Homework 4

Name: Cindy Lai
SID: SID HERE
Partner Name: Kim Dang
Partner SID: 912045263

June 4, 2018

Due 11:59PM June 5, 2018. **READ ALL DIRECTIONS VERY CAREFULLY!** Submit your code, tex files along with a generated PDF. **DO NOT SUBMIT DATA FILES!** For this homework you will be working in groups of two, a group of three will only be allowed with approval due to odd number of students. All programs will be evaluated on the CSIF. Upload your files as a tar gzip file (tgz). Only submit one homework per partner group. This specification is subject to change.

You are designing a database for a university called FakeU. As a trial you have been provided grade data from courses for departments ABC and DEF. The grade data is from Summer of 1989 until Summer of 2012. The data provided is in CSV format, and is only as complete as could be made possible. There may be errors, omissions or redundant data in the files. FakeU like UC Davis is on a quarter system, however they have recently transitioned to a single summer quarter instead of two summer sessions. This has corrupted some of their summer data as all summer session classes have now been grouped into a single summer quarter term. Each course has a course ID (CID), a term it was offered (TERM), a subject (SUBJ), a course number (CRSE), a section (SEC), and number of units (UNITS). Within a course there listings of meetings, the instructor of the meeting (INSTRUCTOR(S)), meeting type (TYPE), day of meeting (DAYS), time of meeting (TIME), meeting building (BUILD), and meeting room (ROOM) are also listed. For each student that takes the course there is a student seat (SEAT), a student ID (SID), the students surname (SURNAME), the students preferred name (PREFNAME), the students (LEVEL), the number of units the student is receiving (UNITS), the students class standing (CLASS), the students major (MAJOR), the grade the student received in the course (GRADE), the students registration status (STATUS), and the students e-mail address (EMAIL). There may be courses that are cross listed between the two departments (e.g. ABC 123 may be cross listed as DEF 456).

You **MUST** put each problem on a separate page with 1a on the second page, for example 1a will be on page 2 and 1b will be on page 3 (this template is already setup for this). You **MUST** put your name and student ID in the provided author section above. **FAILURE TO DO SO MAY RESULT IN NO CREDIT!** The data will be provided on Canvas, and the CSV files will also be on the CSIF in /home/cjnitta/ecs165a/Grades. All submissions will be compared with MOSS, including against past submissions.

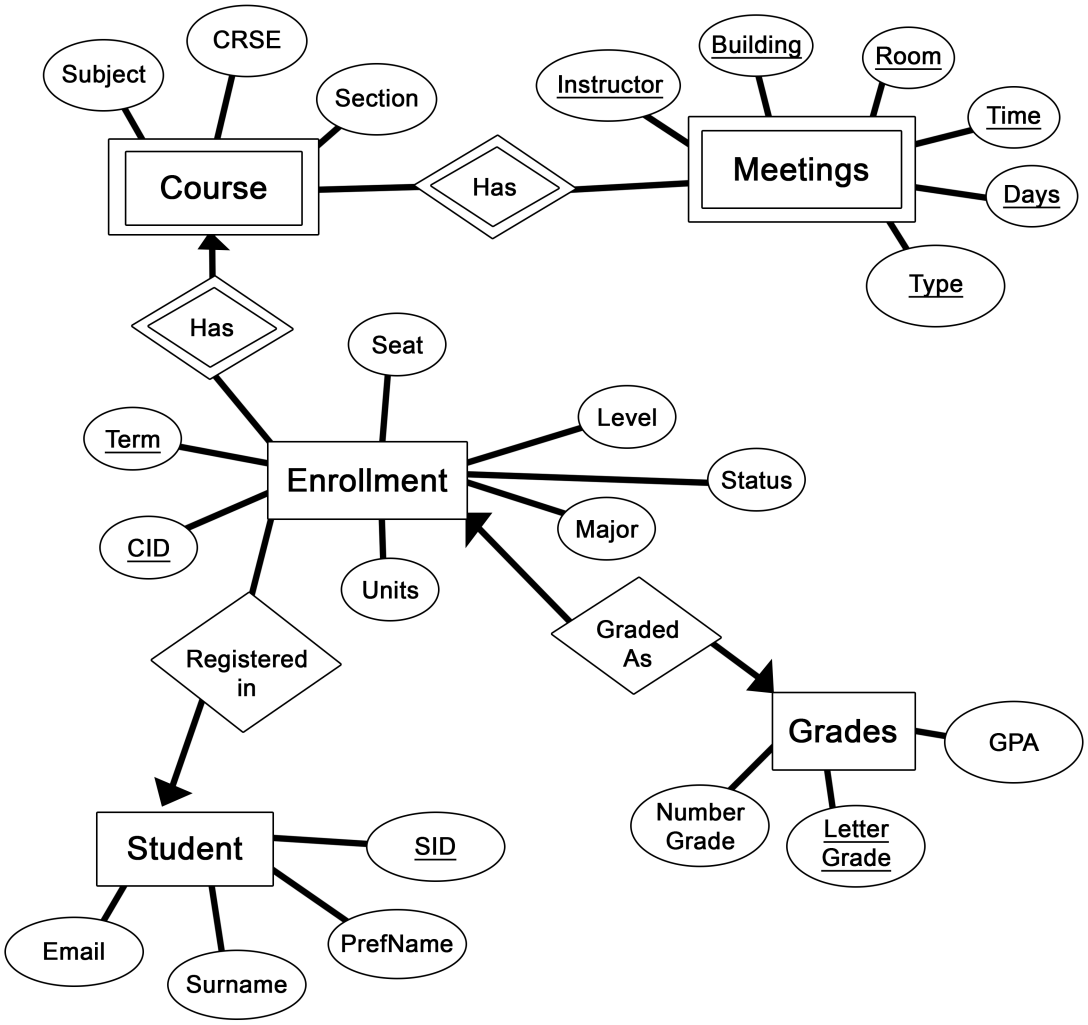
Some useful tips:

- When loading the tuples into the database, insert them in batches. Inserting one tuple at a time may cause the program to take on the order of tens of minutes or hours instead of a few minutes.
- Test a subset of the data first.

Part 1

You will be creating a database schema for your grade data.

a. Provide an ER diagram for your database schema. Only include images generated from vector based programs.



b. Provide a description of the tables in your schema, and their attributes. Make sure you describe how you will store the instructor, student, building, course, etc. information.

For the first table *Student*, the primary key is SID, which can get you a student's name and email. Although SID and *Email* are both unique to each student, SID was chosen as the key because it's used more in relation to the other attributes we're exploring, and in other tables, each student will be referred to by their SID.

The table *Course* is course information of each course enrollment. The primary key is course ID (CID) and $Term$ number. From that, we can get the full course name including $Subject$, $CSRE$, $UNITS$ offered, and *Section* number. We noticed that each section has a unique CID in a $Term$ even if the class is the same, but CID can be reused in different terms.

Each *Course* has at least one $Meeting$. Depending on $MeetingType$, we can find the $Days$, $Time$, $Building$, $Room$, and $Instructor$ for that meeting.

Each entry in *Enrollment* is an indication of a student's status in the term they were enrolled in a certain course. If we know the student's SID and the CID and $Term$ for the course they're taking and when, we can find what *Grade* they got, how many $Units$ they took for that class, the *CourseSeat* they were in for that class, their registration *Status*, *Class* standing, and their *Level*. Since students can change major, each enrollment keeps track of what $Major$ the student currently had declared at the time.

Grade is determined on a numeric scale and has an assigned $NumberGrade$ according to their letter grade.

c. What are the functional (and multivalue) dependencies that you expect to hold for each relation if any. If you don't expect any to hold, describe why not.

The table *Student* has the following functional dependencies:

$SID \rightarrow Surname, PrefName, Email$ $Email \rightarrow Surname, PrefName, SID$

These should hold because both SID and $Email$ is unique to each student.

The table *Grade* has the following functional dependency, which should hold because each letter grade has a specific grade point:

$Grade \rightarrow NumberGrade$

The table *Course* has the following functional dependencies:

$CID, Term \rightarrow Subject, CSRE, Section$

$Subject, CSRE \rightarrow Units$

The table *Meetings* has the following functional dependencies:

$CID, Term, MeetingType \leftarrow Instructor, Days, Time, Building, Room$

The table *Enrollment* has the following functional dependencies and MVDs:

$SID, CID, Term \rightarrow Grade, Major, Units, Class, Seat, Status, Level$ $SID \rightarrow Major$

Part 2

Write a program to load the grade data into a PostgreSQL database called FakeUData that follows your schema. You **MUST** use the database called FakeUData, and should assume it will already be created for you without any tables or data in it. You may **NOT** hardcode usernames in your code, use the USER environmental variable instead if user is needed. Your program can be written in C++ or python, you may **NOT** use standalone SQL or text files that hold your queries. You may **NOT** use shell calls to implement your program. All your queries need to be in your code. If you choose to make a C++ program, you must include a makefile and call the program loadfakeu. Include a readme file with descriptions of any issues/problems. If you choose to make a python program you must specify which version of python you used, and must provide a loadfakeu bash script to launch your python program. The loadfakeu program **MUST** be able to take one optional argument (the directory where the CSV data files will be located). If the argument is omitted, the default is the current working directory. Scripts that require greater than 10 minutes to load all of the data may lose points.

Part 3

Write another program to query your database to calculate the following values, put the results in your write up, some may be best described with a chart instead of raw values. Name your program queryfakeu, it must output the data values for the following queries. The query program does not have to do everything in the SQL queries, but should limit the amount of data transfered. For example it is acceptable to have one SQL query for each unit number (1 - 20) for 3a, but it would be unacceptable to pull all student data on a per student basis and calculate the results.

a. Calculate the percent of students that attempt 1 - 20 units of ABC or DEF per quarter for every unit increment (e.g. 1, 2, 3, ...).

b. Find the easiest and hardest instructors based upon the grades of all the students they have taught in their courses. Provide their name and the average grade they assigned. (Ignore P/NP, S/NS grades)

c. Calculate the average GPA for the students that take each number of units from part a. Assume that the grades have standard grade points (A+ = 4.0, A = 4.0, A- = 3.7, B+ = 3.3...).

d. Find the courses with the highest and lowest pass rates. Assume that F, NP, and NS are not passing grades.

e. Find the list of courses that must be cross listed as they have the same meeting times during the normal quarters. Only list the pair once, put the course name/number string in alphabetically order of the pairs.

f. Find the major that performs the best/worst on average in ABC courses. Repeat the analysis for DEF courses as well.

g. Find the top 5 majors that students transfer from into ABC. What is the percent of students from each of those majors compared to overall transfers?

h. Find the top 5 majors that students transfer to from ABC. What is the percent of students to each of those majors compared to overall transfers out?

Part 4

Extra credit: The Efficient XML Interchange (EXI) is a format for the compact representation of XML information. The CSV files provided for this assignment have been consolidated into a single EXI file (HW4Grades.exi) that is available in the resources section of Canvas. Implement a separate program that it can load the database from the EXI file. You may **NOT** use shell calls, or creation of external temporary files for this part. Name your program or bash script loadfakeuexi.

Part 5

Extra credit: Additional queries/query program.

a. Find the courses that appear to be prerequisites for ABC 203, ABC 210, and ABC 222. For this problem list the courses that the X% of students have taken for every 5% increment from 50% - 100% prior to taking the course. (Add this output to your query program.)

b. Write a program that will find an open room for course expansion. The program must prompt for term, CID, and number students to add. The room(s) returned should be ordered from best to worst fit with up to 5 results. Assume that each room capacity is the maximum number of students listed for any particular meeting in the data files (don't forget that lectures may be split across multiple CIDs). Name this program findroomfakeu.