

## Logic for First Submission

### Problem Statement:

'YourOwnCabs' (YOC) is an online on-demand cab booking service. it was doing around 100 rides per day. Owing to a successful business model and excellent service, the company's business is growing rapidly, and these numbers are breaking their own records every day. YOC's customer base and ride counts are increasing on a day-by-day basis.

It is highly important for business stakeholders to derive quick and on-demand insights regarding the numbers to decide the company's future strategy. Owing to the massive growth in business, it is getting tough for the company's management to obtain the business numbers frequently, as backend MySQL is not capable of catering to all types of queries owing to large volume data.

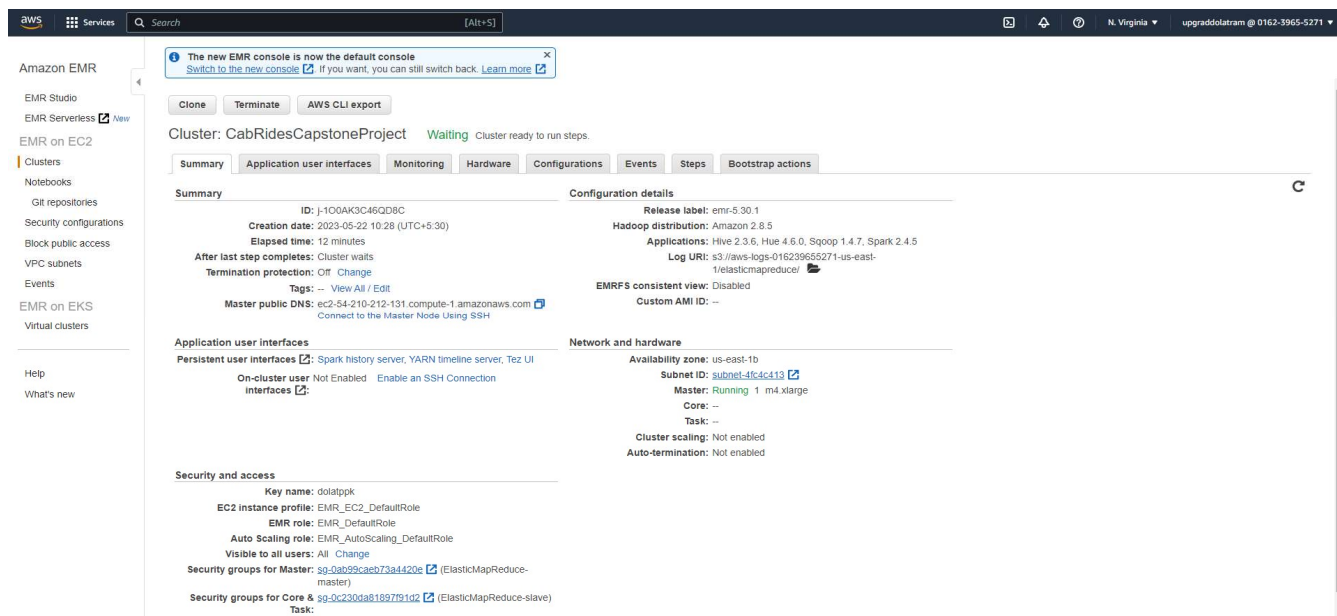
### Proposed Solution:

We, as a big data engineer, must architect and build a solution to provide a data driver analytical solution to analyze their booking and streaming data which can answer their questions backed by data to take business decisions.

1. **Booking data analytics solution:** This is a feature in which the ride-booking data is stored in a way such that it is available for all types of analytics without hampering business. These analyses mostly include daily, weekly and monthly booking counts as well as booking counts by the mobile operating system, average booking amount, total tip amount, etc.
2. **Click-stream data analytics solution:** Click-stream is the application browsing data that is generated on every action taken by the user on the app. This includes link click, button click, screen load, etc. This is relevant to know the user's intent and then take action to engage them more in the app according to their browsing behaviour. Since this is very high-volume data (more than the bookings data), it needs to be architecture quite well and stored in a proper format for analysis.

Booking and Click-Streaming data would be pushed into fully managed Hive distributed data warehouse where analyst can get required data without even worried query speed and storage to support business decisions

Created an EMR instance with services like Hadoop, Sqoop, Hive, Hue and Spark to run the services to complete Cab Ride Capstone Project Analysis.



The screenshot shows the Amazon EMR console interface. At the top, there's a navigation bar with the AWS logo and search bar. Below it, the left sidebar contains a menu with options like EMR Studio, EMR Serverless, EMR on EC2, Clusters, Notebooks, Git repositories, Security configurations, Block public access, VPC subnets, Events, EMR on EKS, and Virtual clusters. The main content area displays the details for a cluster named 'CabRidesCapstoneProject'. The cluster is in the 'Waiting' state, indicating it's ready to run steps. The console shows tabs for Summary, Application user interfaces, Monitoring, Hardware, Configurations, Events, Steps, and Bootstrap actions. The Summary tab is active, showing key information: ID (j-100AK3C46QD8C), Creation date (2023-05-22 10:28 UTC+5:30), Elapsed time (12 minutes), and Master public DNS (ec2-54-210-212-131.compute-1.amazonaws.com). Configuration details include Release label (emr-5.30.1), Hadoop distribution (Amazon 2.8.5), Applications (Hive 2.3.6, Hue 4.6.0, Sqoop 1.4.7, Spark 2.4.5), Log URI, and EMRFS consistent view (Disabled). Network and hardware settings show Availability zone (us-east-1b), Subnet ID (subnet-46dc413), Master (Running 1 m4.xlarge), and Core/Task instances (Not enabled). Security and access details include Key name (dolatppk), EC2 instance profile (EMR\_EC2\_DefaultRole), EMR role (EMR\_DefaultRole), Auto Scaling role (EMR\_AutoScaling\_DefaultRole), and Security groups for Master, Core, and Task instances.

## Step – 1: Reading Streaming data from kafka and load into HDFS

### Reading kafka stream and writing steaming data as .json file in HDFS local directory

Cab booking app event data are pushed in de-capstone3 Topic on Bootstrap-server - 18.211.252.152 Port – 9092. We need to read these streaming data and push into Hive managed table which further can be used for analysis. Below are steps followed to read kafka topic and store these information in Hive table.

#### 1. Reading data kafka.bootstrap.servers:

Pyspark file “spark\_kafka\_to\_local.py” created to read streaming data from the kafka server.

#### 2. Executed this file to dump streaming data into local HDFS directory:

Below listed commands were used to dump data in local directory, validate whether file is created or not and validate records in .json file.

```
spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 s3://dolat-s3/CapstoneFiles/spark_kafka_to_local.py
```

```
[hadoop@ip-172-31-33-215 ~]$ hadoop fs -ls /user/root
Found 2 items
drwxr-xr-x - hadoop hadoop 0 2023-05-23 12:12 /user/root/bookings
drwxr-xr-x - hadoop hadoop 0 2023-05-23 13:00 /user/root/datetime_bookings_agg
[hadoop@ip-172-31-33-215 ~]$ spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 s3://dolat-s3/CapstoneFiles/spark_kafka_to_local.py
```

```
hadoop fs -ls /user/root/clickstream_data_dump
```

```
[hadoop@ip-172-31-33-215 ~]$ hadoop fs -ls /user/root
Found 4 items
drwxr-xr-x - hadoop hadoop 0 2023-05-23 12:12 /user/root/bookings
drwxr-xr-x - hadoop hadoop 0 2023-05-23 13:31 /user/root/clickstream_data_dump
drwxr-xr-x - hadoop hadoop 0 2023-05-23 13:31 /user/root/clickstream_data_dump_cp
drwxr-xr-x - hadoop hadoop 0 2023-05-23 13:00 /user/root/datewise_bookings_agg
[hadoop@ip-172-31-33-215 ~]$ hadoop fs -ls /user/root/clickstream_data_dump
Found 2 items
drwxr-xr-x - hadoop hadoop 0 2023-05-23 13:31 /user/root/clickstream_data_dump/_spark_metadata
-rw-r--r-- 1 hadoop hadoop 1267706 2023-05-23 13:31 /user/root/clickstream_data_dump/part-00000-e08eb778-7b21-4c2b-84cd-9e2c6467c5ea-c000.json
[hadoop@ip-172-31-33-215 ~]$
```

**`hadoop fs -cat /user/root/clickstream_data_dump/part-00000-e08eb778-7b21-4c2b-84cd-9e2c6467c5ea-c000.json | head -n 5`**

```
[hadoop@ip-172-31-33-215 ~]$ hadoop fs -cat /user/root/clickstream_data_dump/part-00000-e08eb778-7b21-4c2b-84cd-9e2c6467c5ea-c000.json | head -n 5
{"value_str":{"customer_id":"26564820","app_version":"3.2.35","OS_version":"Android","lat":"16.4454865","lon":"99.902065
des45711-3914-4450-8c11-b17b8dabb5e1","button_id":"fcb6a68a-1231-11eb-adc1-0242ac120002","is_button_click":"No","is_page_view":"Yes
":"No","is_scroll_down":"Yes","timestamp":"2020-09-14 09:59:07\n\n"}}
{"value_str":{"customer_id":"31906387","app_version":"2.4.7","OS_version":"iOS","lat":"-64.813749","lon":"-133.527040",
45711-3914-4450-8c11-b17b8dabb5e1","button_id":"a95dd57b-779f-49db-819d-b6960483e554","is_button_click":"No","is_page_view":"No",
"Yes","is_scroll_down":"Yes","timestamp":"2020-05-16 16:30:21\n\n"}}
{"value_str":{"customer_id":"25713677","app_version":"3.4.12","OS_version":"Android","lat":"89.943435","lon":"127.313415
b32829e-17ae-11eb-adc1-0242ac120002","button_id":"fcb6a68a-1231-11eb-adc1-0242ac120002","is_button_click":"No","is_page_view":"No",
":"Yes","is_scroll_down":"No","timestamp":"2020-02-09 00:52:13\n\n"}}
{"value_str":{"customer_id":"83474293","app_version":"3.1.8","OS_version":"Android","lat":"-69.939070","lon":"-36.451670
e7bc5fb2-1231-11eb-adc1-0242ac120002","button_id":"e1e98492-17ae-11eb-adc1-0242ac120002","is_button_click":"Yes","is_page_view":"No
":"Yes","is_scroll_down":"No","timestamp":"2020-06-17 10:42:50\n\n"}}
{"value_str":{"customer_id":"63727807","app_version":"2.2.9","OS_version":"iOS","lat":"64.082108","lon":"81.822078",
fb2-1231-11eb-adc1-0242ac120002","button_id":"fcb6a68a-1231-11eb-adc1-0242ac120002","is_button_click":"No","is_page_view":"Yes",
Yes","is_scroll_down":"Yes","timestamp":"2020-07-06 02:51:53\n\n"}}
cat: Unable to write to output stream.
[hadoop@ip-172-31-33-215 ~]$
```

## Reading created json file from HDFS and transforming into structured format as csv file

Dump .json file created in above step is has lot of junk data which needs to be filtered out before loading into Hive managed table for further analysis.

Pyspark file "spark\_local\_flatten.py" created to transform into more structured format as load into .csv file in HDFS directory which will be then loaded in Hive managed table.

### 1. JSON transformation into more formatted .csv file:

Pyspark file "spark\_local\_flatten.py" was created to read streaming dump data from .json file and transform into more structured .csv file and load it into HDFS directory.

### 2. Executed this file to transform data into structured csv file and load local HDFS directory:

Below listed commands were used to transform json file and convert into more structured format.

**`spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 s3://dolat-s3/CapstoneFiles/spark_local_flatten.py`**

```
[hadoop@ip-172-31-33-215 ~]$ spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 s3://dolat-s3/CapstoneFiles/spark_local_flatten.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-c88c7eb4-949f-4a65-a8dd-c3de08117f30;1.0
  confs: [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
:: resolution report :: resolve 413ms :: artifacts dl 12ms
  :: 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 |
|               | number | search|download|evicted| number|download|
-----
| default      | 6      | 0      | 0         | 0      | 6      | 0      |
-----
```

```
hadoop fs -ls /user/root/clickstream_flattened
```

```
[hadoop@ip-172-31-33-215 ~]$ hadoop fs -ls /user/root/clickstream_flattened
Found 2 items
-rw-r--r-- 1 hadoop hadoop 0 2023-05-23 14:03 /user/root/clickstream_flattened/_SUCCESS
-rw-r--r-- 1 hadoop hadoop 403841 2023-05-23 14:03 /user/root/clickstream_flattened/part-00000-199eb4e5-0cac-44a1-a819-151d442489df-c000.csv
[hadoop@ip-172-31-33-215 ~]$
```

```
hadoop fs -cat /user/root/clickstream_flattened/part-00000-199eb4e5-0cac-44a1-a819-151d442489df-c000.csv | head -n 10
```

```
[hadoop@ip-172-31-33-215 ~]$ hadoop fs -cat /user/root/clickstream_flattened/part-00000-199eb4e5-0cac-44a1-a819-151d442489df-c000.csv | head -n 10
customer_id,app_version,OS_version,lat,lon,page_id,button_id,is_page_view,is_scroll_up,is_scroll_down,timestamp
26564920,3.2.35,Android,16.4454865,99.902065,de545711-3914-4450-8c11-b17b8dabb5e1,fcba68aa-1231-11eb-adc1-0242ac120002,No,Yes,No,Yes,""
31906387,2.4.7,iOS,-64.813749,-133.527040,de545711-3914-4450-8c11-b17b8dabb5e1,a95dd57b-779f-49db-819d-b6960483e554,No,No,Yes,Yes,""
25713677,3.4.12,Android,89.943435,127.313415,b328829e-17ae-11eb-adc1-0242ac120002,fcba68aa-1231-11eb-adc1-0242ac120002,No,No,Yes,No,""
83474293,3.1.8,Android,-69.939070,-36.451670,e7bc5fb2-1231-11eb-adc1-0242ac120002,e1e99492-17ae-11eb-adc1-0242ac120002,Yes,No,Yes,No,""
63727807,2.2.9,iOS,64.082108,-81.822078,e7bc5fb2-1231-11eb-adc1-0242ac120002,fcba68aa-1231-11eb-adc1-0242ac120002,No,Yes,Yes,Yes,""
73737907,4.3.19,Android,-18.850508,-116.358375,b328829e-17ae-11eb-adc1-0242ac120002,e1e99492-17ae-11eb-adc1-0242ac120002,No,Yes,No,Yes,""
36927433,3.2.26,iOS,-84.6857245,-146.507678,de545711-3914-4450-8c11-b17b8dabb5e1,a95dd57b-779f-49db-819d-b6960483e554,Yes,Yes,No,Yes,""
12691783,3.3.11,Android,54.3852925,-37.411814,de545711-3914-4450-8c11-b17b8dabb5e1,e1e99492-17ae-11eb-adc1-0242ac120002,Yes,Yes,No,No,""
22635021,4.4.36,iOS,-31.805500,150.655650,e7bc5fb2-1231-11eb-adc1-0242ac120002,a95dd57b-779f-49db-819d-b6960483e554,No,No,No,No,""
cat: Unable to write to output stream.
[hadoop@ip-172-31-33-215 ~]$
```

## Step - 2: Load data from AWS RDS to Hadoop

1. **Data Ingestion with Sqoop :** We need MySQL connector installed to make a bridge in between MySQL database and targeted location to retrieve required dataset from MySQL.

Commands executed to install MySQL connector:-

```
wget https://de-mysql-connector.s3.amazonaws.com/mysql-connector-java-8.0.25.tar.gz
tar -xvf mysql-connector-java-8.0.25.tar.gz
cd mysql-connector-java-8.0.25/
sudo cp mysql-connector-java-8.0.25.jar /usr/lib/sqoop/lib/
```

Executed Code Screenshot:-

```
[hadoop@ip-172-31-45-126 ~]$ wget https://de-mysql-connector.s3.amazonaws.com/mysql-connector-java-8.0.25.tar.gz
--2023-05-22 05:20:16-- https://de-mysql-connector.s3.amazonaws.com/mysql-connector-java-8.0.25.tar.gz
Resolving de-mysql-connector.s3.amazonaws.com (de-mysql-connector.s3.amazonaws.com)... 3.5.27.147, 3.5.27.150, 52.216.96.163,
Connecting to de-mysql-connector.s3.amazonaws.com (de-mysql-connector.s3.amazonaws.com)|3.5.27.147|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4079310 (3.9M) [application/x-gzip]
Saving to: 'mysql-connector-java-8.0.25.tar.gz'

100%[=====] 4079310 37.6 MB/s
2023-05-22 05:20:16 (37.6 MB/s) - 'mysql-connector-java-8.0.25.tar.gz' saved [4079310/4079310]

[hadoop@ip-172-31-45-126 ~]$ tar -xvf mysql-connector-java-8.0.25.tar.gz

[hadoop@ip-172-31-45-126 ~]$ cd mysql-connector-java-8.0.25/
[hadoop@ip-172-31-45-126 mysql-connector-java-8.0.25]$ sudo cp mysql-connector-java-8.0.25.jar /usr/lib/sqoop/lib/
[hadoop@ip-172-31-45-126 mysql-connector-java-8.0.25]$
```



- ### Sqoop Import executed Code Screenshot:-

**Note: 1000 records are retrieved from MySQL RDS and imported in HDFS**

- ```
hadoop fs -ls /user/root/bookings
```

```
[hadoop@ip-172-31-45-126 mysql-connector-java-8.0.25]$ hadoop fs -ls /user/root/bookings
Found 2 items
-rw-r--r--    1 hadoop hadoop           0 2023-05-22 05:23 /user/root/bookings/_SUCCESS
-rw-r--r--    1 hadoop hadoop 165678 2023-05-22 05:23 /user/root/bookings/part-m-00000
[hadoop@ip-172-31-45-126 mysql-connector-java-8.0.25]$
```

- ```
hadoop fs -cat /user/root/bookings/part-m-00000 | head -n 10
```

```
[hadoop@ip-172-31-45-126 mysql-connector-java-8.0.25]$ hadoop fs -cat /user/root/bookings/part-m-00000 | head -n 10
BK8968087150,51811359,15055660,2.2.14,Android,-49.4319655,103.917851,-58.8043875,146.477367,2020-06-23 19:33:10.0,2020-
,3
BK629851904,31663218,60872180,3.4.1,iOS,-83.5408405,175.80085,86.20705,128.367238,2020-05-23 12:22:04.0,2020-08-09 19:0
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:
BK5788246325,58230837,45457267,4.2.47,Android,13.707887,113.499943,54.3812915,-18.437751,2020-03-24 01:30:15.0,2020-05-
BK34242703255,84232510,86494681,4.1.34,Android,-6.091461,-114.649789,22.8449505,70.137827,2020-08-03 19:10:52.0,2020-03-
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
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 04:2
BK9720088219,14327312,94427067,3.1.2,Android,-55.482225,173.362256,65.0121265,51.390751,2020-04-10 15:11:07.0,2020-01-
BK7157532607,46407210,43160003,1.3.4,Android,46.005843,-16.826146,7.6126015,-156.428577,2020-06-09 05:56:31.0,2020-03-1
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
cat: Unable to write to output stream.
[hadoop@ip-172-31-45-126 mysql-connector-java-8.0.25]$
```

5. **Data validation:** Below screenshots we could see that no of bookings/records are exactly matching with validation document.

```
hive> select count(*) from bookings_detail;
Query ID = hadoop_20230522070613_0a860f69-63e0-40d4-bda0-317ddf70a90a
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1684732171643_0004)

-----
VERTICES      MODE      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
-----
Map 1 ..... container  SUCCEEDED    1          1          0          0          0          0
Reducer 2 ..... container  SUCCEEDED    1          1          0          0          0          0
-----
VERTICES: 02/02 [=====>>] 100% ELAPSED TIME: 5.63 s
-----
OK
1000
Time taken: 16.287 seconds, Fetched: 1 row(s)
hive>
```

## Bookings Table Count

Please check the number of records in the bookings table

Number of records - 1000

## Step – 3: Load Aggregated bookings data into Hadoop

Pyspark file “datewise\_bookings\_aggregates\_spark.py” created to aggregate total number of bookings by pickup date and save aggregated in .csv format and save in HDFS location.

### 1. Executed this file to save aggregated file as .csv format into HDFS location

```
spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5
/home/hadoop/datewise_bookings_aggregates_spark.py
```

```
[hadoop@ip-172-31-45-126 ~]$ spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 /home/hadoop/datewise_bookings_aggregates_spark.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-deb30924-f33f-481a-860c-7e29d090972f:1.0
  confs: [default]
```

In below screenshot we could see that datewise bookings aggregated file is saved on .csv format in /user/root/datewise\_bookings\_agg HDFS directory

```
[hadoop@ip-172-31-45-126 ~]$ hadoop fs -ls /user/root/datewise_bookings_agg/
Found 2 items
-rw-r--r-- 1 hadoop hadoop 0 2023-05-22 16:18 /user/root/datewise_bookings_agg/_SUCCESS
-rw-r--r-- 1 hadoop hadoop 3776 2023-05-22 16:18 /user/root/datewise_bookings_agg/part-00000-e40bc997-e4ef-41e4-9bda-6bbc32bf47b7-c000.csv
[hadoop@ip-172-31-45-126 ~]$
```

- Verifying data in saved file:** Executed `head -n 10` bash command to see top 10 records to make sure that pickup\_date level no of bookings are calculated correctly.

```
hadoop fs -cat /user/root/datewise_bookings_agg/part-00000-e40bc997-e4ef-41e4-9bda-6bbc32bf47b7-c000.csv | head -n 10
```

```
[hadoop@ip-172-31-45-126 ~]$ hadoop fs -cat /user/root/datewise_bookings_agg/part-00000-e40bc997-e4ef-41e4-9bda-6bbc32bf47b7-c000.csv | head -n 10
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
[hadoop@ip-172-31-45-126 ~]$
```

- Data validation:** Below screenshots we could see that aggregated booking counts are exactly matching with validation document.

```
hive> select count(*) from datewise_total_bookings;
Query ID = hadoop_20230522164903_abc976d8-1466-446a-b9f4-fe5ace38dbb2
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1684732171643_0009)

-----
VERTICES      MODE      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
-----
Map 1 ..... container  SUCCEEDED    1         1         0         0         0         0
Reducer 2 ..... container  SUCCEEDED    1         1         0         0         0         0
-----
VERTICES: 02/02  [=====>>>] 100%  ELAPSED TIME: 6.37 s
-----
OK
290
Time taken: 14.522 seconds, Fetched: 1 row(s)
hive>
```

## Bookings Aggregates Table Count

Please check the number of records in the bookings aggregates table

Number of records - 289

## ***Step – 4: Load data from AWS RDS to Hadoop creating Hive managed tables and loading relevant data in those tables.***

### **Creating database and tables and Loading the data into these Hive tables from HDFS files:**

Once these tables are created in Hive DW, next step is to load data from files saved in HDFS directory from MySQL Bookings and Kafka Streaming systems.

1. Created database and bookings, clickstream and datewise\_total\_bookings (aggregated) in it:

```
hive> create database cabrides;
OK
Time taken: 1.065 seconds
hive> use cabrides;
OK
Time taken: 0.058 seconds
hive> █
```

```
hive> CREATE TABLE IF NOT EXISTS clickstream_data (
  >   customer_id INT,
  >   app_version STRING,
  >   os_version STRING,
  >   lat DOUBLE,
  >   lon DOUBLE,
  >   page_id STRING,
  >   button_id STRING,
  >   is_button_click BOOLEAN,
  >   is_page_view BOOLEAN,
  >   is_scroll_up BOOLEAN,
  >   is_scroll_down BOOLEAN,
  >   time_stamp TIMESTAMP
  > )
  > COMMENT 'This table will store click streaming data red from kafka';
OK
Time taken: 0.067 seconds
hive> █
```

```
hive> CREATE TABLE IF NOT EXISTS datewise_total_bookings (
  >   pickup_date DATE,
  >   total_bookings INT
  > )
  > COMMENT 'This table will store aggregated count of booking by pickup date';
OK
Time taken: 0.075 seconds
hive> █
```



```
hive> CREATE TABLE IF NOT EXISTS bookings_detail (
  >   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 DECIMAL(10, 2),
  >   tip_amount DECIMAL(10, 2),
  >   currency_code STRING,
  >   cab_color STRING,
  >   cab_registration_no STRING,
  >   customer_rating_by_driver INT,
  >   rating_by_customer INT,
  >   passenger_count INT
  > )
  > COMMENT 'This table will store Bookings data red from MySQL RDS';
OK
Time taken: 0.069 seconds
hive>
```

## 2. Loading data into these table table:

```
LOAD DATA INPATH '/user/root/clickstream_data_dump/part-00000-e08eb778-7b21-4c2b-84cd-9e2c6467c5ea-c000.json' OVERWRITE INTO TABLE clickstream_data;
```

```
hive> LOAD DATA INPATH '/user/root/clickstream_data_dump/part-00000-e08eb778-7b21-4c2b-84cd-9e2c6467c5ea-c000.json' OVERWRITE INTO TABLE clickstream_data;
Loading data to table cabrides.clickstream_data
OK
Time taken: 1.073 seconds
```

```
LOAD DATA INPATH '/user/root/bookings/part-m-00000' OVERWRITE INTO TABLE bookings_detail;
```

```
hive> LOAD DATA INPATH '/user/root/bookings/part-m-00000' OVERWRITE INTO TABLE bookings_detail;
Loading data to table cabrides.bookings_detail
OK
Time taken: 0.615 seconds
hive>
```

```
LOAD DATA INPATH '/user/root/datewise_bookings_agg/part-00000-e40bc997-e4ef-41e4-9bda-6bbc32bf47b7-c000.csv' OVERWRITE INTO TABLE datewise_total_bookings;
```

```
hive> LOAD DATA INPATH '/user/root/datewise_bookings_agg/part-00000-e40bc997-e4ef-41e4-9bda-6bbc32bf47b7-c000.csv' OVERWRITE INTO TABLE datewise_total_bookings;
Loading data to table default.datewise_total_bookings
OK
Time taken: 0.416 seconds
hive>
```

3. **Data validation:** Below screenshots we could see that no of events/records are exactly matching with validation document.

#### Clickstream\_date Table:

```
hive> select count(*) from clickstream_data;
Query ID = hadoop_20230523142148_d6bd8179-6ea4-4fa2-9b50-4b19bc4a9fab
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1684823754082_0014)
```

	VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1 .....	container	SUCCEEDED	1	1	0	0	0	0	0
Reducer 2 .....	container	SUCCEEDED	1	1	0	0	0	0	0

```
VERTICES: 02/02 [=====>>] 100% ELAPSED TIME: 5.20 s
OK
3003
Time taken: 14.435 seconds, Fetched: 1 row(s)
hive>
```

#### Clickstream Table Count

Please check the number of records in the clickstream table

Number of records - 2984

#### Bookings\_detail table:

```
hive> select count(*) from bookings_detail;
Query ID = hadoop_20230522070613_0a860f69-63e0-40d4-bda0-317ddf70a90a
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1684732171643_0004)
```

	VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1 .....	container	SUCCEEDED	1	1	0	0	0	0	0
Reducer 2 .....	container	SUCCEEDED	1	1	0	0	0	0	0

```
VERTICES: 02/02 [=====>>] 100% ELAPSED TIME: 5.63 s
OK
1000
Time taken: 16.287 seconds, Fetched: 1 row(s)
hive>
```

## Bookings Table Count

Please check the number of records in the bookings table

Number of records - 1000

datewise\_total\_bookings table:

```
hive> select count(*) from datewise_total_bookings;
Query ID = hadoop_20230522164903_abc976d8-1466-446a-b9f4-fe5ace38dbb2
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1684732171643_0009)
```

```
-----
      VERTICES      MODE      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
-----
Map 1 ..... container  SUCCEEDED    1         1         0         0         0         0
Reducer 2 ..... container  SUCCEEDED    1         1         0         0         0         0
-----
VERTICES: 02/02  [=====>>] 100%  ELAPSED TIME: 6.37 s
-----
OK
290
Time taken: 14.522 seconds, Fetched: 1 row(s)
hive> █
```

## Bookings Aggregates Table Count

Please check the number of records in the bookings aggregates table

Number of records - 289