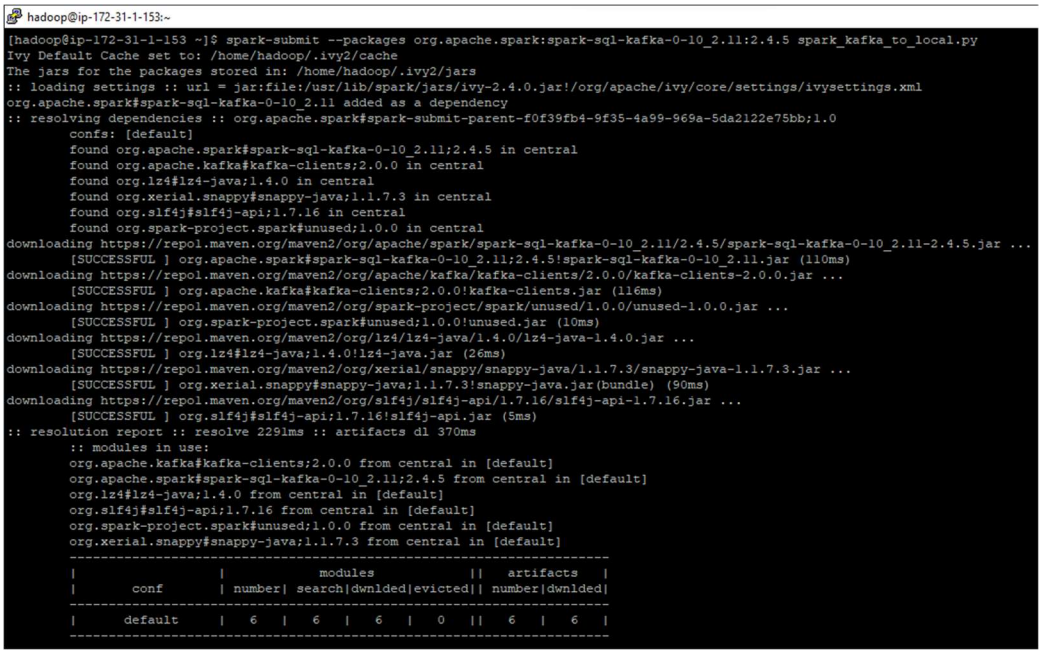
Logic for First Submission

1. spark_kafka_to_local.py

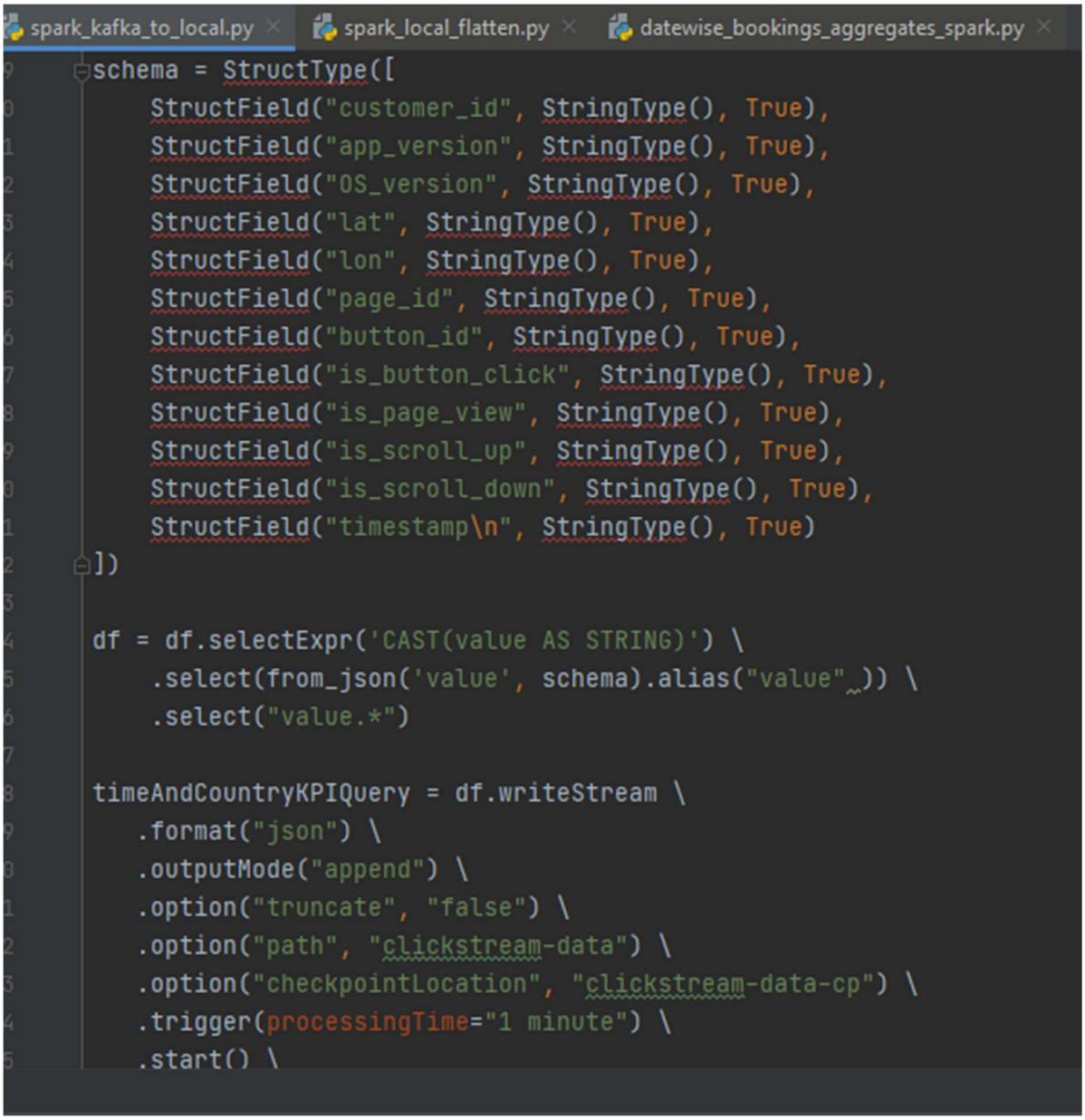
spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 spark_kafka_to_local.py



Here a spark session is created with name “clickstream”. Kafka topic will be downloaded

```
df = spark.readStream \
  .format("kafka") \
  .option("kafka.bootstrap.servers", "18.211.252.152:9092") \
  .option("subscribe", "de-capstone3") \
  .option("auto.offset.reset", "earliest") \
  .option("startingOffsets", "earliest") \
  .load()
```

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



From the code above, 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.

```
from pyspark.sql.types import *
from pyspark.sql.functions import *
from pyspark.sql import functions as F
from pyspark.sql import SparkSession
from pyspark.sql.functions import from_json

spark = SparkSession.builder.appName('clickstream').master('local[1]').getOrCreate()

spark.sparkContext.setLogLevel('WARN')

df = spark.read.json('clickstream-data')

df.repartition(1).write.format("csv").save("clickstream")
```

Create datewise aggregate bookings dataframe

```
df = spark.read \
  .format("jdbc") \
  .option("url", "jdbc:mysql://upgraddetest.cyaielc9bmnf.us-east-1.rds.amazonaws.com/testdatabase") \
  .option("driver", "com.mysql.jdbc.Driver") \
  .option("dbtable", "bookings") \
  .option("user", "student") \
  .option("password", "STUDENT123") \
  .load()
```

```
df = df \
    .groupBy(to_date(df.pickup_timestamp,"dd-MM-yyyy").alias("booking date")) \
    .agg(count("booking_id").alias("Total Bookings")) \
    .orderBy("booking date")
```

Aggregate the dataframe using bookind date.

```
df.repartition(1).write.format("csv").save("bookings")
```

Write the aggregated data to a folder bookings and save as csv.

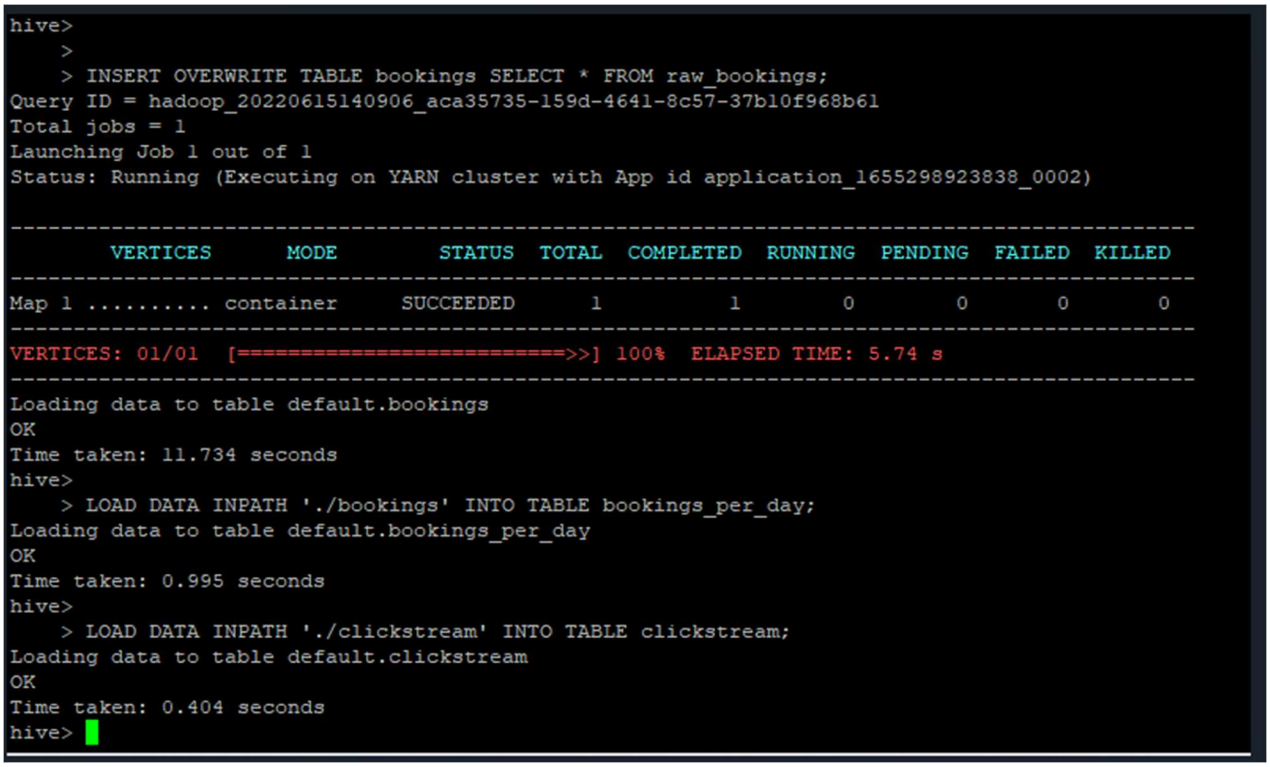
Create hive managed tables and load data:

hadoop@ip-172-31-1-153:~

```
[hadoop@ip-172-31-1-153 ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: false
hive> CREATE TABLE IF NOT EXISTS clickstream(
    >     OS_version STRING,
    >     app_version STRING,
    >     button_id STRING,
    >     customer_id STRING,
    >     is_button_click STRING,
    >     is_page_view STRING,
    >     is_scroll_down STRING,
    >     is_scroll_up STRING,
    >     lat STRING,
    >     lon STRING,
    >     page_id STRING,
    >     `timestamp` STRING)
    > COMMENT 'Data about bookings from a RDS'
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE;
OK
Time taken: 2.054 seconds
hive>
    > CREATE EXTERNAL TABLE raw_bookings (
    >     booking_id STRING,
    >     customer_id INT,
    >     driver_id INT,
    >     customer_app_version STRING,
    >     customer_phone_os_version STRING,
    >     pickup_lat DECIMAL,
    >     pickup_lon DECIMAL,
    >     drop_lat DECIMAL,
    >     drop_lon DECIMAL,
    >     pickup_timestamp TIMESTAMP,
    >     drop_timestamp TIMESTAMP,
    >     trip_fare DECIMAL,
    >     tip_amount DECIMAL,
    >     currency_code STRING,
    >     cab_color STRING,
    >     cab_registration_no STRING,
    >     customer_rating_by_driver INT,
    >     rating_by_customer INT,
    >     passenger_count INT)
```

```
    >     ROW FORMAT DELIMITED
    >     FIELDS TERMINATED BY ','
    >     STORED AS TEXTFILE
    >     location '/home/hadoop/bookings-data';
OK
Time taken: 0.104 seconds
hive>
    > CREATE TABLE IF NOT EXISTS bookings(
    >     booking_id STRING,
    >     customer_id INT,
    >     driver_id INT,
    >     customer_app_version STRING,
    >     customer_phone_os_version STRING,
    >     pickup_lat DECIMAL,
    >     pickup_lon DECIMAL,
    >     drop_lat DECIMAL,
    >     drop_lon DECIMAL,
    >     pickup_timestamp TIMESTAMP,
    >     drop_timestamp TIMESTAMP,
    >     trip_fare DECIMAL,
    >     tip_amount DECIMAL,
    >     currency_code STRING,
    >     cab_color STRING,
    >     cab_registration_no STRING,
    >     customer_rating_by_driver INT,
    >     rating_by_customer INT,
    >     passenger_count INT)
    > COMMENT 'Data about bookings from a RDS'
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE;
OK
Time taken: 0.124 seconds
hive>
    > CREATE TABLE IF NOT EXISTS bookings_per_day(
    >     pickup_timestamp DATE,
    >     bookings INT)
    > COMMENT 'Aggregated Data about bookings from a Bookings table'
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > STORED AS TEXTFILE;
OK
Time taken: 0.104 seconds
```



We have Successfully loaded dataframes to Hive.