CS 348: Assignment 2 (Spring 2024)

Assigned on May 28 and due by 5:00pm EDT on June 10

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.

Note that some of 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.

Assignment submission

Your submission to Marmoset should be a single zip file that contains one SQL file per question. The files must be named `question1.sql`, `question2.sql`, etc. Each of the files must contain a *single* SQL DML statement as a solution for one of queries specified in English below. Each of the files must be accepted cleanly by DB2. The file must not contain the `connect to cs348` command (or any other DB2 commands). Marmoset will automatically prepare all appropriate tables and run tests. For each question there will be one public test and a number of secret tests. You can only see the results of the public test. You can submit multiple times. The score from your last submission will be considered. You must test your solutions locally before submitting to Marmoset to avoid clogging the build queue, which is a shared and limited resource. Also note that Marmoset builds can take a long time and last minute submissions may not return the public test results before the deadline.

Queries that may *not* use aggregation.

1. The professor number and last name of professors in the computer science department (CS) who have taught a student in CS348 whose final grade was less than 60.

2. The professor number and last name of professors in the CS department who has taught neither CS348 nor CS234 at any time in the past, but is teaching one of the two courses currently.

3. The professor number and last name of professors who have taught a CS245 section in which a student obtained a grade that is the lowest ever recorded for CS245.

4. The number, name and year of each student who is fourth year and who has a final grade of at least 90 in every course that he or she has completed in either the CS or the C&O (CO) departments.

5. The minimum and maximum final grade for each section in the past with office hours on both Monday and Friday and that was taught by a professor in the computer science department (CS). The result should include the last name of the professor, and the primary key of the section.

Queries that *may* use aggregation in SQL

6. The number of different third or fourth year students in each section of each course taught by a pure math (PM) professor in past terms. The result, in addition to the count, should include the professor number, professor name, course number and section, and should be sorted first by the name of the professor, then by the professor number, third by the course number, and finally by section. (Note that a section of a course is identified by a term and a section number. Also assume that sorting by section means sorting by term and then by section number. The result will therefore have a total of six columns.)

7. The percentage of professors who have taught in the past at least two sections of a single course during the same term, but for whom this is not the case in 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 hypothetical university grade management system. Information about enrollments, students, professors, departments, courses, and sections record information about both ongoing and past sections for a course. Note that enrollments for any ongoing section will only have null values recorded for the grade. You may also assume that each section 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 section ( \
   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 section values ('CS348', 'F2018', 1, 1)
```

```
insert into section values ('CS348', 'F2018', 2, 1)
insert into section values ('CS348', 'S2020', 1, 1)
insert into section 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 section(cnum,term,section))

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