Hive Case Study

Samanyu Ghose

Problem Statement

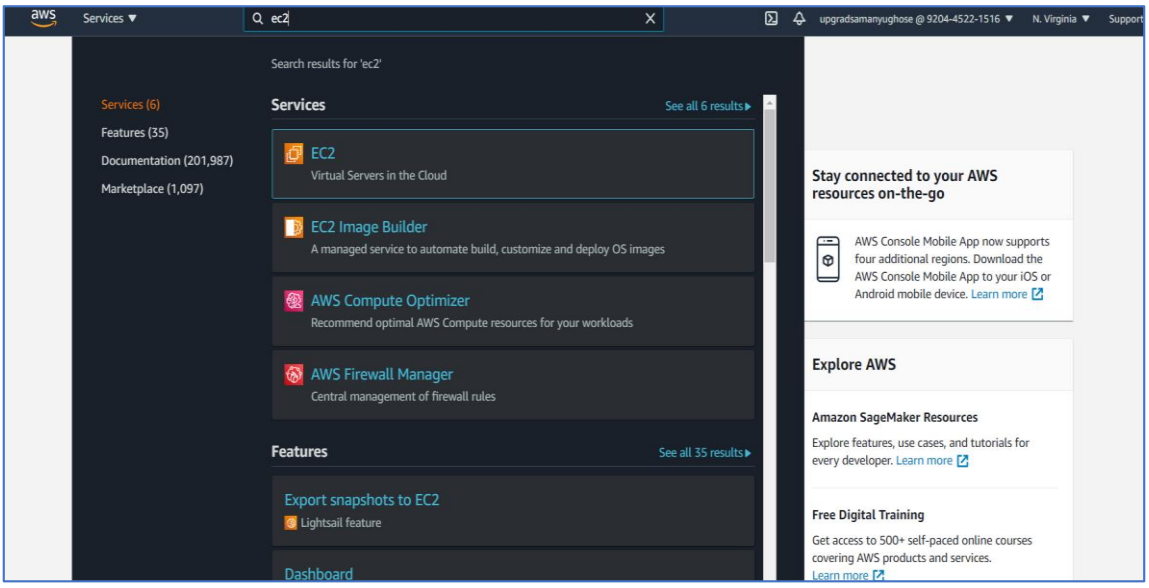
With online sales gaining popularity, tech companies are exploring ways to improve their sales by analyzing customer behavior and gaining insights about product trends. Furthermore, the websites make it easier for customers to find the products that they require without much scavenging. Goal is to, extract data and gather insights from a real-life data set of an e-commerce company.

1. Find the total revenue generated due to purchases made in October.
2. Write a query to yield the total sum of purchases per month in a single output.
3. Write a query to find the change in revenue generated due to purchases from October to November.
4. Find distinct categories of products.
5. Find the total number of products available under each category.
6. Which brand had the maximum sales in October and November combined?
7. Which brands increased their sales from October to November?
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.

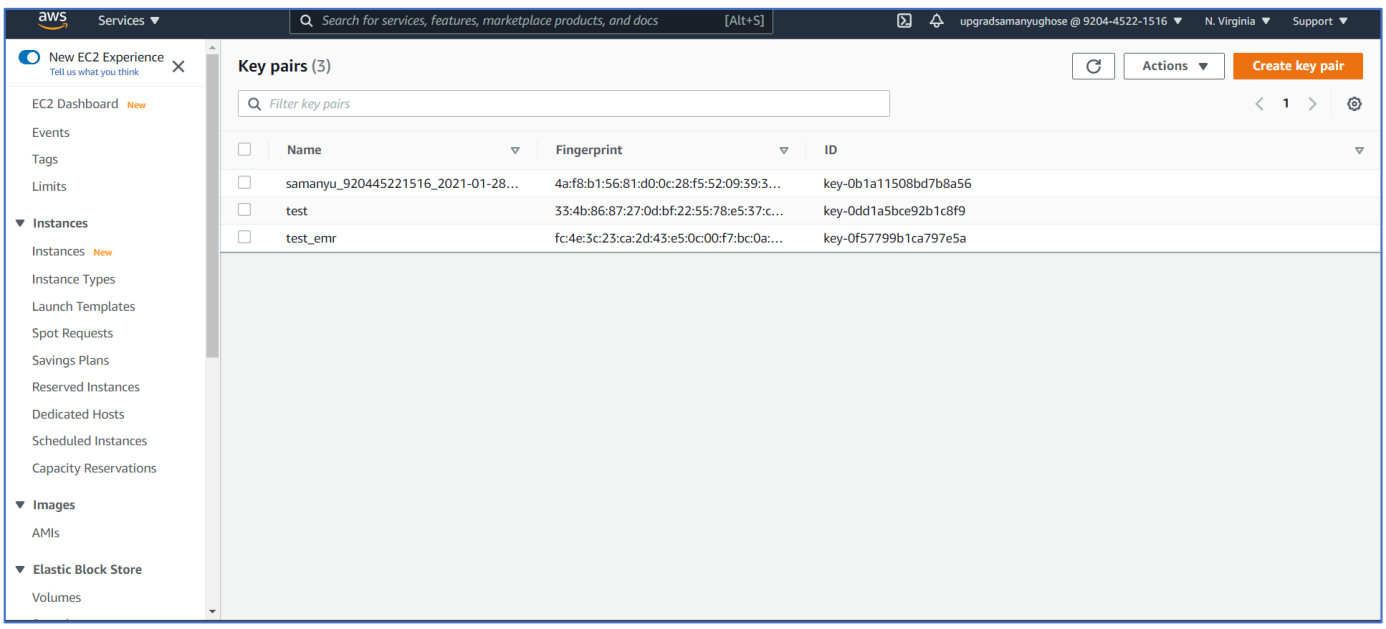
Launching EMR Cluster
Created a 2 node cluster which comprises of following configurations:
1. EMR version – 5.29.0
2. Services selected – Hadoop 2.8.5, Hue 4.4.0 and Hive 2.3.6
3. Selected 1 instance of each, master node & core node of instance type m4.large
4. Selected an appropriate key – pair to SSH into the master node of Amazon EMR cluster

Steps to create EC2, key-pair, Create cluster, load PuttyGen and open Putty command prompt :-

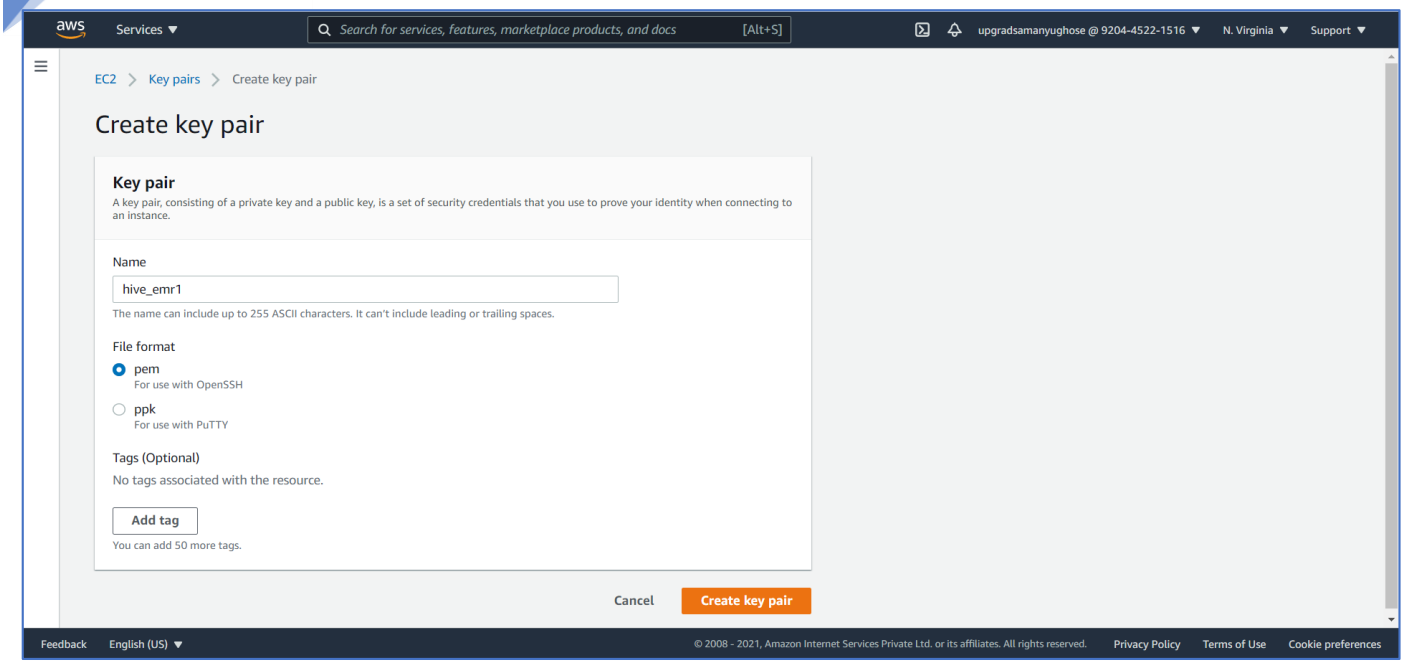
1. First open AWS console and click on EC2



2. Click on key-pair to create a key-pair

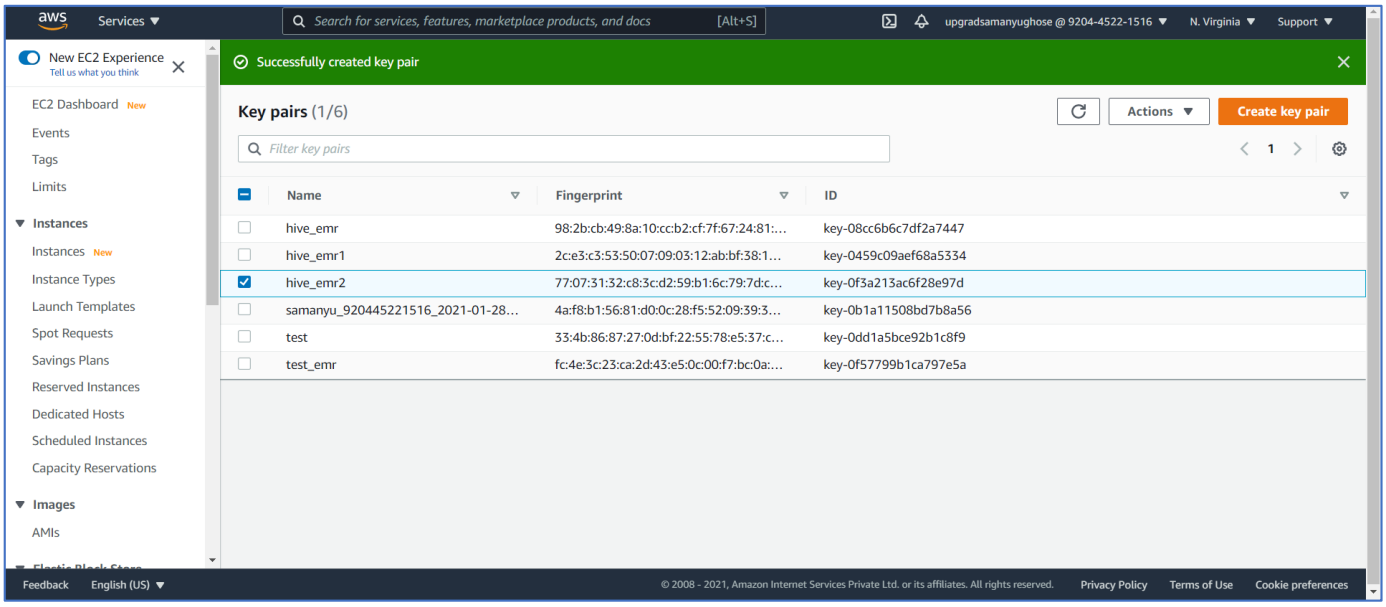


Created a key-pair by the name hive_emr and the file format is .pem file and the file will be downloaded

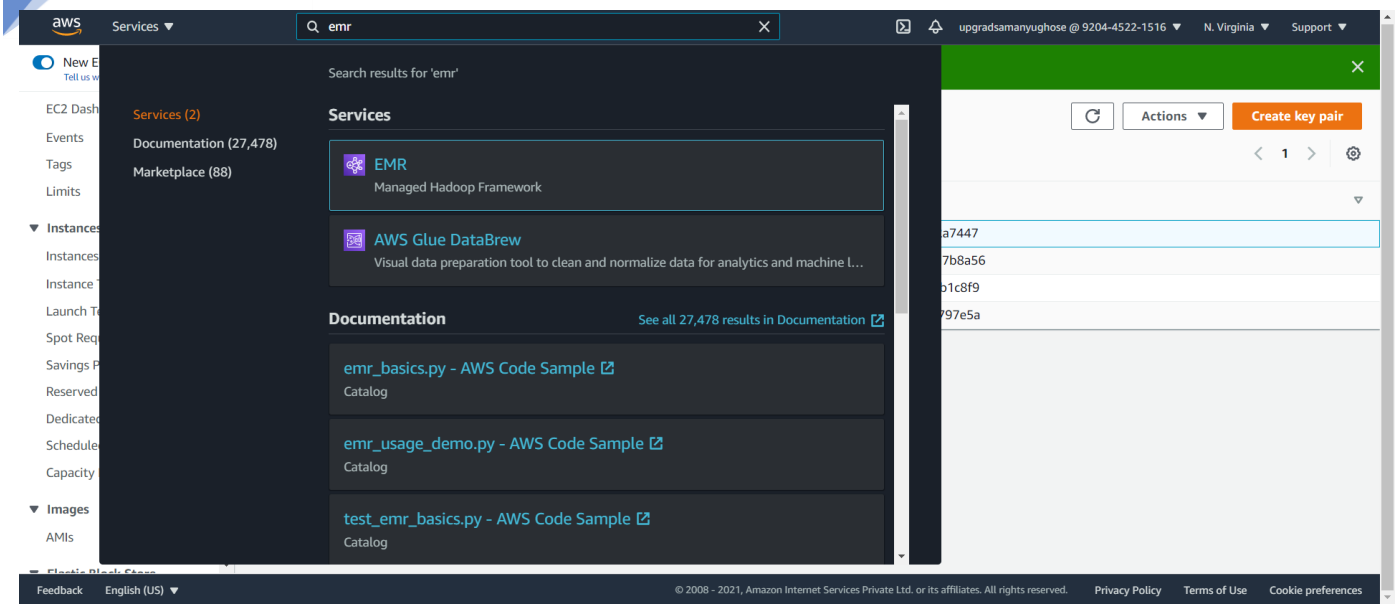


Click on ‘Create key-pair’

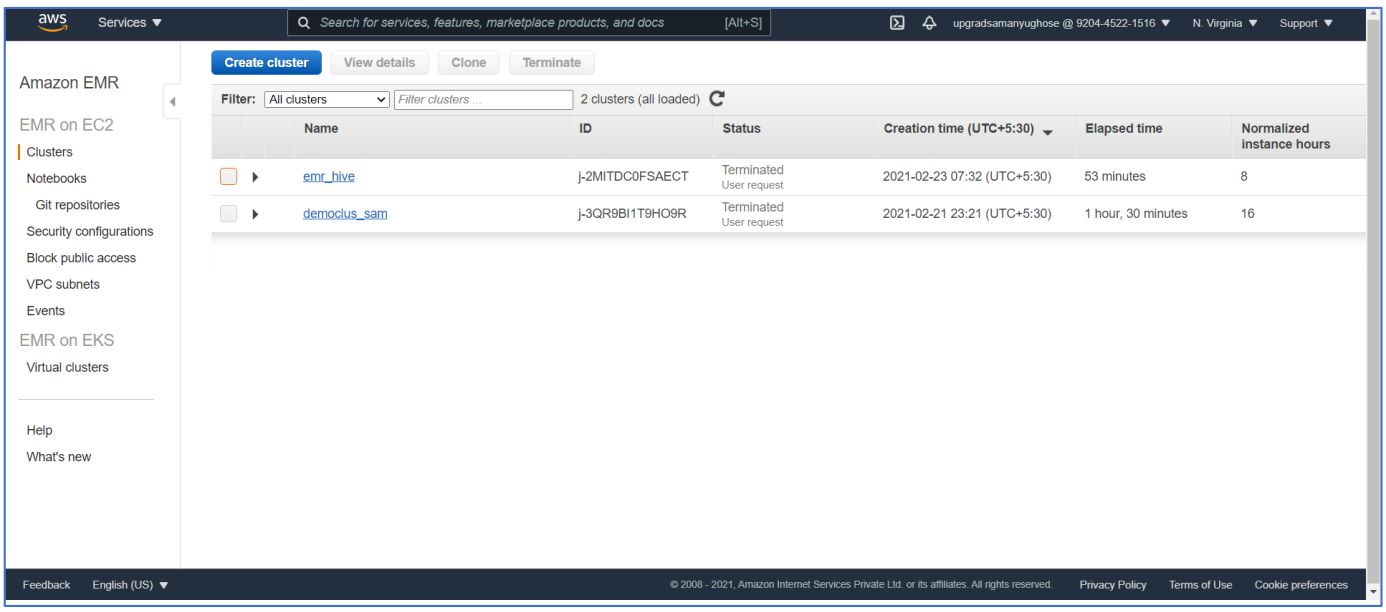
Now we can see that key-pair has been created



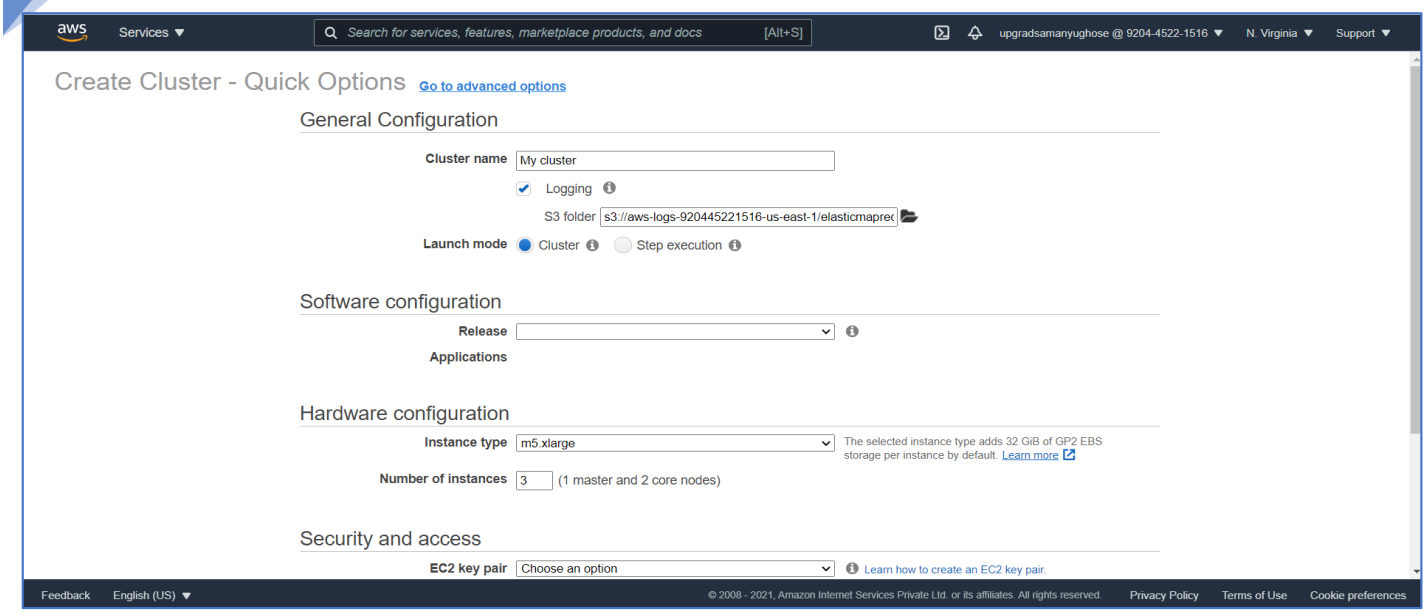
1. Now we will goto AWS console and type EMR and create EMR cluster



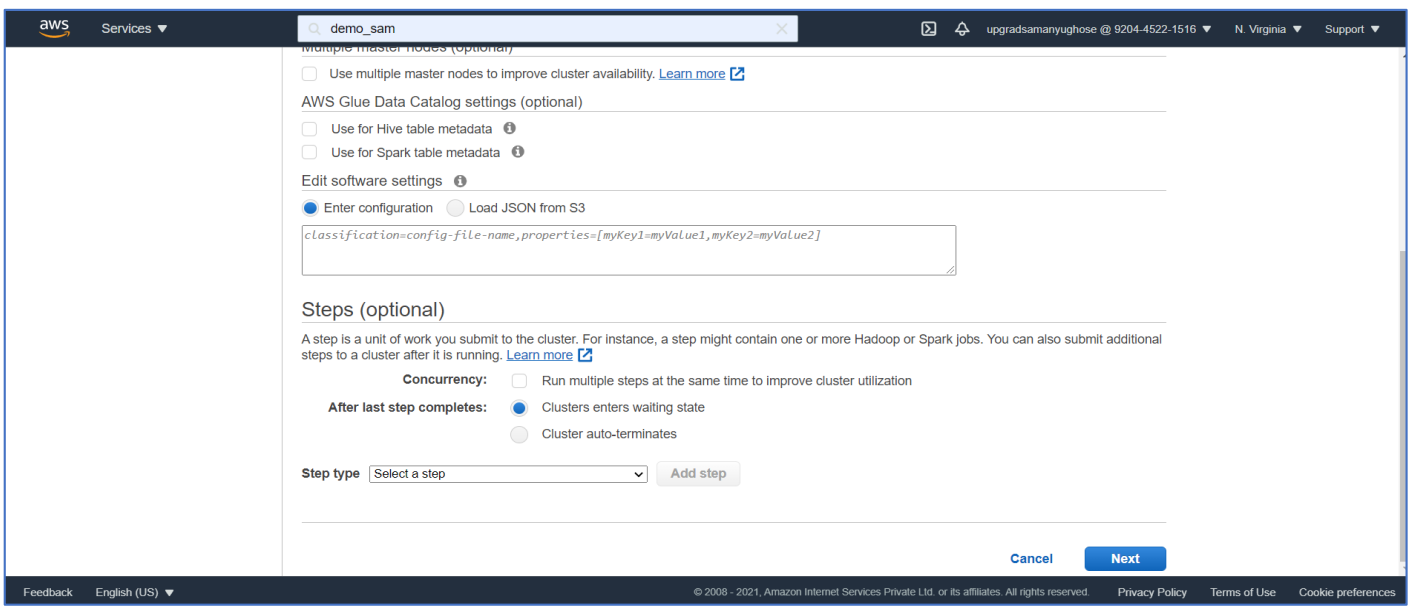
Then we will create cluster by clicking on 'Create cluster'



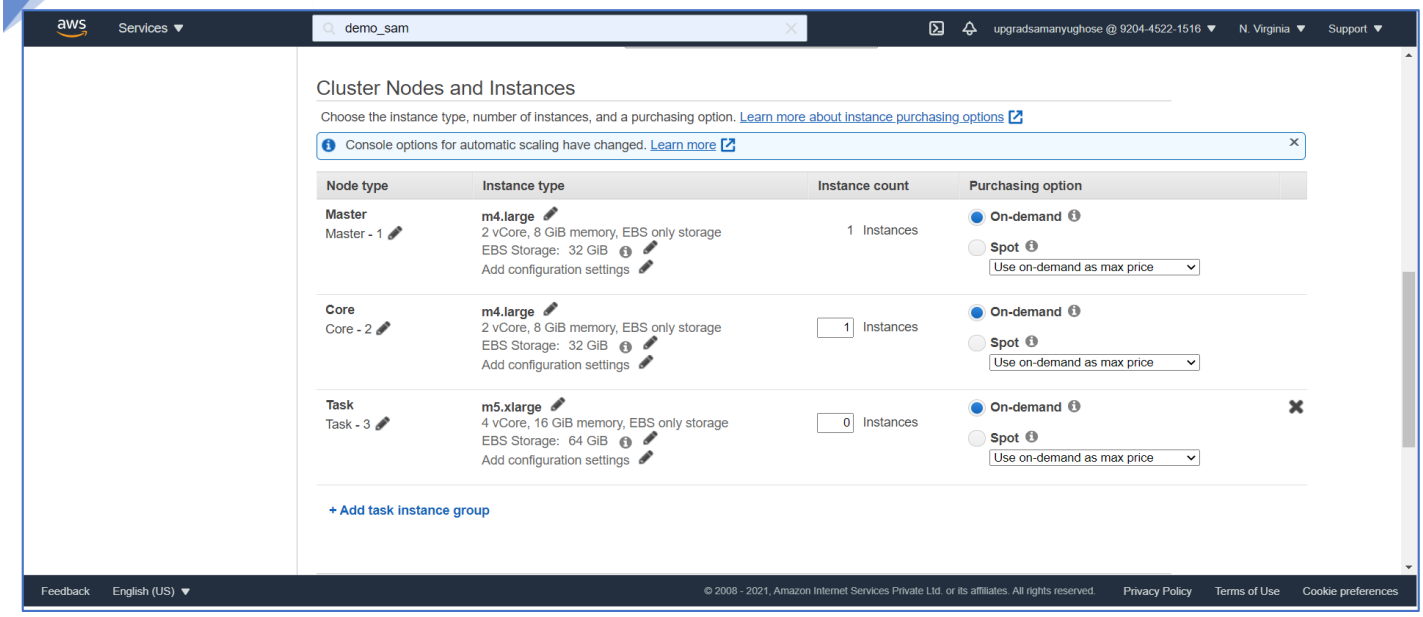
After clicking on 'Create Cluster',goto advanced options

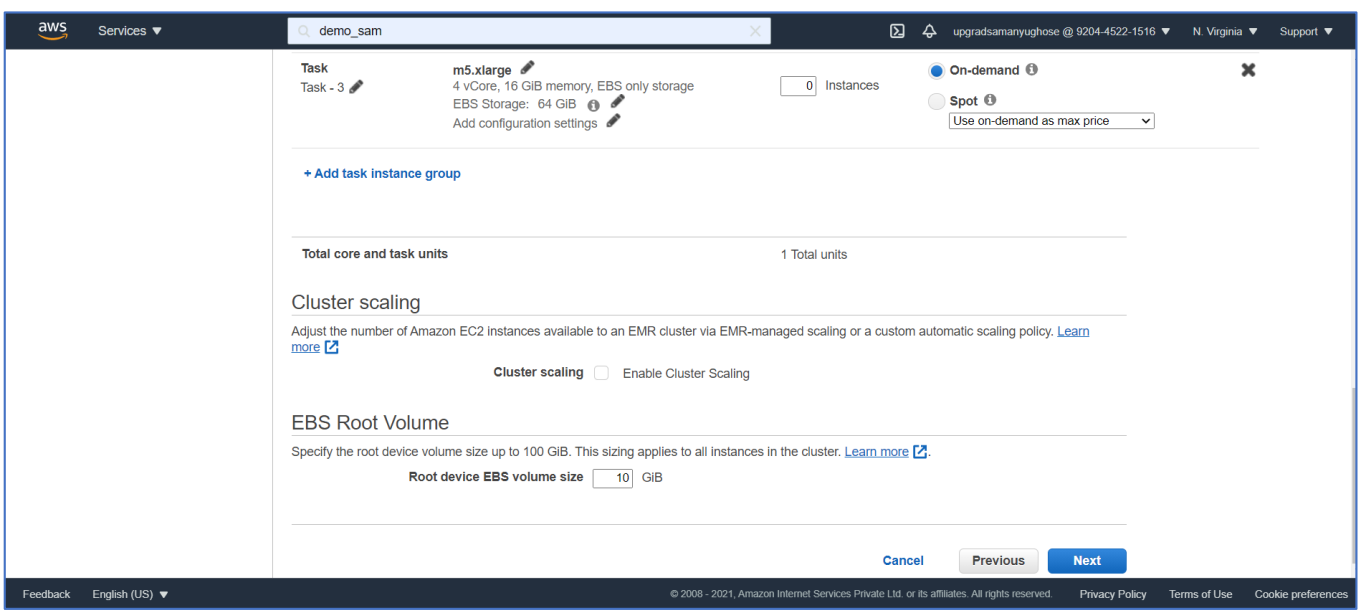


Choose emr-5.29.0 and click on Next

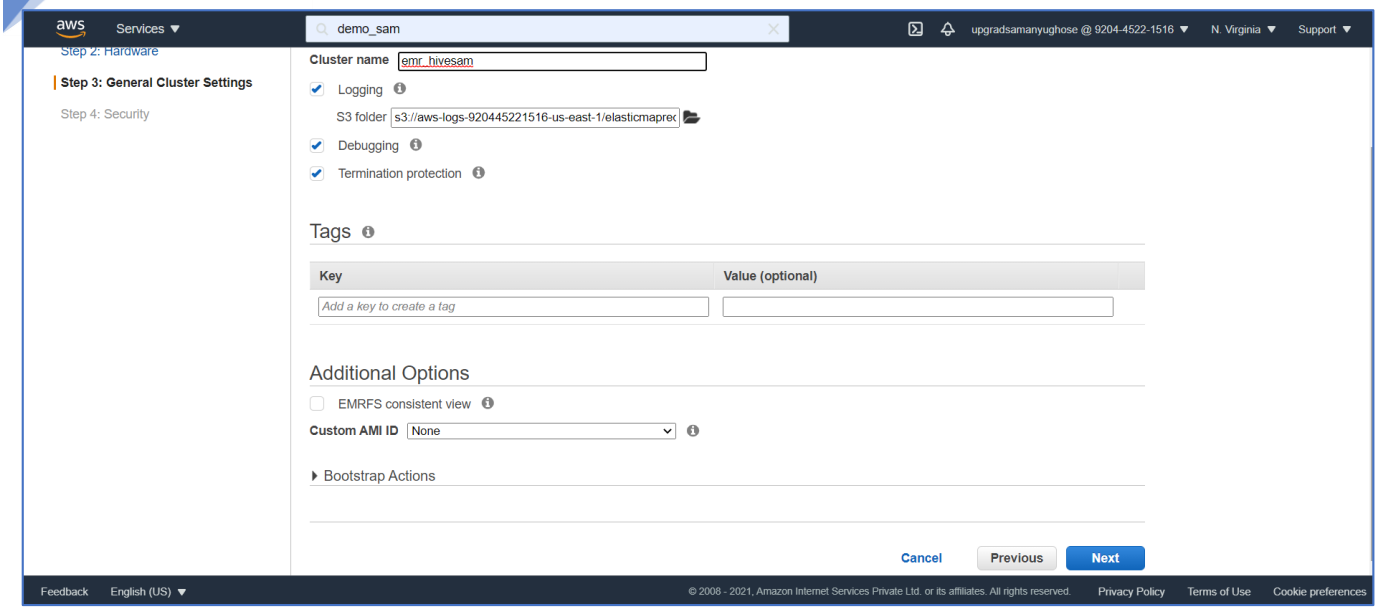


Choose Master and Core instance as 1 and instance type as m4.large in the next screen

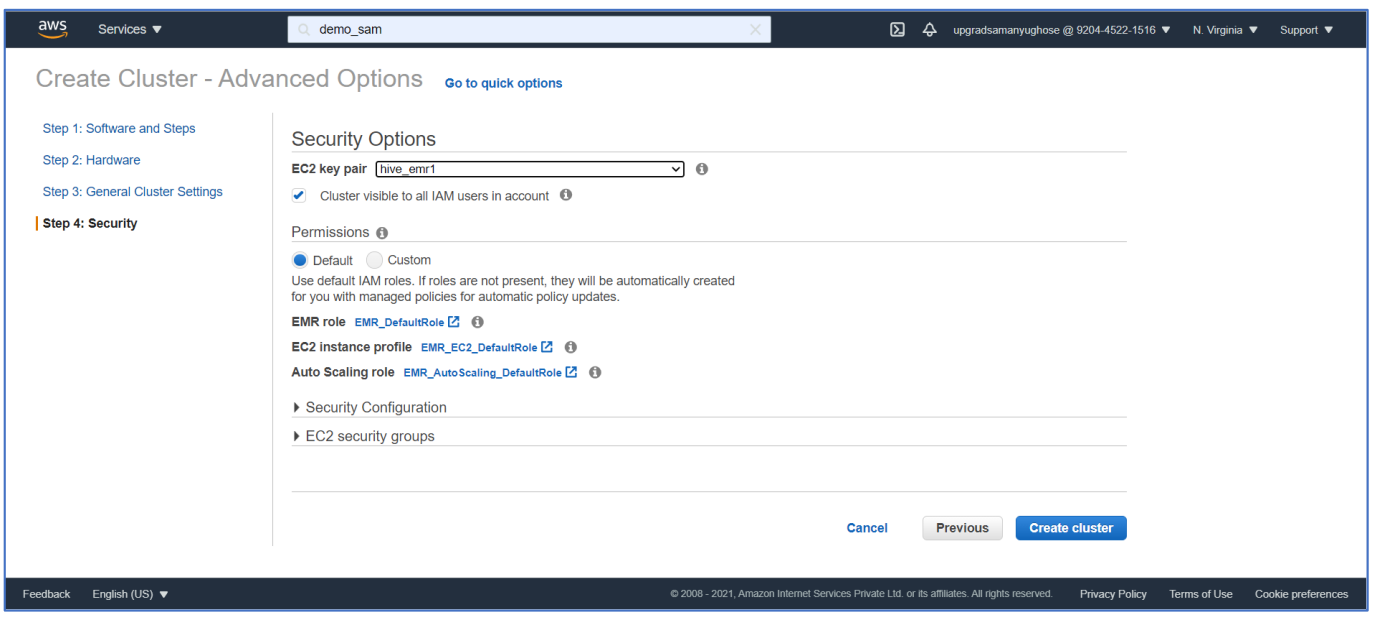




Type cluster name as ‘emr_hivesam’

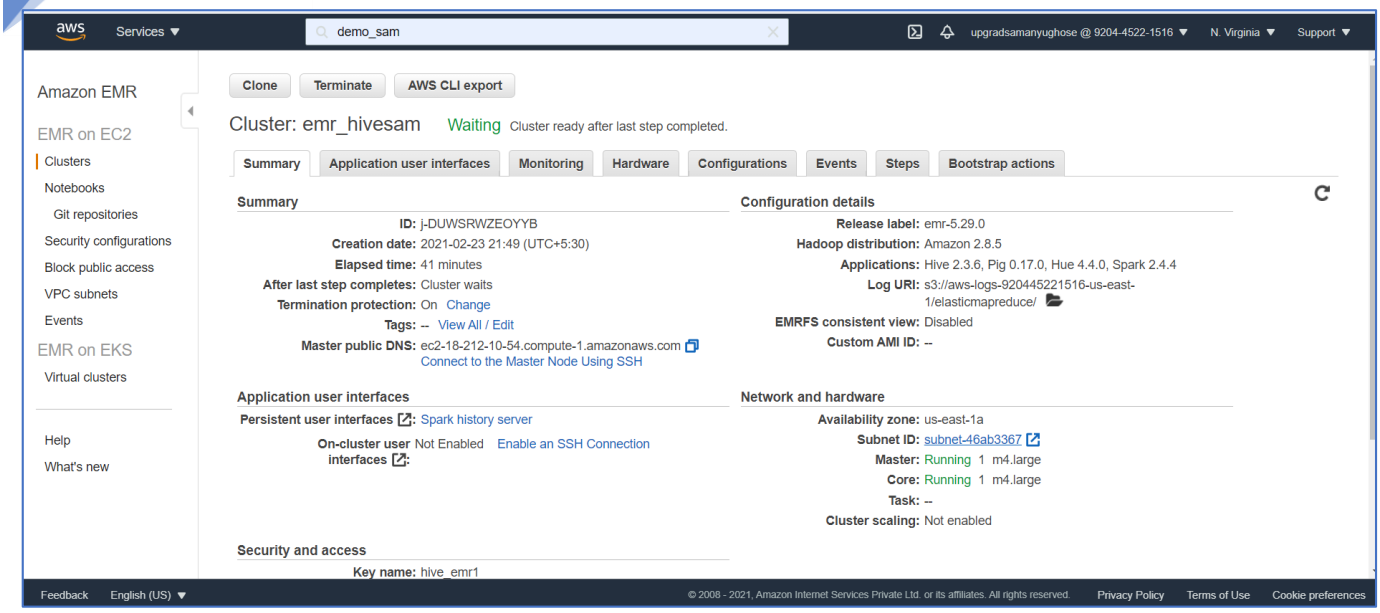


Select EC2 key pair as ‘hive_emr1’ which we have just created

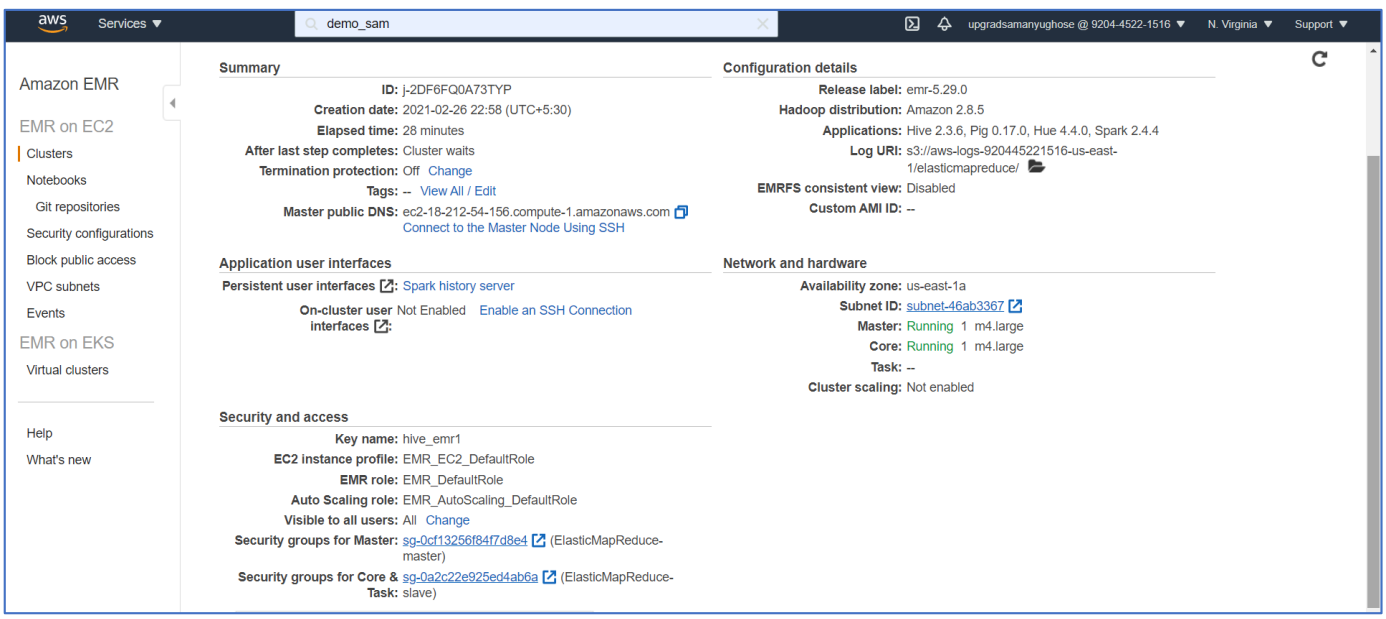


First we will set security for newly created EMR (‘emr_hivesam’) and setting inbound rules for SSH as ‘anywhere’ so that we can use at any number of times when we will accessing the EMR cluster for the 2nd time.

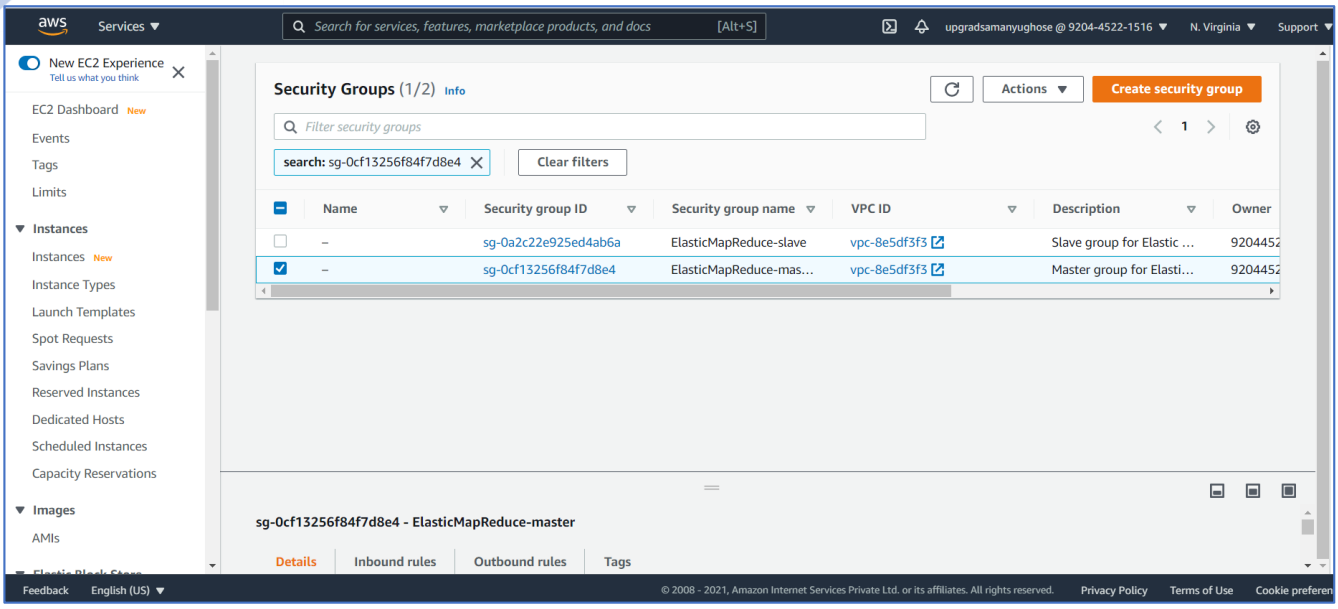
As soon as the cluster status shows ‘waiting’, we will start with the inbound security setting of security rules

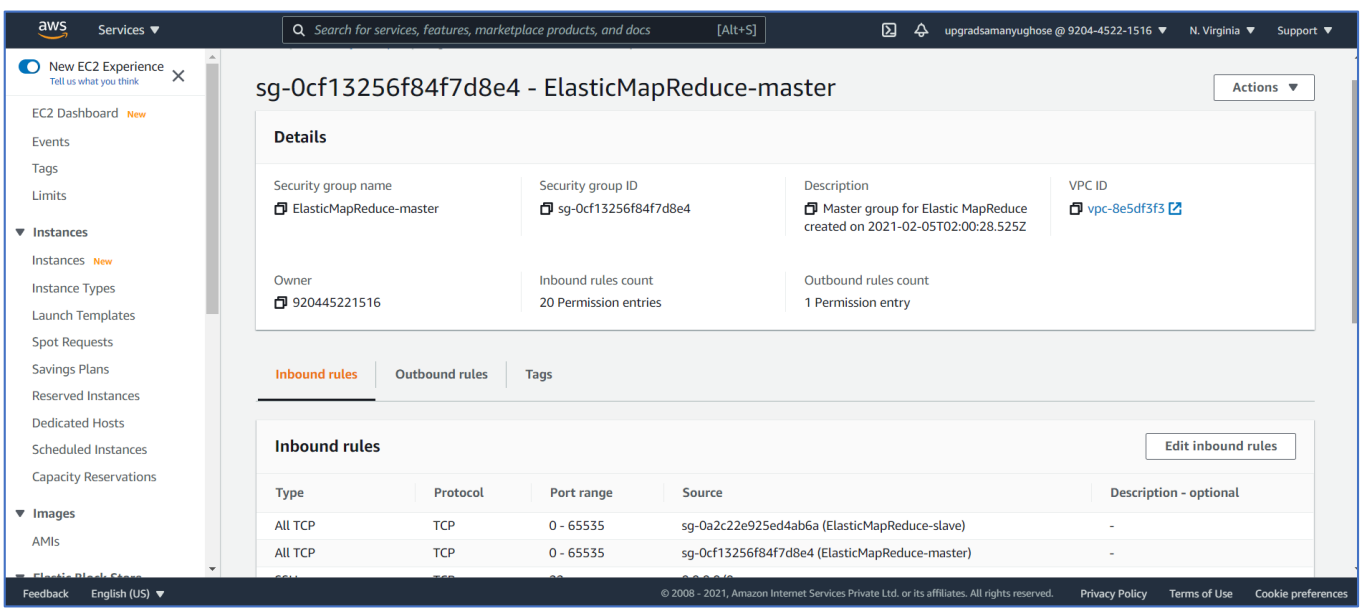


Click on security group for Master

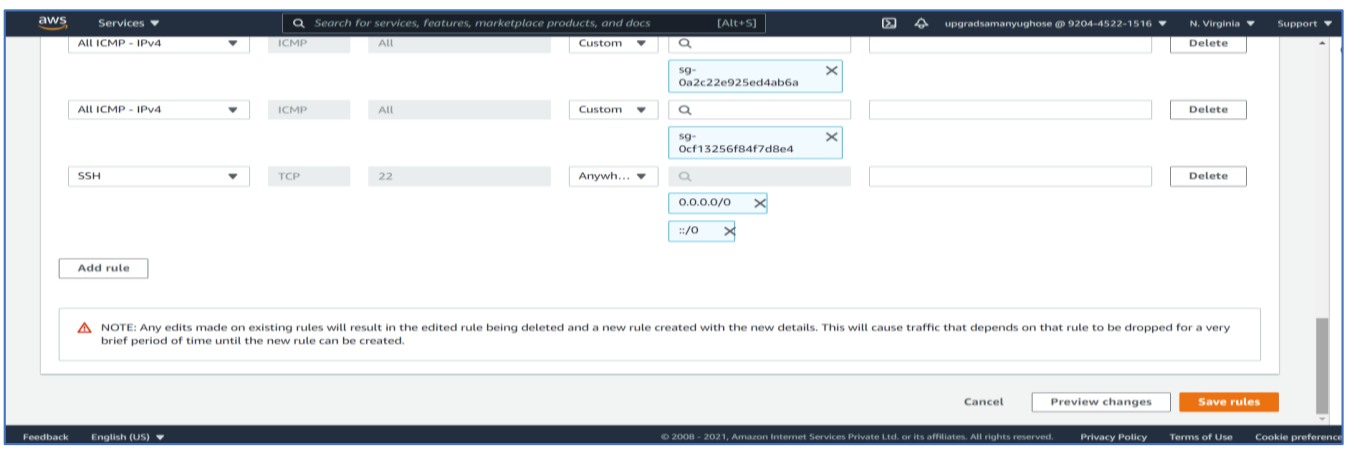


Click on security groups for Master and click on security ID for Master and edit inbound rules

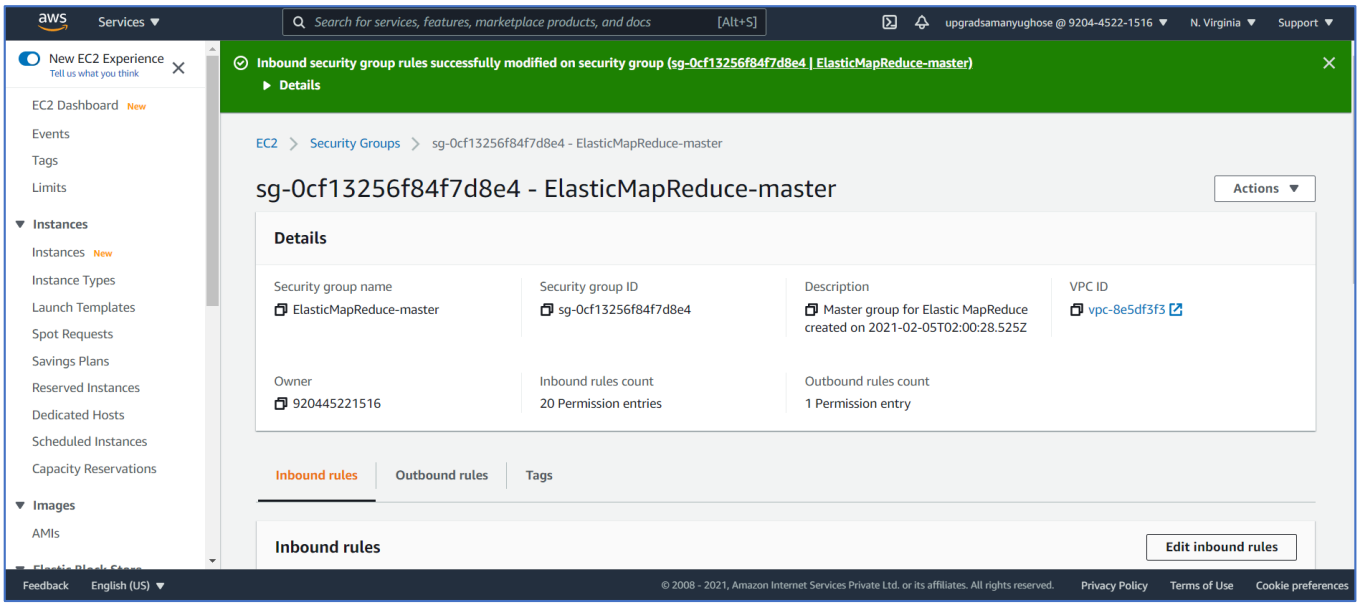




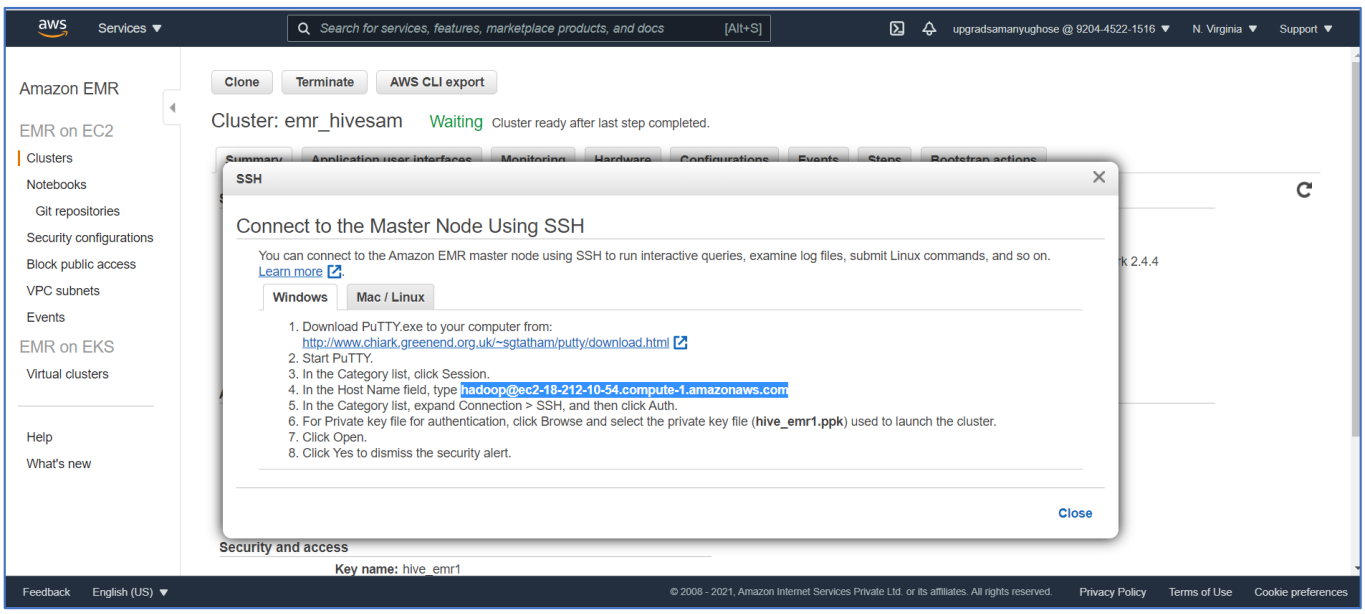
Click on Add rule, select SSH from drop down list and rule as ‘anywhere’ so that every time we clone a cluster, we won’t have to select ‘My IP’ and for testing purpose, it is a common practice to choose the IP as ‘Anywhere’

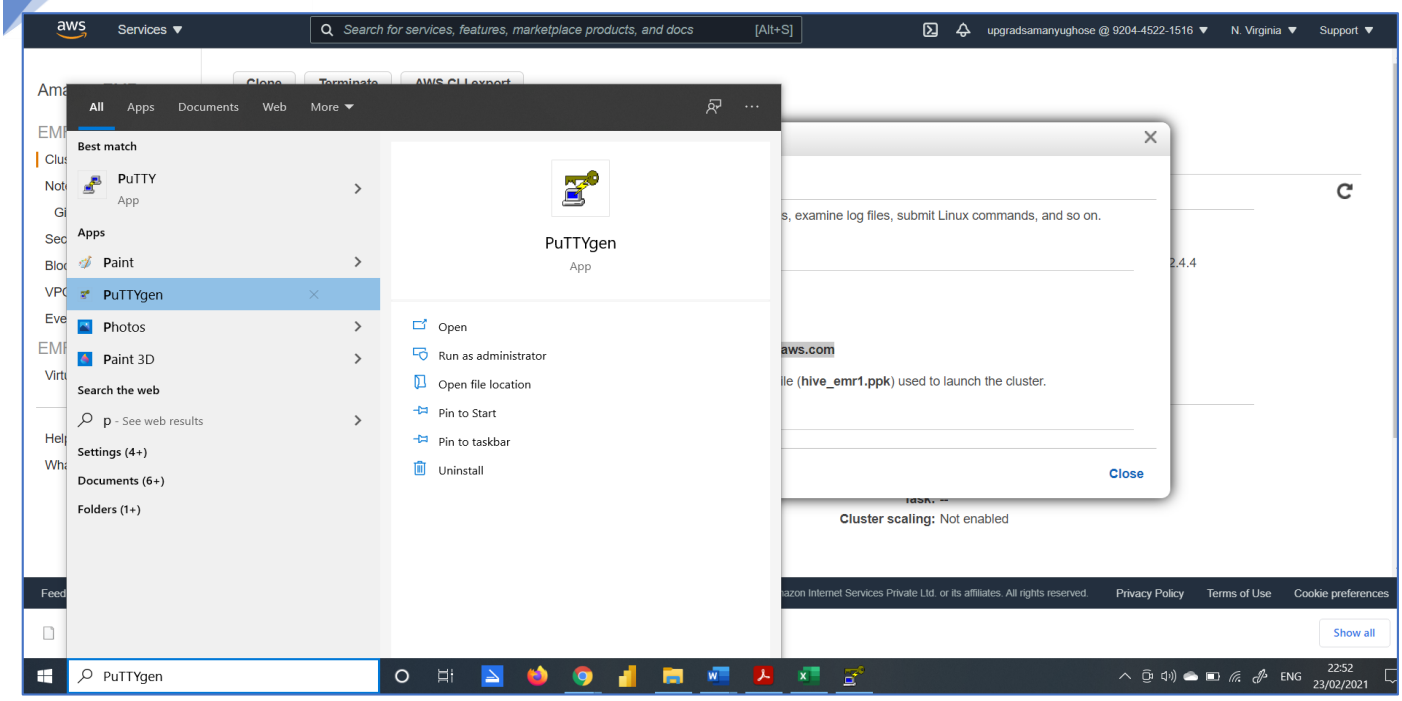


Now inbound security rules has been successfully modified

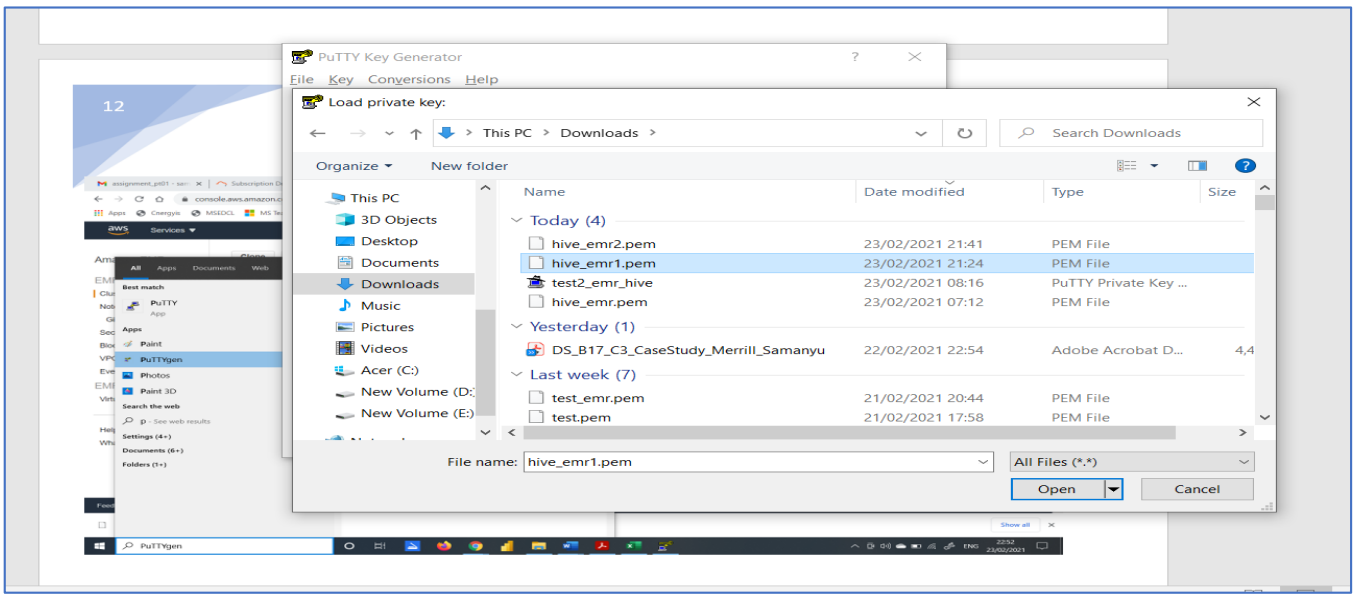


Now goto EMR , Select EMR cluster ‘emr_hivesam’ and select ‘connect to master node using SSH’ under Master public DNS and we will copy the Host name field and paste it in PuttY gen

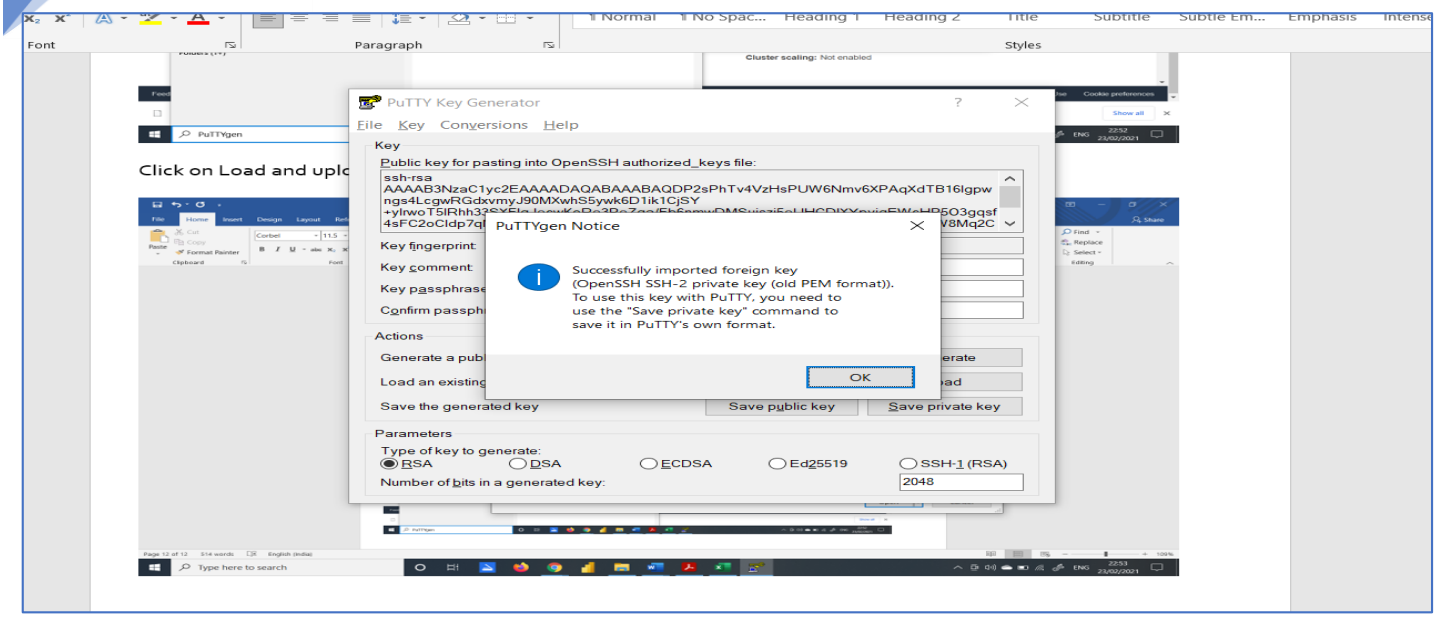




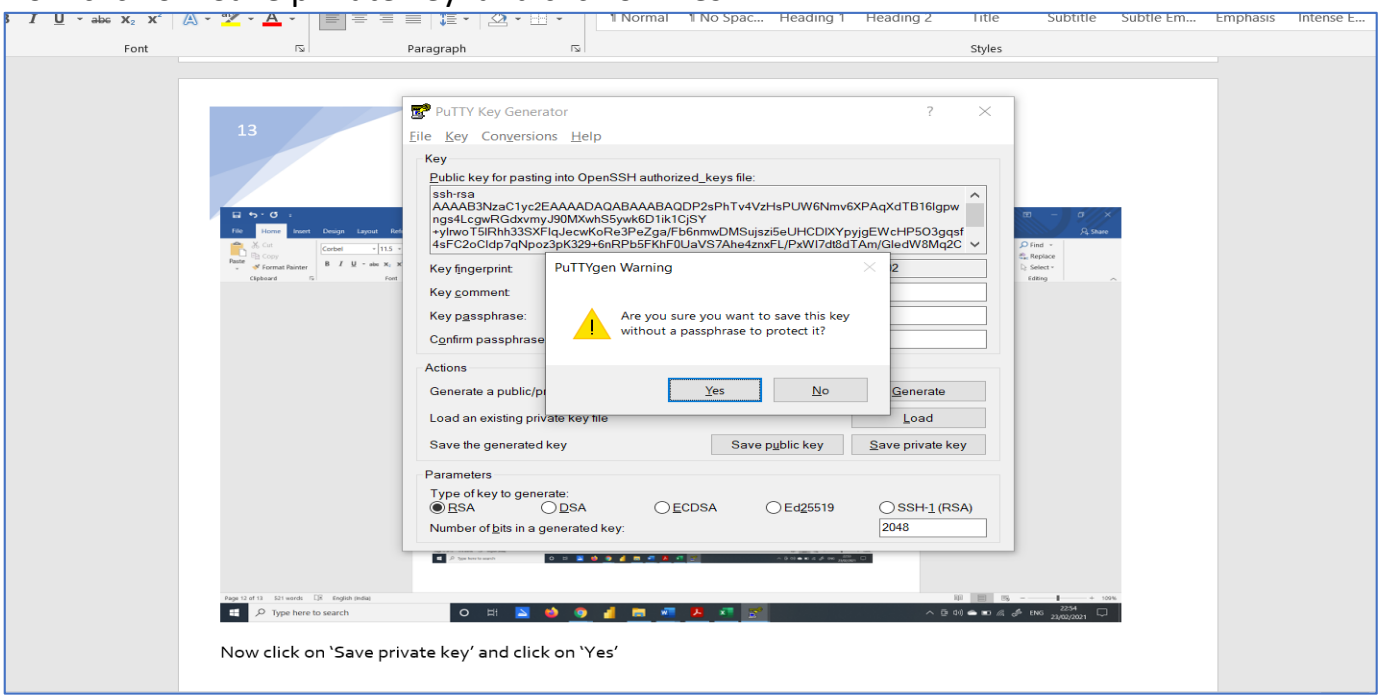
Click on Load and upload ‘hive_emr1.pem’ file



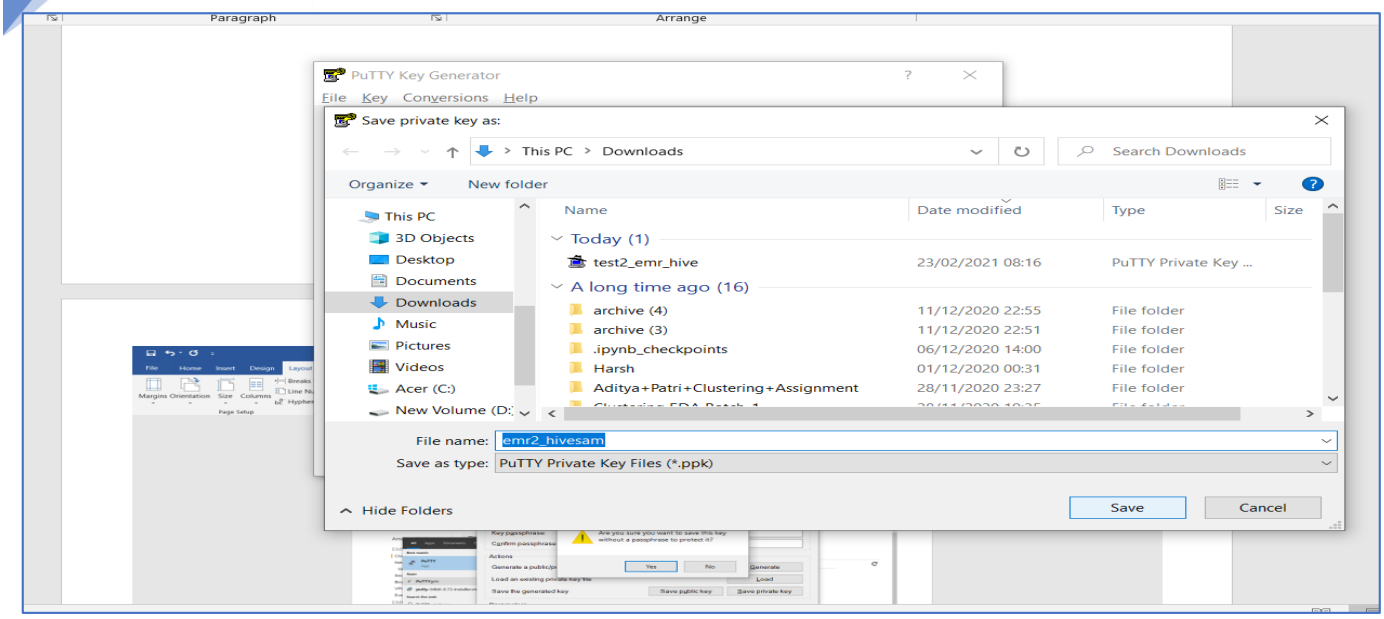
We will get successful message



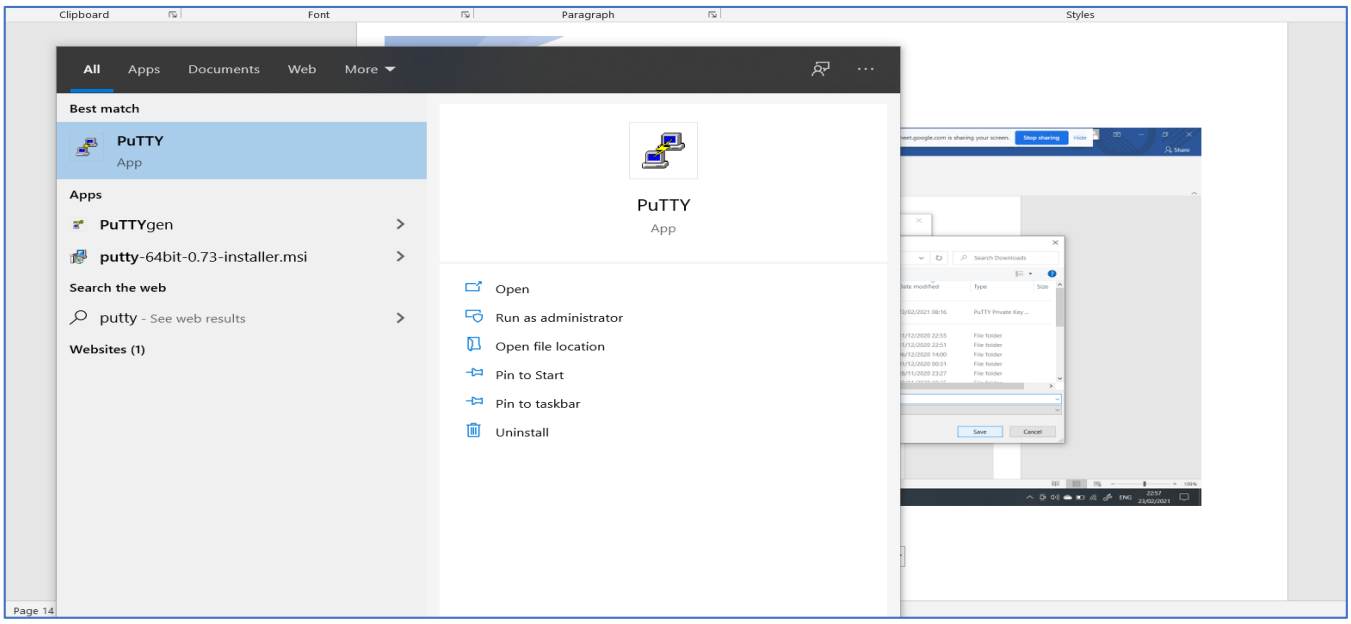
Now click on ‘Save private key’ and click on ‘Yes’

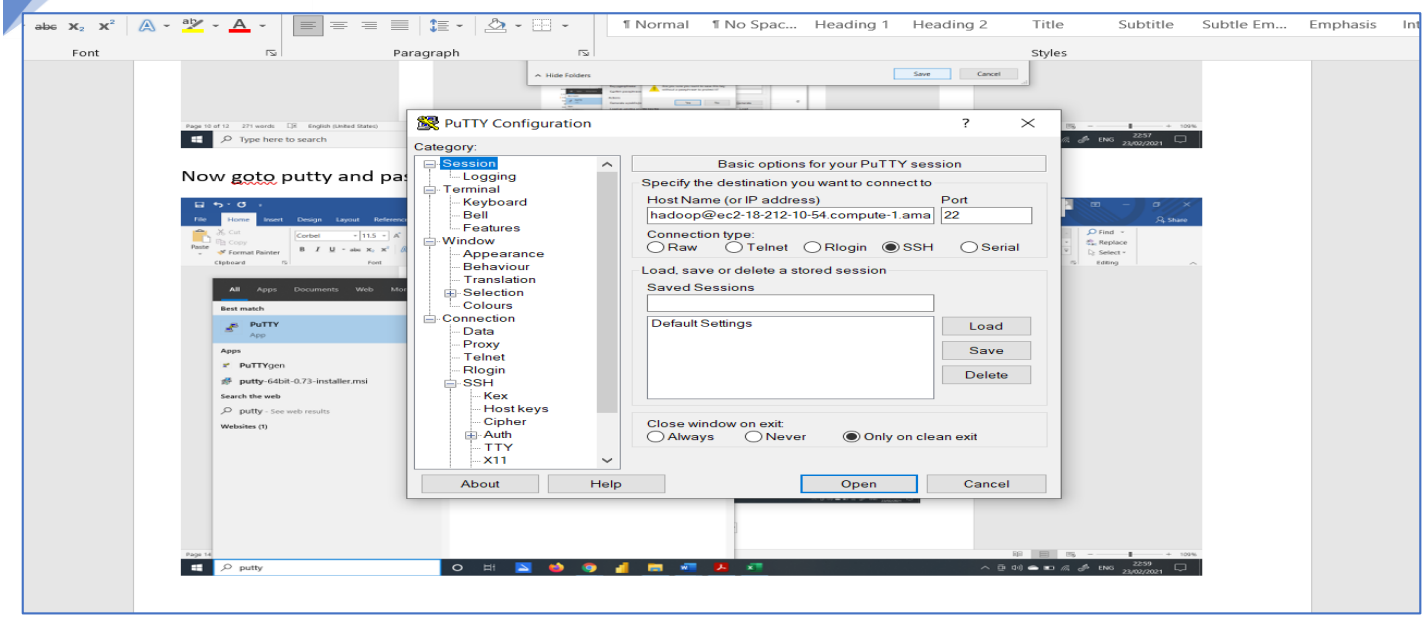


Type the name of .ppk file as ‘emr2_hivesam’

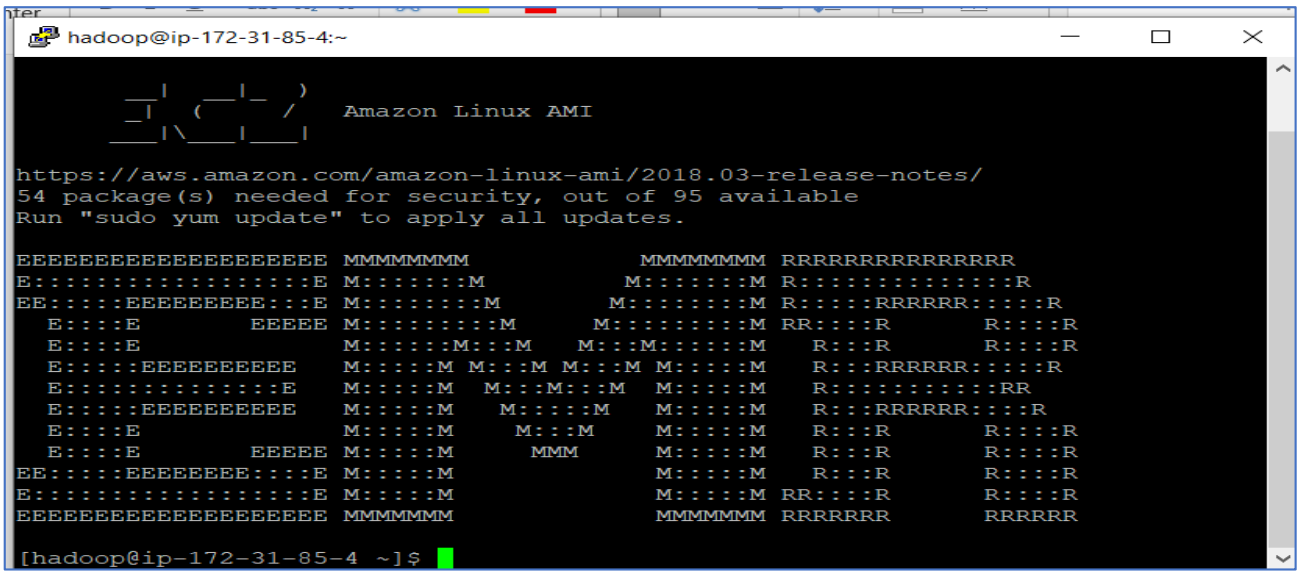


Now goto putty and paste the Master DNS SSH host name





And click on ‘open’ to open Putty terminal and click on ‘Yes’



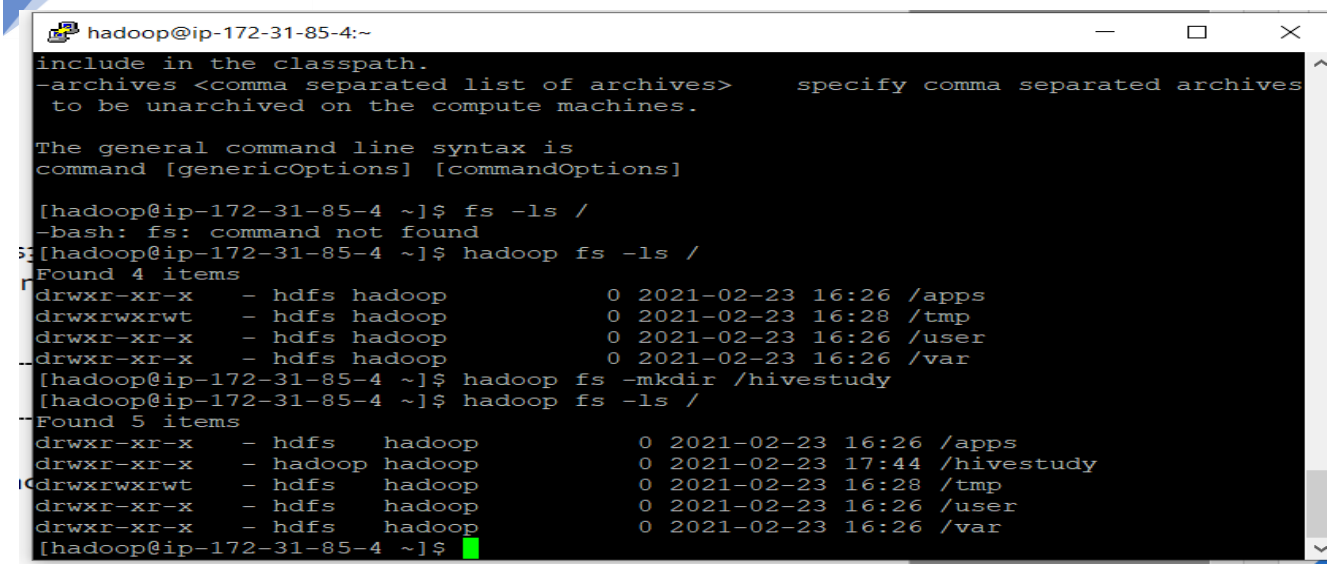
Snapshot: Master node, when connected to EMR cluster

1. Create Directory

First created another folder name hivestudy and placed all related data needed for this assignment in that folder.

Command:

hadoop fs -mkdir /hivestudy

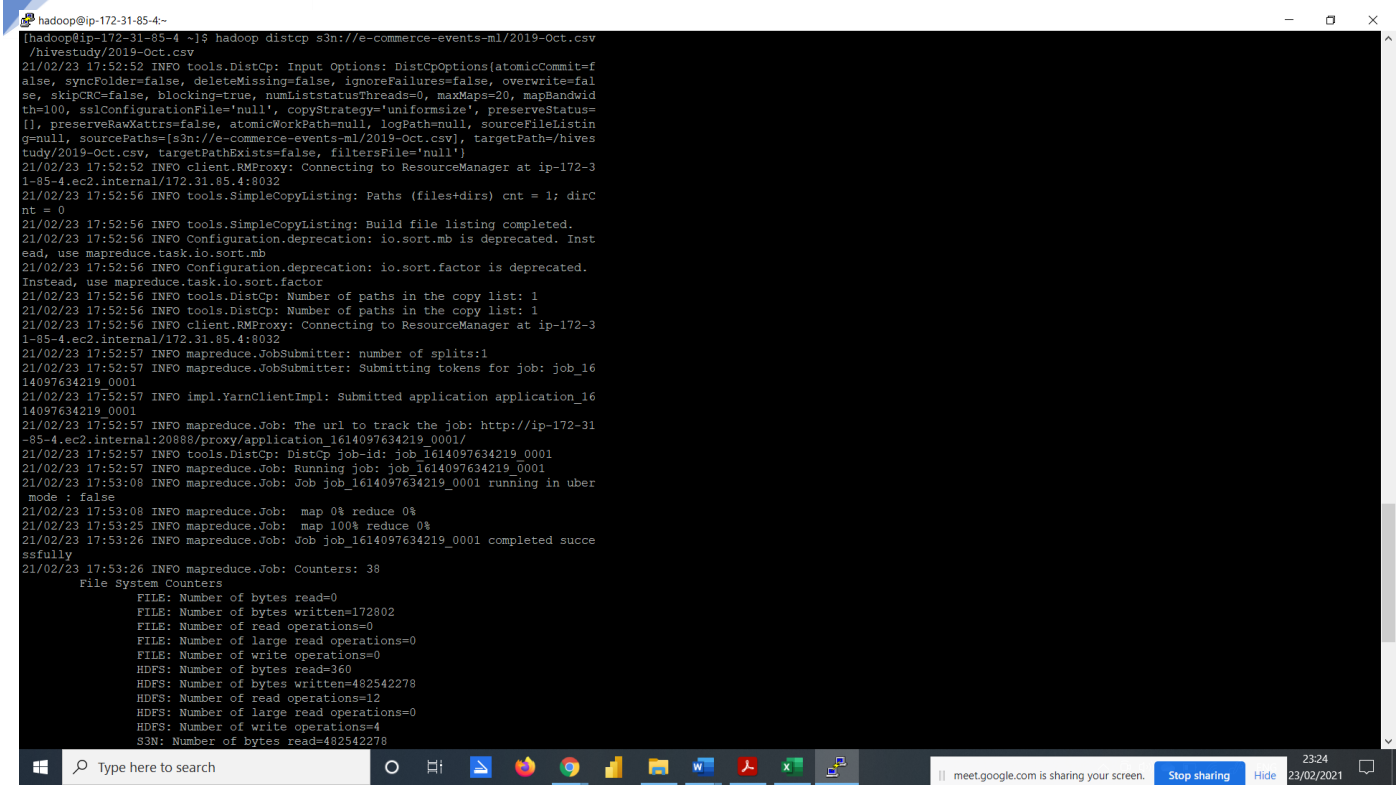


Snapshot: Directory creation

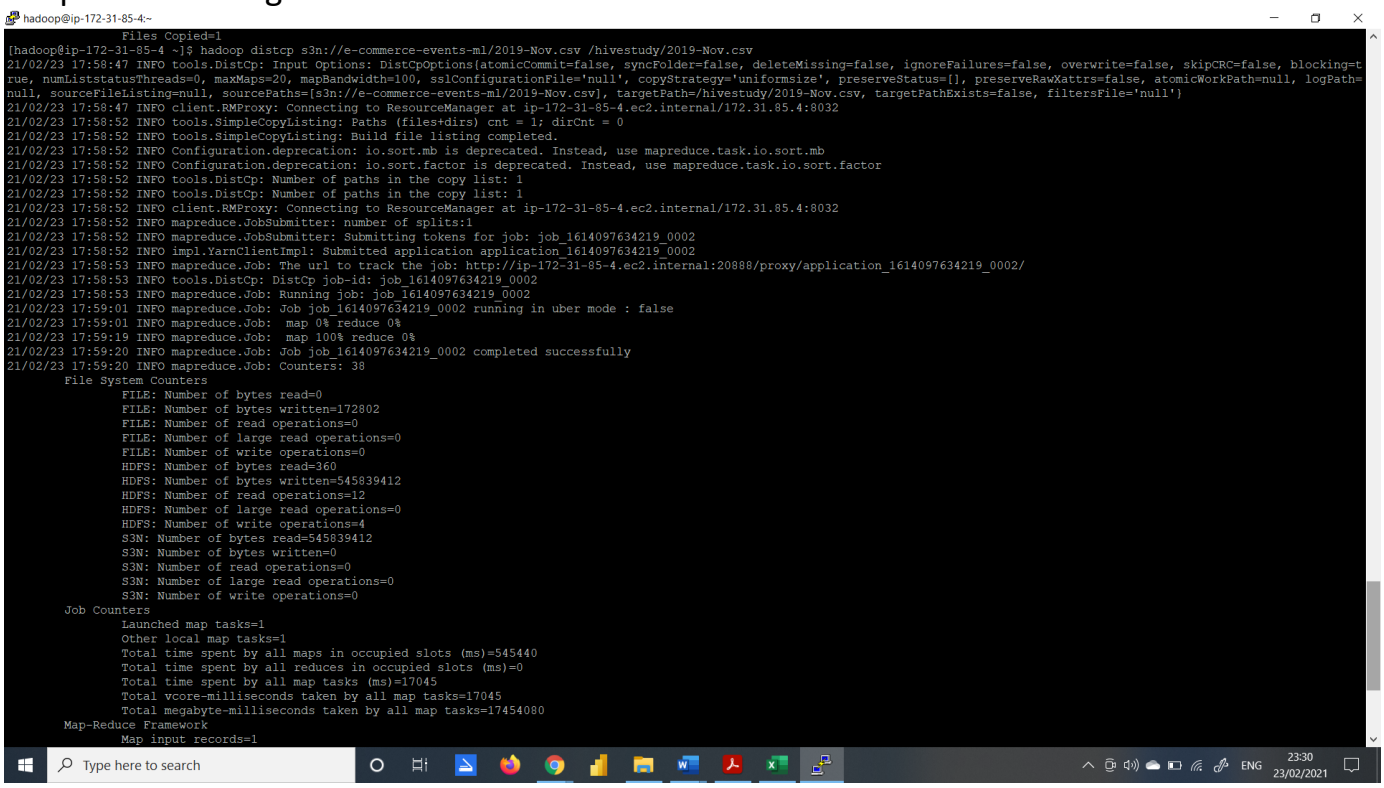
2. Loading data into HDFS

Case study data is present in s3 in two files.
https://e-commerce-events-ml.s3.amazonaws.com/2019-Oct.csv https://e-commerce-events-ml.s3.amazonaws.com/2019-Nov.csv
Both of the files loaded into HDFS under hivestudy directory with file names 2019-Oct.csv & 2019-Nov.csv

Commands:
hadoop distcp s3n://e-commerce-events-ml/2019-Oct.csv /hivestudy/2019-Oct.csv
hadoop distcp s3n://e-commerce-events-ml/2019-Nov.csv /hivestudy/2019-Nov.csv



Snapshot: Loading Oct file into HDFS



Snapshot: Loading Nov file into HDFS

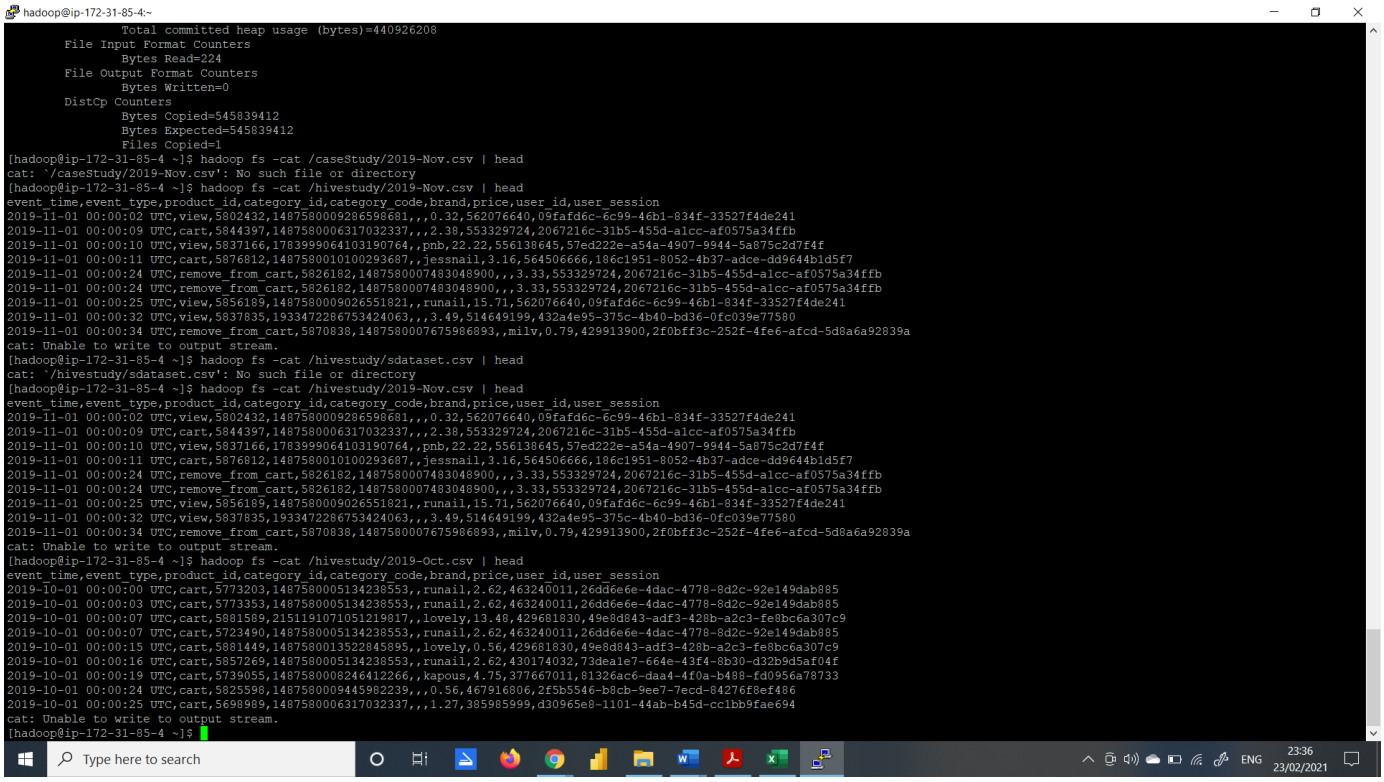
3. Quick view

Taking a quick view of few rows to check whether data has correctly loaded into the file.

Commands:

hadoop fs -cat /hivestudy/2019-Nov.csv | head

hadoop fs -cat /hivestudy/2019-Oct.csv | head



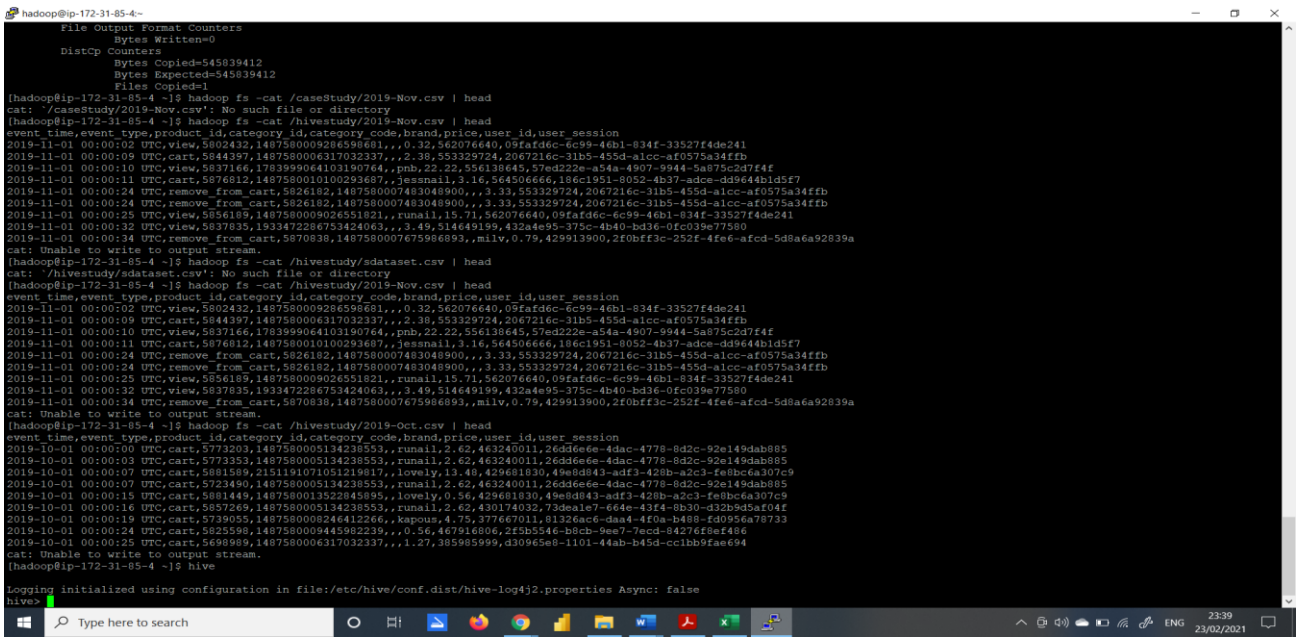
Snapshot: Quick view of loaded data

Hive DDL Querying

1. Enter Hive CLI

Now, that data has been loaded into HDFS, entering into HIVE command line input to perform analysis.

Command: hive



Snapshot: Entering hive CLI

2. Enable headers in CLI

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

```
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: false
hive> set hive.cli.print.header=true;
hive>
```

Snapshot: Enabling headers in hive

3. Create & use database

Command :-

CREATE DATABASE IF NOT EXISTS hivecs WITH DBPROPERTIES('Creator'= ‘ecom','Purpose'='Hive Case Study');

SHOW DATABASES;

USE hivecs;

```
hive> CREATE DATABASE IF NOT EXISTS hivecs WITH DBPROPERTIES('Creator'= 'ecom','
Purpose'='Hive Case Study');
OK
Time taken: 0.791 seconds
hive> SHOW DATABASES;
OK
database_name
default
hivecs
Time taken: 0.15 seconds, Fetched: 2 row(s)
hive>
```

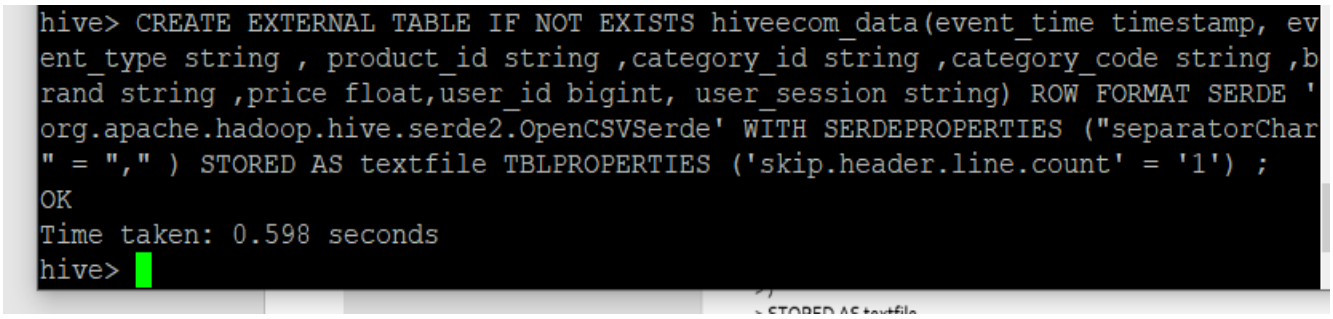
```
hive> USE hivecs;
OK
Time taken: 0.548 seconds
hive>
```

Snapshot: Creating, showing & using databases

4. Create Table Schema
In order to perform analysis on loaded data, a schema needs to be created. Since our files have an extension of .csv, used csvSerde format. By using this format, data types for all the columns get over written to string data type by default

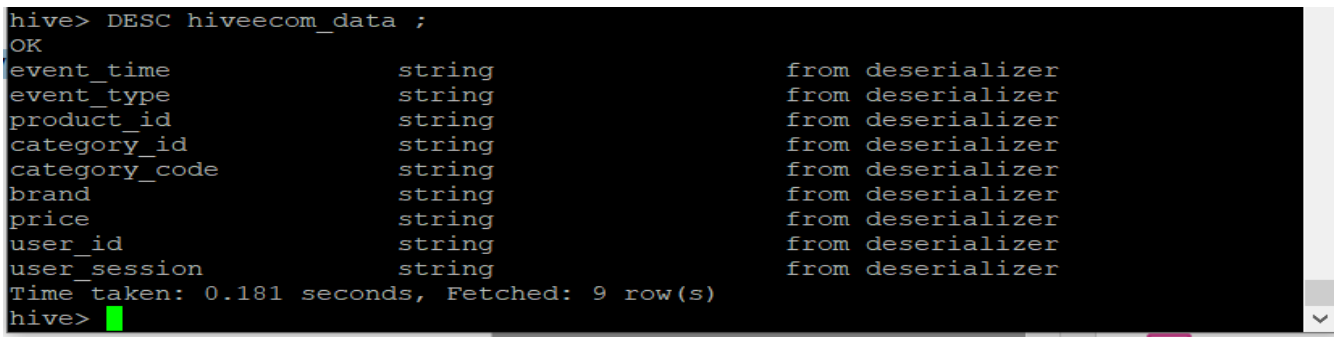
Command :

CREATE EXTERNAL TABLE IF NOT EXISTS hiveecom_data(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' WITH SERDEPROPERTIES ("separatorChar" = ",") STORED AS textfile TBLPROPERTIES ('skip.header.line.count' = '1') ;



Checking the data base

DESC hiveecom_data ;



5. Load Data into table

Now that the schema is created, loaded data into table from HDFS location. In HDFS data is present in two files. Combining the data from those files into a single table.

Commands:

LOAD DATA INPATH '/hivestudy/2019-Oct.csv' into table hiveecom_data ;

LOAD DATA INPATH '/hivestudy/2019-Nov.csv' into table hiveecom_data ;

```
hive> LOAD DATA INPATH '/hivestudy/2019-Oct.csv' into table hiveecom_data ;
Loading data to table hivecs.hiveecom_data
OK
Time taken: 1.186 seconds
hive> LOAD DATA INPATH '/hivestudy/2019-Nov.csv' into table hiveecom_data ;
Loading data to table hivecs.hiveecom_data
OK
Time taken: 0.659 seconds
hive>
```

Checking first few rows of the table:

Command:

SELECT * FROM hiveecom_data limit 3 ;

```
hive> SELECT * FROM hiveecom_data limit 3 ;
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     2
2.22    556138645       57ed222e-a54a-4907-9944-5a875c2d7f4f
Time taken: 1.956 seconds, Fetched: 3 row(s)
hive>
```

6. Create another table schema & load data

In the above data types for all columns of the above table is string type, CsvSerde format gives an error while trying to alter data types in that table. Due to this reason, created a new table with required data types for each column and also added another column for simplicity of our analysis. Once table is created, loaded data from previously created table (hiveecom_data) into the newly created table by doing modifications to its data types

A new column has been added i.e. event_month which is derived from event_time column

```
CREATE TABLE IF NOT EXISTS hiveecom_data1
(event_time timestamp,
event_month string,
event_type string,
product_id string ,
category_id string ,
category_code string ,
brand string ,
price float,
user_id bigint,
user_session string) ;
```

```
hive> CREATE TABLE IF NOT EXISTS hiveecom_data1
    > (event_time timestamp,
    > event_month string,
    > event_type string,
    > product_id string ,
    > category_id string ,
    > category_code string ,
    > brand string ,
    > price float,
    > user_id bigint,
    > user_session string)
    > ;
OK
Time taken: 0.101 seconds
hive>
```

Snapshot: Create table hiveecom_data1

DESC hiveecom_data1 ;

```
hive> DESC hiveecom_data1 ;
OK
col_name        data_type       comment
event_time              timestamp
event_month             string
event_type              string
product_id              string
category_id             string
category_code           string
brand                   string
price                   float
user_id                 bigint
user_session            string
Time taken: 0.045 seconds, Fetched: 10 row(s)
hive>
```

Snapshot: DESC hiveecom_data1

INSERT INTO TABLE hiveecom_data1
SELECT substr(event_time,1,19) AS event_time,
(CASE WHEN event_time REGEXP '-10-' THEN 'Oct' WHEN event_time REGEXP '-11-' THEN 'Nov' ELSE 'Other_months' END) AS event_month,
event_type,
product_id,
category_id,
category_code,
brand,
CAST (price AS float) AS price,
CAST(user_id AS bigint) AS user_id,
user_session
FROM hiveecom_data;

```
hive> INSERT INTO TABLE hiveecom_data1
    > SELECT substr(event_time,1,19) AS event_time,
    > (CASE WHEN event_time REGEXP '-10-' THEN 'Oct' WHEN event_time REGEXP '-11
-' THEN 'Nov' ELSE 'Other_months' END) AS event_month,
    > event_type,
    > product_id,
    > category_id,
    > category_code,
    > brand,
    > CAST (price AS float) AS price,
    > CAST(user_id AS bigint) AS user_id,
    > user_session
    > FROM hiveecom_data;
Query ID = hadoop_20210225162926_04bbcfa7-38e9-4b18-963a-37df3c761b43
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_1614268971294
_0004)

Map 1:  0/2
Map 1:  0/2
Map 1:  0(+1)/2
Map 1:  0(+2)/2
Map 1:  0(+2)/2
Map 1:  0(+2)/2
Map 1:  0(+2)/2
Map 1:  0(+2)/2
Map 1:  0(+2)/2
Map 1:  0(+2)/2
Map 1:  0(+2)/2
Map 1:  0(+2)/2
Map 1:  0(+2)/2
Map 1:  0(+2)/2
Map 1:  0(+2)/2
Map 1:  0(+2)/2
Map 1:  0(+2)/2
Map 1:  0(+2)/2
Map 1:  0(+2)/2
```

Snapshot: Inserting data into the new table

7. Data Quality Checks :-

a. Checked Month & year to make sure data only belongs to 2019 Oct & Nov months.

No discrepancies have been found for this.

SELECT DISTINCT(month(from_unixtime(unix_timestamp(event_time, 'yyyy-MM-dd HH:mm:ss')))) AS event_month FROM hiveecom_data1 ;

```
hive> SELECT DISTINCT(month(from_unixtime(unix_timestamp(event_time, 'yyyy-MM-dd HH:mm:ss')))) AS event_month FROM hiveecom_data1 ;
Query ID = hadoop_20210225163326_d2bcd045-d494-485f-9150-a38655798fc8
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1614268971294_0004)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      8          8        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      4          4        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 55.01 s
--------------------------------------------------------------------------------
OK
event_month
11
10
Time taken: 56.041 seconds, Fetched: 2 row(s)
hive>
```

Snapshot: Month data quality check

SELECT DISTINCT(year(from_unixtime(unix_timestamp(event_time, 'yyyy-MM-dd HH:mm:ss')))) AS event_year FROM hiveecom_data1 ;

```
hive> SELECT DISTINCT(year(from_unixtime(unix_timestamp(event_time, 'yyyy-MM-dd HH:mm:ss')))) AS event_year FROM hiveecom_data1 ;
Query ID = hadoop_20210225163502_5f27d75d-1c18-4456-9ced-f131efef696d
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1614268971294_0004)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      8          8        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      4          4        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 53.27 s
--------------------------------------------------------------------------------
OK
event_year
2019
Time taken: 54.006 seconds, Fetched: 1 row(s)
hive>
```

Snapshot: Year data quality check

b. Checked event_type column if there are more than 4 distinct entries present apart from the ones mentioned in data description table.
No discrepancies were found.

SELECT DISTINCT(event_type) AS event_type FROM hiveecom_data1 ;

```
hive> SELECT DISTINCT(event_type) AS event_type FROM hiveecom_data1 ;
Query ID = hadoop_20210225163627_ecd1fa71-4797-4434-8bd2-31091f833744
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1614268971294_0004)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      8          8        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      4          4        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 28.85 s
--------------------------------------------------------------------------------
OK
event_type
view
purchase
cart
remove_from_cart
Time taken: 29.517 seconds, Fetched: 4 row(s)
hive>
```

Snapshot: Event_type data quality check

c. General statistics regarding data which include how many files are present, rows in the table, raw size & compressed size of the table etc.

ANALYZE TABLE hiveecom_data1 COMPUTE STATISTICS NOSCAN;

```
hive> ANALYZE TABLE hiveecom_data1 COMPUTE STATISTICS NOSCAN;
OK
hiveecom_data1.event_time      hiveecom_data1.event_month      hiveecom_data1.event_type      hiveecom_data1.product_id      hiveecom_data1.category_id      hiveecom_data1.category_codeh
iveecom_data1.brand     hiveecom_data1.price    hiveecom_data1.user_id  hiveecom_data1.user_session
Time taken: 0.329 seconds
hive>
```

Snapshot: Analyze table hiveecom_data1

8.Optimization in Hive

a. Enabling bucketing & vectorized Execution

set hive.enforce.bucketing= true;

set hive.vectorized.execution.enabled = true;

```
hive> set hive.enforce.bucketing= true;
hive> set hive.vectorized.execution.enabled = true;
```

Snapshot: enable bucket and vectorized execution

9. Create partitioned & bucketed table

To get insights stated in problem statement, most queries require categorization of event_time as October & November months. Also, few queries required frequent group by operations on brand column. Since the cardinality of event_month (derived column) and event_type is low, therefore we have performed two level static partitioning on these columns as below.
All insights except for query 4 and 5 had to be drawn when the event_type was 'purchase', hence performed static partitioning
On the other hand, brand column has high cardinalities, chose to perform bucketing (10 buckets) over this column

CREATE TABLE IF NOT EXISTS hive_h1
(event_time timestamp , product_id string, category_id string, brand string, price float, user_id bigint)
PARTITIONED BY (event_month string, event_type string)
CLUSTERED BY (brand) INTO 10 BUCKETS ;

```
hive> CREATE TABLE IF NOT EXISTS hive_h1
    > (event_time timestamp , product_id string, category_id string, brand string, price float, user_id bigint)
    > PARTITIONED BY (event_month string, event_type string)
    > CLUSTERED BY (brand) INTO 10 BUCKETS ;
OK
Time taken: 0.077 seconds
hive>
```

Snapshot: Create table hive_h1

DESC hive_h1 ;

```
hive> DESC hive_h1 ;
OK
col_name            data_type           comment
event_time                      timestamp
product_id                      string
category_id                     string
brand                           string
price                           float
user_id                         bigint
event_month                     string
event_type                      string

# Partition Information
# col_name                      data_type                       comment

event_month                     string
event_type                      string
Time taken: 0.074 seconds, Fetched: 14 row(s)
```

Snapshot: desc hive_h1

10. Load data into partitioned & bucketed table

Loading data twice into the partitioned & bucketed table. Since its static partitioning, first inserted data where month was October and event_type was purchase.
After the first set of data was inserted, we have added data where month was November and event_type was purchase

INSERT INTO TABLE hive_h1
PARTITION (event_month = 'Oct', event_type = 'purchase')
SELECT event_time, product_id ,category_id ,brand ,price , user_id
FROM hiveecom_data1 WHERE event_month = 'Oct' AND event_type = 'purchase' ;

```
hadoop@ip-172-31-81-235:~
Time taken: 0.03 seconds
hive> INSERT INTO TABLE hive_h1
    > PARTITION (event_month = 'Oct', event_type = 'purchase')
    > SELECT event_time, product_id ,category_id ,brand ,price , user_id
    > FROM hiveecom_data1 WHERE event_month = 'Oct' AND event_type = 'purchase'
;
Query ID = hadoop_20210225165146_e2f26b9b-8172-4ae0-bf8e-b5914dec9d4d
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_1614268971294
_0006)

Map 1:  0/8         Reducer 2:  0/10
Map 1:  0/8         Reducer 2:  0/10
Map 1:  0/8         Reducer 2:  0/10
Map 1:  0(+1)/8     Reducer 2:  0/10
Map 1:  0(+2)/8     Reducer 2:  0/10
Map 1:  0(+3)/8     Reducer 2:  0/10
Map 1:  0(+3)/8     Reducer 2:  0/10
Map 1:  0(+3)/8     Reducer 2:  0/10
Map 1:  0(+3)/8     Reducer 2:  0/10
Map 1:  1(+2)/8     Reducer 2:  0/10
Map 1:  1(+3)/8     Reducer 2:  0/10
Map 1:  3(+3)/8     Reducer 2:  0/10
Map 1:  3(+3)/8     Reducer 2:  0/10
Map 1:  3(+3)/8     Reducer 2:  0/10
Map 1:  5(+1)/8     Reducer 2:  0/10
Map 1:  6(+0)/8     Reducer 2:  0/10
Map 1:  6(+1)/8     Reducer 2:  0/10
Map 1:  6(+2)/8     Reducer 2:  0(+1)/10
Map 1:  6(+2)/8     Reducer 2:  0(+1)/10
Map 1:  7(+1)/8     Reducer 2:  0(+2)/10
Map 1:  8/8         Reducer 2:  0(+3)/10
Map 1:  8/8         Reducer 2:  1(+3)/10
Map 1:  8/8         Reducer 2:  2(+3)/10
Map 1:  8/8         Reducer 2:  3(+2)/10
Map 1:  8/8         Reducer 2:  4(+2)/10
Map 1:  8/8         Reducer 2:  4(+3)/10
Map 1:  8/8         Reducer 2:  5(+3)/10
Map 1:  8/8         Reducer 2:  6(+2)/10
Map 1:  8/8         Reducer 2:  6(+3)/10
Map 1:  8/8         Reducer 2:  7(+2)/10
Map 1:  8/8         Reducer 2:  7(+3)/10
Map 1:  8/8         Reducer 2:  8(+2)/10
Map 1:  8/8         Reducer 2:  10/10
Loading data to table hivecs.hive_h1 partition (event_month=Oct, event_type=purchase)
OK
Time taken: 52.358 seconds
```

Snapshot: Insert table hive_h1

INSERT INTO TABLE hive_h1
PARTITION (event_month = 'Nov', event_type = 'purchase')
SELECT event_time, product_id ,category_id ,brand ,price , user_id
FROM hiveecom_data1 WHERE event_month = 'Nov' AND event_type = 'purchase' ;

```
hadoop@ip-172-31-81-235:~
Map 1: 8/8        Reducer 2: 10/10
Loading data to table hivecs.hive_h1 partition (event_month=Oct, event_type=purchase)
OK
Time taken: 52.358 seconds
hive> INSERT INTO TABLE hive_h1
    > PARTITION (event_month = 'Nov', event_type = 'purchase')
    > SELECT event_time, product_id ,category_id ,brand ,price , user_id
    > FROM hiveecom_data1 WHERE event_month = 'Nov' AND event_type = 'purchase'
;
Query ID = hadoop_20210225165440_630ed109-cc3e-4310-8ad3-ef80c3713234
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1614268971294
_0006)

Map 1: 0/8        Reducer 2: 0/10
Map 1: 0/8        Reducer 2: 0/10
Map 1: 0/8        Reducer 2: 0/10
Map 1: 0(+1)/8    Reducer 2: 0/10
Map 1: 0(+2)/8    Reducer 2: 0/10
Map 1: 0(+3)/8    Reducer 2: 0/10
Map 1: 0(+3)/8    Reducer 2: 0/10
Map 1: 0(+3)/8    Reducer 2: 0/10
Map 1: 0(+3)/8    Reducer 2: 0/10
Map 1: 0(+3)/8    Reducer 2: 0/10
Map 1: 1(+3)/8    Reducer 2: 0/10
Map 1: 2(+3)/8    Reducer 2: 0/10
Map 1: 3(+3)/8    Reducer 2: 0/10
Map 1: 4(+2)/8    Reducer 2: 0/10
Map 1: 6(+0)/8    Reducer 2: 0/10
Map 1: 6(+1)/8    Reducer 2: 0/10
Map 1: 6(+2)/8    Reducer 2: 0/10
Map 1: 6(+2)/8    Reducer 2: 0(+1)/10
Map 1: 7(+1)/8    Reducer 2: 0(+2)/10
Map 1: 8/8        Reducer 2: 0(+2)/10
Map 1: 8/8        Reducer 2: 0(+3)/10
Map 1: 8/8        Reducer 2: 1(+3)/10
Map 1: 8/8        Reducer 2: 2(+3)/10
Map 1: 8/8        Reducer 2: 3(+3)/10
Map 1: 8/8        Reducer 2: 5(+3)/10
Map 1: 8/8        Reducer 2: 6(+3)/10
Map 1: 8/8        Reducer 2: 7(+2)/10
Map 1: 8/8        Reducer 2: 7(+3)/10
Map 1: 8/8        Reducer 2: 9(+1)/10
Map 1: 8/8        Reducer 2: 10/10
Loading data to table hivecs.hive_h1 partition (event_month=Nov, event_type=purc
hase)
OK
Time taken: 38.556 seconds
hive>
```

Verifying data load:

Command:

SELECT *

FROM hive_h1 AS h

LIMIT 10 ;

```
hive> SELECT *
    > FROM hive_h1 AS h
    > LIMIT 10 ;
OK
2019-11-17 21:41:15     5875408 2084144451428549153             2.05    464392142       Nov     purchase
2019-11-17 21:41:15     5875407 2084144451428549153             2.05    464392142       Nov     purchase
2019-11-17 21:41:15     5875376 2084144451428549153             2.05    464392142       Nov     purchase
2019-11-17 21:41:15     5875366 2084144451428549153             2.05    464392142       Nov     purchase
2019-11-17 21:41:15     5875364 2084144451428549153             2.05    464392142       Nov     purchase
2019-11-17 21:41:15     5875361 2084144451428549153             2.05    464392142       Nov     purchase
2019-11-17 21:41:15     5875360 2084144451428549153             2.05    464392142       Nov     purchase
2019-11-17 21:41:15     5875359 2084144451428549153             2.05    464392142       Nov     purchase
2019-11-17 21:41:15     5875334 2084144451428549153             2.05    464392142       Nov     purchase
2019-11-17 21:41:15     5754898 1487580005092295511             4.76    464392142       Nov     purchase
Time taken: 0.251 seconds, Fetched: 10 row(s)
hive>
```

Snapshot: Select * from hive_h1

NEED TO TAKE SCREENSHOT OF COMPARISON DATA FOR EXECUTION TIME FOR BOTH TABLE as given below

SELECT * FROM hiveecom_data1 limit 5;

SELECT * FROM hive_h1 limit 5;

Query 1

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

Command before partitioning:

// summing the price column when month was October & event_type was purchase

```
SELECT ROUND(SUM(price),2) AS Oct_Rev
FROM hiveecom_data1
WHERE event_month = 'Oct' AND event_type = 'purchase';
```

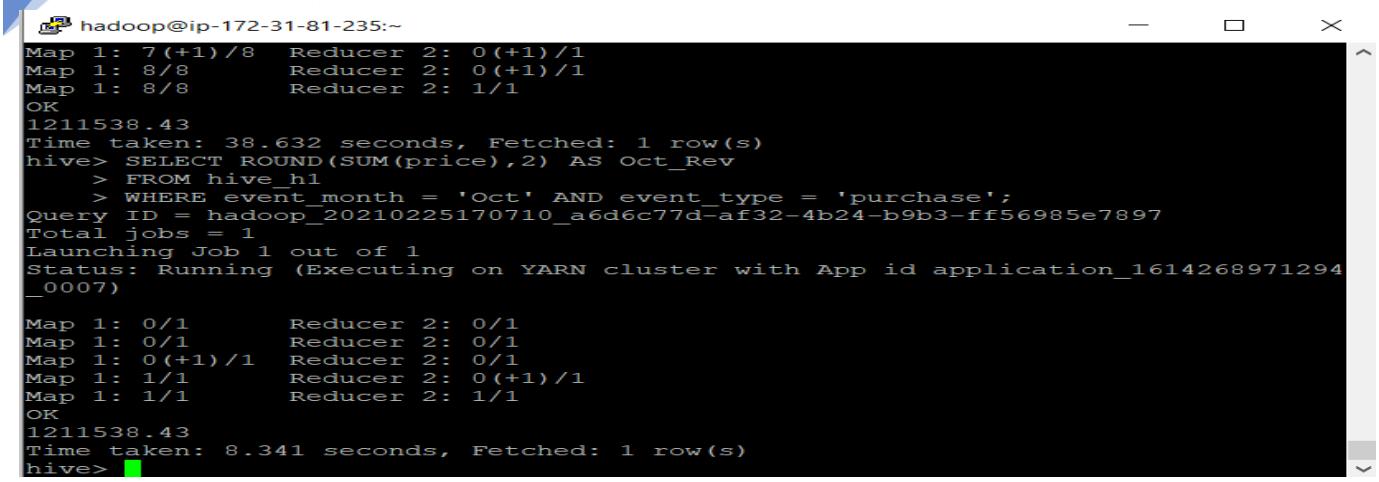
```
hadoop@ip-172-31-81-235:~
2019-11-17 21:41:15      5875408 2084144451428549153              2.05     464392142       Nov     purchase
2019-11-17 21:41:15      5875407 2084144451428549153              2.05     464392142       Nov     purchase
2019-11-17 21:41:15      5875376 2084144451428549153              2.05     464392142       Nov     purchase
2019-11-17 21:41:15      5875366 2084144451428549153              2.05     464392142       Nov     purchase
2019-11-17 21:41:15      5875364 2084144451428549153              2.05     464392142       Nov     purchase
2019-11-17 21:41:15      5875361 2084144451428549153              2.05     464392142       Nov     purchase
2019-11-17 21:41:15      5875360 2084144451428549153              2.05     464392142       Nov     purchase
2019-11-17 21:41:15      5875359 2084144451428549153              2.05     464392142       Nov     purchase
2019-11-17 21:41:15      5875334 2084144451428549153              2.05     464392142       Nov     purchase
2019-11-17 21:41:15      5754898 1487580005092295511              4.76     464392142       Nov     purchase
Time taken: 0.251 seconds, Fetched: 10 row(s)
hive> SELECT ROUND(SUM(price),2) AS Oct_Rev
    > FROM hiveecom_data1
    > WHERE event_month = 'Oct' AND event_type = 'purchase';
Query ID = hadoop_20210225170415_989c5d55-cff7-489b-984f-ef615dc61748
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_1614268971294
_0007)

Map 1: 0/8        Reducer 2: 0/1
Map 1: 0/8        Reducer 2: 0/1
Map 1: 0/8        Reducer 2: 0/1
Map 1: 0(+1)/8    Reducer 2: 0/1
Map 1: 0(+2)/8    Reducer 2: 0/1
Map 1: 0(+3)/8    Reducer 2: 0/1
Map 1: 0(+3)/8    Reducer 2: 0/1
Map 1: 0(+3)/8    Reducer 2: 0/1
Map 1: 0(+3)/8    Reducer 2: 0/1
Map 1: 0(+2)/8    Reducer 2: 0/1
Map 1: 2(+2)/8    Reducer 2: 0/1
Map 1: 2(+3)/8    Reducer 2: 0/1
Map 1: 3(+2)/8    Reducer 2: 0/1
Map 1: 3(+3)/8    Reducer 2: 0/1
Map 1: 3(+3)/8    Reducer 2: 0/1
Map 1: 4(+2)/8    Reducer 2: 0/1
Map 1: 5(+1)/8    Reducer 2: 0/1
Map 1: 6(+0)/8    Reducer 2: 0/1
Map 1: 6(+1)/8    Reducer 2: 0/1
Map 1: 6(+2)/8    Reducer 2: 0/1
Map 1: 6(+2)/8    Reducer 2: 0(+1)/1
Map 1: 7(+1)/8    Reducer 2: 0(+1)/1
Map 1: 8/8        Reducer 2: 0(+1)/1
Map 1: 8/8        Reducer 2: 1/1
OK
1211538.43
Time taken: 38.632 seconds, Fetched: 1 row(s)
hive>
```

Snapshot: Query 1 results (before optimization)

Command after partitioning:

// summing the price column when month was October & event_type was purchase

```
SELECT ROUND(SUM(price),2) AS Oct_Rev
FROM hive_h1
WHERE event_month = 'Oct' AND event_type = 'purchase';
```



Snapshot: Query 1 results (after optimization)

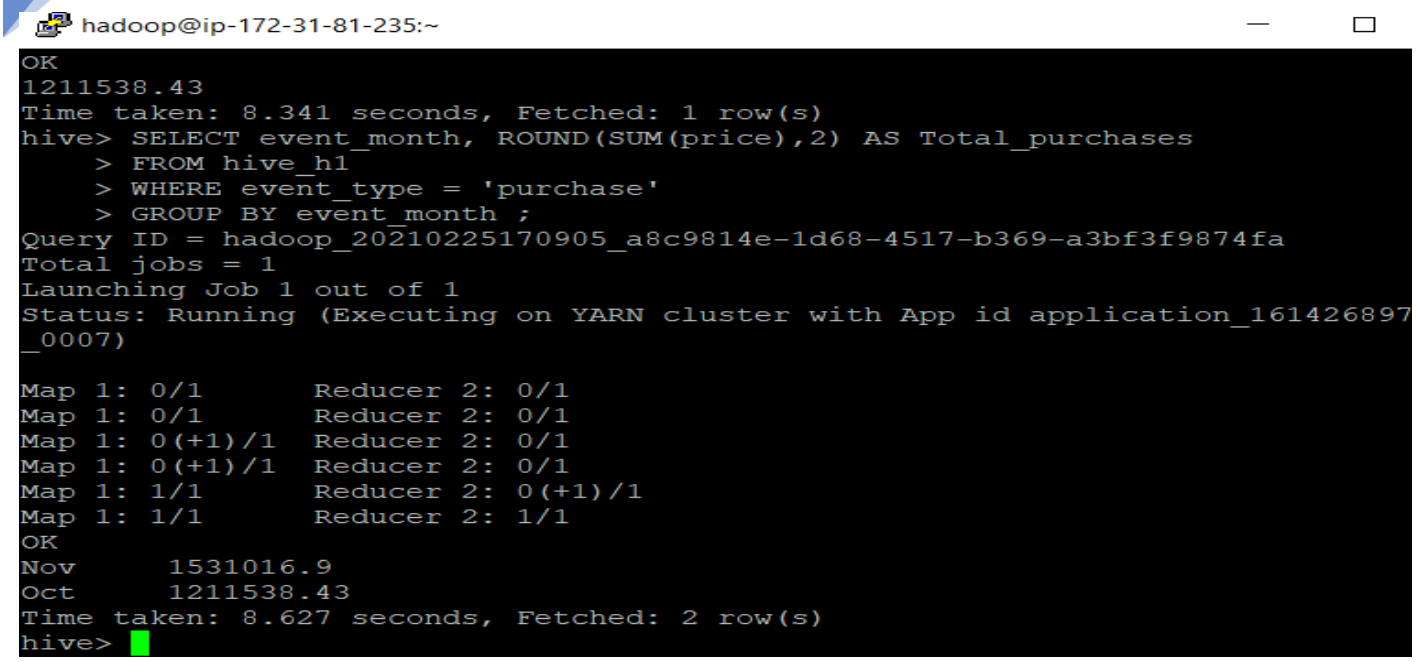
Here without partitioning, it took 38.632 seconds where as with partition, it took 8.341 seconds.
So obviously with partition, it is taking less time for execution so we will use partition table to run rest of queries.

Here the total revenue generated due to purchases made in the month of October is Rs 1211538.43

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

// For each month, aggregating the price where event_type was purchase

```
SELECT event_month, ROUND(SUM(price),2) AS Total_purchases
FROM hive_h1
WHERE event_type = 'purchase'
GROUP BY event_month ;
```



The total sum of purchases in the month of Nov is Rs 1531016.9 and in Oct is Rs 1211538.43

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

// Aggregating prices where event type was purchased based on month of the event

```
WITH month_wise_agg AS (
SELECT event_month,SUM(price) AS current_month_sales
FROM hive_h1
WHERE event_type = 'purchase'
GROUP BY event_month
),
```

// Using lead function making a new column which displays the sales for Nov month in the previous row.

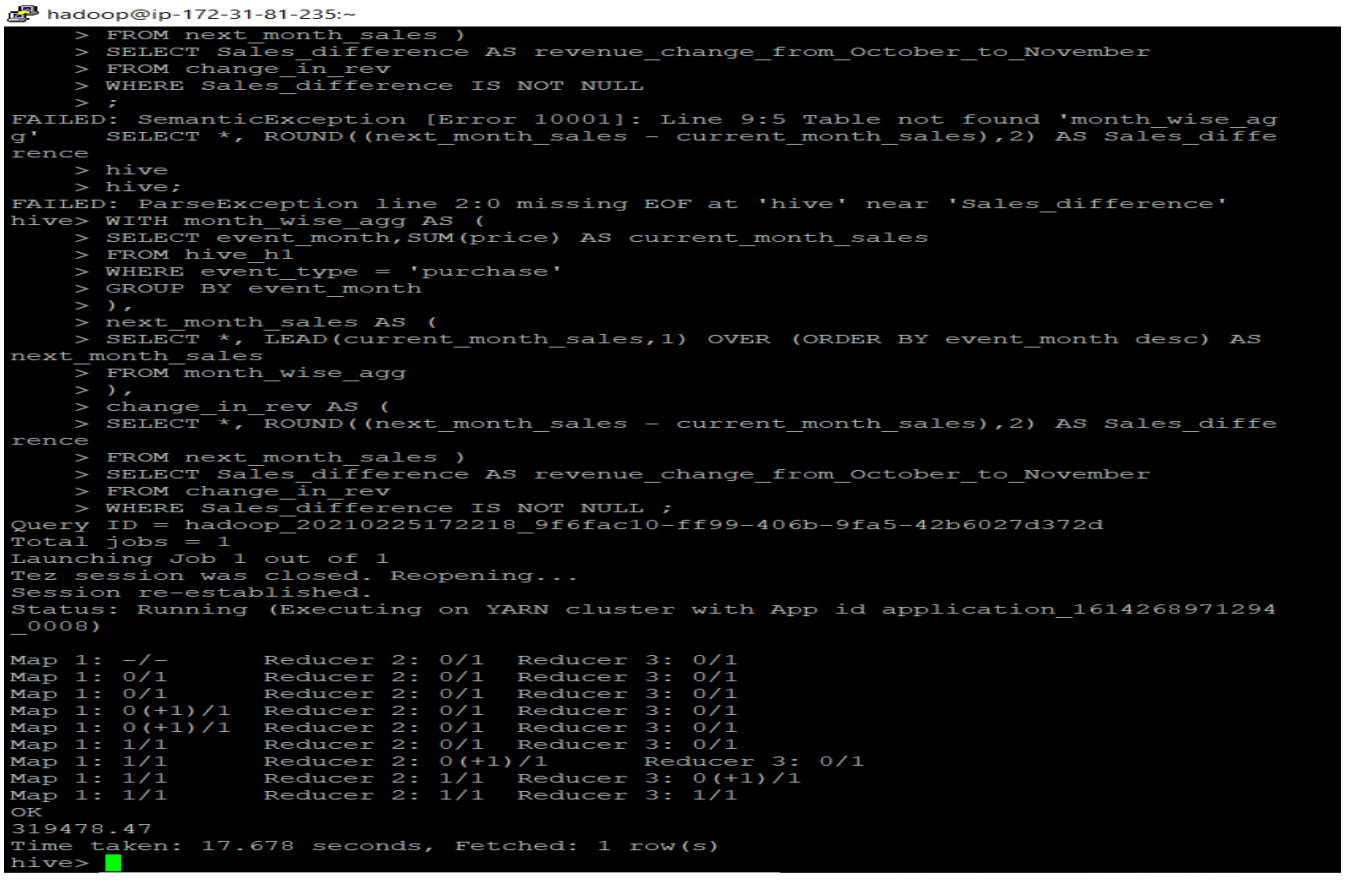
```
next_month_sales AS (
SELECT *, LEAD(current_month_sales,1) OVER (ORDER BY event_month desc) AS
next_month_sales
FROM month_wise_agg
),
```

// Calculating the difference in revenue from Oct month to Nov month

```
change_in_rev AS (
SELECT *, ROUND((next_month_sales - current_month_sales),2) AS Sales_difference
FROM next_month_sales )
```

// Change in revenue from October to November

```
SELECT Sales_difference AS revenue_change_from_October_to_November
FROM change_in_rev
WHERE Sales_difference IS NOT NULL ;
```

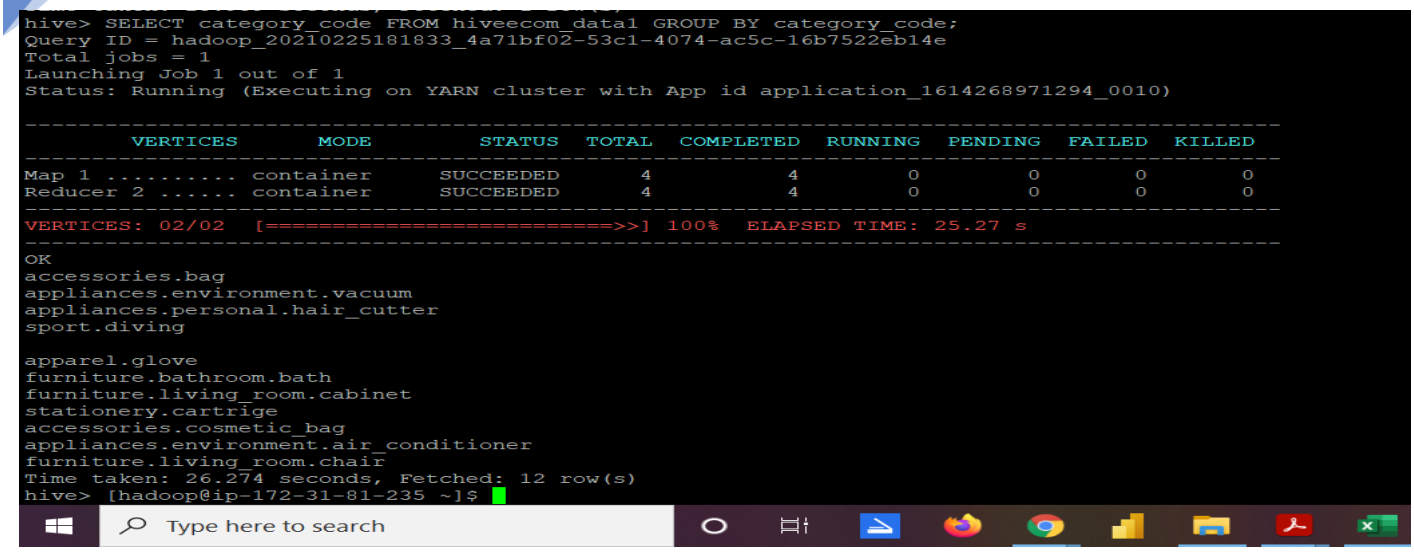


There is an increase in revenue of amount Rs 319478.47 /-

Query 4
Find distinct categories of products.

For this query,we have not used partitioned & bucketed table because that table contains information for event_type = purchase only.

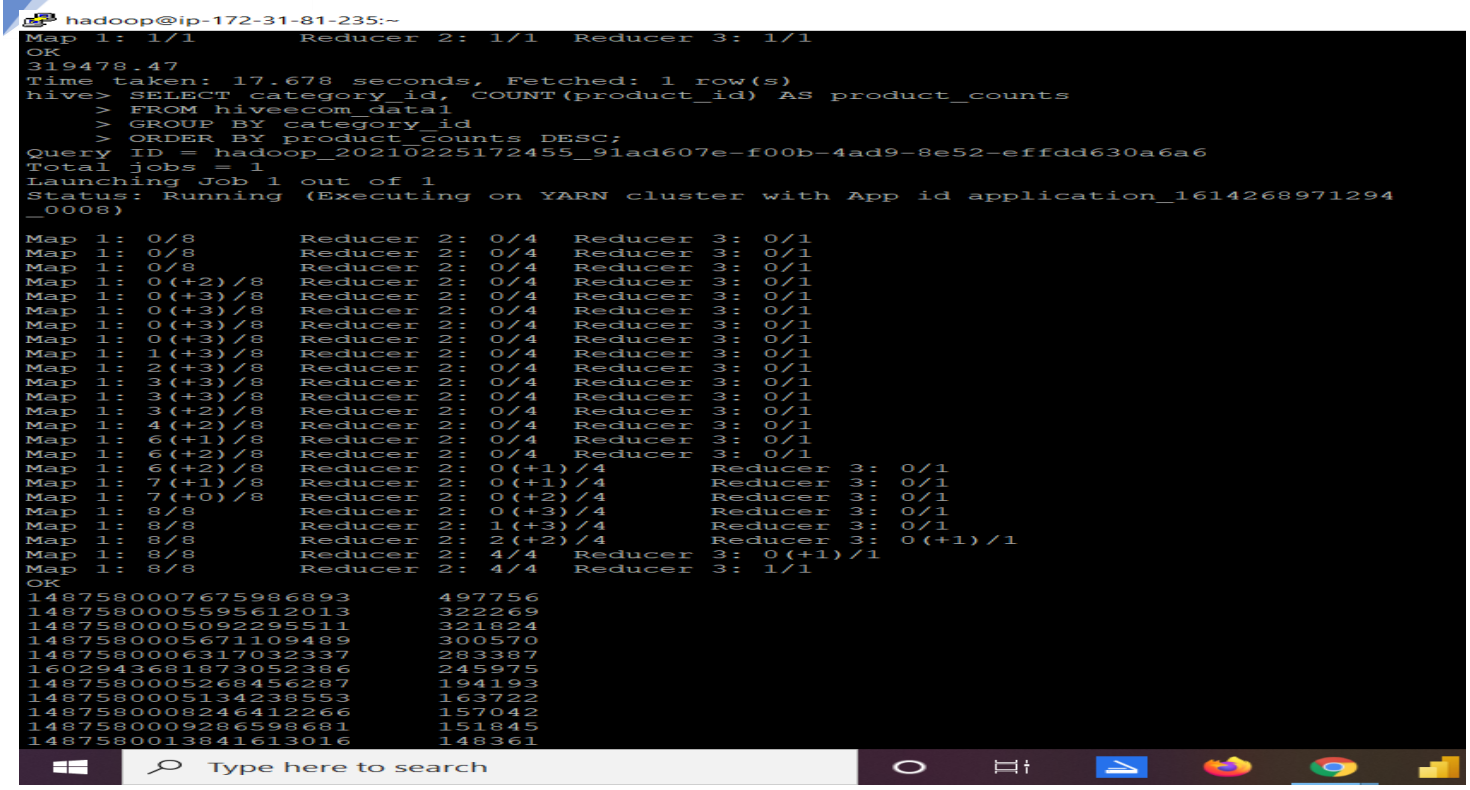
```
SELECT category_code FROM hiveecom_data1 GROUP BY category_code;
```



Query 5
Find the total number of products available under each category. For this query, didn't use the partitioned & bucketed table because that table contains information for event_type = purchase only.

Command:
// Count of products under each category

```
SELECT category_code, COUNT(product_id) AS product_counts
FROM hiveecom_data1
GROUP BY category_code
ORDER BY product_counts DESC;
```

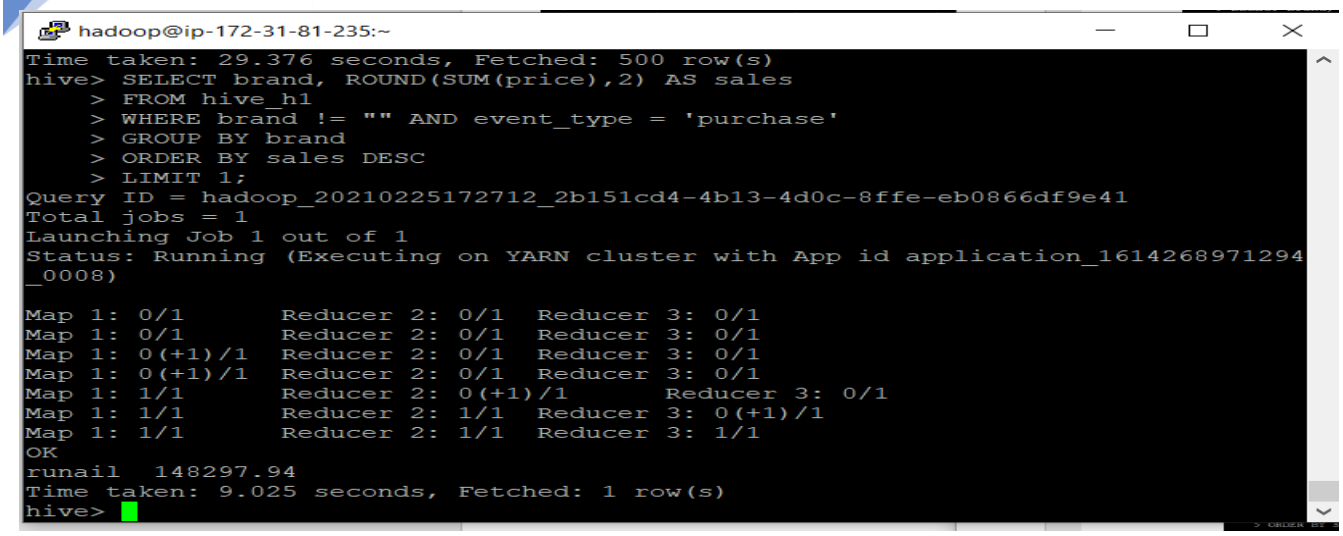


Note: The actual result is quite huge, snapshot only contains few rows of categories with high product counts.

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

// Displaying brand which has most sales in both months combined

```
SELECT brand, ROUND(SUM(price),2) AS sales
FROM hive_h1
WHERE brand != "" AND event_type = 'purchase'
GROUP BY brand
ORDER BY sales DESC
LIMIT 1;
```



Here 'runail' brand has most sales combining both months with a value of 148297.94

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

// Sales in Oct for each brand

```
WITH Oct_table AS (
select brand, ROUND(SUM(price),2) AS Oct_sales
FROM hive_h1
WHERE event_month = 'Oct' AND brand != "" AND event_type = 'purchase'
group by brand ) ,
```

// Sales in Nov for each brand

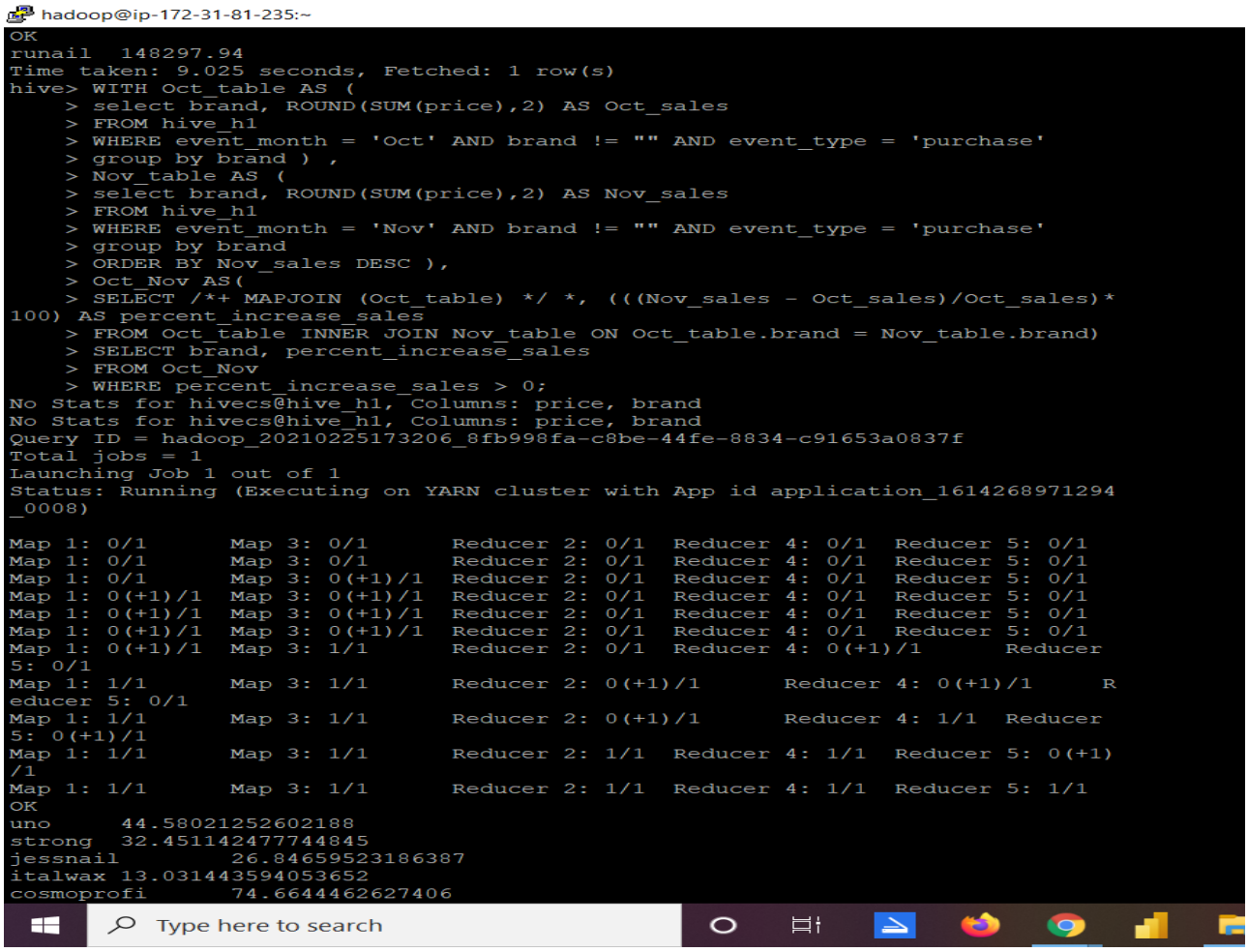
```
Nov_table AS (
select brand, ROUND(SUM(price),2) AS Nov_sales
FROM hive_h1
WHERE event_month = 'Nov' AND brand != "" AND event_type = 'purchase'
group by brand
ORDER BY Nov_sales DESC ),
```

// Combining the sales from Oct and Nov month for each brand and calculating the percentage increase in sales from October to November

```
Oct_Nov AS(
SELECT /*+ MAPJOIN (Oct_table) */ *, (((Nov_sales - Oct_sales)/Oct_sales)*100) AS
percent_increase_sales
```

FROM Oct_table INNER JOIN Nov_table ON Oct_table.brand = Nov_table.brand)

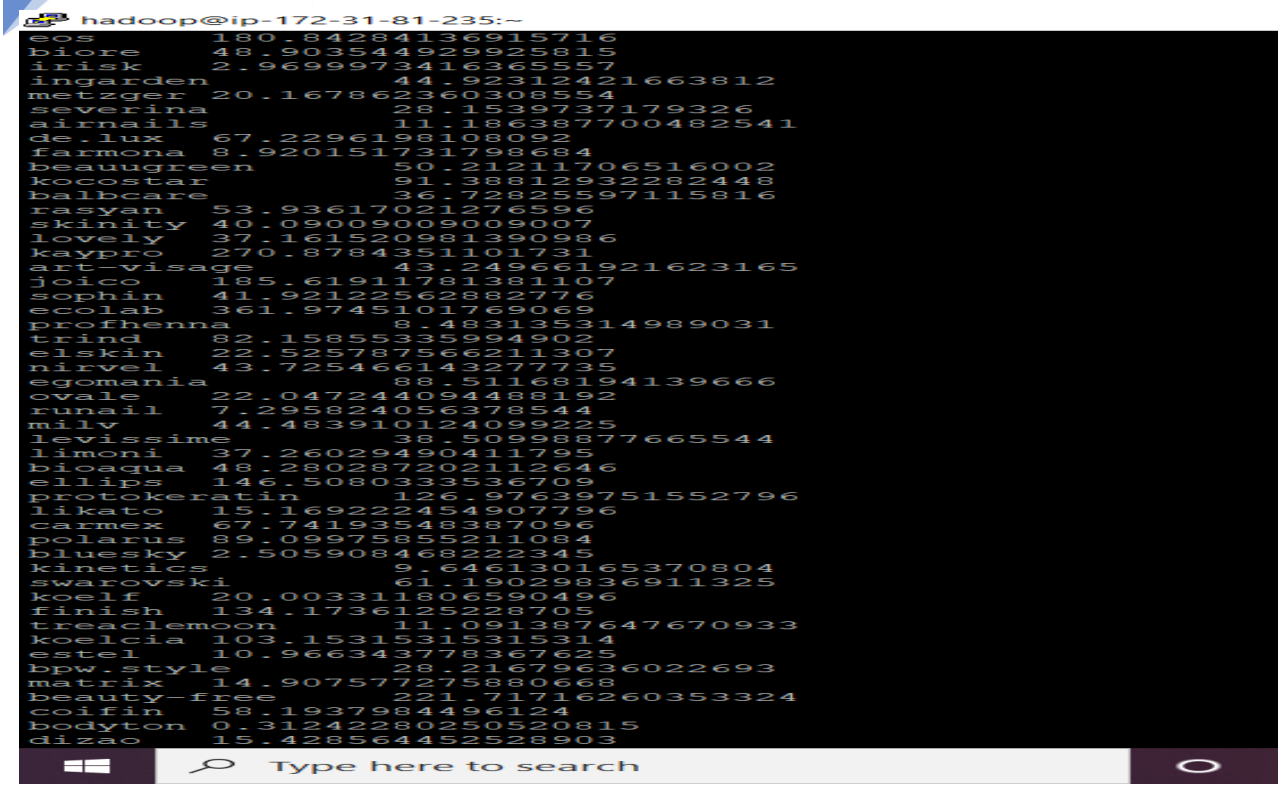
// Displaying only those brand, whose sales have increased
SELECT brand, percent_increase_sales
FROM Oct_Nov
WHERE percent_increase_sales > 0;

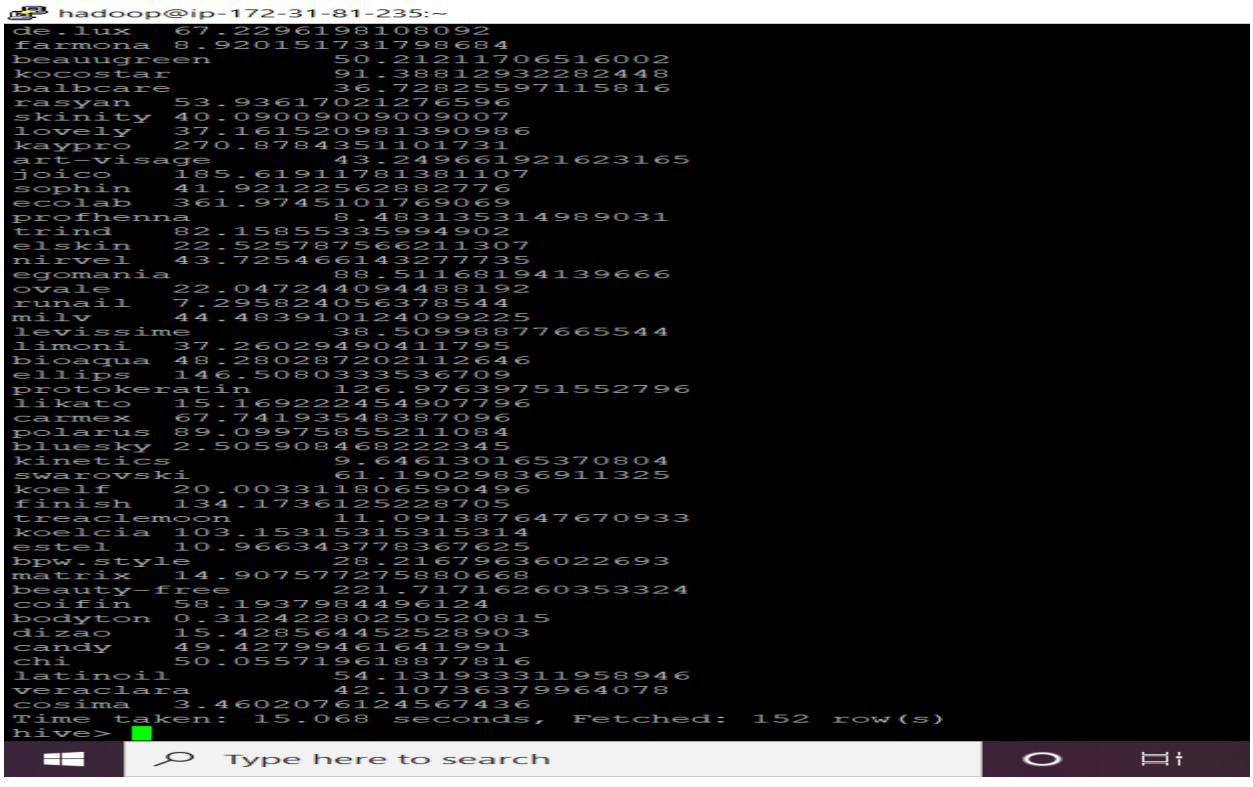


```
hadoop@ip-172-31-81-235:~
strong     32.45114247774484 5
jessnail          26.84659523186387
italwax    13.031443594053652
cosmoprofi        74.66444626274 06
domix      14.678310359480722
staleks    39.389511169954936
marathon          41.099474642035 51
f.o.x      29.48342675296216
smart      32.41632751959724 6
uskusi     10.657550070299692
nagaraku          21.922128090000793
kaaral     15.266871089173073
roubloff          40.740857430915 18
rosi       24.84595585367756
levrana    63.31633653657580 4
depilflax         3.572497201771 6587
lador      18.617687571090 56
missha     66.194940602706 72
zeitun     183.58168938560104
kosmekka          53.488116197183075
insight    19.274087414282747
gehwol     43.028455471181 85
farmavita         54.289023967899496
sanoto     669.81036018836 71
jaguar     0.774877280852200 9
happyfons         36.122057063048 68
s.care     121.253755936803 34
batiste    13.175815639564 99
kiss       93.886846162969 99
konad      9.575172674803 66
osmo       18.081415161560134
entity     49.936419920368 5564
yu-r       148.22593124792 75
igrobeauty        25.583070513569307
cristalinas       36.788812758693275
lowence    133.79591500576 51
kerasys    21.881599405908432
freshbubble       57.621587700031 38
mavala     9.1140230784275 3
cutrin     22.797875538631125
inm        21.939448649399 35
deoproce          3.891554096704 9744
laboratorium      26.782961460446 24
miskin     85.440394836750 2
mane       289.669112142536 2
tertio     4.0819783197832 04
ecocraft          487.827988338 1924
vilenta    17.009109311740897
soleo      4.079333986287959
elizavecca        189.6639727775415
```



```
hadoop@ip-172-31-81-235:~
soleo      4.079333986287959
elizavecca        189.6639727775415
plazan     91.3879846108315 8
aura       111.44729005360333
enjoy      230.2781136638452
profepil          26.413881748071976
marutaka-foot     122.125152377082 49
grace      1.674593737613949 3
glysolid          31.34949089344615
kamill     29.328677987621 01
foamie     129.708904109589 04
supertan          32.042882668254 93
neoleor    19.096982262151 59
fly        58.518086347724626
moyou      80.035026269702 27
masura     5.732698182823044
lianail    178.2060941753043
shik       44.849754579193 12
artex      58.470175489995 02
nitrile    37.225002360494 78
provoc     28.482228046232432
orly       3.181586471331372
greymy     1575.761725436494 5
nefertiti         57.0058239122987 16
beautyblender     38.950977901955 81
blixz      62.772785622593055
grattol    101.640911663357 39
kapous     18.159561873907 958
oniq       16.809152314249 392
freedecor         124.204945963796 63
jas        10.198074095499 79
refectocil        27.958382728685145
benovy     695.852253307944
solomeya          41.380217929146 71
shary      34.924767191155 55
kims       91.504060113925 56
estelare          6.0834963242732 86
godefroy          5.956831663426543
ladykin    35.750099482690 0004
browxenna         4.084466453660 7365
concept    21.285625454354 28
haruyama          31.544220925192924
beautix    16.476160073185 024
yoko       33.698759037148 94
markell    60.250459363957 59
skinlite          36.5846550296039 44
keen       84.311402580918 14
fedua      403.646429935089 77
eos        180.842841369157 16
biore      48.903544929925 815
```



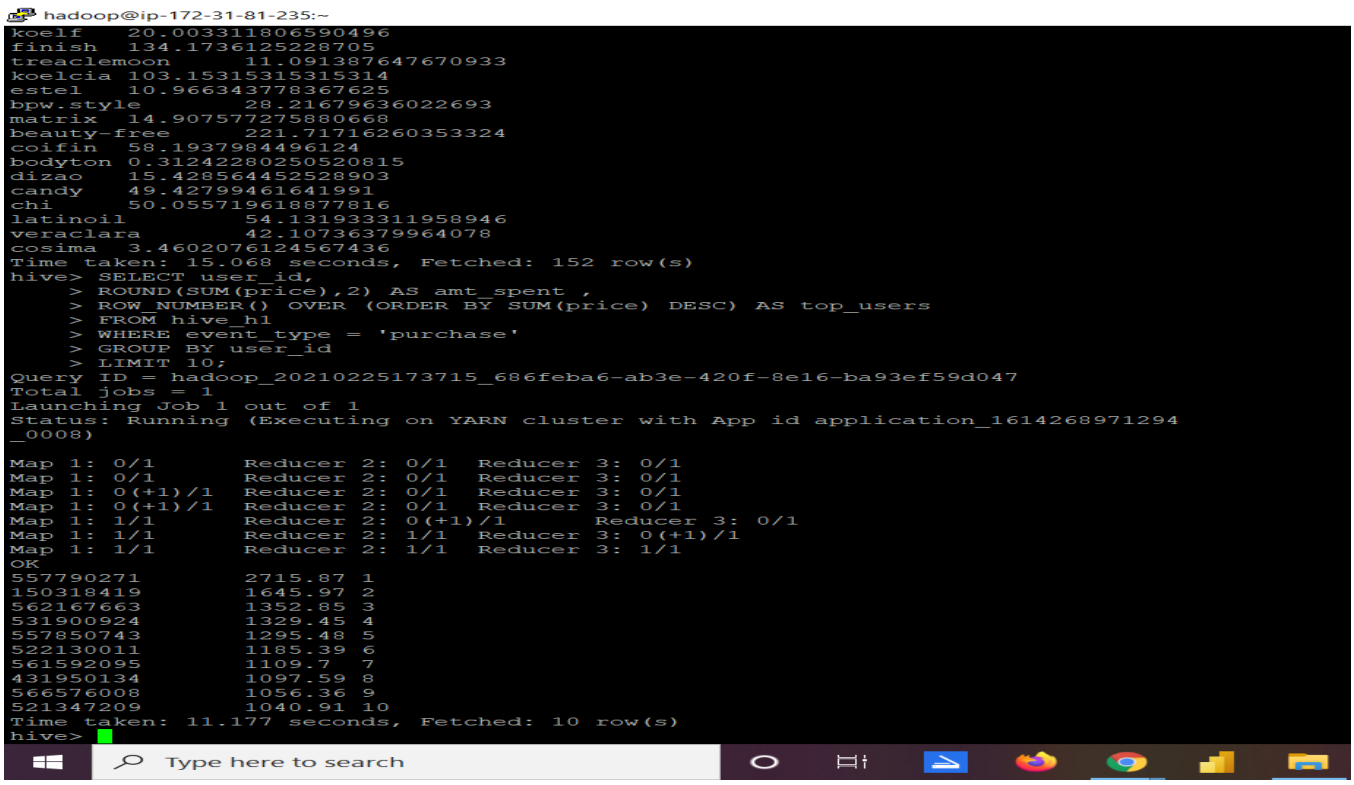


Query 8

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

// Top 10 users which have spent the most

```
SELECT user_id,
ROUND(SUM(price),2) AS amt_spent ,
ROW_NUMBER() OVER (ORDER BY SUM(price) DESC) AS top_users
FROM hive_h1
WHERE event_type = 'purchase'
GROUP BY user_id
LIMIT 10;
```



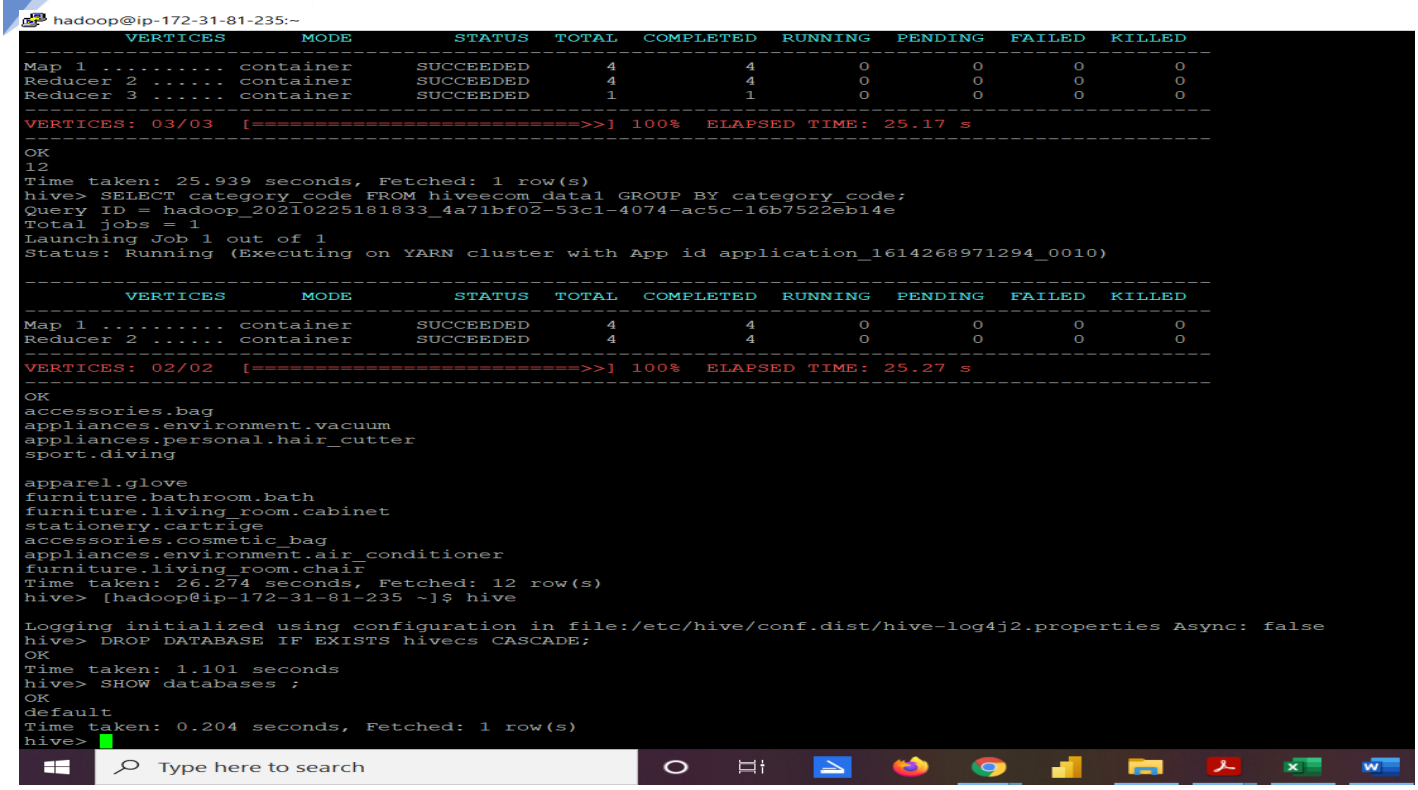
Clearing Up Resources

1. Dropping tables & database

Finally, after getting all the insights needed, dropped the database & its tables in one go by using cascade keyword with drop databases statement.

Command:

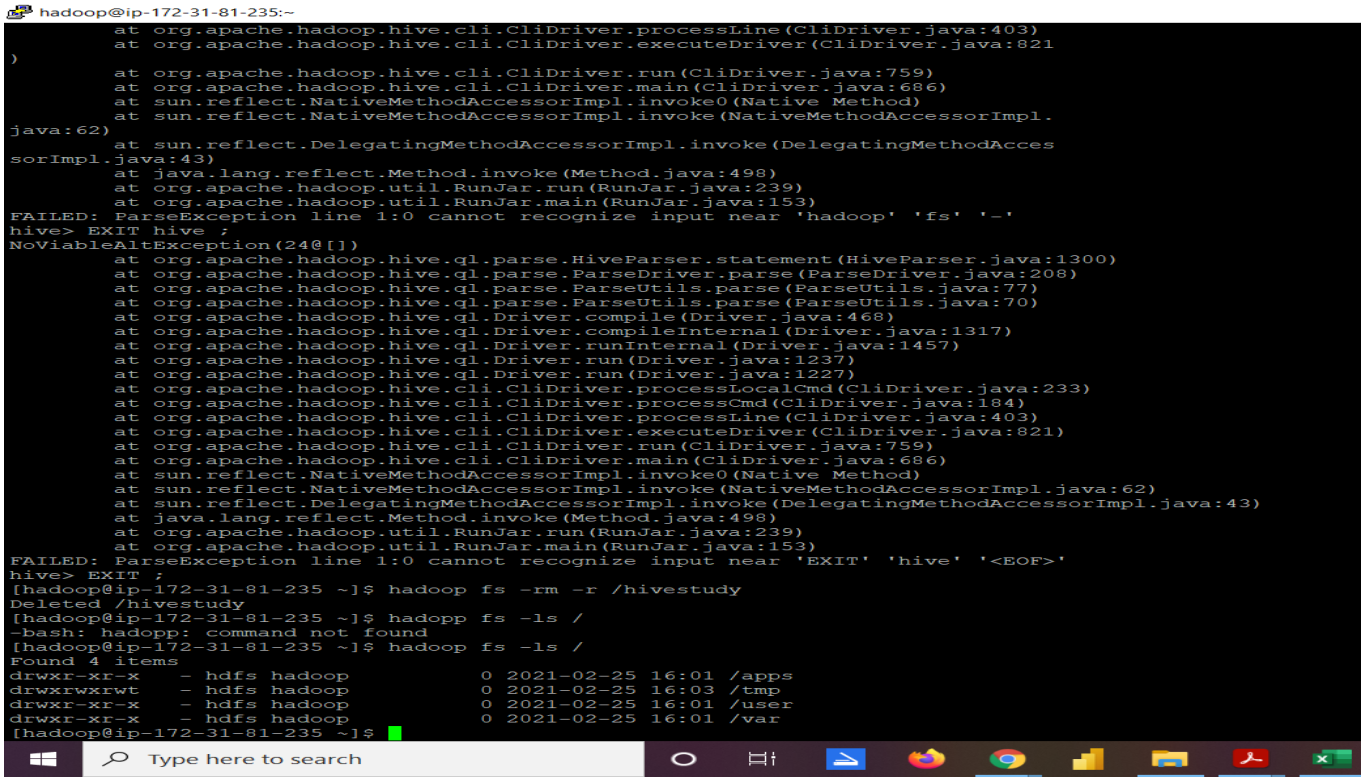
DROP DATABASE IF EXISTS hivecs CASCADE;



2. Exiting hive

Command:

EXIT ;

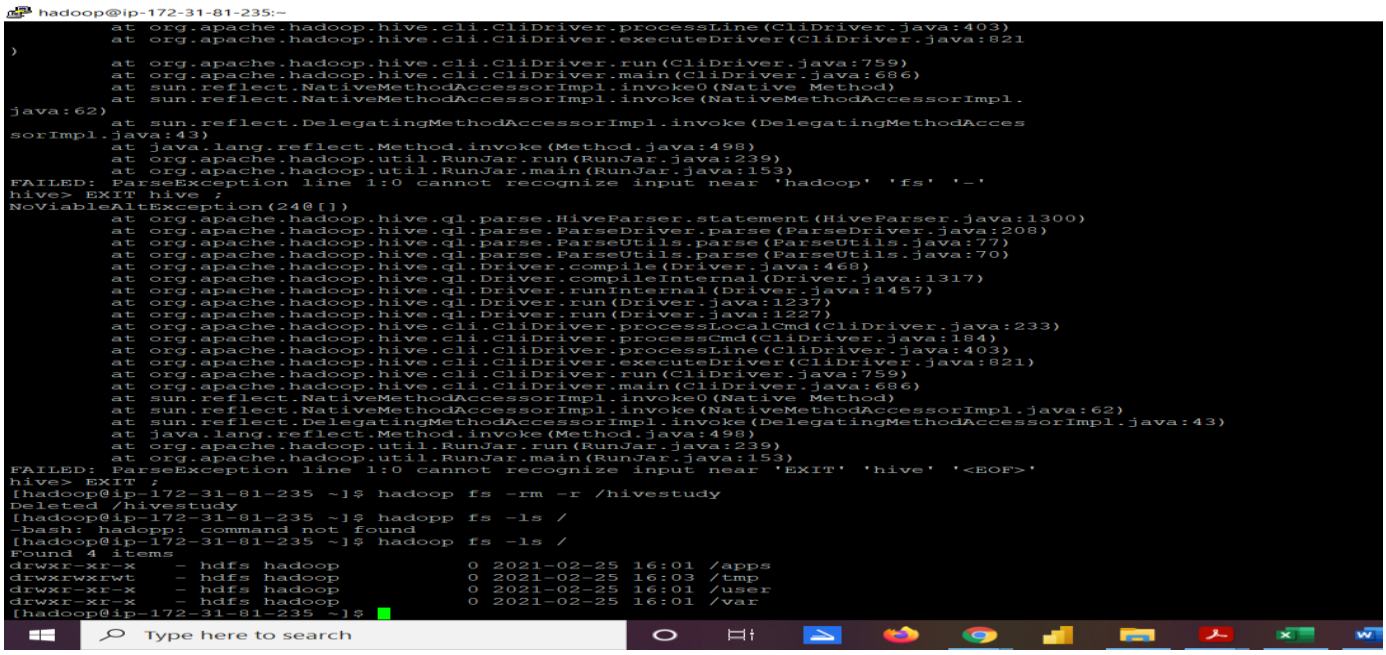


3. Deleting directory

Cleaned up HDFS storage as well by deleting the folder created in which all the data was stored.

Command:

hadoop fs -rm -r /hivestudy



4. Cluster Terminated

Once exited from the SSH terminal, terminated the EMR cluster which was created.