CS34800: Project1
Due Date: 11:59PM, Sunday, March 3, 2019

(There will be a 10% penalty for each late calendar day. After five calendar days, the project will not be accepted.)

Given the following entities, provide the SQL queries corresponding to the questions below:

Note:

1. The schema definition of these tables and sample test data are provided in **tables.sql** and **data.sql**, respectively**.**
2. You should finish all your work in **answer.sql**.
3. We provide **test.sh** script to test your sql queries on sample test data, but we will use different data when grading. Feel free to modify/add your sample test data in order to polish your queries. (You need to verify your answer by yourselves <u>if you change the sample test data</u>. In this case, test.sh may not work.)
4. How to use **test.sh**. First, set the variables "username" and "pword" in the script with your oracle account with "@csora" and your password. Second, execute the **tables.sql** and **data.sql**. Then
    a. "./test.sh" test all 10 queries;
    b. "./test.sh $(Query_number)" test one specific query. For example, "./test.sh 1"
5. Submit your answers via Blackboard.
6. **<u>Do not use PL/SQL for this homework</u>, just a main SQL select statement per question (subqueries, i.e., nested queries, are allowed).**
7. DO NOT delete/change the "-- Query[0-9]*" comment in the **answer.sql** file.
8. Grading: We will use script to grade your projects. There will be **no** partial credit for each query.
9. For those "select top X" problems, assume that there is no tie condition that will influence the result. That is, the values in the column that may affect the result of selecting top X are distinct.

Write SQL queries for questions 1-10 using the following University schema.

Classroom(<u>building</u>, <u>room_number</u>, capacity)
Department(<u>dept_name</u>, building, budget)
Course(<u>course_id</u>, title, dept_name, credits)
Instructor(<u>i_id</u>, name, dept_name, salary)
Section(<u>course_id</u>, <u>sec_id</u>, <u>semester</u>, <u>year</u>, building, room_number, time_slot_id)
Teaches(<u>i_id</u>, <u>course_id</u>, <u>sec_id</u>, <u>semester</u>, <u>year</u>)
Student(<u>s_id</u>, name, dept_name, tot_cred)
Takes(<u>s_id</u>, <u>course_id</u>, <u>sec_id</u>, <u>semester</u>, <u>year</u>, grade)
Advisor(<u>s_id</u>, i_id)

Time_slot(time_slot_id, day, start_hr, start_min, end_hr, end_min)
Prereq(course_id, prereq_id)
Grade_points(grade, points)

1. (10 points) Find the s_ids and names of all students who were taught by an instructor named 'Katz'.
   Output columns: *Name*
   Sort by: *Name* in ascending order

S_ID	*Name*

2. (10 points) Calculate the *grade-point average* of every student.
   Output columns: *S_ID, GradePointAverage*
   Sort by: *GradePointAverage* in descending order
   Note: The GradePointAverage should round up to 2 digits after decimal point.
   (e.g. 3.42857 should be shown as 3.43)

S_ID	*GradePointAverage*

3. (10 pts) Find the enrollment of each section that was offered in the Fall of 2009. Display the *Course_id, sec_id* and the *count* (which is the number of students enrolled in this section.)

   Output columns: *Course_id, sec_id, Count*
   Sort by: *Count* in descending order

Course_id	*sec_id*	*Count*

4. (10 pts) Find the sections that had the maximum enrollment in the Fall of 2009.
   (There could be more than one course section which has the maximum enrollment.)
   Output columns: *Course_id, sec_id*
   Sort by: *Course_id* in ascending order

Course_id	*sec_id*

5. (10 pts) Find the names of the top 4 instructors who have taught the most number of distinct courses. Display also the total number of courses taught.

Output columns: *InstructorName, NumberOfCoursesTaught*

Sort by: *NumberOfCoursesTaught* in descending order (in case of ties order by the *InstructorName*)

InstructorName	*NumberOfCoursesTaught*

6. (10 pts) Find the top 3 semesters in which the most number of courses were offered. (Treat Spring of 2009 and Spring of 2010 as two different semesters)
   Output columns: *Semester, Year, NumberOfCourses*
   Sort by: *NumberOfCourses in descending order*

Semester	*Year*	*NumberOfCourses*

7. (10 pts) Find the top 2 students who have taken the most number of courses.
   Output columns: *S_ID, StudentName, NumberOfCourses*
   Sort by: *NumberOfCourses in descending order*

S_ID	*StudentName*	*NumberOfCourses*

8. (10 pts) Find the top 4 instructors whose courses have the maximum enrollment in all of their courses combined.
   Output columns: *InstructorName, TotalEnrollment*
   Sort by: TotalEnrollment in descending order

InstructorName	*TotalEnrollment*

9. (10 pts) List all the courses offered by the departments 'Comp. Sci.' and 'History'. Output should not contain any duplicates.
   Output columns: *DepartmentName, CourseID*
   Sort by: *CourseID* in ascending order.

DepartmentName	*CourseID*

10. (10 pts) List all the courses that have prerequisites offered by a different department.

Output columns: *Course_id, Course_department, Prereq_id, Prereq_department*
Sort by: *Course_id in ascending order*

Course_id	*Course_dept*	*Prereq_id*	*Prereq_dept*

Submission instructions:

Please submit via Blackboard the following:

Your SQL script (**answer.sql**). It should contain the 10 SQL queries and look like the following:

--Query1

Select.......

………

-- Query10

Select........

A README file containing your first name, last name, and your Purdue email address.