COMS W4111.001-Introduction to Databases
Spring 2021

Project Part 2

(worth 20% of overall Project grade)

Your team and "project mentor"

You will do Part 2 of the Project with the same team as for Part 1. If your team partner dropped the class and you did not submit a contingency plan for this with your Part 1 submission, then unfortunately you will still have to complete the whole project by yourself. If you team partner dropped the class and you did submit a contingency plan for this with your Part 1 submission, then you are welcome to switch to this reduced version of your project.

For continuity, the IA who grades your Part 1 will grade the remaining parts of your project. (In many but not all cases, this is the IA with whom you met to discuss Part 1.) This IA will be your "project mentor" for the remainder of your project; the name of your mentor will be included in your graded Part 1. You are welcome to contact other members of the class staff (including the instructor) about your project, but your project mentor should be your main contact, for continuity, since your mentor will be grading all parts of the project.

As mentioned earlier, both students in a team will receive the same grade for the Project. Team partners are expected to fully collaborate with each other on solving the project. However, communication about project details with somebody other than your partner is not permitted, and is considered cheating. If in doubt about what kinds of consultations are allowed, check with the instructor, or see the Computer Science Department's policies and procedures on academic honesty. Questions of a general nature that may be of interest to the whole class should be posted to Piazza.

Overview of Part 2

For Part 2 of this project, which you should submit electronically, you should implement your relations of Part 1 over our PostgreSQL database server and load your tables with some real or realistic data, as you outlined in Part 1.

What you need to do for Part 2

1. Open and set up your Google Cloud account using the detailed instructions available here. You should have received an email from Joseph with your individual Google Cloud code and also with credentials for your PostgreSQL database account. If you haven't received this email (please check your @columbia.edu email address), please contact Joseph immediately.
2. Familiarize yourself with the PostgreSQL DBMS by reading the materials available at https://www.postgresql.org/docs/12/. You will be using version 12 on Google Cloud. Please check these materials carefully before sending email to the class staff with questions on syntax or supported features.
3. Use `ssh` to connect to your Google Cloud virtual machine, using the instructions available here.
4. Connect to our section's PostgreSQL database server, which is running at `34.73.36.248`, by running:
   `psql -U <uni> -h 34.73.36.248 -d project1`
   Note: You only need to set up one single PostgreSQL account/database per team, even though your team was assigned one account per teammate. Pick just one of the two accounts arbitrarily and work on it together with your teammate. You will need to let us know with your submission which of the team's two PostgreSQL accounts we should use for grading (see "What to submit..." section below, item 2).

   You will be prompted for your PostgreSQL password, which you should have received by email from Joseph. To change your PostgreSQL password when you first connect, which we recommend, type the following inside the psql prompt:
   `ALTER USER <uni> WITH PASSWORD '<new password>'`
   (the single quotes are necessary).
5. Make all suggested changes to the overall design in general, and to the SQL schema in particular, once you receive your graded Part 1. You are likely to have extensive comments from your project mentor in your graded Part 1. Your Part 2 grade will be based in part on how well you have incorporated your project mentor's comments. If you have any questions about these comments once you received them, please contact your project mentor as soon as possible to clarify.
6. Add any additional attribute-based CHECK constraints and tuple-based CHECK constraints (as discussed in class) that you need so as to express any real-world constraints of your application that are missing from your SQL schema. You can find helpful documentation on constraints in PostgreSQL here. Note that PostgreSQL does not support general assertions. You do not need to use triggers for this project, and you can ignore any real-world constraints that you could not capture with either good-style (as discussed in class) attribute- or tuple-based CHECK constraints, or with PRIMARY KEY, UNIQUE, FOREIGN KEY, and NOT NULL constructs. (Hint: "Good-style" attribute- or tuple-based CHECK constraints tend to refer only to the table in which they are defined, never to other tables.)
7. Create all the SQL tables in your revised SQL schema on your PostgreSQL account (see documentation on CREATE TABLE and on data types), including all constraints that you could specify in the table declarations as described above. Suggestion: Please use `varchar(n)` with an appropriate value for n as the domain for variable-length string attributes such as names, instead of a fixed-length char domain. `varchar(n)` will simplify your handling of such attributes in Part 3, particularly if you are following the Web Front-End Option.
8. Insert into each table in your database, on average, at least 10 tuples of real or "realistic" data, as you described in your Part 1 project description. This data will help you test and play with your database. Of course, issue queries of your choice to make sure that everything works as you intend.
9. Write and run three interesting queries, which you will include with your submission. Collectively, the three queries should cover use of multitable joins, aggregation, and WHERE-clause conditions. (These features don't need to appear in each of the three queries, but they should appear in at least one of the three interesting queries.)

IMPORTANT NOTE: So that you don't exhaust your Google Cloud credits --and also to avoid wasting energy-- make sure to turn off your virtual machine on Google Cloud whenever you are not using it, by following the instructions available here.

What to submit and when

You will submit this part of the project electronically on CourseWorks. The deadline is **Feb 24** By now you have joined (for Part 1) one of the "Project Group" groups on CourseWorks. Just as for Part 1, you should submit your project exactly once per team, rather than once per student. To submit your project, you need to be in the Class view (not the Group view) on CourseWorks and then upload your file to the "Part 2" assignment under Assignments. You should submit a single (uncompressed) file containing:

1. The name and UNI of both teammates
2. The PostgreSQL account name for your database on our server (i.e., specify which teammate's UNI we should use to identify the database for your team); this is the database on which we will base our grading
3. Three "interesting" SQL queries over your database, with a sentence or two per query explaining what the query is supposed to compute. The goal of these queries is to help us better understand your application. You will not be graded on these queries, but we strongly suggest that you submit well formed queries that run without problems, so please make sure that you have tested your queries by running them on your database exactly as submitted (use copy and paste).

Grading for Part 2

Your grade for Part 2 will be split as follows:

1. **Quality of final SQL schema and implementation on PostgreSQL: 8 points.** We will evaluate the overall quality of your final SQL schema on PostgreSQL, especially in terms of how thoroughly you incorporated any revisions suggested by your project mentor in the grading of Part 1 of your project.
2. **Quality of constraint handling: 8 points.** We will evaluate how well you managed to capture real-world constraints through primary key, foreign key, unique, and good-style attribute- and tuple-based CHECK constraints.
3. **Quality of the real-world (or at least realistic) data that you loaded into the database: 4 points.**