University of California, Los Angeles
Department of Computer Science

Computer Science 143 **Prof. Ryan Rosario**

Homework 2

Due Monday, April 19, 2021 11:59pm ***via CCLE***

Hints: For some of these queries in Part 2, you will need to use functions on attributes. Check out the list of date and time functions here, as it should be very useful. **You do not need to memorize them!** Note that these functions are *not* aggregation functions because aggregation functions may take multiple inputs and produce one output per group. These functions take one input and return one output without using any grouping variables. This does not mean that your queries will not use the aggregation functions we discussed in class though.

Important! To help the grader, each problem requires you to provide some of the output from your queries. So, you will need to load the data into Postgres.

For parts 1 and 2 of this assignment, use the BART scenario from Homework 1 and for part 3, use the Bird Scooter scenario from Homework 1.

Part 1: SQL

We will again use the BART dataset. To remind yourself about BART, see Homework 1.

```
-- The schema isn't exactly set up this way in the HW2 tarball since we are not modifying data,
-- we don't need to worry about referential integrity. The proper schema is provided for your reference.
CREATE TABLE station(
    Abbreviation char(4) PRIMARY KEY,
    Location varchar(23) NOT NULL,
    Name varchar(50) NOT NULL
);

CREATE TABLE ridecount(
    Origin char(4) REFERENCES station(Abbreviation),
    Destination char(4) REFERENCES station(Abbreviation),
    Throughput integer,
    TStamp timestamp,
    PRIMARY KEY(Origin, Destination, TStamp)
);
```

Note that this schema is slightly different from Homework 1.

Exercises

(a) Write a query that computes the total number of trips that started at each BART station. Output the `Origin` and total Throughput as `total_throughput`. **To help our grader, also submit the first 10 rows of your output without changing the order of the rows.**

(b) Write a query that computes the total number of trips that started at each BART station. Only keep stations that had more than 100,000 entries/boardings. Output the station `Name`, `Location` and total Throughput as `total_throughput`. **To help our grader, also submit the first 10 rows of your output without changing the order of the rows.**

(c) Write a query that computes movement among cities. That is, compute the total number of trips (`Throughput`) between city A and city B and call it `total_rides`. For example, we want to know how many trips started in Oakland and ended in Fremont, and vice verse. Output the city (`Location`) corresponding to `Origin`, the city `Location` corresponding to `Destination` and `total_rides`. Do not output `Origin` or `Destination`, as that would not make sense. Sort from largest to smallest `total_rides` and **report the first 10 rows without doing any additional ordering.** *Hint:* This is very similar to problem 2b in HW 1.

(d) Write a query that finds the maximum `Throughput` ever recorded in this dataset, as well as the `Origin` and `Destination` for this trip, and the date/time (`Tstamp`). How can we do it **without** using `LIMIT` and `ORDER BY`? **In addition to your query, please submit the row.**

Part 2: Relational Algebra

Exercises

(a) Consider the relation `RideCount` from Homework 1. Suppose we filter out all weekend traffic from `RideCount` and create a new relation called `RideCountWeekday` containing information about traffic that occurred on a weekday, since weekend traffic is drastically different the workweek traffic. Using this new relation, write a relational algebra expression that does the following: only keeps tuples from `RideCountWeekday` with non-zero `Throughput`, and with what's remaining, compute the average `Throughput` by `Hour` and route. The output would be one tuple, per hour, per route. **Note:** While ridership may depend on both the day of week and the hour of the day, we are going to ignore the day of the week in this problem.

(b) Suppose the Mayor of San Francisco wants to create a public transit campaign. She wants `Throughput` data for all trips that start in San Francisco and end in San Francisco. The output should be `Origin`, `Destination`, `Date`, `Hour` and `Throughput`, but should only contain tuples where the `Origin` and `Destination` is in San Francisco. Write the relational algebra expression for this problem. You will need to explicitly specify conditions. *Hint:* You need to use both relations.

Part 3: SQL Schemas

In Homework 1, we created a relational schema and diagram for the Bird Scooter example. In this problem, we will create a SQL schema using the `CREATE TABLE` syntax. This means we also need to pick the proper data types for each column. For a description of how Bird Scooter works, see Homework 1.

We need a table to represent a `scooter`. Each of the following statements is designed to give you a hint as to the proper data type.

1. Each scooter has an identifier `scooter_id`, a number. Since Bird is a startup, we assume that there are no more than 10,000 scooters.
2. Each scooter has a flag `status` that marks it as online, offline (broken etc.), and lost/stolen. Each scooter can have only one of these states at a time, and must have a state.

We need a table to represent a `customer` (`user` is a system keyword so I will not use it):

1. Each user has an identifier `user_id`, a number, and we assume that Bird has at most 500,000 users for now.
2. A user is just someone that installed the app, not necessarily someone that will use a scooter. Thus, they may, or may not have a credit card number `ccnum` (16 digits) and expiration date `expdate`. Expiration dates usually look like MM/YY, but to make this simpler so you can use a more apparent data type, it is safe to assume that the card expires at midnight (00:00) on the 1st of the month.
3. Each user must have an `email` address. Assume an email address length is at most 100.

We need a table to represent a `trip`. To keep it simpler, we will include start and end information in this table, but the end of trip information may be missing. Each trip is associated with:

1. a unique identifier `trip_id`, a number. Assume that the total number of rides is not small.
2. exactly one user `user_id` and exactly one scooter `scooter_id`.
3. a `start_time` and `end_time`, which includes the date.
4. a `pickup` and `dropoff` location as a GPS coordinate (a latitude/longitude pair). *Hint:* See the documentation here. Note that latitude and longitude together form a point on a Cartesian plane (actually a sphere, but we will assume Cartesian plane for this problem).

Exercise. Write the SQL schema for the tables discussed above using `CREATE TABLE`. Specify a primary key, or composite primary key using the correct syntax. Specify the proper foreign key relationship on each table (if one exists) using the proper syntax. Try to minimize storage space because we can always promote later.