National University of Singapore
School of Computing

CS2102: Database Systems
Semester 2, 2017/18

Assignment 3 (10 marks)

Due: February 23, 2018 (Friday, 11:59pm)

This assignment is based on the same database schema in Question 2 of Tutorial 3.

`Customers` (cname, area)
`Restaurants` (rname, area)
`Pizzas` (pizza)
`Sells` (rname, pizza, price)
`Likes` (cname, pizza)

Answer each of the following queries using SQL. Remove duplicate records from all query results.

1. Find all restaurants that sell some pizza that Alice likes.

2. Find all pizzas sold by restaurants that are located in the same area as Bob.

3. Find all customers who like at least two different pizzas.

4. Given two restaurants R1 and R2, we say that R1 is more expensive than R2 if for every pizza P that is sold by both R1 and R2, R1's selling price for P is higher than R2's selling price for P. Find all pairs of restaurants (R1,R2) where R1 is more expensive than R2. Exclude restaurant pairs that do not sell any common pizza.

5. Find all customers such that every restaurant that is co-located with them sells only pizzas that they like. Exclude customers who are not co-located with any restaurant.

6. You are given a budget of $40 to buy two distinct pizzas from the same restaurant. Find all restaurants that you could buy from (without exceeding your budget).

7. Three friends (Moe, Larry, and Curly) plan to share three pizzas for dinner at a restaurant that must satisfy all these requirements:

   (a) the three pizzas ordered must be distinct pizzas,

   (b) the total cost of the three pizzas must not exceed $80,

   (c) each of the three pizzas must be liked by at least one of the friends, and

   (d) each of the friends must like at least two of the three pizzas (the pair of pizzas liked by each of them do not necessarily have to be the same).

   Find all (R,P1,P2,P3,TP) tuples where the three friends could share the pizzas {P1,P2,P3} from the restaurant R with a total cost of TP such that $P1 < P2 < P3$.

8. Given a restaurant R, let $numPizza(R)$ denote the number of pizzas sold by R and $priceRange(R)$ denote the difference between the maximum and minimum prices of pizzas sold by R. If R does not sell any pizzas, then $numPizza(R) = 0$ and $priceRange(R) = 0$.

   Given two restaurants, $R1$ and $R2$, we say that $R1$ is *more diverse* than $R2$ if either (a) $numPizza(R1) > numPizza(R2)$ and $priceRange(R1) \geq priceRange(R2)$, or (b) $numPizza(R1) \geq numPizza(R2)$ and $priceRange(R1) > priceRange(R2)$.

   Find all restaurants R such that there does not exist any restaurant $R2$ that is more diverse than R.

9. For each area $A \in \pi_{area}(Customers)$, find the following information:

   (a) the area name A,

   (b) the total number of customers located in A,

   (c) the total number of restaurants located in A, and

   (d) the price of the most expensive pizza sold by restaurants in A; if there are no restaurants selling pizzas in A, show the value 0.

10. Find all restaurants that satisfy the following conditions.

    (a) the restaurant must sell at least 3 pizzas,

    (b) at least one the pizzas sold by the restaurant must be cheaper than $20, and

    (c) for each area $A \in \pi_{area}(Customers)$, there must be at least two customers located in A who like at least one pizza sold by the restaurant. Note that the two customers do not necessarily like the same pizza sold by the restaurant.

1 How to set up

1. Login to `sunfire` server.

2. Download the file http://www.comp.nus.edu.sg/~cs2102/cs2102-assign3.zip as follows.

```
$ cd ~
$ wget http://www.comp.nus.edu.sg/~cs2102/cs2102-assign3.zip
$ unzip cs2102-assign3.zip
```

   The unzipped directory cs2102-assign3/ contains the following files.

   - setup0.sh, setup1.sh - bash scripts to update environment variables
   - check.sh - bash script to display the values of your `PostgreSQL` environment variables.
   - assign3.sql - template SQL script for you to fill in your answers.
   - data/ - directory containing CSV files for seven database instances; e.g., the directory `db1/` contains the CSV files for the first database instance.

- createtables.sql - SQL script to create the database schema.
- solution-files/ - directory containing the solution output files for the seven database instances; e.g., the directory `db1/` contains the solution files for the first database instance.
- test.sql - bash script to compare the outputs of your SQL answers against the provided solution files.

3. Execute setup0.sh if your assigned `PostgreSQL` server is psql0; otherwise, execute setup1.sh.

```
$ cd ~/cs2102-assign3
$ bash setupX.sh
$ source ~/.bash_profile
$ echo $PATH
```

   You should see the directory names `/home/course/cs2102/bin` and `/usr/local/postgres/10-pgdg/bin/64` included in the value of `PATH`. Note that you do not need to repeat this step for subsequent logins.

   If you have not already configured your `~/.pgpass` file, a message will be displayed explaining how to configure this file.

4. Check your `PostgreSQL` environment variable configurations by executing the check.sh script. The following illustrates the output for a user with `PostgreSQL` account "alice" assigned to `PostgreSQL` server "psql0".

```
$ cd ~/cs2102-assign3
$ bash check.sh
PGUSER=alice
PGHOST=psql0
PGDATABASE=cs2102
PGSQL_EDITOR=/usr/local/bin/vim
```

   If the values shown are not the expected values, edit the file `~/.bash_profile` with the necessary changes.

5. Create the database schema as follows.

```
$ psql < createtables.sql
```

2 How to prepare & check your answers

Edit the provided SQL script assign3.sql to fill in your answers for the ten questions. Enter your matriculation number on the line commented with "ENTER YOUR MATRICULATION NUMBER:".

For each question, say Question X, a view has been defined named `qX` in `assign3.sql` with a dummy definition consisting of a single line (e.g., `SELECT 1`) that has the comment "replace this line". To input your answer for Question X, simply search for the view definition `qX` in `assign3.sql`, and replace the single line commented with "replaced this line" with your SQL answer.

You can compare the outputs of your SQL answers against the provided solutions in `solution-files/` using the test.sh script. For example, to compare the outputs for the first database instance, execute "bash test.sh 1". The `test.sh` script will first execute your `assign3.sql` script to create all your view definitions. Next, for each view definition qX, it will generate the output of your view qX in a file named `your-solution/db1/qX.txt` and compare it against the corresponding solution file named `solution-files/db1/qX.txt` using the `diff` command. The `diff` command will create an output file named `diff-dir/db1/qX.txt` which will be empty if your solution file matches the provided solution file; otherwise, the output file will contain the differences between the two solution files. Refer to https://en.wikipedia.org/wiki/Diff_utility for an overview of how to interpret the output created by the `diff` command. Edit `assign3.sql` and run `test.sh` again if necessary to fix any errors.

You can also compare the outputs for more than one database instance: for example, "bash test.sh 2 4" will compare the outputs for both the second and fourth database instances; and "bash test.sh" without any arguments will do so for all the database instances.

3 How to submit

Submit your SQL script `assign3.sql` as follows.

```
$ submit-assign3
You submitted assign3.sql (size: 1622 bytes, #lines: 51) on Friday, February 9, 2018 12:31:13 PM SGT
```

You may make multiple submissions using the `submit-assign3` command and only the latest submission will be graded. Your submission will be auto-graded using the provided (and possibly additional) database instances.