University of California, Los Angeles
**Department of Computer Science**

**Computer Science 143** **Prof. Ryan Rosario**

# Homework 3

*Due Friday, April 30, 2021 at 11:59pm* ***via CCLE***

**Please submit *two* files: one with your written answers to Parts 1abc and 2, and a *separate* file named `hw3.py` for Part 1d. Your script must be runnable from the command line by running `python3 hw3.py` and your code must print your answer (only the answer) to `stdout`. The `print` function is fine for this.**

Note that there are a few key differences in this assignment compared to the previous ones:

1. We have provided the expected output of the queries in an attempt decrease the confusion by the wording. **You still must provide the output from your query so we can verify that you ran it on your own machine.** Copying/pasting the expected output as your own output will not receive credit.
2. Due to the condition of some of the queries we have seen, we are loading the data into the `public` schema so you will not have to prefix everything with the schema name.

## Part 1: Joins and Subqueries

**Hints:** For some of these queries in Part 1, you will need to use functions on attributes. Check out the list of date and time functions here, as it should be very useful. In addition to what we discussed in lecture, you may find the functions `EXTRACT`, `EPOCH`, `CEILING` to be useful.

For these exercises we will write some queries to determine how much to charge Bird Scooter users for each ride they take. The tarball for HW3 contains some simulated (fake) data that you can use to test the syntax of your queries. You may notice that the the data types for these simulated relations are most `varchar` and there are no primary keys specified. This is so we do not give away the schema for those still completing HW2. We have the following relations:

```
customer(user_id, ccnum, expdate, email)
trip_start(trip_id, user_id, scooter_id, time, lat, lon)
trip_end(trip_id, user_id, scooter_id, time, lat, lon)
```

(a) Write a query that computes two new columns: (1) the elapsed time (in minutes) of each trip, and (2) the total cost of the trip. The trip charge is computed as follows: $1 flat rate for each trip plus $0.15 per minute. Fractional minutes should be rounded up (ceiling), so 4.02 minutes becomes 5 minutes. If the trip does not have an end time (scooter was stolen etc.), the length of the trip shall be 24 hours (1440 minutes) and the user should be charged based on 24 hours of use. Your results should include the `trip_id`, `user_id`, `trip_length` and `trip_cost`. There are at least two ways to do this problem. We prefer the method with the subquery and/or join. Compute the length of each trip first, and then compute the cost. Order the results by `trip_id` in ascending order. **Also, submit the top 10 rows of your output, without any special ordering.**

You should get the following output. Pay attention to the `trip_cost` column.

| user_id | trip_id | trip_length | trip_cost |
|---:|---:|---:|---:|
| 20685 | 0 | 2 | 1.30 |
| 34808 | 2 | 3 | 1.45 |
| 25463 | 3 | 1440 | 217.00 |
| 26965 | 4 | 2 | 1.30 |
| 836 | 5 | 1 | 1.15 |
| 3260 | 6 | 5 | 1.75 |
| 13850 | 7 | 3 | 1.45 |
| 23528 | 9 | 4 | 1.60 |
| 35829 | 11 | 1440 | 217.00 |
| 18494 | 12 | 1440 | 217.00 |

(b) Modify your query so that it computes the *total* amount that each user has spent on Bird Scooter. Again, use your previous response and assume that we did not store the intermediate result from the previous part. Report the `user_id` and the total amount spent as `total_spent`. Sort by `user_id` in ascending order. Be careful here.

You should get the following output:

| user_id | total_spent |
|---:|---:|
| 0 | 662.00 |
| 1 | 8.25 |
| 2 | 674.90 |
| 3 | 14.80 |
| 4 | 885.10 |
| 5 | 445.30 |
| 6 | 17.70 |
| 7 | 11.15 |
| 8 | 233.40 |
| 9 | 666.65 |

(c) In class, we learned several ways to classify joins. List all of the adjectives we can use to describe this join. Your choices are:

Inner Outer
Left Right Full
Equi Non-equi
Natural Self

(d) (Please submit this as a separate file `hw3.py`): Write a Python program that connects to the database, executes the query you wrote in the previous part, and produces a report that looks like the following. The format does not need to be identical, just close enough to show that you can use the output of the database and do something with it.

```
BIRD SCOOTER
User Charges for 2021

User ID       Charge
-----------   -----------
0              $ 662.00
1              $ 8.25
etc...
```

**Hint:** Practically all of the code is already written for you in the Lecture 8 slides. All you need to do is substitute in your query, eliminate the query parameter from my example, and then write some code to make the final computation.

## Part 2: Views and Authorization

These are examples of short answer/short essay questions that may appear on the exam.

**Exercises.**

(a) In lecture, we discussed that RDBMS often grant authorizations based on users or *roles*, which are groups, and a user may be a member of zero or more groups. Suppose we have a role called `manager` and user `alice` has this role. First, how would `alice` and `manager` be represented in the authorization graph? How would the privileges `INSERT` and `SELECT` be represented?

(b) Alice can also grant privileges to other users of a database (`WITH GRANT OPTION`), and so can `manager`. But why would it be better for the granting to be done by the `manager` role rather than the `alice` user? Think in terms of the authorization model.

(c) Explain some conditions when a standard `VIEW` cannot be made updatable. Why do you think that is?