Data Enrichment with Apache Spark™

Background

Imagine you are in the ride-hailing business, having thousands of drivers roaming the streets for customers. To monitor and to optimize your operations, you periodically log the current status and the location of each driver. Unfortunately, the driver handsets cannot know all the information you need to satisfy the curiosity of the operations team. That is where you come in.

Driver data

In reality, the incoming driver updates are a never-ending stream of data. Thankfully, the Spark API is much the same for streaming and for non-streaming data. Therefore, we can work offline here, using only a small sample of real driver updates. These updates are provided in the file *drivers.log* as one JSON per line. The meaning of the relevant fields in these JSONs is as follows:

"@timestamp":	*time when event was ingested by server*
"@version":	-
"host":	-
"message":	-
"type":	-
"channel":	-
"level":	-
"i":	*driver ID*
"lt":	*latitude*
"lg":	*longitude*
"a":	-
"b":	-
"s":	*driver status (1 = available, rest is not available)*
"sp":	-
"st":	*service type*
"ts":	*time sent by driver handset*

Geo data

What the operations team wants to know, is how many drivers in which status and in which service type were on the road in a given time interval. So far, so good. The problem is that they do not want total numbers, but numbers broken down per district. These districts are given as a single GeoJSON in the attached file *districts.json*, with each district having a *polygon*, a *district ID*, a *district name*, a *city ID*, and a *city name*.

Tasks

1. Write a Spark job (in python) that enriches each driver update with the 4 fields *district ID*, *district name*, *city ID*, and *city name,* according to which district the driver update came from.
2. These data should then be aggregated by *date*, *hour*, *5-minute interval (there are 12 of them, numbered 1-12 or 0-11)*, *service type,* and *district ID* such that the result best reflects the "supply" of drivers.
3. Finally, write the aggregated table back to a file.

Deliverable

A single compressed archive that contains (a) the python package and/or jupyter notebook that you produced to accomplish the tasks and (b) some documentation detailing your reasoning, your solution design, and in particular your choices regarding the definition of “supply” in task (2). The archive should be returned no later than 24 hours after you have received this assignment.

Instructions

- Please allow some time to clean up your code so that we can understand what you are doing.
- In case you do not complete the entire assignment, please return whatever you have together with any thoughts on how you would proceed from there. Again, allow some time to clean up what you have. We would rather have a readable and well-documented portion of the solution than a messy solution that performs one more step.
- If you feel that there are any ambiguities in the formulation of the task, please point them out and justify the choices you have made.