Big Data - SQL Assignment

Description

For this assignment, you will use a MySQL server that we have set up on the HPC cluster. Log in to dumbo using your HPC credentials (remember, you will need to first ssh into hpc.nyu.edu, and then from there, ssh into dumbo). Once you have logged into dumbo, you can access the MySQL server by typing:

```
mysql -h compute-1-8 -u {yournetid} -p
```

where you should replace "yournetid" with your NYU NetID in the above command. When you are prompted for a password, enter the password for the server that was emailed to you last week.

We have created a database for each of you to use for this assignment. To use your database, type

```
use yournetid;
```

where, again, you should replace "yournetid" with your NYU NetID in the above command.

Please note: We are experiencing degraded performance on dumbo, in particular, when using MySQL. Feel free to work on this assignment on your own computer. You may download the data from the urls given in the section below. We recommend avoiding task 1 on your own machine, unless you take the faster approach using SQL "views".

Please note: If you are having trouble login into MySQL on dumbo, please work on this assignment on your local machine.

Assignment

The NYC Taxi & Limousine Commission (TLC) captures information about each taxi trip in the City. In this assignment, you will use MySQL to "play" with a subset of these data: trips that took place between August 1 and August 7, 2013. You can access the data in:

- `http://vgc.poly.edu/~juliana/courses/BigData2016/Assignments/Assignment1/fare_data_week1.csv`
- `http://vgc.poly.edu/~juliana/courses/BigData2016/Assignments/Assignment1/trip_data_week1.csv`

- `http://vgc.poly.edu/~juliana/courses/BigData2016/Assignments/Assignment1/licenses.csv`

Each trip contains information about pickup and dropoff location, pickup and dropoff time, and a number of other attributes including fare, tip, and distance traveled. The TLC releases the data in two files: one containing trip data and another fare data.

You will start by creating two tables, one for fares and one for trips and load the data into these tables. Here are the CREATE TABLE statements you should use:

```
CREATE TABLE trips (
       medallion varchar(50),
       hack_license varchar(50),
       vendor_id VARCHAR(3),
       rate_code SMALLINT,
       store_and_fwd_flag VARCHAR(3),
       pickup_datetime TIMESTAMP,
       dropoff_datetime TIMESTAMP,
       passenger_count SMALLINT,
       trip_time_in_secs INT,
       trip_distance DECIMAL(12,5),
       pickup_longitude DECIMAL(15,10),
       pickup_latitude DECIMAL(15,10),
       dropoff_longitude DECIMAL(15,10),
       dropoff_latitude DECIMAL(15,10)
);

CREATE TABLE fares (
       medallion varchar(50),
       hack_license varchar(50),
       vendor_id VARCHAR(3),
       pickup_datetime TIMESTAMP,
       payment_type VARCHAR(3),
       fare_amount DECIMAL(15,10),
       surcharge DECIMAL(15,10),
       mta_tax DECIMAL(15,10),
       tip_amount DECIMAL(15,10),
       tolls_amount DECIMAL(15,10),
       total_amount DECIMAL(15,10)
);
```

MySQL allows you to populate tables with data from CSV files. See the following for details: `http://dev.mysql.com/doc/refman/5.1/en/load-data.html` Assuming you saved the trip data in a file called trip_data_week1.csv, you can use the following command to load the trip data into the trips table.

```
LOAD DATA LOCAL INFILE 'trip_data_week1.csv'
INTO TABLE trips
```

```
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES;
```

You can use a similar command to load the fares data into the fares table.

Task 1 - Generate all trips

Write a SQL query that joins the "trips" and "fares" tables and populates a new table called "alltrips". Note that the "fares" and "trips" tables share 4 attributes: `medallion`, `hack_license`, `vendor_id`, `pickup_datetime`.

output schema: `medallion, hack_license, vendor_id, pickup_datetime, rate_code, store_and_fwd_flag, dropoff_datetime, passenger_count, trip_time_in_secs, trip_distance, pickup_longitude, pickup_latitude, dropoff_longitude, dropoff_latitude, payment_type, fare_amount, surcharge, mta_tax, tip_amount, tolls_amount, total_amount`
output order: `medallion, hack_license, vendor_id, pickup_datetime, pickup_longitude`

Please note: If not specifically instructed, you should order by the columns specified in **increasing order**.

Please note: Your solution needs to select all the rows of the created "alltrips" table. Suppose you've created the table and you should do:

```
select * from alltrips;
```

Task 2 - Understand the data

Write the following queries in SQL to help you understand the data.

Task 2a

Find the distribution of fare amounts, i.e., for each amount A, the number of trips that cost A.

output schema: `amount, num_trips`
output order: `amount`

Task 2b

Find the number of trips with a total cost less than $10.

output schema: `num_trips`

Task 2c

Find the distribution of the number of passengers.

output schema: `number_of_passengers, num_trips`
output order: `number_of_passengers`

Task 2d

Find the total revenue (for all taxis) per day. The revenue should include the fare amount, tips, tax, tolls, surcharges.

output schema: `day, total_revenue`
output order: `day`

Task 2e

Find the total number of trips for each taxi.

output schema: `medallion, num_trips`
output order: `medallion`

Task 3 - Identify data problems

Write SQL queries to help answer the questions below to help you identify potential problems with the data.

Task 3a

Is there more than one record for a given taxi at the same time? Find the answer by writing a query.

output schema: `medallion, pickup_datetime`
output order: `medallion`

Task 3b

For each taxi, what is the percentage of trips without GPS coordinates, i.e., all 4 coordinates are recorded as 0's?

output schema: `medallion, percentage_of_trips`
output order: `medallion`

Task 3c

Find the number of different taxis used by each driver.

output schema: `hack_license, num_taxis_used`
output order: `hack_license`

Task 4 - Drivers and vehicles

You will now use one additional data set that contains information about the actual vehicles used in the trip. You can obtain the data in the licenses.csv file from the url listed previously in this document.

Here's the schema you should use:

```
CREATE TABLE medallions (
       medallion varchar(50),
       name varchar(50),
       type varchar(30),
       current_status varchar(10),
       DMV_license_plate varchar(10),
       vehicle_VIN_number varchar(20),
       vehicle_type varchar(10),
       model_year DECIMAL(4),
       medallion_type varchar(30),
       agent_number INTEGER,
       agent_name varchar(30),
       agent_telephone_number varchar(15),
       agent_website varchar(50),
       agent_address varchar(50),
       last_updated_date DATE,
       last_updated_time TIME
);
```

Note: In this task, "total revenue" refers to the sum of fare amounts and "tip percentage" refers to the average ratio of tip to fare amount, i.e., $\frac{1}{n}\sum_{i=1}^{n}\frac{\text{tip\_amount}(i)}{\text{fare\_amount}(i)}$.

Task 4a

Compare trips based on `vehicle_type`: `WAV, HYB, CNG, LV1, DSE, NRML`.

output schema: `vehicle_type, total_trips, total_revenue, avg_tip_percentage`
output order: `vehicle_type`

Task 4b

Compare trips based on medallion_type(Named Driver, Owner must drive).

output schema: `medallion_type, total_trips, total_revenue, avg_tip_percentage`
output order: `medallion_type`

Task 4c

List the top 10 agents by total revenue.

output schema: `agent_name, total_revenue`
output order: `total_revenue (desc), agent_name`

Please note: You need to sort first by decreasing `total_revenue`, and if there is a tie, sort by increasing `agent_name`.

Submission

We will auto-grade your solutions by black-box testing. Your solutions will be run on the hidden final test data (unknown to you when you work on the assignment). We will compare your outputs with the correct answers. For each task, your output must match exactly the answer, otherwise you will score zero points (no partial credits) for the task.

To help eliminate minor issues such as output formatting, we have created a self-testing tool for you at `http://bowenyu.me/task-runner/`. It is highly recommended that you run your solutions on the self-testing tool before submitting your assignment to NYU Classes.

- To test your solution on individual task: Select the task and paste your query source code into the code box. Then click "Test On Sample".
- To test your entire homework submission as a zip file: Select "all-tasks" and upload a zip file containing all the solution source codes. Your file names in the zip must match exactly what is described on the site. Then click "Test All On Samples". It is recommended that you passed every task individually before attempting "all-tasks".

Please note: Each task is run independently. For example, if you create the table "alltrips" in task 1, when you are running your solution for task 2a you won't have the table "alltrips" unless you create it in the solution to task 2a as well.

Please note: The self-testing system is only checking for formatting and basic execution issues of your queries. The system uses sample tables that are only a small portion of the final test data. **Passing tasks on the sample tables do not at all guarantee correctness on the final tests**. If a solution failed the final tests on a task it will score zero points on that task without any partial credit, regardless of whether it passed the sample table. It is your responsibility to check the correctness of your queries.