Hive Case Study

- Srilathaa Vasu, Shaik Athaullah

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. 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.

This is done by tracking their clicks on the website and searching for patterns within them. The clickstream data contains all the logs as to how the customer navigated through the website. It also contains other details such as time spent on every page, etc. From this, tech companies 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.

Data for this case study was available in:

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

Objective:

To extract data and gather insights from an e-commerce company dataset, using AWS EMR and S3, Hadoop and Hive systems.

Steps Involved:

1. Launching an EMR Cluster
2. Loading the data into S3 and ingesting the data to HDFS

3. Launching and creating Hive Schema
4. Optimization of tables
5. Querying – Assignment Questions
6. Improvement of performance after using optimization on tables
7. Cleaning-up

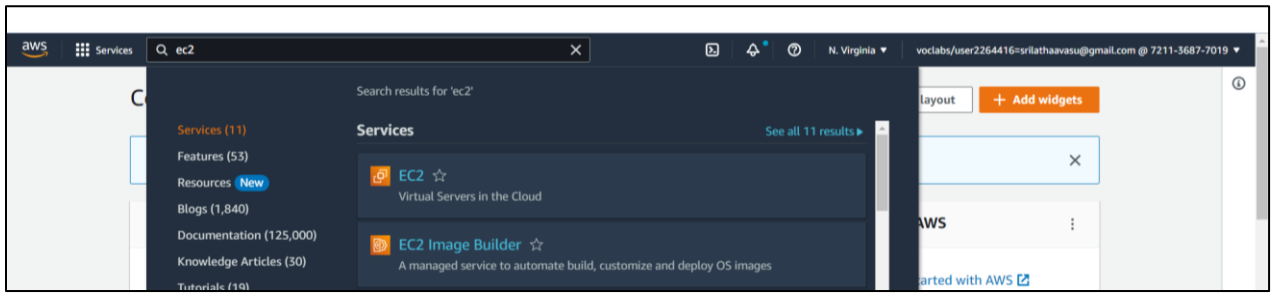
1. Launching an EMR Cluster:

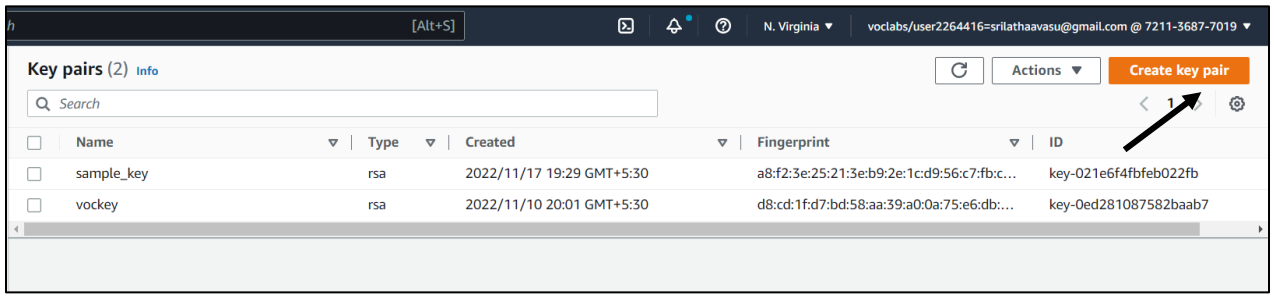
The following steps are followed while launching an EMR cluster,

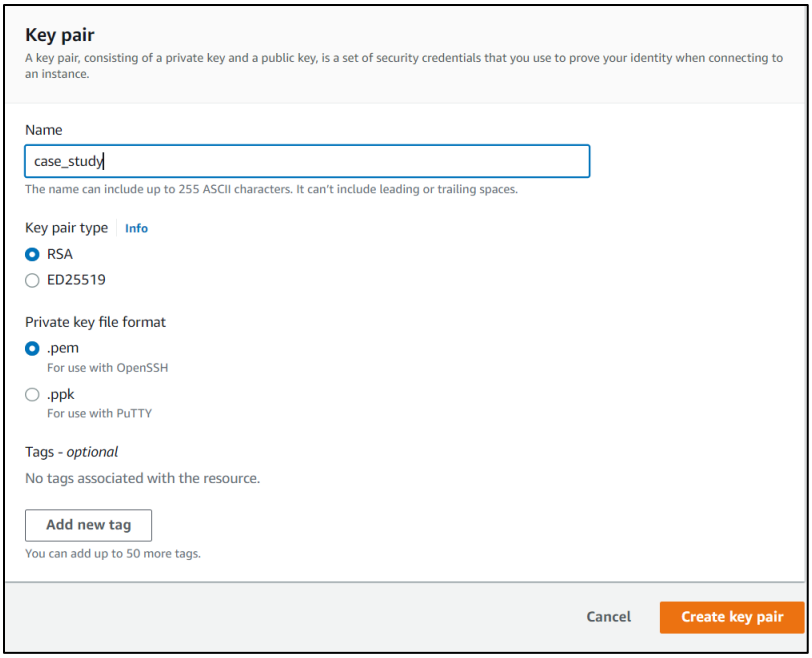
i. Key-pair creation
ii. Configuring & Launching an EMR cluster
iii. Connecting to master node using SSH
iv. Launching of Hive shell

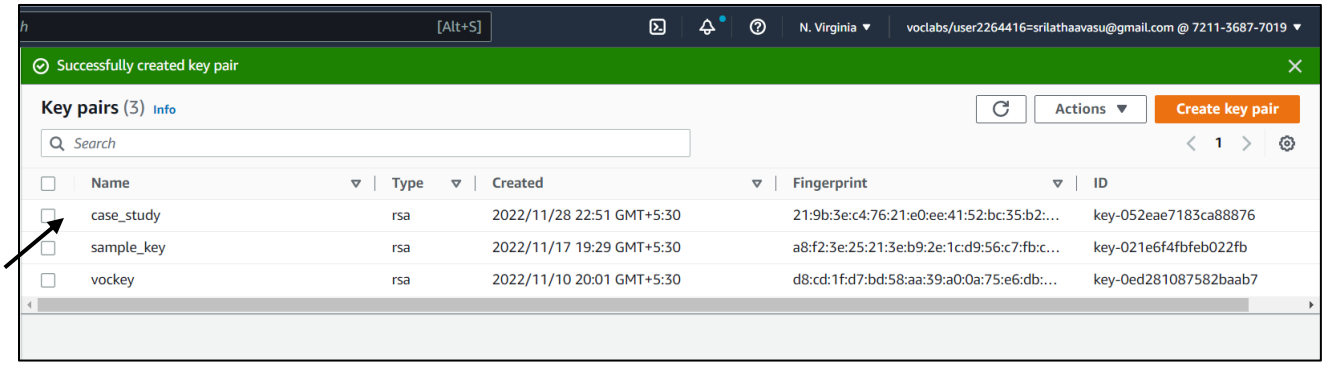
i. Key-pair creation:

We need to create an EC2 key-pair file and download it as .pem file.





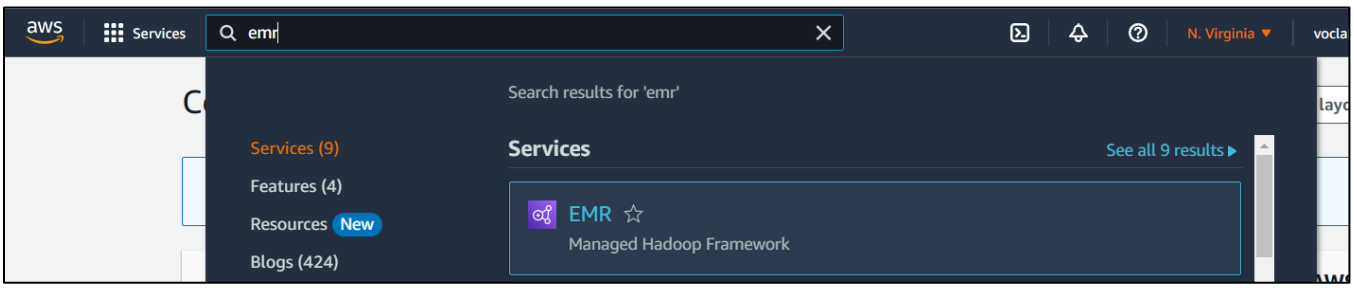


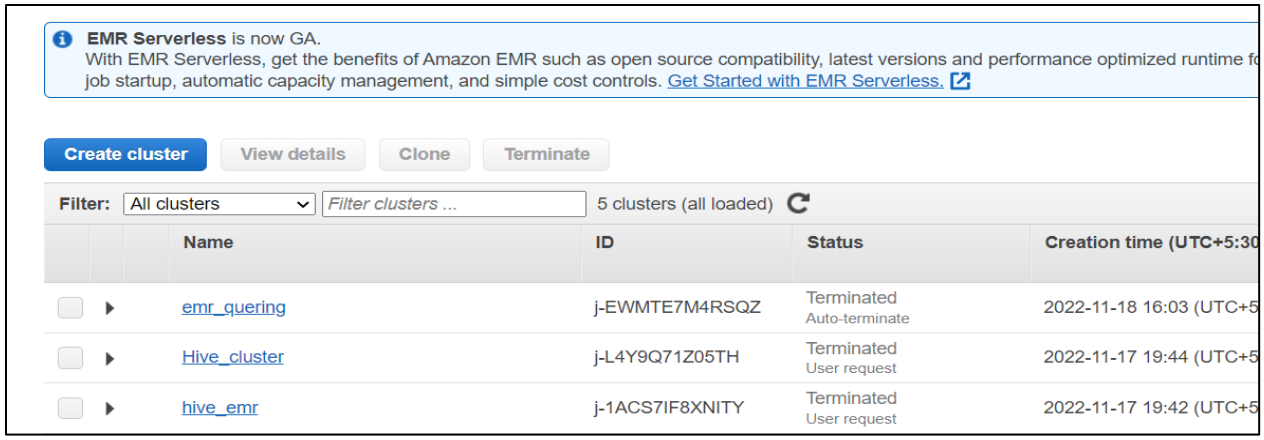


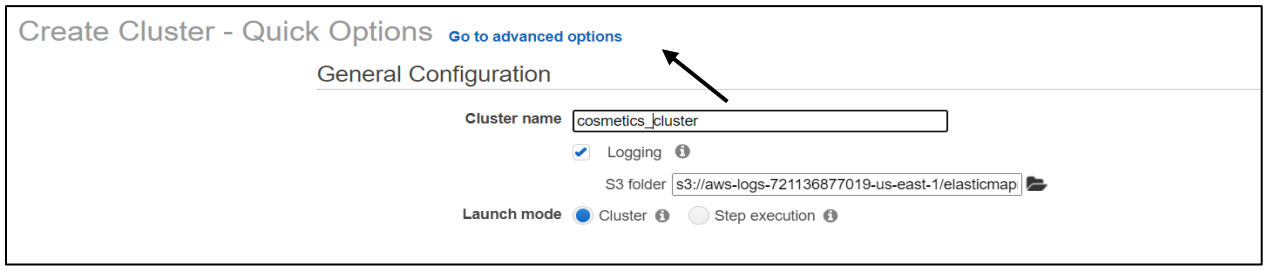
We will use this private key to securely connect to the EMR cluster through SSH.

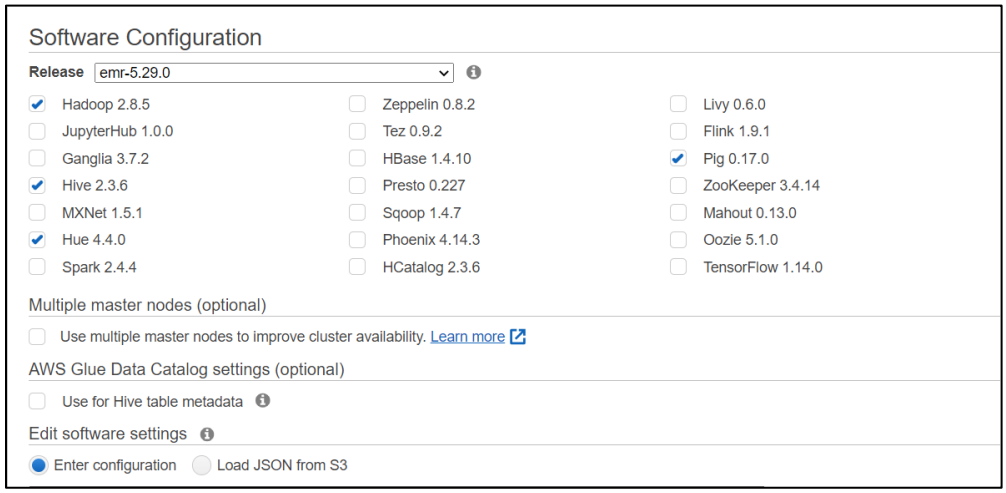
ii. Configuring & Launching an EMR cluster:

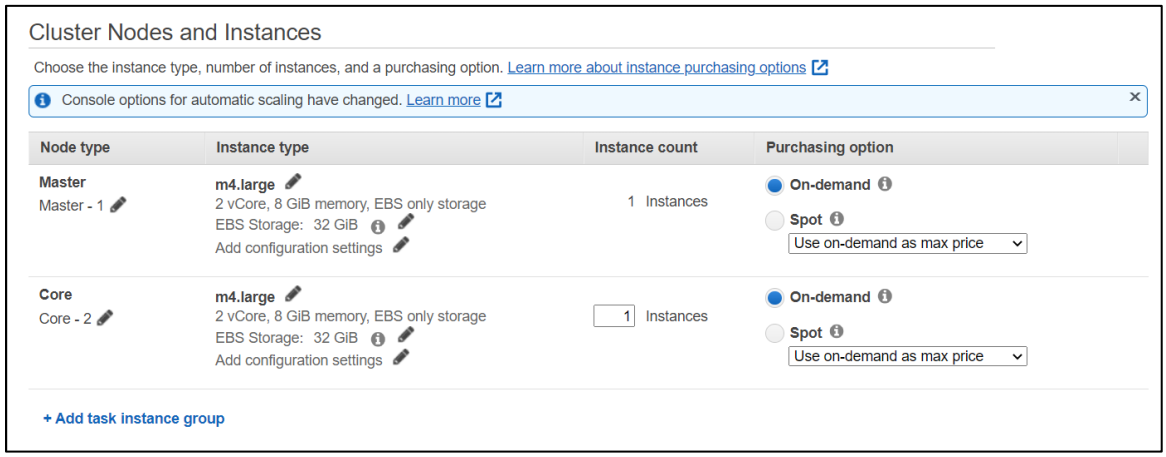
Amazon EMR (previously called Amazon Elastic MapReduce) is a managed cluster platform that simplifies running big data frameworks, such as Hadoop and Apache Spark on AWS to process and analyze vast amounts of data.



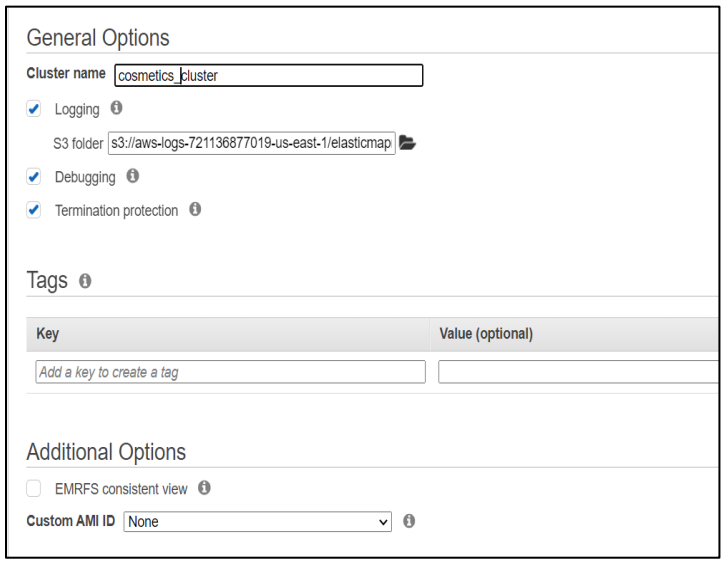

EMR Serverless is now GA.
With EMR Serverless, get the benefits of Amazon EMR such as open source compatibility, latest versions and performance optimized runtime f
job startup, automatic capacity management, and simple cost controls. Get Started with EMR Serverless.
Create cluster
View details
Clone
Terminate
Filter: All clusters
Filter clusters ...
5 clusters (all loaded)
Name
ID
Status
Creation time (UTC+5:30
emr_quering
j-EWMTE7M4RSQZ
Terminated
Auto-terminate
2022-11-18 16:03 (UTC+5
Hive_cluster
j-L4Y9Q71Z05TH
Terminated
User request
2022-11-17 19:44 (UTC+5
hive_emr
j-1ACS7IF8XNITY
Terminated
User request
2022-11-17 19:42 (UTC+5

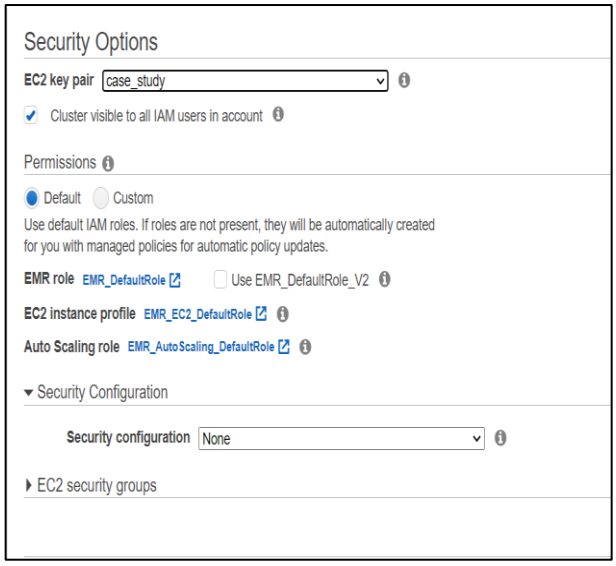

Create Cluster - Quick Options
Go to advanced options
General Configuration
Cluster name
cosmetics_cluster
Logging
S3 folder
s3://aws-logs-721136877019-us-east-1/elasticmap
Launch mode
Cluster
Step execution


Software Configuration
Release
emr-5.29.0
Hadoop 2.8.5
JupyterHub 1.0.0
Ganglia 3.7.2
Hive 2.3.6
MXNet 1.5.1
Hue 4.4.0
Spark 2.4.4
Zeppelin 0.8.2
Tez 0.9.2
HBase 1.4.10
Presto 0.227
Sqoop 1.4.7
Phoenix 4.14.3
HCatalog 2.3.6
Livy 0.6.0
Flink 1.9.1
Pig 0.17.0
ZooKeeper 3.4.14
Mahout 0.13.0
Oozie 5.1.0
TensorFlow 1.14.0
Multiple master nodes (optional)
Use multiple master nodes to improve cluster availability. Learn more
AWS Glue Data Catalog settings (optional)
Use for Hive table metadata
Edit software settings
Enter configuration
Load JSON from S3

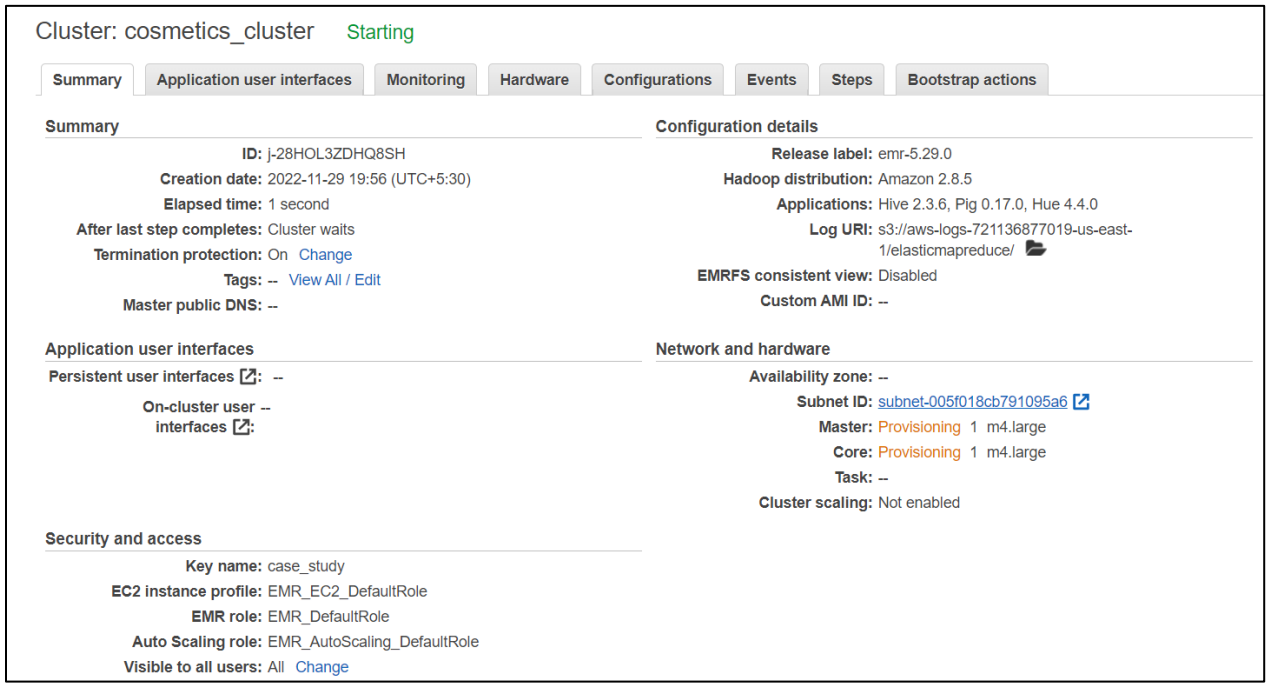

Cluster Nodes and Instances
Choose the instance type, number of instances, and a purchasing option. Learn more about instance purchasing options
Console options for automatic scaling have changed. Learn more
Node type
Instance type
Instance count
Purchasing option
Master
Master - 1
m4.large
2 vCore, 8 GiB memory, EBS only storage
EBS Storage: 32 GiB
Add configuration settings
1 Instances
On-demand
Spot
Use on-demand as max price
Core
Core - 2
m4.large
2 vCore, 8 GiB memory, EBS only storage
EBS Storage: 32 GiB
Add configuration settings
1 Instances
On-demand
Spot
Use on-demand as max price
+ Add task instance group

We have chosen emr-5.29.0 release for this case study.

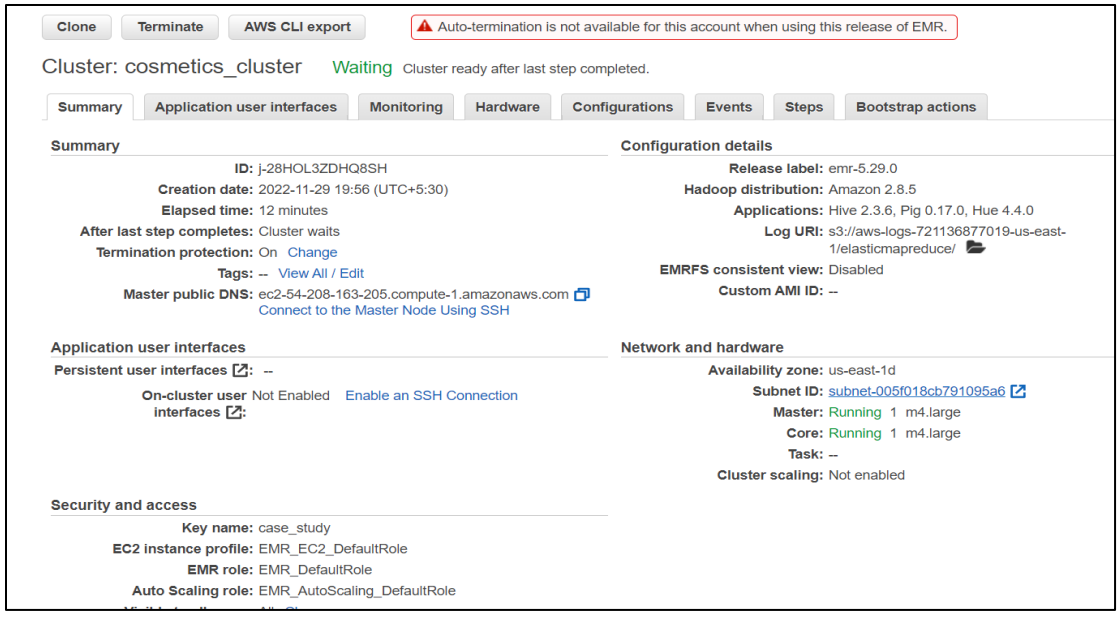




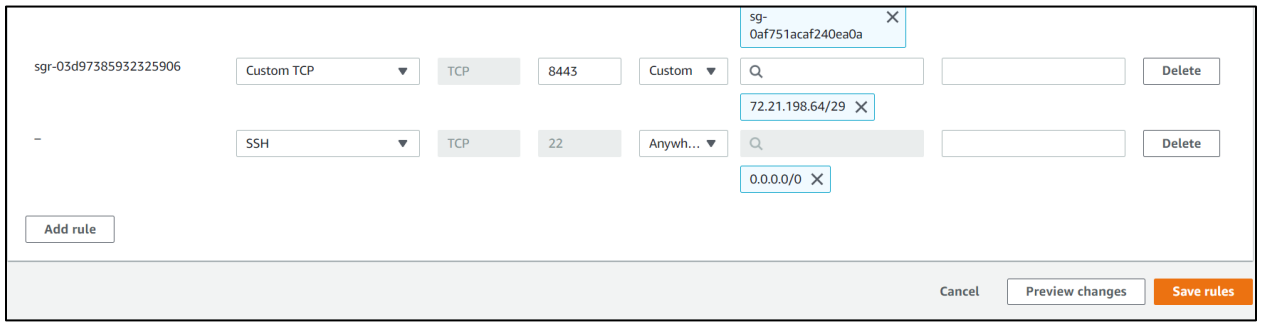
We have selected the “case_study” private key which we have created earlier.



Our cluster Hive Assignment is created and launched successfully and is now in “Waiting” state.

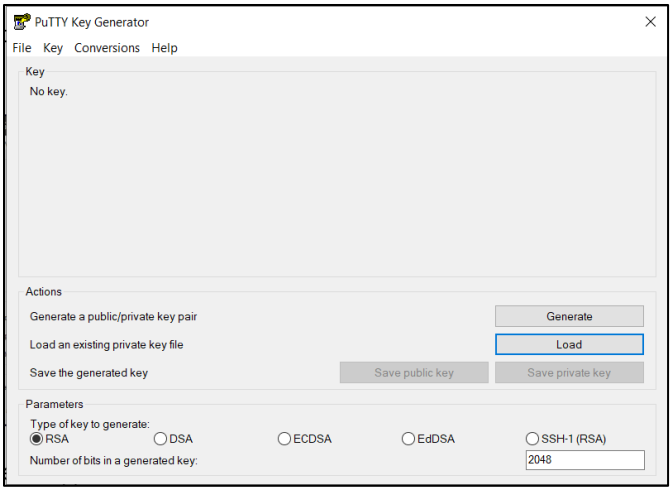


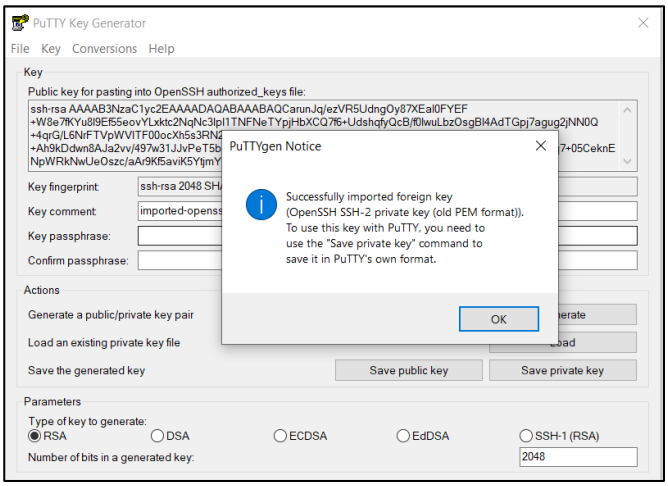
We need to ensure that the port is open to establish a connection. Under the cluster information page, click on the security groups of the master node. Here, we add an additional inbound rule as below.

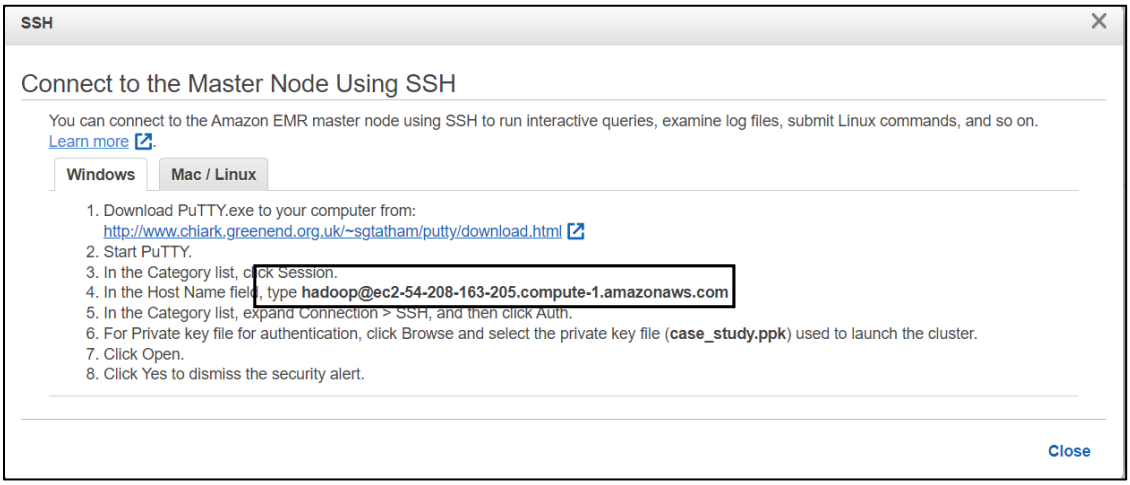


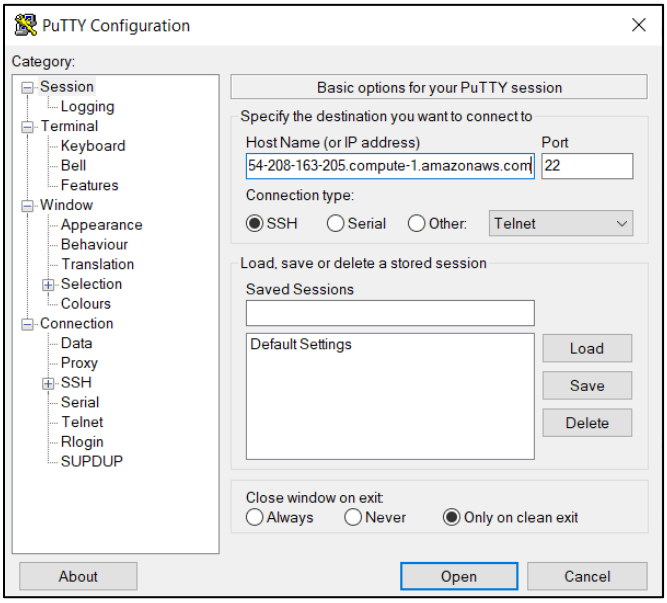
iii. Connecting to master node using SSH:

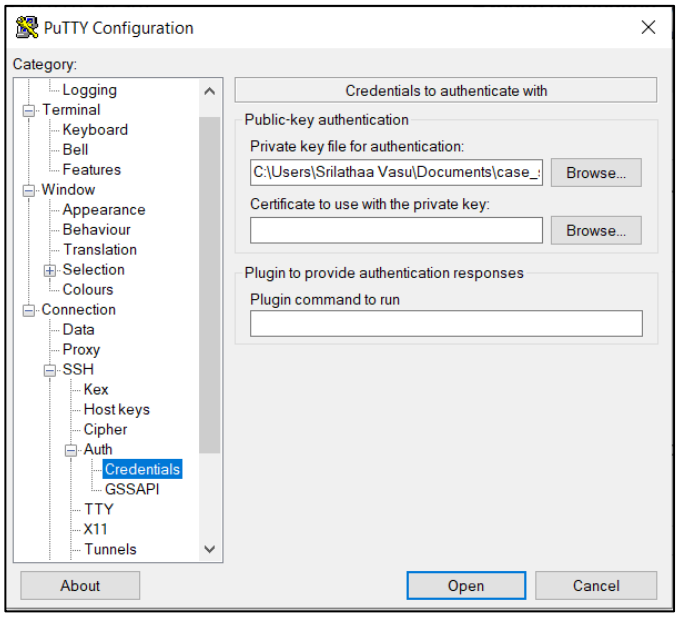
For connecting the master node using SSH, we need to use two programs namely, PuTTY and PuTTYgen. We first need to open PuTTYgen, then load the private key(case_study) and save it as .ppk file at a desired location. Next, we need to open PuTTY to connect to master node. We need to copy Master DNS address from the EMR cluster summary page.


PuTTY Key Generator
File Key Conversions Help
Key
No key.
Actions
Generate a public/private key pair
Generate
Load an existing private key file
Load
Save the generated key
Save public key
Save private key
Parameters
Type of key to generate:
RSA DSA ECDSA EdDSA SSH-1 (RSA)
Number of bits in a generated key:
2048

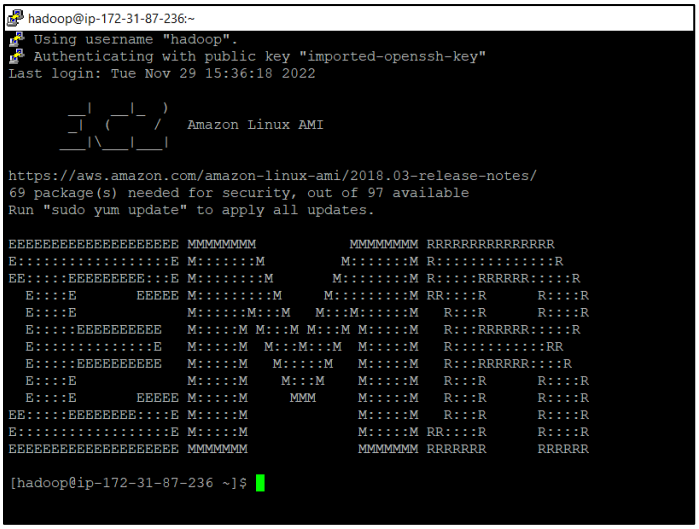

PuTTY Key Generator
File Key Conversions Help
Key
Public key for pasting into OpenSSH authorized_keys file:
Key fingerprint:
Key comment:
Key passphrase:
Confirm passphrase:
PuTTYgen Notice
Successfully imported foreign key
(OpenSSH SSH-2 private key (old PEM format)).
To use this key with PuTTY, you need to
use the "Save private key" command to
save it in PuTTY's own format.
OK
Actions
Generate a public/private key pair
Load an existing private key file
Save the generated key
Save public key
Save private key
Parameters
Type of key to generate:
RSA DSA ECDSA EdDSA SSH-1 (RSA)
Number of bits in a generated key:
2048


SSH
Connect to the Master Node Using SSH
You can connect to the Amazon EMR master node using SSH to run interactive queries, examine log files, submit Linux commands, and so on.
Learn more.
Windows
Mac / Linux
1. Download PuTTY.exe to your computer from:
http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
2. Start PuTTY.
3. In the Category list, click Session.
4. In the Host Name field, type hadoop@ec2-54-208-163-205.compute-1.amazonaws.com
5. In the Category list, expand Connection > SSH, and then click Auth.
6. For Private key file for authentication, click Browse and select the private key file (case_study.ppk) used to launch the cluster.
7. Click Open.
8. Click Yes to dismiss the security alert.
Close


PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
SSH
Serial
Telnet
Rlogin
SUPDUP
Basic options for your PuTTY session
Specify the destination you want to connect to
Host Name (or IP address)
Port
54-208-163-205.compute-1.amazonaws.com
22
Connection type:
SSH Serial Other: Telnet
Load, save or delete a stored session
Saved Sessions
Default Settings
Load
Save
Delete
Close window on exit:
Always Never Only on clean exit
About
Open
Cancel

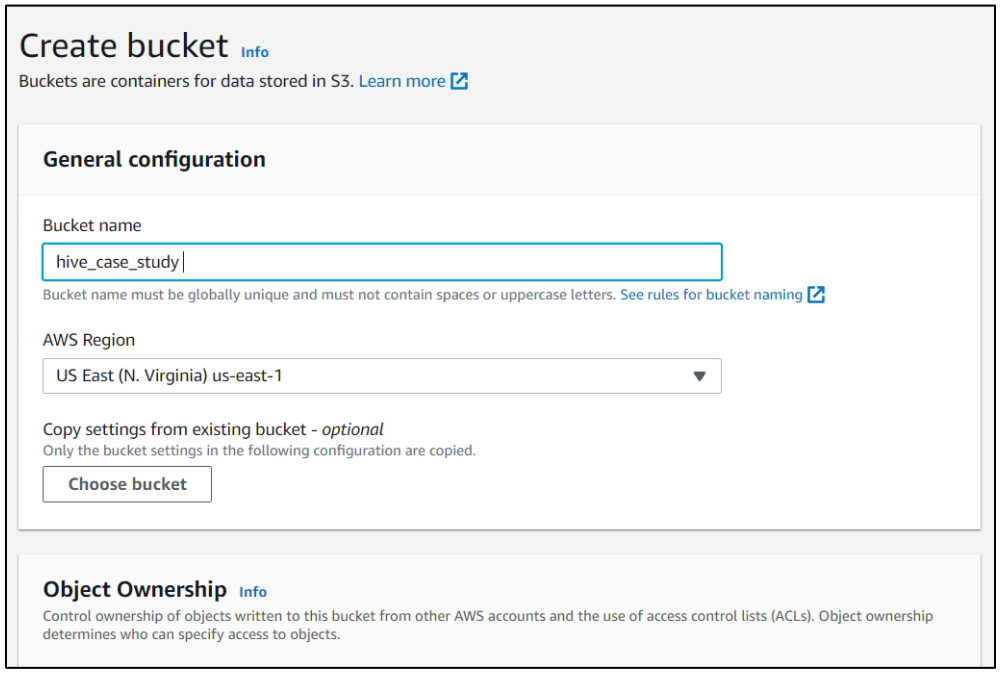

PuTTY Configuration
Category:
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
SSH
Kex
Host keys
Cipher
Auth
Credentials
GSSAPI
TTY
X11
Tunnels
Credentials to authenticate with
Public-key authentication
Private key file for authentication:
C:\Users\Srilathaa Vasu\Documents\case_
Browse...
Certificate to use with the private key:
Browse...
Plugin to provide authentication responses
Plugin command to run
About
Open
Cancel

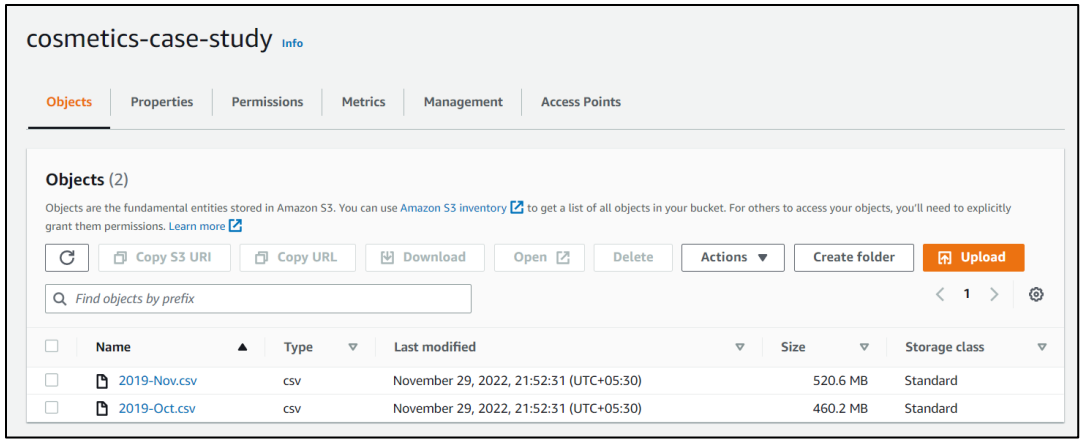
iv. Launching of Hive Shell:



2. Loading the data into S3 and ingesting the data to HDFS:

Uploading databases to S3 buckets:





Creating a temporary directory in HDFS:

Checking the directories already present in HDFS.

Command: *hadoop fs -ls /*
Output:

```
[hadoop@ip-172-31-87-236 ~]$ hadoop fs -ls /
Found 4 items
drwxr-xr-x   - hdfs hadoop          0 2022-11-29 14:33 /apps
drwxrwxrwt   - hdfs hadoop          0 2022-11-29 14:35 /tmp
drwxr-xr-x   - hdfs hadoop          0 2022-11-29 14:33 /user
drwxr-xr-x   - hdfs hadoop          0 2022-11-29 14:33 /var
```

Creating a temporary directory.

Command: *hadoop fs -mkdir /user/hivecasestudy*
hadoop fs -ls /user/
Output:

```
[hadoop@ip-172-31-87-236 ~]$ hadoop fs -mkdir /user/hivecasestudy
[hadoop@ip-172-31-87-236 ~]$ hadoop fs -ls /user/
Found 7 items
drwxrwxrwx   - hadoop hadoop          0 2022-11-29 14:33 /user/hadoop
drwxr-xr-x   - mapred mapred          0 2022-11-29 14:33 /user/history
drwxrwxrwx   - hdfs   hadoop          0 2022-11-29 14:33 /user/hive
drwxr-xr-x   - hadoop hadoop          0 2022-11-29 16:50 /user/hivecasestudy
drwxrwxrwx   - hue    hue             0 2022-11-29 14:33 /user/hue
drwxrwxrwx   - oozie  oozie           0 2022-11-29 14:33 /user/oozie
drwxrwxrwx   - root   hadoop          0 2022-11-29 14:33 /user/root
[hadoop@ip-172-31-87-236 ~]$
```

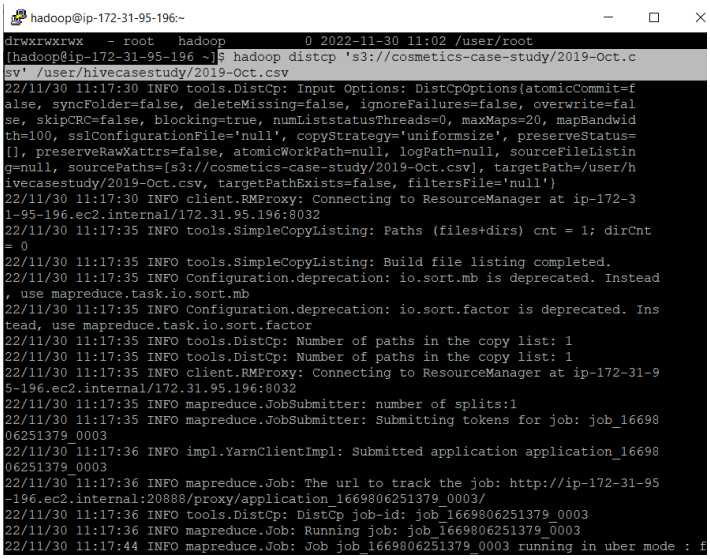
Loading the data to HDFS:

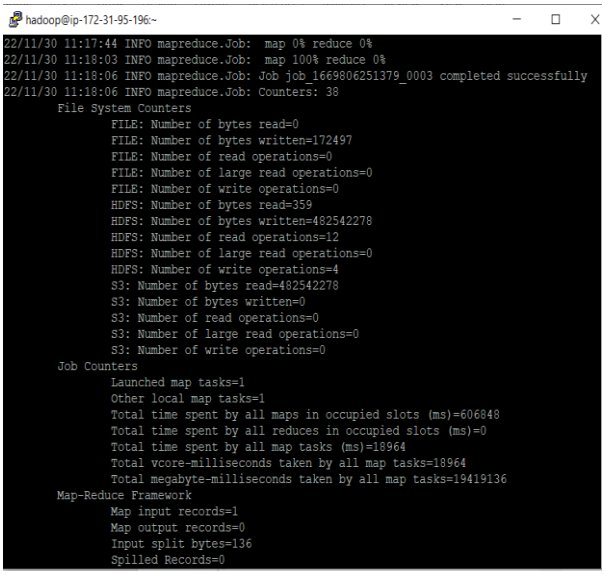
Since the size of the data is large we'll load the data into HDFS from S3 an into the local storage.

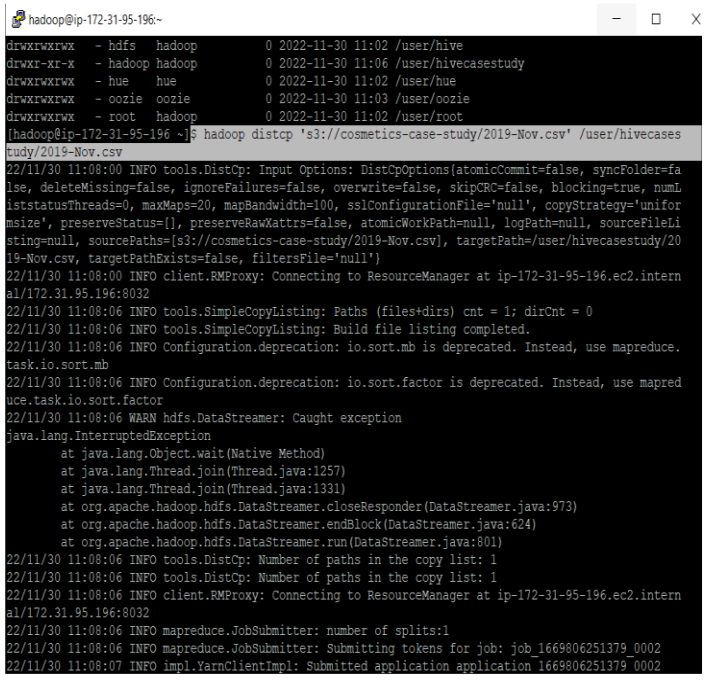
Command:

hadoop distcp 's3://cosmetics-case-study/2019-Oct.csv' /user/hivecasestudy/2019-Oct.csv
hadoop distcp 's3://cosmetics-case-study/2019-Nov.csv' /user/hivecasestudy/2019-Nov.csv

Output:







Checking the loaded files:

Command: *hadoop fs -ls /user/hivecasestudy/*

Output:

```
[hadoop@ip-172-31-95-196 ~]$ hadoop fs -ls /user/hivecasestudy/
Found 2 items
-rw-r--r--   1 hadoop hadoop  545839412 2022-11-30 11:08 /user/hivecasestudy/2019-Nov.csv
-rw-r--r--   1 hadoop hadoop  482542278 2022-11-30 11:18 /user/hivecasestudy/2019-Oct.csv
[hadoop@ip-172-31-95-196 ~]$
```

We can see that the datasets have been loaded successfully.

3. Launching and creating Hive Schema:

Launching Hive:

Command: *hive*

Output:

```
[hadoop@ip-172-31-95-196 ~]$ hive

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.
properties Async: false
hive>
```

Creating a new database for the case study:

Command: *create database if not exists hive_casestudy;*
use hive_casestudy;

Output:

```
hive> create database if not exists hive_casestudy;
OK
Time taken: 1.123 seconds
hive> use hive_casestudy;
OK
```

Create an external table to load the data:

Command: *CREATE EXTERNAL TABLE IF NOT EXISTS sales_details(event_time timestamp,event_type string,product_id string,category_id string,category_code string,brand string,price float, user_id bigint,user_session string) ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'*
>STORED AS TEXTFILE LOCATION '/hivecasestudy'
>TBLPROPERTIES("skip.header.line.count"="1");

Output:

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS sales_details(event_time timestamp, ev
ent_type string, product_id string, category_id string, category_code string, br
and string, price float, user_id bigint, user_session string) ROW FORMAT SERDE '
org.apache.hadoop.hive.serde2.OpenCSVSerde'
    > STORED AS TEXTFILE LOCATION '/user/hivecasestudy/'
    > TBLPROPERTIES ("skip.header.line.count"="1");
OK
Time taken: 0.542 seconds
```

Checking the table 'sales_details':

Command: *desc sales_details;*

Output:

```
hive> desc sales_details;
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: 0.18 seconds, Fetched: 9 row(s)
```

Loading the data into the 'sales_details' table and checking:

Command: *load data inpath '/user/hivecasestudy/2019-Oct.csv' into table sales_details;*
load data inpath '/user/hivecasestudy/2019-Nov.csv' into table sales_details;

Output:

```
hive> load data inpath '/user/hivecasestudy/2019-Oct.csv' into table sales_details;
Loading data to table default.sales_details
OK
Time taken: 1.511 seconds
hive> load data inpath '/user/hivecasestudy/2019-Nov.csv' into table sales_details;
Loading data to table default.sales_details
OK
Time taken: 0.842 seconds
```

Command: *select * from sales_details*
>limit 5;

Output:

```
hive> select * from sales_details
    > limit 5;
OK
sales_details.event_time        sales_details.event_type        sales_details.product_i
d       sales_details.category_id       sales_details.category_code     sales_details.b
rand    sales_details.price     sales_details.user_id   sales_details.user_session
2019-11-01 00:00:02 UTC view    5802432 1487580009286598681                     0.32  5
62076640        09fafd6c-6c99-46b1-834f-33527f4de241
2019-11-01 00:00:09 UTC cart    5844397 1487580006317032337                     2.38  5
53329724        2067216c-31b5-455d-a1cc-af0575a34ffb
2019-11-01 00:00:10 UTC view    5837166 1783999064103190764             pnb     22.22 5
56138645        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: 0.632 seconds, Fetched: 5 row(s)
```

Hive Commands for easy view:

Commands: *set hive.cli.print.header=True;*
set hive.resultset.use.unique.column.names=false;
set hive.exec.dynamic.partition=true;
set hive.exec.dynamic.partition.mode=nonstrict;
set hive.enforce.bucketing=true;

Output:

```
hive> set hive.cli.print.header=True;
hive> set hive.resultset.use.unique.column.names=false;
hive> set hive.exec.dynamic.partition=true;
hive> set hive.exec.dynamic.partition.mode=nonstrict;
hive> set hive.enforce.bucketing=true;
```

Explanation:

- set hive.cli.print.header=True; - To display the column header in output
- set hive.resultset.use.unique.column.names=false; - To remove table name before column name separated by '.'
- set hive.exec.dynamic.partition=true;
- set hive.exec.dynamic.partition.mode=nonstrict; - To enable Dynamic partitioning
- set hive.enforce.bucketing=true; - To enable bucketing

4. Optimization of Tables – Partitioning and Bucketing:

Partitioning and Bucketing are query optimisation techniques. Bucketing is a method of segregating the tables or partitions in hive into parts based on the values of a column which in turn reduces the query execution time.

Attribute ‘event_type’ has unique set of categorical values and it will give us a definite number of partitions (4). So, partitioning will be performed on the attribute ‘event_type’. This will also reduce the load to the name node.

Bucketing will be performed on the attribute ‘price’. Depending on the data size and the size of HDFS block, the number of buckets will be created. We have calculated to create a minimum of eight buckets.

Command: *create external table cosmetics_par(event_time timestamp, product_id string,category_id string, category_code string, brand string, price float, user_id bigint,user_session string)*
> partitioned by (event_type string)
> clustered by (price) into 6 buckets
> row format serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
> stored as textfile location '/user/hivecasestudy/'
> tblproperties ("skip.header.line.count"="1");

Output:

```
hive> create external table cosmetics_par (event_time timestamp, product_id string, ca
tegory_id string, category_code string, brand string, price float, user_id bigint, use
r_session string)
    > partitioned by (event_type string)
    > clustered by (user_id) into 6 buckets
    > row format serde 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
    > stored as textfile location '/user/hivecasestudy/'
    > tblproperties ("skip.header.line.count"="1");
OK
Time taken: 0.307 seconds
```

Checking the new table cos_opt:

Command: *desc cosmetics_par;*

Output:

```
hive> desc cosmetics_par;
OK
event_time              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
event_type              string

# Partition Information
# col_name              data_type               comment

event_type              string
Time taken: 0.467 seconds, Fetched: 14 row(s)
```

Loading data into partitioned and bucketed table 'cosmetics_par' :

Command: *insert into table cosmetics_par partition(event_type)*
> select event_time, product_id, category_id, category_code, brand, price, user_id, user_session, event_type
> from sales_details;

Output:

```
hive> insert into table cosmetics_par partition(event_type)
    > select event_time, product_id, category_id, category_code, brand, price, user_id
, user_session, event_type
    > from sales_details;
Query ID = hadoop_20221201124638_b9a86587-1e00-4e08-959d-af9d0d3e660c
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_1669890043319_0016)

Map 1: -/-      Reducer 2: 0/9
Map 1: 0/4      Reducer 2: 0/9
Map 1: 0/4      Reducer 2: 0/9
Map 1: 0/4      Reducer 2: 0/9
Map 1: 0(+2)/4  Reducer 2: 0/9
Map 1: 0(+3)/4  Reducer 2: 0/9
Map 1: 0(+3)/4  Reducer 2: 0/9
Map 1: 0(+3)/4  Reducer 2: 0/9
Map 1: 0(+3)/4  Reducer 2: 0/9
Map 1: 0(+3)/4  Reducer 2: 0/9
Map 1: 0(+3)/4  Reducer 2: 0/9
Map 1: 0(+3)/4  Reducer 2: 0/9
Map 1: 0(+3)/4  Reducer 2: 0/9
Map 1: 0(+3)/4  Reducer 2: 0/9
Map 1: 0(+3)/4  Reducer 2: 0/9
Map 1: 0(+3)/4  Reducer 2: 0/9
Map 1: 0(+3)/4  Reducer 2: 0/9
Map 1: 0(+3)/4  Reducer 2: 0/9
Map 1: 0(+3)/4  Reducer 2: 0/9
```

```
Map 1: 4/4      Reducer 2: 4(+3)/9
Map 1: 4/4      Reducer 2: 4(+3)/9
Map 1: 4/4      Reducer 2: 4(+3)/9
Map 1: 4/4      Reducer 2: 4(+3)/9
Map 1: 4/4      Reducer 2: 5(+3)/9
Map 1: 4/4      Reducer 2: 5(+3)/9
Map 1: 4/4      Reducer 2: 5(+3)/9
Map 1: 4/4      Reducer 2: 5(+3)/9
Map 1: 4/4      Reducer 2: 5(+3)/9
Map 1: 4/4      Reducer 2: 5(+3)/9
Map 1: 4/4      Reducer 2: 5(+3)/9
Map 1: 4/4      Reducer 2: 5(+3)/9
Map 1: 4/4      Reducer 2: 6(+2)/9
Map 1: 4/4      Reducer 2: 6(+3)/9
Map 1: 4/4      Reducer 2: 6(+3)/9
Map 1: 4/4      Reducer 2: 6(+3)/9
Map 1: 4/4      Reducer 2: 6(+3)/9
Map 1: 4/4      Reducer 2: 7(+2)/9
Map 1: 4/4      Reducer 2: 7(+2)/9
Map 1: 4/4      Reducer 2: 7(+2)/9
Map 1: 4/4      Reducer 2: 7(+2)/9
Map 1: 4/4      Reducer 2: 8(+1)/9
Map 1: 4/4      Reducer 2: 8(+1)/9
Map 1: 4/4      Reducer 2: 9/9
Loading data to table hive_casestudy.cosmetics_par partition (event_type=null)


        Time taken to load dynamic partitions: 0.383 seconds
        Time taken for adding to write entity : 0.006 seconds
OK
```

Verifying partitioning in Hive:

Command: *show partitions cosmetics_par;*

Output:

```
hive> show partitions cosmetics_par;
OK
event_type=cart
event_type=purchase
event_type=remove_from_cart
event_type=view
Time taken: 0.144 seconds, Fetched: 4 row(s)
```

Verifying partitioning and bucketing in Hadoop:

Command: *hadoop fs -ls /user/hivecasestudy/*

Output:

```
[hadoop@ip-172-31-83-157 ~]$ hadoop fs -ls /user/hivecasestudy/
Found 4 items
drwxr-xr-x   - hadoop hadoop          0 2022-12-01 12:51 /user/hivecasestudy/event_typ
e=cart
drwxr-xr-x   - hadoop hadoop          0 2022-12-01 12:52 /user/hivecasestudy/event_typ
e=purchase
drwxr-xr-x   - hadoop hadoop          0 2022-12-01 12:52 /user/hivecasestudy/event_typ
e=remove_from_cart
drwxr-xr-x   - hadoop hadoop          0 2022-12-01 12:52 /user/hivecasestudy/event_typ
e=view
```

We can observe that four partitions have been created.

Command: *hadoop fs -ls /user/hivecasestudy/event_type=purchase*

Output:

```
[hadoop@ip-172-31-83-157 ~]$ hadoop fs -ls /user/hivecasestudy/event_type=purchase/
Found 6 items
-rwxr-xr-x   1 hadoop hadoop   23939488 2022-12-01 12:51 /user/hivecasestudy/event_typ
e=purchase/000000_0
-rwxr-xr-x   1 hadoop hadoop   23978398 2022-12-01 12:51 /user/hivecasestudy/event_typ
e=purchase/000001_0
-rwxr-xr-x   1 hadoop hadoop   23228108 2022-12-01 12:50 /user/hivecasestudy/event_typ
e=purchase/000002_0
-rwxr-xr-x   1 hadoop hadoop   23613588 2022-12-01 12:51 /user/hivecasestudy/event_typ
e=purchase/000003_0
-rwxr-xr-x   1 hadoop hadoop   24113362 2022-12-01 12:51 /user/hivecasestudy/event_typ
e=purchase/000004_0
-rwxr-xr-x   1 hadoop hadoop   24183396 2022-12-01 12:50 /user/hivecasestudy/event_typ
e=purchase/000005_0
```

From the output, we can observe that 6 buckets have been created with index starting from 0-5.

Command : *select * from cosmetics_par limit 5;*

Output:

```
hive> select * from cosmetics_par
    > limit 5;
OK
2019-10-06 20:46:50 UTC 5853630 1487580006317032337                    11.10   557505
362     dfee5efd-eb4a-4d7c-8dc9-0d976ff1ead8    cart
2019-10-08 09:48:05 UTC 5885214 2069804417665728971            cosmoprofi      5.56 5
58012670        a7a775fa-c671-43ae-928d-667eab5e6e48    cart
2019-10-01 07:00:05 UTC 5847446 1487580009286598681                    1.03    555511
769     d29c7b8f-041d-4cbb-acf3-67859a4cc5db    cart
2019-10-08 09:48:03 UTC 5885022 2069804417665728971            cosmoprofi      5.56 5
58012694        f0cda670-7691-41d0-b7d8-c445c0e34926    cart
2019-10-06 20:46:49 UTC 5853630 1487580006317032337                    11.10   557505
362     dfee5efd-eb4a-4d7c-8dc9-0d976ff1ead8    cart
Time taken: 3.334 seconds, Fetched: 5 row(s)
```

Now, we have created two tables namely, sales_details and cosmetics_par. We will be using 'cosmetics_par' table for analysing the data which is a partitioned and bucketed table.

```
hive> show tables;
OK
cosmetics_par
sales_details
Time taken: 0.059 seconds, Fetched: 2 row(s)
```

5. Querying – Assignment Questions:

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

Command: *SELECT ROUND(SUM(price),2) AS total_sales*
> FROM cosmetics_par
> WHERE event_type='purchase' AND
> month(event_time)=10;

Output

```
hive> SELECT ROUND(SUM(price),2) AS total_sales
    > FROM cosmetics_par
    > WHERE event_type='purchase' AND
    > month(event_time)=10;
Query ID = hadoop_20221201181907_2c4cda9e-3169-4036-8e6d-4ca117f872fd
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1669909606668_0012)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      3          3        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 24.80 s
----------------------------------------------------------------------------------------------
OK
total_sales
1211534.31
Time taken: 25.387 seconds, Fetched: 1 row(s)
```

- The total revenue generated in the month of October is Rs.1211534.31

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

Command: *SELECT Month(event_time) as Month, ROUND(SUM(price),2) as total_purchases*
>FROM cosmetics_par
>WHERE event_type='purchase'
>GROUP BY month(event_time);

Output:

```
hive> SELECT month(event_time) as month, ROUND(SUM(price),2) AS total_purchases
    > FROM cosmetics_par
    > WHERE event_type='purchase'
    > GROUP BY month(event_time);
Query ID = hadoop_20221201171315_af38aabd-820a-4376-90fb-eb057953f474
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1669909606668_0010)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      3          3        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 25.97 s
--------------------------------------------------------------------------------
OK
month   total_purchases
10      1211534.31
11      1531007.31
Time taken: 26.668 seconds, Fetched: 2 row(s)
```

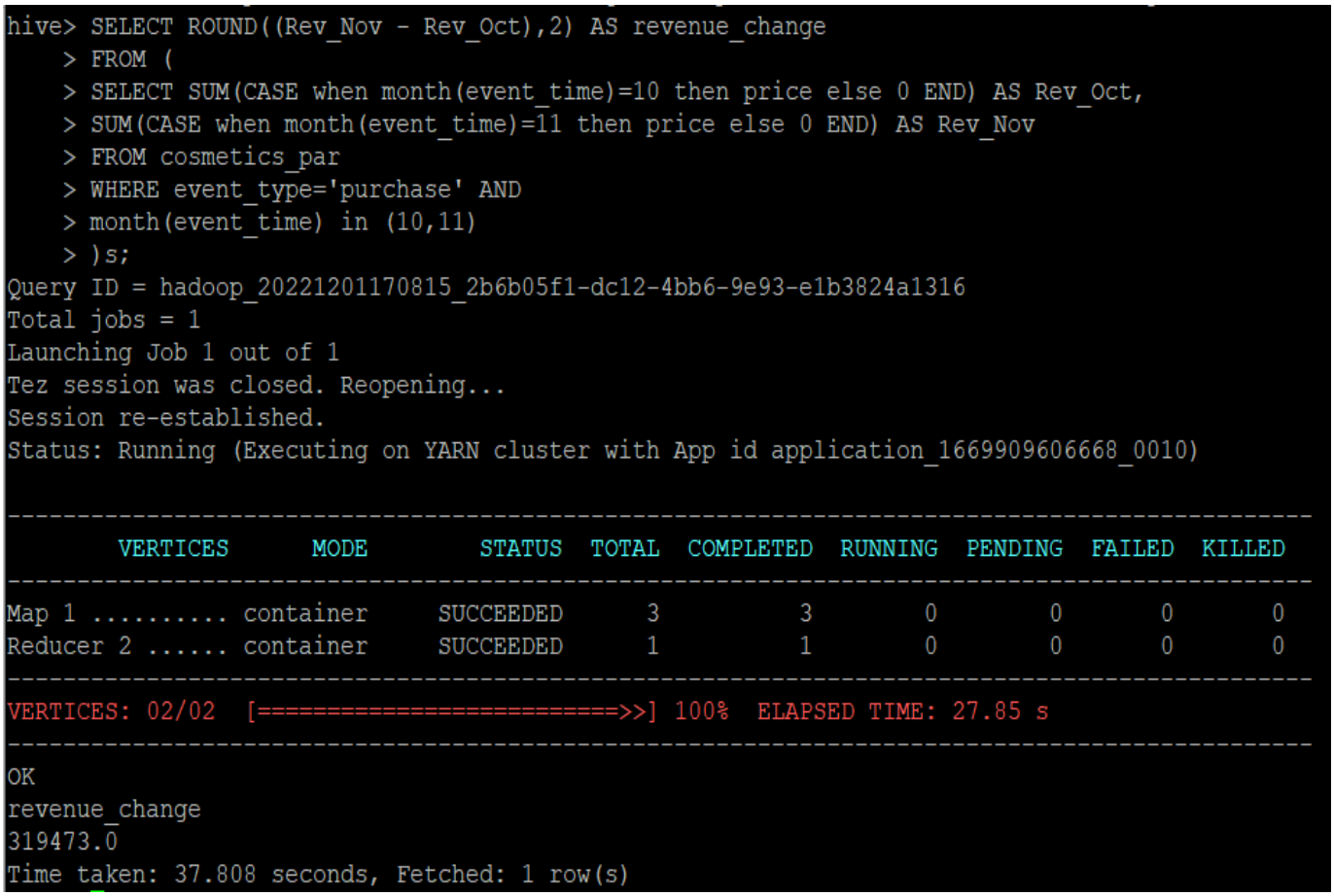
- In the month of October, the total purchases is Rs.1211534.31
- In the month of November, the total purchases is Rs.1531007.31

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

Command: *SELECT ROUND((Rev_Nov – Rev_Oct),2) AS revenue_change*
>FROM (
>SELECT SUM (CASE WHEN month(event_time)=10 THEN price ELSE 0 END) AS Rev_Oct,
> SUM (CASE WHEN month(event_time)=11 THEN price ELSE 0

END) AS Rev_Nov,
>FROM cosmetics_par,
>WHERE event_type='purchase' AND
>month(event_time) in (10,11)
>)s;

Output:



- From the output, we can infer that the difference in revenue is Rs. 3,19,473.00

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

Command: *SELECT DISTINCT SPLIT (category_code, '\\.')[0] AS Distinct_category_list*
>FROM cosmetics_par;

Output:

```
hive> SELECT DISTINCT SPLIT(category_code,'\\.')[0] AS Distinct_category_list
    > FROM cosmetics_par;
Query ID = hadoop_20221201171658_87fa2462-8b39-4361-ac9f-1fdf5851b5f7
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1669909606668_0010)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      6          6        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      5          5        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 74.57 s
--------------------------------------------------------------------------------
OK
distinct_category_list

furniture
appliances
accessories
apparel
sport
stationery
Time taken: 75.222 seconds, Fetched: 7 row(s)
```

- We can observe that there are 6 distinct categories namely, furniture, appliances, accessories, apparel, sport and stationery.

v. Find the total number of products available under each category.

Command: *SELECT SPLIT (category_code, '\\.')[0] AS category, COUNT(product_id) AS total_prod*
>FROM cosmetics_par
>GROUP BY SPLIT(category_code, '\\.')[0]
>ORDER BY total_prod;

Output:

```
hive> SELECT SPLIT(category_code,'\\.')[0] AS category, COUNT(product_id) AS total_prod
    > FROM cosmetics_par
    > GROUP BY SPLIT(category_code,'\\.')[0]
    > ORDER BY total_prod;
Query ID = hadoop_20221201172308_d41edd06-4e6b-4b40-9b0c-5252c4285efd
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1669909606668_0010)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      6          6        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      5          5        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 74.83 s
--------------------------------------------------------------------------------
OK
category        total_prod
sport   2
accessories     12929
apparel 18232
furniture       23604
stationery      26722
appliances      61736
        8594871
Time taken: 75.845 seconds, Fetched: 7 row(s)
```

The total number of products under each category are as follows:

- Sport : 2
- Accessories : 12929
- Apparel : 18232
- Furniture : 23604
- Stationery : 26722
- Appliances : 61736

vi. Which brand had the maximum sales in October and November combined?

Command: *SELECT brand, ROUND(SUM(price),2) AS sales_total*
>FROM cosmetics_par
>WHERE event_type= 'purchase'
>GROUP BY brand
> ORDER BY sales_total DESC
>LIMIT 5;

Output:

```
hive> SELECT brand,ROUND(SUM(price),2) as sales_total
    > FROM cosmetics_par
    > WHERE event_type='purchase'
    > GROUP BY brand
    > ORDER BY sales_total DESC
    > LIMIT 5;
Query ID = hadoop_20221201173150_1986cdf5-d678-408e-b7b6-66e6c8ca9de5
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1669909606668_0010)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      3          3        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: 23.76 s
--------------------------------------------------------------------------------
OK
brand   sales_total
        1094179.77
runail  148297.94
grattol 106918.25
irisk   92538.0
uno     86341.78
Time taken: 24.547 seconds, Fetched: 5 row(s)
```

- From the output, we can observe that 'Runail' is a popular brand with high sales in both months. The total sales is Rs.148297.94

vii. Which brands increased their sales from October to November?

Command: *WITH sales_total AS (*
>SELECT brand,
>SUM (CASE WHEN month(event_time)=10 THEN price ELSE 0 END) AS oct_sales,
>SUM (CASE WHEN month(event_time)=11 THEN price ELSE 0 END) AS nov_sales
>FROM cosmetics_par
>WHERE event_type = 'purchase' AND month(event_time) in (10,11)
>GROUP BY brand)
>SELECT brand, ROUND(oct_sales,2), ROUND(nov_sales,2), ROUND((nov_sales-oct_sales),2) AS increased_sales
>FROM sales_total
>WHERE (nov_sales-oct_sales)>0
>ORDER BY increased_sales DESC;

Output:

```
hive> WITH sales_total AS (
    > SELECT brand,
    > SUM (CASE WHEN month(event_time)=10 THEN price ELSE 0 END) AS oct_sales,
    > SUM (CASE WHEN month(event_time)=11 THEN price ELSE 0 END) AS nov_sales
    > FROM cosmetics_par
    > WHERE event_type='purchase' AND
    > month(event_time) in (10,11)
    > GROUP BY brand
    > )
    > SELECT brand, ROUND(oct_sales,2), ROUND(nov_sales,2), ROUND((nov_sales-oct_sales),2) AS increased_
sales
    > FROM sales_total
    > WHERE (nov_sales-oct_sales)>0
    > ORDER BY increased_sales DESC;
Query ID = hadoop_20221201175202_0fb2b8e6-a47c-4960-8afa-0183562c6b88
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_1669909606668_0011)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      3          3        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: 34.66 s
--------------------------------------------------------------------------------
OK
brand   _c1     _c2     increased_sales
        474677.31       619502.46       144825.15
grattol 35445.54        71472.71        36027.17
uno     35302.03        51039.75        15737.72
lianail 5892.84 16394.24        10501.4
ingarden        23161.39        33566.21        10404.82
strong  29196.63        38671.27        9474.64
jessnail        26287.84        33345.23        7057.39
cosmoprofi      8322.81 14536.99        6214.18
polarus 6013.72 11371.93        5358.21
runail  71539.28        76758.66        5219.38
freedecor       3421.78 7671.8  4250.02
staleks 8519.73 11875.61        3355.88
bpw.style       11572.15        14836.36        3264.21
```

```
staleks 8519.73 11875.61        3355.88
bpw.style       11572.15        14836.36        3264.21
lovely  8704.38 11939.06        3234.68
marathon        7280.75 10273.1 2992.35
haruyama        9390.69 12352.91        2962.22
yoko    8756.91 11707.88        2950.97
italwax 21940.24        24799.37        2859.13
benovy  409.62  3259.97 2850.35
kaypro  881.34  3268.7  2387.36
estel   21756.75        24142.67        2385.92
concept 11032.14        13380.4 2348.26
kapous  11927.16        14093.08        2165.92
f.o.x   6624.23 8577.28 1953.05
masura  31263.71        33056.74        1793.03
milv    3904.94 5642.01 1737.07
beautix 10493.95        12222.95        1729.0
artex   2730.64 4327.25 1596.61
domix   10472.05        12009.17        1537.12
shik    3341.2  4839.72 1498.52
smart   4457.26 5902.14 1444.88
roubloff        3491.36 4913.77 1422.41
levrana 2243.56 3664.1  1420.54
oniq    8425.41 9841.65 1416.24
irisk   45591.96        46946.04        1354.08
severina        4775.88 6120.48 1344.6
joico   705.52  2015.1  1309.58
zeitun  708.66  2009.63 1300.97
beauty-free     554.17  1782.86 1228.69
swarovski       1887.93 3043.16 1155.23
de.lux  1659.7  2775.51 1115.81
metzger 5373.45 6457.16 1083.71
markell 1768.75 2834.43 1065.68
sanoto  157.14  1209.68 1052.54
nagaraku        4369.74 5327.68 957.94
ecolab  262.85  1214.3  951.45
art-visage      2092.71 2997.8  905.09
levissime       2227.5  3085.31 857.81
missha  1293.83 2150.28 856.45
solomeya        1899.7  2685.8  786.1
rosi    3077.04 3841.56 764.52
refectocil      2716.18 3475.58 759.4
kaaral  4412.43 5086.07 673.64
kosmekka        1181.44 1813.37 631.93
kinetics        6334.25 6945.26 611.01
browxenna       14331.37        14916.73        585.36
airnails        5118.9  5691.52 572.62
```

```
kosmekka        1181.44 1813.37 631.93
kinetics        6334.25 6945.26 611.01
browxenna       14331.37        14916.73        585.36
airnails        5118.9  5691.52 572.62
uskusi  5142.27 5690.31 548.04
coifin  903.0   1428.49 525.49
s.care  412.68  913.07  500.39
limoni  1308.9  1796.6  487.7
matrix  3243.25 3726.74 483.49
gehwol  1089.07 1557.68 468.61
greymy  29.21   489.49  460.28
bioaqua 942.89  1398.12 455.23
farmavita       837.37  1291.97 454.6
sophin  1067.86 1515.52 447.66
yu-r    271.41  673.71  402.3
kiss    421.55  817.33  395.78
naomi   0.0     389.0   389.0
lador   2083.61 2471.53 387.92
ellips  245.85  606.04  360.19
jas     3318.96 3657.43 338.47
lowence 242.84  567.75  324.91
nitrile 847.28  1162.68 315.4
shary   871.96  1176.49 304.53
kims    330.04  632.04  302.0
happyfons       801.92  1091.59 289.67
kocostar        310.85  594.93  284.08
insight 1443.7  1721.96 278.26
candy   534.96  799.38  264.42
bluesky 10307.24        10565.53        258.29
beauugreen      511.51  768.35  256.84
protokeratin    201.25  456.79  255.54
trind   298.07  542.96  244.89
entity  479.71  719.26  239.55
skinlite        651.94  890.45  238.51
provoc  827.99  1063.82 235.83
fedua   52.38   263.81  211.43
ecocraft        41.16   241.95  200.79
keen    236.35  435.62  199.27
mane    66.79   260.26  193.47
freshbubble     318.7   502.34  183.64
matreshka       0.0     182.67  182.67
chi     358.94  538.61  179.67
cristalinas     427.63  584.95  157.32
farmona 1692.46 1843.43 150.97
```

cristalinas	427.63	584.95	157.32
farmona	1692.46	1843.43	150.97
latinoil	249.52	384.59	135.07
miskin	158.04	293.07	135.03
elizavecca	70.53	204.3	133.77
nefertiti	233.52	366.64	133.12
finish	98.38	230.38	132.0
igrobeauty	513.66	645.07	131.41
dizao	819.13	945.51	126.38
osmo	645.58	762.31	116.73
batiste	772.4	874.17	101.77
carmex	145.08	243.36	98.28
eos	54.34	152.61	98.27
depilflax	2707.07	2803.78	96.71
enjoy	41.35	136.57	95.22
kerasys	430.91	525.2	94.29
aura	83.95	177.51	93.56
plazan	101.37	194.01	92.64
koelf	422.73	507.29	84.56
nirvel	163.04	234.33	71.29
konad	739.83	810.67	70.84
egomania	77.47	146.04	68.57
cutrin	299.37	367.62	68.25
laboratorium	246.5	312.52	66.02
inm	288.02	351.21	63.19
dewal	0.0	61.29	61.29
marutaka-foot	49.22	109.33	60.11
kares	0.0	59.45	59.45
profhenna	679.23	736.85	57.62
koelcia	55.5	112.75	57.25
balbcare	155.33	212.38	57.05
elskin	251.09	307.65	56.56
foamie	35.04	80.49	45.45
ladykin	125.65	170.57	44.92
likato	296.06	340.97	44.91
mavala	409.04	446.32	37.28
vilenta	197.6	231.21	33.61
beautyblender	78.74	109.41	30.67
biore	60.65	90.31	29.66
orly	902.38	931.09	28.71

```
ladykin 125.65  170.57  44.92
likato  296.06  340.97  44.91
mavala  409.04  446.32  37.28
vilenta 197.6   231.21  33.61
beautyblender   78.74   109.41  30.67
biore   60.65   90.31   29.66
orly    902.38  931.09  28.71
estelare        444.81  471.87  27.06
profepil        93.36   118.02  24.66
blixz   38.95   63.4    24.45
binacil 0.0     24.26   24.26
godefroy        401.22  425.12  23.9
glysolid        69.73   91.59   21.86
veraclara       50.11   71.21   21.1
juno    0.0     21.08   21.08
kamill  63.01   81.49   18.48
treaclemoon     163.37  181.49  18.12
supertan        50.37   66.51   16.14
barbie  0.0     12.39   12.39
deoproce        316.84  329.17  12.33
rasyan  18.8    28.94   10.14
fly     17.14   27.17   10.03
tertio  236.16  245.8   9.64
jaguar  1102.11 1110.65 8.54
soleo   204.2   212.53  8.33
neoleor 43.41   51.7    8.29
moyou   5.71    10.28   4.57
bodyton 1376.34 1380.64 4.3
skinity 8.88    12.44   3.56
helloganic      0.0     3.1     3.1
grace   100.92  102.61  1.69
cosima  20.23   20.93   0.7
ovale   2.54    3.1     0.56
Time taken: 45.29 seconds, Fetched: 161 row(s)
```

- From the output, we can observe that 161 brands increased their sales from the month of October to November.
- ‘Grattol’ brand has the highest per month increase of Rs. 36027.17
- ‘Ovale’ brand has the lowest per month increase of Rs. 0.56

viii. 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.

Command: *SELECT user_id, ROUND(SUM(price),2) AS spent_money*
>FROM cosmetics_par
>WHERE event_type= 'purchase'
>GROUP BY user_id
>ORDER BY spent_money DESC
>LIMIT 10;

Output:

```
hive> SELECT user_id, ROUND(SUM(price),2) AS spent_money
    > FROM cosmetics_par
    > WHERE event_type='purchase'
    > GROUP BY user_id
    > ORDER BY spent_money DESC
    > LIMIT 10;
Query ID = hadoop_20221201173522_9d4770b6-1149-458a-a367-16c45b53c2f3
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1669909606668_0010)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      3          3        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: 27.07 s
----------------------------------------------------------------------------------------------
OK
user_id spent_money
557790271       2715.87
150318419       1645.97
562167663       1352.85
531900924       1329.45
557850743       1295.48
522130011       1185.39
561592095       1109.7
431950134       1097.59
566576008       1056.36
521347209       1040.91
Time taken: 27.757 seconds, Fetched: 10 row(s)
```

The top 10 users of the website are as follows:

- 557790271 : 2715.87
- 150318419 : 1645.97
- 562167663 : 1352.85
- 531900924 : 1329:45

- 557850743 : 1295.48
- 522130011 : 1185.39
- 561592095 : 1109.7
- 431950134 : 1097.59
- 566576008 : 1056.36
- 521347209 : 1040.91

6. Improvement of performance after using optimization on tables

The execution time of query are reduced by using Hive optimization techniques. We had created an optimized table “cosmetics_par” through partitioning and bucketing in the start of the case study.

We will now observe the importance of optimization techniques, by querying two questions using the table “sales_details” and “cosmetics_par”. We have selected Question 6 & Question 8 for the demonstration.

Q1. Write a query to yield the total sum of purchases per month in a single output.

Command: *SELECT Month(event_time) as Month, ROUND(SUM(price),2) as total_purchases*
>FROM sales_details
>WHERE event_type=’purchase’
>GROUP BY month(event_time);

Output:

```
hive> SELECT month(event_time) AS month, ROUND(SUM(price),2) AS total_purchases
    > FROM sales_details
    > WHERE event_type='purchase'
    > GROUP BY month(event_time);
Query ID = hadoop_20221201180819_e42f05ee-3fc6-48d6-b082-525e955a6a3a
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1669909606668_0012)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      2          2        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      3          3        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 59.43 s
--------------------------------------------------------------------------------
OK
month   total_purchases
10      1211538.43
11      1531016.9
Time taken: 60.192 seconds, Fetched: 2 row(s)
hive>
```

Command: *SELECT Month(event_time) as Month, ROUND(SUM(price),2) as total_purchases*
>FROM cosmetics_par
>WHERE event_type=’purchase’
>GROUP BY month(event_time); (Optimised querying)

Output:

```
hive> SELECT month(event_time) as month, ROUND(SUM(price),2) AS total_purchases
    > FROM cosmetics_par
    > WHERE event_type='purchase'
    > GROUP BY month(event_time);
Query ID = hadoop_20221201171315_af38aabd-820a-4376-90fb-eb057953f474
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1669909606668_0010)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      3          3        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 25.97 s
--------------------------------------------------------------------------------
OK
month   total_purchases
10      1211534.31
11      1531007.31
Time taken: 26.668 seconds, Fetched: 2 row(s)
```

- We can observe that the results are similar for both queries but the query execution time differs.
- We see the query execution time for non optimized query is 60.192 seconds. We observed that, the query execution time for optimized query is 26.668 seconds. This is a huge difference, as the data size increase the increase in time will be substantially higher.

Q2. 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.

Command: *SELECT user_id, ROUND(SUM(price),2) AS spent_money*
>FROM sales_details
>WHERE event_type= 'purchase'
>GROUP BY user_id
>ORDER BY spent_money DESC
>LIMIT 10;

Output:

```
hive> SELECT user_id, ROUND(SUM(price),2) AS spent_money
    > FROM sales_details
    > WHERE event_type='purchase'
    > GROUP BY user_id
    > ORDER BY spent_money DESC
    > LIMIT 10;
Query ID = hadoop_20221201175852_2da3a69c-bd1a-4902-a346-4d77986518dc
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_1669909606668_0012)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      2          2        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      3          3        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 62.68 s
--------------------------------------------------------------------------------
OK
user_id spent_money
557790271       2715.87
150318419       1645.97
562167663       1352.85
531900924       1329.45
557850743       1295.48
522130011       1185.39
561592095       1109.7
431950134       1097.59
566576008       1056.36
521347209       1040.91
Time taken: 71.187 seconds, Fetched: 10 row(s)
```

Command: *SELECT user_id, ROUND(SUM(price),2) AS spent_money*
>FROM cosmetics_par
>WHERE event_type= 'purchase'
>GROUP BY user_id
>ORDER BY spent_money DESC
>LIMIT 10; (Optimized query)

Output:

```
hive> SELECT user_id, ROUND(SUM(price),2) AS spent_money
    > FROM cosmetics_par
    > WHERE event_type='purchase'
    > GROUP BY user_id
    > ORDER BY spent_money DESC
    > LIMIT 10;
Query ID = hadoop_20221201173522_9d4770b6-1149-458a-a367-16c45b53c2f3
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1669909606668_0010)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      3          3        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: 27.07 s
--------------------------------------------------------------------------------
OK
user_id spent_money
557790271       2715.87
150318419       1645.97
562167663       1352.85
531900924       1329.45
557850743       1295.48
522130011       1185.39
561592095       1109.7
431950134       1097.59
566576008       1056.36
521347209       1040.91
Time taken: 27.757 seconds, Fetched: 10 row(s)
```

- We can observe that the results are similar for both queries but the query execution time differs.
- We see the query execution time for non optimized query is 71.187 seconds. We observed that, the query execution time for optimized query is 27.757 seconds.

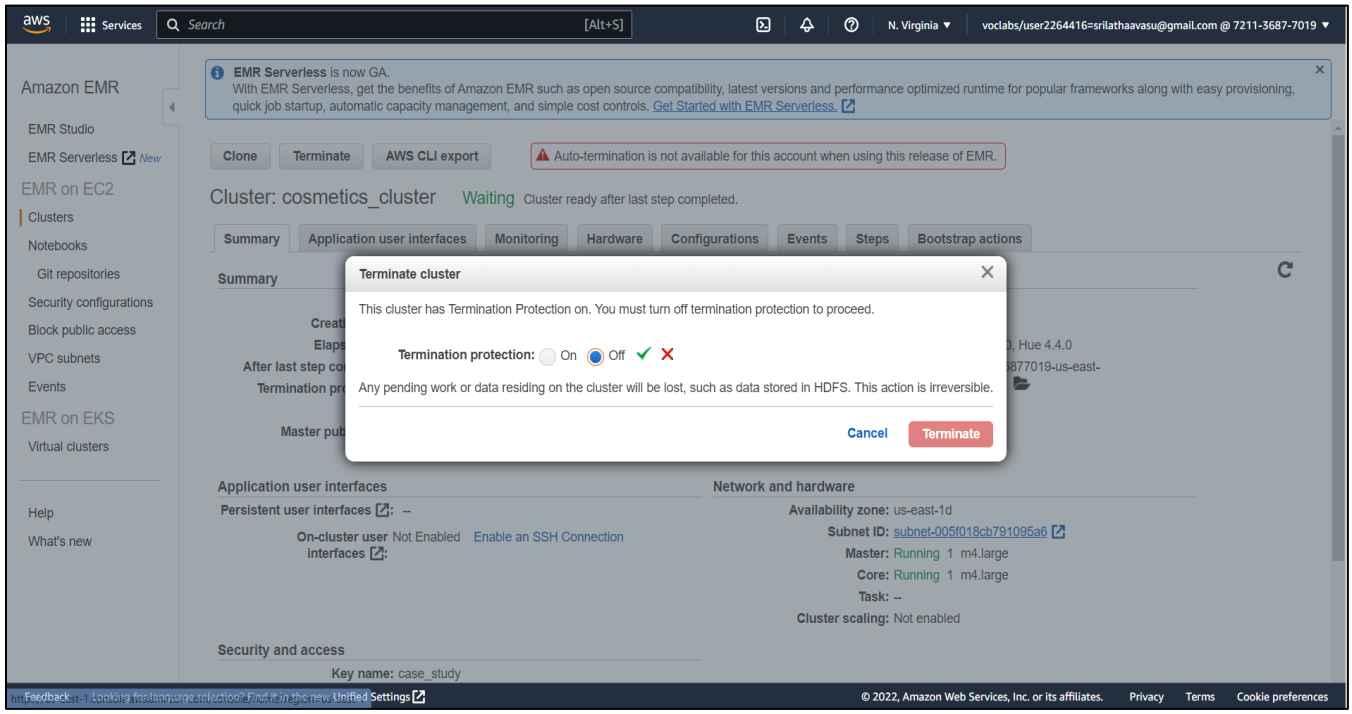
This is a huge difference, as the data size increase the increase in time will be substantially higher.

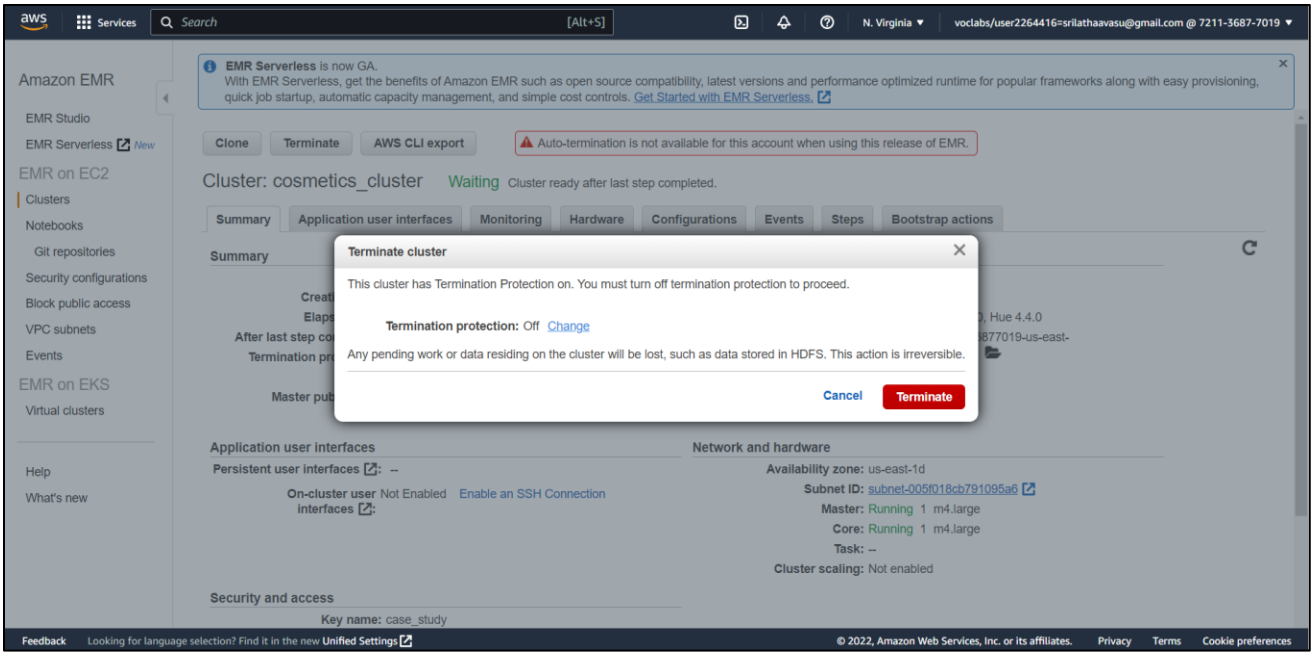
- Thus, partitioning and bucketing plays a crucial role to reduce the query execution time.

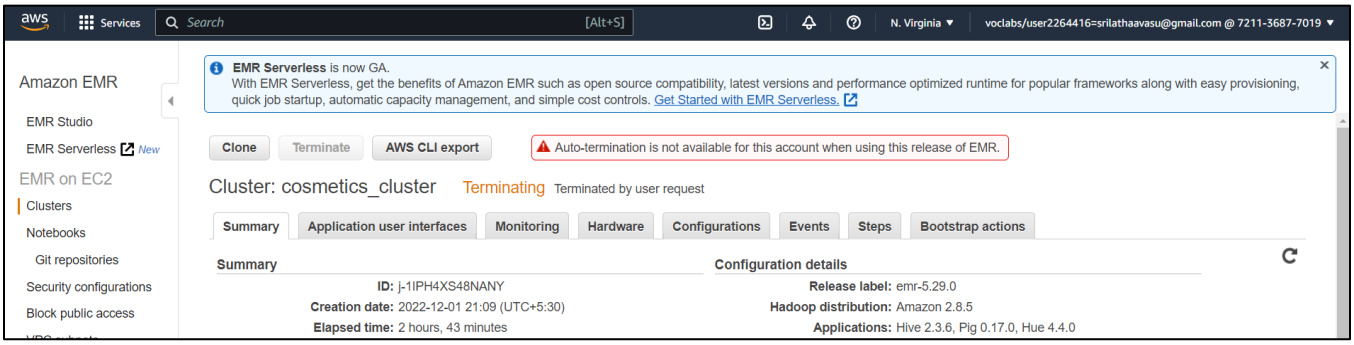
7. Cleaning-up :

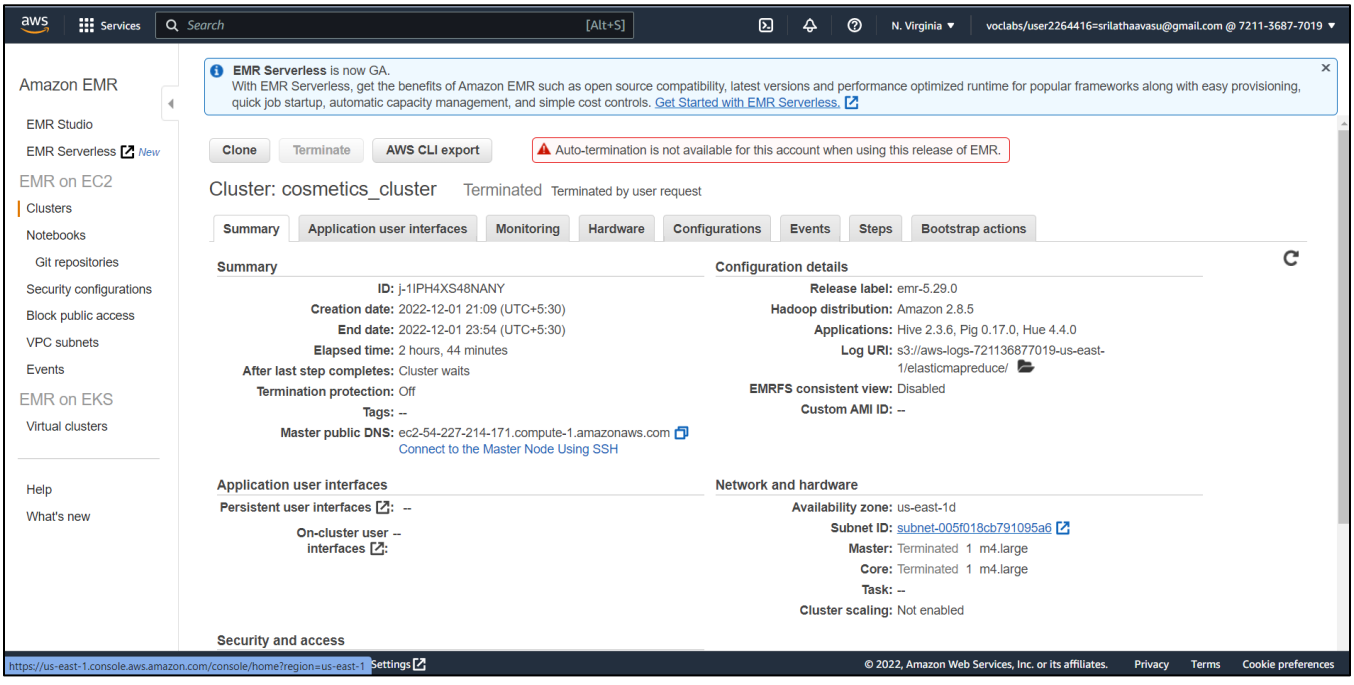
Once we are done with the analysis, we can drop the databases and quit hive and then terminate the EMR Cluster.

```
hive> show databases;
OK
default
hive_casestudy
Time taken: 0.03 seconds, Fetched: 2 row(s)
hive> drop database hive_casestudy CASCADE;
OK
Time taken: 0.356 seconds
hive> show databases;
OK
default
Time taken: 0.017 seconds, Fetched: 1 row(s)
```




Amazon EMR
EMR Studio
EMR Serverless New
EMR on EC2
Clusters
Notebooks
Git repositories
Security configurations
Block public access
VPC subnets
Events
EMR on EKS
Virtual clusters
Help
What's new
EMR Serverless is now GA.
With EMR Serverless, get the benefits of Amazon EMR such as open source compatibility, latest versions and performance optimized runtime for popular frameworks along with easy provisioning, quick job startup, automatic capacity management, and simple cost controls. Get Started with EMR Serverless.
Clone
Terminate
AWS CLI export
Auto-termination is not available for this account when using this release of EMR.
Cluster: cosmetics_cluster
Waiting
Cluster ready after last step completed.
Summary
Application user interfaces
Monitoring
Hardware
Configurations
Events
Steps
Bootstrap actions
Terminate cluster
This cluster has Termination Protection on. You must turn off termination protection to proceed.
Termination protection: Off Change
Any pending work or data residing on the cluster will be lost, such as data stored in HDFS. This action is irreversible.
Cancel
Terminate
Application user interfaces
Persistent user interfaces: --
On-cluster user interfaces: Not Enabled Enable an SSH Connection
Network and hardware
Availability zone: us-east-1d
Subnet ID: subnet-005f018cb791095a6
Master: Running 1 m4.large
Core: Running 1 m4.large
Task: --
Cluster scaling: Not enabled
Security and access
Key name: case_study
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Amazon EMR
EMR Studio
EMR Serverless New
EMR on EC2
Clusters
Notebooks
Git repositories
Security configurations
Block public access
EMR Serverless is now GA.
With EMR Serverless, get the benefits of Amazon EMR such as open source compatibility, latest versions and performance optimized runtime for popular frameworks along with easy provisioning, quick job startup, automatic capacity management, and simple cost controls. Get Started with EMR Serverless.
Clone
Terminate
AWS CLI export
Auto-termination is not available for this account when using this release of EMR.
Cluster: cosmetics_cluster
Terminating
Terminated by user request
Summary
Application user interfaces
Monitoring
Hardware
Configurations
Events
Steps
Bootstrap actions
Summary
ID: j-1IPH4XS48NANY
Creation date: 2022-12-01 21:09 (UTC+5:30)
Elapsed time: 2 hours, 43 minutes
Configuration details
Release label: emr-5.29.0
Hadoop distribution: Amazon 2.8.5
Applications: Hive 2.3.6, Pig 0.17.0, Hue 4.4.0


Amazon EMR
EMR Studio
EMR Serverless New
EMR on EC2
Clusters
Notebooks
Git repositories
Security configurations
Block public access
VPC subnets
Events
EMR on EKS
Virtual clusters
Help
What's new
EMR Serverless is now GA.
With EMR Serverless, get the benefits of Amazon EMR such as open source compatibility, latest versions and performance optimized runtime for popular frameworks along with easy provisioning, quick job startup, automatic capacity management, and simple cost controls. Get Started with EMR Serverless.
Clone
Terminate
AWS CLI export
Auto-termination is not available for this account when using this release of EMR.
Cluster: cosmetics_cluster
Terminated
Terminated by user request
Summary
Application user interfaces
Monitoring
Hardware
Configurations
Events
Steps
Bootstrap actions
Summary
ID: j-1IPH4XS48NANY
Creation date: 2022-12-01 21:09 (UTC+5:30)
End date: 2022-12-01 23:54 (UTC+5:30)
Elapsed time: 2 hours, 44 minutes
After last step completes: Cluster waits
Termination protection: Off
Tags: --
Master public DNS: ec2-54-227-214-171.compute-1.amazonaws.com
Connect to the Master Node Using SSH
Configuration details
Release label: emr-5.29.0
Hadoop distribution: Amazon 2.8.5
Applications: Hive 2.3.6, Pig 0.17.0, Hue 4.4.0
Log URI: s3://aws-logs-721136877019-us-east-1/elasticmapreduce/
EMRFS consistent view: Disabled
Custom AMI ID: --
Application user interfaces
Persistent user interfaces: --
On-cluster user interfaces: --
Network and hardware
Availability zone: us-east-1d
Subnet ID: subnet-005f018cb791095a6
Master: Terminated 1 m4.large
Core: Terminated 1 m4.large
Task: --
Cluster scaling: Not enabled
Security and access
https://us-east-1.console.aws.amazon.com/console/home?region=us-east-1
Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences