CS 775/875 hw 6 3/15/2021 Due date: 3/22/2021

Instructions:

1. Electronic submission: You assignment is **due by 11:00 PM, 3/22**.
2. Accessing MySQL:
   (a) You may only access MySQL from basalt, therefore write and test your queries on basalt.
   (b) Create a directory called *hw6*. Write all your queries in *hw6*.
   (c) Download **culinary-hw6.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 *hw6* on agate.
   (c) From directory *hw4*, submit your queries using the command:
   $\sim cs775/submit$ 6 $q*$
   (d) 6 is the assignment number. If you want to resubmit, then you need to use 6a, 6b, 6c,...... for assignment number (not 6).
   (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 *hw5.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

NOTE: Remember to DROP VIEW if you create views.

1. (5 points) **q1**: All students who register for more than three classes in the same school.

   Result has schema (ssn, school, count).

```
+------+--------------------------+----------+
| ssn  | school                   | count(*) |
+------+--------------------------+----------+
| 1111 | Charlie's Cooking School |        4 |
| 2222 | Charlie's Cooking School |        4 |
| 3333 | Charlie's Cooking School |        4 |
+------+--------------------------+----------+
3 rows in set (0.001 sec)
```

2. (5 points) **q2**:

   Retrieve non-staff students and their newly earned credits based on current registrations.

   Result has schema (ssn, name, credits).

```
+------+-------+---------+
| ssn  | name  | credits |
+------+-------+---------+
| 2222 | mary  |      26 |
| 3333 | steve |      26 |
| 8888 | tommy |       3 |
+------+-------+---------+
3 rows in set (0.001 sec)
```

3. (5 points) **q3**:

   Retrieve the staff member who earns the maximum total wages in the current schedule.

   Result has schema (name, maximumwage).

```
+-------+----------+
| name  | sumwages |
+-------+----------+
| alice |     1200 |
+-------+----------+
1 row in set (0.001 sec)
```

4. (5 points) **q4**:

   For each school, retrieve students who have outstanding balances (> 0). Output should be ordered by school (asc) and totalbalance (desc). totalbalance is the sum of outstanding balance for a student.

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

```
+-----------------------------+-------+------+--------------+
| school                      | name  | ssn  | sum(balance) |
+-----------------------------+-------+------+--------------+
| Charlie's Cooking School    | joe   | 1111 |         2210 |
| Charlie's Cooking School    | steve | 3333 |         2022 |
| Charlie's Cooking School    | mary  | 2222 |         1514 |
| Hans's Haute Cuisine        | steve | 3333 |            8 |
| Jacque's Culinary Adventure | steve | 3333 |           11 |
+-----------------------------+-------+------+--------------+
5 rows in set (0.001 sec)
```

5. **q5**:

   Retrieve students and their total outstanding balance if it is greater than 0. If a student has NULL outstanding balance or a 0 outstanding balance, then print NULL for totalbalance.

   Result has schema (ssn, name, totalbalance).

```
+------+-------+--------------+
| ssn  | name  | totalbalance |
+------+-------+--------------+
| 1111 | joe   |         2210 |
| 2222 | mary  |         1514 |
| 3333 | steve |         2041 |
| 8888 | tommy |         NULL |
+------+-------+--------------+
4 rows in set (0.001 sec)
```

6. (5 points) **q6**: Retrieve total non-staff student total credits by adding the new credits from current registrations to existing totalcredits in the students table. Print each non-staff student with total credits.

   Result has schema (ssn, name, totalcredits).

```
+------+-------+--------------+
| ssn  | sname | totalcredits |
+------+-------+--------------+
| 2222 | mary  |           51 |
| 3333 | steve |           61 |
| 8888 | tommy |           26 |
+------+-------+--------------+
3 rows in set (0.001 sec)
```