University of California, Los Angeles
Department of Computer Science

Computer Science 143 **Prof. Ryan Rosario**

Homework 1

Due Wednesday, April 18, 2018 11:59pm ***via CCLE***

Please remember the following:

1. Homework is mostly graded on completion. We may grade a few parts, but it will never be the majority of the grade on the assignment. So try your best, and focus on solving the problems. Consider homework (and studying the solutions) as practice for the midterm.
2. Homework must be submitted digitally, on CCLE. We will not do any paper grading. You can use a text file, but if you use Word, a PDF is preferred rather than a DOC file.
3. If there are any exercises that are difficult to do digitally (such as diagrams or math), consider scanning your drawing or math, or using a graphics program (even a readable MS Paint is fine) or Equation Editor.
4. Solutions will be posted.

Part 1: Schemas and Architecture

Suppose you are working for the data team at *Bird Scooter*, a Santa Monica based startup that aims to revolutionize how people get around on wheels.

How the Bird Scooter service works: a user installs an app on their phone and enters their credit card information. The app shows a map of deployed scooters nearby. The user scans a QR code on the scooter using the app, which activates the scooter for use and begins the clock for per-minute and per-use billing. The user rides the scooter for a distance, for a certain number of minutes. When the user is done with the scooter, he/she leaves it somewhere, and marks the trip as complete in the app.

Each scooter has an identifier, and assume that since Bird is a startup, it only has 10,000 scooters deployed. Each scooter also has a flag that specifies a scooter as online (deployed), offline (not in use for whatever reason), or lost/stolen. Finally, each scooter is assigned to a home location that rarely changes. For example, some scooters may be assigned to UCLA, some may be assigned to Santa Monica, and others to Austin. Occasionally, a Santa Monica scooter may be reassigned to UCLA depending on demand, but we expect that such a change is very rare.

Assume that Bird currently only 500,000 registered users. A user is someone that has installed the app. Each user has an identifier, but not all users have a credit card number on file as many users install the app and then never ride a scooter. There is also an expiration date [present if a credit card number is present] and an email address.

Assume, for simplicity, that the app communicates over the Internet directly to a database server. Being a startup, this is probably how it is done, actually.

Exercises

(a) Develop a schema for the `scooter` and `user` table based on the requirements and use case described on the previous page. Write the schema as a `CREATE TABLE` statement. Specify a primary key, or composite primary key using the correct syntax. Mark (with a comment) which attributes, if any, are foreign keys (to determine this, you may have to answer the other parts first). Try to minimize storage space.

(b) Each interaction between a user and a Bird scooter is called a *trip.* and we will create a schema for this `trip` table. Assume each trip has a unique identifier. The app will use this database table to determine where an available scooter is located. It will also (somehow) be used to determine the amount to charge the user. Additionally, data scientists would like to be able to use this table to determine daily and hourly trends in when users start and end scooter use and also identify scooter hotspots: areas where people frequently activate scooters and park them (just assume a location is a GPS coordinate: latitude and longitude, which can be represented numerically). Write the `CREATE TABLE` statement for this table. Specify the primary key. Mark (with a comment) which attributes, if any, are foreign keys (to determine this, you may have to answer the other parts first). Try to minimize storage space.

(c) For the `trip` table, there are two ways that we can write data to the table from the app. First, we could insert a row when the user activates the scooter, and then modify the row when the user parks the scooter and ends the session. Second, we can simply cache the ride data on the phone, and at the end of the session transmit this data to the database server to be inserted as a row. What are the advantages and disadvantages of both of these methods? Which would you (an employee at Bird) prefer and why? Is there an even better way?

(d) Using the tables you just developed in all of Part 1, draw the schema diagram. For an example of a schema diagram, see Figure 2.8 in the text (page 47), or page 23 in the lecture slides for Lecture 2.

Some other things to think about:

- Would you include the number of minutes the trip lasted in the `trip` table? Why or why not?
- What are the advantages and disadvantages of including the `charge` to the user in the `trips` table?

Part 2: SQL and Relational Algebra

If you wish to check your syntax, you can load this dataset into MySQL by following the directions on CCLE under Homework 1.

Hints: For some of these queries, you will need to use functions on attributes. Check out the list of date and time functions here, as it should be very useful: `https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html`. **You do not need to memorize them!** Note that these 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.

Bay Area Rapid Transit (BART) is a subway system that stops at various places on the Bay Area Peninsula, City of San Francisco, underwater to Oakland and the East Bay. When a user inserts a ticket, or scans a pass card on a turnstile, BART records that a user entered the subway system. On exit, the passenger does the same thing to exit the station (inserts a ticket or scans a fare card) and BART records that the user's journey is complete. Throughput can be defined as the number of passengers that entered at origin A, and exited at destination B.

In this exercise, we will do various queries on this data to answer several interesting questions.

This dataset consists of BART ridership data from 2017. The schema for the two tables in this database are provided below in case you do not wish to use the data.

```
-- This table is for your own information to see which station is which.
CREATE TABLE station(
    Abbreviation CHAR(4),
    Description VARCHAR(1000),
    Location VARCHAR(23),
    Name VARCHAR(50)
);

CREATE TABLE rides2017(
    Origin CHAR(4),
    Destination CHAR(4),
    Throughput INT,
    DateTime DATETIME
);
```

Exercises

(a) Write a query to compute the total throughput (passengers, or number of trips) by *time of day* for the year of 2017. The result should contain only the hour of day, named `hour`, and the number of trips named `trips`.

(b) Write a query that lists the one pair of station codes that had the largest throughput on the weekdays in 2017.

(c) Write a query that returns the 5 destinations that saw the highest average throughput on Mondays between 7am and 10am, and rank them from highest to lowest. Return the destinations and their averages.

(d) It is 2018. Suppose you are working with BART to expand stations and services based on 2017 ridership data. You want to know which stations are *overcrowded*. Suppose we consider an originating station as overcrowded if the maximum hourly throughput across all one-hour periods in the year is greater than 100 times the average hourly throughput across all one-hour periods in the year. Write a query that retrieves all originating stations that were overcrowded in 2017. (Passengers spend more time in originating stations because they must wait for the train there.)

(e) Suppose we take the **result** from part (a) and call it `hourly_ridership`. Given the following query, write the equivalent expression in the relational algebra.

```
SELECT
    hour,
    trips / 100
FROM hourly_ridership
WHERE (hour >= 7 AND hour < 10) OR (hour >= 17 AND hour < 19);
```

(f) Suppose we want to study how the weather affects how busy particular stations are. In the `Occupancy` relation, we have the name of a station called `Station`, the `DateTime`, and the number of people passing through the station called `Riders`, as attributes. In the `Weather` relation, we have the `Station`, `DateTime`, `Temperature`, `Condition` (for simplicity assume a string, like "cloudy") attributes. We are *only* interested in comparing occupancy during "sunny" hours and "rainy" hours, and we only care about `Station`, `DateTime`, `Riders` and `Condition`. Write an expression in the relational algebra that represents this context.