Homework 3 (SQL)

Released Friday 9/20 5:00 pm
Due Friday 9/27 6:00 pm

Problem 1 (100 pts)

Consider a database containing information about bars, beers, and bar-goers.

Drinker (*name*, *address*)
Bar (*name*, *address*)
Beer (*name*, *brewer*)
Frequents (*drinker*, *bar*, *times_a_week*)
Likes (*drinker*, *beer*)
Serves (*bar*, *beer*, price)

Write the following queries in SQL. To set up the sample database called `beers` (even if you have set it up previously, you should repeat this process to refresh it), issue the following commands in your container shell:

```
$DBCOURSE/sync.sh
$DBCOURSE/examples/db-beers/setup.sh
```

Then, type "`psql beers`" to run PostgreSQL's interpreter. For additional tips, see "Help → PostgreSQL Tips" on the course website.

You should check that your queries return the intended answers on our sample database. For grading, however, your answers may be tested on other databases with the same schema but different contents, so your queries need to be correct in general to receive full credits.

As soon as you get a working solution for one part of this problem, say (a), record your query in a plain-text file named **`a-query.sql`** (replace "`a`" with "`b`", "`c`", and other parts as appropriate).

You can submit any number of files at once. An autograder will run automatically on your submission and give you a report of what it finds — the `db0` test database it uses is identical to the sample one you have, while `db1` is a hidden test database. You can use the autograder report to help you debug your queries and resubmit. If you cannot get a query to parse correctly or return the right answer, include your best attempt and explain it in comments, to earn possible partial credit.

You cannot use cross product of two relations in a query without a valid WHERE condition (join the tables), i.e., if you had solved a similar problem in HW1 for RA using cross product, you cannot use that here. You may get 0 points even if your query passes the autograder if we find cross products in your queries.

Result rows should have not any duplicates unless otherwise stated.

Remember to ***(re)submit all files*** in your final submission.

(a) (25 points) For each beer `Ben` likes, find bars among all bars `Ben` frequents that serve this beer at the highest price. If a beer liked by `Ben` is not served by any bar he frequents, your output should be (beer, `NULL`).
Note: If there are multiple bars frequented by Ben that serve a beer `Ben` likes at the highest price, you should output all of them.
[Output columns: beer, bar]

(b) (25 points) Find all combinations of two drinkers and a bar where drinker1 and drinker2 both frequent this bar and that this bar serves at least one beer they both like.
drinker1 should be lexicographically smaller than drinker2.
Sort your output based on the number of beers liked in common, then by bar name, then drinker1 name, then drinker2 name. All are in ascending order.
Note: [Output columns: drinker1, drinker2, bar, num_common_beers]

(c) (25 points) Find names of bars that **only** serve beers that are liked by at least one drinker who frequents that bar. In other words, we discard bars such that they serve some beer that is not liked by any drinkers who frequent that bar. Remove duplicates.
Note: We only consider bars that serve some beers. If a bar does not serve any beer at all, don't include it in the output.
[Output columns: bar]

(d) (25 points) Find all (drinker1, drinker2) pairs such that drinker2 likes all the beers drinker1 likes, but **not all** beers liked by drinker2 are liked by drinker1.
[Output columns: drinker1, drinker2]

Extra Credit -(e) (10 points)

(e) (10 points) For each bar, find its *Fan Drinkers* and *Foe Drinkers*. A *Fan Drinker* is a drinker who frequents this bar the highest number of times per week ***and*** likes the maximum number of beers served at this bar (hence some bars may not have any *Fan Drinker* at all). A *Foe Drinker* is a drinker who frequents this bar the least number of times per week ***and*** likes the least number of beers served at this bar (consider the case where the drinker does not like any beer served at this bar) – (some bars may not have any *Foe Drinker* at all). Note that a drinker has to frequent the bar at least once to become a *Fan Drinker* or *Foe Drinker*. There can be multiple *Fan Drinkers* and *Foe Drinkers* for each bar. Format your output as (bar, drinker) where "drinker" can be

either a *Fan Drinker* or a *Foe Drinker*. If a bar has neither *Fan Drinkers* nor *Foe Drinkers*, do not include the bar in your output. Remove duplicates. That is, for a bar, if a drinker is both a *Fan Drinker* and also a *Foe Drinker*, we only record this (bar, drinker) pair once.
[Output columns: bar, drinker]

Extra Credit -(f) (10 points)

(f) (10 points) Find the names of all drinkers who frequent **every** bar that serves **only** beers they like.
Note: cross product of two relations is not allowed.
[Output columns: drinker]