CMPUT 291 - File and Database Management (Fall 2019)

Assignment 2 Spec

CMPUT 291 - Introduction to File and Database Management
Fall 2019

Assignment Two

(individual assignment)

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

Clarifications:

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

- **Oct 7**. Here is a marking rubric for the Assignment 2.
- **Oct 7**. Q5 should read "..., i.e. the sum of their demerit points within the past two years is at least 15"
- **Oct 4**. In Q10, the text "...one of those tickets involves..." should be interpreted as "...at least one of those tickets involves..."
- **Oct 2**. Q2 is asking for people who are born to either the same father or the same mother as Michael Fox.
- **Oct 3**. In Q4, Michael Fox is treated as the father.

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.

- *persons(fname, lname, bdate, bplace, address, phone)*
- *births(regno, fname, lname, regdate, regplace, gender, f_fname, f_lname, m_fname, m_lname)*
- marriages (regno, regdate, regplace, p1_fname, p1_lname, p2_fname, p2_lname)
- vehicles(vin,make,model,year,color)
- registrations(regno, regdate, expiry, plate, vin, fname, lname)
- tickets(tno,regno,fine,violation,vdate)
- demeritNotices(ddate, fname, lname, points, desc)

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.

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.

(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). When the query asks for name of a person, it means both first name and last name.

1. Find the names and the phone numbers of persons who have had a 1969 Chevrolet Camaro registered under their names at some point.
2. Find the names of people other than Michael Fox who are born to the same parent as Michael Fox.
3. Find the names of persons who have the same grandfather as Michael Fox. If X and Y denote the grandfathers of Michael Fox (from mother and father sides respectively), then we want to find all grand children of X and Y. The result should exclude Michael Fox.
4. Who is the oldest child of Michael Fox. In case of ties, return all those ties.
5. Find the names of persons who have accumulated 15 or more demerit points within the past two years, i.e. the sum of their demerit points within the past two years is more than 15. *Hint*: Check out the date and time functions in SQLite.
6. Who is the partner of Michael Fox. In case of multiple marriages, return the one from the latest marriage.*Hint*: Check out the *limit* clause for sqlite. You may also find subqueries in the from clause useful.
7. For each color of a car with a registration that does not expire at least for another month, find the average number of tickets issued per registration, the average amount of fine given, and the maximum amount of fine given. Include colors with no tickets in the output with zero counts (if applicable) or null values. *Hint*: you may find outer join useful.
8. For each year of a car, find the most frequent make and the most frequent car color. In case of ties, list all those ties.

9. Create a view called *personDetails* with columns fname, lname, bdate, bplace, carsowned, and ticketsRcvd. The view includes for each person, fname, lname, bdate, bplace, the number of different cars registered under the person name in the past year, and the number of different tickets given to those registered cars within the past year. Include people who have no cars registered under their names or no tickets with zero values.
10. Using the view created in Q9, for every person who has received at least 3 different tickets within the past year and one of those tickets involves a 'red light' violation (i.e. 'red light' appears in the violation text, e.g. 'red light crossing', 'crossing red light at 114 St and 87 Ave'), list the name of the person and the make and the model of the car for which the red violation ticket is given.

(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. Make sure your data is correct and meets the expectation of the assignment. 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
   ```
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: Monday, 7 October 2019, 2:10 PM

◄ Assignment 1 submission Jump to... SQL statements for "Rel. Algebra and SQL" lab ►