CSC343 Fall 2012
Assignment 2
Interactive & Embedded SQL Queries

Distribution date: Thursday, October 18, 2012
Due date: Thursday, November 1, 2012 11:59 p.m.

Instructions

1. Read this assignment thoroughly before you proceed. Failure to follow an instruction can affect your grade.
2. The database tables you are using are defined in **A2.ddl** which must be downloaded from the assignment webpage.
3. You must hand in your work electronically using the UNIX submit command. You must submit the following 5 files:
   a) **team.txt** information about your team (whether it is a team of one or two students)
   b) **a2drop** drop statements.
   c) **a2tables** creation statements for query result tables.
   d) **a2sql** your queries for this assignment.
   e) **Assignment2.java** your java code. Be careful to submit the .java file, **not** the .class file.

   When you have completed the assignment, move or copy your files in a directory (e.g., assignment2), and use the following command to electronically submit your files within that directory:

   % submit -c csc343h -a A2 team.txt a2drop a2tables a2sql Assignment2.java

   You can also submit the files individually after you complete each part of the assignment – simply execute the submit command and give the filename that you wish to submit. You may submit your solutions as many times as you wish prior to the submission deadline (you might need to use the ***–f*** flag of the submit command). Make sure you name your files exactly as stated (including lower/upper case letters). Failure to do so will result in a mark of 0 being assigned.

   Once you have submitted, be sure to check that you have submitted the correct version of each file; new or missing files will not be accepted after the due date. You may check the status of your submission using the command

   % submit -l -N A2 csc343h

   where -l is a hyphen followed by the letter 'ell'.

Interactive SQL Queries [45 marks]

In this section, you must create views and queries **to be run in *psql* on the CDF machine.** In order to ensure that everything is run in the correct order (by the markers), you must create three files containing your statements, which can be read into *psql* and executed using the *psql* command:
\i <FILENAME>.

The files must be as follows:

1. **a2drop**: statements to drop any and all views and tables that you create.
2. **a2tables:** DDL statements to create the tables which will hold the query results (Query1 through Query7), as well as any intermediate views you may require for this assignment.
3. **a2sql**: all queries that populate the QueryX Tables (e.g., Query1, Query2, etc.) with results that satisfy the questions below.

Express the following queries in SQL. Follow these rules:

- The output of each query must be stored in a new table. You must create the table definition for the tables that are used to store the results of your queries (e.g., Query1, Query2, etc.). These definitions should be saved in **a2tables**.
- The final statement to insert your query results should look something like:
  "INSERT INTO Query1 (SELECT … <complete your SQL query here> …)"
- 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.
- All of your statements must run on PostgreSQL on the CDF machine, so be sure to populate your tables with test data and run all your statements on CDF prior to submission.

NOTE: Failure to do this may cause your query to fail when (automatically) tested, and you will lose marks.

1. [**5 marks**] Find the number of female students in the 'Computer Science' department who are in their fourth year of study.
   Output Table: **Query1**
   Attributes: *num* (the number of students)

2. [**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: **Query2**
   Attributes: *dname* (the department name)

3. [**5 marks**] For each department find the best student. A student's rank is determined by his or her average grade on all of the courses they have completed. Courses for the current semester should not be included. (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: **Query3**
   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)

4. [**5 marks**] Find the year(s) between 2001 and 2006 in which the 'Computer Science' department had the highest course enrollment, compared to other years. 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. (Use calendar years, not academic years starting in the Fall.)
   Output Table: **Query4**
   Attributes: *year* (the year with highest enrollment)
   enrollment (the course enrollment for the year)

5. [**5 marks**] Find all the courses in 'Computer Science' department that have been taught only in the summer semester. Do not report duplicates.
   Output Table: **Query5**
   Attributes: *cname* (the name of the course)

6. [**10 marks**] List all the students and courses for all situations where a student has enrolled in a course (this includes courses a student is taking now) without having taken all the prerequisites in a preceding semester. Report a student as many times as the number of courses they have enrolled in but are missing at least one prerequisite for (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**] Of all the courses offered in the 'Computer Science' department which had an enrollment of at least 3 students, find the course with the highest average marks and the course with the lowest average marks. Include all tied results. (Hint: You 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)

Embedded SQL Queries [55 marks]

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. This will halt the "automarker" and you will probably end up with a zero.
- The database, username, and password must be passed as parameters, never "hard-coded".
- 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.
- When rows of data are returned as a String, they must be in the following contiguous format:
  - Columns are separated with a colon ":". There is no colon after the last column of a row.
  - Rows are separated with a pound-sign "#". There is no pound-sign after the last row.
  - Leading and trailing spaces are eliminated; e.g., a result set that includes 4 rows of the columns firstname, lastname might look like:

    "fnameA:lnameA#fnamaB:lnameB#fnameC:lnameC#fnameD:lnameD"

Class name	**Description**
Assignment2.java	Allows several interactions with a postgreSQL database. The second assignment for CSC343 Fall 2012.

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, int age,String sex, String dname, int yearOfStudy)	Inserts a row into the student table. dname is the name of the department. You have to 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.

Method	**Description**
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 dcode)	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. The list of courses should follow the contiguous format described above, and contain the following attributes in the order shown: "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%.
String query7()	Execute query 7 (highest and lowest average department marks) described in the Interactive SQL section above. Instead of inserting the results in a table, return them as a String in the same format as is specified for the output table for the query. Be sure to follow the prestated String rules involving colons and pound-signs for your return String. Do not use the views that you created in the SQL part. If you need views, create them when you execute function query7().
boolean updateDB()	Create a table containing all the female students in the "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.