Spark Project

DS730

In this project, you will be running some Spark code on the Hortonworks system to solve several problems related to NYC taxi information. We will be interacting with Spark using Apache Zeppelin for Scala and on the command line with PySpark and SparkR.

I strongly encourage solving these problems using Scala as I imagine most of you are rather proficient with Python and/or R. If you are using Scala, create a new note in Zeppelin and call your note ProjectFive. Use spark2 as the default interpreter. Each problem should be self-contained. In other words, do not rely on question 1 being run before part 3, for example.

You have a taxi2018.csv file stored in an HDFS folder called /user/zeppelin/taxi/. Link to information about the data and schema can be found here:

https://data.cityofnewyork.us/Transportation/2018-Yellow-Taxi-Trip-Data/t29m-gskq

A few things to note about this dataset:

a. The file must be stored in the HDFS folder like this: `/user/zeppelin/taxi/taxi2018.csv`
b. This is a different file from the one you used in the activity.
c. The column headings are different from the file in the activity.
d. Your code may not be run on this exact dataset. However, the first line (i.e. the schema) will be identical, the formatting of the rows will be identical (e.g. a column that is an integer will still be an integer in the tested dataset) and the file will be in the same location as noted in step a.
e. You do not need to worry about time zones or ensuring that the conversion was done correctly for the location. All you need to use is the unix_timestamp function as shown in the activity for the questions involving time.

Answer the following questions and store your answers in a file called **sparkProjectOutput.txt**. There are no formatting requirements for the output or answers. As long as your output and answers are in a reasonable format, it is acceptable.

1. (5 pts) Not including the header row, how many rows are there in the file?

2. (10 pts) One might posit that a single passenger is the most common taxi ride. Which was more common: a single passenger_count or a passenger_count with more than 1 person?

3. (15 pts) It is possible that people who paid a toll went further than people who didn't pay a toll. What is the average trip_distance for people who had tolls_amount greater than 0? What is the average trip_distance for people who paid nothing in tolls?

4. (20 pts) What are the average fare_amounts for each month and which month has the highest average fare_amount? You should use the pickup date/time as the month to which a row belongs. You should take the sum of the fare_amounts and divide it by the total number of rows for that month. To ensure we have reliable data, you should filter out all rows where the **fare_amount** is less than or equal to 0. You should filter out all rows where the (fare_amount / trip_distance) is greater than 10,000. An obvious side-effect of this is to filter out all rows with a trip_distance of 0.

5. (15 pts) Who tips better: people who pay with a credit card before noon or people who pay with credit card from noon until the end of the day (i.e. 12:00:00pm - 11:59:59pm)? We will use the dropoff time to determine when a person paid. The payment_type has a numeric score of 1 if the person paid by a credit card. To figure out who tipped the best, take the sum of the tip_amount and divide it by the sum of the fare_amount. You should print out the (sum(tip_amount) / sum(fare_amount)) for each of the 2 requested groups.

6. (15 pts) What are the top ten worst rides with respect to time and distance? In other words, who sat in the taxi the longest and went the shortest distance? You should remove any ride whose trip_distance is 0. We want to maximize the following calculation:

   (number of seconds in the taxi / trip_distance)

   To ensure you are picking the correct row, your answer should include all information about the row (i.e. the entire row). Your answers must include a new column representing the above calculation. You only need to display 10 answers and do not need to worry about ranks.

7. (30 pts) Imagine that you work in NYC. You are considering working for a ride-sharing company to make a few extra dollars later in the evening. However, you don't have that much time to spend driving other people around. You realize you have 60 minutes that you can start your rides between 4:00:00pm and finish by 11:00:00pm[1]. You have determined that any total_amount that is over $200 is not going to be possible so those ought to be filtered out. You want to maximize the amount of money you could earn so you want to find the best 60 minute period between 4:00:00pm and 11:00:00pm, inclusive, to start your rides. You do not care about days. In other words, a ride that starts at 9:45:13pm on June 12th is in the same 60 minute time slot as a ride that starts at 9:45:13pm on August 15th.

   Your goal is this, find the 60 minute time slot where you maximize the average (mean) total_amount. Your 60 minute time slot answer is only considering rides starting (i.e. tpep_pickup_datetime) between 4:00:00pm and 11:00:00pm, inclusive. You should ignore the dropoff date/time as it is not important for this problem. You are only considering rides that are $200 or less. Your answer should go down to the second. In other words, your answer should be something similar to 9:47:15pm - 10:47:14pm[2]. Your entire 60 minute time slot must be between 4:00:00pm and 11:00:00pm, again, inclusive. Your answer should not go outside of these boundaries. Be sure that your code outputs your answer as a full 60 minute time slot like this: 6:36:29pm - 7:36:28pm. To ensure that we end up with the same answer, you should assume that all possible windows will start with tpep_pickup_datetime's that are from rows in the dataset. In other words, your window should always look 60 minutes ahead from the current row you are considering. For example, if there were no rides that had a tpep_pickup_datetime of 6:46:45pm, then you do not need to calculate the answer for 6:46:45pm - 7:46:44pm. See the last page for details on why this matters.

[1] In other words, your entire 60 minute timeslot must be between 4:00:00pm and 11:00:00pm.
[2] Note that the answer does not go to 10:47:15pm. Including an extra second would go over the 60 minute window. If you wanted 5 numbers starting at 3, your answer is 3,4,5,6,7. The number that is 5 away from 3 is not included. In the same manner, the time exactly 60 minutes ahead of 9:47:15pm is not included in the calculation.

What To Submit

Store your answers in a reasonable format in a text document called **sparkProjectOutput.txt.** Along with your answers, submit a single **ProjectFive.json** file with multiple paragraphs. Create a **`p5.zip`** file and store your sparkProjectOutput.txt along with your json file. Upload your p5.zip file to the dropbox.

Problem 7 explained further. Assume you have the following data in your dataset:

modified tpep_pickup_datetime	total_amount
4:39:00pm	10
4:45:00pm	50
5:25:00pm	50
5:41:00pm	10

A best 60 minute window could be something like 4:40:00pm - 5:39:59pm and the answer would be $50. However, such a starting/ending time doesn't exist in the data. If one only looks at rows in the dataset, it would be much harder to find the correct window. For example, one cannot just say that the window starts at a time that is in the data. If one simply says, start at the times in the dataset and go forward 60 minutes, then one would end up with 4 windows… none of which are the true correct answer:

4:39:00pm - 5:38:59pm : $36.66
4:45:00pm - 5:44:59pm : $36.66
5:25:00pm - 6:24:59pm : $30.00
5:41:00pm - 6:40:59pm : $10.00

If one just went the other way and said the time were the end of some window, then the 4 windows would be the following:

3:39:01pm - 4:39:00pm : $10.00
3:45:01pm - 4:45:00pm : $30.00
4:25:01pm - 5:25:00pm : $36.66
4:41:01pm - 5:41:00pm : $36.66

Therefore, one might need to consider arbitrary windows with respect to the time in the row to find the true correct answer.

The good thing is that our dataset is huge and most times are accounted for in the dataset. However, this is an issue that one would need to consider in a real problem.