Data engineering task

I have tried to solve the task as much as possible. I decided to use Spark because it seemed the most familiar with pandas in Python. Since I did not have much prior with Spark I was learning it on the go. I also had to study the Parquet format (column-oriented) and the Avro format (row-oriented) and why they are used in the domain of big data and when to use which format.

Description and definition of the single steps of your pipeline

The first challenge was about downloading the data from the website. After reading up on various websites and blogs I decided to use the public s3 bucket on Amazon to download the csv files instead of creating a scrapper to download the data form the website directly.

Initially, the data is downloaded from the public s3 bucket from the Amazon server. The process of downloading has been divided in two functions

1. A function which defines the csv urls and the filenames
2. The second function which downloads the files and stores them locally.

The second challenge was about matching the two schemas of the yellow taxi data and the green taxi data. For that, I have created a schema structure but it has lesser columns than the original data because looking at the requirement of the data scientist team I decided to have only columns which were needed at the time. But after examining both the data of the yellow taxi and the green taxi, it makes sense to merge the both datasets as all the columns are the same, only the pickup date and drop off date columns have a naming difference and the green taxi data has an extra column which we can have null values for in the yellow taxi data.

After the data was loaded using Spark, the schema definition had to be implemented which after exploring the data it makes sense to use the green taxi data.

However, in the current implementation I load both the green and yellow taxi data separately using Spark, and then I use the pick up time and drop off time columns and create new columns with having the to_timestamp function of the Spark library. This helps us to have the columns in date_time format which can be directly queried using SQL.

One idea that I had was when I was creating the new columns, I could have dropped the old ones and had the columns as the same name in both the taxi data and then we could have merged both the data frames.

I then use the Spark functions to convert to both the Parquet format and the Avro format. Afterwards, I have not loaded the parquet format data to run the queries which would have been much faster.

I have written the queries in SQL but I have not run the queries on the datasets. Due to most of my time in the learning curve of Spark and pipeline construction I was not able to complete all the parts of the task.

Definition of your output datasets. The formats including the schema definition (data types, names, etc)

The output is the data in parquet format and the avro format in multiple partitions.

Code of the queries (SQL or other query languages)

Queries: Query 1: We group the data of yellow and green taxi by hour of the pickup date and find the average trip distance by taking average on trip distance and then union the result of yellow and green taxi.

Query 2: We group the data of yellow and green taxi by week and day of week of the pickup date and find the count of single rider trip and union the results. Using that we group the by week and find min count.

Query 3: We group the data of yellow and green taxi by hour of the pickup date and find the count and then union the result. Using that group data we ordered the data in descending order of the count and limit the data to retrieve first 3 records.

Instructions on how to run and deploy your pipeline

Since the whole task is done as a single Python file with parts of the pipeline defined as function it is enough to run the file and get the results.