Logic For First Submission

Before anything, make sure you create a user in hdfs file system and give ownership of the newly created folder to store the results to the user.
For eg, user used below 0s 'ec2-user' and the folder created to store the results are 'cab_ride_analysis'

1. **sudo -i**
2. **su – hdfs**
3. **hadoop fs -mkdir /user/ec2-user**
4. **hadoop fs -mkdir /user/ec2-user/cab_ride_analysis**
5. **hadoop fs -chown ec2-user /user/ec2-user/cab_ride_analysis**

Task 1: Write a job to consume clickstream data from Kafka and ingest to Hadoop.

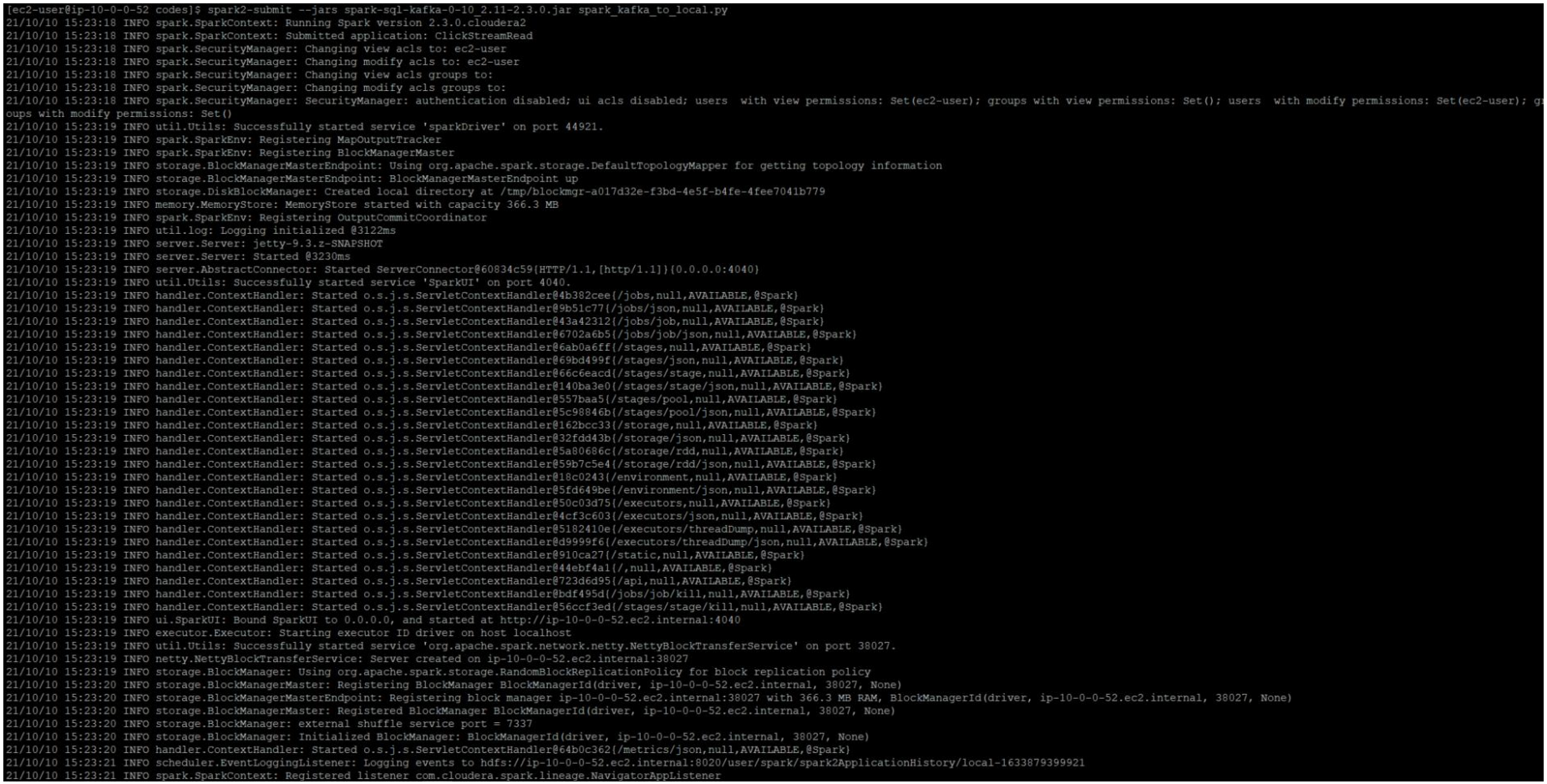
To perform the above task, you need to run two pyspark scripts, "spark_kafka_to_local.py" and "spark_local_flatten.py".

Step 1: Please run the below command to download the Spark-SQL-Kafka jar file. This jar will be used to run the Spark Streaming-Kafka codes. Please copy-paste the below command in your EC2 instance terminal.
"wget https://ds-spark-sql-kafka-jar.s3.amazonaws.com/spark-sql-kafka-0-10_2.11-2.3.0.jar"

Step 2: Running the first script will stream the data into the HDFS files system as raw data. No functions are applied here, only the data from kafka is stored into the HDFS. You can define the kafka server and topic name in lines 12 and 13. The command to run the same is given below.

"spark2-submit --jars spark-sql-kafka-0-10_2.11-2.3.0.jar spark_kafka_to_local.py"

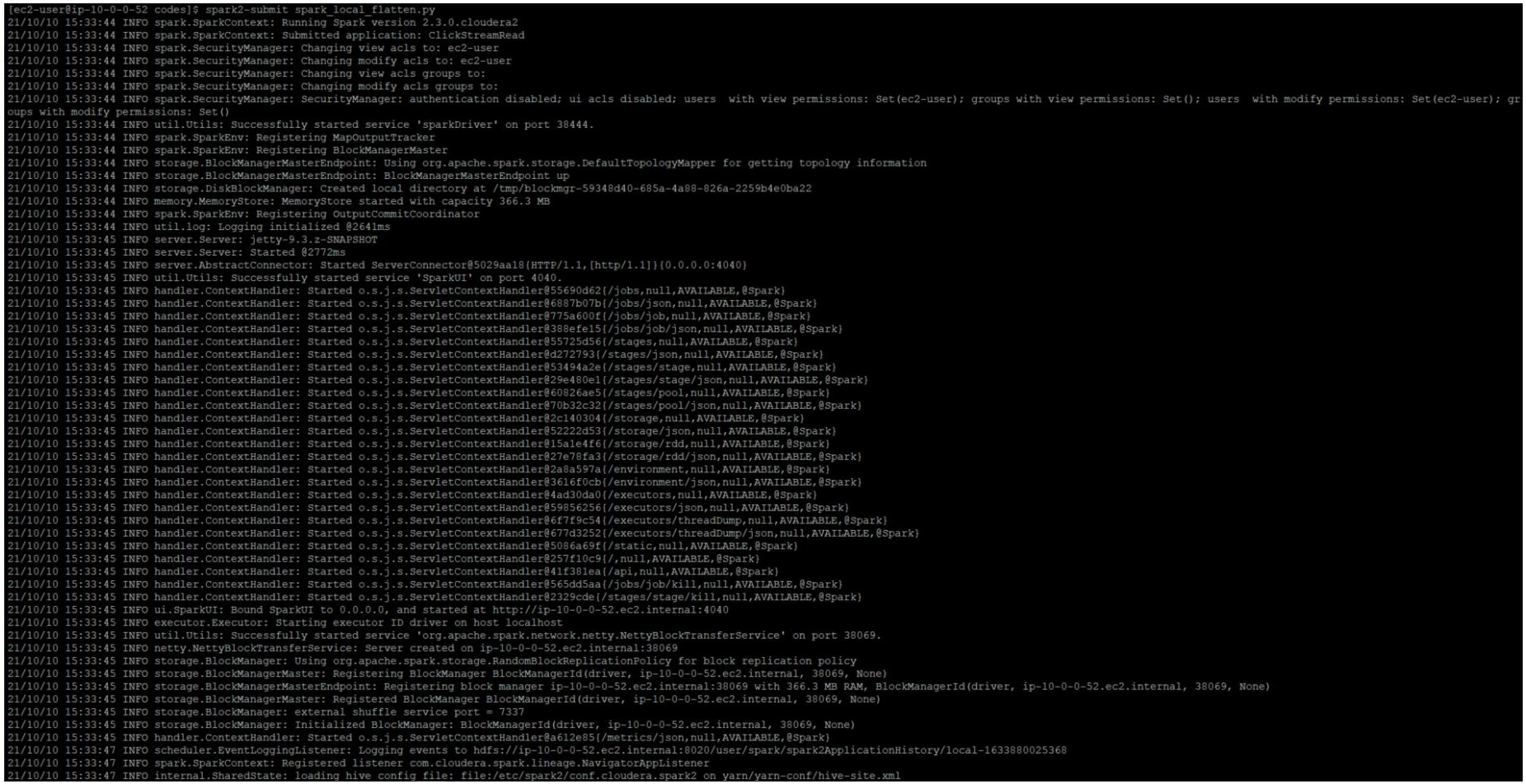
Screenshot Step 2:
https://drive.google.com/file/d/1fDA1ttLyYzaLvuEqeixcXDUyekMi1jwE/view?usp=sharing



Step 3: Running the second script will flatten the data into a more structured format and save it as a CSV in HDFS, so hive can fetch it from there. Command to run the script is given below.

"spark2-submit spark_local_flatten.py"

Screenshots Step 3:
https://drive.google.com/file/d/1zZcfOSQpK6L3vReR5B9baGlQbDlVVGcV/view?usp=sharing



https://drive.google.com/file/d/1zZcfOSQpK6L3vReR5B9baGlQbDlVVGcV/view?usp=sharing

```
21/10/10 15:33:53 INFO scheduler.DAGScheduler: Job 0 finished: json at NativeMethodAccessorImpl.java:0, took 0.749357 s
StructType(List(StructField(cusomter_id,StringType,true),StructField(app_version,StringType,true),StructField(OS_version,StringType,true),StructField(lat,StringType,true),StructField(lon,StringType,true),StructField(page_id,StringType,tr
ue),StructField(button_id,StringType,true),StructField(is_button_click,StringType,true),StructField(is_page_view,StringType,true),StructField(is_scroll_up,StringType,true),StructField(is_scroll_down,StringType,true),StructField(timestamp
,StringType,true)))
```

https://drive.google.com/file/d/1gOglGnu0pLYcDWhOtmkDXUVPG_1mho-5/view?usp=sharing

```
21/10/10 15:33:54 INFO scheduler.DAGScheduler: Job 1 finished: showString at NativeMethodAccessorImpl.java:0, took 0.285980 s
+-----------+-----------+----------+-----------+-----------+--------------------+--------------------+---------------+------------+------------+--------------+---------+
|cusomter_id|app_version|OS_version|        lat|        lon|             page_id|           button_id|is_button_click|is_page_view|is_scroll_up|is_scroll_down|timestamp|
+-----------+-----------+----------+-----------+-----------+--------------------+--------------------+---------------+------------+------------+--------------+---------+
|   26564820|     3.2.35|   Android| 16.4454865|  99.902065|de545711-3914-445...|fcba68aa-1231-11e...|             No|         Yes|          No|           Yes|     null|
|   31906387|      2.4.7|       iOS| -64.813749|-133.527040|de545711-3914-445...|a95dd57b-779f-49d...|             No|          No|         Yes|           Yes|     null|
|   25713677|     3.4.12|   Android|  89.943435| 127.313415|b328829e-17ae-11e...|fcba68aa-1231-11e...|             No|          No|         Yes|            No|     null|
|   83474293|      3.1.8|   Android| -69.939070| -36.451670|e7bc5fb2-1231-11e...|e1e99492-17ae-11e...|            Yes|          No|         Yes|            No|     null|
|   63727807|      2.2.9|       iOS|  64.082108| -81.822078|e7bc5fb2-1231-11e...|fcba68aa-1231-11e...|             No|         Yes|         Yes|           Yes|     null|
|   73737907|     4.3.19|   Android| -18.850508|-116.358375|b328829e-17ae-11e...|e1e99492-17ae-11e...|             No|         Yes|          No|           Yes|     null|
|   36927433|     3.2.26|       iOS|-84.6857245|-146.507678|de545711-3914-445...|a95dd57b-779f-49d...|            Yes|         Yes|          No|           Yes|     null|
|   12691783|     3.3.11|   Android| 54.3852925| -37.411814|de545711-3914-445...|e1e99492-17ae-11e...|            Yes|         Yes|          No|            No|     null|
|   22635021|     4.4.36|       iOS| -31.805500| 150.655650|e7bc5fb2-1231-11e...|a95dd57b-779f-49d...|             No|          No|          No|            No|     null|
|   23593546|     1.2.16|   Android|  8.8918475| -83.929878|de545711-3914-445...|e1e99492-17ae-11e...|            Yes|          No|         Yes|            No|     null|
+-----------+-----------+----------+-----------+-----------+--------------------+--------------------+---------------+------------+------------+--------------+---------+
only showing top 10 rows

21/10/10 15:33:55 INFO datasources.FileSourceStrategy: Pruning directories with:
21/10/10 15:33:55 INFO datasources.FileSourceStrategy: Post-Scan Filters:
21/10/10 15:33:55 INFO datasources.FileSourceStrategy: Output Data Schema: struct<value_str: string>
21/10/10 15:33:55 INFO execution.FileSourceScanExec: Pushed Filters:
21/10/10 15:33:55 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 1
21/10/10 15:33:55 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
21/10/10 15:33:55 INFO datasources.SQLHadoopMapReduceCommitProtocol: Using output committer class org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
21/10/10 15:33:55 INFO memory.MemoryStore: Block broadcast_4 stored as values in memory (estimated size 337.9 KB, free 365.2 MB)
21/10/10 15:33:55 INFO memory.MemoryStore: Block broadcast_4_piece0 stored as bytes in memory (estimated size 30.6 KB, free 365.2 MB)
21/10/10 15:33:55 INFO storage.BlockManagerInfo: Added broadcast_4_piece0 in memory on ip-10-0-0-52.ec2.internal:38069 (size: 30.6 KB, free: 366.2 MB)
21/10/10 15:33:55 INFO spark.SparkContext: Created broadcast 4 from save at NativeMethodAccessorImpl.java:0
21/10/10 15:33:55 INFO execution.FileSourceScanExec: Planning scan with bin packing, max size: 5462515 bytes, open cost is considered as scanning 4194304 bytes.
21/10/10 15:33:55 INFO spark.SparkContext: Starting job: save at NativeMethodAccessorImpl.java:0
21/10/10 15:33:55 INFO scheduler.DAGScheduler: Got job 2 (save at NativeMethodAccessorImpl.java:0) with 1 output partitions
21/10/10 15:33:55 INFO scheduler.DAGScheduler: Final stage: ResultStage 2 (save at NativeMethodAccessorImpl.java:0)
21/10/10 15:33:55 INFO scheduler.DAGScheduler: Parents of final stage: List()
21/10/10 15:33:55 INFO scheduler.DAGScheduler: Missing parents: List()
```

Task 2: Write a script to ingest the relevant bookings data from AWS RDS to Hadoop.

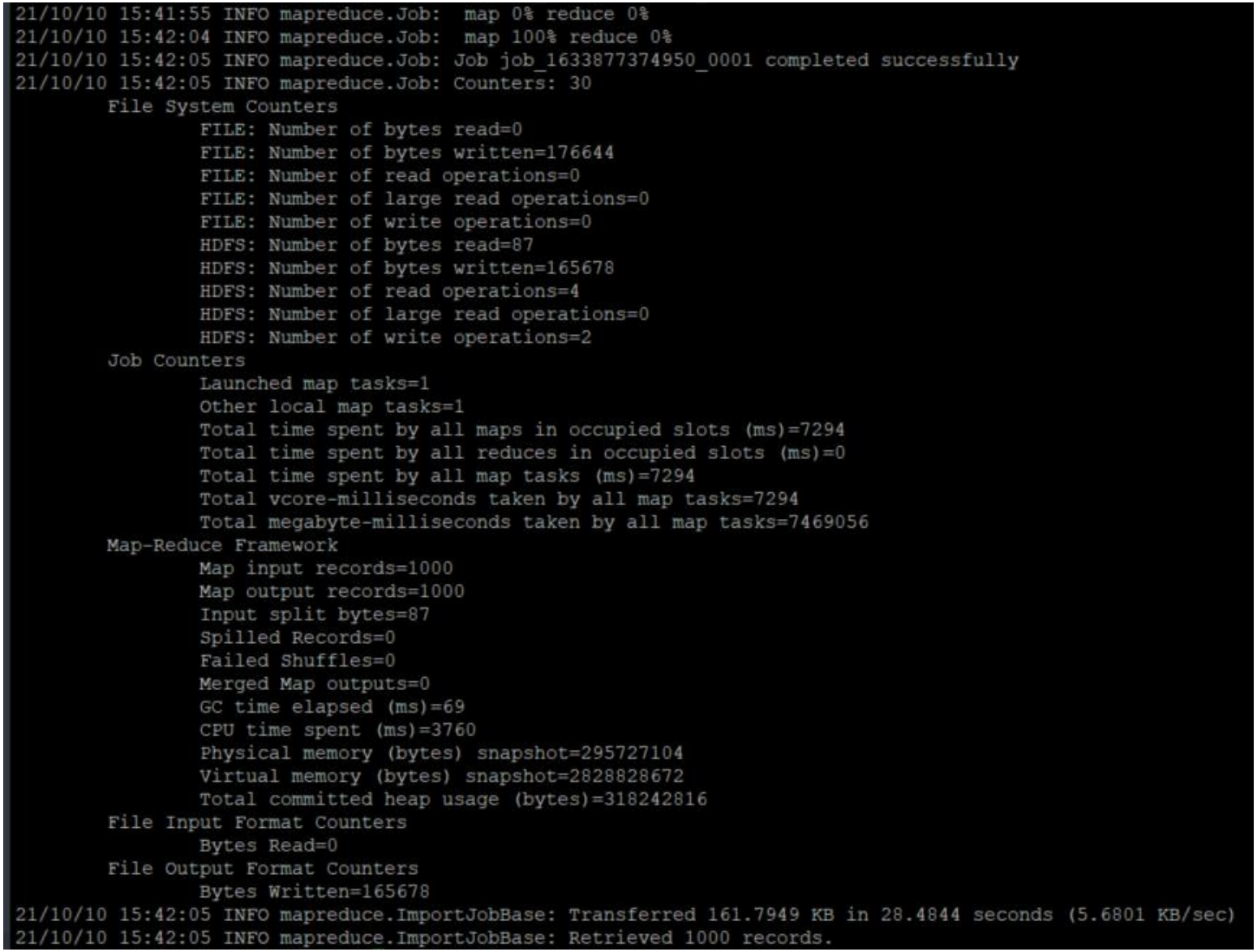
Run the following sqoop import command to import the bookings data from AWS RDS to Hadoop

"sqoop import --connect jdbc:mysql://upgraddetest.cyaielc9bmnf.us-east-1.rds.amazonaws.com/testdatabase --table bookings --username student --password STUDENT123 --target-dir /user/ec2-user/cab_ride_analysis/sqoop/bookings -m 1"

Screenshots Task 2:
https://drive.google.com/file/d/1uMyMzD3HH4V7fTpvL_eOwH6N5MhUHPgi/view?usp=sharing

```
[ec2-user@ip-10-0-0-52 codes]$ sqoop import --connect jdbc:mysql://upgraddetest.cyaielc9bmnf.us-east-1.rds.amazonaws.com/testdatabase --table bookings --username student --password STUDENT123 --target-dir /user/ec2-user/cab_ride_analysi
/sqoop1/bookings -m 1
Warning: /opt/cloudera/parcels/CDH-5.15.1-1.cdh5.15.1.p0.4/bin/../lib/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
21/10/10 15:41:32 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.15.1
21/10/10 15:41:32 WARN tool.BaseSqoopTool: Setting your password on the command-line is insecure. Consider using -P instead.
21/10/10 15:41:32 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
21/10/10 15:41:32 INFO tool.CodeGenTool: Beginning code generation
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'. The driver is automatically registered via the SPI and manual loading of the driver class is generally unnecessary.
21/10/10 15:41:33 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `bookings` AS t LIMIT 1
21/10/10 15:41:33 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `bookings` AS t LIMIT 1
21/10/10 15:41:33 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce
Note: /tmp/sqoop-ec2-user/compile/fe32b44f2642536ff717edad6e4a2bd8/bookings.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
21/10/10 15:41:36 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-ec2-user/compile/fe32b44f2642536ff717edad6e4a2bd8/bookings.jar
21/10/10 15:41:36 WARN manager.MySQLManager: It looks like you are importing from mysql.
21/10/10 15:41:36 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
21/10/10 15:41:36 WARN manager.MySQLManager: option to exercise a MySQL-specific fast path.
21/10/10 15:41:36 INFO manager.MySQLManager: Setting zero DATETIME behavior to convertToNull (mysql)
21/10/10 15:41:36 INFO mapreduce.ImportJobBase: Beginning import of bookings
21/10/10 15:41:36 INFO Configuration.deprecation: mapred.jar is deprecated. Instead, use mapreduce.job.jar
21/10/10 15:41:37 INFO Configuration.deprecation: mapred.map.tasks is deprecated. Instead, use mapreduce.job.maps
21/10/10 15:41:37 INFO client.RMProxy: Connecting to ResourceManager at ip-10-0-0-52.ec2.internal/10.0.0.52:8032
21/10/10 15:41:42 INFO db.DBInputFormat: Using read commited transaction isolation
21/10/10 15:41:43 INFO mapreduce.JobSubmitter: number of splits:1
21/10/10 15:41:43 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1633877374950_0001
21/10/10 15:41:44 INFO impl.YarnClientImpl: Submitted application application_1633877374950_0001
21/10/10 15:41:44 INFO mapreduce.Job: The url to track the job: http://ip-10-0-0-52.ec2.internal:8088/proxy/application_1633877374950_0001/
21/10/10 15:41:44 INFO mapreduce.Job: Running job: job_1633877374950_0001
```

https://drive.google.com/file/d/17tCRzdiHgYiJlloL88h1MrbxKmWo-Uik/view?usp=sharing



```
21/10/10 15:41:55 INFO mapreduce.Job:  map 0% reduce 0%
21/10/10 15:42:04 INFO mapreduce.Job:  map 100% reduce 0%
21/10/10 15:42:05 INFO mapreduce.Job: Job job_1633877374950_0001 completed successfully
21/10/10 15:42:05 INFO mapreduce.Job: Counters: 30
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=176644
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=87
                HDFS: Number of bytes written=165678
                HDFS: Number of read operations=4
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Other local map tasks=1
                Total time spent by all maps in occupied slots (ms)=7294
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=7294
                Total vcore-milliseconds taken by all map tasks=7294
                Total megabyte-milliseconds taken by all map tasks=7469056
        Map-Reduce Framework
                Map input records=1000
                Map output records=1000
                Input split bytes=87
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=69
                CPU time spent (ms)=3760
                Physical memory (bytes) snapshot=295727104
                Virtual memory (bytes) snapshot=2828828672
                Total committed heap usage (bytes)=318242816
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=165678
21/10/10 15:42:05 INFO mapreduce.ImportJobBase: Transferred 161.7949 KB in 28.4844 seconds (5.6801 KB/sec)
21/10/10 15:42:05 INFO mapreduce.ImportJobBase: Retrieved 1000 records.
```

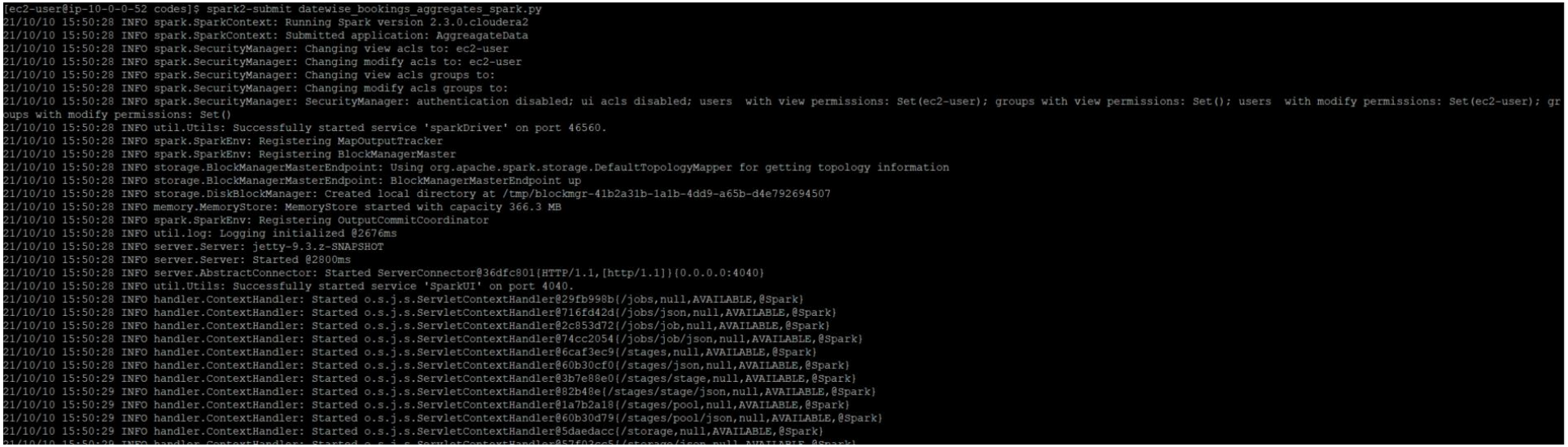
Task 3: Create aggregators for finding date-wise total bookings using the Spark script.

Run the “datewise_bookings_aggregates_spark.py” file like shown below to create a csv file with date-wise aggregated bookings total table.

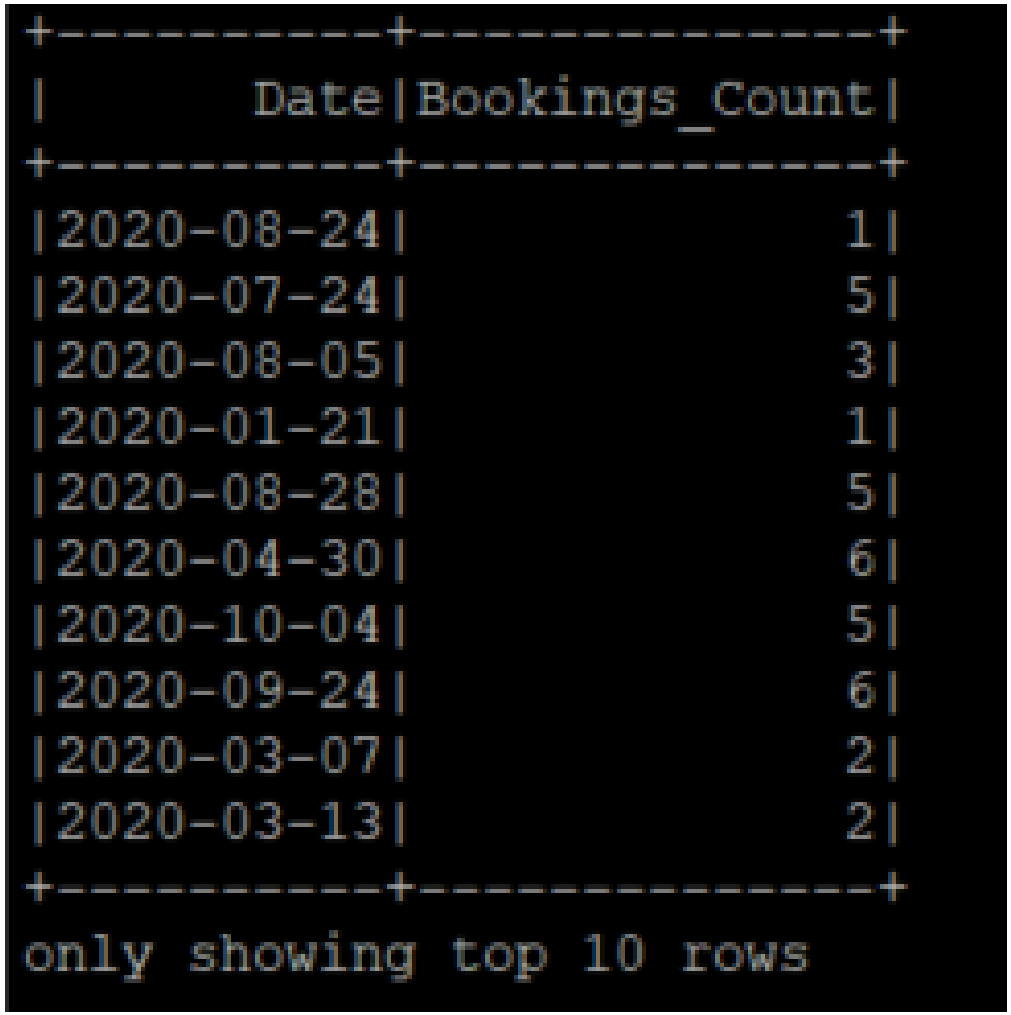
“spark2-submit datewise_bookings_aggregates_spark.py”

Screenshots Task 3:

https://drive.google.com/file/d/1ZRRRM5YzhdjZcqyAsMohc_c-vfy4QUqX/view?usp=sharing



https://drive.google.com/file/d/1A32qV6yImsjKZucypyh-wMGw6wfDXL1Z/view?usp=sharing



Date	Bookings_Count
2020-08-24	1
2020-07-24	5
2020-08-05	3
2020-01-21	1
2020-08-28	5
2020-04-30	6
2020-10-04	5
2020-09-24	6
2020-03-07	2
2020-03-13	2

only showing top 10 rows

Task 4:

To create a hive table, you need to log into the hive command line interface first, from hdfs user type in “hive” and press enter.
“Create database if not exists cab_ride_analysis;” – This command will create the database which we will be using.
You can use the database by just doing a “***use cab_ride_analysis;”***

- **Create a Hive-managed table from clickstream data.**

To create the table run the following command:

“CREATE TABLE if not exists clickstreamData(customer_id BIGINT, app_version STRING, os_version STRING, lat DECIMAL(8, 6), lon DECIMAL(9,6), page_id STRING, button_id STRING, is_button_click STRING, is_page_view STRING, is_scroll_up STRING, is_scroll_down STRING, timestamp TIMESTAMP) row format delimited fields terminated by '' lines terminated by '\n' stored as textfile;”

To load the data into the table:

“load data inpath '/user/ec2-user/cab_ride_analysis/kafka/clickstreamdump/csv/part-00000-e2eb68ab-a30b-48c9-af5c-b74052a5222d-c000.csv' into table clickstreamdata;”

https://drive.google.com/file/d/1J8ZIfdCkjsX-TOgepMWF1s3Xnz06jj7S/view?usp=sharing

```
hive> use cab_ride_analysis1;
OK
Time taken: 1.774 seconds
hive> CREATE TABLE if not exists clickstreamData( customer_id BIGINT, app_version STRING, os_version STRING, lat DECIMAL(8, 6), lon DECIMAL(9,6), page_id STRING, button_id STRING, is_button_click STRING, is_page_view STRING, is_scroll_up
 STRING, is_scroll_down STRING, timestamp TIMESTAMP ) row format delimited fields terminated by '' lines terminated by '\n' stored as textfile;
OK
Time taken: 0.301 seconds
hive> load data inpath '/user/ec2-user/cab_ride_analysis/kafka1/clickstreamdump/csv/part-00000-58859f67-cba5-4553-b9e4-8f9ae86e4cdd-c000.csv' into table clickstreamdata;
Loading data to table cab_ride_analysis1.clickstreamdata
Table cab_ride_analysis1.clickstreamdata stats: [numFiles=1, totalSize=398247]
OK
Time taken: 0.454 seconds
hive>
```

- **Create a Hive-managed table for bookings data.**

To create the table run the following command:

“CREATE TABLE if not exists bookings(booking_id STRING, customer_id BIGINT, driver_id BIGINT, customer_app_version STRING, customer_phone_os_version STRING, pickup_lat DECIMAL(8,6), pickup_lon DECIMAL(9,6), drop_lat DECIMAL(8,6), drop_lon DECIMAL(9,6), pickup_timestamp TIMESTAMP, drop_timestamp TIMESTAMP, trip_fare INT, tip_amount INT, currency_code STRING, cab_color STRING, cab_registration_no STRING, customer_rating_by_driver INT, rating_by_customer INT, passenger_count BIGINT) row format delimited fields terminated by ',' lines terminated by '\n' stored as textfile;”

To load the data into the table:

“load data inpath '/user/ec2-user/cab_ride_analysis/sqoop/bookings/part-m-00000' into table bookings;”

https://drive.google.com/file/d/1J8ZlfdCkjsX-TOgepMWF1s3Xnz06jj7S/view?usp=sharing

```
hive> use cab_ride_analysis1;
OK
Time taken: 1.774 seconds
hive> CREATE TABLE if not exists clickstreamData( customer_id BIGINT, app_version STRING, os_version STRING, lat DECIMAL(8, 6), lon DECIMAL(9,6), page_id STRING, button_id STRING, is_button_click STRING, is_page_view STRING, is_scroll_up
 STRING, is_scroll_down STRING, timestamp TIMESTAMP ) row format delimited fields terminated by '' lines terminated by '\n' stored as textfile;
OK
Time taken: 0.301 seconds
hive> load data inpath '/user/ec2-user/cab_ride_analysis/kafka1/clickstreamdump/csv/part-00000-58859f67-cba5-4553-b9e4-8f9ae86e4cdd-c000.csv' into table clickstreamdata;
Loading data to table cab_ride_analysis1.clickstreamdata
Table cab_ride_analysis1.clickstreamdata stats: [numFiles=1, totalSize=398247]
OK
Time taken: 0.454 seconds
hive>
```

- **Create a Hive-managed table for aggregated data in Task 3.**

To create the table run the following command:

"CREATE TABLE aggBookings(Date DATE, count INT) row format delimited fields terminated by '|' lines terminated by '\n' stored as textfile;"

To load the data into the table:

"load data inpath '/user/ec2-user/cab_ride_analysis/aggBookings/results/part-00000-46a0ff75-0ba6-4c2b-b1d3-69fb467d5624-c000.csv' into table aggBookings;"

https://drive.google.com/file/d/1o32kxxwAigTBHw12uaHuAeRQXlaB7uM9/view?usp=sharing

```
hive> CREATE TABLE aggBookings(Date DATE, count INT) row format delimited fields
 terminated by '|' lines terminated by '\n' stored as textfile;
OK
Time taken: 0.368 seconds
hive>
    > load data inpath '/user/ec2-user/cab_ride_analysis/aggBookings1/results/pa
rt-00000-42906ce7-d823-4d9b-9f6a-e473ec086d69-c000.csv' into table aggBookings;
Loading data to table cab_ride_analysis1.aggbookings
Table cab_ride_analysis1.aggbookings stats: [numFiles=1, totalSize=3778]
OK
Time taken: 0.371 seconds
```

At the end of this documents, all the data from Kafka and RDS is stored in our Hive Managed table, even the aggregated Bookings data is saved in Hive.