CMPUT 291 - File and Database Management (Winter 2022)

Assignment 2 Spec

CMPUT 291 - Introduction to File and Database Management
Winter 2022

Assignment Two

(individual assignment)

Due date: Feb *11 at 5pm* (submission details)

Clarifications:

No clarification will be posted here after 5pm on Feb 10th.

- Feb 10: The number of customers in Q9 refers to the number of different people, meaning each customer is not counted more than once.
- Feb 6: In Q4, there can be two rows in the output for a month (if, for example, the query is issued in the middle of the month and a customer has watched movies in the second part of the month last year and the first part of the month this year).
- Feb 5: Here is a marking rubric.

Introduction

The goal of this assignment is to improve your skills of writing declarative queries on a relational database, in general, and also to improve your fluency in SQL (and SQLite).

You have been provided with the following relational schema.

- moviePeople(pid, name, birthYear)
- movies(mid, title, year, runtime)
- casts(mid, pid, role)
- recommendations(watched, recommended, score)
- customers(cid, name)
- sessions(sid, cid, sdate, duration)
- watch(sid, cid, mid, duration)
- follows(cid, pid)

The tables are derived from the specification of Assignment 1 and the names of the tables and columns should give the semantics, except minor differences which are explicit in table definitions, insert statements or queries. In particular, the badge names are unique and some tables and attributes may have been either removed or renamed for simplicity.

Creating the database

Using the SQL statements provided, create the above tables in SQLite3 on Lab machines with some data. Here is a small initial data to get you started (right click to save both files on your local machine).

(90 marks) Queries

Write down the following queries in SQL and run them in SQLite3 over the database created. You will be writing ONE SQL statement for every query (here One SQL statement starts with a SELECT and ends with a semicolon but may include multiple select statements combined in the form of subqueries and/or using set operations). Your SQL queries for questions 1-3 cannot use any of aggregation, grouping, or nesting (set operations are ok).

1. For each customer, find all recommended movies based on the movies they have watched. For this and all other queries in this assignment, a movie is considered watched if the customer has watched at least 50% of it based on the duration that is recorded. Return the customer id and the id of recommended movies and order the result based on the recommendation scores with the highest score shown first.
2. Find the (id and name of) cast members of The Shawshank Redemption who have acted in at least two other movies. The matches should be case insensitive. *Hint:* Check out built-in scalar functions in SQLite.
3. Of the movies produced this year, which ones are recommended for people who have watched a movie acted by Morgan Freeman. For example, if m1 is movie acted by Morgan Freeman and m2 is recommended for people who have watched m1, we want to return

the id and title of m2 if it is produced this year. The matches again should be case insensitive. *Hint:* Check out Date and Time functions in SQLite.

4. For every customer and every month within the past 365 days, find the number of movies watched within the month. A movie is within a month if the session in which the movie is watched starts within the month. The result includes the id and name of the customer and the month for which the quantity is reported.
5. Find the (id and name of) customers whose total session time is more than 20min but every one of their session is less then 5min.
6. For every customer, find the id of the customer, the number of sessions, the total duration of the sessions, and the number of movies watched. If a customer has no sessions, the customer will be reported with those quantities set to zero. *Hint*: you may find outer join and subqueries in the from clause useful.
7. For each cast member, find the id and the name of the cast member and the number of customers who follow them and have watched all movies by that cast member. Order the results based on the number of such customers.
8. Find the (id and title of) movies with at least 3 cast members and all their cast members having at least two followers.
9. Create a view called *hotMovies* with columns mid, title, year, runtime, watchCnt, recommended, and score. The view includes for every movie that is watched by more than 5 customers, the movie id, title, year, the number of people who watched it, and the set of all recommended movies with their scores. If a movie has no recommended movies, it will appear in the result with null values in the last two fields.
10. Using the view created in Q9, find for each customer who have watched at least two hot movies, the id and name of the customer and all other recommended movies concatenated and separated by commas. *Hint:* Check out built-in aggregate functions in SQLite.

(upto 5 bonus marks for the first 3 people) Preparing test data

Written queries should be tested for correctness and bug fixes, very much like programs written in any programming language. For testing, you need to have enough data in your tables such that all your queries are meaningful and non-trivial (e.g. the returned answers are not empty). You are encouraged to share your data with your classmates or use data prepared by them. *To make this collaboration happen, there will be up to 5 bonus marks (at the instructor's discretion) to the first 3 people who prepare a test data and share it with the rest of the class. To quality, your data must be correct and meets the expectation of the assignment (i.e., the returned answer for every question is not empty and enough cases are tested). If you are sharing your test data, please post it to the course discussion forum.* Put all your *insert* statements in a file called *a2-data.sql*. Make sure to put down your name, email and a date when it is published or revised at the beginning of the file as a comment line (e.g. -- Data prepared by <firstname lastname>, <email address>, and published on <date>). If you are using data prepared by someone else, leave the identification line unchanged.

(10 marks) Testing and report

Starting from scratch, create your database as

```
sqlite3   a2.db   <a2-tables.sql
```

and populate your tables using data file a2-data.sql (prepared in the previous step) as

```
sqlite3   a2.db   <a2-data.sql
```

Put all your SQL queries in a file named a2-queries.sql; Add the following line at the beginning of the file

```
.echo on
```

and the following line before each SQL query (replacing X with the query number).

```
--Question X
```

Run your queries on your data file as

```
sqlite3   a2.db   <a2-queries.sql >a2-script.txt
```

You will be submitting both a2-data.sql and a2-script.txt electronically as described in the instructions for submissions.

Instructions for Submissions

We will make use of some automated tools in testing your queries. Thus it is important that you follow the following instructions closely.

1. Your queries will be tested under a TA account with the provided tables. Do not use any table or column names other than those provided.

2. Write each query in a separate file. Your solution must have **one SQL statement for each query**. In other words, you cannot use views or temporary tables unless you are explicitly asked to do so. The first query must be saved in a file named 1.sql, the second query in a file named 2.sql, and so on until the tenth query, which is to be saved in a file called 10.sql (**the names are important!**).

3. The first line of each query file must have the command:

   ```
   .print Question X - CCID
   ```

   where *X* is the number of the query and CC*ID* is your CCID. For example, the first line of the third query file for the user with ccid 'drafiei' will be:

   ```
   .print Question 3 - drafiei
   ```

   The rest of each file must contain the SQL query you are submitting and nothing else.

4. Include with your submission a README.txt file that has your name, ccid, lab section, and the list of people you collaborated with (as much as it is allowed within the course policy) or the line "I declare that I did not collaborate with anyone in this assignment". A submission without a README.txt file or with missing information will lose 5% of the total mark.

5. Bundle all your queries, insert statements (a2-data.sql) and scripts (a2-script.txt) into a single tarfile by executing the Unix command (everything should be on one line):
   ```
   tar -czf a2.tgz README.txt a2-data.sql a2-script.txt 1.sql 2.sql 3.sql 4.sql 5.sql 6.sql 7.sql 8.sql 9.sql 10.sql
   ```
6. Submit the file *a2.tgz* at the submission page after logging into eclass.
   Eclass does not support versioning (unfortunately) and each new submission replaces your previous one. This makes last minute submissions somewhat risky. Avoid last minute submissions as much as you can, and check your submissions after an upload to make sure the right content is uploaded. A common mistake is to use a wrong tar command and submit a corrupt file.

Last modified: Thursday, 10 February 2022, 7:58 AM

◄ Assignment 1 submission

Jump to...

quiz2 ►