**Assignment 2 Spec**

**CMPUT 291 - Introduction to File and Database Management  
Fall 2020**

**Assignment Two**

**(individual assignment)**

**Due date: *Oct 11 at 5pm* (**[**submission details**](https://eclass.srv.ualberta.ca/mod/page/view.php?id=4329135#submission)**)**

**Clarifications:**

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

* **Oct 10.**In Q10, you will be returning 4 quantities for each city. The third quantity is "for each city, the average number of questions posted within the past month by users of the city." For example, if Edmonton has 100 users and the total number of questions posted by users from Edmonton within the past month is 200, then this average will be 2.
* **Oct 9.**[Here](https://eclass.srv.ualberta.ca/pluginfile.php/6089861/mod_page/content/51/a2-rubric.txt?time=1602308621787) is a marking rubric for this assignment.
* **Oct 7.**In Q8, the numbers will be zero if a user does not have any question, answer, vote, etc.
* **Oct 3.**In Q6, "the number of votes" for a tag refers to the total number of votes on all posts marked with that tag.
* **Oct 1**. In Q3, the question may or may not have answers after 3 days of its posting. As long as it is not answered within 3 days after its posting, the question should be included in the output.
* **Oct 1**. Unless explicitly asked otherwise (e.g. Q2), all string matches are case-sensitive.
* **Oct 1**. In Q7, "the number of posts" for a given day and tag refers to the number of posts that are on that day and have the tag. Also if none of the posts in a day use any tag, that day will not be returned.

**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.

* *users(uid, name, city, crdate)*
* *badges(bname, type)*
* ubadges(uid, bdate, bname)
* posts(pid, pdate, title, body, poster)
* tags (pid, tag)
* votes(pid, vno, vdate, uid)
* questions(pid, theaid)
* answers(pid, qid)

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](https://eclass.srv.ualberta.ca/pluginfile.php/6089861/mod_page/content/51/a2-tables.sql?time=1601226775142), create the above tables in SQLite3 on Lab machines with some data. Here is [a small initial data](https://eclass.srv.ualberta.ca/pluginfile.php/6089861/mod_page/content/51/a2-initial-data.sql?time=1601228442978) 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. Find the id of users with a gold badge and a question post.
2. Find the id and title of questions that have the text "relational database" either in title or both terms "relational" and "database" appear in tags. The matches should be case insensitive. *Hint*: Check out [built-in scalar functions in SQLite](https://sqlite.org/lang_corefunc.html).
3. Find the id and title of questions that are not answered within three days after their postings. *Hint:* Check out Date and Time functions in SQLite.
4. Find the id of users who have answered more than two of their own questions.
5. Find the id of those users who have posted both questions and answers and their posts have received more than 4 votes overall.
6. Find top three tags where posts marked with those tags receive the largest number of votes. For each such tag, report the number of posts and the total number of votes on those posts. *Hint:* Check out the *limit* clause in SQLite.
7. For each day of a post, find the most frequently used tag for that day (in case of ties, return them all). The result will include for each day and each tag used in that day, the number of posts.
8. For each user, report the user id, the number of questions, the number of answers, the total number of votes casted, and the total number of votes received (on all their posts). Report every user who has a non-zero count in at least one category. *Hint*: you may find outer join and subqueries in the from clause useful.
9. Create a view called *questionInfo* with columns pid, uid, theaid, voteCnt, and ansCnt. The view includes for each question posted within the past month, the question id, the id of the user who posted it, the accepted answer id (or null if none), the number of votes (zero if no votes), and the number of answers (zero if no answers).
10. Using the view created in Q9, for each user city, find the number of users, the total number of gold badges given to the users of the location, the average number of questions per user of the location within the past month, and the total number of votes received by posts made within the past month by users from that location.

**(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* - CC*ID*

where *X* is the number of the query and CCI*D* 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.

1. 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.
2. 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

1. Submit the file *a2.tgz* at the [submission page](https://eclass.srv.ualberta.ca/mod/assign/view.php?id=4617959) 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.