Logic

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

1. create a python file (spark_kafka_to_local.py) using vi command. This file will contain the code which will ingest the relevant data from Kafka into hadoop.

   vi spark_kafka_to_local.py

2. Explaining the code in spark_kafka_to_local.py

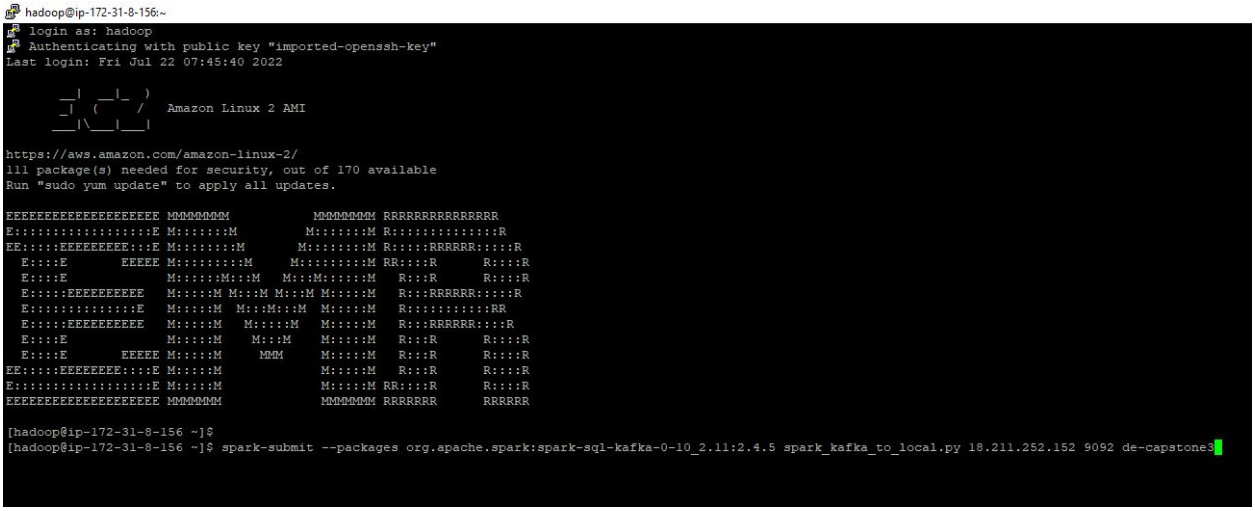
```
#  2.1 Importing the modules

import sys
import os
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *
from pyspark.sql.functions import from_json
from pyspark.sql.window import Window

# 2.2 Initializing Spark Session

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

# 2.3 Creating Dataframe from Kafka Data

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

clickstream_df.printSchema()

# 2.4 Transform dataframe by dropping few columns and changing value column data type

clickstream_df = clickstream_df \
```

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

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

# 2.5 Writing the dataframe to local file directory and keep it running until terminated

clickstream_df.writeStream \
    .outputMode("append") \
    .format("json") \
    .option("truncate","false") \
    .option("path", "clickstream_data") \
    .option("checkpointLocation","clickstream_checkpoint") \
    .start() \
    .awaitTermination()
```

3. Run Spark Submit command, to ingest the relevant data from Kafka into hadoop.

spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 spark_kafka_to_local.py 18.211.252.152 9092 de-capstone3



© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

```
[hadoop@ip-172-31-8-156 ~]$
[hadoop@ip-172-31-8-156 ~]$ spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 spark_kafka_to_local.py 18.211.252.152 9092 de-capstone3
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-57e07081-cc45-41dc-a2de-a757c3704b0a;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
downloading https://repo1.maven.org/maven2/org/apache/spark/spark-sql-kafka-0-10_2.11/2.4.5/spark-sql-kafka-0-10_2.11-2.4.5.jar ...
        [SUCCESSFUL ] org.apache.spark#spark-sql-kafka-0-10_2.11;2.4.5!spark-sql-kafka-0-10_2.11.jar (32ms)
downloading https://repo1.maven.org/maven2/org/apache/kafka/kafka-clients/2.0.0/kafka-clients-2.0.0.jar ...
        [SUCCESSFUL ] org.apache.kafka#kafka-clients;2.0.0!kafka-clients.jar (93ms)
downloading https://repo1.maven.org/maven2/org/spark-project/spark/unused/1.0.0/unused-1.0.0.jar ...
        [SUCCESSFUL ] org.spark-project.spark#unused;1.0.0!unused.jar (4ms)
downloading https://repo1.maven.org/maven2/org/lz4/lz4-java/1.4.0/lz4-java-1.4.0.jar ...
        [SUCCESSFUL ] org.lz4#lz4-java;1.4.0!lz4-java.jar (22ms)
downloading https://repo1.maven.org/maven2/org/xerial/snappy/snappy-java/1.1.7.3/snappy-java-1.1.7.3.jar ...
        [SUCCESSFUL ] org.xerial.snappy#snappy-java;1.1.7.3!snappy-java.jar(bundle) (89ms)
downloading https://repo1.maven.org/maven2/org/slf4j/slf4j-api/1.7.16/slf4j-api-1.7.16.jar ...
        [SUCCESSFUL ] org.slf4j#slf4j-api;1.7.16!slf4j-api.jar (4ms)
:: resolution report :: resolve 1799ms :: artifacts dl 254ms
        :: modules in use:
        org.apache.kafka#kafka-clients;2.0.0 from central in [default]
        org.apache.spark#spark-sql-kafka-0-10_2.11;2.4.5 from central in [default]
        org.lz4#lz4-java;1.4.0 from central in [default]
        org.slf4j#slf4j-api;1.7.16 from central in [default]
        org.spark-project.spark#unused;1.0.0 from central in [default]
        org.xerial.snappy#snappy-java;1.1.7.3 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   6   |   6   |   6   |   0   ||   6   |   6   |
        ---------------------------------------------------------------------
```

```
[hadoop@ip-172-31-8-156 ~]$ hadoop fs -ls
Found 2 items
drwxr-xr-x   - hadoop hadoop          0 2022-07-22 07:53 clickstream_checkpoint
drwxr-xr-x   - hadoop hadoop          0 2022-07-22 07:53 clickstream_data
[hadoop@ip-172-31-8-156 ~]$ hadoop fs -ls clickstream_data
Found 2 items
drwxr-xr-x   - hadoop hadoop          0 2022-07-22 07:53 clickstream_data/_spark_metadata
-rw-r--r--   1 hadoop hadoop    1267605 2022-07-22 07:53 clickstream_data/part-00000-9ff66f8b-d201-4aa3-8308-d053d4171d08-c000.json
[hadoop@ip-172-31-8-156 ~]$ hadoop fs -cat clickstream_data/part-00000-9ff66f8b-d201-4aa3-8308-d053d4171d08-c000.json |wc -l
3000
[hadoop@ip-172-31-8-156 ~]$
```

hadoop@ip-172-31-8-156:~

```
24353727,4.3.19,iOS,-50.925941,-126.568406,b328829e-17ae-11eb-adc1-0242ac120002,ele99492-17ae-11eb-adc1-0242ac120002,Yes,Yes,No,Yes,""
35269540,1.3.34,Android,81.834394,12.828982,de545711-3914-4450-8c11-b17b8dabb5e1,a95dd57b-779f-49db-819d-b6960483e554,Yes,No,Yes,No,""
25268401,3.3.3,Android,11.641779,-56.069123,de545711-3914-4450-8c11-b17b8dabb5e1,fcba68aa-1231-11eb-adc1-0242ac120002,Yes,No,Yes,No,""
95484571,1.2.29,iOS,77.143769,136.631731,b328829e-17ae-11eb-adc1-0242ac120002,ele99492-17ae-11eb-adc1-0242ac120002,Yes,No,No,Yes,""
31982975,4.1.37,iOS,70.187645,14.638339,e7bc5fb2-1231-11eb-adc1-0242ac120002,a95dd57b-779f-49db-819d-b6960483e554,Yes,Yes,Yes,Yes,""
40411724,1.3.19,iOS,6.371419,-15.472575,e7bc5fb2-1231-11eb-adc1-0242ac120002,a95dd57b-779f-49db-819d-b6960483e554,Yes,Yes,Yes,Yes,""
28353041,4.3.20,iOS,-48.737680,-89.546389,b328829e-17ae-11eb-adc1-0242ac120002,fcba68aa-1231-11eb-adc1-0242ac120002,Yes,Yes,Yes,Yes,""
64812182,4.4.3,Android,-82.7232805,102.600468,e7bc5fb2-1231-11eb-adc1-0242ac120002,fcba68aa-1231-11eb-adc1-0242ac120002,No,No,Yes,Yes,""
51541388,2.1.29,iOS,72.473664,61.959163,de545711-3914-4450-8c11-b17b8dabb5e1,a95dd57b-779f-49db-819d-b6960483e554,No,Yes,No,Yes,""
81230503,2.2.24,iOS,3.5693165,45.538827,b328829e-17ae-11eb-adc1-0242ac120002,ele99492-17ae-11eb-adc1-0242ac120002,Yes,No,Yes,No,""
75133221,1.3.19,Android,22.6653995,119.978964,de545711-3914-4450-8c11-b17b8dabb5e1,ele99492-17ae-11eb-adc1-0242ac120002,No,No,No,Yes,""
61336226,4.3.10,iOS,-47.988462,29.857122,e7bc5fb2-1231-11eb-adc1-0242ac120002,fcba68aa-1231-11eb-adc1-0242ac120002,No,No,Yes,Yes,""
94408812,2.3.27,Android,77.2576795,7.088533,b328829e-17ae-11eb-adc1-0242ac120002,ele99492-17ae-11eb-adc1-0242ac120002,Yes,No,Yes,Yes,""
12466128,3.1.24,Android,-57.6214705,150.225772,de545711-3914-4450-8c11-b17b8dabb5e1,ele99492-17ae-11eb-adc1-0242ac120002,No,No,Yes,Yes,""
49661295,2.1.22,Android,23.489369,-77.724578,de545711-3914-4450-8c11-b17b8dabb5e1,ele99492-17ae-11eb-adc1-0242ac120002,No,Yes,Yes,Yes,""
34870660,3.2.13,iOS,-67.454532,-105.107223,b328829e-17ae-11eb-adc1-0242ac120002,ele99492-17ae-11eb-adc1-0242ac120002,Yes,Yes,Yes,No,""
57123301,2.2.13,Android,-7.4278195,-161.670646,de545711-3914-4450-8c11-b17b8dabb5e1,a95dd57b-779f-49db-819d-b6960483e554,No,No,Yes,Yes,""
48927284,1.3.12,iOS,5.9788155,151.064029,e7bc5fb2-1231-11eb-adc1-0242ac120002,fcba68aa-1231-11eb-adc1-0242ac120002,No,Yes,No,No,""
85177527,4.2.10,iOS,-51.8301325,-70.399319,e7bc5fb2-1231-11eb-adc1-0242ac120002,a95dd57b-779f-49db-819d-b6960483e554,Yes,No,No,Yes,""
47743867,3.3.20,iOS,-13.990733,-178.175285,b328829e-17ae-11eb-adc1-0242ac120002,a95dd57b-779f-49db-819d-b6960483e554,Yes,No,No,No,""
46894536,4.2.31,iOS,-59.551650,115.343099,e7bc5fb2-1231-11eb-adc1-0242ac120002,fcba68aa-1231-11eb-adc1-0242ac120002,Yes,No,Yes,Yes,""
87861168,4.4.27,iOS,-9.0299885,87.753166,de545711-3914-4450-8c11-b17b8dabb5e1,fcba68aa-1231-11eb-adc1-0242ac120002,Yes,No,No,Yes,""
98750961,1.4.24,Android,-67.056611,-42.210252,b328829e-17ae-11eb-adc1-0242ac120002,ele99492-17ae-11eb-adc1-0242ac120002,Yes,Yes,Yes,No,""
34177709,2.4.1,iOS,-65.870590,112.347735,de545711-3914-4450-8c11-b17b8dabb5e1,ele99492-17ae-11eb-adc1-0242ac120002,Yes,No,Yes,Yes,""
35577566,2.3.18,Android,28.5281265,28.317597,e7bc5fb2-1231-11eb-adc1-0242ac120002,ele99492-17ae-11eb-adc1-0242ac120002,Yes,Yes,Yes,No,""
39790044,3.3.32,iOS,60.0575415,-36.785274,b328829e-17ae-11eb-adc1-0242ac120002,fcba68aa-1231-11eb-adc1-0242ac120002,No,Yes,Yes,No,""
49451715,2.3.8,iOS,77.0398825,174.942366,e7bc5fb2-1231-11eb-adc1-0242ac120002,ele99492-17ae-11eb-adc1-0242ac120002,Yes,Yes,No,Yes,""
87559592,2.2.17,Android,-72.2719865,-66.732794,e7bc5fb2-1231-11eb-adc1-0242ac120002,fcba68aa-1231-11eb-adc1-0242ac120002,No,Yes,No,No,""
20011383,2.1.30,Android,58.3581915,27.525051,b328829e-17ae-11eb-adc1-0242ac120002,ele99492-17ae-11eb-adc1-0242ac120002,Yes,No,No,Yes,""
83519078,4.2.9,iOS,78.4456805,-83.204209,b328829e-17ae-11eb-adc1-0242ac120002,a95dd57b-779f-49db-819d-b6960483e554,No,No,Yes,No,""
35953253,1.3.11,iOS,29.422319,106.217372,de545711-3914-4450-8c11-b17b8dabb5e1,fcba68aa-1231-11eb-adc1-0242ac120002,No,No,Yes,Yes,""
79134831,1.3.25,iOS,-43.676692,-13.693995,de545711-3914-4450-8c11-b17b8dabb5e1,ele99492-17ae-11eb-adc1-0242ac120002,Yes,Yes,Yes,No,""
74932660,2.4.11,Android,-16.258038,-67.903992,b328829e-17ae-11eb-adc1-0242ac120002,ele99492-17ae-11eb-adc1-0242ac120002,No,No,No,Yes,""
48948016,1.2.37,Android,28.116246,80.131224,b328829e-17ae-11eb-adc1-0242ac120002,a95dd57b-779f-49db-819d-b6960483e554,Yes,No,No,Yes,""
70449656,2.3.23,Android,-38.571418,102.352612,de545711-3914-4450-8c11-b17b8dabb5e1,ele99492-17ae-11eb-adc1-0242ac120002,No,Yes,No,Yes,""
41398860,4.1.10,Android,-58.4709445,-56.966952,e7bc5fb2-1231-11eb-adc1-0242ac120002,a95dd57b-779f-49db-819d-b6960483e554,No,No,Yes,Yes,""
35844721,2.3.22,Android,0.9971151,-56.869833,de545711-3914-4450-8c11-b17b8dabb5e1,fcba68aa-1231-11eb-adc1-0242ac120002,Yes,Yes,No,Yes,""
49195530,4.1.38,Android,-65.173138,-45.282215,de545711-3914-4450-8c11-b17b8dabb5e1,fcba68aa-1231-11eb-adc1-0242ac120002,No,No,No,Yes,""
92467259,1.1.28,iOS,-13.9285225,166.976040,e7bc5fb2-1231-11eb-adc1-0242ac120002,ele99492-17ae-11eb-adc1-0242ac120002,Yes,No,No,No,""
39856207,4.4.21,iOS,-45.9151095,-93.708618,de545711-3914-4450-8c11-b17b8dabb5e1,ele99492-17ae-11eb-adc1-0242ac120002,Yes,No,No,Yes,""
47664043,3.3.14,Android,-13.842446,117.938952,e7bc5fb2-1231-11eb-adc1-0242ac120002,a95dd57b-779f-49db-819d-b6960483e554,Yes,No,No,Yes,""
38729595,1.1.32,iOS,-86.7711605,-133.680955,e7bc5fb2-1231-11eb-adc1-0242ac120002,a95dd57b-779f-49db-819d-b6960483e554,Yes,Yes,Yes,No,""
52083661,1.3.6,Android,-20.051056,-113.408925,b328829e-17ae-11eb-adc1-0242ac120002,a95dd57b-779f-49db-819d-b6960483e554,No,No,Yes,No,""
21248817,2.2.14,iOS,25.7465345,-115.096705,e7bc5fb2-1231-11eb-adc1-0242ac120002,ele99492-17ae-11eb-adc1-0242ac120002,Yes,No,Yes,No,""
32887975,2.3.23,Android,-14.8909615,-168.408479,b328829e-17ae-11eb-adc1-0242ac120002,a95dd57b-779f-49db-819d-b6960483e554,Yes,No,Yes,No,""
79298369,4.1.11,iOS,-3.7870705,128.420903,b328829e-17ae-11eb-adc1-0242ac120002,ele99492-17ae-11eb-adc1-0242ac120002,No,Yes,No,Yes,""
74700232,3.1.6,iOS,38.679099,-26.254784,b328829e-17ae-11eb-adc1-0242ac120002,a95dd57b-779f-49db-819d-b6960483e554,No,No,No,No,""
52434368,4.4.13,Android,86.2803345,-4.684262,e7bc5fb2-1231-11eb-adc1-0242ac120002,a95dd57b-779f-49db-819d-b6960483e554,No,Yes,No,No,""
```

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

4. create another python file (spark_local_flatten.py) using vi command. This file will clean the loaded Kafka data to a more structured format

   vi spark_local_flatten.py

5. Explaining the code in spark_local_flatten.py

```
# 5.1 Importing the modules

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

# 5.2 Initializing Spark Session

spark = SparkSession.builder \
    .master("local") \
    .appName("CapstoneProject") \
    .getOrCreate()

# 5.3 Reading json data into dataframe

clickstream_df = spark.read.json('clickstream_data/part-00000-9ff66f8b-d201-4aa3-8308-d053d4171d08-c000.json')


# 5.4 Extrating columns from jason value in dataframe and create new dataframe with new cloumns

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

# 5.5 Printing the Dataframe schema

print(clickstream_df.schema)
```

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

5.6 Printing 10 records from the dataframe

clickstream_df.show(10)

5.7 Saving the dataframe to csv file with headers in local file directory

clickstream_df.coalesce(1).write.format('com.databricks.spark.csv').mode('overwrite').save('clickstream_data_flatten/', header = 'true')

6. Run Spark Submit command

   spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 spark_local_flatten.py

7. Make a directory using mkdir command

   hadoop fs -mkdir clickstream_data_flatten

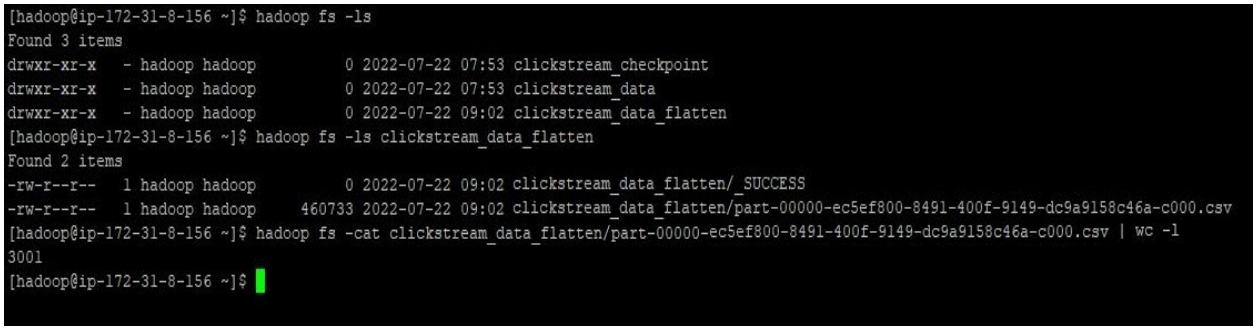
8. Loading the data from local file system to hadoop file system

   hadoop fs- put ~/clickstream_data_flatten clickstream_data_flatten

9. Checking the data file in hadoop

   hadoop fs -ls clickstream_data_flatten

   hadoop fs -cat clickstream_data_flatten/part-00000-ec5ef800-8491-400f-9149-dc9a9158c46a-c000.csv | wc –l



© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

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

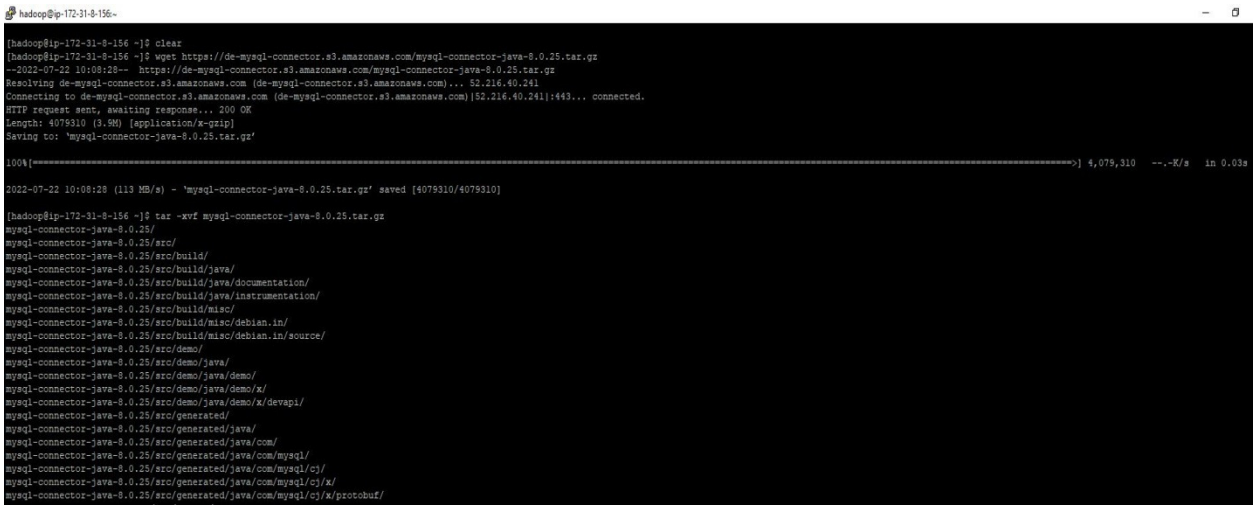
1. First we need to setup MySQL Connector on AWS EMR

   a. Run the following command to install the MySQL connector jar file:

   wget https://de-mysql-connector.s3.amazonaws.com/mysql-connector-java-8.0.25.tar.gz

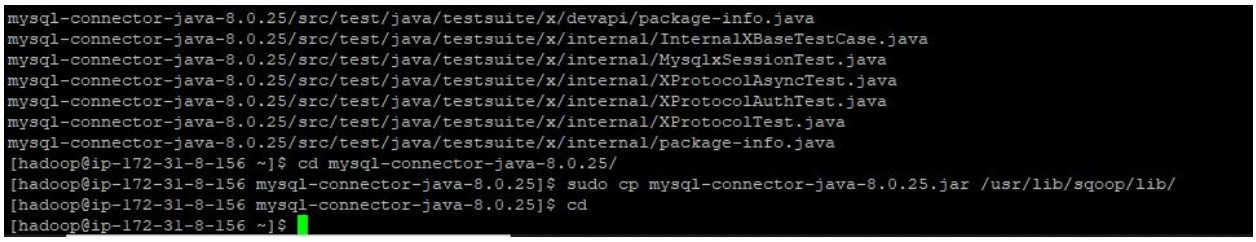
   b. Run the following step to extract the MySQL connector tar file

   tar -xvf mysql-connector-java-8.0.25.tar.gz



   c. go to the MySQL Connector directory and then copy it to the Sqoop library to complete the installation.

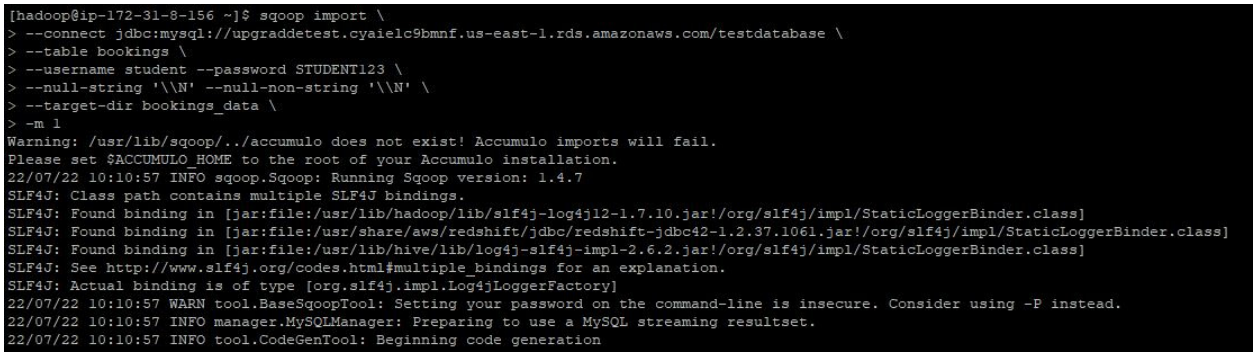
   cd mysql-connector-java-8.0.25/
   sudo cp mysql-connector-java-8.0.25.jar /usr/lib/sqoop/lib/

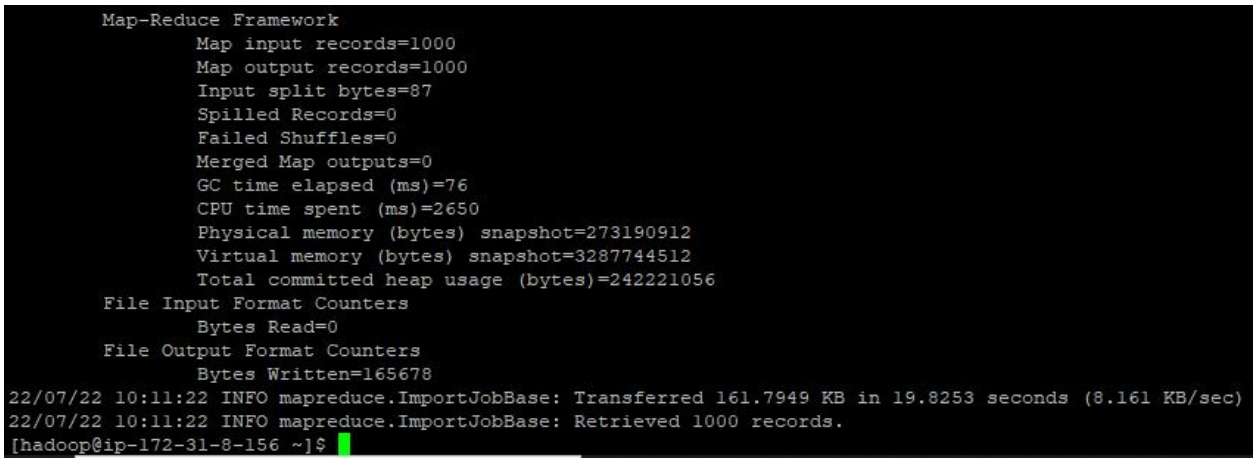


© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

2. Now we run the Sqoop import command to import data from AWS RDS to Hadoop

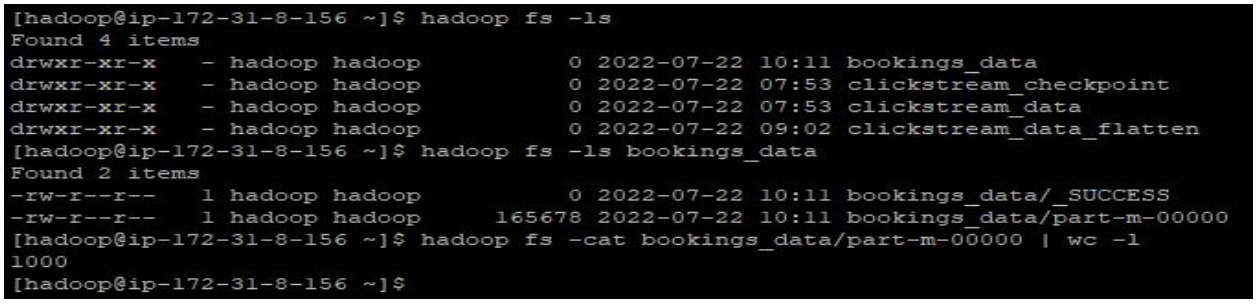
```
sqoop import \
--connect jdbc:mysql://upgraddetest.cyaielc9bmnf.us-east-1.rds.amazonaws.com/testdatabase \
--table bookings \
--username student --password STUDENT123 \
--null-string '\\N' --null-non-string '\\N' \
--target-dir bookings_data \
-m 1
```





3. Use this command to check the imported data

```
hadoop fs -ls bookings_data
hadoop fs -cat bookings_data/part-m-00000 | wc –l
```



© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

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

1. create a python file (datewise_bookings_aggregates_spark.py) using vi command. This file will contain the code which will aggregate the booking data for finding datewise total bookings

   datewise_bookings_aggregates_spark.py

2. Explaining the code in datewise_bookings_aggregates_spark.py

```
# 2.1 Importing the modules

from pyspark.sql.types import *
from pyspark.sql.functions import *
from pyspark.sql import functions as F
from pyspark.sql import SparkSession
from pyspark.sql.functions import from_json
from pyspark.sql.types import TimestampType, IntegerType, FloatType, ArrayType,LongType
from pyspark.sql import functions as func

# 2.2 Initializing Spark Session

spark = SparkSession  \
    .builder  \
    .appName("StructuredSocketRead")  \
    .getOrCreate()
spark.sparkContext.setLogLevel('ERROR')

# 2.3 Defining the schema

schema1 = StructType([
        StructField("booking_id", StringType()),
        StructField("customer_id",  LongType()),
        StructField("driver_id",  LongType()),
        StructField("customer_app_version", StringType()),
        StructField("customer_phone_os_version", StringType()),
        StructField("pickup_lat", DoubleType()),
        StructField("pickup_lon", DoubleType()),
```

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

```
StructField("drop_lat", DoubleType()),
StructField("drop_lon", DoubleType()),
StructField("pickup_timestamp", TimestampType()),
StructField("drop_timestamp", TimestampType()),
StructField("trip_fare", IntegerType()),
StructField("tip_amount", IntegerType()),
StructField("currency_code", StringType()),
StructField("cab_color", StringType()),
StructField("cab_registration_no", StringType()),
StructField("customer_rating_by_driver", IntegerType()),
StructField("rating_by_customer", IntegerType()),
StructField("passenger_count", IntegerType())
])

# 2.4 Reading the file "part-m-00000" in bookings_data folder in hadoop

df=spark.read.csv("bookings_data/part-m-00000", schema=schema1)

# 2.5 Creating data from column "pickup_date" and "pickup_timestamp"

df = df.withColumn("pickup_date", func.to_date(func.col("pickup_timestamp")))

# 2.6 Group the data by "pickup_date"

date = df.groupBy('pickup_date').count()

# 2.7 Saving the datewise total booking data in .csv format

date.coalesce(1).write.format('csv').save("datewise_aggregated_data/")

# 2.8 Saving the booking data in .csv format

df.coalesce(1).write.format('com.databricks.spark.csv').mode('overwrite').save('booking_data_csv/',
header = 'true')
```

3. Run Spark Submit command

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

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

```
[hadoop@ip-172-31-8-156 ~]$ spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 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-185e9042-7bd7-4b78-b33b-e7283db28905;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 386ms :: artifacts dl 10ms
        :: modules in use:
        org.apache.kafka#kafka-clients;2.0.0 from central in [default]
        org.apache.spark#spark-sql-kafka-0-10_2.11;2.4.5 from central in [default]
        org.lz4#lz4-java;1.4.0 from central in [default]
        org.slf4j#slf4j-api;1.7.16 from central in [default]
        org.spark-project.spark#unused;1.0.0 from central in [default]
        org.xerial.snappy#snappy-java;1.1.7.3 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   6   |   0   |   0   |   0   ||   6   |   0   |
        ---------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-parent-185e9042-7bd7-4b78-b33b-e7283db28905
        confs: [default]
        0 artifacts copied, 6 already retrieved (0kB/10ms)
22/07/22 10:23:09 INFO SparkContext: Running Spark version 2.4.5-amzn-0
```

```
        tracking URL: http://ip-172-31-8-156.ec2.internal:20888/proxy/application_1658475782346_0002/
        user: hadoop
22/07/22 10:23:25 INFO YarnClientSchedulerBackend: Application application_1658475782346_0002 has started running.
22/07/22 10:23:25 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 39735.
22/07/22 10:23:25 INFO NettyBlockTransferService: Server created on ip-172-31-8-156.ec2.internal:39735
22/07/22 10:23:25 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy
22/07/22 10:23:25 INFO YarnSchedulerBackend$YarnSchedulerEndpoint: ApplicationMaster registered as NettyRpcEndpointRef(spark-client://YarnAM)
22/07/22 10:23:25 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, ip-172-31-8-156.ec2.internal, 39735, None)
22/07/22 10:23:25 INFO BlockManagerMasterEndpoint: Registering block manager ip-172-31-8-156.ec2.internal:39735 with 1028.8 MB RAM, BlockManagerId(driver, ip-172-31-8-156.ec2.internal, 39735, None)
22/07/22 10:23:25 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, ip-172-31-8-156.ec2.internal, 39735, None)
22/07/22 10:23:25 INFO BlockManager: external shuffle service port = 7337
22/07/22 10:23:25 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, ip-172-31-8-156.ec2.internal, 39735, None)
22/07/22 10:23:25 INFO JettyUtils: Adding filter org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter to /metrics/json.
22/07/22 10:23:26 INFO EventLoggingListener: Logging events to hdfs:/var/log/spark/apps/application_1658475782346_0002
22/07/22 10:23:26 INFO Utils: Using initial executors = 50, max of spark.dynamicAllocation.initialExecutors, spark.dynamicAllocation.minExecutors and spark.executor.instances
22/07/22 10:23:26 INFO YarnClientSchedulerBackend: SchedulerBackend is ready for scheduling beginning after reached minRegisteredResourcesRatio: 0.0
22/07/22 10:23:27 INFO SharedState: loading hive config file: file:/etc/spark/conf.dist/hive-site.xml
22/07/22 10:23:27 INFO SharedState: Setting hive.metastore.warehouse.dir ('null') to the value of spark.sql.warehouse.dir ('hdfs:///user/spark/warehouse').
22/07/22 10:23:27 INFO SharedState: Warehouse path is 'hdfs:///user/spark/warehouse'.
22/07/22 10:23:27 INFO JettyUtils: Adding filter org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter to /SQL.
22/07/22 10:23:27 INFO JettyUtils: Adding filter org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter to /SQL/json.
22/07/22 10:23:27 INFO JettyUtils: Adding filter org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter to /SQL/execution.
22/07/22 10:23:27 INFO JettyUtils: Adding filter org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter to /SQL/execution/json.
22/07/22 10:23:27 INFO JettyUtils: Adding filter org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter to /static/sql.
22/07/22 10:23:28 INFO StateStoreCoordinatorRef: Registered StateStoreCoordinator endpoint
[hadoop@ip-172-31-8-156 ~]$
```

4. Make a directory using mkdir command

   hadoop fs -mkdir datewise_aggregated_data

5. Loading the data from local file system to hadoop file system

   hadoop fs- put ~/ datewise_aggregated_data datewise_aggregated_data

6. Checking the data file in hadoop

   hadoop fs -ls date_aggregated_data

   hadoop fs -cat datewise_aggregated_data /part-00000-a085f9bc-ecd9-4ee7-b21b-3329d4fabfd3-c000.csv | wc –l

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

```
[hadoop@ip-172-31-8-156 ~]$ hadoop fs -ls
Found 7 items
drwxr-xr-x   - hadoop hadoop          0 2022-07-22 10:23 .sparkStaging
drwxr-xr-x   - hadoop hadoop          0 2022-07-22 10:23 booking_data_csv
drwxr-xr-x   - hadoop hadoop          0 2022-07-22 10:11 bookings_data
drwxr-xr-x   - hadoop hadoop          0 2022-07-22 07:53 clickstream_checkpoint
drwxr-xr-x   - hadoop hadoop          0 2022-07-22 07:53 clickstream_data
drwxr-xr-x   - hadoop hadoop          0 2022-07-22 09:02 clickstream_data_flatten
drwxr-xr-x   - hadoop hadoop          0 2022-07-22 10:23 datewise_aggregated_data
[hadoop@ip-172-31-8-156 ~]$ hadoop fs -ls datewise_aggregated_data
Found 2 items
-rw-r--r--   1 hadoop hadoop          0 2022-07-22 10:23 datewise_aggregated_data/_SUCCESS
-rw-r--r--   1 hadoop hadoop       3758 2022-07-22 10:23 datewise_aggregated_data/part-00000-a085f9bc-ecd9-4ee7-b21b-3329d4fabfd3-c000.csv
[hadoop@ip-172-31-8-156 ~]$ hadoop fs -cat datewise_aggregated_data/part-00000-a085f9bc-ecd9-4ee7-b21b-3329d4fabfd3-c000.csv | wc -l
289
[hadoop@ip-172-31-8-156 ~]$
```

```
[hadoop@ip-172-31-8-156 ~]$ hadoop fs -ls booking_data_csv
Found 2 items
-rw-r--r--   1 hadoop hadoop          0 2022-07-22 10:23 booking_data_csv/_SUCCESS
-rw-r--r--   1 hadoop hadoop     182968 2022-07-22 10:23 booking_data_csv/part-00000-ee83ab78-57ec-4982-9d50-e2ea2e04d9bf-c000.csv
[hadoop@ip-172-31-8-156 ~]$ hadoop fs -cat booking_data_csv/part-00000-ee83ab78-57ec-4982-9d50-e2ea2e04d9bf-c000.csv | wc -l
1001
[hadoop@ip-172-31-8-156 ~]$
```

Task 4: Create Hive Managed Tables

1. First create a database

create database if not exists cab_booking_data ;
use cab_booking_data ;

```
[hadoop@ip-172-31-8-156 ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: false
hive> show databases;
OK
default
Time taken: 0.596 seconds, Fetched: 1 row(s)
hive>
```

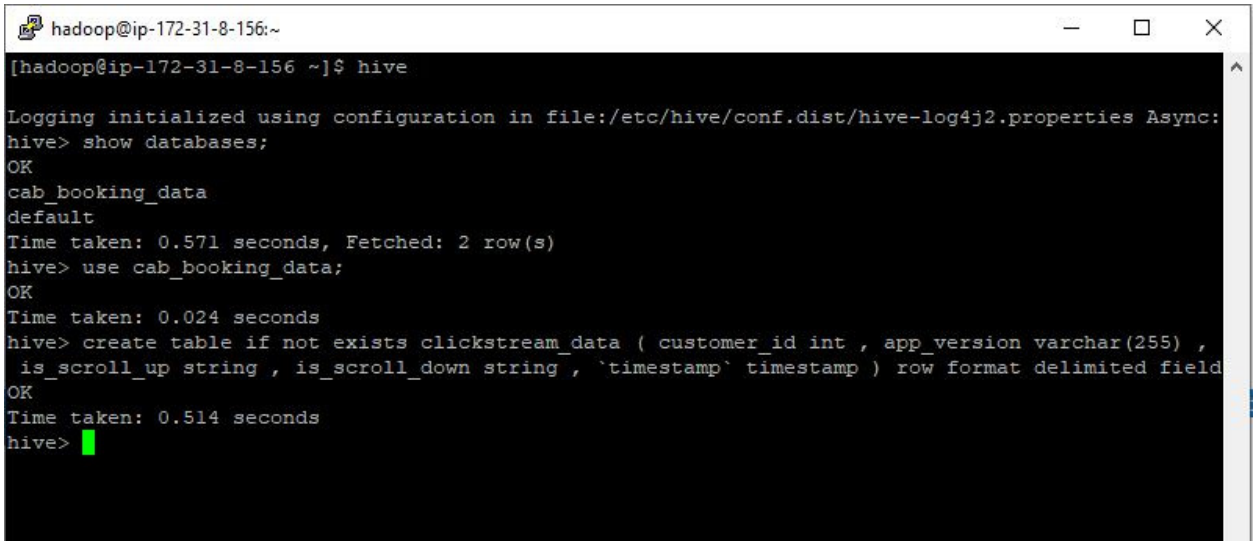
```
[hadoop@ip-172-31-8-156 ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: false
hive> show databases;
OK
default
Time taken: 0.596 seconds, Fetched: 1 row(s)
hive> create database if not exists cab_booking_data with dbproperties('creator' = 'Prashant', 'date' = '2022-07-22');
OK
Time taken: 0.119 seconds
hive> set hive.cli.print.header=true ;
hive> describe database extended cab_booking_data ;
OK
db_name comment location        owner_name      owner_type      parameters
cab_booking_data                hdfs://ip-172-31-8-156.ec2.internal:8020/user/hive/warehouse/cab_booking_data.db        hadoop  USER    {date=2022-07-22, creator=Prashant}
Time taken: 0.041 seconds, Fetched: 1 row(s)
hive>
```

```
hive> show databases;
OK
cab_booking_data
default
Time taken: 0.875 seconds, Fetched: 2 row(s)
hive> use cab_booking_data ;
OK
Time taken: 0.025 seconds
hive>
```

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

2. Creating a Hive-managed table for clickstream data

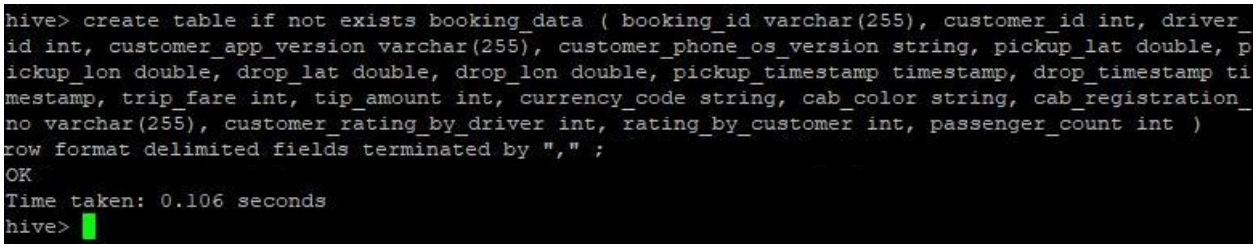
```
create table if not exists clickstream_data (
        customer_id int ,
        app_version varchar(255),
        os_version string,
        lat varchar(255),
        lon varchar(255),
        page_id varchar(255),
        button_id varchar(255),
        is_button_click string,
        is_page_view string,
        is_scroll_up string,
        is_scroll_down string,
        `timestamp` timestamp
)
row format delimited fields terminated by "," ;
```



© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

3. Creating a Hive-managed table for bookings data

```
create table if not exists booking_data (
        booking_id varchar(255),
        customer_id int,
        driver_id int,
        customer_app_version varchar(255),
        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 varchar(255),
        customer_rating_by_driver int,
        rating_by_customer int,
        passenger_count int
)
row format delimited fields terminated by "," ;
```



© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

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

```
create table if not exists datewise_aggregated_data (
        `date` string,
        count int
)
row format delimited fields terminated by "," ;
```

```
hive> create table if not exists datewise_aggregated_data ( `date` string, count int ) row format
 delimited fields terminated by "," ;
OK
Time taken: 0.096 seconds
hive>
```

5. Command to load the data into Hive tables

a. load data inpath 'clickstream_data_flatten/ part-00000-ec5ef800-8491-400f-9149-dc9a9158c46a-c000.csv ' into table clickstream_data ;
b. load data inpath 'booking_data_csv/ part-00000-ee83ab78-57ec-4982-9d50-e2ea2e04d9bf-c000.csv' into table booking_data ;
c. load data inpath 'date_aggregated_data/ part-00000-a085f9bc-ecd9-4ee7-b21b-3329d4fabfd3-c000.csv' into table datewise_aggregated_data ;

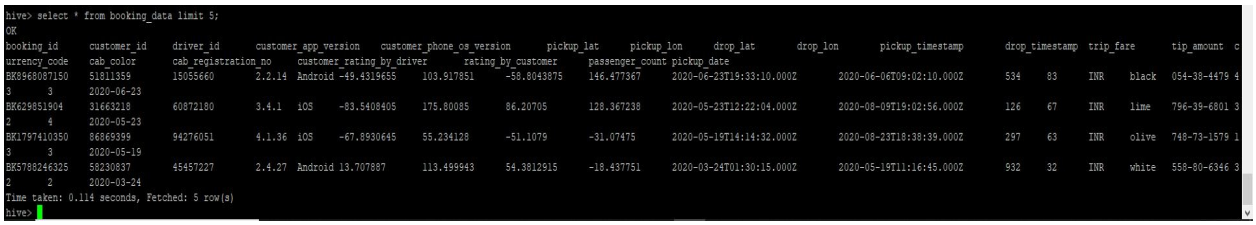
```
hive> load data inpath 'clickstream_data_flatten/part-00000-ec5ef800-8491-400f-9149-dc9a9158c46a-
c000.csv' into table clickstream_data ;
Loading data to table cab_booking_data.clickstream_data
OK
Time taken: 0.995 seconds
hive>
```

```
hive> load data inpath 'booking_data_csv/part-00000-ee83ab78-57ec-4982-9d50-e2ea2e04d9bf-c000.csv
' into table booking_data ;
Loading data to table cab_booking_data.booking_data
OK
Time taken: 0.445 seconds
hive>
```

```
hive> load data inpath 'datewise_aggregated_data/part-00000-a085f9bc-ecd9-4ee7-b21b-3329d4fabfd3-
c000.csv' into table datewise_aggregated_data ;
Loading data to table cab_booking_data.datewise_aggregated_data
OK
Time taken: 0.962 seconds
hive>
```

```
hive> select * from clickstream_data limit 5;
OK
customer_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       de545711-3914-4450-8c11-b17b8dabb5e1    fcba68aa-1231-11eb-adc1-0242ac120002    No      Yes     No      Yes     "2020-09-14 09:59:07"
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 16:30:21"
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 00:52:13"
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 10:42:50"
Time taken: 0.207 seconds, Fetched: 5 row(s)
hive>
```

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved



 # Task 5: Calculate the total number of different drivers for each customer.

<Hive Query for Task 5>

select customer_id , count(distinct driver_id) from booking_data group by customer_id order by customer_id asc;

< Query Explanation>

In this task, we selected 2 columns, customer_id and driver_id from booking_data table and grouped this by customer_id. We used COUNT() function for driver_id column to count the number of rows returned with DISTINCT to select unique rows from the table booking_data. In this way we can get the details required in task 5 i.e. to calculate the the total number of different drivers for each customer.

<Screenshot after executing Query and results obtained after executing the queries >

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

```
hive> select customer_id , count(distinct driver_id) from booking_data group by customer_id order by customer_id asc;
Query ID = hadoop_20220802200451_c8733194-f4f6-44a3-b583-a4558bde97bc
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1659451488083_0009)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      2          2        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 6.09 s
--------------------------------------------------------------------------------
OK
10022393        1
10058402        1
10339567        1
10435129        1
10555335        1
10592274        1
10614890        1
10678994        1
11264797        1
11353346        1
11418437        1
11438890        1
11454977        1
11479815        1
11518953        1
11580321        1
11596512        1
11608791        1
11655671        1
11757536        1
11764909        1
11860278        1
11981042        1
12106105        1
12142182        1
12312603        1
12334699        1
12367832        1
12856708        1
12885363        1
12913608        1
12914577        1
12966909        1
13015449        1
13229062        1
13262795        1
13356177        1
13387493        1
13389366        1
13442644        1
13500355        1
13590084        1
13791801        1
```

```
98071220        1
98253165        1
98294088        1
98377631        1
98482988        1
98507803        1
98571434        1
98660410        1
98669490        1
98673467        1
98816412        1
98827653        1
98859613        1
99267721        1
99386188        1
99451585        1
99475693        1
99507862        1
99520051        1
99542814        1
99633146        1
99749252        1
99811420        1
99827820        1
99943608        1
99947969        1
Time taken: 10.148 seconds, Fetched: 1000 row(s)
hive>
```

 # Task 6: Calculate the total rides taken by each customer.

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

<Hive Query for Task 6>

```
select customer_id ,count(distinct booking_id) from booking_data group by customer_id order by customer_id asc;
```

< Query Explanation>

In this task, we selected 2 columns, customer_id and booking _id from booking_data table and grouped this by customer_id and ordered by customer_id in ascending order. We used COUNT() function for booking _id column to count the number of rows returned with DISTINCT to select unique rows from the table booking_data.

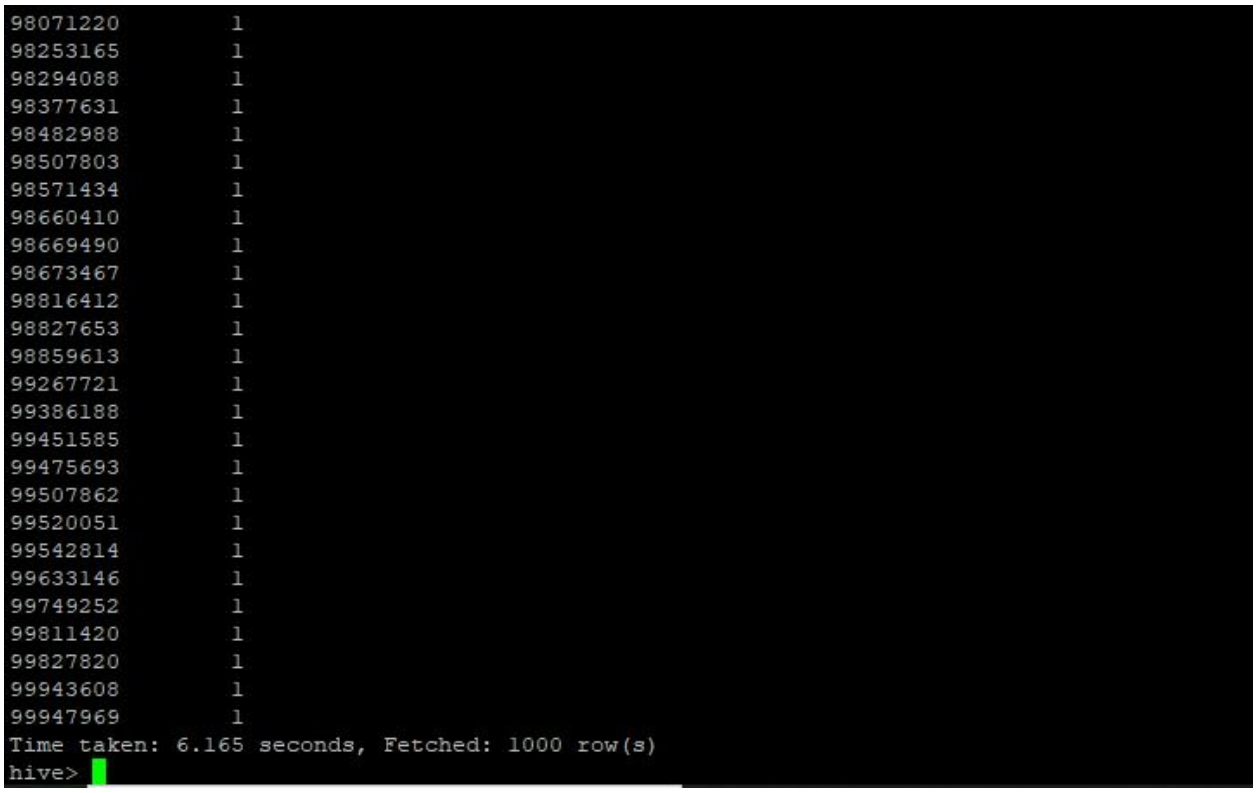
In this way we can get the details required in task 6 i.e. to calculate the total rides taken by each customer.

<Screenshot after executing Query and results obtained after executing the queries >

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

```
hive> select customer_id ,count(distinct booking_id) from booking_data group by customer_id order by customer_id asc;
Query ID = hadoop_20220802201429_50b45bad-6e74-4fe2-b280-e073d5e5b3bd
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1659451488083_0010)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ......   container     SUCCEEDED      2          2        0        0       0       0
Reducer 3 ......   container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 5.48 s
--------------------------------------------------------------------------------
OK
10022393        1
10058402        1
10339567        1
10435129        1
10555335        1
10592274        1
10614890        1
10678994        1
11264797        1
11353346        1
11418437        1
11438890        1
11454977        1
11479815        1
11518953        1
11580321        1
11596512        1
11608791        1
11655671        1
11757536        1
11764909        1
11860278        1
11981042        1
12106105        1
12142182        1
12312603        1
12334699        1
12367832        1
12856708        1
12885363        1
12913608        1
12914577        1
12966909        1
13015449        1
13229062        1
13262795        1
13356177        1
13387493        1
13389366        1
```

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved



 # Task 7: Find the total visits made by each customer on the booking page and the total 'Book Now' button presses. This can show the conversion ratio.
The booking page id is 'e7bc5fb2-1231-11eb-adc1-0242ac120002'.
The Book Now button id is 'fcba68aa-1231-11eb-adc1-0242ac120002'. You also need to calculate the conversion ratio as part of this task. Conversion ratio can be calculated as Total 'Book Now' Button Press/Total Visits made by customer on the booking page.

<Hive Query for Task 7>

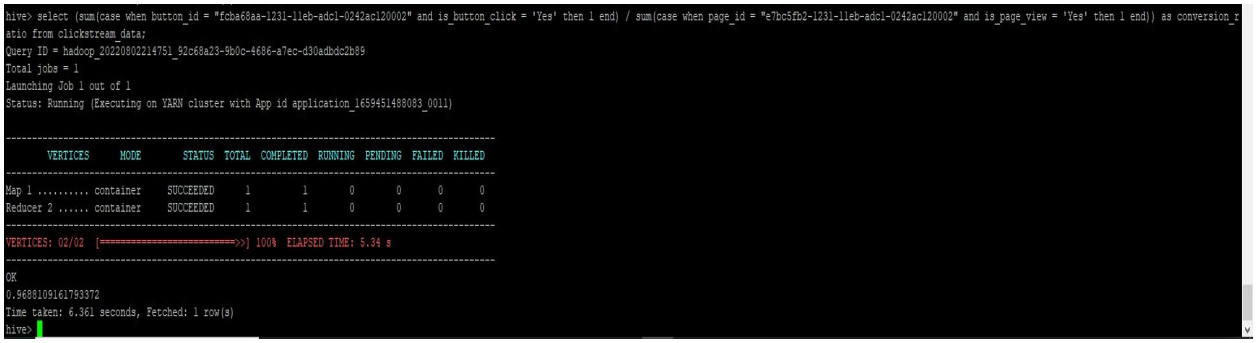
```
select (sum(case when button_id = "fcba68aa-1231-11eb-adc1-0242ac120002" and is_button_click = 'Yes' then 1 end) / sum(case when page_id = "e7bc5fb2-1231-11eb-adc1-0242ac120002" and is_page_view = 'Yes' then 1 end)) as conversion_ratio from clickstream_data;
```

< Query Explanation>

In this task, first we use clickstream_data table, and we used SUM() function to sum up values where button_id = "fcba68aa-1231-11eb-adc1-0242ac120002" and is_button_click = 'Yes'. Once again, we used SUM() function to sum up values where page_id = "e7bc5fb2-1231-11eb-adc1-0242ac120002" and is_page_view = 'Yes' and fiinally, we divide the first sum by the second sum to get conversion ratio. In this way we can get the details required in task 7.

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

<Screenshot after executing Query and results obtained after executing the queries >



hive> select (sum(case when button_id = "fcba68aa-1231-11eb-adc1-0242ac120002" and is_button_cl
atio from clickstream_data;
Query ID = hadoop_20220802214751_92c68a23-9b0c-4686-a7ec-d30adbdc2b89
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1659451488083_0011)

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: 5.34 s
OK
0.9688109161793372
Time taken: 6.361 seconds, Fetched: 1 row(s)
hive>

 # Task 8: Calculate the count of all trips done on black cabs.

<Hive Query for Task 8>

select cab_color ,count(distinct driver_id) from booking_data where cab_color in ('black') group by cab_color ;

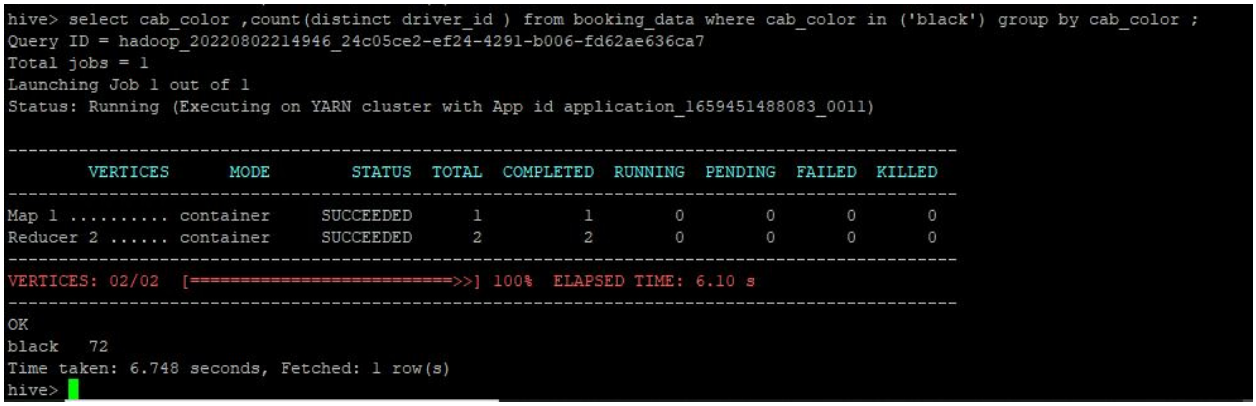
< Query Explanation>

In this task, we selected 2 columns, cab_color and driver_id from booking_data table and we used WHERE clause to filter the data where cab_color is ‘black’ and then we grouped this by cab_color. We used COUNT() function for driver_id column to count the number of rows returned with DISTINCT to select unique rows from the table booking_data.

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

In this way we can get the details required in task 8 i.e. to calculate the count of all trips done on black cabs.

<Screenshot after executing Query and results obtained after executing the queries >



 # Task 9: Calculate the total amount of tips given date wise to all drivers by customers.

<Hive Query for Task 9>

select pickup_date, sum(tip_amount) from booking_data group by pickup_date order by pickup_date asc;

< Query Explanation>

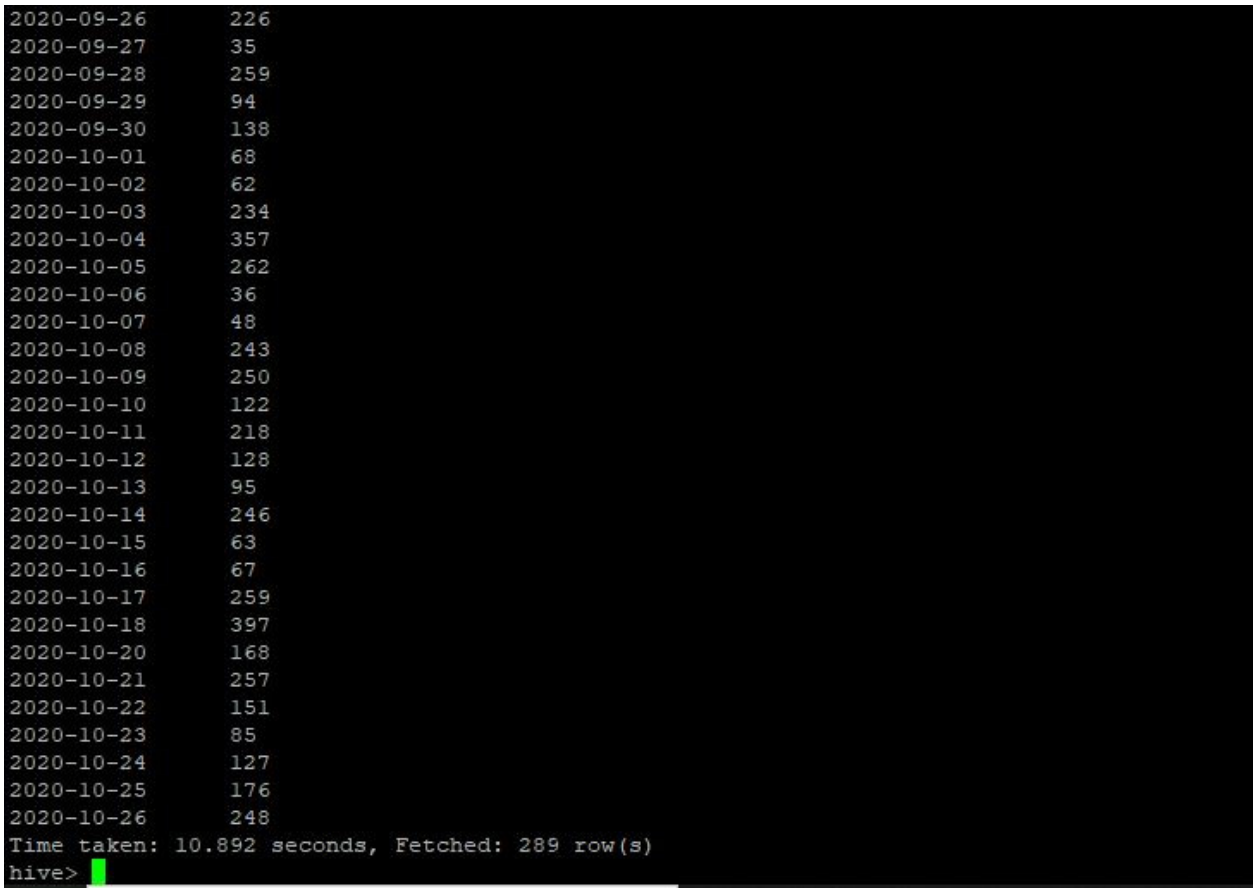
In this task, we selected 2 columns, pickup_date and tip_amount from booking_data table and then we order this by pickup_date in ascending order. We used SUM() function for tip_amount column to return the total amount of tips given. In this way we can get the details required in task 9 i.e. to calculate the total amount of tips given date wise to all drivers by customers.

<Screenshot after executing Query and results obtained after executing the queries >

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

```
hive> select pickup_date, sum(tip_amount) from booking_data group by pickup_date order by pickup_date asc;
Query ID = hadoop_20220803083826_72cb3603-c8db-4e05-8dcb-b15833784a13
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1659509066141_0011)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      2          2        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 10.13 s
--------------------------------------------------------------------------------
OK
2020-01-01      59
2020-01-02      95
2020-01-03      11
2020-01-04      123
2020-01-05      134
2020-01-06      189
2020-01-07      148
2020-01-08      111
2020-01-09      48
2020-01-10      77
2020-01-11      81
2020-01-12      109
2020-01-14      142
2020-01-15      338
2020-01-16      155
2020-01-17      296
2020-01-18      240
2020-01-20      210
2020-01-21      5
2020-01-23      148
2020-01-24      472
2020-01-25      98
2020-01-26      209
2020-01-27      231
2020-01-28      567
2020-01-29      123
2020-01-30      112
2020-01-31      256
2020-02-01      317
2020-02-02      338
2020-02-03      191
2020-02-04      258
2020-02-05      212
2020-02-06      154
2020-02-07      91
2020-02-08      270
2020-02-09      266
2020-02-10      115
2020-02-11      3
2020-02-12      252
2020-02-13      147
2020-02-15      108
2020-02-16      133
2020-02-17      519
2020-02-18      120
2020-02-19      33
2020-02-20      105
2020-02-21      34
2020-02-22      233
```

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved



 # Task 10: Calculate the total count of all the bookings with ratings lower than 2 as given by customers in a particular month.

<Hive Query for Task 10>

select date_format(pickup_timestamp,'yyyy-mm') ,count(rating_by_customer) from booking_data where rating_by_customer < 2 group by date_format(pickup_timestamp,'yyyy-mm') ;

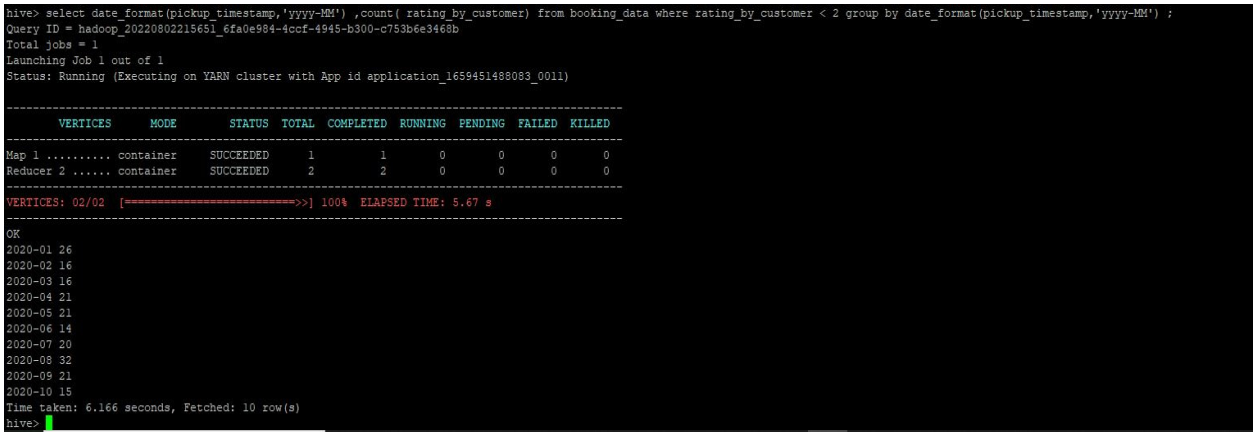
< Query Explanation>

In this task, we selected 2 columns, pickup_timestamp and rating_by_customer from booking_data table and we used where clause to filter the data where rating_by_customer < 2 and then we grouped this by date_format(pickup_timestamp,'yyyy-mm'). We used DATE_FORMAT() function to extract the pickup date for pickup_timestamp column in 'yyyy-mm' format and COUNT() function for rating_by_customer column to count the number of rows returned.

In this way we can get the details required in task 10 i.e. to calculate the total count of all the bookings with ratings lower than 2 as given by customers in a particular month.

© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved

<Screenshot after executing Query and results obtained after executing the queries >



 # Task 11: Calculate the count of total iOS users.

<Hive Query for Task 11>

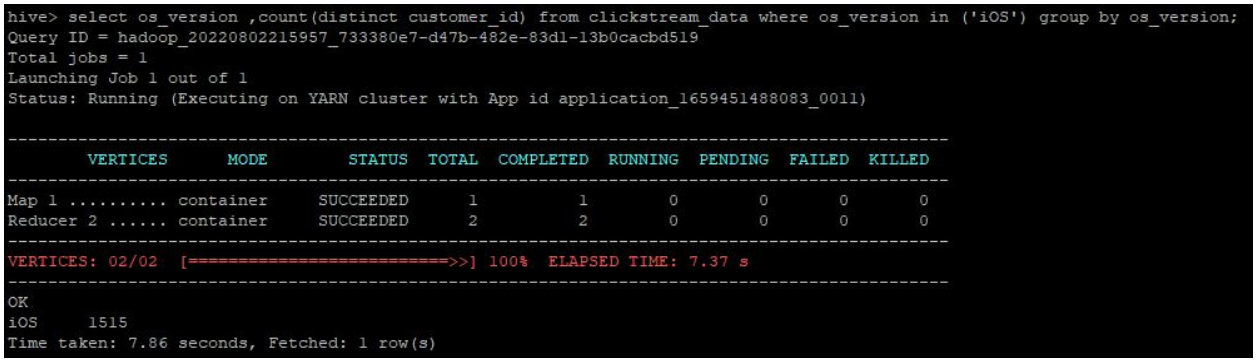
select os_version ,count(distinct customer_id) from clickstream_data where os_version in ('iOS') group by os_version;

< Query Explanation>

In this task, we selected 2 columns, os_version and customer_id from clickstream_data table and we used WHERE clause to filter the data where os_version is ‘iOS’ and then we grouped this by os_version. We used COUNT() function for customer_id column to count the number of rows returned with DISTINCT to select unique rows from the table clickstream_data.

In this way we can get the details required in task 11 i.e. to calculate the count of total iOS users.

<Screenshot after executing Query and results obtained after executing the queries >



© Copyright 2020. upGrad Education Pvt. Ltd. All rights reserved