Homeworks for Big Data Technologies 2022/2023

Common:

Prague offers information about the movements of the public transport vehicles via its data platform Golemio. https://golemio.cz/data
We collect this data and push them to Kafka topic. Information how to connect to Kafka will be provided 9.11. on the practice. On the Courseware you can see your assigned homework – details below.

Data are preprocessed and divided to 5 Kafka topics. This script basically produces 5 streams of data by vehicle type of city transportation.

Kafka Topics names:

- kafka_topic_trains= 'trains'
- kafka_topic_buses= 'buses'
- kafka_topic_regbuses= 'regbuses'
- kafka_topic_trams= 'trams'
- kafka_topic_boats= 'boats'

Use JAAS config to create consumer reading Kafka stream

```
JAAS = 'org.apache.kafka.common.security.scram.ScramLoginModule required username="USERNAME"
password="PWD";'
df = spark.readStream \
  .format("kafka")\
  .option("kafka.bootstrap.servers",          "b-2-public.bdffelkafka.3jtrac.c19.kafka.us-east-
1.amazonaws.com:9196, b-1-public.bdffelkafka.3jtrac.c19.kafka.us-east-1.amazonaws.com:9196") \
  .option("kafka.sasl.mechanism", "SCRAM-SHA-512")\
  .option("kafka.security.protocol", "SASL_SSL") \
  .option("kafka.sasl.jaas.config", JAAS) \
  .option("subscribe", "TOPICNAME") \
  .option("startingOffsets", "earliest") \
  .load()
```

Later, when processing streams, you will need to map json to DF. Use Structure below to map it.

```
from pyspark.sql.types import *

schema_pid = StructType([
    StructField('geometry', StructType([
        StructField('coordinates', ArrayType(StringType()), True),
        StructField('type', StringType())])),
    StructField('properties', StructType([
        StructField('last_position', StructType([
            StructField('bearing', IntegerType()),
            StructField('delay', StructType([
                StructField("actual", IntegerType()),
                StructField("last_stop_arrival", StringType()),
                StructField("last_stop_departure", StringType())])),
            StructField("is_canceled", BooleanType()),
            StructField('last_stop', StructType([
```

```
                StructField("arrival_time", StringType()),
                StructField("departure_time", StringType()),
                StructField("id", StringType()),
                StructField("sequence", IntegerType())])),
            StructField('next_stop', StructType([
                StructField("arrival_time", StringType()),
                StructField("departure_time", StringType()),
                StructField("id", StringType()),
                StructField("sequence", IntegerType())])),
            StructField("origin_timestamp", StringType()),
            StructField("shape_dist_traveled", StringType()),
            StructField("speed", StringType()),
            StructField("state_position", StringType()),
            StructField("tracking", BooleanType())])),
        StructField('trip', StructType([
            StructField('agency_name', StructType([
                StructField("real", StringType()),
                StructField("scheduled", StringType())])),
            StructField('cis', StructType([
                StructField("line_id", StringType()),
                StructField("trip_number", StringType())])),
            StructField('gtfs', StructType([
                StructField("route_id", StringType()),
                StructField("route_short_name", StringType()),
                StructField("route_type", IntegerType()),
                StructField("trip_headsign", StringType()),
                StructField("trip_id", StringType()),
                StructField("trip_short_name", StringType())])),
            StructField("origin_route_name", StringType()),
            StructField("sequence_id", IntegerType()),
            StructField("start_timestamp", StringType()),
            StructField("vehicle_registration_number", IntegerType()),
            StructField('vehicle_type', StructType([
                StructField("description_cs", StringType()),
                StructField("description_en", StringType()),
                StructField("id", IntegerType())])),
            StructField("wheelchair_accessible", BooleanType()),
            StructField("air_conditioned", BooleanType())]))])),
    StructField("type", StringType())
])
```

1 - How many vehicles are in operation at each part of the day and what is their average speed?

From the data stream, implement a stream processing application that will track the number of vehicles and their types through the day. Calculate their average speed so that you can compare its evolution in different parts of the day.

Input: stream

Output: a report showing the evolution within a given day

Commented [BA1]: How many and what public transport type (tram, bus etc...) is active during the day? Segment the day per hours. Count in stream application amount of vehicles of specific type and calculate theirs average speed. As a bonus, segment the day to smaller interval than one hour.
Input: Kafka stream
Output: Report with amount and average speed for every ype of vehicle. Do you see something interesting, any trends?

2 – Locations with the greatest decrease in delays

From the data stream, implement a stream processing application that will monitor the delay of traffic -> where delays are fastest decreasing. Detect the locations where the most traffic "spikes" occur repeatedly.

Commented [BA2]: The most, like first 3?

Input: Stream

Output: GPS coordinates of the "fastest delay minimization" locations, dashboard map showing these locations

3 - Detection of the bus behind

From the stream data, implement a stream processing application that will detect the event when one bus catches the previous one. This means that buses that have an interval of, for example, 6 minutes are following closely behind. Save the locations where this happens most often and plot them on a map, along with information about which routes they are.

Input: Stream

Output: GPS coordinates of bus arrivals, dashboard map showing locations with many bus arrival events

Commented [BA3]: Possible vehilce collision (concurrent stay) at one bus stop
Detect buses, which may occur on one bus/tram/boat stop at the same time. We will consider, that collision may happen when the fisrt bus on the bus stop and the following buses have time gap only 3 minutes or lower, ie. the first bus must stay (or had delay), and following buses are on time (or sooner). Collect places, where this happens, type of vehicle and short route name. Find top 10 collision places.
Input: Kafka stream
Ouput: GPS coords of bus stops, dashboard map with amount of possible collisions.

4 - Differences by direction of travel - to and from Prague

From the data stream, implement a stream processing application that will calculate differences in delay for suburban lines - arrival and departure delays.

Input: stream

Output: Dashboard map with arrivals to Prague and marking the differences in delays during the day