**MySQL & Python Ride Share App**

**CPSC408**

***PART 1***

Suppose that a new start-up is trying to create a rideshare app and hires you to design their database. Think about the main information that the app must keep track of and create some deliverables to show your client as you go. You must create (1) an ER diagram (2) a fully filled out schema (3) create the database on MySQL locally (4) add some data to test it (5) create a simple interactive python program.

For the interactive part:

* The user should be able to give their ID and you will determine whether they are a driver or a user (cannot be both for simplicity)
* If they are a driver, you should give them the following options:
  + turn drive mode on: updates a flag on their file that they are able to drive
* if they are a rider, you should give them the following options:
  + find a driver: matches them with a driver that has their drive mode on activated
    - the rider will provide the following info:
      * pick up location
      * drop off location
    - and you will provide the rider with a ride ID
    - then it will go back to main menu
  + rate my driver:
    - the rider will provide their own ID and desired rating
      * You should look up the rider’s last ride and get the driver’s ID
      * Then, calculate the driver’s new rating by taking their current rating + their new rating and dividing by 2.

***Keep in mind you must include at least enough attributes to be able to write the interactive portion so make sure you read the instructions first!***

***PART 2***

We will be utilizing MySQL and Python along with all of our prior knowledge on relational databases to build our first db from scratch. You will first need to download MySQL by following the instructions that have been up on Canvas. Then, you will be ready to start the following steps. MAKE SURE YOU REMEMBER YOUR MYSQL PASSWORD!

***Connecting Python to MySQL locally:***

import mysql.connector

mydb = mysql.connector.connect(host="localhost",

user="root",

password="Password!",

auth\_plugin='mysql\_native\_password')

print(mydb)

mydb.close()

* This should print something like:

<mysql.connector.connection.MySQLConnection object at 0x7f8a94f70430>

* If you get an error, it is most likely because you used the wrong password.
* Now we are ready to create a database.

***Creating the DB:***

* Write this code between print(mydb) and mydb.close()
* We first have to create a cursor object that will let us run SQL commands.

# create cursor obj to interact with mySQL

mycursor = mydb.cursor()

# create the DB

mycursor.execute("CREATE SCHEMA RideShare;")

# show the databases that exist in my local mySQL

mycursor.execute("SHOW DATABASES")

for x in mycursor:

print(x)

* You should get some pre-made DBs printed along with the new DB you have created <RideShare>.
* Now edit your connection code to add the following: database="RideShare"

import mysql.connector

mydb = mysql.connector.connect(host="localhost",

user="root",

password="Password!",

auth\_plugin='mysql\_native\_password',

**database="RideShare")**

print(mydb)

mydb.close()

***Adding the schema (tables):***

* Use the usual SQL commands for adding tables inside of an execute statement.

mycursor.execute("CREATE TABLE tableName (a1 dataType, a2 dataType)”)

***Adding data:***

* if you have a list of tuples with your values :

sql = "INSERT INTO tableName (a1, a2) VALUES (%s, %s)"

vals = [

('v1', 'v2),

('v1', 'v2),

('v1', 'v2),

('v1', 'v2)]

mycursor.executemany(sql, val)

mydb.commit()

print(mycursor.rowcount, "was inserted.")

* use mycursor.execute(sql,val) for one single row of values

***To use select statements:***

mycursor.execute("SELECT \* FROM customers")

myresult = mycursor.fetchall()

for x in myresult:

print(x)

***NOTE: For DDL your changes will happen by just using .execute() whereas for DQL and DML you must first use .commit() then .execute()***

***PART 3***

***THE HOMEWORK PORTION***

You have now finished implementing the client’s original requirements and presented the product to them. However, they have realized they want extra features for their app.

You should implement the following:

* When the drivers log in and change their status to available, they should be matched with a rider who is requesting a ride (keep track of riders’ status).
  + The driver will be given the following info about the ride:
    - pick up location
    - drop off location
    - star rating of the rider requesting the ride
  + The driver should have the option to accept or deny a ride
    - if they accept then the rider gets matched with the ride and the driver’s and rider’s status change to inactive so that they don’t get matched again.
    - if they deny change the driver’s status to inactive.
  + to test this implementation, you can run the code in a few different terminals at the same time. One acting as a driver and one or two acting as riders.
* implement the same star rating system for drivers as you did for riders (drivers should be able to rate their last passenger)
* ***rider requests ride and gives details***
* ***program waits until there Is a driver available (you can use sleep() for a few mins so you have time to run code from the driver’s terminal)***
* ***driver turns their driving mode on***
* ***they are matched with the rider***
* ***they accept or deny***
* ***if they accept give driver details to the rider***
* ***if not then tell the rider you couldn’t find a ride and turn driver mode off***