

# 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

```
hadoop@ip-172-31-1-153:~$ spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 spark_kafka_to_local.py
Ivy Default Cache set to: /home/hadoop/.ivy2/cache
The jars for the packages stored in: /home/hadoop/.ivy2/jars
:: loading settings :: url = jar:file:/usr/lib/spark/jars/ivy-2.4.0.jar!/org/apache/ivy/core/settings/ivysettings.xml
org.apache.spark#spark-sql-kafka-0-10_2.11 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-f0f39fb4-9f35-4a99-9e9a-5da2122e75bb;1.0
conf:: [default]
  found org.apache.spark#spark-sql-kafka-0-10_2.11:2.4.5 in central
  found org.apache.kafka#kafka-clients;2.0.0 in central
  found org.lz4#lz4-java;1.4.0 in central
  found org.xerial.snappy#snappy-java;1.1.7.3 in central
  found org.slf4j#slf4j-api;1.7.16 in central
  found org.spark-project.spark#unused;1.0.0 in central
downloading https://repol.maven.org/maven2/org/apache/spark/spark-sql-kafka-0-10_2.11/2.4.5/spark-sql-kafka-0-10_2.11-2.4.5.jar ...
[SUCCESSFUL ] org.apache.spark#spark-sql-kafka-0-10_2.11:2.4.5!spark-sql-kafka-0-10_2.11.jar (110ms)
downloading https://repol.maven.org/maven2/org/apache/kafka/kafka-clients/2.0.0/kafka-clients-2.0.0.jar ...
[SUCCESSFUL ] org.apache.kafka#kafka-clients;2.0.0!kafka-clients.jar (116ms)
downloading https://repol.maven.org/maven2/org/spark-project/spark/unused/1.0.0/unused-1.0.0.jar ...
[SUCCESSFUL ] org.spark-project.spark#unused;1.0.0!unused.jar (10ms)
downloading https://repol.maven.org/maven2/org/lz4/lz4-java/1.4.0/lz4-java-1.4.0.jar ...
[SUCCESSFUL ] org.lz4#lz4-java;1.4.0!lz4-java.jar (26ms)
downloading https://repol.maven.org/maven2/org/xerial/snappy/snappy-java/1.1.7.3/snappy-java-1.1.7.3.jar ...
[SUCCESSFUL ] org.xerial.snappy#snappy-java;1.1.7.3!snappy-java.jar(bundle) (90ms)
downloading https://repol.maven.org/maven2/org/slf4j/slf4j-api/1.7.16/slf4j-api-1.7.16.jar ...
[SUCCESSFUL ] org.slf4j#slf4j-api;1.7.16!slf4j-api.jar (5ms)
:: resolution report :: resolve 2291ms :: artifacts dl 370ms
:: modules in use:
org.apache.kafka#kafka-clients;2.0.0 from central in [default]
org.apache.spark#spark-sql-kafka-0-10_2.11:2.4.5 from central in [default]
org.lz4#lz4-java;1.4.0 from central in [default]
org.slf4j#slf4j-api;1.7.16 from central in [default]
org.spark-project.spark#unused;1.0.0 from central in [default]
org.xerial.snappy#snappy-java;1.1.7.3 from central in [default]
-----
|               | modules | artifacts |
| conf | number| search|dwnlded|evicted|| number|dwnlded|
|-----|-----|-----|-----|
| default | 6 | 6 | 6 | 0 | | 6 | 6 |
```

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

```
spark_kafka_to_local.py x spark_local_flatten.py x datewise_bookings_aggregates_spark.py x
9 schema = StructType([
10     StructField("customer_id", StringType(), True),
11     StructField("app_version", StringType(), True),
12     StructField("OS_version", StringType(), True),
13     StructField("lat", StringType(), True),
14     StructField("lon", StringType(), True),
15     StructField("page_id", StringType(), True),
16     StructField("button_id", StringType(), True),
17     StructField("is_button_click", StringType(), True),
18     StructField("is_page_view", StringType(), True),
19     StructField("is_scroll_up", StringType(), True),
20     StructField("is_scroll_down", StringType(), True),
21     StructField("timestamp\n", StringType(), True)
22 ])
23
24 df = df.selectExpr('CAST(value AS STRING)') \
25     .select(from_json('value', schema).alias("value")) \
26     .select("value.*")
27
28 timeAndCountryKPIQuery = df.writeStream \
29     .format("json") \
30     .outputMode("append") \
31     .option("truncate", "false") \
32     .option("path", "clickstream-data") \
33     .option("checkpointLocation", "clickstream-data-cp") \
34     .trigger(processingTime="1 minute") \
35     .start() \
```

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.cyaieic9bmnf.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 booking 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



```

hive>
>
> INSERT OVERWRITE TABLE bookings SELECT * FROM raw_bookings;
Query ID = hadoop_20220615140906_aca35735-159d-4641-8c57-37b10f968b61
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1655298923838_0002)

-----
VERTICES      MODE      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
-----
Map 1 ..... container      SUCCEEDED      1          1          0          0          0          0
-----
VERTICES: 01/01 [=====>>] 100%  ELAPSED TIME: 5.74 s
-----
Loading data to table default.bookings
OK
Time taken: 11.734 seconds
hive>
> LOAD DATA INPATH './bookings' INTO TABLE bookings_per_day;
Loading data to table default.bookings_per_day
OK
Time taken: 0.995 seconds
hive>
> LOAD DATA INPATH './clickstream' INTO TABLE clickstream;
Loading data to table default.clickstream
OK
Time taken: 0.404 seconds
hive> █

```

We have Successfully loaded dataframes to Hive.