Logic for First Submission

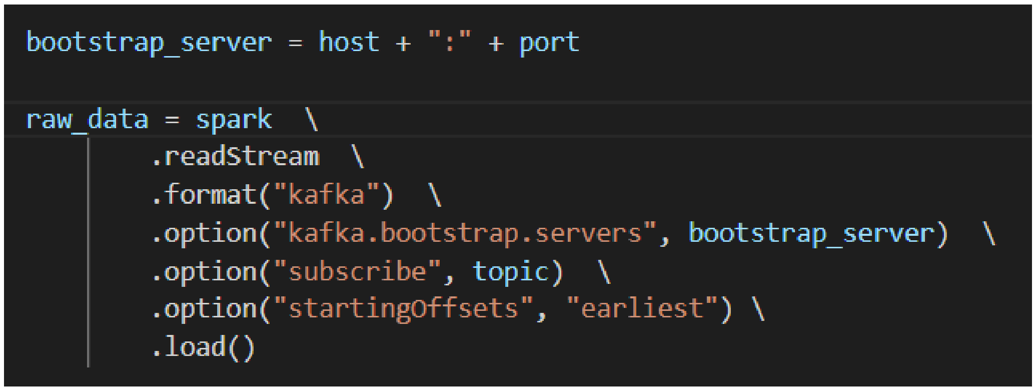
1. spark_kafka_to_local.py

```
if len(sys.argv) != 4:
        print("Usage: spark-submit spark_kafka_to_local.py <hostname> <port>
<topic>")
        exit(-1)

    host = sys.argv[1]
    port = sys.argv[2]
    topic = sys.argv[3]

    spark = SparkSession  \
            .builder  \
            .appName("CapstoneProject") \
            .getOrCreate()
    #spark.set("stopGracefullyOnShutdown", "true")
    spark.sparkContext.setLogLevel('ERROR')
```

Here, we receive 3 inputs which are hostname, port and Kafka topic. And a spark session is created with name "CapstoneProject".



```
bootstrap_server = host + ":" + port

raw_data = spark  \
        .readStream  \
        .format("kafka")  \
        .option("kafka.bootstrap.servers", bootstrap_server)  \
        .option("subscribe", topic)  \
        .option("startingOffsets", "earliest") \
        .load()
```

This sub code is used to create starting process of reading streaming data

```
schema = StructType() \
            .add("customer_id", StringType()) \
            .add("app_version", StringType()) \
            .add("OS_version", StringType()) \
            .add("lat", StringType()) \
            .add("lon", StringType()) \
            .add("page_id", StringType()) \
            .add("button_id", StringType()) \
            .add("is_button_click", StringType()) \
            .add("is_page_view", StringType()) \
            .add("is_scroll_up", StringType()) \
            .add("is_scroll_down", StringType()) \
            .add("timestamp\n", TimestampType())

    raw_data = raw_data.selectExpr("cast(value as string)") \
        .select(from_json("value", schema).alias("temp")).select("temp.*") \
        .withColumnRenamed("timestamp\n", "timestamp")

    console_output = raw_data.writeStream \
            .format("console") \
            .outputMode("append") \
            .option("truncate", "true") \
            .start()

    csv_output = raw_data.writeStream \
            .format("csv") \
            .outputMode("append") \
            .option("truncate", "false") \
            .option("path", "/user/ec2-user/capstone_project/warehouse/capstone")
\
            .option("checkpointLocation", "/user/ec2-
user/capstone_project/warehouse/checkpoint") \
            .start()

    console_output.awaitTermination(900)
    csv_output.awaitTermination()
```

From the code below, we create a schema that will be used in dataframe. The dataframe will come in as string, so we need to convert **values** to a separated-column table. And then rename the column from

timestamp\n to **timestamp**. After that, we can write the data to console and store it as csv format in HDFS.

2. spark_local_flatten_datewise_aggregates.py

Here, it is quite similar to the previous one. We also create SparkSession.

```
spark = SparkSession  \
            .builder  \
            .appName("CapstoneProject")  \
            .enableHiveSupport() \
            .getOrCreate()
    spark.sparkContext.setLogLevel('ERROR')
```

Then create a schema and load data for bookings data.

```
# create a schema
    schema = StructType(
        [
            StructField("booking_id", StringType(), False),
            StructField("customer_id", IntegerType(), True),
            StructField("driver_id", IntegerType(), True),
            StructField("customer_app_version", StringType(), True),
            StructField("customer_phone_os_version", StringType(), True),
            StructField("pickup_lat", FloatType(), True),
            StructField("pickup_lon", FloatType(), True),
            StructField("drop_lat", FloatType(), True),
            StructField("drop_lon", FloatType(), True),
            StructField("pickup_timestamp", TimestampType(), True),
            StructField("drop_timestamp", TimestampType(), True),
            StructField("trip_fare", IntegerType(), True),
            StructField("tip_amount", IntegerType(), True),
            StructField("currency_code", StringType(), True),
            StructField("cab_color", StringType(), True),
            StructField("cab_registration_no", StringType(), True),
            StructField("customer_rating_by_driver", IntegerType(), True),
            StructField("rating_by_customer", IntegerType(), True),
            StructField("passenger_count", IntegerType(), True),
        ]
    )

    # load bookings data
    bookings = spark.read.csv("/user/ec2-user/capstone_project/rds-data",
schema=schema)
```

Load clicking stream data and change column names

```
clicking_stream = spark.read.csv("/user/ec2-
user/capstone_project/warehouse/capstone/*.csv", sep=",", inferSchema=True)

    # change column names
    clicking_stream = clicking_stream.withColumnRenamed("_c0", "customer_id") \
            .withColumnRenamed("_c1", "app_version") \
            .withColumnRenamed("_c2", "os_version") \
            .withColumnRenamed("_c3", "lat") \
            .withColumnRenamed("_c4", "lon") \
            .withColumnRenamed("_c5", "page_id") \
            .withColumnRenamed("_c6", "button_id") \
            .withColumnRenamed("_c7", "is_button_click") \
            .withColumnRenamed("_c8", "is_page_view") \
            .withColumnRenamed("_c9", "is_scroll_up") \
            .withColumnRenamed("_c10", "is_scroll_down") \
            .withColumnRenamed("_c11", "timestamp")
```

Create datewise aggregate bookings table

```
datewise_bookings = bookings \
            .withColumn("date", to_date("pickup_timestamp")) \
            .groupBy("date").agg(count("date").alias("total_bookings")).orderBy("
date")
```

Finally, create Hive database and load all 3 tables to Hive

```
# create Hive database
    spark.sql("create database if not exists capstone_project")

    # write dataframes to Hive
    bookings.write.mode("overwrite").saveAsTable("capstone_project.bookings")
    clicking_stream.write.mode("overwrite").saveAsTable("capstone_project.clickin
g_stream")
    datewise_bookings.write.format("orc").mode("overwrite").saveAsTable("capstone
_project.datewise_bookings")
    print("Successfully loaded dataframes to Hive")
```

When we see a message “Successfully loaded dataframes to Hive”, then it is successful!