Logic For First Submission

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

Steps:

1. **Import all libraries required for running the code.**

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

2. **Initialize the spark session and set app name as Kafka to local**

```
spark = SparkSession \
    .builder \
    .appName("Kafka-to-local") \
    .getOrCreate()
spark.sparkContext.setLogLevel('ERROR')
```

3. **Read data from Kafka by subscribing to de-capstone3 topic**

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

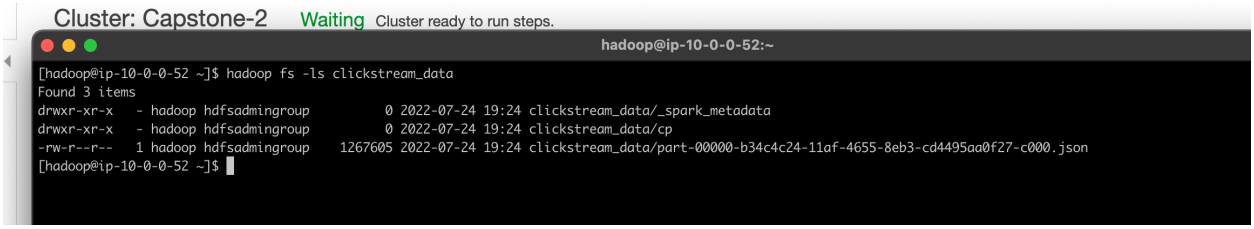
4. **Drop the columns which are not required and change the column name of value to value_str**

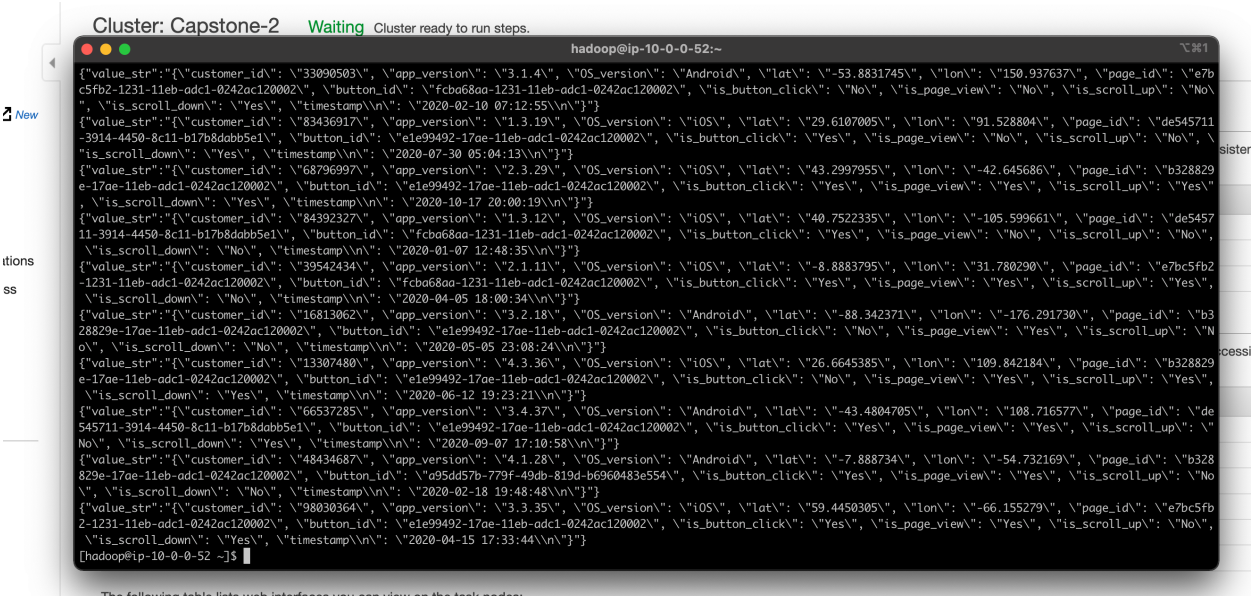
```
updated_data= kafka_read \
    .withColumn('value_str',kafka_read['value'].cast('string').alias('key_str')).drop('value') \
    .drop('key','topic','partition','offset','timestamp','timestampType')
```

5. **Write the data to hadoop using below code. We must use path where the raw clickstream data is stored as well as checkpoint path in HDFS**

```
clickstream_data = updated_data.writeStream \
  .format("json") \
  .outputMode("append") \
```

```
.option("truncate", "false") \
.option("path", "clickstream_data/") \
.option("checkpointLocation", "clickstream_data/cp/") \
.start()

clickstream_data.awaitTermination()
```





Flattening File:

Running spark_local_flattening.py to clean data for data analysis.

1. **Importing required libraries**

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
```

2. **Initialize Spark Session**

```
spark=SparkSession.builder.appName("Kafka-to-HDFS").master("local").getOrCreate()
spark
#Reading data from hdfs
df=spark.read.json("clickstream.json")
df.show(10,truncate=False)
```

3. Selecting the columns from the clickstream data set

```
df=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").alias("timestamp")
        )
df.printSchema()
df.show(10)
```

4. Writing the dataset to HDFS

```
df.coalesce(1).write.format('csv').mode('overwrite').save('/user/hadoop/clickstream_flattened',he
ader='true')
```

```
[hadoop@ip-10-0-0-52 ~]$ hadoop fs -ls clickstream_flattened
Found 2 items
-rw-r--r--   1 hadoop hdfsadmingroup          0 2022-07-24 20:44 clickstream_flattened/_SUCCESS
-rw-r--r--   1 hadoop hdfsadmingroup     403733 2022-07-24 20:44 clickstream_flattened/part-00000-6da4f575-4e02-471f-b726-1ad6f743c073-c000.csv
[hadoop@ip-10-0-0-52 ~]$
```

```
+------------------------------------------------------------------------------------------------
------------------------------------------------------------+
|value_str
                                                            |
+------------------------------------------------------------------------------------------------
------------------------------------------------------------+
|{"customer_id": "26564820", "app_version": "3.2.35", "OS_version": "Android", "lat": "16.4454865", "lon": "99.902065", "page_id": "de545711-3914-4450-8c11-b17b8dabb5e1", "button_id": "fcba68aa-1231-11eb-adc1-0242ac120
002", "is_button_click": "No", "is_page_view": "Yes", "is_scroll_up": "No", "is_scroll_down": "Yes", "timestamp\n": "2020-09-14 09:59:07\n"}  |
|{"customer_id": "31906387", "app_version": "2.4.7", "OS_version": "iOS", "lat": "-64.813749", "lon": "-133.527040", "page_id": "de545711-3914-4450-8c11-b17b8dabb5e1", "button_id": "a95dd57b-779f-49db-819d-b6960483e554
", "is_button_click": "No", "is_page_view": "No", "is_scroll_up": "Yes", "is_scroll_down": "Yes", "timestamp\n": "2020-05-16 16:30:21\n"}      |
|{"customer_id": "25713677", "app_version": "3.4.12", "OS_version": "Android", "lat": "89.943435", "lon": "127.313415", "page_id": "b328829e-17ae-11eb-adc1-0242ac120002", "button_id": "fcba68aa-1231-11eb-adc1-0242ac120
002", "is_button_click": "No", "is_page_view": "No", "is_scroll_up": "Yes", "is_scroll_down": "No", "timestamp\n": "2020-02-09 00:52:13\n"}   |
|{"customer_id": "83474293", "app_version": "3.1.8", "OS_version": "Android", "lat": "-69.939070", "lon": "-36.451670", "page_id": "e7bc5fb2-1231-11eb-adc1-0242ac120002", "button_id": "e1e99492-17ae-11eb-adc1-0242ac120
002", "is_button_click": "Yes", "is_page_view": "No", "is_scroll_up": "Yes", "is_scroll_down": "No", "timestamp\n": "2020-06-17 10:42:50\n"}  |
|{"customer_id": "63727807", "app_version": "2.2.9", "OS_version": "iOS", "lat": "64.082108", "lon": "-81.822078", "page_id": "e7bc5fb2-1231-11eb-adc1-0242ac120002", "button_id": "fcba68aa-1231-11eb-adc1-0242ac120002",
 "is_button_click": "No", "is_page_view": "Yes", "is_scroll_up": "Yes", "is_scroll_down": "Yes", "timestamp\n": "2020-07-06 02:51:53\n"}       |
|{"customer_id": "73737907", "app_version": "4.3.19", "OS_version": "Android", "lat": "-18.850508", "lon": "-116.358375", "page_id": "b328829e-17ae-11eb-adc1-0242ac120002", "button_id": "e1e99492-17ae-11eb-adc1-0242ac1
20002", "is_button_click": "No", "is_page_view": "Yes", "is_scroll_up": "No", "is_scroll_down": "Yes", "timestamp\n": "2020-04-26 06:18:16\n"}|
|{"customer_id": "36927433", "app_version": "3.2.26", "OS_version": "iOS", "lat": "-84.6857245", "lon": "-146.507678", "page_id": "de545711-3914-4450-8c11-b17b8dabb5e1", "button_id": "a95dd57b-779f-49db-819d-b6960483e5
54", "is_button_click": "Yes", "is_page_view": "Yes", "is_scroll_up": "No", "is_scroll_down": "Yes", "timestamp\n": "2020-02-06 10:21:18\n"}  |
|{"customer_id": "12691783", "app_version": "3.3.11", "OS_version": "Android", "lat": "54.3852925", "lon": "-37.411814", "page_id": "de545711-3914-4450-8c11-b17b8dabb5e1", "button_id": "e1e99492-17ae-11eb-adc1-0242ac12
0002", "is_button_click": "Yes", "is_page_view": "Yes", "is_scroll_up": "No", "is_scroll_down": "No", "timestamp\n": "2020-08-08 04:23:56\n"} |
|{"customer_id": "22635021", "app_version": "4.4.36", "OS_version": "iOS", "lat": "-31.805500", "lon": "150.655650", "page_id": "e7bc5fb2-1231-11eb-adc1-0242ac120002", "button_id": "a95dd57b-779f-49db-819d-b6960483e554
", "is_button_click": "No", "is_page_view": "No", "is_scroll_up": "No", "is_scroll_down": "No", "timestamp\n": "2020-08-02 00:33:50\n"}        |
|{"customer_id": "23593546", "app_version": "1.2.16", "OS_version": "Android", "lat": "8.8918475", "lon": "-83.929878", "page_id": "de545711-3914-4450-8c11-b17b8dabb5e1", "button_id": "e1e99492-17ae-11eb-adc1-0242ac120
002", "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"}  |
+------------------------------------------------------------------------------------------------
------------------------------------------------------------+
```

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

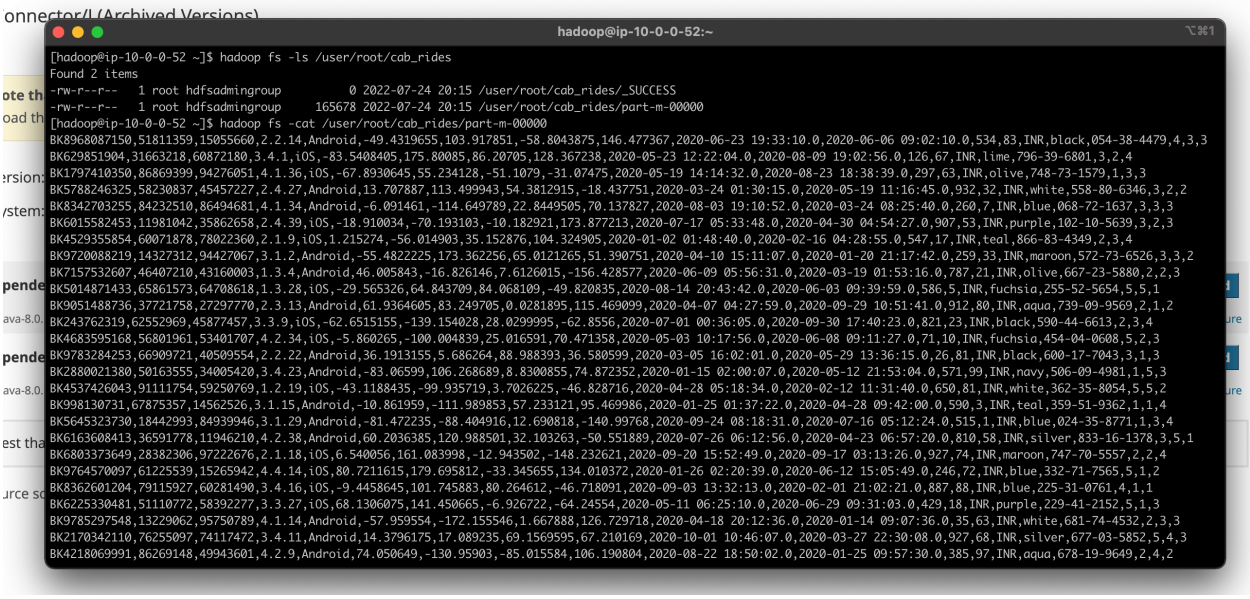
Steps:
For performing any analysis on booking data , first we need to import data from AWS RDS to HDFS.

Sqoop Import Command

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

To view imported data

hadoop fs -ls /user/hadoop/cab_rides



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

Steps:

1. **Import all libraries required for running the code.**

from pyspark.sql import SparkSession
*from pyspark.sql.functions import **

2. **Initialize the spark session and set app name as Kafka to local**

spark=SparkSession.builder.appName("datewise_bookings_aggregates_spark").master("local")
.getOrCreate()
spark.sparkContext.setLogLevel('ERROR')

3. **Read the data from HDFS which was loaded using Sqoop.**

df=spark.read.csv("/user/hadoop/cab_rides/part-m-00000")
df.show(10)
df.printSchema()

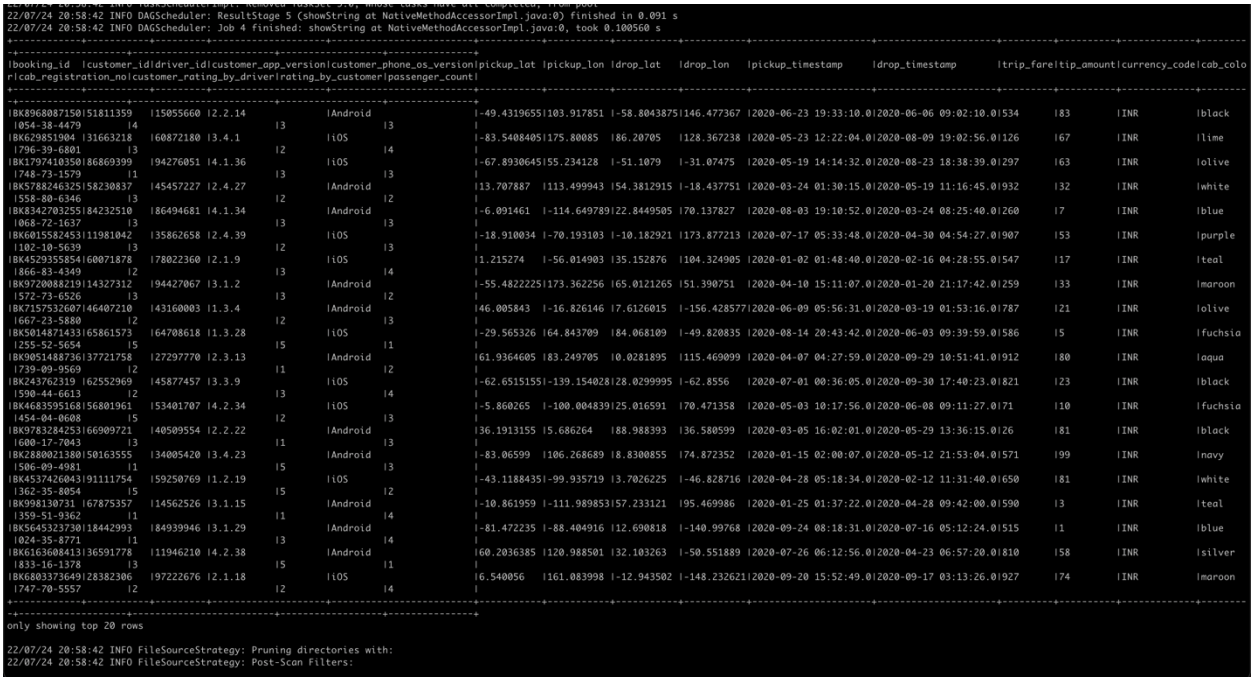
```
22/07/24 20:58:40 INFO DAGScheduler: Job 1 finished: showString at NativeMethodAccessorImpl.java:0, took 0.238127 s
22/07/24 20:58:40 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
+----------+-------+-----+-----+-----+----------+----------+---------+---------+--------------------+----+----+----+-------+----------+----+----+----+
|       _c0|    _c1|  _c2|  _c3|  _c4|       _c5|       _c6|      _c7|      _c8|                 _c9|                _c10|_c11|_c12|_c13|    _c14|      _c15|_c16|_c17|_c18|
+----------+-------+-----+-----+-----+----------+----------+---------+---------+--------------------+----+----+----+-------+----------+----+----+----+
|BK8968087150|51811359|150556660|2.2.14|Android|-49.4319655| 103.917851|-58.8043875| 146.477367|2020-06-23 19:33:...|2020-06-06 09:02:...| 534|  83| INR|  black|054-38-4479|   4|   3|   3|
| BK6298519041|31663218|60872180| 3.4.1|    iOS|-83.5408405|  175.80085|  86.20705| 128.367238|2020-05-23 12:22:...|2020-08-09 19:02:...| 126|  67| INR|   lime|796-39-6801|   3|   2|   4|
|BK1797410350|86869399|942760514|4.1.36|    iOS|-67.8930645|  55.234128|  -51.1079|  -31.07475|2020-05-19 14:14:...|2020-08-23 18:38:...| 297|  63| INR|  olive|748-73-1579|   1|   3|   3|
|BK5788246325|58230837|45457227|2.4.27|Android|  13.707887| 113.499943| 54.3812915| -18.437751|2020-03-24 01:30:...|2020-05-19 11:16:...| 932|  32| INR|  white|558-80-6346|   3|   2|   2|
|BK8342703255|84232510|86494681|4.1.34|Android|  -6.091461|-114.649789| 22.8449505|  70.137827|2020-08-03 19:10:...|2020-03-24 08:25:...| 260|   7| INR|   blue|068-72-1637|   3|   3|   3|
|BK6015582453|11981042|35862658|2.4.39|    iOS| -18.910034| -70.193103| -10.182921| 173.877213|2020-07-17 05:33:...|2020-04-30 04:54:...| 907|  53| INR| purple|102-10-5639|   3|   2|   3|
|BK4529355854|60071878|78022360| 2.1.9|    iOS|   1.215274| -56.014903|  35.152876| 104.324905|2020-01-02 01:48:...|2020-02-16 04:28:...| 547|  17| INR|   teal|866-83-4349|   2|   3|   4|
|BK9720088219|14327312|94427067| 3.1.2|Android|-55.4822225| 173.362256| 65.0121265|  51.390751|2020-04-10 15:11:...|2020-01-20 21:17:...| 259|  33| INR| maroon|572-73-6526|   3|   3|   2|
|BK7157532607|46407210|43160003| 1.3.4|Android|  46.005843| -16.826146|  7.6126015|-156.428577|2020-06-09 05:56:...|2020-03-19 01:53:...| 787|  21| INR|  olive|667-23-5880|   2|   2|   3|
|BK5014871433|65861573|64708618|1.3.28|    iOS| -29.565326|  64.843709|  84.068109| -49.820835|2020-08-14 20:43:...|2020-06-03 09:39:...| 586|   5| INR|fuchsia|255-52-5654|   5|   5|   1|
+----------+-------+-----+-----+-----+----------+----------+---------+---------+--------------------+----+----+----+-------+----------+----+----+----+
only showing top 10 rows

root
 |-- _c0: string (nullable = true)
 |-- _c1: string (nullable = true)
 |-- _c2: string (nullable = true)
 |-- _c3: string (nullable = true)
 |-- _c4: string (nullable = true)
 |-- _c5: string (nullable = true)
 |-- _c6: string (nullable = true)
 |-- _c7: string (nullable = true)
 |-- _c8: string (nullable = true)
 |-- _c9: string (nullable = true)
 |-- _c10: string (nullable = true)
 |-- _c11: string (nullable = true)
 |-- _c12: string (nullable = true)
 |-- _c13: string (nullable = true)
 |-- _c14: string (nullable = true)
 |-- _c15: string (nullable = true)
 |-- _c16: string (nullable = true)
 |-- _c17: string (nullable = true)
 |-- _c18: string (nullable = true)

22/07/24 20:58:40 INFO ContextCleaner: Cleaned accumulator 2
```

4. **Rename the columns**

new_col =
["booking_id","customer_id","driver_id","customer_app_version","customer_phone_os_version",
"pickup_lat","pickup_lon","drop_lat",
"drop_lon","pickup_timestamp","drop_timestamp","trip_fare","tip_amount","currency_code","cab
_color","cab_registration_no","customer_rating_by_driver","rating_by_customer","passenger_co
unt"]

*new_df = df.toDF(*new_col)*

new_df.show(truncate=False)

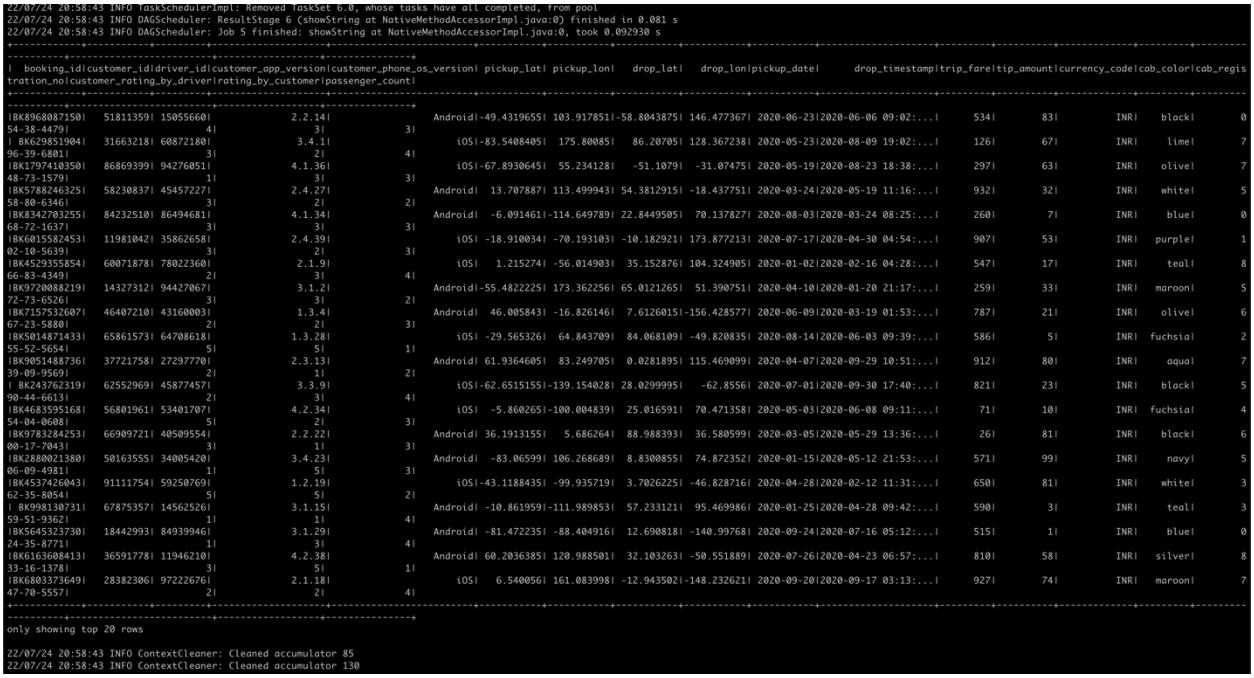


```
22/07/24 20:58:42 INFO DAGScheduler: ResultStage 5 (showString at NativeMethodAccessorImpl.java:0) finished in 0.091 s
22/07/24 20:58:42 INFO DAGScheduler: Job 4 finished: showString at NativeMethodAccessorImpl.java:0, took 0.100560 s
+----------+-----------+---------+--------------------+----------------------+----------+----------+----------+----------+--------------------+--------------------+---------+-------------+-------------+---------+
|booking_id|customer_id|driver_id|customer_app_version|customer_phone_os_version|pickup_lat|pickup_lon|drop_lat  |drop_lon  |pickup_timestamp    |drop_timestamp      |trip_fare|tip_amount   |currency_code|cab_color|
|cab_registration_no|customer_rating_by_driver|rating_by_customer|passenger_count|
+----------+-----------+---------+--------------------+----------------------+----------+----------+----------+----------+--------------------+--------------------+---------+-------------+-------------+---------+
|BK8968087150|51811359  |15055660 |2.2.14              |Android               |-49.4319655|103.917851|-58.8043875|146.477367|2020-06-23 19:33:10.0|2020-06-06 09:02:10.0|534      |83           |INR          |black    |
|054-38-4479 |4         |3        |3                   |
|BK6298851904|31663218  |60872180 |3.4.1               |iOS                   |-83.5408405|175.80085 |86.20705  |128.367238|2020-05-23 12:22:04.0|2020-08-09 19:02:56.0|126      |67           |INR          |lime     |
|796-39-6801 |3         |2        |4                   |
|BK1797410350|86869399  |94276051 |4.1.36              |iOS                   |-67.8930645|55.234128 |-51.1079  |-31.07475 |2020-05-19 14:14:32.0|2020-08-23 18:38:39.0|297      |63           |INR          |olive    |
|748-73-1579 |1         |3        |3                   |
|BK5788246325|58230837  |45457227 |2.4.27              |Android               |13.707887 |113.499943|54.3812915|-18.437751|2020-03-24 01:30:15.0|2020-05-19 11:16:45.0|932      |32           |INR          |white    |
|558-80-6346 |3         |2        |2                   |
|BK8342703255|84232510  |86494681 |4.1.34              |Android               |-6.091461 |-114.649789|22.8449505|70.137827 |2020-08-03 19:10:52.0|2020-03-24 08:25:40.0|260      |7            |INR          |blue     |
|068-72-1637 |3         |3        |3                   |
|BK6015582453|11981042  |35862658 |2.4.39              |iOS                   |-18.910034|-70.193103|-10.182921|173.877213|2020-07-17 05:33:48.0|2020-04-30 04:54:27.0|907      |53           |INR          |purple   |
|102-10-5639 |3         |2        |3                   |
|BK4529355854|60071878  |78022360 |2.1.9               |iOS                   |1.215274  |-56.014903|35.152876 |104.324905|2020-01-02 01:48:40.0|2020-02-16 04:28:55.0|547      |17           |INR          |teal     |
|866-83-4349 |2         |3        |4                   |
|BK9720088219|14327312  |94427067 |3.1.2               |Android               |-55.4822225|173.362256|65.0121265|51.390751 |2020-04-10 15:11:07.0|2020-01-20 21:17:42.0|259      |33           |INR          |maroon   |
|572-73-6526 |3         |3        |2                   |
|BK7157532607|46407210  |43160003 |1.3.4               |Android               |46.005843 |-16.826146|7.6126015 |-156.428577|2020-06-09 05:56:31.0|2020-03-19 01:53:16.0|787      |21           |INR          |olive    |
|667-23-5880 |2         |2        |3                   |
|BK5014871433|65861573  |64708618 |1.3.28              |iOS                   |-29.565326|64.843709 |84.068109 |-49.820835|2020-08-14 20:43:42.0|2020-06-03 09:39:59.0|586      |5            |INR          |fuchsia  |
|255-52-5654 |5         |5        |1                   |
|BK9051488736|37721758  |27297770 |2.3.13              |Android               |61.9364605|83.249705 |0.0281895 |115.469099|2020-04-07 04:27:59.0|2020-09-29 10:51:41.0|912      |80           |INR          |aqua     |
|739-09-9569 |2         |1        |2                   |
|BK243762319 |62552969  |45877457 |3.3.9               |iOS                   |-62.6515155|-139.154028|28.0299995|-62.8556  |2020-07-01 00:36:05.0|2020-09-30 17:40:23.0|821      |23           |INR          |black    |
|590-44-6613 |2         |3        |4                   |
|BK4683595168|56801961  |53401707 |4.2.34              |iOS                   |-5.860265 |-100.004839|25.016591 |70.471358 |2020-05-03 10:17:56.0|2020-06-08 09:11:27.0|71       |10           |INR          |fuchsia  |
|454-04-0608 |5         |2        |3                   |
|BK9783284253|66909721  |40509554 |2.2.22              |Android               |36.1913155|5.686264  |88.988393 |36.580599 |2020-03-05 16:02:01.0|2020-05-29 13:36:15.0|26       |81           |INR          |black    |
|600-17-7043 |3         |1        |3                   |
|BK2880021380|50163555  |34005420 |3.4.23              |Android               |-83.06599 |106.268689|8.8300855 |74.872352 |2020-01-15 02:00:07.0|2020-05-12 21:53:04.0|571      |99           |INR          |navy     |
|506-09-4981 |1         |5        |3                   |
|BK4537426043|91111754  |59250769 |1.2.19              |iOS                   |-43.1188435|-99.935719|3.7026225 |-46.828716|2020-04-28 05:18:34.0|2020-02-12 11:31:40.0|650      |81           |INR          |white    |
|362-35-8054 |5         |5        |2                   |
|BK998130731 |67875357  |14562526 |3.1.15              |Android               |-10.861959|-111.989853|57.233121 |95.469986 |2020-01-25 01:37:22.0|2020-04-28 09:42:00.0|590      |3            |INR          |teal     |
|359-51-9362 |1         |1        |4                   |
|BK5645323730|18442993  |84939946 |3.1.29              |Android               |-81.472235|-88.404916|12.690818 |-140.99768|2020-09-24 08:18:31.0|2020-07-16 05:12:24.0|515      |1            |INR          |blue     |
|024-35-8771 |1         |3        |4                   |
|BK6163608413|36591778  |11946210 |4.2.38              |Android               |60.2036385|120.988501|32.103263 |-50.551889|2020-07-26 06:12:56.0|2020-04-23 06:57:20.0|810      |58           |INR          |silver   |
|833-16-1378 |3         |5        |1                   |
|BK6803373649|28382306  |97222676 |2.1.18              |iOS                   |6.540056  |161.083998|-12.943502|-148.232621|2020-09-20 15:52:49.0|2020-09-17 03:13:26.0|927      |74           |INR          |maroon   |
|747-70-5557 |2         |2        |4                   |
+----------+-----------+---------+--------------------+----------------------+----------+----------+----------+----------+--------------------+--------------------+---------+-------------+-------------+---------+
only showing top 20 rows

22/07/24 20:58:42 INFO FileSourceStrategy: Pruning directories with:
22/07/24 20:58:42 INFO FileSourceStrategy: Post-Scan Filters:
```

5. **Converting pickup_timestamp to date by extracting date from pickup_timestamp for aggregation**

new_df=new_df.select("booking_id","customer_id","driver_id","customer_app_version","customer_phone_os_version","pickup_lat","pickup_lon","drop_lat", "drop_lon",to_date(col('pickup_timestamp')).alias('pickup_date').cast("date"),"drop_timestamp","trip_fare","tip_amount","currency_code","cab_color","cab_registration_no","customer_rating_by_driver","rating_by_customer","passenger_count")

new_df.show()



6. Aggregation on pickup date.

agg_df=new_df.groupBy("pickup_date").count().orderBy("pickup_date")

agg_df.show()

```
22/07/24 20:58:44 INFO DAGScheduler: ResultStage 9 (showString at NativeMethodAccessor
22/07/24 20:58:44 INFO DAGScheduler: Job 7 finished: showString at NativeMethodAccesso
22/07/24 20:58:44 INFO CodeGenerator: Code generated in 19.87628 ms
+-----------+-----+
|pickup_date|count|
+-----------+-----+
| 2020-01-01|    1|
| 2020-01-02|    3|
| 2020-01-03|    2|
| 2020-01-04|    2|
| 2020-01-05|    2|
| 2020-01-06|    3|
| 2020-01-07|    2|
| 2020-01-08|    4|
| 2020-01-09|    2|
| 2020-01-10|    2|
| 2020-01-11|    3|
| 2020-01-12|    3|
| 2020-01-14|    2|
| 2020-01-15|    5|
| 2020-01-16|    3|
| 2020-01-17|    4|
| 2020-01-18|    4|
| 2020-01-20|    4|
| 2020-01-21|    1|
| 2020-01-23|    4|
+-----------+-----+
only showing top 20 rows

22/07/24 20:58:44 INFO FileSourceStrategy: Pruning directories with:
22/07/24 20:58:44 INFO FileSourceStrategy: Post-Scan Filters:
22/07/24 20:58:44 INFO FileSourceStrategy: Output Data Schema: struct<_c9: string>
22/07/24 20:58:44 INFO FileSourceScanExec: Pushed Filters:
22/07/24 20:58:44 INFO FileSourceScanExec: Pushed Filters:
```

7. **Command to move above grouped data into HDFS.**

agg_df.coalesce(1).write.format('csv').mode('overwrite').save('/user/hadoop/datewise_bookings_agg',header='true')

```
[hadoop@ip-10-0-0-52 ~]$ hadoop fs -ls datewise_bookings_agg
Found 2 items
-rw-r--r--   1 hadoop hdfsadmingroup          0 2022-07-24 20:58 datewise_bookings_agg/_SUCCESS
-rw-r--r--   1 hadoop hdfsadmingroup       3776 2022-07-24 20:58 datewise_bookings_agg/part-00000-8bff7fc8-bf32-4cbd-bc83-5b82c09640cc-c000.csv
[hadoop@ip-10-0-0-52 ~]$
```

8. **Copy file to local**

hadoop fs -get /user/hadoop/datewise_bookings_agg/part-00000-8bff7fc8-bf32-4cbd-bc83-5b82c09640cc-c000.csv /home/hadoop/

Task 4:

Steps:
First, we must create database with capstone1 name. The tables which I have created for all three tasks are clickstream_data, booking_data, aggregate_datewise. Here the data is stored in csv format, so while creating tables, we have to use fields terminated by comma (,) .

Create a Hive-managed table for clickstream data.

```
create table if not exists clickstream_data (
    customer_id int,
    app_version string,
    os_version string,
    lat double,
    lon double,
    page_id varchar(100),
    button_id varchar(100),
    is_button_click string,
    is_page_view string,
    is_scroll_up string,
    is_scroll_down string,
    timestamp timestamp )
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
stored as textfile;
```

load data local inpath '/hadoop/clickstream/clickstream_flattened.csv' into table clickstream_data;

Create a Hive-managed table for bookings data.

```
create table if not exists booking_data (
    booking_id string,
    customer_id int,
    driver_id int,
    customer_app_version string,
    customer_phone_os_version string,
    pickup_lat double,
    pickup_lon double,
    drop_lat double,
    drop_lon double,
    pickup_timestamp timestamp,
    drop_timestamp timestamp,
    trip_fare int,
    tip_amount int,
```

```
    currency_code string,
    cab_color string,
    cab_registration_no int,
    customer_rating_by_driver varchar(100),
    rating_by_customer int,
    passenger_count int)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
stored as textfile;

load data local inpath '/hadoop/booking_data/part-m-00000' into table booking_data;
```

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

```
create table if not exists aggregate_datewise(
    pickup_date date,
    booking_id_count int
    )
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
stored as textfile;

load data local inpath '/hadoop/agg_datewise' into table aggregate_datewise;
```