number of albums checked out by a single patron (user designates the patron)
* 5c. [Checkpoint #5]: Find the most popular actor in the database (i.e. the one who has had the most lent movies)
* 5d. [Checkpoint #5]: Find the most listened to artist in the database (use the running time of the album and number of times the album has been lent out to calculate)
* 5e. [Checkpoint #4]: Find the patron who has checked out the most videos and the total number of videos they have checked out.

**SQLite JDBC library for Java and development IDE**

* You are going to use eclipse (IDE).
* You will need the SQLite JDBC library. Download SQLite JDBC Driver and connect to the SQLite database via JDBC. Go to Carmen and download the sqlite-jdbc-3.32.3.2.jar

Add the jar file to your Java project. To add the .jar file to your eclipse project, right-click on the project name -> select build path -> select external archives -> navigate to the .jar file folder and select it.

*Note*: *For more general information about Installing the Java Database Connectivity (JDBC) API on different DBMS read instructions at:*

* [*https://www.codejava.net/java-se/jdbc/jdbc-driver-library-download*](https://www.codejava.net/java-se/jdbc/jdbc-driver-library-download) (links for different DBMS drivers)
* <https://docs.oracle.com/javase/tutorial/jdbc/basics/gettingstarted.html>.

.

Be sure to include in your program the best practices to prevent SQL injection, use prepared statements, sanitize input. Make improvements to the initial program according to best practices. (Name the text file with all the queries: “QueriesJavaApp.txt”. Include the file in the submission).

You need to submit:

* + The Java file with the Java project, database and any other file required in your implementation.
  + Run the program, test it and Include screen outputs for each one of the menu options

Once you have completed all of the questions for parts 1 - 5, create a ZIP archive containing the binary SQLite file and the text files. For part 5, export your Java project to a zip file and submit both, appropriately naming the zip files, to the Carmen Dropbox. Include screenshots of the output for each query.

**Make sure your queries work against your database and provide your expected output before you submit them! The Java application must compile, run, and behave as expected.**

No syntax or running errors should exist in the submission. If the grader cannot construct and populate the database with your code, nor compile/execute your application, then he will be unable to give you feedback.

1. Each team member, individually, needs to fill out the Peer-evaluation form provided and submit it to Carmen.

Please DO NOT zip the report file when you submit so that the grader can give you detailed feedback in Carmen.