CSE 6242 / CX 4242: Data and Visual Analytics | Georgia Tech | Fall 2020

Homework 3: Spark, Docker, DataBricks, AWS and GCP

By our 32+ awesome TAs of CSE6242A,Q,OAN,O01,O3/CX4242A for our 1200+ students

Submission Instructions and Important Notes

It is important that you carefully read the following instructions and those about deliverables at the end of each question, or you may lose points.

1. Always check to make sure you are using the **most up-to-date assignment** (version number at bottom right of this document).
2. Submit a single zipped file, called "HW3-GTusername.zip" that unzips to a folder called "HW3-GTusername", containing all the deliverables including source code/scripts, data files, and readme. Example: "HW3-jdoe3.zip" if GT account username is "jdoe3". Your GT username is the one with letters and numbers. Only .zip is allowed; no other format will be accepted.
    a. At the end of this assignment, we have specified a folder structure you must use to organize your files. 5 points will be deducted for not following this strictly.
    b. Due to the large class size, we may need to use auto-grading scripts to grade some of your deliverables, to help speed up grading, so we can return graded work to you sooner. Thus, it is extremely important that you strictly follow the instructions.
    c. Do not include any intermediate files you may have generated while working on the task, unless your work is absolutely dependent on it to get the final result — there are rarely any situations that would justify such a need.
    d. Wherever you are asked to write down an explanation for the task you perform, stay within the word limit or you may lose points.
3. You may discuss high-level ideas with other students at the "whiteboard" level (e.g., how cross validation works, use hashmap instead of array) and review any relevant materials online. However, each student must write up and submit his or her own answers.
4. All incidents of suspected dishonesty, plagiarism, or violations of the Georgia Tech Honor Code will be subject to the institute's Academic Integrity procedures (e.g., reported to and directly handled by the Office of Student Integrity (OSI)). Consequences can be severe, e.g., academic probation or dismissal, grade penalties, a 0 grade for assignments concerned, and prohibition from withdrawing from the class**.**
5. You should submit your work by the official **Due** date on Canvas, the same date specified on the course schedule.
    a. Every homework assignment deliverable comes with a 48-hour "grace period". You do **not** need to ask before using this grace period.
    b. You may re-submit your work before the grace period expires **without penalty**, but Canvas will mark your submission as **"late"**.
    c. Canvas automatically appends a "version number" to files that you re-submit. You do not need to worry about these version numbers, and there is no need to delete old submissions. **We will only grade the most recent submission.**
    d. Any deliverable submitted after the grace period will get **0 credit**. We recommend that you submit your work before the grace period begins.
    e. We will **not** consider late submission of any missing parts of a deliverable. To make sure you have

submitted everything, download your submitted files to double check. If your submitting large files, you are responsible for making sure they get uploaded to the system in time. You have 48 hours to verify your submissions!

===== DO THIS NOW: Create AWS Educate account =====

You will receive an email from support@awseducate.com. Follow the **AWS Setup Tutorial** here to **create your account**

EXTREMELY IMPORTANT REMINDERS:

a. Check your "spam" email folders for emails that you may be receiving from AWS. Many students reported emails get pushed to those folders automatically.
b. Shut down everything (YES! EVERYTHING!) when you're done with the instances (do not leave them on over weekends/holidays!). Highest record from previous classes went above $2000! As AWS Educate starter accounts have spending cap enabled by default, this "overspending" should no longer be possible. In case it happens, the good news is you can call AWS to explain the situation and they should be able to waive the charges -- call (phone) AWS customer care immediately (not email) so you can resolve the issue quickly (usually within minutes).

==========

Download the HW3 Skeleton before you begin.

Grading

You can score a maximum of 100 points in this assignment.

Homework Overview

Many modern-day datasets are huge and truly exemplify "big data". For example, the Facebook social graph is petabytes large (over 1M GB); every day, Twitter users generate over 12 terabytes of messages; and the NASA Terra and Aqua satellites each produce over 300 GB of MODIS satellite imagery per day. These raw data are far too large to even fit on the hard drive of an average computer, let alone to process and analyze. Luckily, there are a variety of modern technologies that allow us to process and analyze such large datasets in a reasonable amount of time. For the bulk of this assignment, you will be working with a dataset of over 1 billion individual taxi trips from the New York City Taxi & Limousine Commission (TLC). Further details on this dataset are available here.

In Q1, you will work with a subset of the TLC dataset to get warmed up with PySpark. Apache Spark is a framework for distributed computing, and PySpark is its Python API. You will use this tool to answer questions such as "what are the top 10 most common trips in the dataset"? You will be using your own machine for computation, using an environment defined by a Docker container.

In Q2, you will perform further analysis on a different subset of the TLC dataset using Spark on DataBricks, a platform combining datasets, machine learning models, and cloud compute. This part of the assignment will be completed in the Scala programming language, a modern general-purpose language with a robust support

for functional programming. The Spark distributed computing framework is in fact written using Scala.

In Q3, you will use PySpark on AWS using Elastic MapReduce (EMR), and in Q4 you will use Spark on Google Cloud Platform, to analyze even larger samples from the TLC dataset.

Finally, in Q5 you will use the Microsoft Azure ML Studio to implement a regression model to predict automobile prices using a sample dataset already included in the Azure workspace. A main goal of this assignment is to help students gain exposure to a variety of tools that will be useful in the future (e.g., future project, research, career). The reasoning behind intentionally including AWS, Azure and GCP (most courses use only one), because we want students to be able to try and compare these platforms as they evolve rapidly. This will help the students in the future should they need to select a cloud platform to use, they can make more informed decisions and be able to get started right away.

You will find that a number of computational tasks in this assignment are not very difficult, and there seems to be quite a bit of "setup" to do before getting to the actual "programming" part of the problem. The reasoning behind this design is because for many students, this assignment is the very first time they use *any* cloud services; they are new to the pay-per-use model, and they have never used a cluster of machines. There are over 1000 students in CSE 6242 (campus and online) and CX 4242 combined. This means we have students coming from a great variety of backgrounds. We wished we could provide every student unlimited AWS credit so that they can try out many services and write programs that are more complex. Over the past offering of this course, we have been gradually increasing the "programming" part and reducing much of the "setup" (e.g., the use of Docker, Databricks and Jupyter notebooks were major improvements). We will continue to further reduce the setup that students need to perform in future offerings of this course.

Q1 [15 points] Analyzing trips data with PySpark

Follow these instructions to download and setup a preconfigured Docker image that you will use for this assignment.

> **Why use Docker?** In earlier iterations of this course, students installed software on their own machines, and we (both students and instructor team) ran into all sorts of issues, some of which could not be resolved satisfactorily. Docker allows us to distribute a cross platform, preconfigured image with all of the requisite software and correct package versions. Once Docker is installed and the container is running, access Jupyter by browsing to http://localhost:6242. There is no need to install any additional Java or PySpark dependencies as they are all bundled as part of the Docker container.

Imagine that your boss gives you a large dataset which contains trip information of New York City Taxi and Limousine Commission (TLC). You are asked to provide summaries for the most common trips, as well as information related to fares and traffic. This information might help in positioning taxis depending on the demand at each location.

You are provided with a Jupyter notebook (q1.ipynb) file which you will complete using PySpark using the provided Docker image.

Tasks

You will use the yellow_tripdata_2019-01_short.csv dataset. This dataset is a modified record of the NYC Green Taxi trips and includes information about the pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, fare amounts, payment types, and driver-reported passenger counts. When processing the data or performing calculations, **do not round any values**. Download the data here.

a. **[1 pt]** You will be modifying the function `clean_data` to clean the data. Cast the following columns into the specified data types:

a. `passenger_count` - integer
b. `total_amount` - float
c. `tip_amount` - float
d. `trip_distance` - float
e. `fare_amount` - float
f. `tpep_pickup_datetime` - timestamp
g. `tpep_dropoff_datetime` – timestamp

b. **[4 pts]** You will be modifying the function `common_pair`. Find the top 10 pickup-dropoff location pairs having the highest number of trips (`count`). The location pairs should be ordered by `count` in descending order. If two or more pairs have the same number of trips, break the tie using the trip amount per distance travelled (`trip_rate`). Use columns `total_amount` and `trip_distance` to calculate the trip amount per distance. In certain situations, the pickup and dropoff locations may be the same (include such entries as well).

Output:

PULocationID	DOLocationID	Count	trip_rate
1	2	23	5.242345
3	3	5	6.61345634

c. **[4 pts]** You will be modifying the function `time_of_cheapest_fare`. Divide each day into two periods: Day (from 9am to 8:59:59pm, both inclusive), and Night (from 9pm to 8:59:59am, both inclusive). Calculate the average total amount per unit distance travelled (use column `total_amount`) for both time periods. Sort the result by `trip_rate` in ascending order to determine when the fare rate is the cheapest.

Output:

day_night	trip_rate
Day	4.2632344561
Night	6.42342882

d. **[4 pts]** You will be modifying the function `passenger_count_for_most_tip` . Filter the data for trips having fares (`fare_amount`) greater than $2 and the number of passengers (`passenger_count`) greater than 0. Calculate the average fare and tip (`tip_amount`) for all passenger group sizes and calculate the tip percent (`tip_amount * 100 / fare_amount`). Sort the result in descending order of tip percent to obtain the group size that tips the most generously.

Output:

passenger_count	tip_percent

2	14.22345234
1	12.523334576
3	12.17345231

e. **[3 pts]** You will be modifying the function `day_with_traffic` . Sort the days of the week in descending order of traffic (day having the highest traffic should be at the top). Calculate traffic for a particular day using the average speed of all taxi trips on that day of the week. Calculate speed using the trip time and trip distance, as distance per hour. If the `average_speed` is equal for multiple days, order the days alphabetically. A day with low average speed indicates high levels of traffic. The average speed may be 0 indicating very high levels of traffic. Not all days of the week may be present in the data (Do not include these missing days of the week in your output). Use `date_format` along with the appropriate pattern letters to format the day of the week such that it matches the example output below.

Output:

day_of_week	average_speed
Fri	0.953452345
Mon	5.2424622
Tue	9.23345272

Deliverables:

[15 points] q1.ipynb the notebook for the given question with your code

a) **[1 pt]** `clean_data` to clean the data
b) **[4 pts]** `common_pair` to find the top 10 pickup-dropoff pairs
c) **[4 pts]** `time_of_cheapest_fare` to find when the trips are cheapest (Day vs Night)
d) **[3 pts]** `passenger_count_for_most_tip` to find which group size tips most generously
e) **[3 pts]** `day_with_traffic` to find which day has the highest traffic (slowest trips)

Q2 [30 pts] Analyzing dataset with Spark/Scala on Databricks

Tutorial

First, go over this Spark on Databricks Tutorial, to learn the basics of creating Spark jobs, loading data, and working with data.

You will analyze nyc-tripdata.csv[1] using Spark and Scala on the Databricks platform. (A short description of how Spark and Scala are related can be found here.) You will also need to use the taxi zone lookup table using taxi_zone_lookup.csv that maps the location ID into the actual name of the region in NYC. The nyc-tripdata dataset is a modified record of the NYC Green Taxi trips and includes information about the pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, fare amounts, payment types, and driver-reported passenger counts.

[1] Graph derived from the NYC Taxi and Limousine Commission

VERY IMPORTANT

1. Use only **Firefox, Safari or Chrome** when configuring anything related to Databricks. The setup process has been verified to work on these browsers.
2. **Carefully** follow the instructions in the Databricks Setup Guide. Open the link in a private browser window if you get 'Permission Denied' message.
    a. You **must** choose the Databricks Runtime (DBR) version as "**6.6 (includes Apache Spark 2.4.5, Scala 2.11)**". We will grade your work using this version.
    b. You **must not** choose the default DBR version of >= 7.2
    c. Note that you do not need to install Scala or spark in your local machine. They are provided with the DBR environment.
3. **You must use only** Scala DataFrame operations for this question. Scala DataFrames are just another name for Spark DataSet of rows. You can use DataSet API in Spark to work on these DataFrames. Here is a Spark document that will help you get started on working with DataFrames in Spark. You will lose points if you use SQL queries, Python, or R to manipulate a DataFrame.
    a. Refer to this link to understand how to avoid using other languages. After selecting the default language as SCALA, do not use the language magic %<language> with other languages like %r, %python, %sql etc. The language magics are used to override the default language, which you **must not** do for this assignment.
    b. You **must not** use full SQL queries in lieu of the Spark DataFrame API. That is, you **must not** use functions like *sql()*, which allows you to directly write full SQL queries like spark.sql ("SELECT* FROM col1 WHERE …"). This should be df.select("*") instead.
4. The provided template Scala notebook *q2.dbc* (in hw3-skeleton) has provided you with code that reads in a data file *nyc-tripdata.csv.* The input data is loaded into a DataFrame, inferring the schema using reflection (Refer to the Databricks Setup Guide above). It also contains code that **filters the data** to only keep the rows where the pickup location is different from the drop location, and the trip distance is strictly greater than 2.0 (>2.0).
    a. All tasks listed below **must** be performed on this filtered DataFrame.
    b. Carefully read the instructions in the notebook, which include hints for solving the problems.

Tasks

a) List the top-5 most popular drop locations for:
    i. **[2 pts]** "DOLocationID", sorted in descending order. If there is a tie, then one with a lower "DOLocationID" gets listed first.
    ii. **[2 pts]**"PULocationID", sorted in descending order. If there is a tie, then one with a lower "PULocationID" gets listed first.

b) **[4 pts]** List the top-3 locationID's with the maximum overall activity. Here number of overall activity in a LocationID is simply the sum of all pickups and all drop offs at that LocationID. In case of a tie, the lower LocationID gets listed first.

c) **[4 pts]** List all the boroughs (of NYC: Manhattan, Brooklyn, Queens, Staten Island, Bronx along with "Unknown" and "EWR") in the order of having the highest to lowest number of activities (here number of activities means the sum of all pickups and all drop offs at that LocationID), together with the total number of activity counts for each borough in NYC.

d) **[5 pts]** List the top 2 days of week with the largest number of (daily) average pickups, along with the values of average number of pickups on each of the two days. The day of week should be a string with its full name, for example, "Monday" instead of a number 1 or "Mon".

e) **[6 pts]** For each particular hour of a day (0 to 23, 0 being midnight) - in their order from 0 to 23 (inclusive), find the zone in Brooklyn borough with the LARGEST number of pickups.

f) **[7 pts]** Find which 3 different days of the January, in Manhattan, saw the largest percentage increment in pickups compared to previous day, in the order from largest increment % to smallest increment %.

List the results of the above tasks in the provided **q2_results.csv** file under the relevant sections. **These pre-formatted sections also show you the required output format from your Scala code with the necessary columns --- while column names can be different, their resulting values must be correct.**

- You must **manually enter** the output generated into the corresponding sections of the *q2_results.csv* file, preferably using some spreadsheet software like MS-Excel (but make sure to keep the csv format). For generating the output in the Scala notebook, refer to `show()` and `display()` functions of Scala.
- Note that you can edit this csv file using text editor, but please be mindful about putting the results under designated columns.

Do not modify anything other than filling in those required output values on this csv file. **We grade by running the Spark Scala code you write and also by looking at your results listed in this file**. So, make sure that your output is actually obtained from the Spark Scala code you write.

Hint: You may find some of the following DataFrame operations helpful:
`toDF, join, select, groupBy, orderBy, filter, agg, Window(), partitionBy, orderBy etc.`

Deliverables

VERY IMPORTANT: Rename your notebook as **q2_yourGTusername** (e.g., q2_jdoe3) and export your solution in the three formats: .dbc, .scala and .html.

- **q2_yourGTusername.dbc:** Your solution as Scala Notebook archive file (.dbc) exported from Databricks. See the Databricks Setup Guide on creating an exportable archive for details.
- **q2_yourGTusername.scala**: Your solution as a Scala source file exported from Databricks. See the Databricks Setup Guide on creating an exportable source file for details.
- **q2_yourGTusername.html**: Your solution as a HTML file exported from Databricks. See the Databricks Setup Guide on how to export your code as an HTML.

q2_results.csv: The output results from your Scala code in the Databricks q2 notebook file. You must carefully copy the outputs of the *display()*/*show()* function into a file titled ***q2_results.csv*** under the relevant sections. Please double check and compare with your actual output with the results you copied.

Q3 [35 points] Analyzing Large Amount of Data with PySpark on AWS

VERY IMPORTANT: Use Firefox, Safari or Chrome when configuring anything related to AWS.

You will try out PySpark for processing data on Amazon Web Services (AWS). Here you can learn more about PySpark and how it can be used for data analysis. You will be completing a task that may be accomplished using a commodity computer (e.g., consumer-grade laptops or desktops). However, we would like you to use this exercise as an opportunity to learn distributed computing on Amazon EC2, and to gain experience that will help you tackle more complex problems.

The services you will primarily be using are Amazon S3 storage, Amazon Elastic Cloud Computing (EC2) virtual servers in the cloud, and Amazon Elastic MapReduce (EMR) managed Hadoop framework. You will be creating an S3 bucket, running code through EMR, then storing the output into that S3 bucket.

For this question, you will only use up a very small fraction of your AWS credit.

AWS Guidelines

Please read the AWS Setup Tutorial to set up your AWS account. Instructions are provided both as a written guide, and a video tutorial.

Datasets

In this question, you will use a dataset of trip records provided by the New York City Taxi and Limousine Commision (TLC). Further details on this dataset are available here and here. From these pages [1] [2], you can explore the structure of the data, however you will be accessing the dataset directly through AWS via the code outlined in the homework skeleton. You will be working with two samples of this data, one small, and one much larger.

EXTREMELY IMPORTANT: Both the datasets are in the **US East (N. Virginia)** region. Using machines in other regions for computation would incur data transfer charges. Hence, set your region to **US East (N. Virginia)** in the beginning (not Oregon, which is the default). **This is extremely important, otherwise your code may not work, and you may be charged extra.**

Goal

You work at NYC TLC, and since the company bought a few new taxis, your boss has asked you to locate potential places where taxi drivers can pick up more passengers. Of course, the more profitable the locations are, the better. Your boss also tells you not to worry about short trips for **any** of your analysis, so only analyze trips which are **2 miles or longer**.

First, find the **20** most popular drop off locations in the Manhattan borough by finding which of these destinations had the greatest **passenger count**.

Now, analyze all pickup locations, regardless of borough.

- For each pickup location determine
  - the **average total amount** per trip,
  - the total **count** of all trips that start at that location, and

    - the **count** of all trips that start at that location and end at one of most popular drop off locations.
- Using the above values,
    - determine the **proportion** of trips that end in one of the popular drop off locations (# trips that end in drop off location divided by total # of trips) and
    - multiply that proportion by the **average total amount** to get a **weighted profit value** based on the probability of passengers going to one of the popular destinations.

Bear in mind, your boss is not as savvy with the data as you are and is not interested in location IDs. To make it easy for your boss, provide the **Borough** and **Zone** for each of the top 20 pickup locations you determined. To help you evaluate the correctness of your output, we have provided you with the output for the small dataset.

Note: Please strictly follow the formatting requirements for your output as shown in the small dataset output file. You can use https://www.diffchecker.com/ to make sure the formatting is correct. Improperly formatted outputs may not receive any points, as we may not know how to interpret them.

Tasks

You are provided with a python notebook (q3_pyspark.ipynb) file which you will complete and load into EMR. You are provided with the load_data() function, which loads two PySpark DataFrames. The first is **trips** which contains a DataFrame of trip data, where each record refers to one (1) trip. The second is **lookup** which maps a LocationID to its information. It can be linked to either the PULocationID or DOLocationID fields in the trips DataFrame.

The following functions must be completed for full credit.

VERY IMPORTANT

- Ensure that the parameters for each function remain as defined and the output order and names of the fields in the PySpark DataFrames are maintained.
- Do not import any functions which were not already imported within the skeleton.
- Use PySpark methods to complete this assignment, not SQL commands.

a) **[1 pts] user()**
    i. Returns your GT Username as a string (e.g., gburdell3)
b) **[2 pts] long_trips(trips)**
    i. This function filters trips to keep only trips longer than 2 miles.
    ii. Returns PySpark DataFrame with the same schema as **trips**
    iii. **Note: Parts c, d and e will use the result of this function**
c) **[6 pts] manhattan_trips(trips, lookup)**
    i. This function determines the top 20 locations with a *DOLocationID* in Manhattan by passenger count.
    ii. Returns a PySpark DataFrame with the schema (DOLocationID, pcount)
d) **[6 pts] weighted_profit(trips, mtrips)**
    i. This function determines
        i. the average *total_amount*,

ii. the *total count of trips*, and
iii. the *total count of trips ending in the top 20 destinations*
iv. and return the *weighted_profit* as discussed earlier in the homework document.
v. Returns a PySpark DataFrame with the schema (PULocationID, weighted_profit) for the *weighted_profit* as discussed earlier in this homework document.

ii.

e) **[5 pts] final_output(wp, lookup)**
   i. This function
      i. takes the results of *weighted_profit*,
      ii. links it to the *borough* and *zone* through the **lookup** data frame, and
      iii. returns the top 20 locations with the highest *weighted_profit*.
   ii. Returns a PySpark DataFrame with the schema (Zone, Borough, weighted_income)

Once you have implemented all these functions, run the main() function, which is already implemented, and update the line of code to include the name of your output s3 bucket and a location. **This function will fail** if the output directory already exists, so make sure to **change it each time** you run the function.

Example: `final.write.csv('s3://cse6242-bburdell3/output3.csv')`

Your output file will appear in a folder in your s3 bucket as a csv file with a name which is similar to *part-0000-4d992f7a-0ad3-48f8-8c72-0022984e4b50-c000.csv*. Download this file and **rename it to q3_output.csv** for submission. Do **not** make any other changes to the file.

Hint: Refer to commands such as filter, join, groupBy, agg, limit, sort, withColumnRenamed and withColumn.

Deliverables

1. **[20 pts] q3_pyspark.ipynb**: The PySpark notebook for the question (using the **larger** data set).

2. **[15 pts] q3_output.csv**: Output **(comma-separated)** (using the **larger** data set).

Note: Please strictly follow the guidelines below, otherwise your answer may not be graded.

1. Ensure that file names (case sensitive) are correct.
2. Ensure file extensions (.csv, .ipynb) are correct.
3. Double check that you are submitting the correct set of files --- we only want the script and output from the larger dataset. Also, double check that you are writing the right dataset's output to the right file.
4. You are welcome to store your script's output in any bucket you choose, as long as you can download and submit the correct files.
5. Do not make any manual changes to the output files

Q4 [10 points] Analyzing a Large Dataset using Spark on GCP

VERY IMPORTANT: Use Firefox, Safari or Chrome when configuring anything related to GCP.

GCP Guidelines

Instructions to setup GCP Credits, GCP Storage and Dataproc Cluster are provided as a video tutorials (here and here) and also written instructions.

Goal

The goal of this question is to familiarize you with creating storage buckets/clusters and running Spark programs on Google Cloud Platform. This question asks you to create a new Google Storage Bucket and load the NYC Taxi & Limousine Commission Dataset. You are also provided with a Jupyter Notebook (q4_pyspark-gcp.ipynb) file which you will load and complete in Google Dataproc Cluster. Inside the notebook, you are provided with the load_data() function, which you will complete to load a PySpark DataFrame from the Google Storage Bucket you created as part of this question. Using this PySpark DataFrame, you will complete the following tasks.

Tasks

a) **[1 pts]** Function load_data() to load data from Google Storage Bucket into Spark DataFrame
b) **[1.5 pts]** Function exclude_no_pickuplocations() to exclude trips with no pickup locations
c) **[1.5** pts] Function exclude_no_tripdistance() to exclude trips with no distance
d) **[2 pts]** Function include_fare_range() to include trips with fare between the range of $20 to $60 (including exact $20 and $60 fares)

After performing all the above operations on the data in sequence, perform the following functions on the **filtered data**.

e) **[2 pts]** Function get_highest_tip() to identify the highest tip in the filtered data. The Jupyter Notebook cell sequence automatically uses the filtered data
f) **[2 pts]** Function get_total_toll() to calculate the total toll amount for the filtered data. The Jupyter Notebook cell sequence automatically uses the filtered data

VERY IMPORTANT: you must first perform task a, b, c, and d in sequence, BEFORE performing task e and f.

You will use the data file yellow_tripdata_2019-01.csv. Each line represents a single taxi trip consisting of following comma separated columns. All columns are of string data type. You must convert the highlighted columns below into float data type when completing this question:

- VendorID
- tpep_pickup_datetime
- tpep_dropoff_datetime
- passenger_count
- **trip_distance (float data type)**
- RatecodeID
- store_and_fwd_flag
- PULocationID
- DOLocationID
- payment_type
- **fare_amount (float data type)**
- extra

11

- mta_tax
- **tip_amount (float data type)**
- **tolls_amount (float data type)**
- improvement_surcharge
- total_amount

Deliverables:

q4_pyspark-gcp.ipynb: The PySpark notebook for the question.

a) **[1 pts]** load_data() function to load data from google storage bucket into spark data frame
b) **[1.5 pts]** exclude_no_pickuplocations() function to exclude trips with no pickup location
c) **[1.5 pts]** exclude_no_tripdistance() function to exclude trips with no trip distance
d) **[2 pts]** include_fare_range() function to include only the trips with fare amounts between $20 to $60 (including exact $20 and $60 fares)
e) **[2 pts]** get_highest_tip() function to get the highest tip value
f) **[2 pts]** get_total_toll() function to calculate the total toll amount for the subset of this dataset

Q5 [10 points] Regression: Automobile price prediction, using Azure ML Studio

Note: Create and use a free workspace instance on Azure Studio. Please use your Georgia Tech username (e.g., jdoe3) to login.

Goal

Primary purpose of this question is to introduce you to Microsoft Azure Machine Learning Studio, familiarize you to its basic functionalities and typical machine learning workflows. Go through the "Automobile price prediction" tutorial and complete the following tasks.Tasks

You will manually modify the given file**q5_results.csv** by adding to it the results from the following tasks (e.g., using a plain text editor). Your solution will be autograded. Hence,

- DO NOT change the order of the questions.
- Report the exact numerical values that you get in your output, and DO NOT round any of them.
- When manually entering a value into the csv file, append it immediately after a comma, so there will be NO space between the comma and your value, and no trailing spaces after your value.

a) Update your GT username in the q5_results.csv file to replace gburdell3.
b) **[3 pts]** Repeat the experiment described in the tutorial and report values of all metrics as mentioned in the '*Evaluate Model'* section of the tutorial.

c) **[3 pts]** Repeat this experiment with a different value of '*Fraction of rows in the first output'* in the split module. Change the value to 0.8 from the originally set value, 0.75. Report corresponding values of the metrics.

d) **[4 pts]** Evaluate the model using 5-fold cross-validation (CV). Select parameters in the module *'Partition and sample'* (Partition and Sample) in accordance with the figure below. Report the values of Root Mean Squared Error (RMSE) and Coefficient of Determination for each of the five folds (1st fold corresponds to fold number 0 and so on).

To summarize, in order to complete sub question (d), you need to complete the following:

A. Import the entire dataset (Automobile Price Data (Raw))
B. Clean the missing data by dropping rows with missing values (Do not "exclude the normalized losses" from the original tutorial)
C. Partition and sample the data.
D. Create a new model: Linear Regression (add the default Linear regression i.e. do not change any values here)
E. Finally, perform cross validation on the dataset.
F. Visualize/report the values.

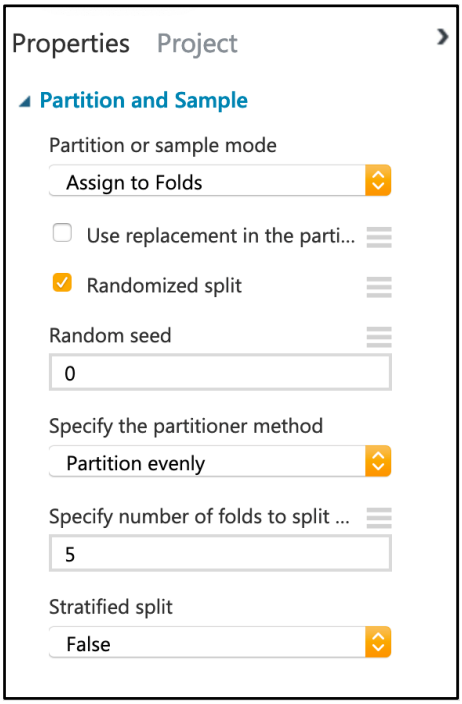

Figure: Property Tab of Partition and Sample Module

Hint: For part 4, follow each of the outline steps carefully. This should result in 5 blocks in your final workflow (including the Automobile price data (Raw) block).

Deliverables

1. **[10pt] q5_results.csv:** a csv file containing results for all of the three parts.

Extremely Important: folder structure & content of submission zip file

We understand that some of you may work on this assignment until just prior to the deadline, rushing to submit your work before the submission window closes. **Please take the time** to validate that **all files** are present in your submission and that you have not forgotten to include any deliverables! If a deliverable is not submitted, you will receive **zero** credit for the affected portion of the assignment — this is a very sad way to lose points, since you have already done the work!

You are submitting a single **zip** file **HW3-GTusername.zip** (e.g., HW3-jdoe3.zip).

The files included in each question's folder have been clearly specified at the end of the question's problem description.

The zip file's folder structure must exactly be (when unzipped):

```
HW3-GTUsername/
    Q1/
        q1.ipynb
    Q2/
        q2_yourGTusername.dbc
        q2_yourGTusername.scala
        q2_yourGTusername.html
        q2_results.csv
    Q3/
        q3_pyspark.ipynb
        q3_output.csv
    Q4/
        q4_pyspark-gcp.ipynb
    Q5/
        q5_results.csv
```