CS 348: Assignment 2 for Sections 1 and 2

Spring 2020

(due by 5pm EDT on Wednesday, June 10th)

Overview
For this assignment, you must use your Unix accounts and DB2 to compose and evaluate a number of SQL queries over an ENROLLMENT database. The DDL for the relational schema and sample INSERTs for a simple instance are given below, and a plain text source file with these commands is available on Learn. Note that all submissions must use the supplied SQL DDL code, and also that proper testing will involve replacing the supplied INSERT commands with alternative test data.

As with the first assignment, you are given a requirement for each query in English, and your task is to write source code in the SQL query language that implements the requirement.

Assignment submission
Assignment submission should be a plain text file uploaded to dropbox on Learn with answers to each question given as SQL queries. You are to ensure that sufficient commenting is also included to clearly distinguish which queries are answers to which questions.

Note that some the requirements stipulate conditions on what features of SQL may be used in your source code, in particular whether or not `GROUP BY` clauses and aggregate functions may be used. Part of the grading for your answers in these cases relate to these conditions. The rest of grading will be based on two additional criteria: (1) *validity*, the query implements the requirement, and (2) *readability*, how easy it is to understand your source code. This means that the efficiency of your source code, as determined by the DB2 query compiler, will not be a factor in any grading.

Queries that may *not* use aggregation in SQL

1. The student number and first name of students in at least their second year who have obtained a grade higher than 90 in at least two courses their previous year.
2. The professor number and last name of each professor in the CS department who has taught neither CS348 nor CS234 at any time in the past.
3. The number and last name of professors who have taught a CS245 class in which a student obtained a grade that is among the highest ever recorded for CS245.
4. The minimum and maximum final grade for each class in the past with office hours on both Monday mornings and Friday afternoons and that was taught by a professor in the computer science department. The result should include the last name of the professor, and the primary key of the class.

Queries that *may* use aggregation in SQL

5. A count of the number of different students in each term for any course that has never been taught in the past or currently by either a computer science (CS) or applied math (AM) professor. Each result should identify the course, the term and said count, and all results should be sorted by term, starting with the earliest.
6. The percentage of professors who have never taught in the past at least two classes for two different courses during the same term, and for whom this is not the case with the current term. Note that a percentage should be a number between 0 and 100, and that you can assume there is at least one professor.

The ENROLLMENT Schema
The following CREATE TABLE commands define the metadata for a hy-

pothetical university grade management system. Information about enrollments, students, professors, departments, courses, and classes record information about both ongoing and past classes for a course. Note that enrollments for any ongoing class will only have null values recorded tor the grade. You may also assume that each class has at least one enrollment and that a department has at least one professor. Also listed are sample INSERT commands to illustrate how values for attributes such as `cnum`, `term` and `time` are formatted.

```
create table professor ( \
   pnum     integer not null, \
   lastname varchar(30) not null, \
   office   char(6) not null, \
   dept     char(2) not null, \
   primary key (pnum))

insert into professor values (1, 'Weddell', 'DC3346', 'CS')

create table student ( \
   snum      integer not null, \
   firstname varchar(30) not null, \
   year      integer not null,  \
   primary key (snum))

insert into student values (1, 'Mary', 3)
insert into student values (2, 'Fred', 2)

create table course ( \
   cnum      char(5) not null, \
   cname     varchar(50) not null, \
   primary key (cnum))

insert into course values ('CS348', 'Intro to Databases')

create table class ( \
   cnum      char(5) not null, \
   term      char(5) not null, \
   section   integer not null, \
   pnum      integer not null, \
   primary key (cnum,term,section), \
   foreign key (cnum) references course(cnum), \
   foreign key (pnum) references professor(pnum))

insert into class values ('CS348', 'F2018', 1, 1)
insert into class values ('CS348', 'F2018', 2, 1)
insert into class values ('CS348', 'S2020', 1, 1)
insert into class values ('CS348', 'S2020', 2, 1)

create table officehour ( \
   cnum      char(5) not null, \
   term      char(5) not null, \
   pnum      integer not null, \
   day       varchar(10) not null, \
   time      char(5) not null, \
   primary key (cnum,term,pnum,day,time), \
   foreign key (cnum) references course(cnum), \
   foreign key (pnum) references professor(pnum))

insert into officehour values ('CS348', 'S2020', 1, 'Tuesday', '09:00')
insert into officehour values ('CS348', 'S2020', 1, 'Thursday', '14:30')

create table enrollment ( \
   snum      integer not null, \
   cnum      char(6) not null, \
   term      char(5) not null, \
   section   integer not null, \
   grade     integer, \
   primary key (snum,cnum,term,section), \
   foreign key (snum) references student(snum), \
   foreign key (cnum,term,section) references class(cnum,term,section))

insert into enrollment values (1, 'CS348', 'F2018', 1, 88)
insert into enrollment values (2, 'CS348', 'S2020', 1, null)
```