Hive Case Study

By

Sayujya Vartak & Karthika Devi R

Problem Statement

With online sales gaining popularity, tech companies are exploring ways to improve their sales by analysing customer behaviour and gaining insights about product trends. Furthermore, the websites make it easier for customers to find the products they require without much scavenging. Needless to say, the role of big data analysts is among the most sought-after job profiles of this decade. Therefore, as part of this assignment, we will be challenging you, as a big data analyst, to extract data and gather insights from a real-life data set of an e-commerce company.

One of the most popular use cases of Big Data is in eCommerce companies such as Amazon or Flipkart. So before we get into the details of the dataset, let us understand how eCommerce companies make use of these concepts to give customers product recommendations. This is done by tracking your clicks on their website and searching for patterns within them. This kind of data is called a clickstream data. Let us understand how it works in detail.

The clickstream data contains all the logs as to how you navigated through the website. It also contains other details such as time spent on every page, etc. From this, they make use of data ingesting frameworks such as Apache Kafka or AWS Kinesis in order to store it in frameworks such as Hadoop. From there, machine learning engineers or business analysts use this data to derive valuable insights.

Objective

- We will be working with a public clickstream dataset of a cosmetics store. Using this dataset, our job is to extract valuable insights from the given data.

- Link for Datasets -
    - https://e-commerce-events-ml.s3.amazonaws.com/2019-Oct.csv
    - https://e-commerce-events-ml.s3.amazonaws.com/2019-Nov.csv.

We are required to provide answers to the questions given below.

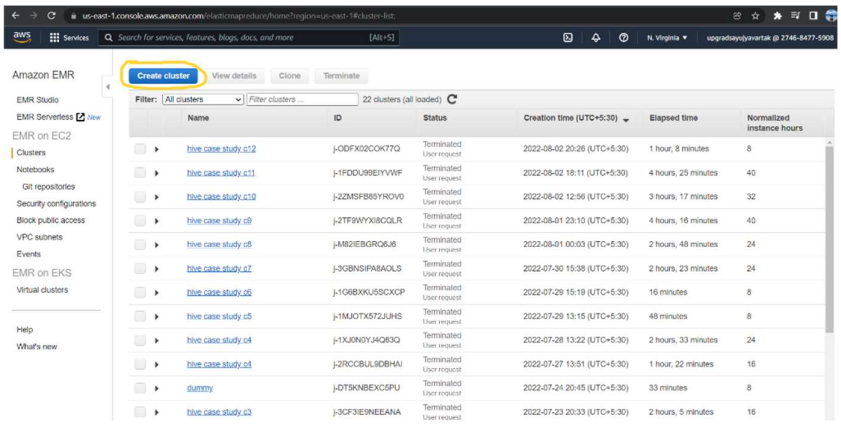
- Find the total revenue generated due to purchases made in October.
- Write a query to yield the total sum of purchases per month in a single output.
- Write a query to find the change in revenue generated due to purchases from October to November.
- Find distinct categories of products. Categories with null category code can be ignored.
- Find the total number of products available under each category.
- Which brand had the maximum sales in October and November combined?
- Which brands increased their sales from October to November?
- Your company wants to reward the top 10 users of its website with a Golden Customer plan. Write a query to generate a list of top 10 users who spend the most

General Approach

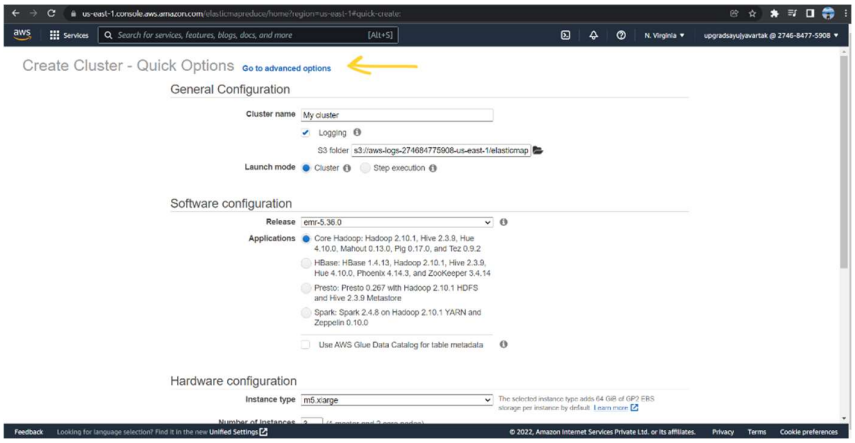
- Copying the data set into the HDFS:
    - Launch an EMR cluster that utilizes the Hive services, and
    - Move the data from the S3 bucket into the HDFS
- Creating the database and launching Hive queries on your EMR cluster:
    - Create the structure of your database,
    - Use optimized techniques to run your queries as efficiently as possible
    - Show the improvement of the performance after using optimization on any single query.
    - Run Hive queries to answer the questions given above.
- Cleaning up
    - Drop your database, and
    - Terminate your cluster

1. Creating and Launching an EMR cluster

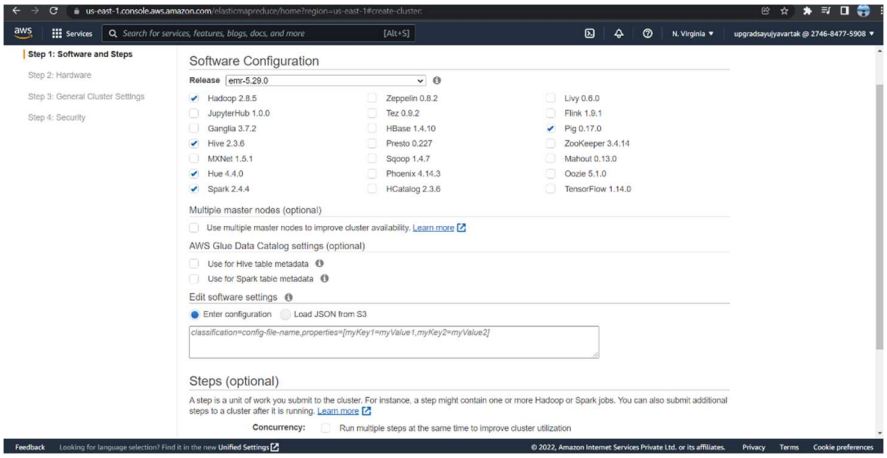
Click on create cluster



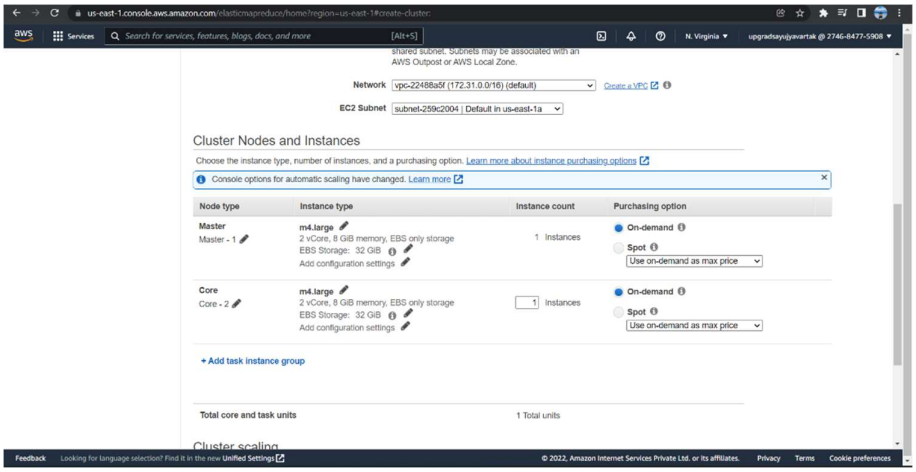
Go to Advanced Options

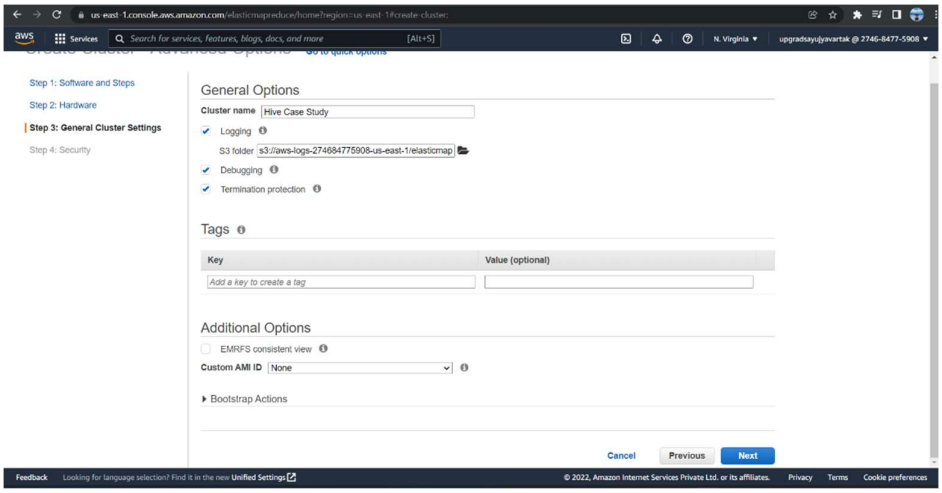


Choosing EMR 5.29.0

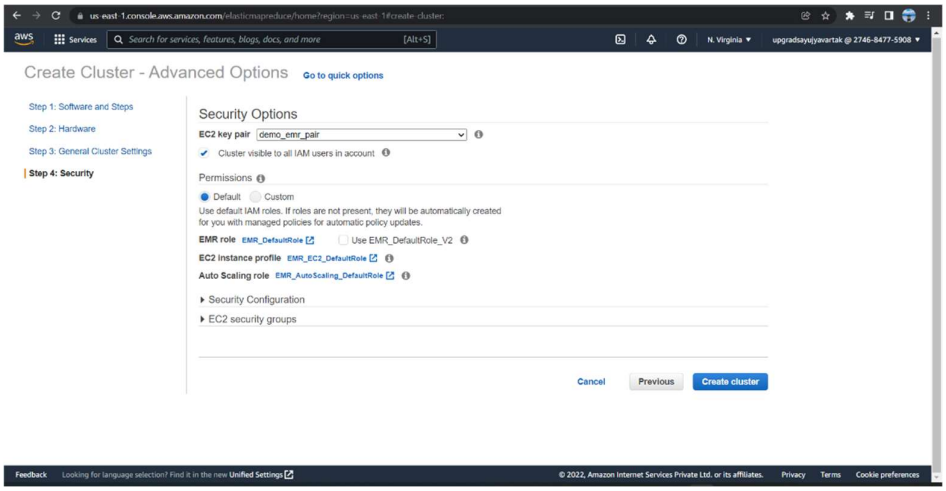


Selecting 1 master node and 1 core node(both m4.large)

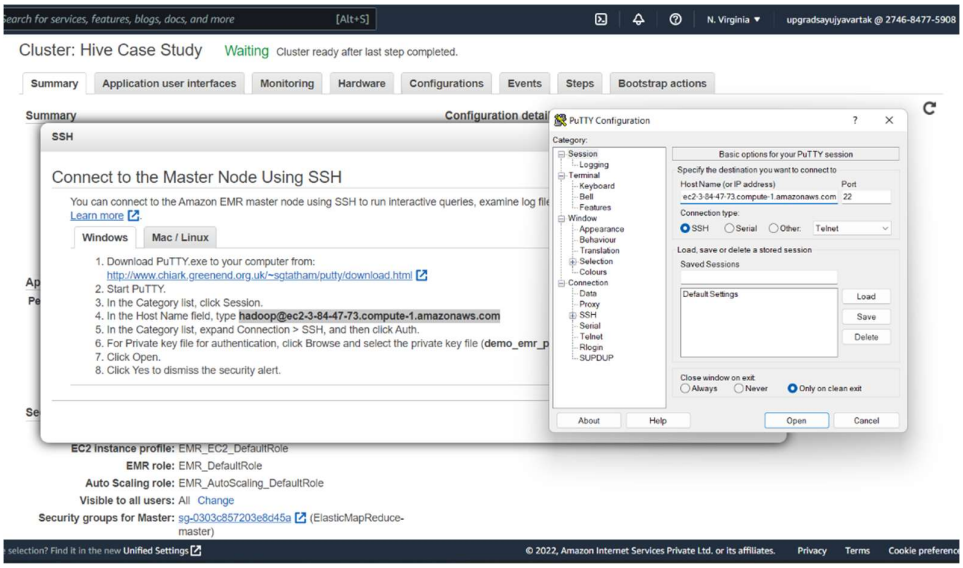


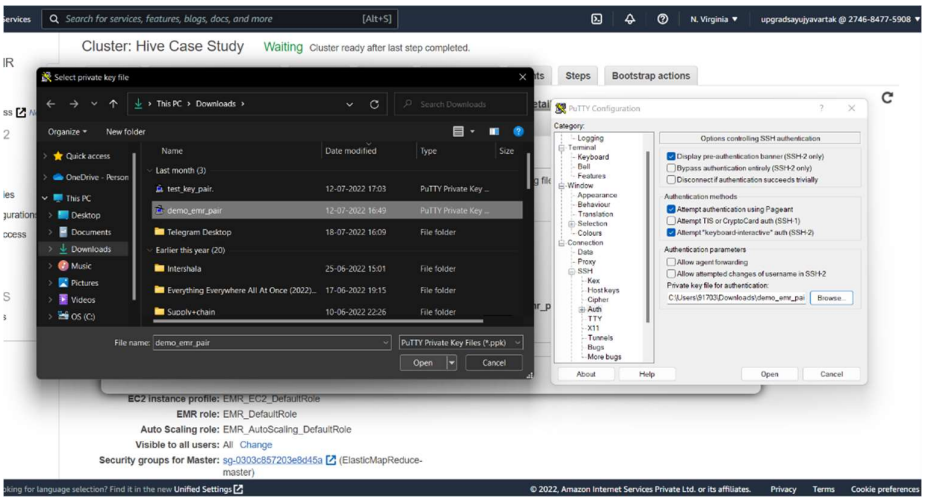


Selecting EC2 Key-Pair

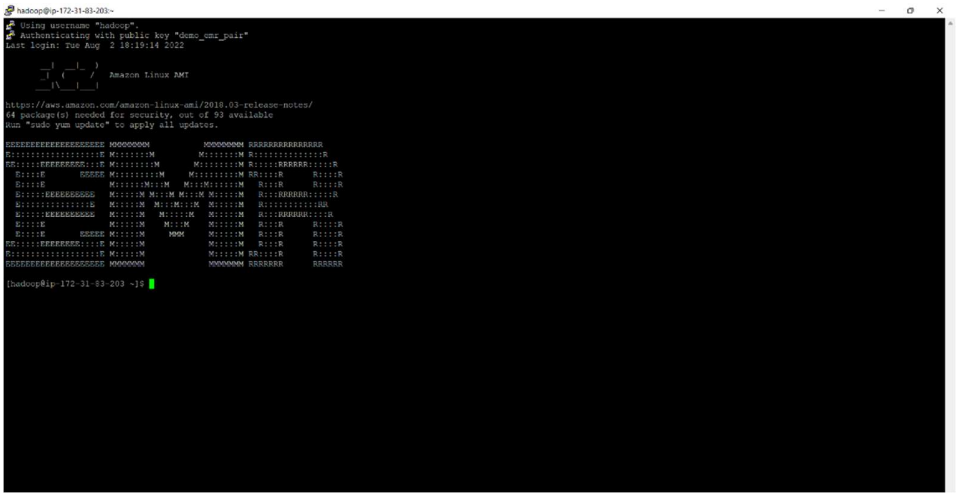


Once the cluster starts, we connect to master node using SSH by giving it in Host Name field in Putty and giving the EC2 key-pair in Auth





EMR cluster connected



2. Moving Data from S3 to HDFS

Firstly, we will create a directory to store the files we import from S3

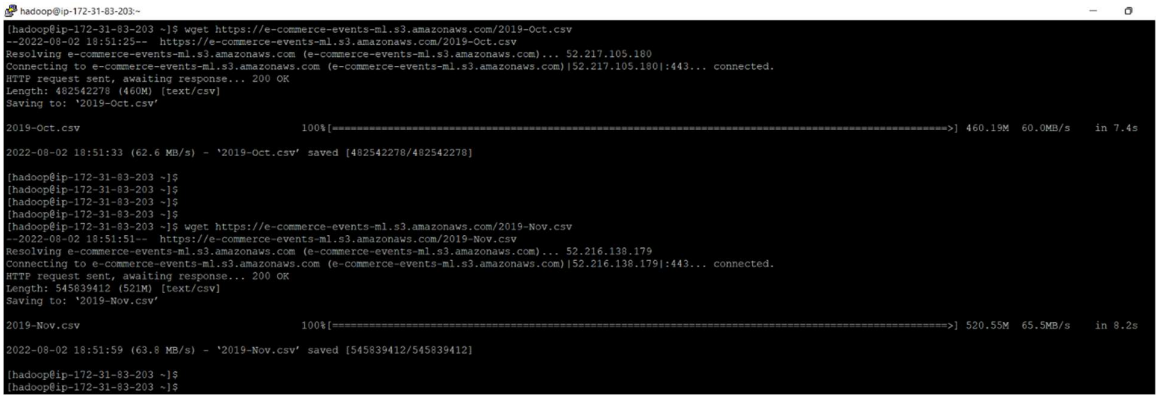
Creating a directory named ecom19

Command:- hadoop fs -mkdir /tmp/ecom19

```
[hadoop@ip-172-31-83-203 ~]$
[hadoop@ip-172-31-83-203 ~]$ hadoop fs -mkdir /tmp/ecom19
[hadoop@ip-172-31-83-203 ~]$
```

Importing both the dataset files one by one using ‘wget’

Commands:- **wget https://e-commerce-events-ml.s3.amazonaws.com/2019-Oct.csv**
wget https://e-commerce-events-ml.s3.amazonaws.com/2019-Nov.csv



Checking if files are imported

Command:- ls

Copying both the files one by one into the previously created directory ‘ecom19’

Commands:- **hadoop fs -put ./2019-Oct.csv /tmp/ecom19**
hadoop fs -put ./2019-Nov.csv /tmp/ecom19

Checking if the files have been copied

Command:- hadoop fs -ls /tmp/ecom19/

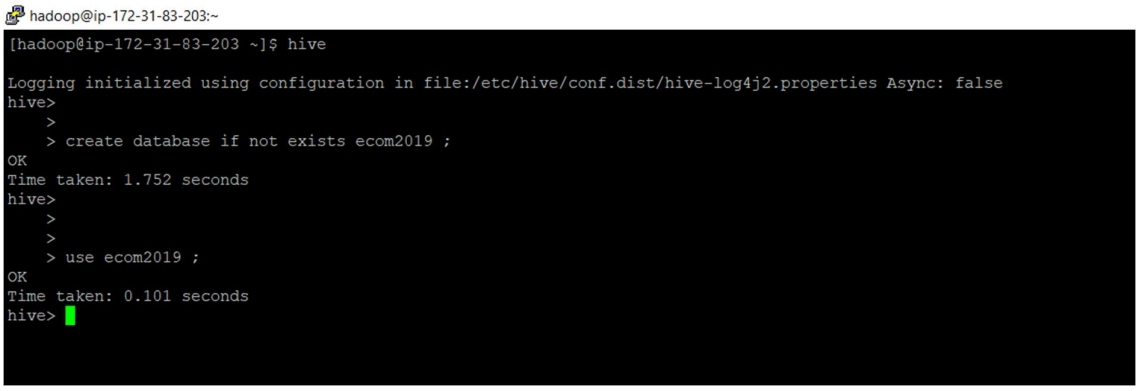
3. Database and Table creation

- We will first create database then create a table using OpenCSVSerde and import data of both files into it.
- Then we will create another table with appropriate data types of columns without partition as OpenCSVSerde has a limitation that all the columns in it are of string data type.
- Then we will create a partitioned table based on the column which has less cardinality and is mostly queried

Creating Database and selecting that database

Command:- create database if not exists ecom2019 ;

use ecom2019 ;



Creating table and importing data into it

Commands:-

create external table if not exists ecom19_serde (event_time timestamp, event_type string, product_id string, category_id string, category_code string, brand string, price decimal(10,3), user_id bigint, user_session string)
row format serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
with serdeproperties ('separatorChar' = ',')
stored as textfile
location '/tmp/ecom19/'
tblproperties ('skip.header.line.count' = '1') ;

```
    >
    >
    >
    > create external table if not exists ecom19_serde (event_time timestamp, event_type string, product_id string, category_id string, category_code string, brand string, price decimal(10,
3), user_id bigint, user_session string)
    > row format serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
    > with serdeproperties ('separatorChar' = ',')
    > stored as textfile
    > location '/tmp/ecom19/'
    > tblproperties ('skip.header.line.count' = '1') ;
OK
Time taken: 0.941 seconds
hive>
```

Checking table columns and dtypes and few records

Command:- desc ecom19_serde ;

```
    >
    >
    >
    > desc ecom19_serde ;
OK
event_time              string                  from deserializer
event_type              string                  from deserializer
product_id              string                  from deserializer
category_id             string                  from deserializer
category_code           string                  from deserializer
brand                   string                  from deserializer
price                   string                  from deserializer
user_id                 string                  from deserializer
user_session            string                  from deserializer
Time taken: 0.265 seconds, Fetched: 9 row(s)
hive>
    >
    >
    >
    >
```

Command:- SELECT * FROM ecom19_serde LIMIT 5 ;

```
    >
    >
    > select * from ecom19_serde limit 5 ;
OK
2019-11-01 00:00:02 UTC view    5802432 1487580009286598681                     0.32    562076640       09fafd6c-6c99-46b1-834f-33527f4de241
2019-11-01 00:00:09 UTC cart    5844397 1487580006317032337                     2.38    553329724       2067216c-31b5-455d-a1cc-af0575a34ffb
2019-11-01 00:00:10 UTC view    5837166 1783999064103190764             pnb     22.22   556138645       57ed222e-a54a-4907-9944-5a875c2d7f4f
2019-11-01 00:00:11 UTC cart    5876812 1487580010100293687             jessnail        3.16    564506666       186c1951-8052-4b37-adce-dd9644b1d5f7
2019-11-01 00:00:24 UTC remove_from_cart        5826182 1487580007483048900                     3.33    553329724       2067216c-31b5-455d-a1cc-af0575a34ffb
Time taken: 0.233 seconds, Fetched: 5 row(s)
hive>
    >
```

Now Creating a non-serde table and importing data into it

Command:-

create external table if not exists ecom19_data (event_time timestamp, event_type string, product_id string, category_id string, category_code string, brand string, price decimal(10,3) , user_id bigint, user_session string) ;

```
    >
    >
    >
    >
    > create external table if not exists ecom19_data (event_time timestamp, event_type string, product_id string, category_id string, category_code string, brand string, price decimal(10,3
) , user_id bigint, user_session string)  ;
OK
Time taken: 0.406 seconds
hive>
    >
```

Command: - desc ecom19_data ;

```
    >
    >
    > desc ecom19_data ;
OK
event_time              timestamp
event_type              string
product_id              string
category_id             string
category_code           string
brand                   string
price                   decimal(10,3)
user_id                 bigint
user_session            string
Time taken: 0.061 seconds, Fetched: 9 row(s)
hive>
```

Printing Header

Command:- set hive.cli.print.header = true ;

```
    >
    > set hive.cli.print.header = true ;
hive>
```

Inserting data from ecom19_serde to ecom19_data table.

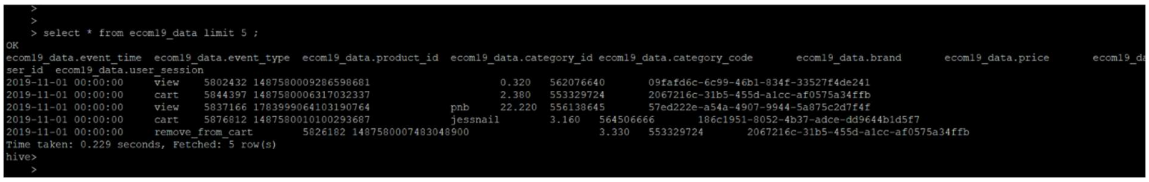
Here we have used 'TO_UTC_TIMESTAMP' function for parsing timestamp data which is of UTC format

Command:-

INSERT INTO TABLE ecom19_data
SELECT TO_UTC_TIMESTAMP(DATE_FORMAT(event_time,'yyyy-MM-dd HH:mm:ss.SSS'),'UTC') as event_time, event_type , product_id , category_id , category_code , brand , price , user_id , user_session FROM ecom19_serde ;

```
    >
    >
    > insert into table ecom19_data select to_utc_timestamp(date_format(event_time,'yyyy-MM-dd HH:mm:ss.SSS'),'UTC') as event_time, event_type , product_id , category_id , category_code , b
rand , price , user_id , user_session from ecom19_serde ;
Query ID = hadoop_20220802190047_bd328db4-1d99-4bf6-8353-dfebc7dcd08a
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1659462073453_0002)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ......... container     SUCCEEDED      2          2        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 01/01  [==========================>>] 100%  ELAPSED TIME: 242.37 s
--------------------------------------------------------------------------------
Loading data to table default.ecom19_data
OK
Time taken: 246.702 seconds
hive>
    >
```

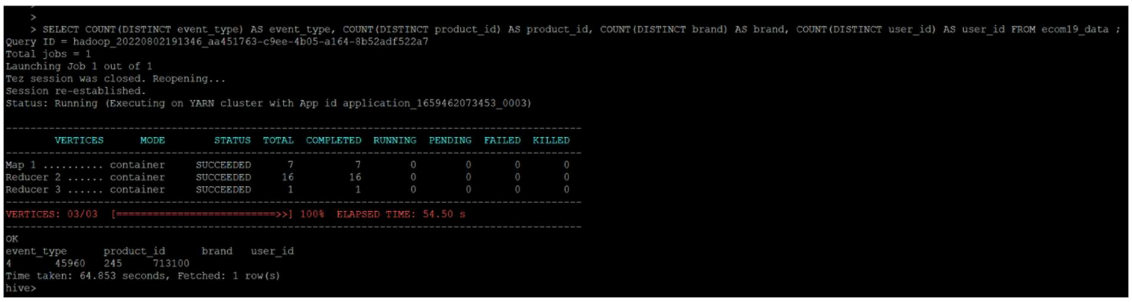
Command:- SELECT * FROM ecom19_data LIMIT 5 ;



Now checking for number of distinct values in columns so to partition accordingly

Command:-

```
SELECT COUNT(DISTINCT event_type) AS event_type,
       COUNT(DISTINCT product_id) AS product_id,
       COUNT(DISTINCT brand) AS brand,
       COUNT(DISTINCT user_id) AS user_id
FROM ecom19_data ;
```

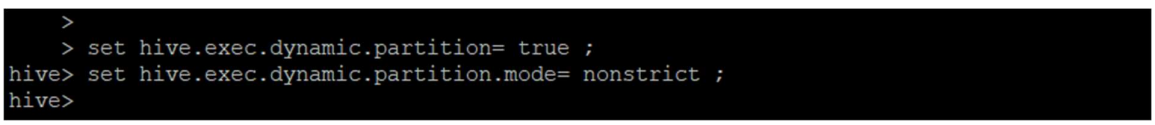


Insight :-

- **The column 'event_type' has only cardinality of 4 and is also an important column in querying.**
- **Also Bucketing any column didn't improve the performance of queries so we will be only using partitioning in this case.**

Setting Dynamic Partitioning to True and creating partitioned table

Commands:- set hive.exec.dynamic.partition= true ;
set hive.exec.dynamic.partition.mode= nonstrict ;



Command:-

create external table if not exists ecom19_data_part (event_time timestamp, product_id string, category_id string, category_code string, brand string, price decimal(10,3) , user_id bigint, user_session string)
partitioned by (event_type string) ;

```
    >
    > create external table if not exists ecom19_data_part (event_time timestamp, product_id string, category_id string, category_code string, brand string, price decimal(10,3) , user_id bi
gint, user_session string)  partitioned by (event_type string)  ;
OK
Time taken: 0.086 seconds
hive>
    >
```

Checking the columns of created table and inserting data into it

Command:- desc ecom19_data_part ;

```
    >
    > desc ecom19_data_part ;
OK
col_name        data_type       comment
event_time              timestamp
product_id              string
category_id             string
category_code           string
brand                   string
price                   decimal(10,3)
user_id                 bigint
user_session            string
event_type              string

# Partition Information
# col_name              data_type               comment

event_type              string
Time taken: 0.1 seconds, Fetched: 14 row(s)
hive>
    >
```

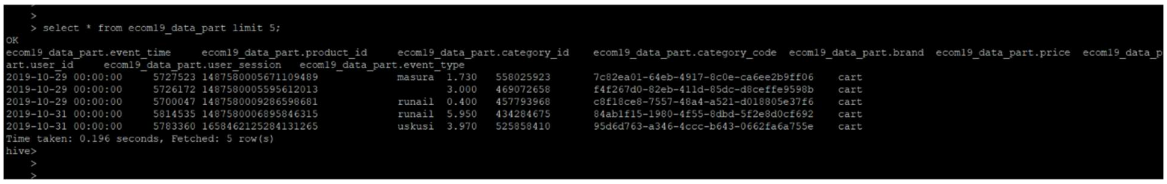
Command:-

INSERT INTO TABLE ecom19_data_part PARTITION (event_type)
SELECT event_time, product_id , category_id , category_code , brand , price , user_id , user_session, event_type
FROM ecom19_data ;

```
    >
    >
    > insert into table ecom19_data_part partition (event_type)
    > select event_time, product_id , category_id , category_code , brand , price , user_id , user_session, event_type  from ecom19_data ;
Query ID = hadoop_20220802192018_16a162cf-99d4-4b4c-b694-2f47c0f15153
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_1659462073453_0004)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      7          7        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      4          4        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 126.14 s
--------------------------------------------------------------------------------
Loading data to table default.ecom19_data_part partition (event_type=null)

Loaded : 4/4 partitions.
        Time taken to load dynamic partitions: 0.61 seconds
        Time taken for adding to write entity : 0.001 seconds
OK
event_time      product_id      category_id     category_code   brand   price   user_id user_session    event_type
Time taken: 138.296 seconds
hive>
    >
```

Checking partitions and if data in imported properly

Command:- SELECT * FROM ecom19_data_part LIMIT 5;



Command:- show partitions ecom19_data_part ;

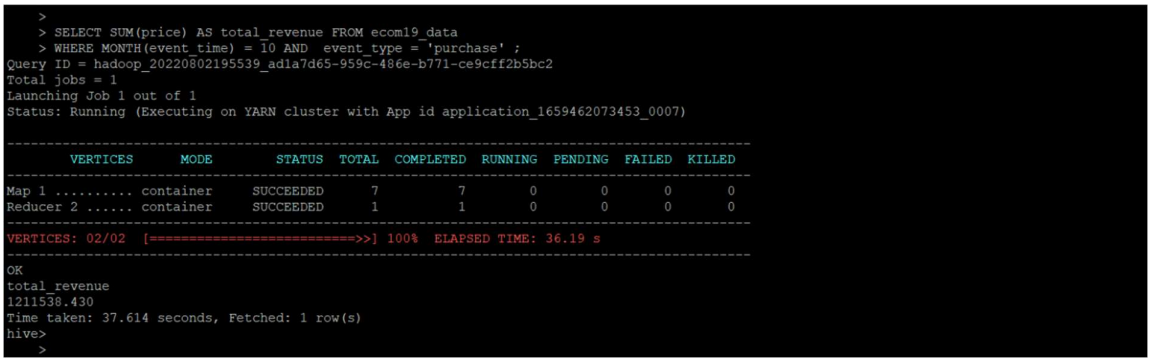
4. Querying and Analysis

We will query into non-partitioned table(ecom19_data) and partitioned table(ecom19_data_part) simultaneously to check the query optimization

1) Find the total revenue generated due to purchases made in October.

Query :- (Non-partitioned**)**

SELECT SUM(price) AS total_revenue FROM ecom19_data
WHERE MONTH(event_time) = 10 AND event_type = 'purchase' ;



Query :- (Partitioned)

SELECT SUM(price) AS total_revenue FROM ecom19_data_part
WHERE MONTH(event_time) = 10 AND event_type = 'purchase' ;

```
    >
    > SELECT SUM(price) AS total_revenue FROM ecom19_data_part
    > WHERE MONTH(event_time) = 10 AND  event_type = 'purchase' ;
Query ID = hadoop_20220802195638_4fe4a485-e915-4f2e-b711-cc4aae571b71
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1659462073453_0007)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      2          2        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 15.86 s
--------------------------------------------------------------------------------
OK
total_revenue
1211538.430
Time taken: 17.204 seconds, Fetched: 1 row(s)
hive>
    >
```

Insight :- Total Revenue generated due to purchases in month of October was 1211530.430

2) Write a query to yield the total sum of purchases per month in a single output.

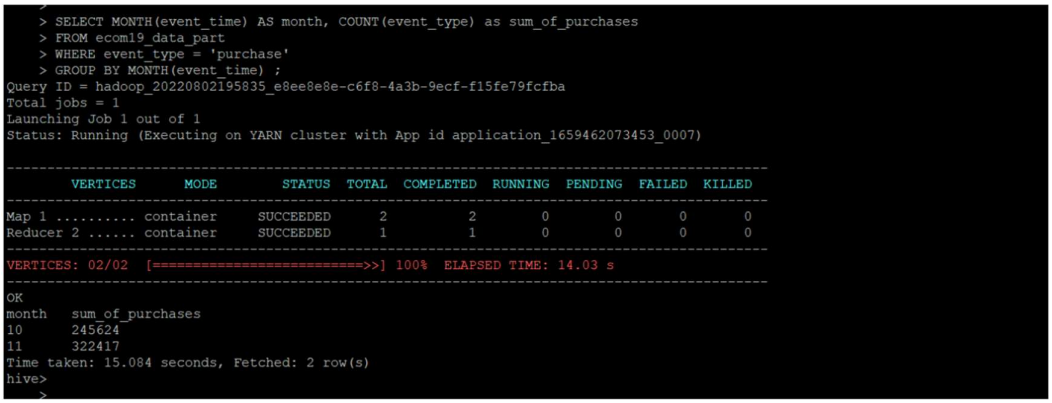
Query :- (Non-partitioned)

SELECT MONTH(event_time) AS month, COUNT(event_type) as sum_of_purchases
FROM ecom19_data
WHERE event_type = 'purchase'
GROUP BY MONTH(event_time) ;

```
    >
    > SELECT MONTH(event_time) AS month, COUNT(event_type) as sum_of_purchases
    > FROM ecom19_data
    > WHERE event_type = 'purchase'
    > GROUP BY MONTH(event_time) ;
Query ID = hadoop_20220802195741_de2c7767-5f85-46a6-a4db-5fa7826e5aff
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1659462073453_0007)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      7          7        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      2          2        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 29.96 s
--------------------------------------------------------------------------------
OK
month   sum_of_purchases
10      245624
11      322417
Time taken: 30.634 seconds, Fetched: 2 row(s)
hive>
    >
```

Query :- (Partitioned)

```
SELECT MONTH(event_time) AS month, COUNT(event_type) as sum_of_purchases
FROM ecom19_data_part
WHERE event_type = 'purchase'
GROUP BY MONTH(event_time) ;
```



Insight :- The total sum of purchases made in October were 2,45,624 while for November the sum of purchases increased to 3,22,417

3) Write a query to find the change in revenue generated due to purchases from October to November.

Query :- (Non-partitioned)

```
SELECT LEAD(SUM(price), 1) OVER (ORDER BY MONTH(event_time)) - SUM(price) AS
change_in_revenue
FROM ecom19_data
WHERE event_type = 'purchase'
GROUP BY MONTH(event_time)
LIMIT 1 ;
```

```
    >
    > SELECT LEAD(SUM(price), 1) OVER (ORDER BY MONTH(event_time)) - SUM(price) AS change_in_revenue
    > FROM ecom19_data
    > WHERE event_type = 'purchase'
    > GROUP BY MONTH(event_time)
    > LIMIT 1 ;
Query ID = hadoop_20220802200039_bc108e49-7f10-4389-98bd-f41068f37010
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1659462073453_0007)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      7          7        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: 30.03 s
--------------------------------------------------------------------------------
OK
change_in_revenue
319478.470
Time taken: 30.775 seconds, Fetched: 1 row(s)
hive>
    >
```

Query :- **(**Partitioned**)**

SELECT LEAD(SUM(price), 1) OVER (ORDER BY MONTH(event_time)) - SUM(price) AS change_in_revenue
FROM ecom19_data_part
WHERE event_type = 'purchase'
GROUP BY MONTH(event_time)
LIMIT 1 ;

```
    >
    >
    > SELECT LEAD(SUM(price), 1) OVER (ORDER BY MONTH(event_time)) - SUM(price) AS change_in_revenue
    > FROM ecom19_data_part
    > WHERE event_type = 'purchase'
    > GROUP BY MONTH(event_time)
    > LIMIT 1 ;
Query ID = hadoop_20220802200131_f6f6f1af-5a13-47bd-8503-8f6989c7163a
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1659462073453_0007)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      2          2        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 15.20 s
--------------------------------------------------------------------------------
OK
change_in_revenue
319478.470
Time taken: 15.945 seconds, Fetched: 1 row(s)
hive>
    >
```

Insight :- The change in revenue from October to November was 319478.470 i.e. the revenue increased.

4) Find distinct categories of products. Categories with null category code can be ignored.

Query :- **(**Non-partitioned**)**

SELECT DISTINCT SPLIT(category_code,'\\.')[0] AS category
FROM ecom19_data
WHERE SPLIT(category_code,'\\.')[0] <> '' ;

```
    >
    > SELECT DISTINCT SPLIT(category_code,'\\.')[0]  AS category
    > FROM ecom19_data
    > WHERE SPLIT(category_code,'\\.')[0] <> '';
Query ID = hadoop_20220802200216_bc7e1af0-3df5-4c19-9aa5-d168215489fa
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1659462073453_0007)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      7          7        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      4          4        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 32.79 s
--------------------------------------------------------------------------------
OK
category
accessories
appliances
furniture
stationery
apparel
sport
Time taken: 33.408 seconds, Fetched: 6 row(s)
hive>
    >
```

Query :- (Partitioned**)**

SELECT DISTINCT SPLIT(category_code,'\\.')[0] AS category
FROM ecom19_data_part
WHERE SPLIT(category_code,'\\.')[0] <> '' ;

```
    >
    > SELECT DISTINCT SPLIT(category_code,'\\.')[0]  AS category
    > FROM ecom19_data_part
    > WHERE SPLIT(category_code,'\\.')[0] <> '';
Query ID = hadoop_20220802200306_bcf109ac-664f-4787-870a-3c0a493b5db7
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1659462073453_0007)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      7          7        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      4          4        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 34.39 s
--------------------------------------------------------------------------------
OK
category
accessories
appliances
furniture
stationery
apparel
sport
Time taken: 35.043 seconds, Fetched: 6 row(s)
hive>
    >
```

Insight :- The distinct categories of products are ‘accessories’, ‘appliances’, ‘furniture’, ‘stationary’, ’apparel’ and ‘sport’

5) Find the total number of products available under each category.

Query :- (Non-partitioned**)**

```
SELECT SPLIT(category_code,'\\.')[0] AS  category,
        COUNT(product_id) as product_count
FROM ecom19_data
WHERE SPLIT(category_code,'\\.')[0]  <>  ''
GROUP BY SPLIT(category_code,'\\.')[0]
ORDER BY product_count DESC;
```

```
    >
    > SELECT SPLIT(category_code,'\\.')[0] AS category, COUNT(product_id) as product_count
    > FROM ecom19_data
    > WHERE SPLIT(category_code,'\\.')[0] <> ''
    > GROUP BY SPLIT(category_code,'\\.')[0]
    > ORDER BY product_count DESC;
Query ID = hadoop_20220802202229_ba959c4a-b5c4-4787-ab62-c0234b6a8827
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_1659462073453_0009)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........  container     SUCCEEDED      7          7        0        0       0       0
Reducer 2 ......  container     SUCCEEDED      4          4        0        0       0       0
Reducer 3 ......  container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 34.13 s
--------------------------------------------------------------------------------
OK
category        product_count
appliances      61736
stationery      26722
furniture       23604
apparel 18232
accessories     12929
sport   2
Time taken: 43.272 seconds, Fetched: 6 row(s)
hive>
    >
```

Query :- (Partitioned**)**

```
SELECT SPLIT(category_code,'\\.')[0] AS  category,
        COUNT(product_id) as product_count
FROM ecom19_data_part
WHERE SPLIT(category_code,'\\.')[0]  <>  ''
GROUP BY SPLIT(category_code,'\\.')[0]
ORDER BY product_count DESC;
```

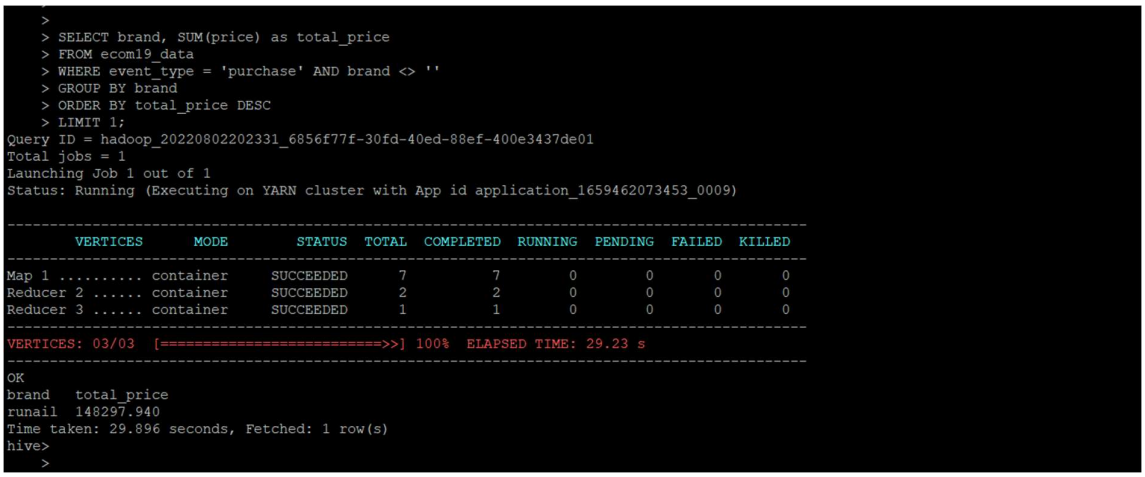
```
    >
    >
    > SELECT SPLIT(category_code,'\\.')[0] AS category, COUNT(product_id) as product_count
    > FROM ecom19_data_part
    > WHERE SPLIT(category_code,'\\.')[0] <> ''
    > GROUP BY SPLIT(category_code,'\\.')[0]
    > ORDER BY product_count DESC;
Query ID = hadoop_20220802201226_12ee4063-0463-4348-99a9-f99c4bb1129a
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_1659462073453_0008)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........  container     SUCCEEDED      7          7        0        0       0       0
Reducer 2 ......  container     SUCCEEDED      4          4        0        0       0       0
Reducer 3 ......  container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 34.67 s
--------------------------------------------------------------------------------
OK
category        product_count
appliances      61736
stationery      26722
furniture       23604
apparel 18232
accessories     12929
sport   2
Time taken: 43.682 seconds, Fetched: 6 row(s)
hive>
    >
```

Insight :- **‘Appliances’ category has the highest number of products i.e. 61,736 while ‘sport’ has only 2**

6) Which brand had the maximum sales in October and November combined?

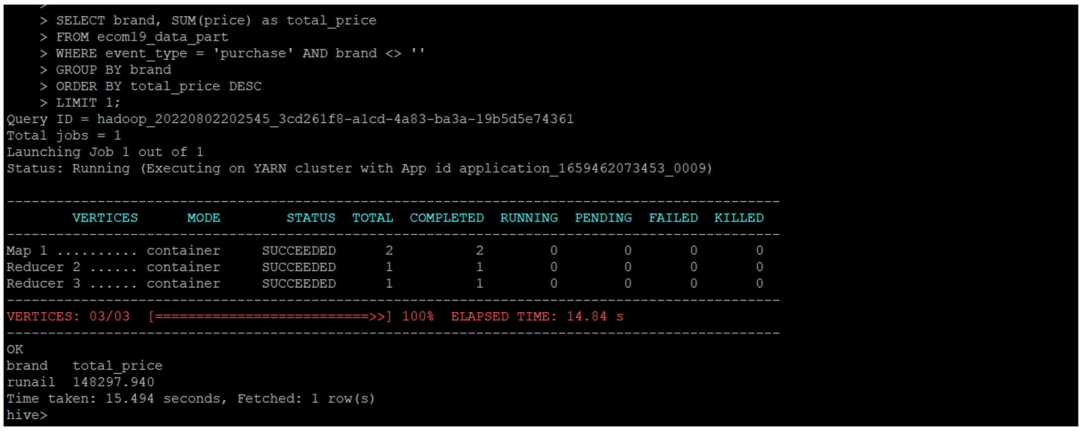
Query :- **(**Non-partitioned**)**

```
SELECT brand,
        SUM(price) as total_price
FROM ecom19_data
WHERE event_type = 'purchase' AND brand <> ''
GROUP BY brand
ORDER BY total_price DESC
LIMIT 1;
```



Query :- **(**Partitioned**)**

```
SELECT brand,
        SUM(price) as total_price
FROM ecom19_data_part
WHERE event_type = 'purchase' AND brand <> ''
GROUP BY brand
ORDER BY total_price DESC
LIMIT 1;
```



Insight :- ‘Runail’ is the brand having highest sales in month of October and November combined.

7) Which brands increased their sales from October to November?

Query :- (Non-partitioned**)**

```
WITH monthly_revenue AS (
SELECT brand,
SUM(CASE WHEN MONTH(event_time)=10 THEN price ELSE 0 END) AS oct_revenue,
SUM(CASE WHEN MONTH(event_time)=11 THEN price ELSE 0 END) AS nov_revenue
FROM ecom19_data
WHERE event_type='purchase' AND brand <> ''
GROUP BY brand
)
SELECT brand, nov_revenue - oct_revenue AS  revenue_difference
FROM monthly_revenue
WHERE (nov_revenue - oct_revenue) > 0
ORDER BY revenue_difference DESC ;
```

```
    >
    > WITH monthly_revenue AS (
    > SELECT brand,
    > SUM(CASE WHEN MONTH(event_time)=10 THEN price ELSE 0 END) AS oct_revenue,
    > SUM(CASE WHEN MONTH(event_time)=11 THEN price ELSE 0 END) AS nov_revenue
    > FROM ecom19_data
    > WHERE event_type='purchase' AND brand <> ''
    > GROUP BY brand
    > )
    > SELECT brand, nov_revenue-oct_revenue AS  revenue_difference
    > FROM monthly_revenue
    > WHERE (nov_revenue - oct_revenue) > 0
    > ORDER BY revenue_difference DESC ;
Query ID = hadoop_20220802202623_ba0e0b6b-ceb3-4f12-b4b0-437db50b4e14
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1659462073453_0009)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      7          7        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: 32.56 s
----------------------------------------------------------------------------------------------
OK
brand   revenue_difference
grattol 36027.170
uno     15737.720
lianail 10501.400
ingarden        10404.820
strong  9474.640
jessnail        7057.390
cosmoprofi      6214.180
polarus 5358.210
runail  5219.380
freedecor       4250.020
staleks 3355.880
bpw.style       3265.290
lovely  3234.680
marathon        2992.350
haruyama        2962.220
yoko    2950.970
italwax 2859.130
benovy  2850.350
```

hadoop@ip-172-31-83-203:~

```
benovy  2850.350
kaypro  2387.360
estel   2385.920
concept 2348.260
kapous  2165.920
f.o.x   1953.050
masura  1792.390
milv    1737.070
beautix 1729.000
artex   1596.610
domix   1537.120
shik    1498.520
smart   1444.880
roubloff        1422.410
levrana 1420.540
oniq    1416.240
irisk   1354.080
severina        1344.600
joico   1309.580
zeitun  1300.970
beauty-free     1228.690
swarovski       1155.230
de.lux  1115.810
metzger 1083.710
markell 1065.680
sanoto  1052.540
nagaraku        957.940
ecolab  951.450
art-visage      905.090
levissime       857.810
missha  856.450
solomeya        786.100
rosi    764.520
refectocil      759.400
kaaral  673.640
kosmekka        631.930
kinetics        611.010
browxenna       585.360
airnails        572.620
uskusi  548.040
coifin  525.490
s.care  500.390
limoni  487.700
matrix  483.490
gehwol  468.610
greymy  460.280
bioaqua 455.230
farmavita       454.600
sophin  447.660
```

hadoop@ip-172-31-83-203:~

```
sophin  447.660
yu-r    402.300
kiss    395.780
naomi   389.000
lador   387.920
ellips  360.190
jas     338.470
lowence 324.910
nitrile 315.400
shary   304.530
kims    302.000
happyfons       289.670
kocostar        284.080
insight 278.260
candy   264.420
bluesky 258.290
beauugreen      256.840
protokeratin    255.540
trind   244.890
entity  239.550
skinlite        238.510
provoc  235.830
fedua   211.430
ecocraft        200.790
keen    199.270
mane    193.470
freshbubble     183.640
matreshka       182.670
chi     179.670
cristalinas     157.320
farmona 150.970
latinoil        135.070
miskin  135.030
elizavecca      133.770
nefertiti       133.120
finish  132.000
igrobeauty      131.410
dizao   126.380
osmo    116.730
batiste 101.770
carmex  98.280
eos     98.270
depilflax       96.710
enjoy   95.220
kerasys 94.290
aura    93.560
plazan  92.640
koelf   84.560
nirvel  71.290
```

hadoop@ip-172-31-83-203:~

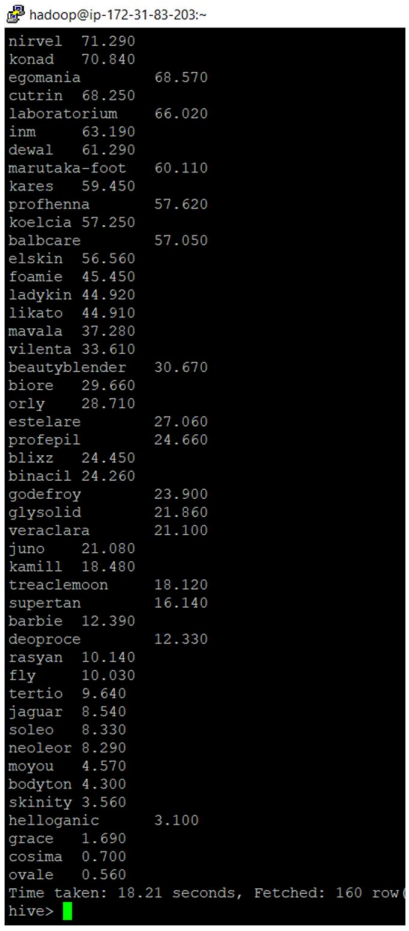
```
nirvel  71.290
konad   70.840
egomania        68.570
cutrin  68.250
laboratorium    66.020
inm     63.190
dewal   61.290
marutaka-foot   60.110
kares   59.450
profhenna       57.620
koelcia 57.250
balbcare        57.050
elskin  56.560
foamie  45.450
ladykin 44.920
likato  44.910
mavala  37.280
vilenta 33.610
beautyblender   30.670
biore   29.660
orly    28.710
estelare        27.060
profepil        24.660
blixz   24.450
binacil 24.260
godefroy        23.900
glysolid        21.860
veraclara       21.100
juno    21.080
kamill  18.480
treaclemoon     18.120
supertan        16.140
barbie  12.390
deoproce        12.330
rasyan  10.140
fly     10.030
tertio  9.640
jaguar  8.540
soleo   8.330
neoleor 8.290
moyou   4.570
bodyton 4.300
skinity 3.560
helloganic      3.100
grace   1.690
cosima  0.700
ovale   0.560
Time taken: 33.222 seconds, Fetched: 160 row(s)
hive>
```

Query :- (Partitioned)

```
WITH monthly_revenue AS (
SELECT brand,
SUM(CASE WHEN MONTH(event_time)=10 THEN price ELSE 0 END) AS oct_revenue,
SUM(CASE WHEN MONTH(event_time)=11 THEN price ELSE 0 END) AS nov_revenue
FROM ecom19_data_part
WHERE event_type='purchase' AND brand <> ''
GROUP BY brand
)
SELECT brand, nov_revenue-oct_revenue AS  revenue_difference
FROM monthly_revenue
WHERE (nov_revenue - oct_revenue) > 0
ORDER BY revenue_difference DESC ;
```

```
    >
    > WITH monthly_revenue AS (
    > SELECT brand,
    > SUM(CASE WHEN MONTH(event_time)=10 THEN price ELSE 0 END) AS oct_revenue,
    > SUM(CASE WHEN MONTH(event_time)=11 THEN price ELSE 0 END) AS nov_revenue
    > FROM ecom19_data_part
    > WHERE event_type='purchase' AND brand <> ''
    > GROUP BY brand
    > )
    > SELECT brand, nov_revenue-oct_revenue AS  revenue_difference
    > FROM monthly_revenue
    > WHERE (nov_revenue - oct_revenue) > 0
    > ORDER BY revenue_difference DESC ;
Query ID = hadoop_20220802202940_f79fb69d-9649-4ff3-aa37-a69978477185
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1659462073453_0009)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      2          2        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 17.57 s
----------------------------------------------------------------------------------------------
OK
brand   revenue_difference
grattol 36027.170
uno     15737.720
lianail 10501.400
ingarden        10404.820
strong  9474.640
jessnail        7057.390
cosmoprofi      6214.180
polarus 5358.210
runail  5219.380
freedecor       4250.020
staleks 3355.880
bpw.style       3265.290
lovely  3234.680
marathon        2992.350
haruyama        2962.220
yoko    2950.970
italwax 2859.130
benovy  2850.350
```

Output Continued



Insight :- About 160 brands increased their sales going from October to November. While grattol had highest increase of 36027.170 in sales, ovale had the least with only 0.560 increase

8) Your company wants to reward the top 10 users of its website with a Golden Customer plan. Write a query to generate a list of top 10 users who spend the most

Query :- (Non-partitioned**)**

```
SELECT user_id,
        SUM(price) as total_expense
FROM ecom19_data
WHERE event_type = 'purchase'
GROUP BY user_id
ORDER BY total_expense DESC
LIMIT 10;
```

```
    > SELECT user_id, SUM(price) as total_expense
    > FROM ecom19_data
    > WHERE event_type = 'purchase'
    > GROUP BY user_id
    > ORDER BY total_expense DESC
    > LIMIT 10;
Query ID = hadoop_20220802203213_7e2562e8-9d17-40ae-8ca8-060a7fdaa20b
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1659462073453_0009)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      7          7        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: 31.84 s
--------------------------------------------------------------------------------
OK
user_id total_expense
557790271       2715.870
150318419       1645.970
562167663       1352.850
531900924       1329.450
557850743       1295.480
522130011       1185.390
561592095       1109.700
431950134       1097.590
566576008       1056.360
521347209       1040.910
Time taken: 32.444 seconds, Fetched: 10 row(s)
hive>
```

Query :- (Partitioned**)**

```
SELECT user_id,
        SUM(price) as total_expense
FROM ecom19_data_part
WHERE event_type = 'purchase'
GROUP BY user_id
ORDER BY total_expense DESC
LIMIT 10;
```

```
    >
    >
    > SELECT user_id, SUM(price) as total_expense
    > FROM ecom19_data_part
    > WHERE event_type = 'purchase'
    > GROUP BY user_id
    > ORDER BY total_expense DESC
    > LIMIT 10;
Query ID = hadoop_20220802203346_1132066a-2b88-4bab-a4d7-f3b037706fc2
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1659462073453_0009)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   container     SUCCEEDED      2          2        0        0       0       0
Reducer 2 ......   container     SUCCEEDED      1          1        0        0       0       0
Reducer 3 ......   container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 13.99 s
--------------------------------------------------------------------------------
OK
user_id total_expense
557790271       2715.870
150318419       1645.970
562167663       1352.850
531900924       1329.450
557850743       1295.480
522130011       1185.390
561592095       1109.700
431950134       1097.590
566576008       1056.360
521347209       1040.910
Time taken: 14.607 seconds, Fetched: 10 row(s)
hive>
    >
```

Insight:-_This is the top10 list of customers which spend the most

5. Cleaning Up

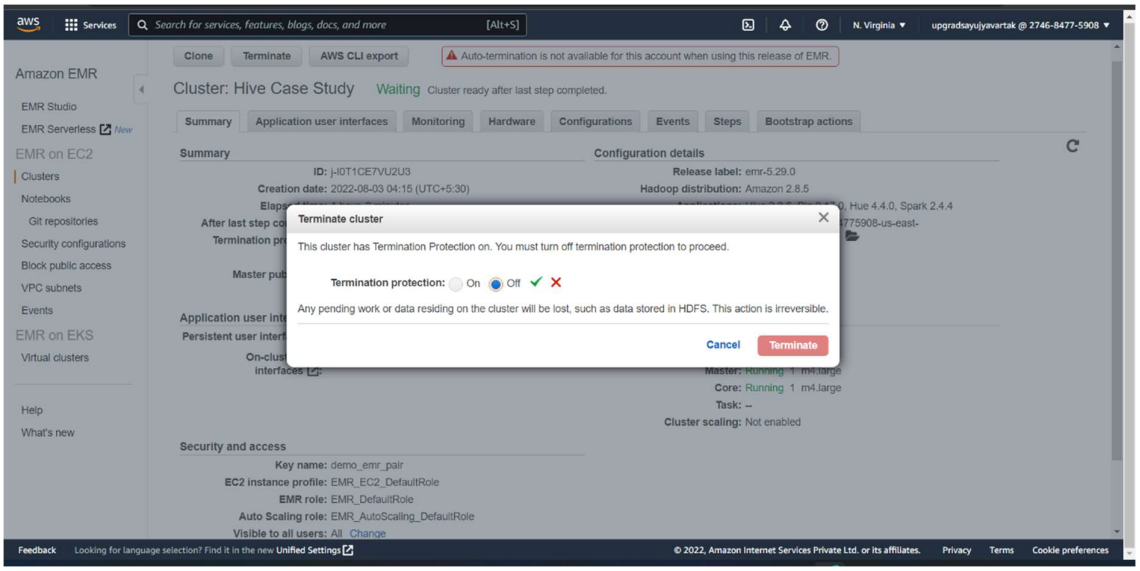
We will first drop the database and then terminate the cluster

Dropping the database, ‘Cascade’ keyword deletes the database schema even if there’s data using that schema, If we don’t use it we get a error.

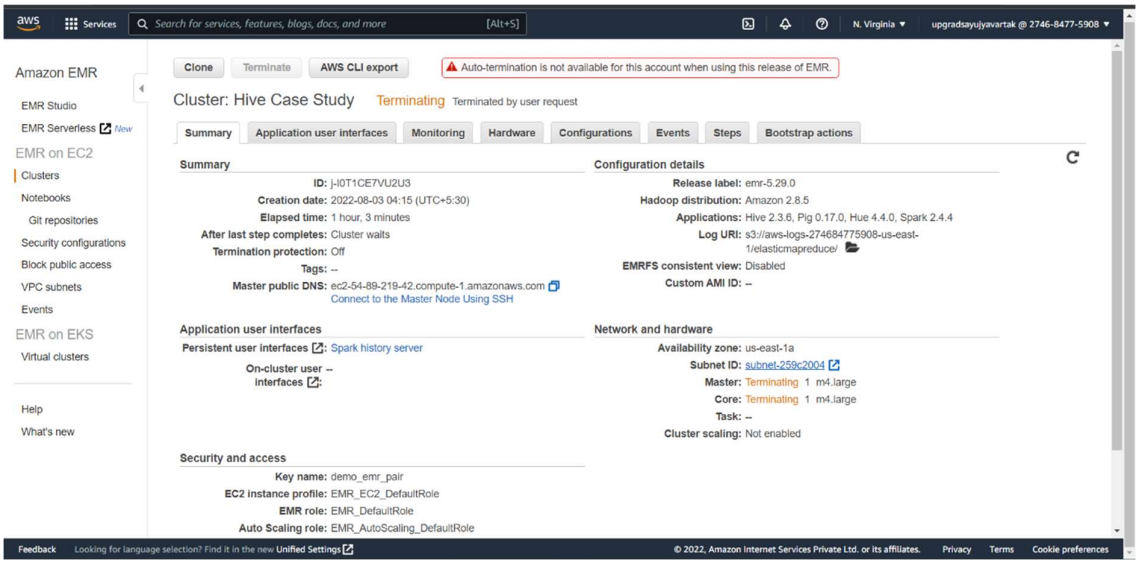
Command:- drop database ecom2019 cascade ;

```
    >
    >
    > drop database ecom2019 cascade ;
OK
Time taken: 0.888 seconds
hive>
    >
```

Turning off termination protection



Terminating EMR cluster



Summary

1) We Imported the dataset from S3 to HDFS in EMR cluster
2) Created tables with and without partitions and buckets to observe the difference in query performance
3) Optimized and answered all the asked Queries
4) **The queries took half the time to run in Partitioned table than that in non-partitioned table, <u>thus we optimized our query time by about half</u>**
5) After our analysis was done, we dropped the database and terminated the EMR cluster

Inference

1) The total sum of purchases increased from October to November
2) The total revenue also increased going from October to November with more than 50% brands increasing their sales from October to November
3) The general trend of revenue and sales is upwards and they will increase in following months