Database Systems, CSCI 4380-01
Homework # 5
Due Thursday October 10, 2019 at 11:59:59 PM

Homework Statement. This homework is worth 2% of your total grade. If you choose to skip it, Midterm #2 will be worth 2% more. This homework will concentrate on SQL skills. You will be able to test your queries against real data and expected output.

If you want to create the database on your local database server, you can find the database dump file at:

`http://www.cs.rpi.edu/~sibel/DBS_Past_Materials/Fall2019/`

You can create a database for this data, let's call it `hw5` on `psql` shell:

```
psql> create database hw5;
```

and then load the data (after unzipping) using the following Unix command (on Linux and Macs):

```
cat hw5.dmp  | psql hw5
```

The data model for this homework is given in the end of this homework. We have data regarding AirBnB listings, rental and hotel prices, as well as reviews. All data is from New York City. This data is sampled down from a much larger database for simplicity (less than 2% of the full data) and has multiple copies of the same hotel due to noise (same name but different hotel ids).

Database Server Use Rules

If you want to install and create the database on your own computer, you can use the data scripts I used. You do not have to have a database server installed to do this homework. This database will be created as `hw5` on the shared database server at:

`http://rpidbclass.info`

Feel free to use it for testing your queries, but please be considerate of others when using the server. Here are a few ground rules:

- Server response to load can be unpredictable. So, be patient if you see some slow down. This is a medium sized database for a 100+ student class, so you can expect a serious slow down near the homework deadline. Please do not wait till the last minute to submit your homeworks.
- Test your queries one at a time. Best set up is using a browser and a text editor. Write queries elsewhere and test in the server with cut and paste.
- Make every effort to read your queries before submitting. A forgotten join condition may mean disaster. Check join conditions first, then run your queries.

- Remember if you have an unresponsive query, it will continue to use system resources even if you quit your browser. So, opening a new browser window will solve your problem but may slow things down for everyone. Queries should terminate after 2 minutes, so if you increased the load with a bad query, then wait for your query to end before submitting another.

  If you are experiencing problems with a query, read it carefully before running it again in a separate window. Remember: missing join conditions is the difference between: 2,094,266,610,000 tuples and 3335 tuples.

- If the server is not responsive, let us know on Piazza. I will see if some jobs need to be killed or whether server needs to be made more powerful. Please be patient.

- Please do not include a query that does not run in your homework submission. I will run all your queries in a batch job and an incomplete query will cause me a great deal of problems.

1 Problem Description

Write the following queries in SQL. In all your queries, use the simplest possible expression possible. As these are fairly simple queries, they should be pretty fast to execute.

Query 1 Return the distinct name of all hotels with known latitude and longitude values and zip value 10019. Find and return the distance of the hotel to the latitude and longitude location: `40.750572,-73.9957077` (Penn Station) and order the hotels in increasing order to this location. Format the distance to only return 2 digits after comma.

Note: I created a function called `geo_distance(lat1,long1,lat2,long2)` for computing distance between two lat/long values you can use for this. See below for usage examples. The hand crafted function may not be fully correct but is hopefully close. Of course, it computes the distance using "as the crow flies" method.

Remember you can cast between values, such as `numeric(4,2)`.

```
select geo_distance(40.7484445,-73.9878531,40.750572,-73.9957077);
```

Query 2 Find listings with at least 200 reviews (`listings.number_of_reviews`), are available at least one day in October and has at least one review posted after '6/1/2019'. Return the listing id, name and the date and first 50 characters of reviews posted after '6/1/2019', order by listing id.

Query 3 Find AirBnB reviews that contain the words `spacious` or `stylish` (in lower case like this), posted by a reviewer who have posted at least one other review. Return the reviewer name and review id, order by review id.

Query 4 Return the name and county of all regions with at least one listing with price greater than `money '$5000'` and have no median rental prices greater than `5000` in `2019`. Order by name and county.

Query 5 Find average number of AirBnB reviews for each region and county (using `listings.number_of_reviews`) order by the average number of reviews desc. Return the regions/county with at least `50` average reviews.

SUBMISSION INSTRUCTIONS. You will use SUBMITTY for this homework.

Submit a single ASCII text file named `username_hw5ans.sql` that contains all your queries to SUBMITTY. I will post submission instructions later for this on Piazza. We will use Submitty for all SQL homeworks. However, Submitty is not yet set up, so this may take some time.

Your script should be formatted as shown below:

```
-- Print your answer and RCS id first
SELECT 'Student: Sibel Adali (adalis@rpi.edu)';

-- Print the name of each query before the query output
-- Pay close attention to the columns requested as well as the
-- requirements for ordering of results for each comparison

SELECT 'Query 1';

-- Replace this with your answer for Query 1.
SELECT count(*) FROM listings ;

--- Repeat this pattern for each query

SELECT 'Query 2';

-- Replace this with your answer for Query 2.
SELECT count(*) FROM rental_prices ;


SELECT 'Query 3';

-- Replace this with your answer for Query 3.
SELECT count(*) FROM hotels ;
```

Database Schema

The database consists of a number of data tables collected from three different sites: Airbnb listings, Zillow data on rental prices and TripAdvisor reviews of Hotels. All data is for New York City. Note that this is real data, so it may be noisy. Make a habit of using simple queries to first explore the data to understand various issues.

```
-- AirBnB listings for homes
create table listings (
     id                  INT PRIMARY KEY
     , name               VARCHAR(1000)
     , host_id            INT
     , host_name          VARCHAR(1000)
     , county             VARCHAR(100)
     , region             VARCHAR(1000)
     , latitude           FLOAT
     , longitude          FLOAT
     , room_type          VARCHAR(100)
     , price              MONEY
     , minimum_nights     INT
     , number_of_reviews  INT
     , last_review        DATE
     , reviews_per_month  FLOAT
     , calculated_host_listings_count INT
     , availability_365   INT
) ;

-- Availability of Listings on specific days
create table calendar (
      listing_id      INT
      , cdate         DATE
      , available     BOOLEAN
      , price         MONEY
      , adjusted_price MONEY
      , minimum_nights INT
      , maximum_nights INT
      , PRIMARY KEY(listing_id, cdate)
      , FOREIGN KEY (listing_id) REFERENCES listings(id)
) ;

-- Reviews for AirBnB lists
create table reviews(
      listing_id    INT
      , id           INT PRIMARY KEY
      , rdate        DATE
      , reviewer_id  INT
      , reviewer_name VARCHAR(100)
      , comments     TEXT
      , FOREIGN KEY (listing_id) REFERENCES listings(id)
);

-- Median rental prices in New York City for specific home types
-- for various months. Data is not available for all months.
create table rental_prices(
      hometype       VARCHAR(100)
      , region       VARCHAR(100)
      , county       VARCHAR(100)
      , yearmonth    DATE
```

```
       , median_rental FLOAT
       , primary key (hometype, region, county, yearmonth)
);

-- Data dump of hotels in New York City from Trip advisor, with
-- price range and approximate address.
create table hotels(
      id          INT PRIMARY KEY
      , name      VARCHAR(100)
      , low_price FLOAT
      , high_price FLOAT
      , street1   VARCHAR(200)
      , street2   VARCHAR(200)
      , zip       VARCHAR(200)
      , imgurl    VARCHAR(200)
      , latitude  VARCHAR(20)
      , latitude VARCHAR(20)
);

-- Data dump of reviews of hotels from Trip Advisor
create table ta_reviews(
       hotelid          INT
   , title          VARCHAR(200)
   , author         VARCHAR(100)
   , reviewid       VARCHAR(100)
   , rdate          DATE
   , author_location VARCHAR(100)
   , service        FLOAT
   , cleanliness    FLOAT
   , overall        FLOAT
   , value          FLOAT
   , sleep_quality FLOAT
   , rooms          FLOAT
   , location       FLOAT
   , PRIMARY KEY (hotelid, reviewid)
   , FOREIGN KEY (hotelid) REFERENCES hotels(id)

);

CREATE OR REPLACE FUNCTION geo_distance(
              lat1 float
              , long1 float
      , lat2 float
      , long2 float) RETURNS FLOAT AS $$
DECLARE
   dlat float;
   dlong float;
   a float;
   c float;
   pi float;
BEGIN
    select pi() into pi;
    -- Convert to radians before applying the formulas
    lat1 = lat1*pi / 180.0;
    long1 = long1*pi / 180.0;
    lat2 = lat2*pi / 180.0;
    long2 = long2*pi / 180.0;

    -- Now the real work.
```

```
    dlat = (lat1-lat2);
    dlong = long1-long2;
    a = sin(dlat/2.0)*sin(dlat/2.0) + cos(lat1)*cos(lat2)*sin(dlong/2.0)*sin(dlong/2.0);
    c = (6371.0/1.609) *2*atan(sqrt(a)/ sqrt(1-a));
    return c;
END ;
$$ LANGUAGE plpgsql;
```