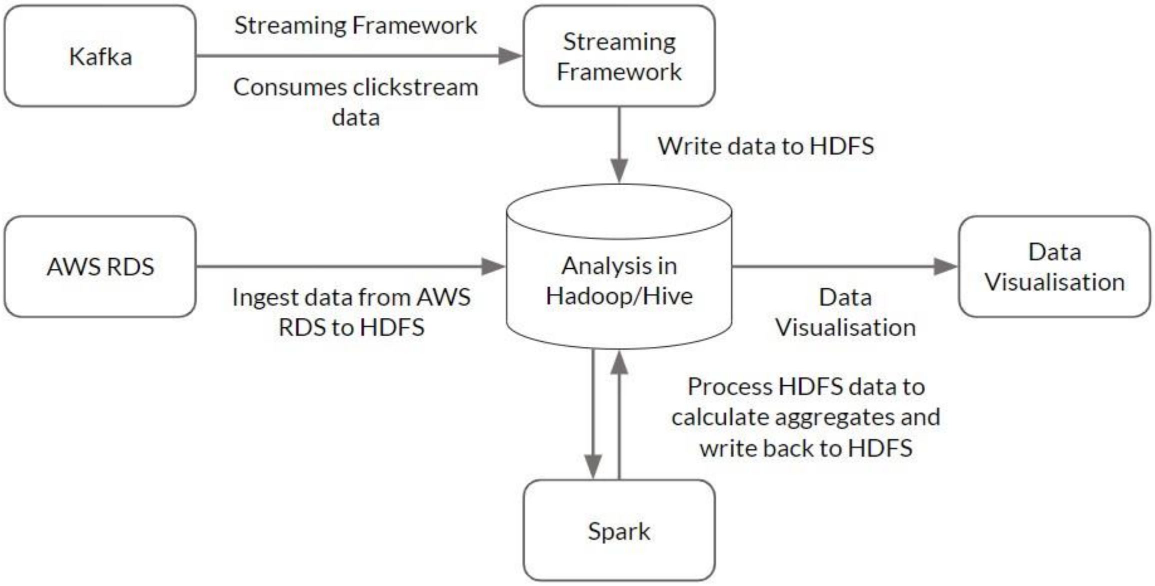
Logic For First Submission

Overall Architecture:



According to the given architecture, we have performed the following steps:

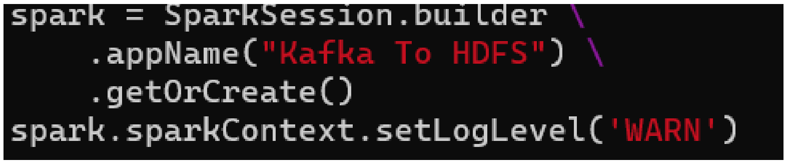
In the first step, we ingest stream data from Kafka into a stream processing framework. Then, we consume batch data from AWS RDS into HDFS. Where data aggregation is necessary, we read and write our data from Spark to HDFS. The final aggregated data is stored in Hive tables. The data in Hive tables is later queried to calculate different metrics.

For the first submission (MID), we consume data from Kafka and RDS and load them into Hive managed tables. This process is split into four tasks:

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

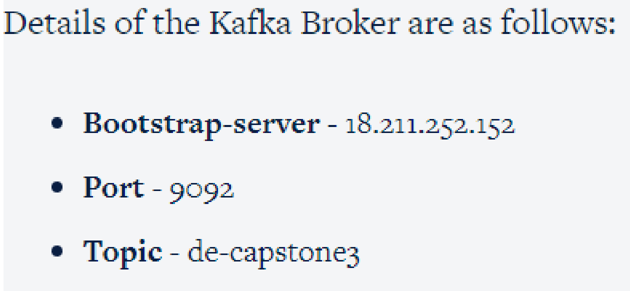
Please find commented file “**spark_kafka_to_local.py**”

1) At first, we create a Spark session with application name “Kafka to HDFS”



```
spark = SparkSession.builder \
    .appName("Kafka To HDFS") \
    .getOrCreate()
spark.sparkContext.setLogLevel('WARN')
```

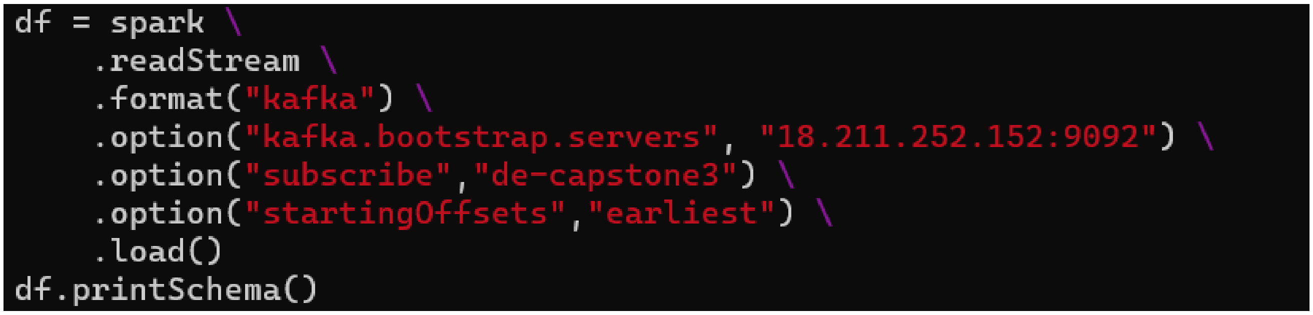
2) Then we read the incoming clickstream data from Kafka and storing the data in the hdfs. To do this, we use the credentials provided on the platform.



Details of the Kafka Broker are as follows:

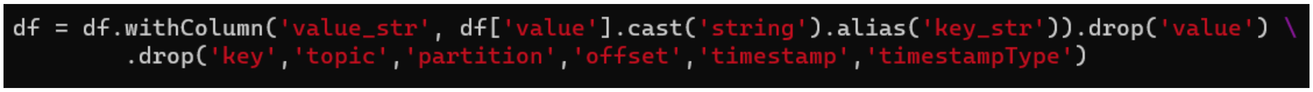
- **Bootstrap-server** - 18.211.252.152
- **Port** - 9092
- **Topic** - de-capstone3

The code to read the data is shown below:



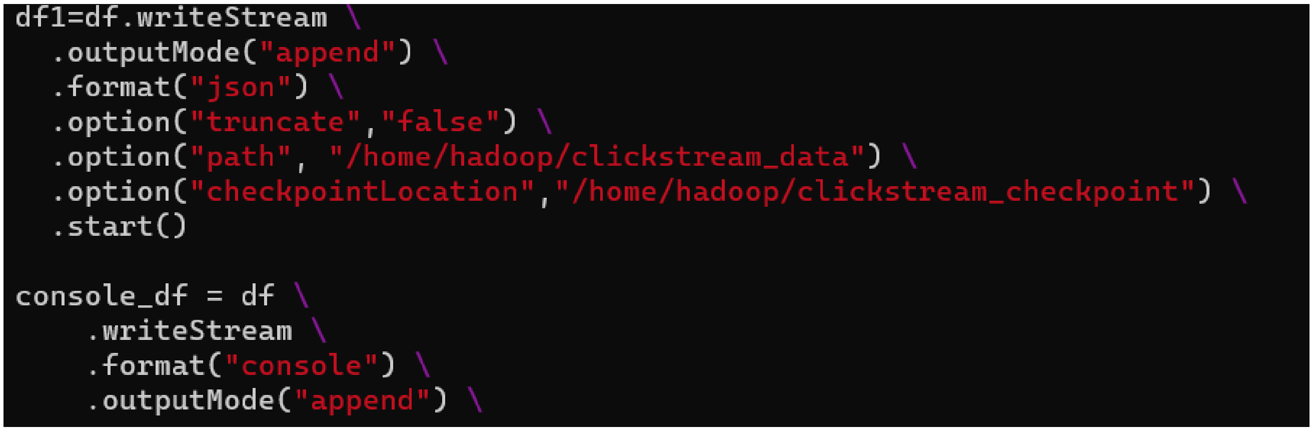
```
df = spark \
    .readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "18.211.252.152:9092") \
    .option("subscribe","de-capstone3") \
    .option("startingOffsets","earliest") \
    .load()
df.printSchema()
```

3) Then we transform the data by dropping a few columns and changing **value** column data type



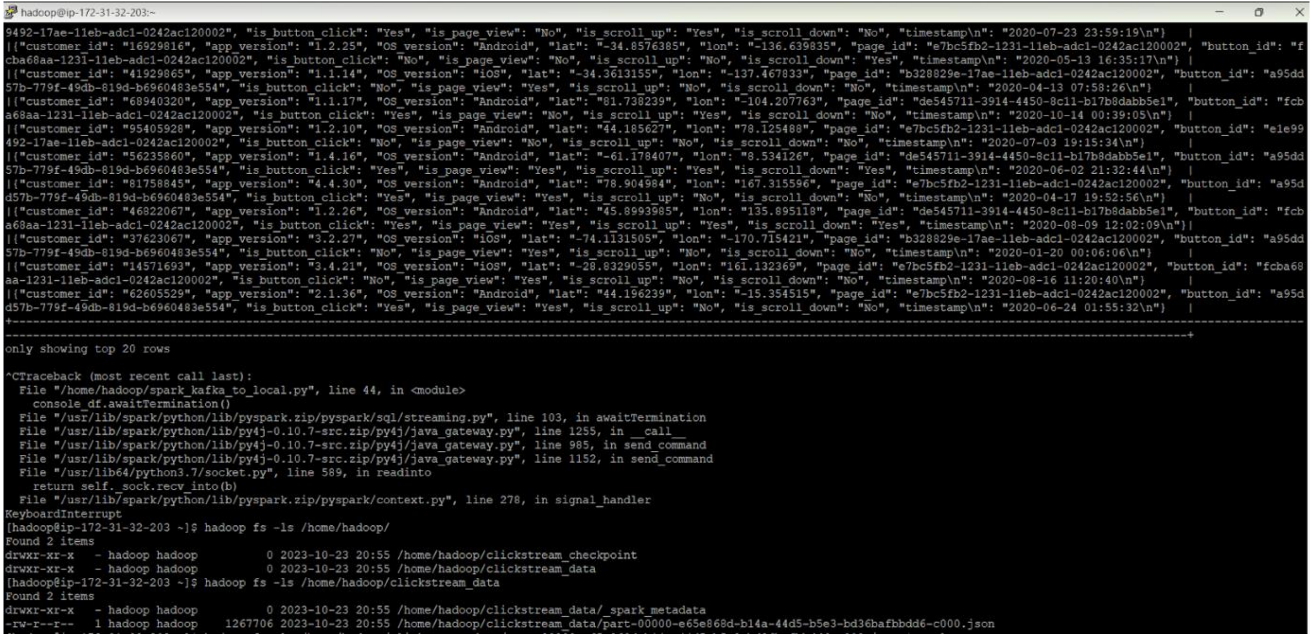
```
df = df.withColumn('value_str', df['value'].cast('string').alias('key_str')).drop('value') \
        .drop('key','topic','partition','offset','timestamp','timestampType')
```

4) We write the data to hdfs directory and keep it running until terminated (Write stream in json format).



```
df1=df.writeStream \
  .outputMode("append") \
  .format("json") \
  .option("truncate","false") \
  .option("path", "/home/hadoop/clickstream_data") \
  .option("checkpointLocation","/home/hadoop/clickstream_checkpoint") \
  .start()

console_df = df \
    .writeStream \
    .format("console") \
    .outputMode("append") \
```

Code execution output:
After running the code, we get the following output



```
9492-17ae-11eb-adc1-0242ac120002", "is_button_click": "Yes", "is_page_view": "No", "is_scroll_up": "Yes", "is_scroll_down": "No", "timestamp\n": "2020-07-23 23:59:19\n"}   |
|{"customer_id": "16929816", "app_version": "1.2.25", "OS_version": "Android", "lat": "-34.8576385", "lon": "-136.639835", "page_id": "e7bc5fb2-1231-11eb-adc1-0242ac120002", "button_id": "f
cba68aa-1231-11eb-adc1-0242ac120002", "is_button_click": "No", "is_page_view": "No", "is_scroll_up": "No", "is_scroll_down": "Yes", "timestamp\n": "2020-05-13 16:35:17\n"} |
|{"customer_id": "41929865", "app_version": "1.1.14", "OS_version": "iOS", "lat": "-34.3613155", "lon": "-137.467833", "page_id": "b328829e-17ae-11eb-adc1-0242ac120002", "button_id": "a95dd
57b-779f-49db-819d-b6960483e554", "is_button_click": "No", "is_page_view": "Yes", "is_scroll_up": "No", "is_scroll_down": "No", "timestamp\n": "2020-04-13 07:58:26\n"}   |
|{"customer_id": "68940320", "app_version": "1.1.17", "OS_version": "Android", "lat": "81.738239", "lon": "-104.207763", "page_id": "de545711-3914-4450-8c11-b17b8dabb5e1", "button_id": "fcb
a68aa-1231-11eb-adc1-0242ac120002", "is_button_click": "Yes", "is_page_view": "No", "is_scroll_up": "Yes", "is_scroll_down": "No", "timestamp\n": "2020-10-14 00:39:05\n"}  |
|{"customer_id": "95405928", "app_version": "1.2.10", "OS_version": "Android", "lat": "44.185627", "lon": "78.125488", "page_id": "e7bc5fb2-1231-11eb-adc1-0242ac120002", "button_id": "e1e99
492-17ae-11eb-adc1-0242ac120002", "is_button_click": "No", "is_page_view": "No", "is_scroll_up": "No", "is_scroll_down": "No", "timestamp\n": "2020-07-03 19:15:34\n"}   |
|{"customer_id": "56235860", "app_version": "1.4.16", "OS_version": "Android", "lat": "-61.178407", "lon": "8.534126", "page_id": "de545711-3914-4450-8c11-b17b8dabb5e1", "button_id": "a95dd
57b-779f-49db-819d-b6960483e554", "is_button_click": "Yes", "is_page_view": "Yes", "is_scroll_up": "Yes", "is_scroll_down": "Yes", "timestamp\n": "2020-06-02 21:32:44\n"}  |
|{"customer_id": "81758845", "app_version": "4.4.30", "OS_version": "Android", "lat": "78.904984", "lon": "167.315596", "page_id": "e7bc5fb2-1231-11eb-adc1-0242ac120002", "button_id": "a95d
d57b-779f-49db-819d-b6960483e554", "is_button_click": "Yes", "is_page_view": "Yes", "is_scroll_up": "No", "is_scroll_down": "No", "timestamp\n": "2020-04-17 19:52:56\n"}   |
|{"customer_id": "46822067", "app_version": "1.2.26", "OS_version": "Android", "lat": "45.8993985", "lon": "135.895118", "page_id": "de545711-3914-4450-8c11-b17b8dabb5e1", "button_id": "fcb
a68aa-1231-11eb-adc1-0242ac120002", "is_button_click": "Yes", "is_page_view": "Yes", "is_scroll_up": "Yes", "is_scroll_down": "Yes", "timestamp\n": "2020-08-09 12:02:09\n"}|
|{"customer_id": "37623067", "app_version": "3.2.27", "OS_version": "iOS", "lat": "-74.1131505", "lon": "-170.715421", "page_id": "b328829e-17ae-11eb-adc1-0242ac120002", "button_id": "a95dd
57b-779f-49db-819d-b6960483e554", "is_button_click": "No", "is_page_view": "Yes", "is_scroll_up": "No", "is_scroll_down": "No", "timestamp\n": "2020-01-20 00:06:06\n"}   |
|{"customer_id": "14571693", "app_version": "3.4.21", "OS_version": "iOS", "lat": "-28.8329055", "lon": "161.132369", "page_id": "e7bc5fb2-1231-11eb-adc1-0242ac120002", "button_id": "fcba68
aa-1231-11eb-adc1-0242ac120002", "is_button_click": "No", "is_page_view": "Yes", "is_scroll_up": "No", "is_scroll_down": "No", "timestamp\n": "2020-08-16 11:20:40\n"}   |
|{"customer_id": "62605529", "app_version": "2.1.36", "OS_version": "Android", "lat": "44.196239", "lon": "-15.354515", "page_id": "e7bc5fb2-1231-11eb-adc1-0242ac120002", "button_id": "a95d
d57b-779f-49db-819d-b6960483e554", "is_button_click": "Yes", "is_page_view": "Yes", "is_scroll_up": "No", "is_scroll_down": "No", "timestamp\n": "2020-06-24 01:55:32\n"}   |
+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------------+

only showing top 20 rows

^CTraceback (most recent call last):
  File "/home/hadoop/spark_kafka_to_local.py", line 44, in <module>
    console_df.awaitTermination()
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/sql/streaming.py", line 103, in awaitTermination
  File "/usr/lib/spark/python/lib/py4j-0.10.7-src.zip/py4j/java_gateway.py", line 1255, in __call__
  File "/usr/lib/spark/python/lib/py4j-0.10.7-src.zip/py4j/java_gateway.py", line 985, in send_command
  File "/usr/lib/spark/python/lib/py4j-0.10.7-src.zip/py4j/java_gateway.py", line 1152, in send_command
  File "/usr/lib64/python3.7/socket.py", line 589, in readinto
    return self._sock.recv_into(b)
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/context.py", line 278, in signal_handler
KeyboardInterrupt
[hadoop@ip-172-31-32-203 ~]$ hadoop fs -ls /home/hadoop/
Found 2 items
drwxr-xr-x   - hadoop hadoop          0 2023-10-23 20:55 /home/hadoop/clickstream_checkpoint
drwxr-xr-x   - hadoop hadoop          0 2023-10-23 20:55 /home/hadoop/clickstream_data
[hadoop@ip-172-31-32-203 ~]$ hadoop fs -ls /home/hadoop/clickstream_data
Found 2 items
drwxr-xr-x   - hadoop hadoop          0 2023-10-23 20:55 /home/hadoop/clickstream_data/_spark_metadata
-rw-r--r--   1 hadoop hadoop    1267706 2023-10-23 20:55 /home/hadoop/clickstream_data/part-00000-e65e868d-b14a-44d5-b5e3-bd36bafbbdd6-c000.json
```

We now clean the loaded Kafka data to a more structured format and save it to a csv file.

Please find commented file “**spark_local_flatten.py**”

1) Firstly, we get the existing spark session “Kafka to HDFS”

```
spark = SparkSession.builder \
    .appName("Kafka To HDFS") \
    .getOrCreate()
spark.sparkContext.setLogLevel('WARN')
```

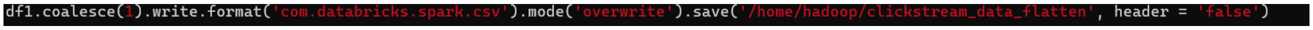
2) Then we read the clickstream data that we stored in json format into the dataframe

```
df = spark.read.json('/home/hadoop/clickstream_data/part*')
```

3) We extract the columns from json **value_str** in dataframe and create a new dataframe with new columns

```
df1 = df.select(
    get_json_object(df["value_str"],"$.customer_id").alias("customer_id"),
    get_json_object(df["value_str"],"$.app_version").alias("app_version"),
    get_json_object(df["value_str"],"$.OS_version").alias("OS_version"),
    get_json_object(df["value_str"],"$.lat").alias("lat"),
    get_json_object(df["value_str"],"$.lon").alias("lon"),
    get_json_object(df["value_str"],"$.page_id").alias("page_id"),
    get_json_object(df["value_str"],"$.button_id").alias("button_id"),
    get_json_object(df["value_str"],"$.is_button_click").alias("is_button_click"),
    get_json_object(df["value_str"],"$.is_page_view").alias("is_page_view"),
    get_json_object(df["value_str"],"$.is_scroll_up").alias("is_scroll_up"),
    get_json_object(df["value_str"],"$.is_scroll_down").alias("is_scroll_down"),
    get_json_object(df["value_str"],"$.timestamp\n").alias("click_timestamp")
)
```

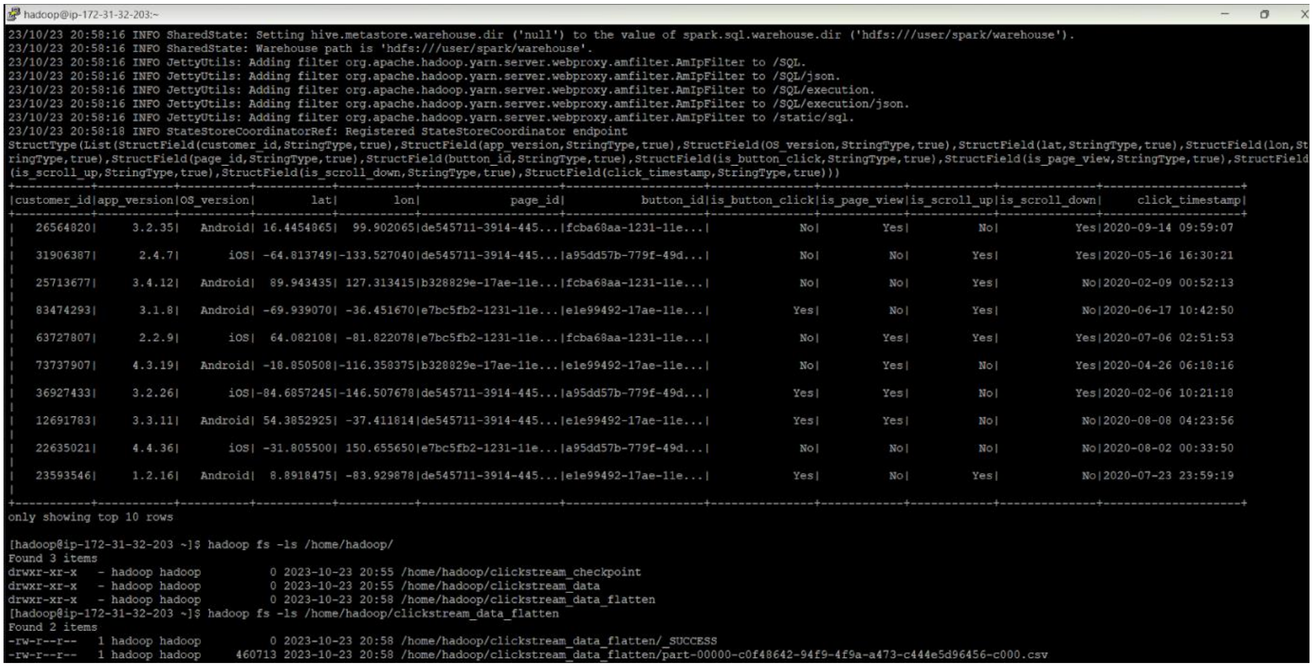
4) We then save the new dataframe to a csv file in hdfs directory



```
df1.coalesce(1).write.format('com.databricks.spark.csv').mode('overwrite').save('/home/hadoop/clickstream_data_flatten', header = 'false')
```

The steps to run the python files and the steps is provided in commented file “**step1.pdf**”

Code execution output:
After running the code, we get the following output



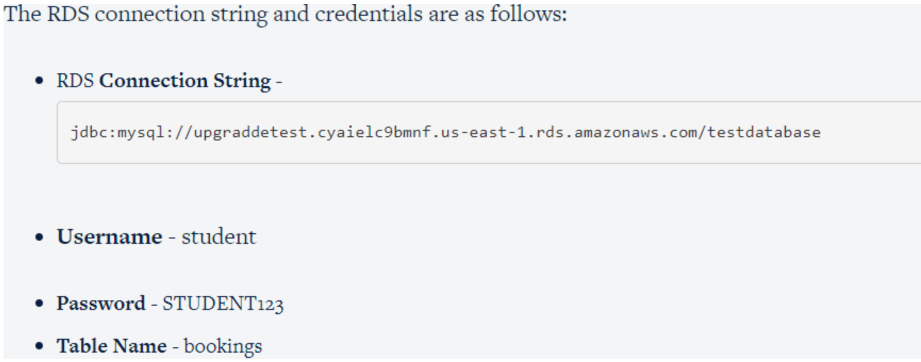
hadoop@ip-172-31-32-203:~

```
23/10/23 20:58:16 INFO SharedState: Setting hive.metastore.warehouse.dir ('null') to the value of spark.sql.warehouse.dir ('hdfs:///user/spark/warehouse').
23/10/23 20:58:16 INFO SharedState: Warehouse path is 'hdfs:///user/spark/warehouse'.
23/10/23 20:58:16 INFO JettyUtils: Adding filter org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter to /SQL.
23/10/23 20:58:16 INFO JettyUtils: Adding filter org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter to /SQL/json.
23/10/23 20:58:16 INFO JettyUtils: Adding filter org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter to /SQL/execution.
23/10/23 20:58:16 INFO JettyUtils: Adding filter org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter to /SQL/execution/json.
23/10/23 20:58:16 INFO JettyUtils: Adding filter org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter to /static/sql.
23/10/23 20:58:18 INFO StateStoreCoordinatorRef: Registered StateStoreCoordinator endpoint
StructType(List(StructField(customer_id,StringType,true),StructField(app_version,StringType,true),StructField(OS_version,StringType,true),StructField(lat,StringType,true),StructField(lon,St
ringType,true),StructField(page_id,StringType,true),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(click_timestamp,StringType,true)))
+-----------+-----------+----------+-----------+-----------+--------------------+--------------------+---------------+------------+------------+--------------+-------------------+
|customer_id|app_version|OS_version|        lat|        lon|             page_id|           button_id|is_button_click|is_page_view|is_scroll_up|is_scroll_down|    click_timestamp|
+-----------+-----------+----------+-----------+-----------+--------------------+--------------------+---------------+------------+------------+--------------+-------------------+
|   26564820|     3.2.35|   Android| 16.4454865|  99.902065|de545711-3914-445...|fcba68aa-1231-11e...|             No|         Yes|          No|           Yes|2020-09-14 09:59:07
|
|   31906387|      2.4.7|       iOS| -64.813749|-133.527040|de545711-3914-445...|a95dd57b-779f-49d...|             No|          No|         Yes|           Yes|2020-05-16 16:30:21
|
|   25713677|     3.4.12|   Android|  89.943435| 127.313415|b328829e-17ae-11e...|fcba68aa-1231-11e...|             No|          No|         Yes|            No|2020-02-09 00:52:13
|
|   83474293|      3.1.8|   Android| -69.939070| -36.451670|e7bc5fb2-1231-11e...|e1e99492-17ae-11e...|            Yes|          No|         Yes|            No|2020-06-17 10:42:50
|
|   63727807|      2.2.9|       iOS|  64.082108| -81.822078|e7bc5fb2-1231-11e...|fcba68aa-1231-11e...|             No|         Yes|         Yes|           Yes|2020-07-06 02:51:53
|
|   73737907|     4.3.19|   Android| -18.850508|-116.358375|b328829e-17ae-11e...|e1e99492-17ae-11e...|             No|         Yes|          No|           Yes|2020-04-26 06:18:16
|
|   36927433|     3.2.26|       iOS|-84.6857245|-146.507678|de545711-3914-445...|a95dd57b-779f-49d...|            Yes|         Yes|          No|           Yes|2020-02-06 10:21:18
|
|   12691783|     3.3.11|   Android| 54.3852925| -37.411814|de545711-3914-445...|e1e99492-17ae-11e...|            Yes|         Yes|          No|            No|2020-08-08 04:23:56
|
|   22635021|     4.4.36|       iOS| -31.805500| 150.655650|e7bc5fb2-1231-11e...|a95dd57b-779f-49d...|             No|          No|          No|            No|2020-08-02 00:33:50
|
|   23593546|     1.2.16|   Android|  8.8918475| -83.929878|de545711-3914-445...|e1e99492-17ae-11e...|            Yes|          No|         Yes|            No|2020-07-23 23:59:19
|
+-----------+-----------+----------+-----------+-----------+--------------------+--------------------+---------------+------------+------------+--------------+-------------------+
only showing top 10 rows

[hadoop@ip-172-31-32-203 ~]$ hadoop fs -ls /home/hadoop/
Found 3 items
drwxr-xr-x   - hadoop hadoop          0 2023-10-23 20:55 /home/hadoop/clickstream_checkpoint
drwxr-xr-x   - hadoop hadoop          0 2023-10-23 20:55 /home/hadoop/clickstream_data
drwxr-xr-x   - hadoop hadoop          0 2023-10-23 20:58 /home/hadoop/clickstream_data_flatten
[hadoop@ip-172-31-32-203 ~]$ hadoop fs -ls /home/hadoop/clickstream_data_flatten
Found 2 items
-rw-r--r--   1 hadoop hadoop          0 2023-10-23 20:58 /home/hadoop/clickstream_data_flatten/_SUCCESS
-rw-r--r--   1 hadoop hadoop     460713 2023-10-23 20:58 /home/hadoop/clickstream_data_flatten/part-00000-c0f48642-94f9-4f9a-a473-c444e5d96456-c000.csv
```

Task 2: Write a job to ingest the bookings data from AWS RDS to Hadoop

Please find commented file “**step2.pdf**” for the sqoop command

- To do this, we write a sqoop job using the credentials provided on the platform



The RDS connection string and credentials are as follows:

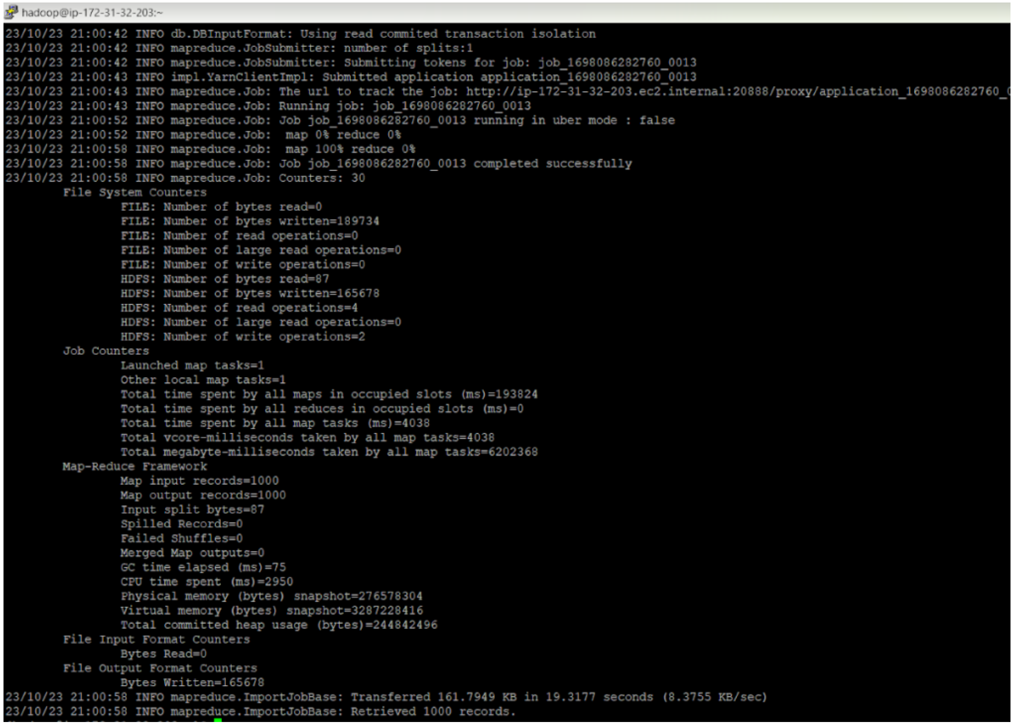
- RDS **Connection String** -

```
jdbc:mysql://upgraddetest.cyaielc9bmnf.us-east-1.rds.amazonaws.com/testdatabase
```

- **Username** - student
- **Password** - STUDENT123
- **Table Name** - bookings

```
sqoop import \
--connect jdbc:mysql://upgraddetest.cyaielc9bmnf.us-east-1.rds.amazonaws.com/testdatabase \
--table bookings \
--username student --password STUDENT123 \
--target-dir /home/hadoop/bookings_data \
-m1
```

Code execution output:
After running the code, we get the following output



```
23/10/23 21:00:42 INFO db.DBInputFormat: Using read commited transaction isolation
23/10/23 21:00:42 INFO mapreduce.JobSubmitter: number of splits:1
23/10/23 21:00:42 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1698086282760_0013
23/10/23 21:00:43 INFO impl.YarnClientImpl: Submitted application application_1698086282760_0013
23/10/23 21:00:43 INFO mapreduce.Job: The url to track the job: http://ip-172-31-32-203.ec2.internal:20888/proxy/application_1698086282760_
23/10/23 21:00:43 INFO mapreduce.Job: Running job: job_1698086282760_0013
23/10/23 21:00:52 INFO mapreduce.Job: Job job_1698086282760_0013 running in uber mode : false
23/10/23 21:00:52 INFO mapreduce.Job:  map 0% reduce 0%
23/10/23 21:00:58 INFO mapreduce.Job:  map 100% reduce 0%
23/10/23 21:00:58 INFO mapreduce.Job: Job job_1698086282760_0013 completed successfully
23/10/23 21:00:58 INFO mapreduce.Job: Counters: 30
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=189734
                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)=193824
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=4038
                Total vcore-milliseconds taken by all map tasks=4038
                Total megabyte-milliseconds taken by all map tasks=6202368
        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)=75
                CPU time spent (ms)=2950
                Physical memory (bytes) snapshot=276578304
                Virtual memory (bytes) snapshot=3287228416
                Total committed heap usage (bytes)=244842496
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=165678
23/10/23 21:00:58 INFO mapreduce.ImportJobBase: Transferred 161.7949 KB in 19.3177 seconds (8.3755 KB/sec)
23/10/23 21:00:58 INFO mapreduce.ImportJobBase: Retrieved 1000 records.
```

Task 3: Create aggregates for finding total bookings date-wise using Spark script

To prepare aggregates, data is read from HDFS, processed by a processing framework such as Spark and written back to HDFS to create a Hive table for the aggregated data.

Please find commented file “**datewise_bookings_aggregates_spark.py**”

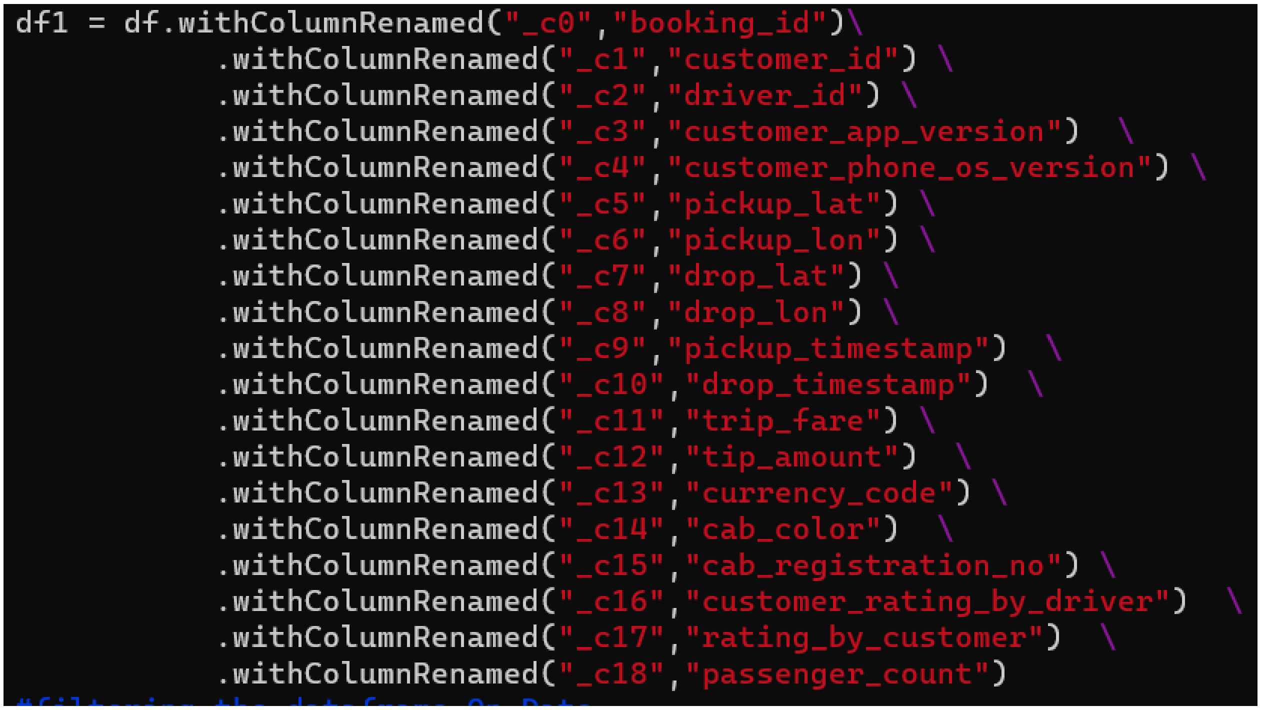
1) To perform aggregation of data, we first create a Spark session with application name “datewise_booking”

```
spark = SparkSession.builder.appName("datewise_booking").getOrCreate()
sc = spark.sparkContext
sc
```

2) Then we read the bookings data into the dataframe

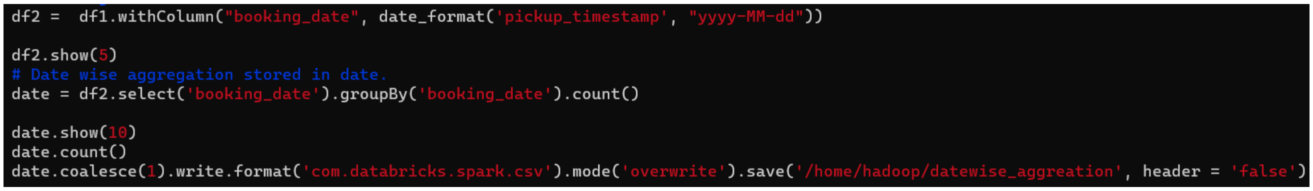
```
df = spark.read.csv("/home/hadoop/bookings_data/part*",inferSchema = True)
```

3) After the above step, we create a new dataframe with renamed columns



```
df1 = df.withColumnRenamed("_c0","booking_id")\
            .withColumnRenamed("_c1","customer_id") \
            .withColumnRenamed("_c2","driver_id") \
            .withColumnRenamed("_c3","customer_app_version")  \
            .withColumnRenamed("_c4","customer_phone_os_version") \
            .withColumnRenamed("_c5","pickup_lat") \
            .withColumnRenamed("_c6","pickup_lon") \
            .withColumnRenamed("_c7","drop_lat") \
            .withColumnRenamed("_c8","drop_lon") \
            .withColumnRenamed("_c9","pickup_timestamp")  \
            .withColumnRenamed("_c10","drop_timestamp")  \
            .withColumnRenamed("_c11","trip_fare") \
            .withColumnRenamed("_c12","tip_amount")  \
            .withColumnRenamed("_c13","currency_code") \
            .withColumnRenamed("_c14","cab_color")  \
            .withColumnRenamed("_c15","cab_registration_no") \
            .withColumnRenamed("_c16","customer_rating_by_driver")  \
            .withColumnRenamed("_c17","rating_by_customer")  \
            .withColumnRenamed("_c18","passenger_count")
#filtering the dataframe On Date
```

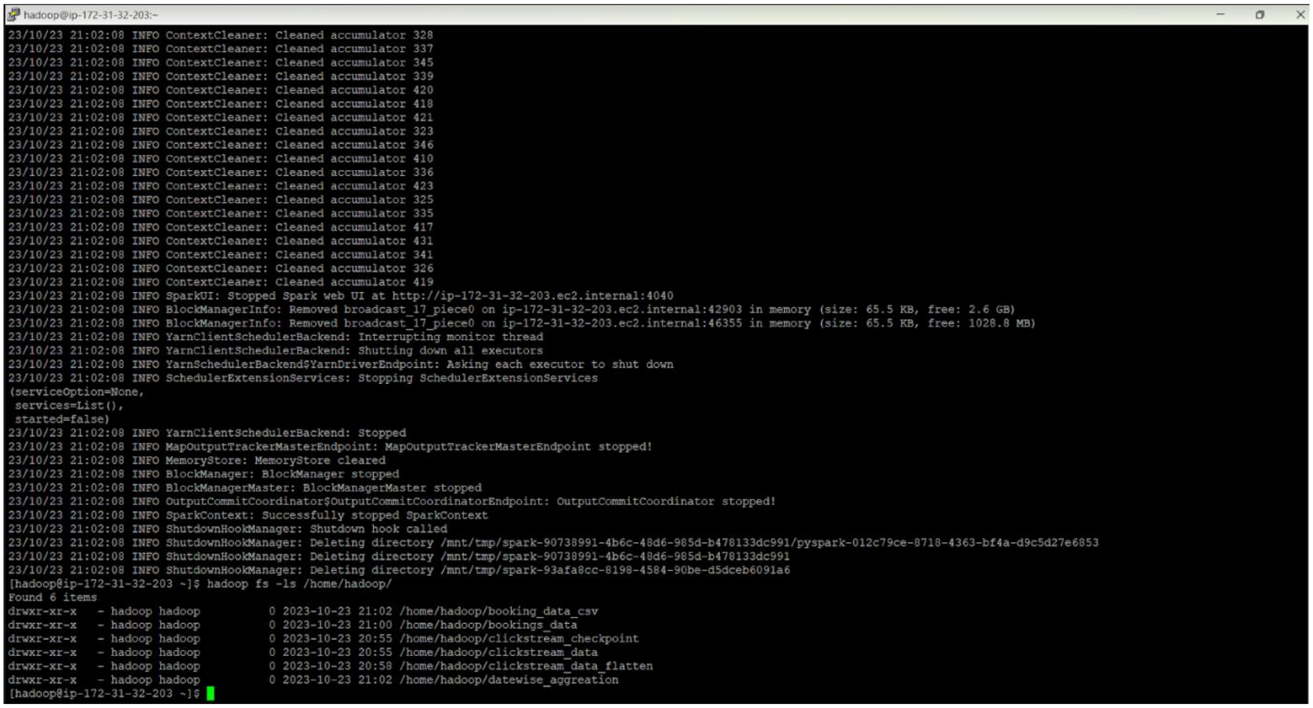
4) Then we create aggregations based on date and store the transformed data in a csv file



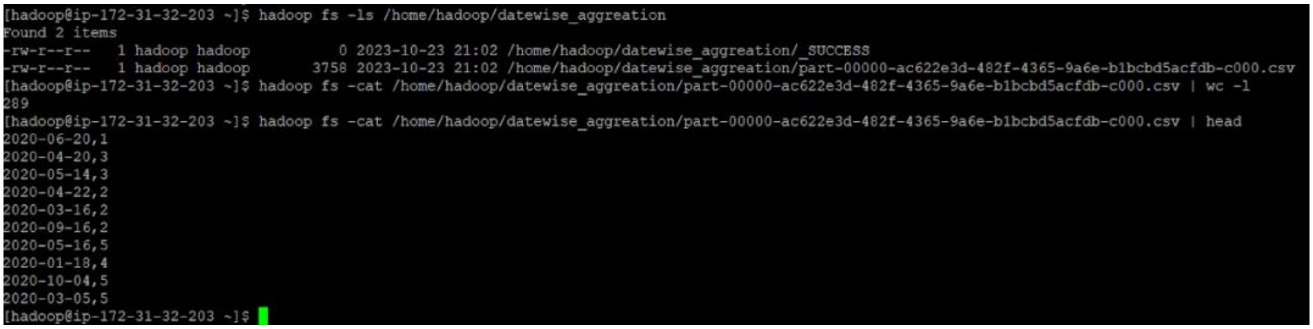
```
df2 =  df1.withColumn("booking_date", date_format('pickup_timestamp', "yyyy-MM-dd"))

df2.show(5)
# Date wise aggregation stored in date.
date = df2.select('booking_date').groupBy('booking_date').count()

date.show(10)
date.count()
date.coalesce(1).write.format('com.databricks.spark.csv').mode('overwrite').save('/home/hadoop/datewise_aggreation', header = 'false')
```

The steps to run the python file and the steps are provided in commented file “**step3.pdf**”

Code execution output:
After running the code, we get the following output



```
23/10/23 21:02:08 INFO ContextCleaner: Cleaned accumulator 328
23/10/23 21:02:08 INFO ContextCleaner: Cleaned accumulator 337
23/10/23 21:02:08 INFO ContextCleaner: Cleaned accumulator 345
23/10/23 21:02:08 INFO ContextCleaner: Cleaned accumulator 339
23/10/23 21:02:08 INFO ContextCleaner: Cleaned accumulator 420
23/10/23 21:02:08 INFO ContextCleaner: Cleaned accumulator 418
23/10/23 21:02:08 INFO ContextCleaner: Cleaned accumulator 421
23/10/23 21:02:08 INFO ContextCleaner: Cleaned accumulator 323
23/10/23 21:02:08 INFO ContextCleaner: Cleaned accumulator 346
23/10/23 21:02:08 INFO ContextCleaner: Cleaned accumulator 410
23/10/23 21:02:08 INFO ContextCleaner: Cleaned accumulator 336
23/10/23 21:02:08 INFO ContextCleaner: Cleaned accumulator 423
23/10/23 21:02:08 INFO ContextCleaner: Cleaned accumulator 325
23/10/23 21:02:08 INFO ContextCleaner: Cleaned accumulator 335
23/10/23 21:02:08 INFO ContextCleaner: Cleaned accumulator 417
23/10/23 21:02:08 INFO ContextCleaner: Cleaned accumulator 431
23/10/23 21:02:08 INFO ContextCleaner: Cleaned accumulator 341
23/10/23 21:02:08 INFO ContextCleaner: Cleaned accumulator 326
23/10/23 21:02:08 INFO ContextCleaner: Cleaned accumulator 419
23/10/23 21:02:08 INFO SparkUI: Stopped Spark web UI at http://ip-172-31-32-203.ec2.internal:4040
23/10/23 21:02:08 INFO BlockManagerInfo: Removed broadcast_17_piece0 on ip-172-31-32-203.ec2.internal:42903 in memory (size: 65.5 KB, free: 2.6 GB)
23/10/23 21:02:08 INFO BlockManagerInfo: Removed broadcast_17_piece0 on ip-172-31-32-203.ec2.internal:46355 in memory (size: 65.5 KB, free: 1028.8 MB)
23/10/23 21:02:08 INFO YarnClientSchedulerBackend: Interrupting monitor thread
23/10/23 21:02:08 INFO YarnClientSchedulerBackend: Shutting down all executors
23/10/23 21:02:08 INFO YarnSchedulerBackend$YarnDriverEndpoint: Asking each executor to shut down
23/10/23 21:02:08 INFO SchedulerExtensionServices: Stopping SchedulerExtensionServices
(serviceOption=None,
 services=List(),
 started=false)
23/10/23 21:02:08 INFO YarnClientSchedulerBackend: Stopped
23/10/23 21:02:08 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
23/10/23 21:02:08 INFO MemoryStore: MemoryStore cleared
23/10/23 21:02:08 INFO BlockManager: BlockManager stopped
23/10/23 21:02:08 INFO BlockManagerMaster: BlockManagerMaster stopped
23/10/23 21:02:08 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
23/10/23 21:02:08 INFO SparkContext: Successfully stopped SparkContext
23/10/23 21:02:08 INFO ShutdownHookManager: Shutdown hook called
23/10/23 21:02:08 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-90738991-4b6c-48d6-985d-b478133dc991/pyspark-012c79ce-8718-4363-bf4a-d9c5d27e6853
23/10/23 21:02:08 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-90738991-4b6c-48d6-985d-b478133dc991
23/10/23 21:02:08 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-93afa8cc-8198-4584-90be-d5dceb6091a6
[hadoop@ip-172-31-32-203 ~]$ hadoop fs -ls /home/hadoop/
Found 6 items
drwxr-xr-x   - hadoop hadoop          0 2023-10-23 21:02 /home/hadoop/booking_data_csv
drwxr-xr-x   - hadoop hadoop          0 2023-10-23 21:00 /home/hadoop/bookings_data
drwxr-xr-x   - hadoop hadoop          0 2023-10-23 20:55 /home/hadoop/clickstream_checkpoint
drwxr-xr-x   - hadoop hadoop          0 2023-10-23 20:55 /home/hadoop/clickstream_data
drwxr-xr-x   - hadoop hadoop          0 2023-10-23 20:58 /home/hadoop/clickstream_data_flatten
drwxr-xr-x   - hadoop hadoop          0 2023-10-23 21:02 /home/hadoop/datewise_aggreation
[hadoop@ip-172-31-32-203 ~]$
```

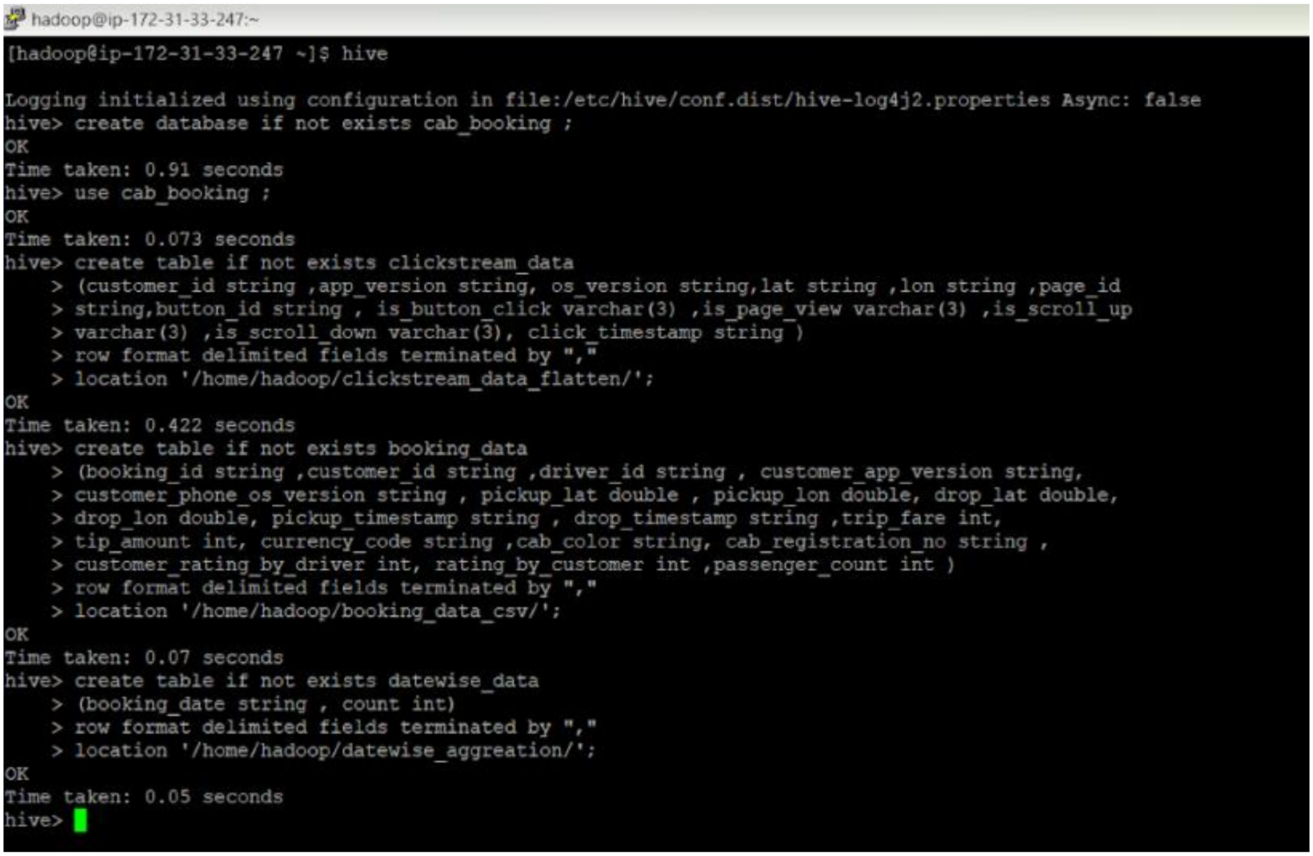


```
[hadoop@ip-172-31-32-203 ~]$ hadoop fs -ls /home/hadoop/datewise_aggreation
Found 2 items
-rw-r--r--   1 hadoop hadoop          0 2023-10-23 21:02 /home/hadoop/datewise_aggreation/_SUCCESS
-rw-r--r--   1 hadoop hadoop       3758 2023-10-23 21:02 /home/hadoop/datewise_aggreation/part-00000-ac622e3d-482f-4365-9a6e-b1bcbd5acfdb-c000.csv
[hadoop@ip-172-31-32-203 ~]$ hadoop fs -cat /home/hadoop/datewise_aggreation/part-00000-ac622e3d-482f-4365-9a6e-b1bcbd5acfdb-c000.csv | wc -l
289
[hadoop@ip-172-31-32-203 ~]$ hadoop fs -cat /home/hadoop/datewise_aggreation/part-00000-ac622e3d-482f-4365-9a6e-b1bcbd5acfdb-c000.csv | head
2020-06-20,1
2020-04-20,3
2020-05-14,3
2020-04-22,2
2020-03-16,2
2020-09-16,2
2020-05-16,5
2020-01-18,4
2020-10-04,5
2020-03-05,5
[hadoop@ip-172-31-32-203 ~]$
```

Task 4: Hive managed Table Creation

Please find commented file “**step4.pdf**”

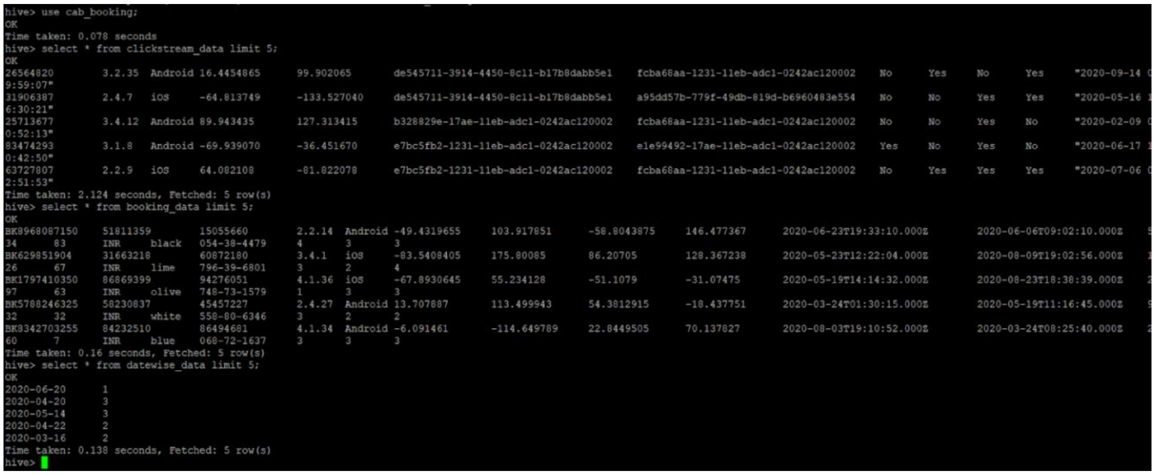
1) Create a Hive-managed table for clickstream data
2) Create a Hive-managed table for bookings data
3) Create a Hive-managed table for aggregated data in Task 3



```
[hadoop@ip-172-31-33-247 ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: false
hive> create database if not exists cab_booking ;
OK
Time taken: 0.91 seconds
hive> use cab_booking ;
OK
Time taken: 0.073 seconds
hive> create table if not exists clickstream_data
    > (customer_id string ,app_version string, os_version string,lat string ,lon string ,page_id
    > string,button_id string , is_button_click varchar(3) ,is_page_view varchar(3) ,is_scroll_up
    > varchar(3) ,is_scroll_down varchar(3), click_timestamp string )
    > row format delimited fields terminated by ","
    > location '/home/hadoop/clickstream_data_flatten/';
OK
Time taken: 0.422 seconds
hive> create table if not exists booking_data
    > (booking_id string ,customer_id string ,driver_id string , customer_app_version string,
    > customer_phone_os_version string , pickup_lat double , pickup_lon double, drop_lat double,
    > drop_lon double, pickup_timestamp string , drop_timestamp string ,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 int )
    > row format delimited fields terminated by ","
    > location '/home/hadoop/booking_data_csv/';
OK
Time taken: 0.07 seconds
hive> create table if not exists datewise_data
    > (booking_date string , count int)
    > row format delimited fields terminated by ","
    > location '/home/hadoop/datewise_aggreation/';
OK
Time taken: 0.05 seconds
hive> 
```

Code execution output:

After creating the Hive tables, we check if the data is properly loaded into them



```
hive> use cab_booking;
OK
Time taken: 0.078 seconds
hive> select * from clickstream_data limit 5;
OK
26564820        3.2.35  Android 16.4454865      99.902065       de545711-3914-4450-8c11-b17b8dabb5e1    fcba68aa-1231-11eb-adc1-0242ac120002    No      Yes     No      Yes     "2020-09-14 0
9:59:07"
31906387        2.4.7   iOS     -64.813749      -133.527040     de545711-3914-4450-8c11-b17b8dabb5e1    a95dd57b-779f-49db-819d-b6960483e554    No      No      Yes     Yes     "2020-05-16 1
6:30:21"
25713677        3.4.12  Android 89.943435       127.313415      b328829e-17ae-11eb-adc1-0242ac120002    fcba68aa-1231-11eb-adc1-0242ac120002    No      No      Yes     No      "2020-02-09 0
0:52:13"
83474293        3.1.8   Android -69.939070      -36.451670      e7bc5fb2-1231-11eb-adc1-0242ac120002    e1e99492-17ae-11eb-adc1-0242ac120002    Yes     No      Yes     No      "2020-06-17 1
0:42:50"
63727807        2.2.9   iOS     64.082108       -81.822078      e7bc5fb2-1231-11eb-adc1-0242ac120002    fcba68aa-1231-11eb-adc1-0242ac120002    No      Yes     Yes     Yes     "2020-07-06 0
2:51:53"
Time taken: 2.124 seconds, Fetched: 5 row(s)
hive> select * from booking_data limit 5;
OK
BK8968087150    51811359        15055660        2.2.14  Android -49.4319655     103.917851      -58.8043875     146.477367      2020-06-23T19:33:10.000Z        2020-06-06T09:02:10.000Z        5
34      83      INR     black   054-38-4479     4       3       3
BK629851904     31663218        60872180        3.4.1   iOS     -83.5408405     175.80085       86.20705        128.367238      2020-05-23T12:22:04.000Z        2020-08-09T19:02:56.000Z        1
26      67      INR     lime    796-39-6801     3       2       4
BK1797410350    86869399        94276051        4.1.36  iOS     -67.8930645     55.234128       -51.1079        -31.07475       2020-05-19T14:14:32.000Z        2020-08-23T18:38:39.000Z        2
97      63      INR     olive   748-73-1579     1       3       3
BK5788246325    58230837        45457227        2.4.27  Android 13.707887       113.499943      54.3812915      -18.437751      2020-03-24T01:30:15.000Z        2020-05-19T11:16:45.000Z        9
32      32      INR     white   558-80-6346     3       2       2
BK8342703255    84232510        86494681        4.1.34  Android -6.091461       -114.649789     22.8449505      70.137827       2020-08-03T19:10:52.000Z        2020-03-24T08:25:40.000Z        2
60      7       INR     blue    068-72-1637     3       3       3
Time taken: 0.16 seconds, Fetched: 5 row(s)
hive> select * from datewise_data limit 5;
OK
2020-06-20      1
2020-04-20      3
2020-05-14      3
2020-04-22      2
2020-03-16      2
Time taken: 0.138 seconds, Fetched: 5 row(s)
hive> 
```