CSCC43 UTSC Summer 2021

Assignment 2

Interactive & Embedded SQL Queries

Due Date: 12 July, 11:59pm

Read the note on academic Integrity.

Academic Integrity: https://utsc.utoronto.ca/aacc/academic-integrity-matters

Instructions:

1. Read this assignment thoroughly before you proceed. Failure to follow instructions can affect your grade.
2. Download Assignment2 files that includes:
    a. Database schema files: a2.ddl, a2.sql
    b. Java skeleton file: Assignment2.java
3. Submit your work electronically using MarkUs. Your submission must include the following files:
    a. a2.sql
    your queries for the interactive SQL part of the assignment (can include any view creation statement). If you define any views for a question, you must drop those after you have populated the answer table for that question.
    b. Assignment2.java
    your java code for the embedded SQL part of the assignment. Be careful to submit the .java file (not the .class file.). To get you started, we provide a skeleton of this file that you must download from the assignment webpage.

For this assignment, you can work in a group of 2 to 3(max) students.

Note: Late submissions will not be accepted/marked.

SQL Queries [50 marks]

In this section, you must edit the file a2.sql and add SQL statements that can be **run in psql on the MathLab machine**. Your SQL statements can create views and queries that will populate the result tables *query1, query2, …, query7* with tuples that satisfy the questions below. In order to ensure that everything is run in the correct order (by the markers or the automaker), you should add all your SQL statements in the file a2.sql that we have provided.

You can assume that the a2.ddl file has been read and executed in psql, before your a2.sql file is executed.

Follow these rules:

- The output of each query must be stored in a result table in file **a2.sql**. We provide the definitions/attributes of these tables in next section (*Query1, Query2, …, Query7*).
- For each of the queries below, your final statement should populate the respective answer table (QueryX) with the correct tuples. It should look something like:

  "INSERT INTO queryX (SELECT … …)"

  where X is the correct index [1, …,7].
- To answer each of the questions, you are encouraged to create virtual **views** that can keep intermediate results and can be used to build your final INSERT INTO QueryX statement. Do not create actual tables. Remember that you have to drop the views you have created, after each INSERT INTO QueryX statement (i.e., after you have populated the result table).
- Your tables **must** match the output tables specified for each query. The attribute names must be **identical** to those specified in italics, and they must be in the specified order. Also, make sure to sort the results according to the attributes and ordering we specify in each question.
- We are not providing a sample database to test your answers, but you are encouraged to create one. We will test the validity of your queries against our own test database.
- All of your statements must run on PostgreSQL on the MathLab machine, so be sure to populate your tables with test data and run all your statements on MathLab prior to submission.

NOTE: Failure to do follow the instructions may cause your queries to fail if (automatically) tested, and you will lose marks.

Express the following queries in SQL.

1. [5 marks] Find the department with the highest number of instructors who do not have a PhD degree.
   If there is a tie, return all tied departments.
   Output Table: **Query1**
   attributes: *dname* (the department name)

2. [5 marks] Find the total number of female students in the fourth year who are in 'Computer Science' department.
   Output Table: **Query2**
   attributes: *num* (the number of students)

3. [5 marks] Course enrollment for a year is calculated as the sum of the total number of students enrolled in each course for all courses in all semesters for that year. Find the year between 2016 and 2020 in which the “Computer Science” department had the highest course enrollment, compared to other years.
   (Hint: Use the calendar year; do not worry about starting a year in the Fall.)
   Output Table: **Query3**
   attributes: *year* (the year with highest enrollment)
   enrollment (the course enrollment for the year)

4. [5 marks] Find all the courses in 'Computer Science' department that are taught only in summer semester. Do not report duplicates.
   Output Table: **Query4**
   attributes: *cname* (the name of the course)

5. [10 marks]
   Determine excellent student for each Department.
   To find such student, compute the rank for that student determined by the average grade of all of the courses they have completed. If there is a course taken by student in current semester, do not include that course.
   Hint: The current semester occurs in the largest year/term value.
   If there is a tie for first place, include all students in the tie.
   Output Table: **Query5**
   attributes: *dept* (the name of the department),
   sid (the student id),
   sfirstname (student first name),
   slastname (student last name),
   avgGrade (average grade for the student)

6. [10 marks] Suppose a student has taken a course without previously having taken the prerequisites. List all such students and courses. Report a student as many times as the number of courses with no prerequisites that they have taken (but not for each prerequisite).
   Output Table: **Query6**
   attributes: *fname* (student first name),
   lname (student last name),
   cname (course name),
   year (the year)
   semester (the semester),

7. [10 marks] Find out the highest and the lowest average marks for those courses with enrollment of at least 3 students in the 'Computer Science' department
   If there is a tie, include all tied results.

Hint: Do not need to consider the current semester which does not yet have any marks.
Output Table: **Query7**
attributes: *cname* (name of the course)
semester (the semester),
year (the year),
avgmark (the average mark for that course)

JDBC and SQL Queries [50 marks – 5 for each method]

For this part of the assignment, you will create the class Assignment2.java which will allow you to process queries using JDBC. We will use the standard tables provided in the a2.ddl for this assignment. If you feel you need an intermediate view to execute a query in a method, you must create it in that method. You must also drop it before exiting that method.

Rules:

- Standard input and output must not be used.
- The database, username, and password must be passed as parameters, never "hard-coded". We will use your connectDB() method defined below to connect to the database with our own credentials.
- Be sure to close all unused statements and result sets.
- All return values will be String, boolean or int values.
- A successful action (Update, Delete) is when:
    - It doesn't throw an SQL exception, and
    - The number of rows to be updated or deleted is correct.

Class name	Description
Assignment2.java	Allows several interactions with a postgreSQL database

Instance Variables (you may want to add more)

Type	Description
`Connection`	The database connection for this session.

Methods (you may want to add helper methods.)

Constructor	Description
`Assignment2()`	Identifies the postgreSQL driver using Class.forName method.

Method	Description
`boolean connectDB(String URL, String username, String password)`	Using the String input parameters which are the URL, username, and password respectively, establish the `Connection` to be used for this session. Returns **`true`** if the connection was successful.
`boolean disconnectDB()`	Closes the connection. Returns true if the closure was successful.
`boolean insertStudent(int sid, String lastName, String firstName, String sex, int age, String dcode, int yearOfStudy)`	Inserts a row into the student table. *dcode* is the code of the department. You must check if the department exists, if the sex is one of the two values ('M' or 'F') and if the year of study is a valid number (>0 && < 6). Returns true if the insertion was successful, false otherwise.
`int getStudentsCount(String dname)`	Returns the number of students in department *dname*. Returns -1 if an error occurs.
`String getStudentInfo(int sid)`	Returns a string with student information of student with student id sid. The output is "firstName:lastName:sex:age:yearOfStudy:department". Returns an empty string "" if the student does not exist.
`boolean chgDept(String dcode, String newName)`	Changes the department name to the department name supplied (*newName*). Accepts the *dcode* and new department name as `Strings` (in that order). Returns **`true`** if the change was successful, **`false`** otherwise.
`boolean deleteDept(String)`	Deletes the department identified by the input String *dcode*. Returns **`true`** if the deletion was successful, **`false`** otherwise.
`String listCourses(int sid)`	Returns a string with all the courses a student with student id *sid* has taken. Each course will be in a separate line, e.g. "courseName1:department:semester:year:grade# courseName2:department:semester:year:grade# ... " Returns an empty string "" if the student does not exist.
`boolean updateGrades(int csid)`	Increases the grades of all the students who took a course in the course section identified by *csid* by 10% :) Returns **true** if the update was successful, **false** otherwise. Do not not allow marks to go over 100%

`boolean updateDB()`	Create a table containing all the female students in "Computer Science" department who are in their fourth year of study. The name of the table is *femaleStudents* and the attibutes are: sid INTEGER (student id) fname CHAR (20) (first name) lname CHAR (20) (last name) Returns **true** if the database was successfully updated, **false** otherwise