**SQL Programming Problems** 

**Overview**

Our problem domain is movie trivia, and the database that we've created contains facts about movies, directors, actors, and actresses. In particular, we've included information on the following:

* the winners of the Academy Awards for Best Picture, Best Director, Best Actress, Best Actor, Best Supporting Actress, and Best Supporting Actor from the Oscars' inception in 1929 to the present, as well as the films for which the acting and directing awards were won
* the 200 top-grossing films of all time (as of September 18, 2015) -- i.e., the films that have made the most money -- as well as some former top-grossers that have since fallen out of the top 200
* the five top-billed cast members of each film included in the database

The source of our data is a great Web site called [imdb.com](http://www.imdb.com/), the Internet Movie Database.

Below are the schema of the five tables in the database. The names of the primary-key attributes are italicized.

The **Movie** relation has the following attributes:

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Data Type** | **Description** |
| *id* | CHAR(7) | a unique id assigned to the film |
| name | VARCHAR(64) | the name of the film |
| year | INTEGER | the year the film was released |
| rating | VARCHAR(5) | the MPAA rating of the film; if the film is unrated or has a non-standard rating, this value is NULL. |
| runtime | INTEGER | the running time of the film in minutes |
| genre | VARCHAR(16) | the genre(s) of the film, formed by concatenating one-letter genre codes (e.g., A for action and D for drama) action (A), adventure (V), animation (N), comedy (C), drama (D), family (F), fantasy (Y), horror (H), mystery (M), musical (L), romance (R), sci-fi (S), thriller (T), war (W), western (E); no assumptions should be made about the order in which these codes appear in a given genre string |
| earnings\_rank | INTEGER | the earnings rank of the film (only included for the top 200; NULL otherwise) |

The **Person** relation has the following attributes:

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Data Type** | **Description** |
| *id* | CHAR(7) | a unique id assigned to the person |
| name | VARCHAR(64) | the person's full name (first name first) |
| dob | DATE | the person's date of birth, if known; NULL otherwise |
| pob | VARCHAR(128) | the person's place of birth, if known; NULL otherwise |

The **Actor** relation has the following attributes:

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Data Type** | **Description** |
| *actor\_id* | CHAR(7) | the id attribute of the actor; a foreign key that references Person(id) |
| *movie\_id* | CHAR(7) | the id attribute of a movie in which the actor appeared; a foreign key that references Movie(id) |

The **Director** relation has the following attributes:

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Data Type** | **Description** |
| *director\_id* | CHAR(7) | the id attribute of the director; a foreign key that references Person(id) |
| *movie\_id* | CHAR(7) | the id attribute of a movie that he or she directed; a foreign key that references Movie(id) |

The **Oscar** relation has the following attributes:

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Data Type** | **Description** |
| movie\_id | CHAR(7) | the id attribute of the movie for which the award was won; a foreign key that references Movie(id) |
| person\_id | CHAR(7) | the id attribute of the actor, actress, or director who won the award; if type = 'BEST-PICTURE' (see below), this attribute is NULL; a foreign key that references Person(id) |
| type | VARCHAR(23) | the type of Oscar, which will be one of the following strings: 'BEST-PICTURE', 'BEST-DIRECTOR', 'BEST-ACTRESS', 'BEST-ACTOR', 'BEST-SUPPORTING-ACTRESS', 'BEST-SUPPORTING-ACTOR' |
| year | INTEGER | the year in which the Oscar was won |

(Note: We chose not to specify a primary key for the Oscar relation. One reason for our decision is that you need the combination of all four attributes in order to guarantee uniqueness. In addition, one of those four attributes (person\_id) has NULL values in some tuples, and NULL values are not allowed in primary-key attributes. Instead of specifying a primary key, we used a UNIQUE clause to specify that the combination of the four attributes must be unique.)

For example, Tom Hanks won the Best Actor Oscar in 1994 for his performance in the movie *Philadelphia*, which was directed by Jonathan Demme. If Hanks had an id of 1234567, Demme an id of 9876543, and the movie an id of 2222222, then the following tuples would appear in the database:

* (1234567, 2222222) in Actor
* (9876543, 2222222) in Director
* (2222222, 1234567, 'BEST-ACTOR', 1994) in Oscar

along with a tuple for the movie itself in Movie, and tuples for Hanks and Demme in Person.

**Getting Started**

In your work on this part of the assignment, you will be using SQLite, a lightweight, open-source RDBMS. More precisely, SQLite is a database library that other applications can use to manage a relational database on their behalf.

We will be using an application called SQLite Manager, which is an add-on for Firefox. It provides a user-friendly way to work with a relational database that is managed by SQLite.

**Downloading the necessary software**

1. If you don't already have Firefox installed on your machine, you should [download and install it](http://www.mozilla.org/en-US/firefox/new/) now.
2. Once you have Firefox installed, start it up and **use Firefox** to go to the following page: <https://addons.mozilla.org/en-us/firefox/addon/sqlite-manager/>.
3. Click the green *Add to Firefox* button to add SQLite Manager to Firefox. If asked, click the *Install Now* button.
4. Close and relaunch Firefox to allow the add-on to take effect.

**Downloading the necessary files**

1. Download the movie database by right-clicking on the following link: [movie.sqlite](http://sites.fas.harvard.edu/~cscie66/assignments/ps1/movie.sqlite)   
   Use the *Save link as...* (or equivalent) option to save it somewhere that you will be able to find it later. This file contains all of the tables from the movie database. However, because it isn't a text file, you will *not* be able to read it using a text editor. Instead, SQLite Manager will process it for you, according to the SQL commands that you enter.
2. Download the following template for your answers: [ps1\_queries.txt](http://sites.fas.harvard.edu/~cscie66/assignments/ps1/ps1_queries.txt)   
   Use the *Save link as...* (or equivalent) option to save it somewhere that you will be able to find it later.

**Using SQLite Manager**

1. Start up Firefox if it isn't already running.
2. Click the Firefox menu button and choose *SQLite Manager*, which should open up SQLiteManager a separate window.
   * If you don't see *SQLite Manager* in the menu, click *Customize* and find *SQLite Manager* under *Additional Tools and Features*. Drag the *SQLite Manager* icon over to an open slot in the menu, and then click *Exit Customize*.
   * In older versions of Firefox, you may need to choose *Web Developer->SQLite Manager*.
3. In the SQLite Manager window, use the *Database->Connect Database* menu option to find and open the *movie.sqlite* database file that you downloaded above.
4. To explore the contents of the database, click on the name of a table in the left-hand navigation bar, and then click on either the *Structure* or *Browse & Search* tabs.   
   **Note:** SQLite adds a *rowid* column to each table. You should ignore this column for the purposes of this assignment.
5. Use the *Execute SQL* tab to perform queries on the database, replacing the sample command that is provided (if any) with your own SQL command, and clicking the*Run SQL* button.

**Important:** If you are using TextEdit on a Mac to write your queries, and you cut and paste a query from TextEdit into SQLite Manager, you need to make sure that you change any curly quotes into straight quotes. Otherwise, SQLite Manager will give you an error. To fix the problem, simply delete the curly quotes in SQLite Manager and retype them as straight quotes.

**Guidelines:**

* Put your queries in the *p1\_queries.txt* file that you downloaded above. While we encourage you to come up with your answers through interactive sessions with the database, your queries must ultimately be included in this file.
* Use a text editor to edit the file, and make sure that you keep the file as a [plain-text](http://sites.fas.harvard.edu/~cscie66/assignments/plain_text.html) file.
* Unless otherwise stated, each of the problems must be solved by means of **a single query** (i.e., each query should have a single semi-colon). Use nested subqueries as needed.
* Your queries should *not* use a *LIMIT* clause.
* Your queries should employ the standard version of SQL found in the lecture notes. They should *not* rely on special SQLite features such as its ability to handle combinations of regular attributes and aggregate operators in the SELECT clause.
* Make sure that the results produced by your queries contain exact answers to the problems. We should not have to infer the answer from the results. For example, if we ask you how many films meet a given criterion, we want a number, not a list of the films. In addition, your results should not include any extraneous information.
* You should **not** include the results of the queries, but only the queries themselves.
* Make sure that you actually enter your answers in SQLite Manager to ensure that they work!

1. Cate Blanchett has been nominated for a Best Actress Oscar for her performance in *Carol*. If she wins, it will be her third Oscar. Write a query to find the names of the movies for which she won her earlier Oscars, the type of award that she won, and the year of each of those awards. The result of your query should be tuples of the form (movie name, award type, year won).
2. Leonardo DiCaprio and Matt Damon are two of this year's nominees for Best Actor. Both have been nominated for this award in the past, but neither of them has won it yet (although Damon did win an Oscar for Best Screenplay). Write a single query to find the places of birth and dates of birth for these two actors. The result of the query should be tuples of the form (name of person, place of birth, date of birth). *Hint:* If your initial query does not produce any results, you may want to reconsider the logical operator (AND, OR, NOT) that you are using in your WHERE clause.
3. Write a query that finds the names of all foreign-born directors (i.e., directors born outside the US) who have directed films in which a foreign-born actor or actress won an Oscar. You may assume that all US-born persons have the word 'USA' at the end of their place of birth. *Hint:* You may find it helpful for your FROM clause to include multiple instances of at least one of the tables in the database.
4. A number of this year's Best Picture nominees are also nominated for Best Director. Write a query that finds all years in which the same movie won both Best Picture and Best Director. The result of the query should be tuples of the form (year awarded, movie). *Hint:* You may find it helpful for your FROM clause to include multiple instances of at least one of the tables in the database.
5. Write a query that determines the number of movies in the database that are more than 200 minutes long. The result of the query should be a single number.
6. Julia Louis-Dreyfus won an Emmy last year for her role in the HBO series *Veep*. Write a query to determine the names and dates of birth of all people in the database whose first name is Julia. Use pattern-matching as needed, and make sure that you only request people whose first name is exactly 'Julia'. *Hint:* Think about how to construct a pattern that obtains people whose first name is 'Julia' without getting people whose first name begins with 'Julia' (e.g., Julianne) or people who have 'Julia' as all or part of their last name.
7. Write a query that determines the names of the movies in the database that are shorter than all of the movies that have been named Best Picture? *Hint:* You will need a subquery.
8. Which directors have been most successful at the box office? Write a query that finds all directors who have directed **at least three** of the 200 top-grossing films. The result of the query should consist of tuples of the form (director, number of top grossers). List them in order of the number of top-grossers they have directed, from the highest number to the lowest. You may assume that all directors in the database have a unique name.
9. Does the Academy reward top-grossing films? Write a query that summarizes the Oscars won by each of the top 25 grossing films. The results of your query should contain tuples of the form (earnings rank, movie name, award type). If a given movie won multiple awards, it should have one tuple for each award; if it won no awards, it should have a single tuple in which the third value of the tuple is NULL. Sort the tuples by earnings rank.
10. Write a query that determines the number of actors and actresses in the database who have *not* acted in any of 200 top-grossing movies. The result of the query should be a single number, and the count should *not* include people who are only in the database as a director.
11. Clint Eastwood appears in the database as both an actor and a director. Write a query that lists all of the movies in the database with which he is associated and the function(s) that he played in each case. If he served as both an actor and a director for a given film, you should have two entries for that film in the results -- one for actor and one for director. Sort the tuples so that multiple entries for a given movie will appear together. The comparable results table for Woody Allen looks like this:
12. +--------------------------+----------+
13. | movie | function |
14. +--------------------------+----------+
15. | Annie Hall | actor |
16. | Annie Hall | director |
17. | Blue Jasmine | director |
18. | Bullets Over Broadway | director |
19. | Hannah and Her Sisters | director |
20. | Mighty Aphrodite | director |
21. | Vicky Cristina Barcelona | director |
22. +--------------------------+----------+

(Note: Allen also acted in some of the other films, but the database doesn't include that information because he wasn't among the five top-billed cast members in those films.)

*Hints:*

* + You can create tuples that include a constant-valued attribute by including the constant value in the SELECT clause. For example: SELECT Foo.bar, 'hello' ... will produce tuples whose second value is always 'hello'.
  + You may find it helpful to use one of the set operations available in SQL.
  + Use aliases to obtain the correct column headings.

1. Write a query to determine the last (i.e., most recent) time that a G-rated movie won Best Picture. Your answer should be a single tuple consisting of the name of the movie and the year in which it won Best Picture.

*Hints:*

* + You will need to use a subquery.
  + "most recent" is equivalent to having the largest year.

1. Elisabeth Moss is the actress who played Peggy Olson in the AMC series *Mad Men*. She also appears in our database because of her work in the movie *Girl, Interrupted*. When Moss was added to the database, her place of birth was not available on IMDb, and her date of birth was listed incorrectly. Write a single SQL command that changes her place of birth to 'Los Angeles, California, USA' and her date of birth to '1982-7-24'. You may assume that there is only one person named Elisabeth Moss in the database. *Hint:* You should *not* perform an INSERT command, because there is already a tuple for Moss in the database. Instead, you need a command that allows you to modify the appropriate attributes in that tuple.
2. **Grad-credit problem** (*required of grad-credit students; may be completed by other students for "partial" extra credit*)  
   Assume that you can measure the success of an actor or actress by his or her *movie count* - the total number of movies from this database in which he or she is listed as an actor. Is it true that winners of the Best Supporting Actress Oscar have less successful careers than winners of the other three acting Oscars? Write a query that computes the average movie counts for winners of each type of acting Oscar. Your results should have four tuples of the form (Oscar type, avg. movie count for winners of that Oscar type).

If a person wins the same type of Oscar twice, his or her movie count should only contribute *once* to the average for that type of Oscar. If a person wins multiple types of Oscars, his or her movie count should contribute to the average for each type.

*Hint*: You may find it helpful to use at least one subquery in the FROM clause.

1. **Grad-credit problem** (*required of grad-credit students; may be completed by other students for "partial" extra credit*)  
   Find all actors or actresses in the database who have a Bacon number of 2--i.e., they haven't been in a movie with Kevin Bacon, but they've been in a movie with someone who has. (For more on this movie trivia game, see [http://oracleofbacon.org](http://oracleofbacon.org/).) For this question only, you may use multiple queries and store the results of the earlier queries in temporary tables so that they can be used by the subsequent queries. To store the results of a query in a temporary table, precede the SELECT command with the line
2. CREATE TEMPORARY TABLE *tablename* AS

For example:

CREATE TEMPORARY TABLE BigRooms AS

SELECT name, capacity

FROM Room

WHERE capacity > 200;

These temporary tables will be dropped automatically at the end of the current SQLite Manager session.