CS 775/875 hw 4 3/1/2021 Due date: 3/8/2021

Instructions:

1. Electronic submission: You assignment is **due by 11:00 PM, 3/8**.
2. Accessing MySQL:
   (a) You may only access MySQL from basalt, therefore write and test your queries on basalt.
   (b) Create a directory called *hw4*. Write all your queries in *hw4*.
   (c) Download **culinary.sql**; follow directions provided in lab2 to create the culinary tables in your database on basalt.
   (d) Each query should be in a separate file named **q***i*.sql, where $i = 1, 2, ..., 6$. You should have **q1.sql, q2.sql, ..., q6.sql** corresponding to each query, by order.
3. Submit instructions:
   (a) **Queries must be submitted from agate not basalt.**
   (b) Copy your queries from basalt to agate using ftp, scp, or rsync. Copy files to a directory *hw4* on agate.
   (c) From directory *hw4*, submit your queries using the command:
   $\sim cs775/submit\ 4\ q*$
   (d) 4 is the assignment number. If you want to resubmit, then you need to use 4a, 4b, 4c,..... for assignment number (not 4).
   (e) We have had submission problems in the past. In order to ensure that you get credit for you work, make a tar file of your final submission using the command
   *tar -zcpvf hw4.tar q**
   Do not touch *hw4.tar* until you get back your graded assignment. The tar file keeps a dated copy of submitted files in your directory.
4. The TA will be grading your assignment by using the following command:
   mysql –user username **–password=**password dbname < q1.sql > q1.out
   where *q1.sql* is the input file and the output result is redirected to *q1.out*. Note that there are two hyphens (-) before –user and –password.
   If the username is **xyz** and the password is **zzzzz** the grading command will be:
   mysql –user xyz **–password=**zzzzz xyz < q1.sql > q1.out
   for each query. The dbname, xyz, is the same as your username.
   Please note that the TA will use a different instance of the database while grading.

5. Late policy: 1 day late: 2 points off, 2 days late: 4 points off; $>$ 2 days late: will not be graded.
6. The relevant reading material is from Chapter 6 and Chapter 7.1.
7. The queries are mostly similar to the queries from previous assignments. I think that SQL queries are often easier than RA queries.
8. To test some of the queries, you may have to add data to the culinary database.

Notes about the database:

- The database stores information about different culinary courses. A course is offered by a school and consists of 1 or more levels. Each level is numbered for a course starting with 1 and increasing by 1. (See culinary.sql file for an example.)
- Staff members are either a chef or an assistant.
- Any staff member can also register to be a student in a class (on dates other than the ones s/he works).
- All queries regarding courses refer to just codes (do not check classdates); all queries regarding offerings refer to code + classdate.

Queries

For each of the following queries, write the SQL statement required to produce the desired output according to the "result schema."

1. (5 points) **q1**: Retrieve the names of all staff members. Order the results first by their job and then by their name (both ascending order).

   Result has schema (name, job).

2. (5 points) **q2**: Retrieve staff members who are chefs for the same courses on which they are also students. (On different dates, of course.)

   Result has schema (ssn, name, school, course).

3. (5 points) **q3**: Retrieve courses with no registrations.

   Result has schema (school, course, level, classdate).

4. (5 points) **q4**: Retrieve staff members who never participate in a course whose description contains the text 'Pasta' either as students or on duty.

   Result has schema (ssn, name)

5. (5 points) **q5**: Retrieve students who do not take offerings held in 'Berlin' or 'Naples.'

   Result has schema (name, ssn).

6. (5 points) **q6**: List students who either have no registrations or exactly one registration.

   Result has schema: (ssn, sname)