HIVE CASE STUDY

Click-Stream-Data - Cosmetic Company

* Abhirami S

Problem Synopsis: 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. 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.

Our Case Study Objective:

To extract data and gather insights from a real-life data set of an e-commerce company, using AWS EMR and S3, and Hadoop and Hive systems.

Steps Involved:

1. [Launching an EMR Cluster](#_1._Launching_an)
2. [Loading the data into S3 and ingesting the data to HDFS](#_Loading_the_data_1)
3. [Creating Hive Schema & Optimization of tables](#_Creating_Hive_Schema)
4. [Querying – Assignment Questions](#_Querying_–_Assignment)
5. [Demonstration - Improvement of performance after using optimization on tables](#_Demonstration_-_Improvement)
6. [Cleaning-up](#_Cleaning-up)

# Launching an EMR Cluster

When launching an EMR Cluster the following steps were followed:

[1.1 Key Pair Creation](#_Key_Pair_Creation)

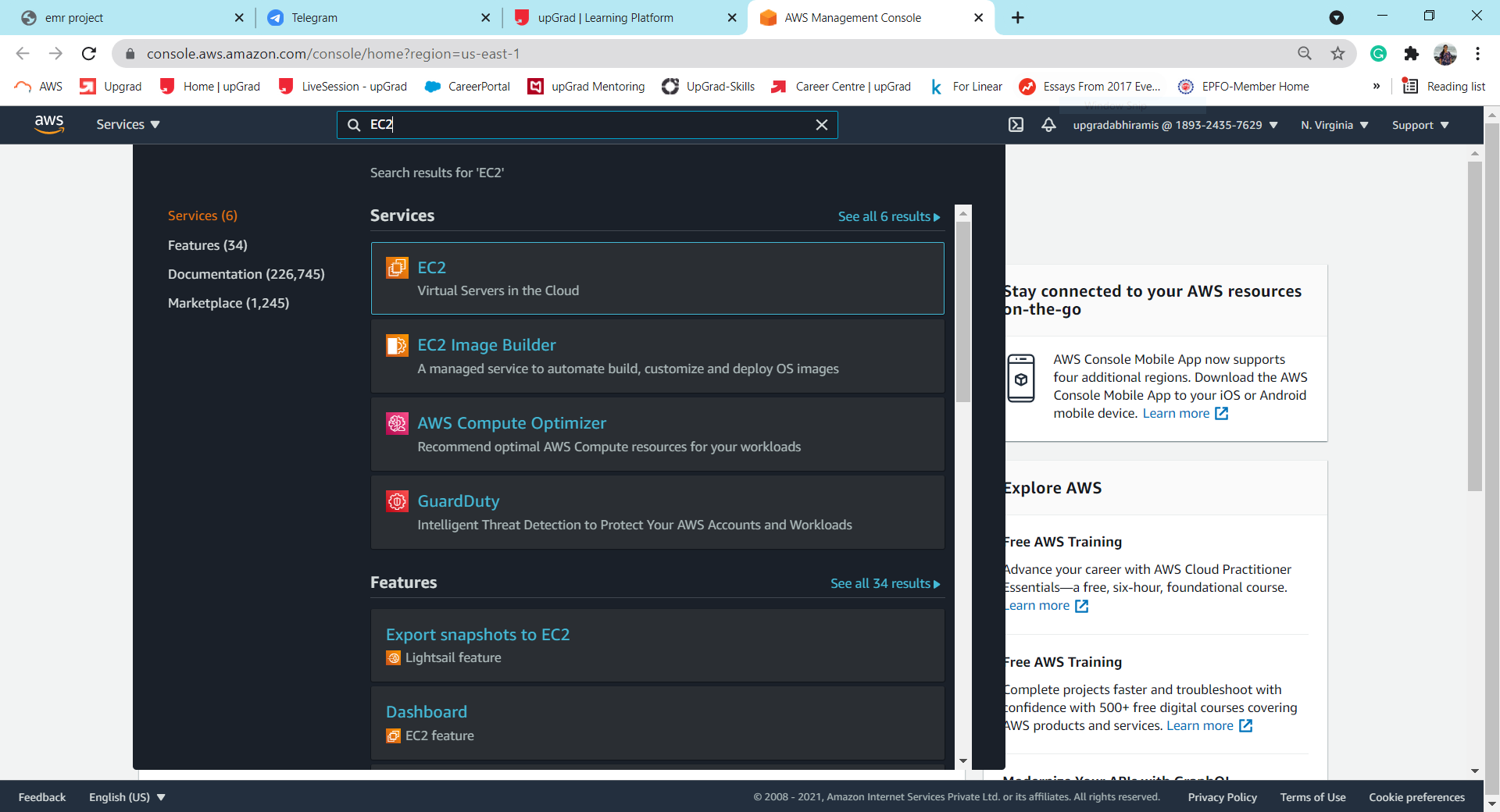
[1.2 Configuring & Launching an EMR Cluster](#_Configuring_&_Launching)

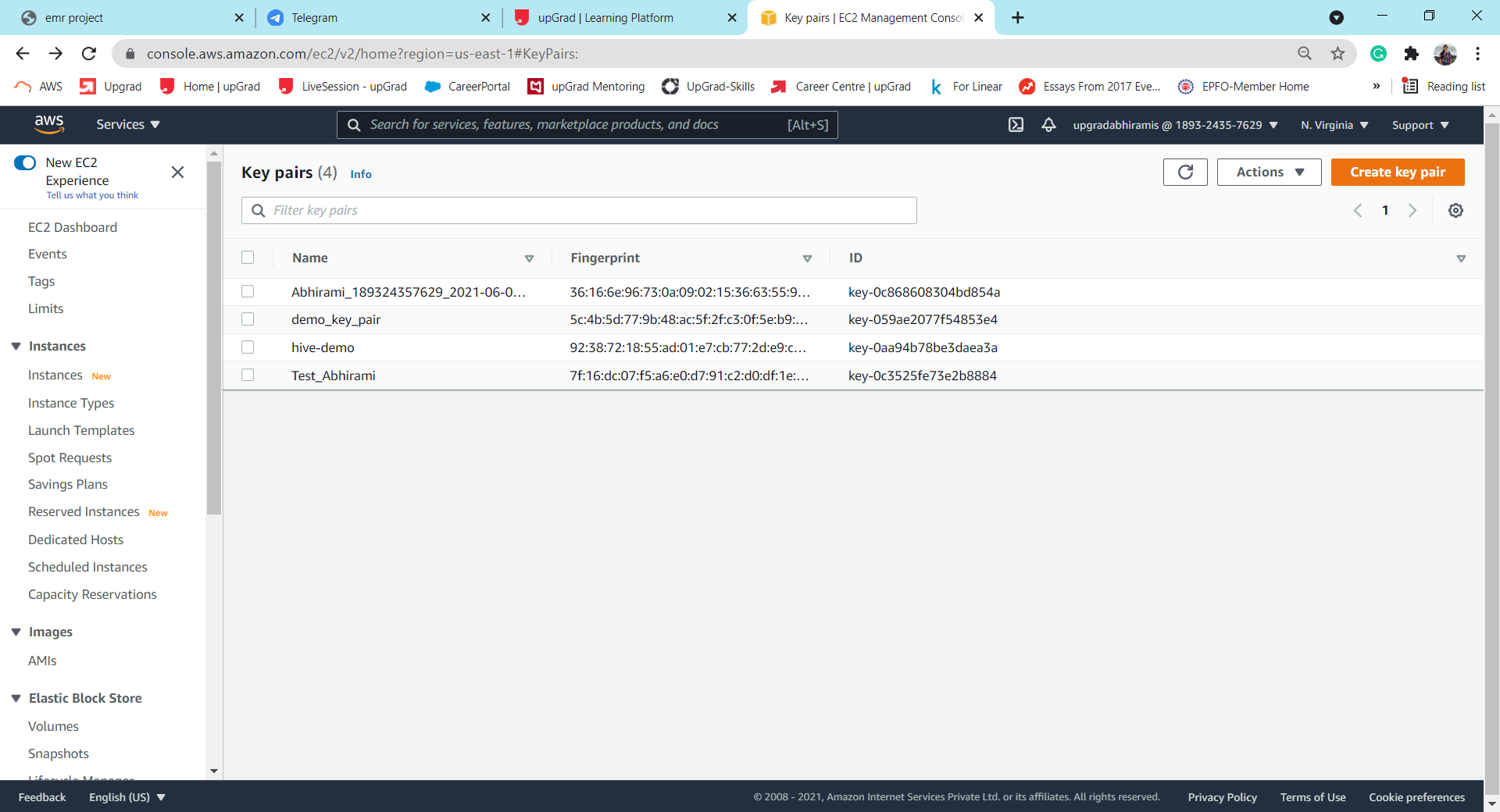
[1.3 Connecting to the Master Node using SSH](#_Connecting_to_the)

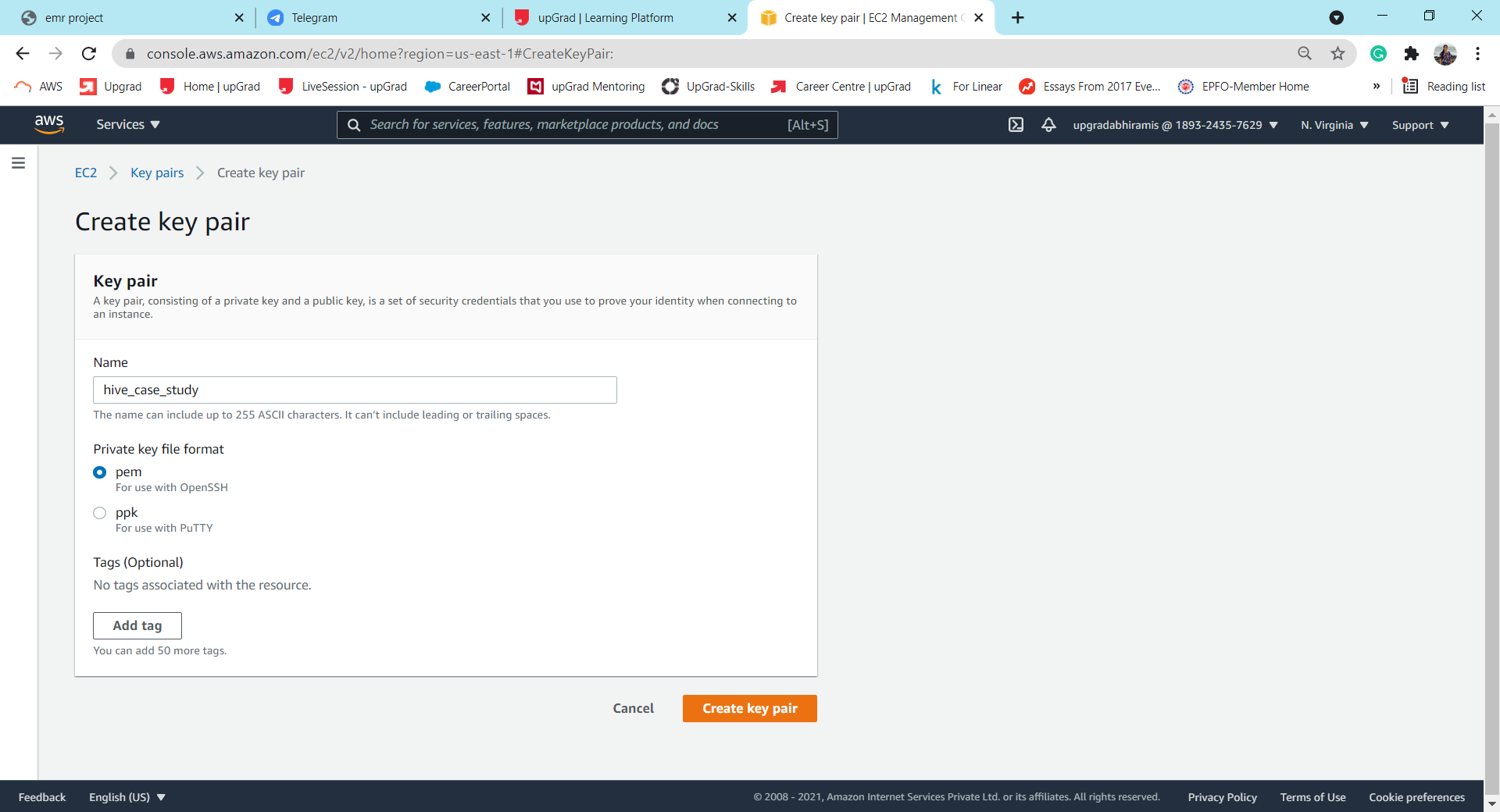
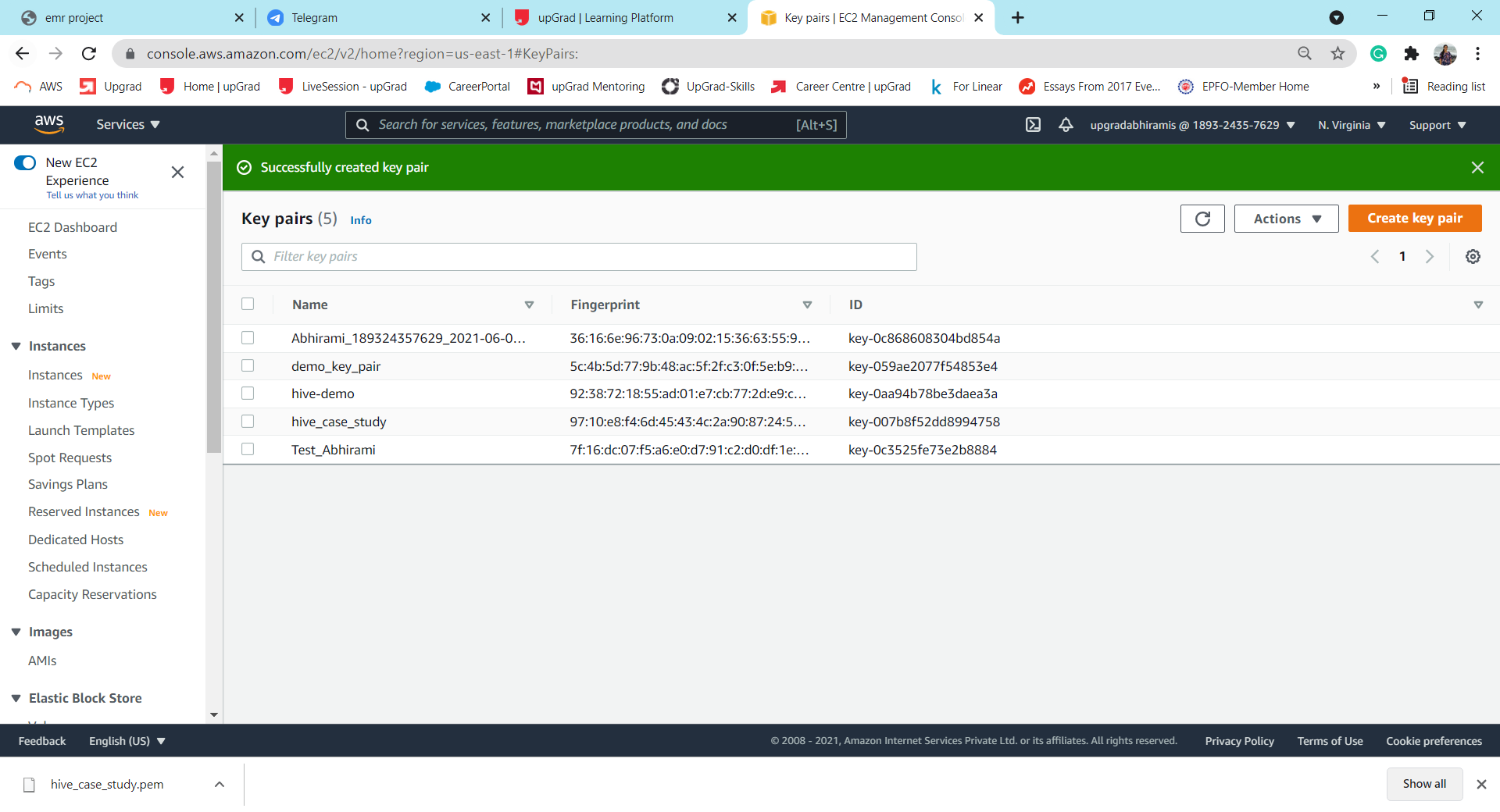
[1.4 Master Node connected & Hive shell launched](#_Master_Node_connected)

## **Key Pair Creation**

Before initiating an EMR Cluster, we need to create an EC2 key-pair file and download it as .pem file.



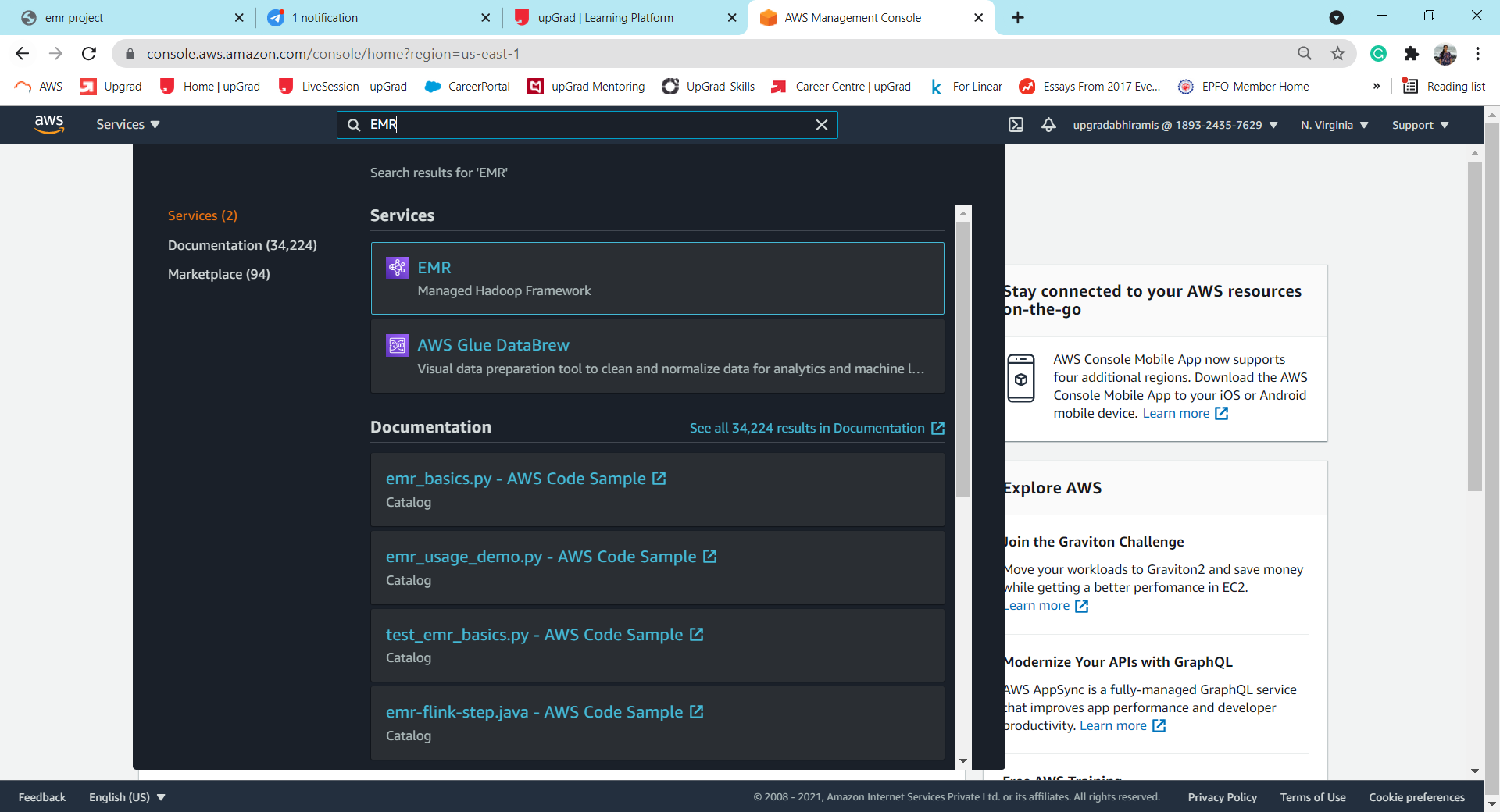


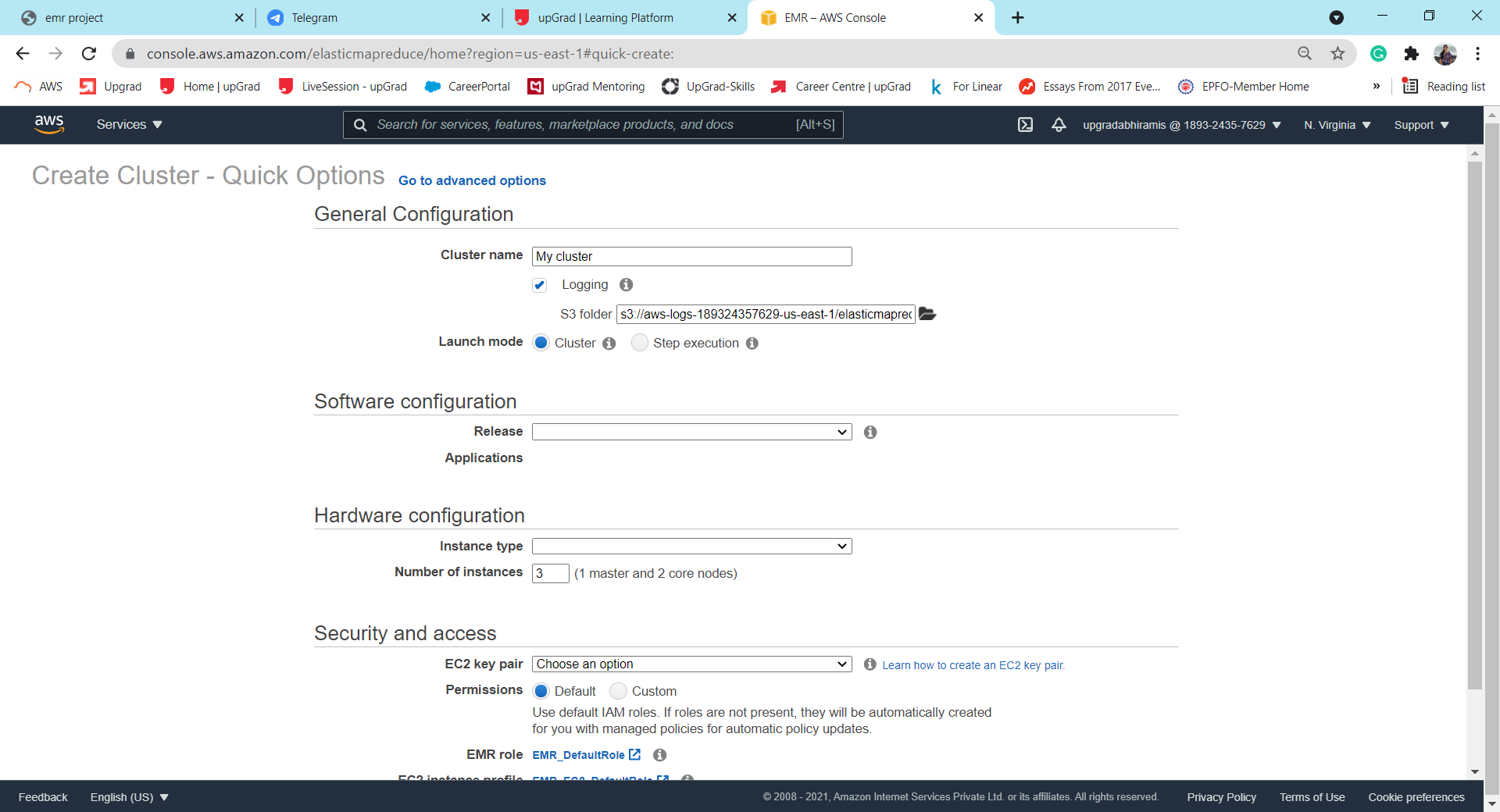
 

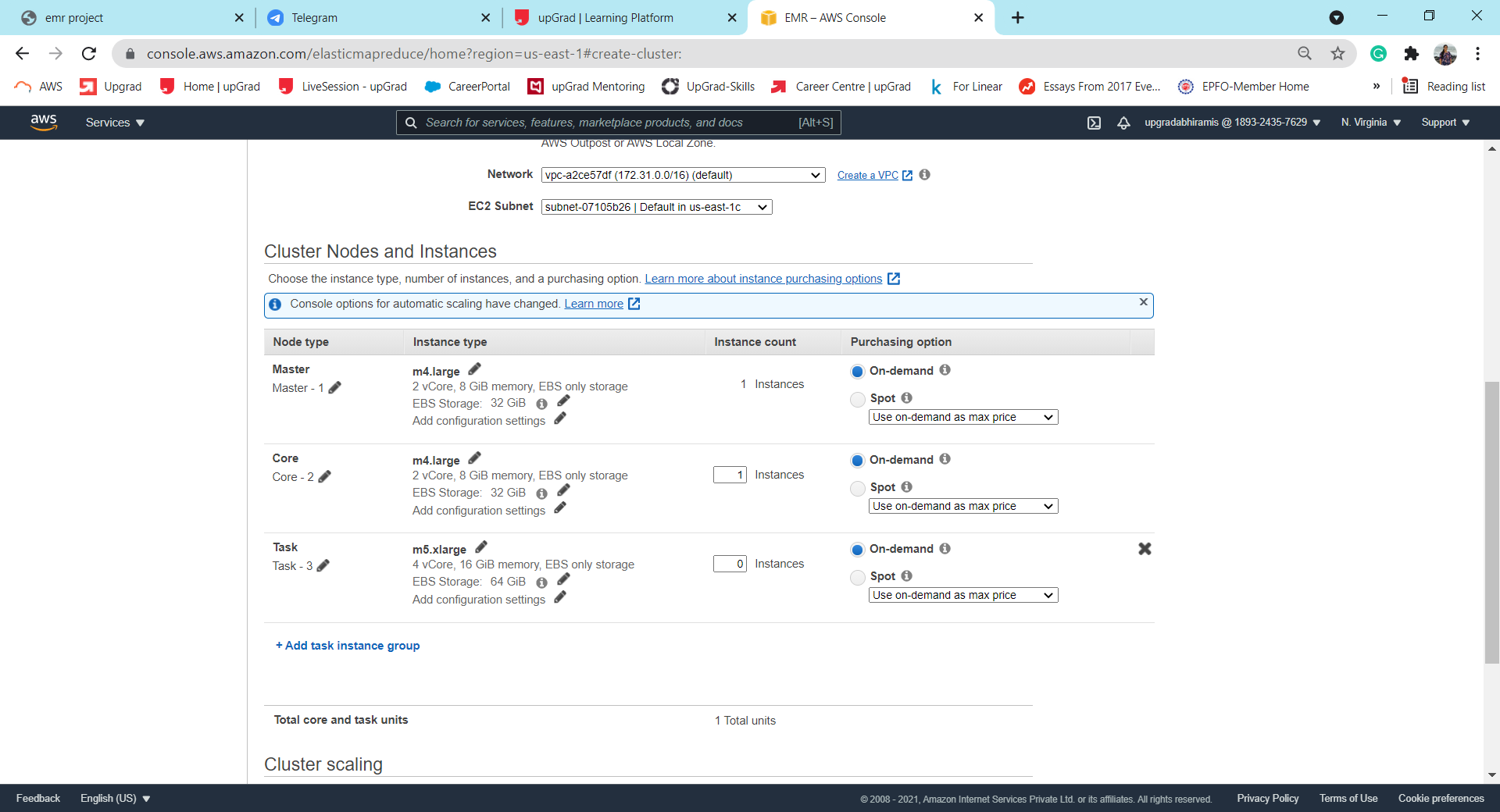
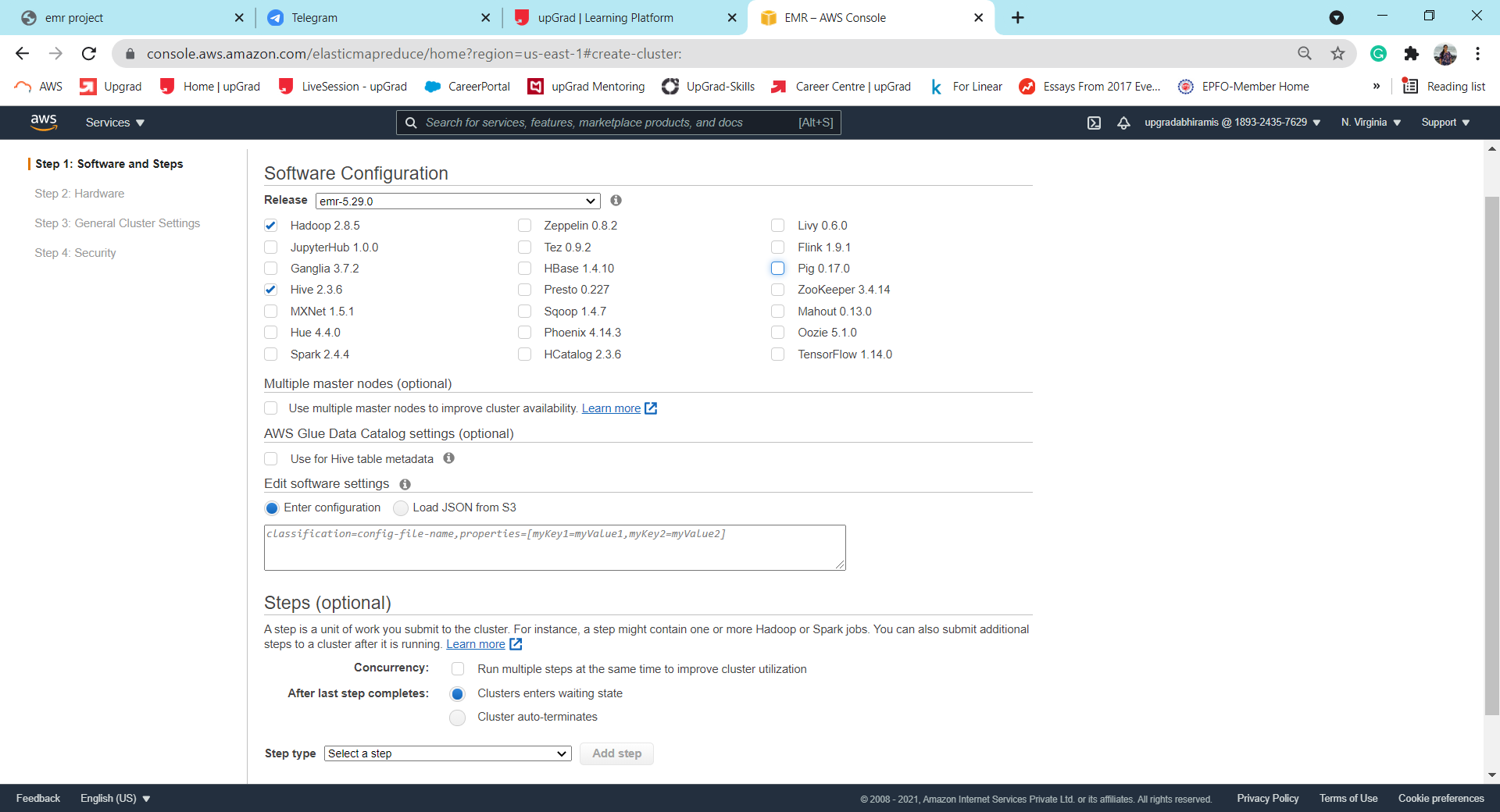
We have completed the creation of key-value pair. Now we move to creating, configuring and launching the EMR Cluster. We use this private key to securely connect to the EMR cluster through SSH.

## **Configuring & Launching an EMR Cluster**

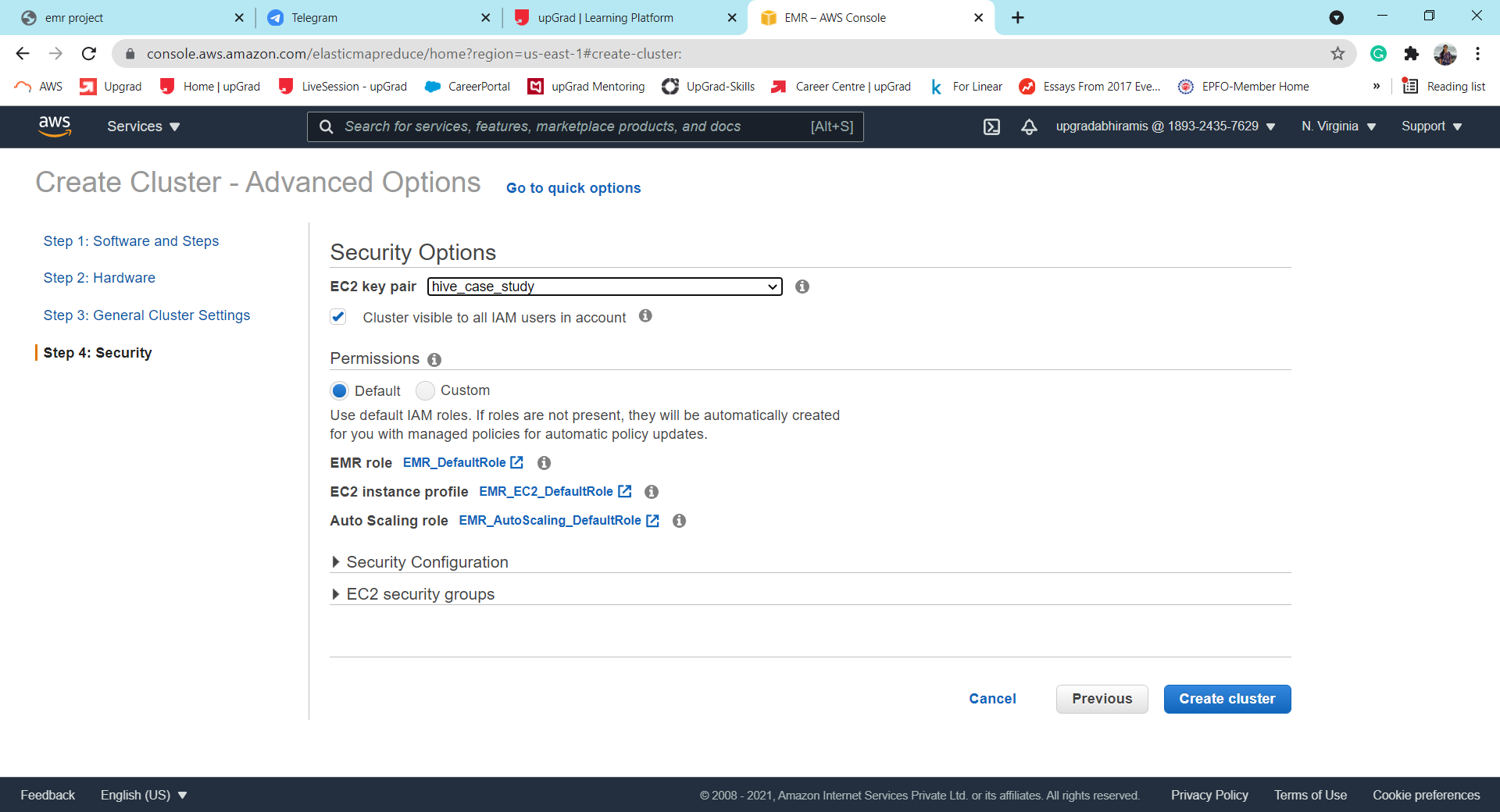
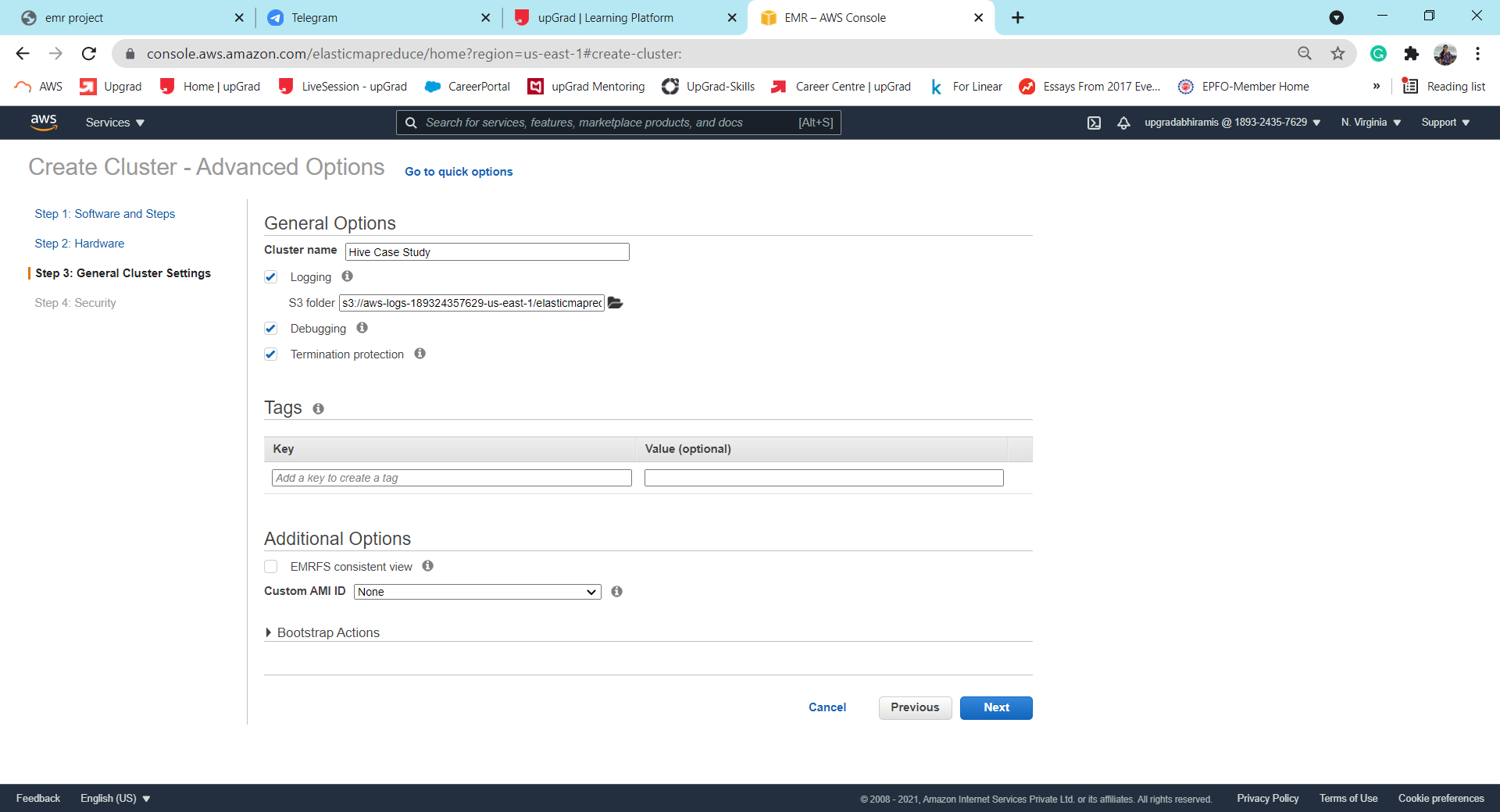
AWS EMR is a managed hadoop framework on which the Hadoop framework will run.





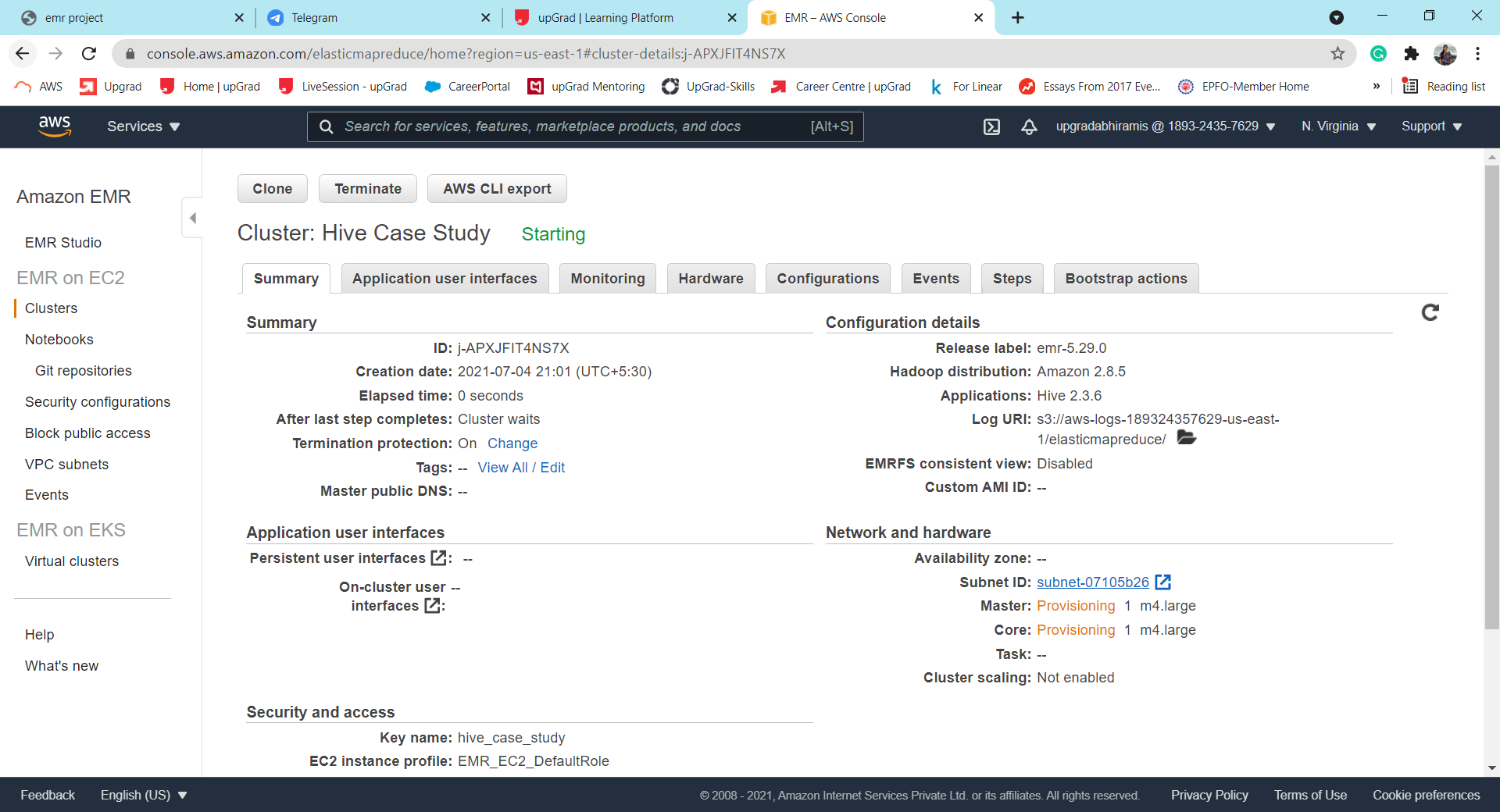


We have chosen emr-5.29.0 release for this case study (as suggested).

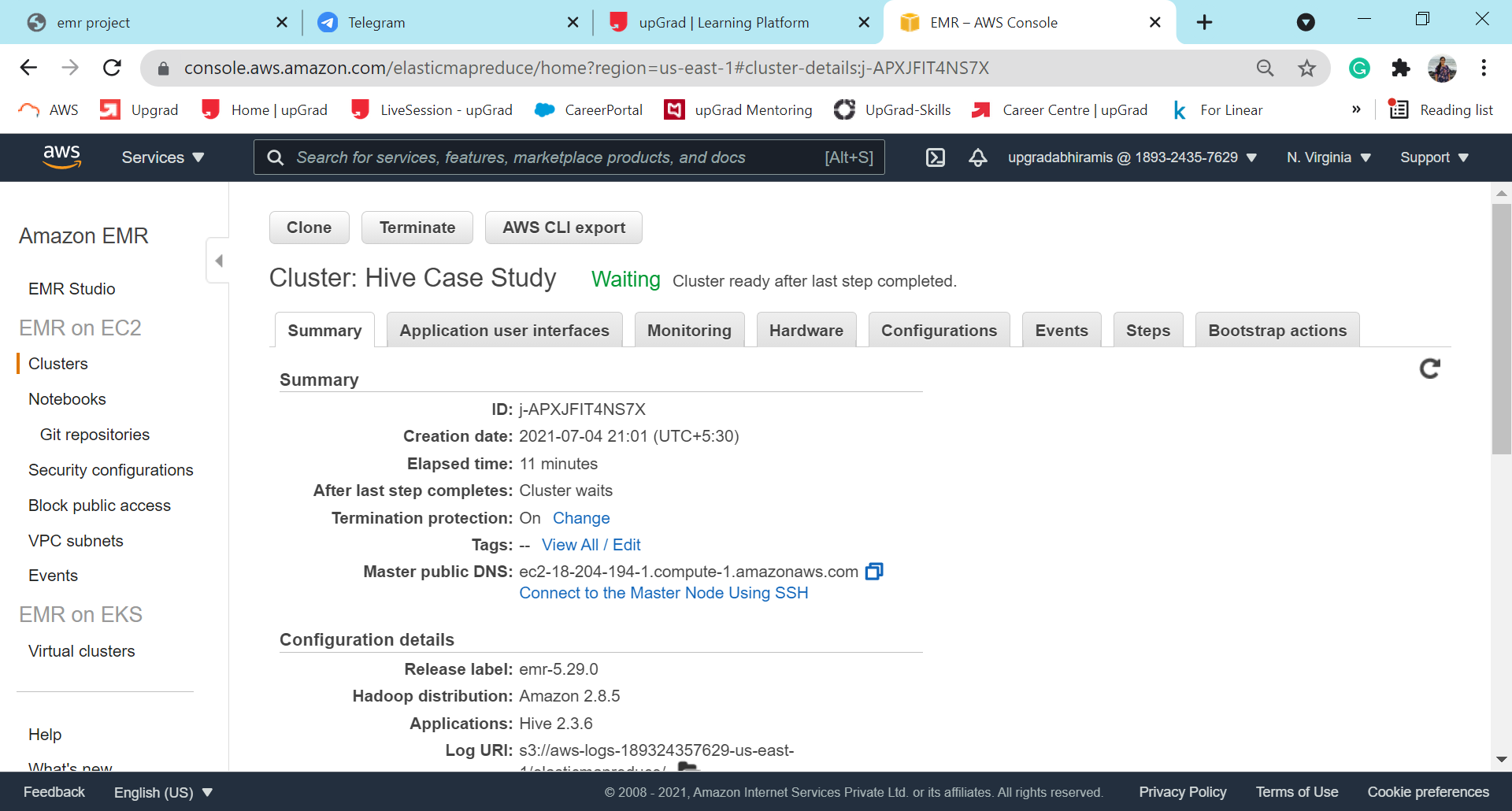


We select the “hive\_case\_study” private key created earlier.

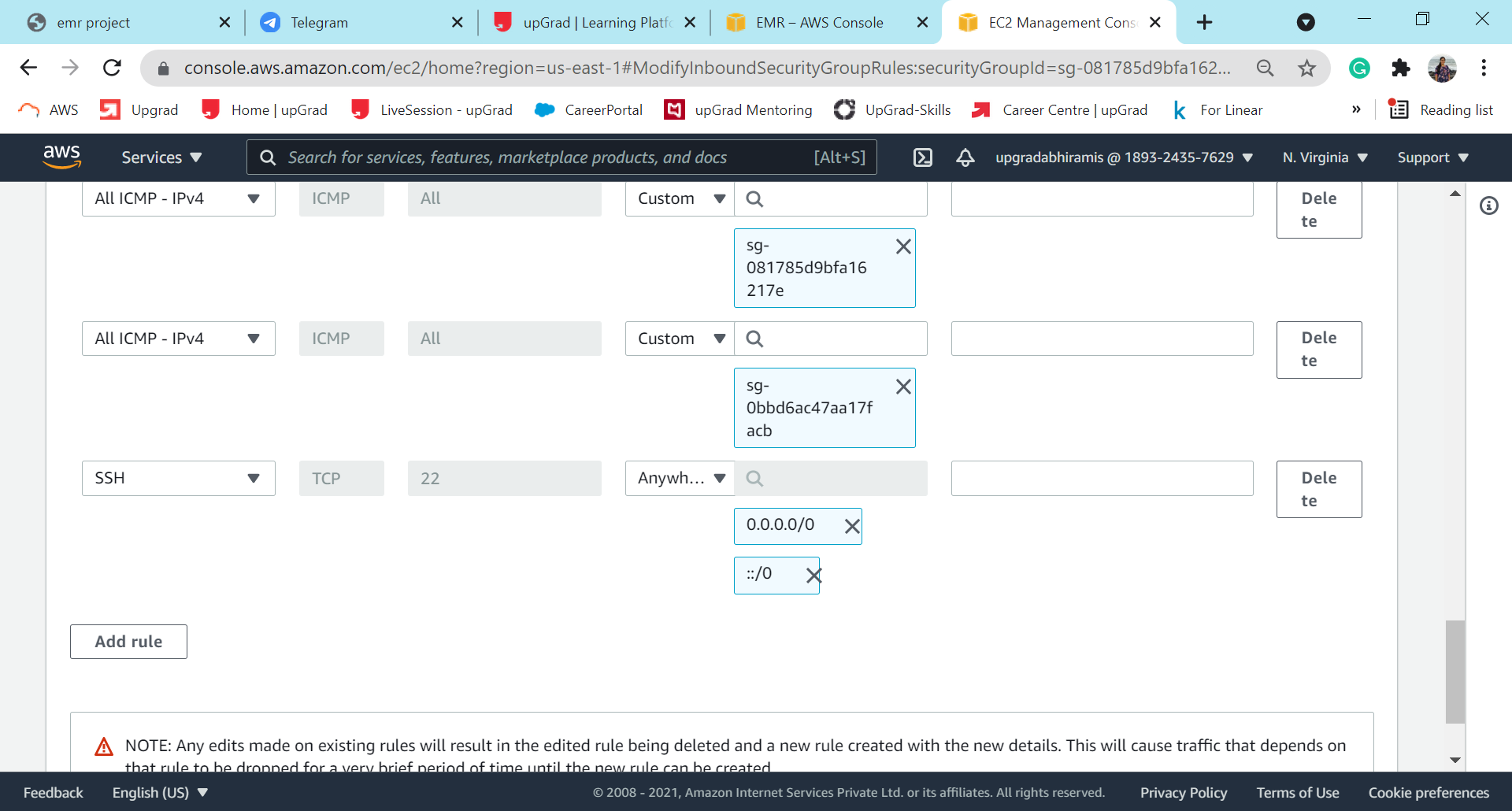
We can see below that the cluster has started ‘lauching’. This will take about 15 mins to be ready.



Once ready, the cluster it displays the message “Waiting”.



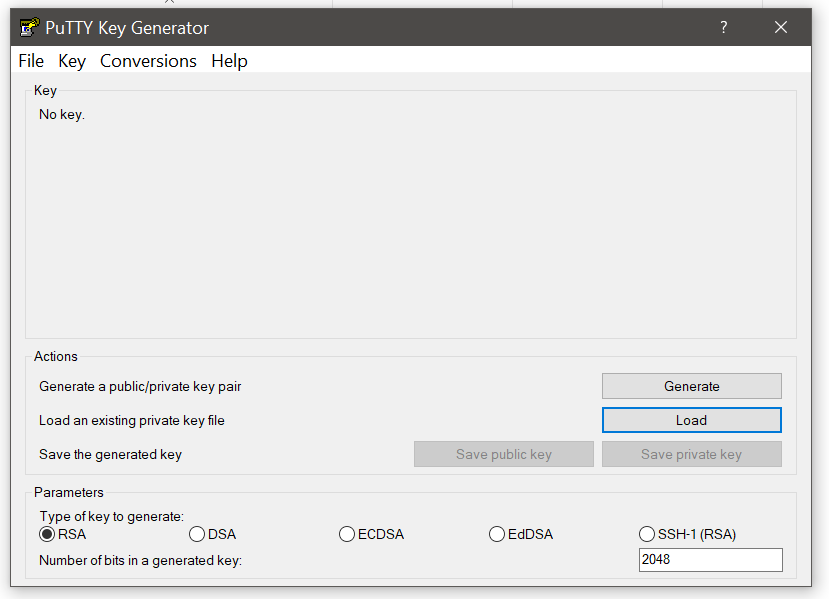
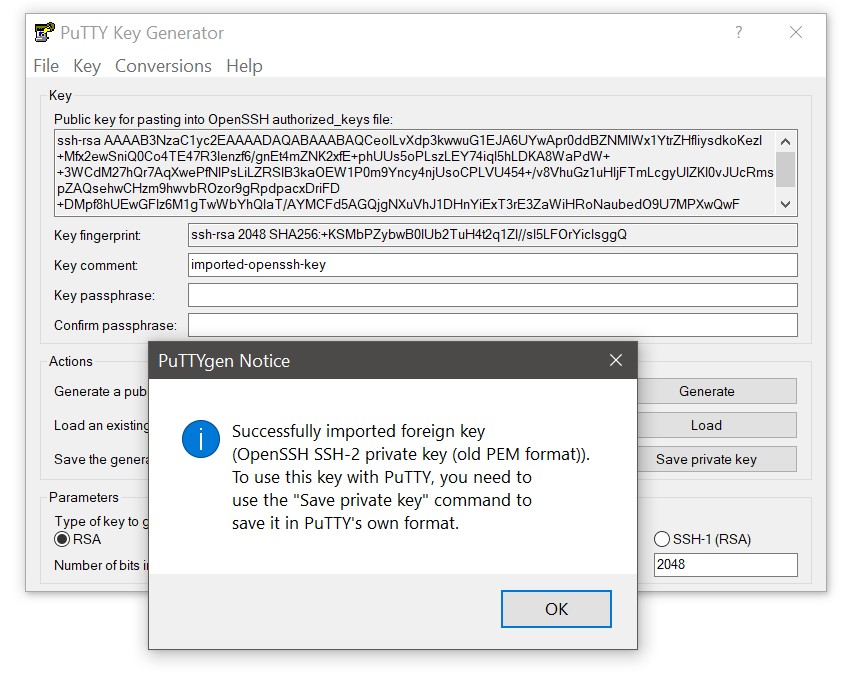
Before moving to the connecting to SSH. We need to ensure that the port is open to establish a connection. For this, under ‘Security groups for Master node’ , we select the ‘Security Group ID’ from the landing page. Here, we add an additional inbound rule as below.



## **Connecting to the Master Node using SSH (Windows)**

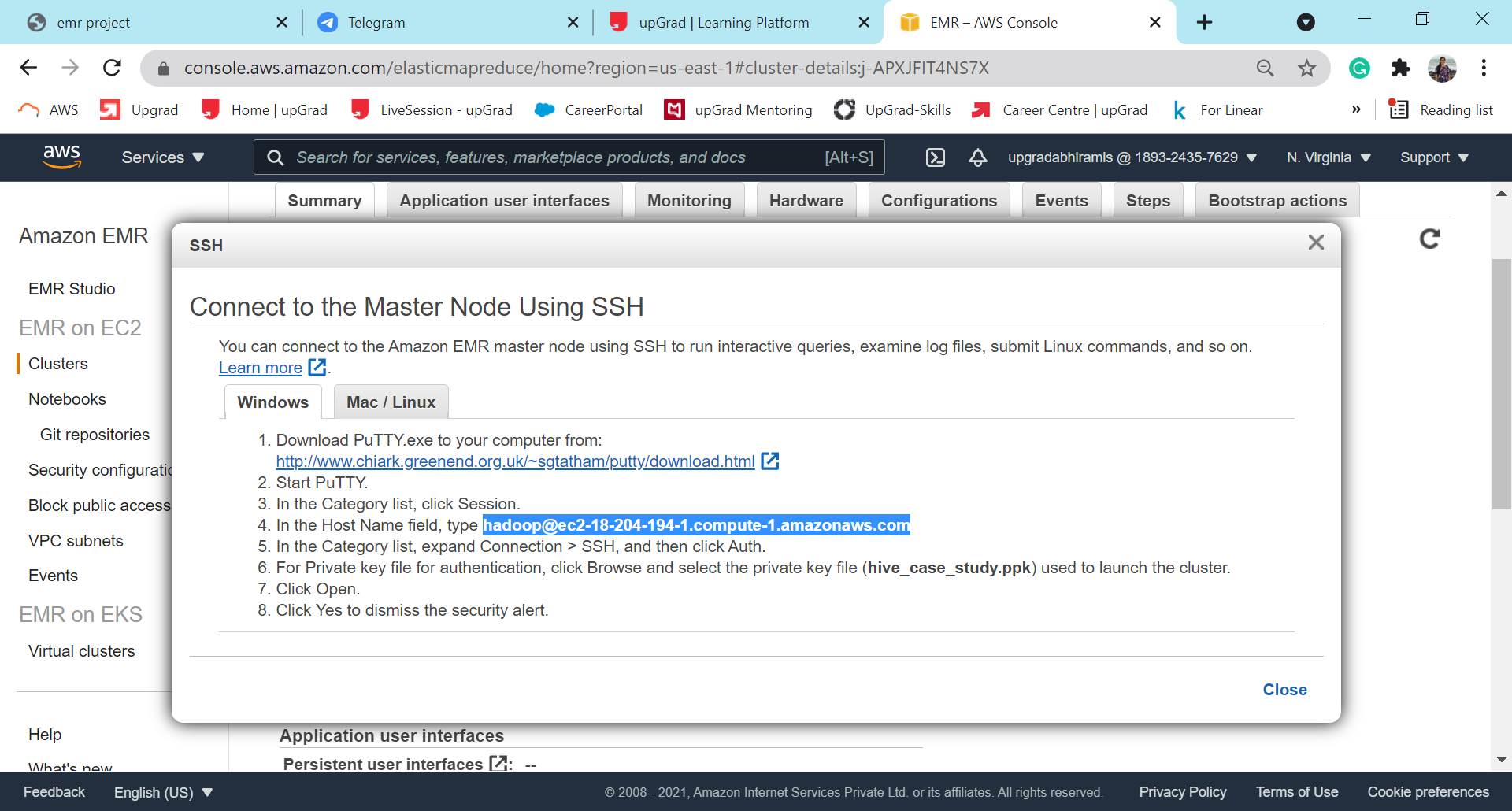
We have used two programs for this; PuTTY & PuTTYgen

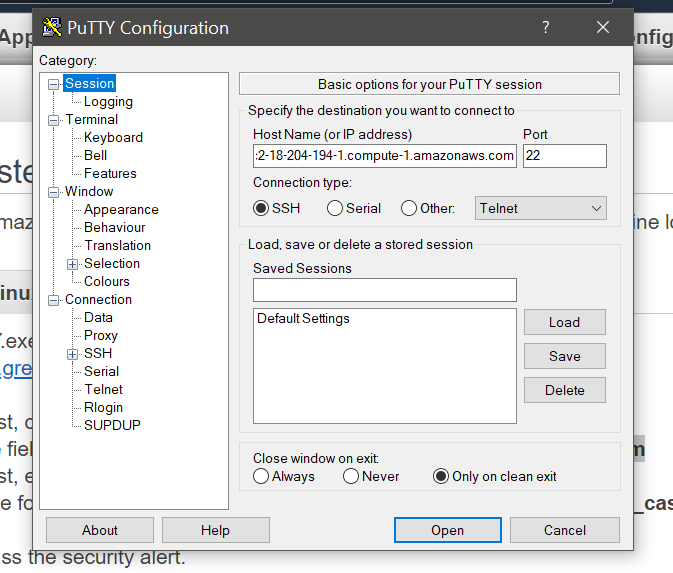
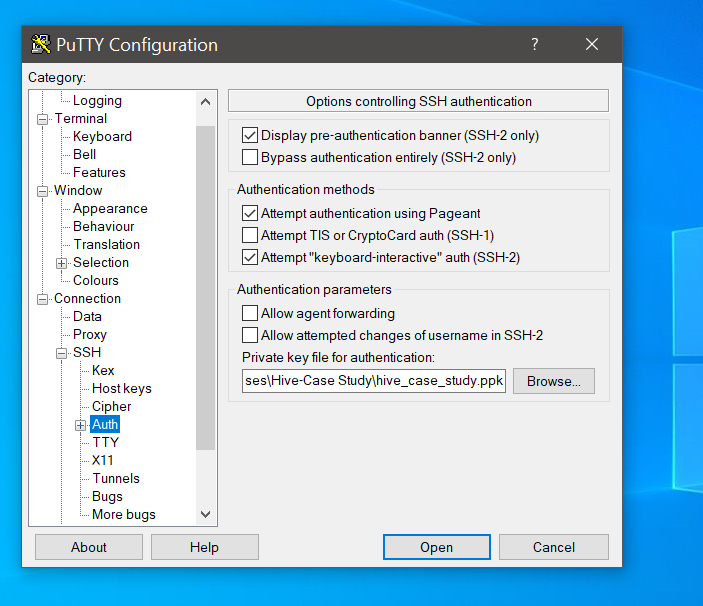
We first open PuTTYgen and load our private key(hive\_case\_study) and save it as a .ppk file.

We save the .ppk at the desired location.

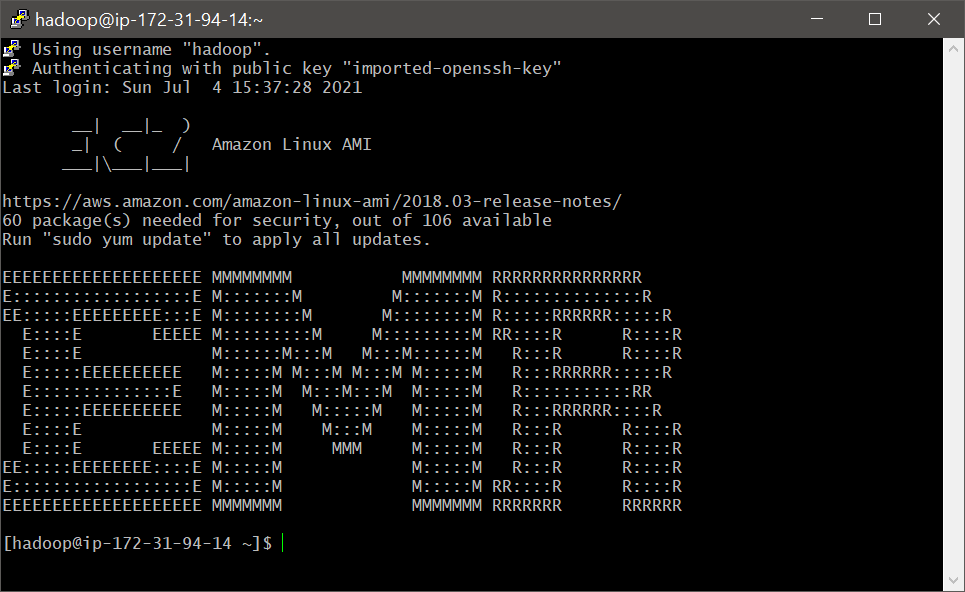
We next open PuTTY to connect to the Master Node. We copy the Public DNS address from our EMR Cluster.



We input the Public DNS Address & load the private key in the PuTTY window.

## **Master Node connected & Hive shell launched**



# Loading the data into S3 and ingesting the data to HDFS

This was done in the following steps:

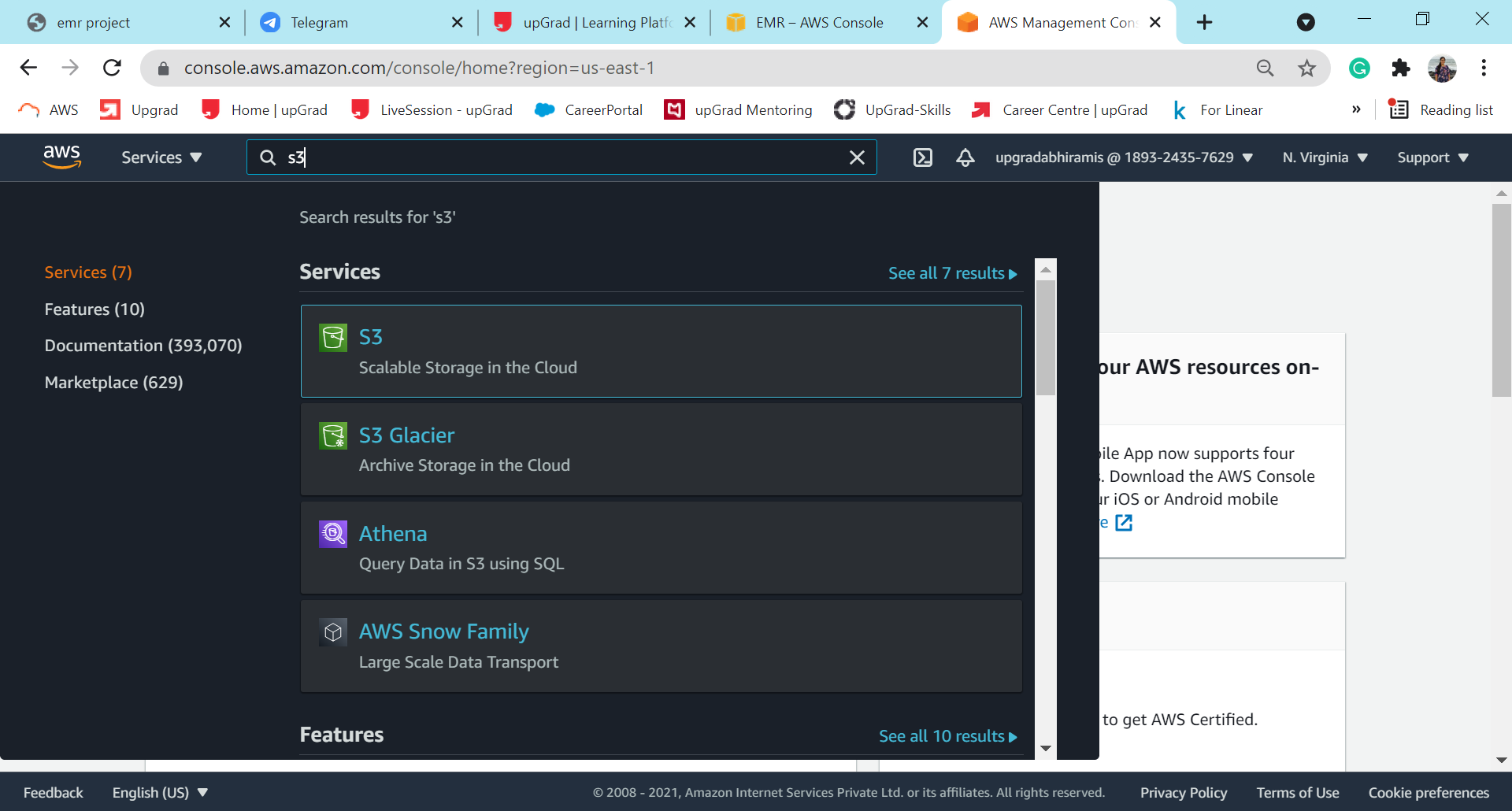
[2.1 Uploading databases to S3 Bucket](#_Uploading_databases_to)

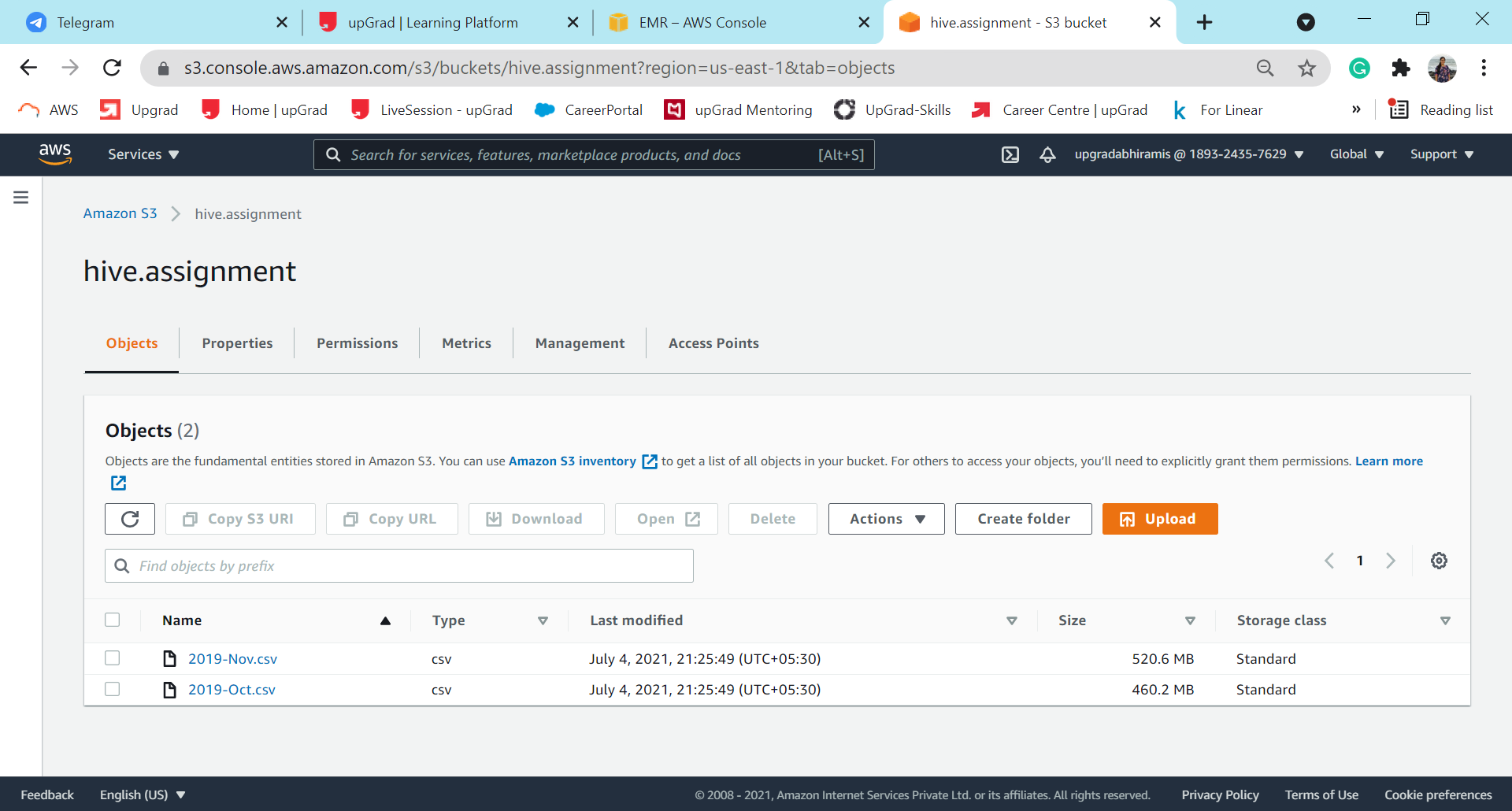
[2.2 Creating a temporary directory in HDFS](#_Creating_a_temporary)

[2.3 Loading the data to HDFS](#_Loading_the_data)

## **Uploading databases to S3 Bucket**

We first create a bucket in S3 to store the files. We have named this ‘hive.assignment’.





## **Creating a temporary directory in HDFS**

- Checking the directories already present in HDFS.

Command: hadoop fs -ls /

Output:

|  |
| --- |
| [hadoop@ip-172-31-91-68 ~]$ hadoop fs -ls /  Found 4 items  drwxr-xr-x - hdfs hadoop 0 2021-07-04 16:37 /apps  drwxrwxrwt - hdfs hadoop 0 2021-07-04 16:38 /tmp  drwxr-xr-x - hdfs hadoop 0 2021-07-04 16:37 /user  drwxr-xr-x - hdfs hadoop 0 2021-07-04 16:37 /var |



Comment: The built-in directories present in HDFS are displayed.

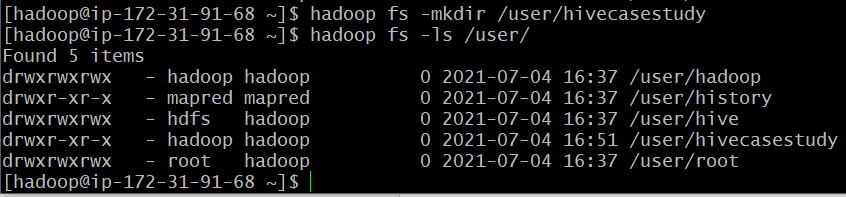
- Creating a temporary directory

Command: hadoop fs -mkdir /user/hivecasestudy

hadoop fs -ls /user/

Output:

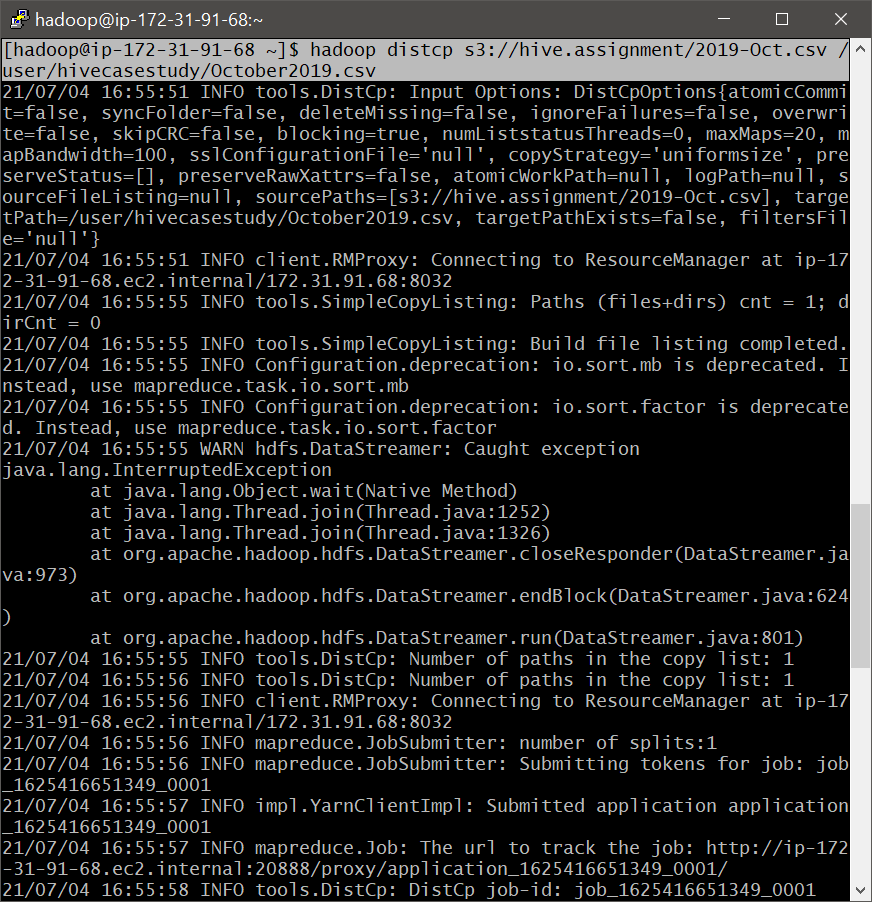
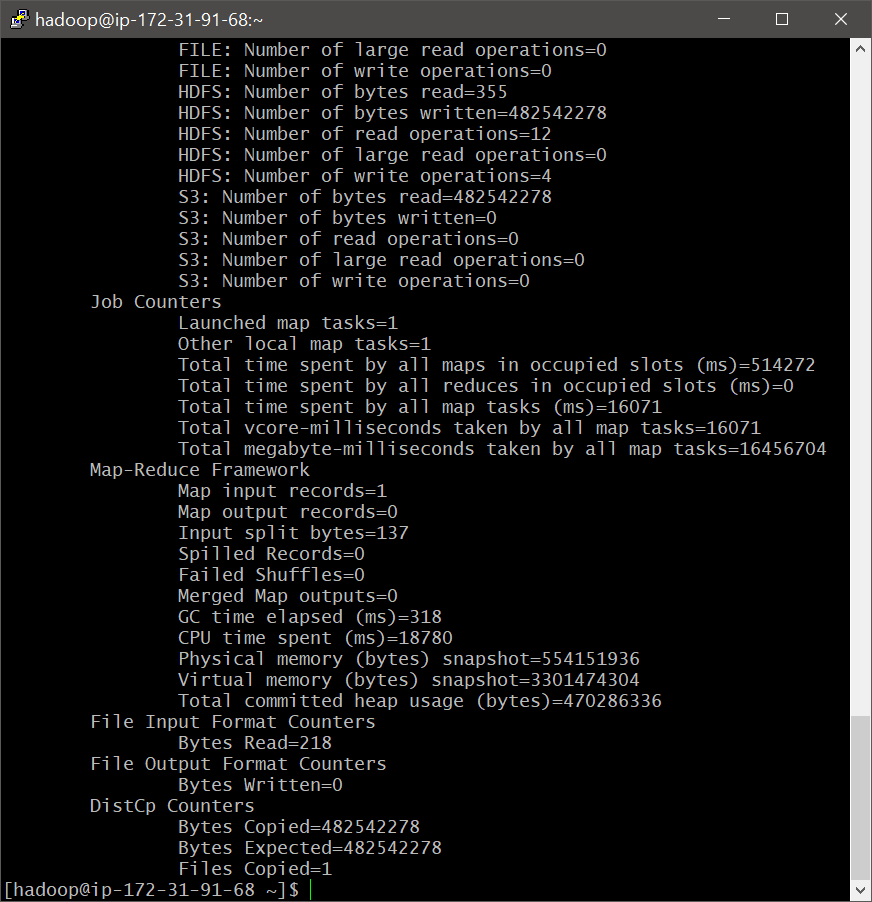
|  |
| --- |
| [hadoop@ip-172-31-91-68 ~]$ hadoop fs -mkdir /user/hivecasestudy  [hadoop@ip-172-31-91-68 ~]$ hadoop fs -ls /user/  Found 5 items  drwxrwxrwx - hadoop hadoop 0 2021-07-04 16:37 /user/hadoop  drwxr-xr-x - mapred mapred 0 2021-07-04 16:37 /user/history  drwxrwxrwx - hdfs hadoop 0 2021-07-04 16:37 /user/hive  drwxr-xr-x - hadoop hadoop 0 2021-07-04 16:51 /user/hivecasestudy  drwxrwxrwx - root hadoop 0 2021-07-04 16:37 /user/root |



## **Loading the data to HDFS**

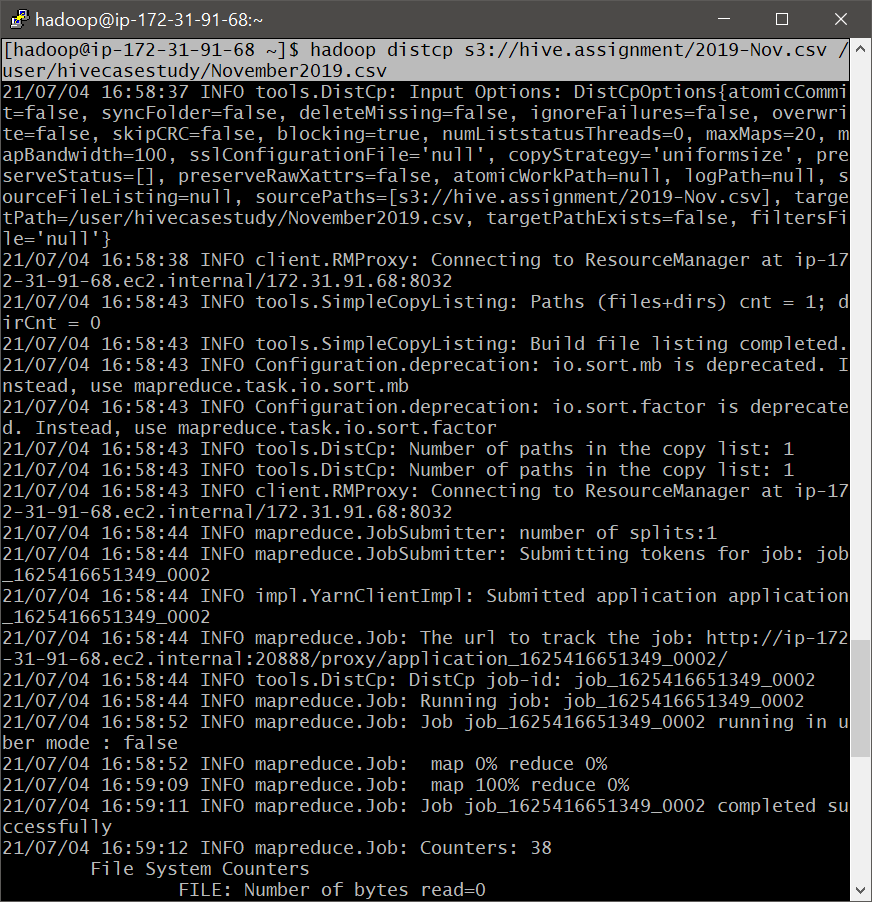
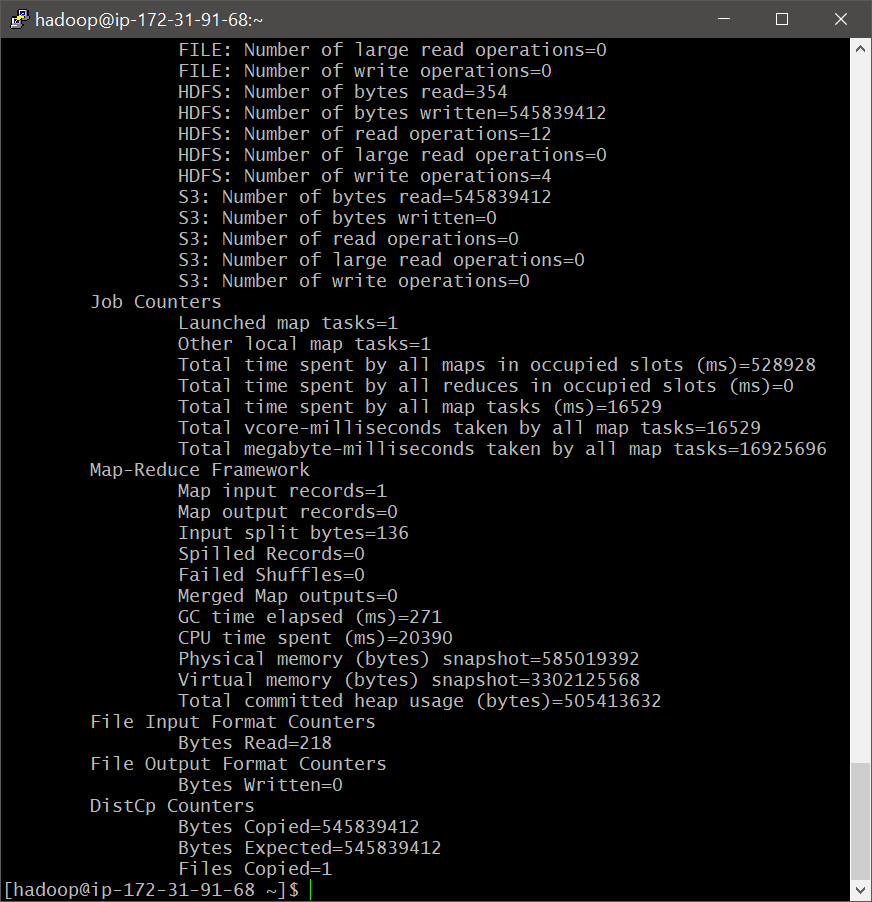
Command: hadoop distcp s3://hive.assignment/2019-Oct.csv /user/hivecasestudy/October2019.csv

Output:

Command: hadoop distcp s3://hive.assignment/2019-Nov.csv /user/hivecasestudy/November2019.csv

Output:

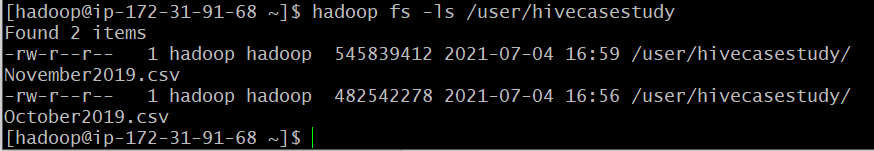
Comment: We have successfully loaded both the files into HDFS.

- Checking the successful loading of data files

Command: hadoop fs -ls /user/hivecasestudy

Output:

|  |
| --- |
| [hadoop@ip-172-31-91-68 ~]$ hadoop fs -ls /user/hivecasestudy  Found 2 items  -rw-r--r-- 1 hadoop hadoop 545839412 2021-07-04 16:59 /user/hivecasestudy/ November2019.csv  -rw-r--r-- 1 hadoop hadoop 482542278 2021-07-04 16:56 /user/hivecasestudy/ October2019.csv |



Comment: Now, that we can see the files present. We can move to Hive to create the schema and start the querying.

# Creating Hive Schema & Optimization of tables

We have broadly followed the following steps during creation of hive schema and tables.

[3.1 Create a table](#_Create_a_table)

[3.2 Enabling easy view-ability](#_3.2_Enabling_easy)

[3.3 Optimization of tables - Partitioning and Bucketing](#_3.3_Optimization_of)

[3.4 Inserting data into partitioned and bucketed table](#_3.4_Inserting_data)

## **Create a table**

We first enter into the Hive Shell.

Command: hive

Output:



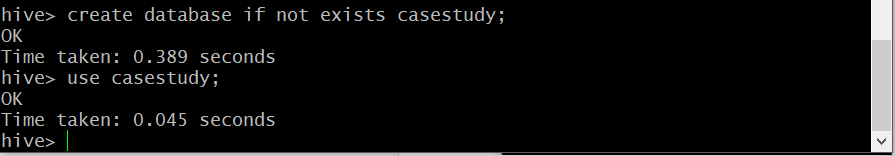
- Creating a new Database for the casestudy

Command: create database if not exists casestudy;

use casestudy;

Output:

|  |
| --- |
| hive> create database if not exists casestudy;  OK  Time taken: 0.389 seconds  hive> use casestudy;  OK  Time taken: 0.045 seconds |



- Creating a external table to load the data

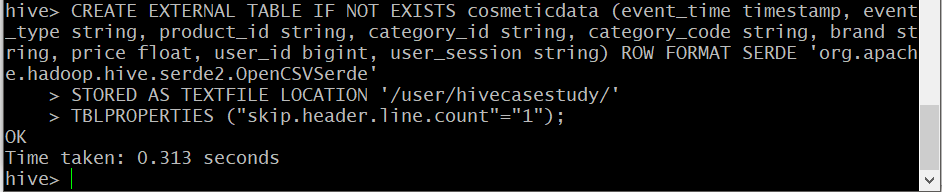
Command: CREATE EXTERNAL TABLE IF NOT EXISTS cosmeticdata (event\_time timestamp, event\_type string, product\_id string, category\_id string, category\_code string, brand string, price float, user\_id bigint, user\_session string) ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'

STORED AS TEXTFILE LOCATION '/user/hivecasestudy/'

TBLPROPERTIES ("skip.header.line.count"="1");

Output:

|  |
| --- |
| hive> CREATE EXTERNAL TABLE IF NOT EXISTS cosmeticdata (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 '/user/hivecasestudy/'  > TBLPROPERTIES ("skip.header.line.count"="1");  OK  Time taken: 0.313 seconds |

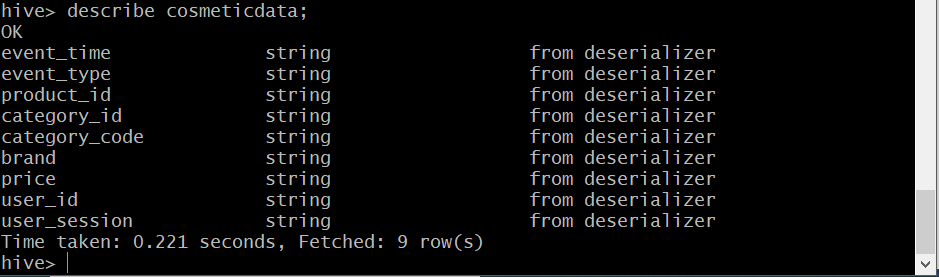


- Checking the table ‘cosmeticdata;

Command: describe cosmeticdata;

Output:

|  |
| --- |
| hive> describe cosmeticdata;  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: 0.221 seconds, Fetched: 9 row(s) |



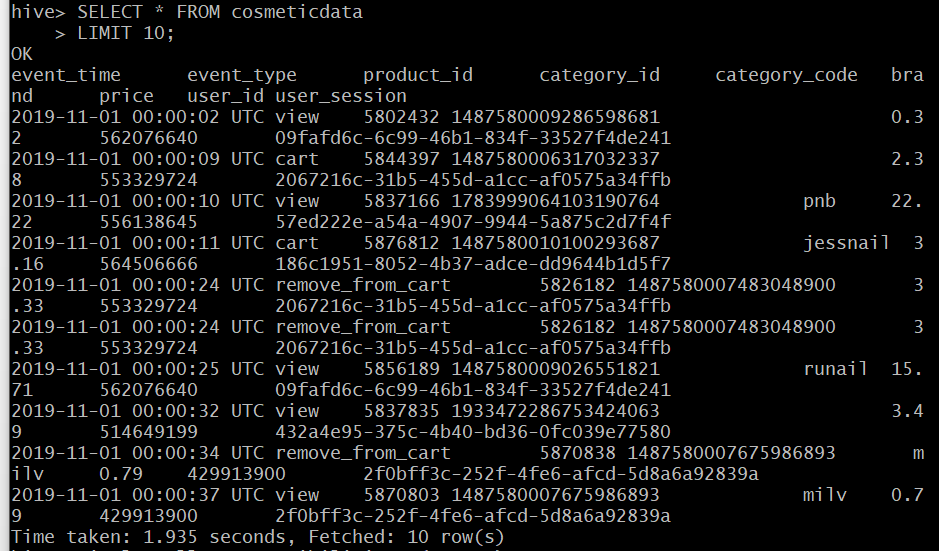
- Checking the successful creation of table and transfer of data into the table

Command: SELECT \* FROM cosmeticdata

LIMIT 10;

Output:

|  |
| --- |
| hive> SELECT \* FROM cosmeticdata  > LIMIT 10;  OK  event\_time event\_type product\_id category\_id category\_code brand price user\_id user\_session  2019-11-01 00:00:02 UTC view 5802432 1487580009286598681 0.32 562076640 09fafd6c-6c99-46b1-834f-33527f4de241  2019-11-01 00:00:09 UTC cart 5844397 1487580006317032337 2.38 553329724 2067216c-31b5-455d-a1cc-af0575a34ffb  2019-11-01 00:00:10 UTC view 5837166 1783999064103190764 pnb 22.22 556138645 57ed222e-a54a-4907-9944-5a875c2d7f4f  2019-11-01 00:00:11 UTC cart 5876812 1487580010100293687 jessnail 3.16 564506666 186c1951-8052-4b37-adce-dd9644b1d5f7  2019-11-01 00:00:24 UTC remove\_from\_cart 5826182 1487580007483048900 3.33 553329724 2067216c-31b5-455d-a1cc-af0575a34ffb  2019-11-01 00:00:24 UTC remove\_from\_cart 5826182 1487580007483048900 3.33 553329724 2067216c-31b5-455d-a1cc-af0575a34ffb  2019-11-01 00:00:25 UTC view 5856189 1487580009026551821 runail 15.71 562076640 09fafd6c-6c99-46b1-834f-33527f4de241  2019-11-01 00:00:32 UTC view 5837835 1933472286753424063 3.49 514649199 432a4e95-375c-4b40-bd36-0fc039e77580  2019-11-01 00:00:34 UTC remove\_from\_cart 5870838 1487580007675986893 milv 0.79 429913900 2f0bff3c-252f-4fe6-afcd-5d8a6a92839a  2019-11-01 00:00:37 UTC view 5870803 1487580007675986893 milv 0.79 429913900 2f0bff3c-252f-4fe6-afcd-5d8a6a92839a  Time taken: 1.935 seconds, Fetched: 10 row(s) |



## **3.2 Enabling easy view-ability**

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

set hive.resultset.use.unique.column.names=false;

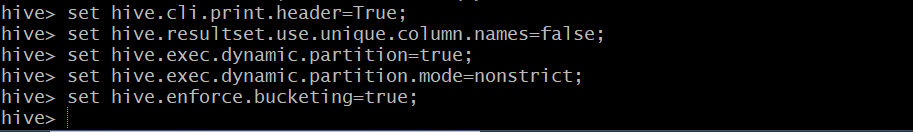
set hive.exec.dynamic.partition=true;

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; |



Comment: The below explanation is provided for the executed commands.

1. To display the column header in output
2. To remove table name before column name separated by ‘.’
3. & 4. To enable Dynamic partitioning

5. To enable Bucketing

## **3.3 Optimization of tables - Partitioning and Bucketing**

We will be partitioning on the attribute “event\_type”. We have chosen this because it has unique set of categorical values and it will give us a definitive number of partitions (4). This will also reduce the load to the Namenode.

We will be bucketing on the attribute “price”. The number of buckets to be created should depend on the data size and the size of HDFS block (default value is 128 MB). Based on the above information and our data size, we have calculated to create a minimum of 8 buckets.

Command: CREATE EXTERNAL TABLE cosmetic\_partbuck (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 8 BUCKETS

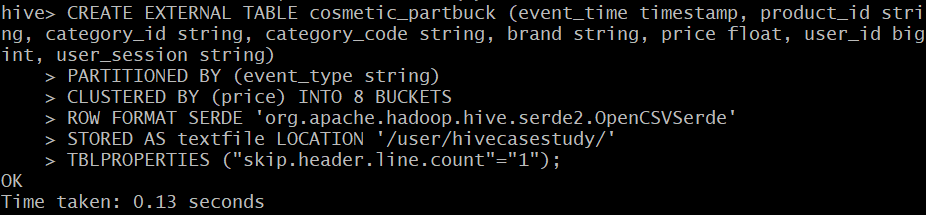
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'

STORED AS textfile LOCATION '/user/hivecasestudy/'

TBLPROPERTIES ("skip.header.line.count"="1");

Output:

|  |
| --- |
| hive> CREATE EXTERNAL TABLE cosmetic\_partbuck (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 8 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: 0.13 seconds |

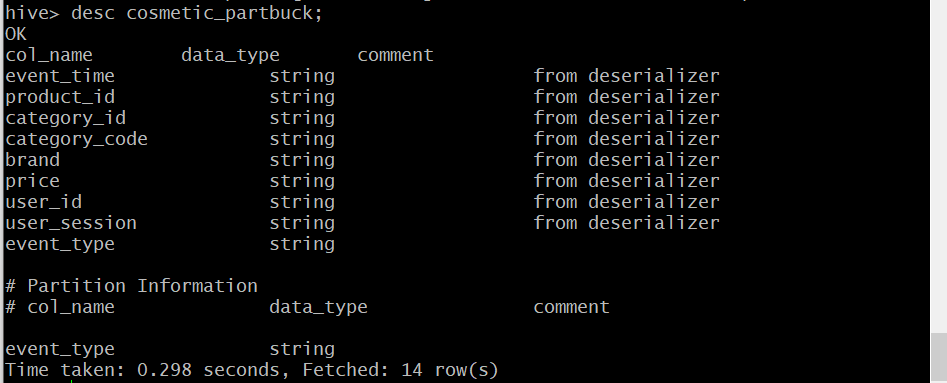


- Checking the new table ‘cosmetic\_partbuck;

Command: desc cosmetic\_partbuck;

Output:

|  |
| --- |
| hive> desc cosmetic\_partbuck;  OK  col\_name data\_type comment  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: 0.298 seconds, Fetched: 14 row(s) |



- Verifying partition creation in Hive

Command: show partitions cosmetic\_partbuck;

Output:

|  |
| --- |
| hive> show partitions cosmetic\_partbuck;  OK  partition  event\_type=cart  event\_type=purchase  event\_type=remove\_from\_cart  event\_type=view  Time taken: 0.082 seconds, Fetched: 4 row(s) |



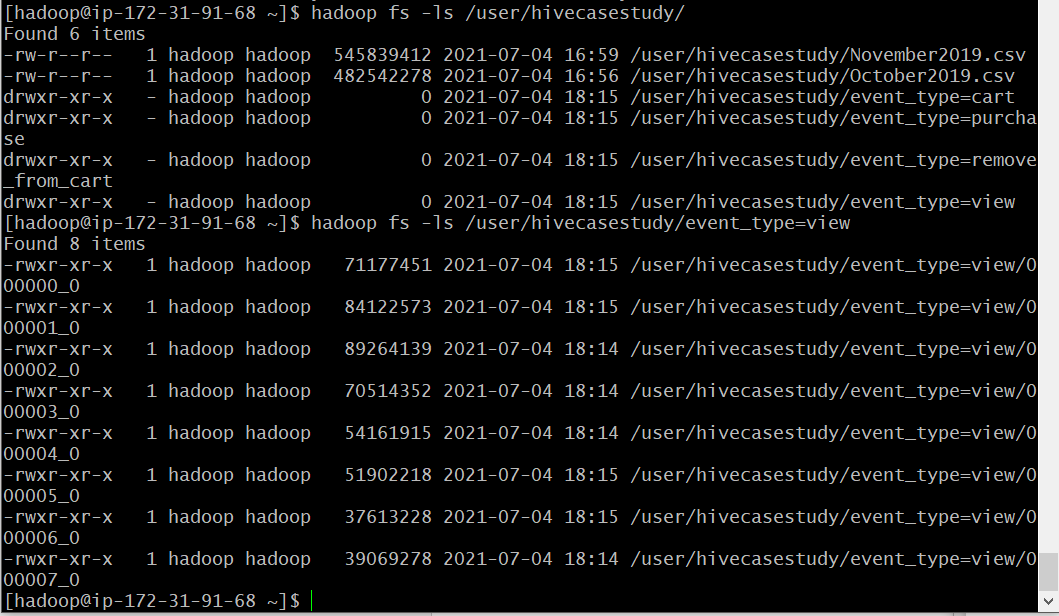
- Verifying partition creation in Hadoop

We exit from hive environment by executing 'exit;' command. In the hadoop environment we verify the partitioning.

Command: hadoop fs -ls /user/hivecasestudy/

Output:

|  |
| --- |
| [hadoop@ip-172-31-91-68 ~]$ hadoop fs -ls /user/hivecasestudy/  Found 6 items  -rw-r--r-- 1 hadoop hadoop 545839412 2021-07-04 16:59 /user/hivecasestudy/November2019.csv  -rw-r--r-- 1 hadoop hadoop 482542278 2021-07-04 16:56 /user/hivecasestudy/October2019.csv  drwxr-xr-x - hadoop hadoop 0 2021-07-04 18:15 /user/hivecasestudy/event\_type=cart  drwxr-xr-x - hadoop hadoop 0 2021-07-04 18:15 /user/hivecasestudy/event\_type=purchase  drwxr-xr-x - hadoop hadoop 0 2021-07-04 18:15 /user/hivecasestudy/event\_type=remove\_from\_cart  drwxr-xr-x - hadoop hadoop 0 2021-07-04 18:15 /user/hivecasestudy/event\_type=view |



Comment: We see that 4 partitions have been created.

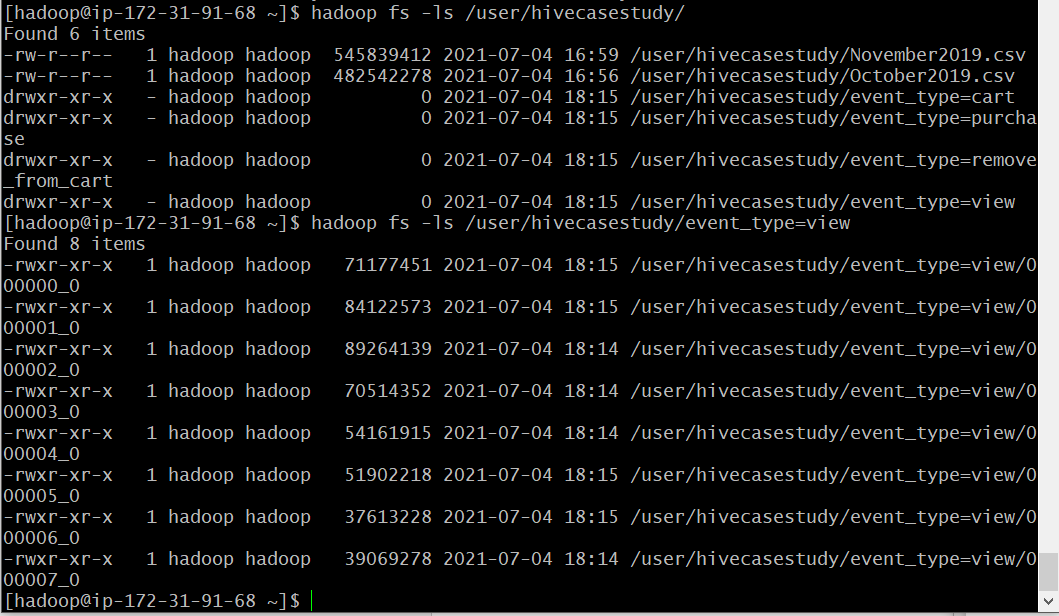
- Verifying Bucketing creation in Hadoop

We check if the bucketing has been executed, under the partition “event\_type=view”.

Command: hadoop fs -ls /user/hivecasestudy/event\_type=view

Output:

|  |
| --- |
| [hadoop@ip-172-31-91-68 ~]$ hadoop fs -ls /user/hivecasestudy/event\_type=view  Found 8 items  -rwxr-xr-x 1 hadoop hadoop 71177451 2021-07-04 18:15 /user/hivecasestudy/event\_type=view/000000\_0  -rwxr-xr-x 1 hadoop hadoop 84122573 2021-07-04 18:15 /user/hivecasestudy/event\_type=view/000001\_0  -rwxr-xr-x 1 hadoop hadoop 89264139 2021-07-04 18:14 /user/hivecasestudy/event\_type=view/000002\_0  -rwxr-xr-x 1 hadoop hadoop 70514352 2021-07-04 18:14 /user/hivecasestudy/event\_type=view/000003\_0  -rwxr-xr-x 1 hadoop hadoop 54161915 2021-07-04 18:14 /user/hivecasestudy/event\_type=view/000004\_0  -rwxr-xr-x 1 hadoop hadoop 51902218 2021-07-04 18:15 /user/hivecasestudy/event\_type=view/000005\_0  -rwxr-xr-x 1 hadoop hadoop 37613228 2021-07-04 18:15 /user/hivecasestudy/event\_type=view/000006\_0  -rwxr-xr-x 1 hadoop hadoop 39069278 2021-07-04 18:14 /user/hivecasestudy/event\_type=view/000007\_0  [hadoop@ip-172-31-91-68 ~]$ |



Comment: We see that 8 buckets have been created, with the indexing starting from 0-7. We can safely assume that, similar buckets have been created under all the partitions.

## **3.4 Inserting data into partitioned and bucketed table**

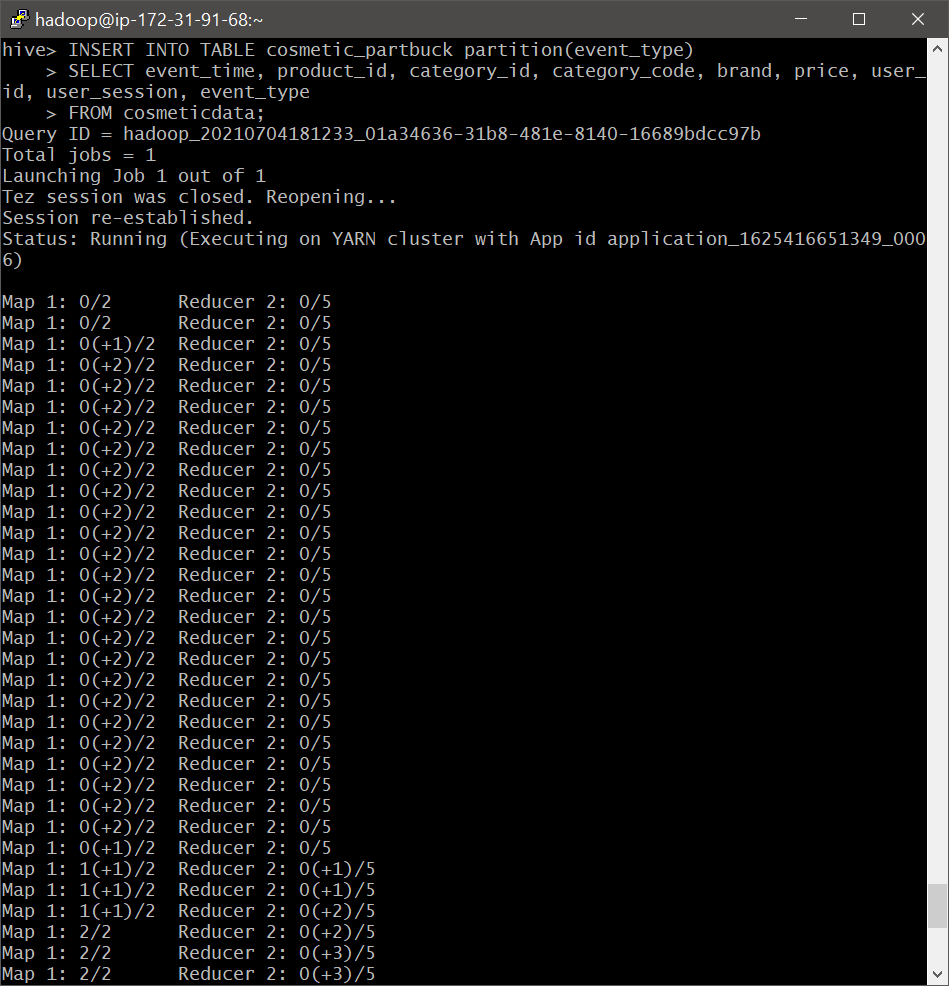
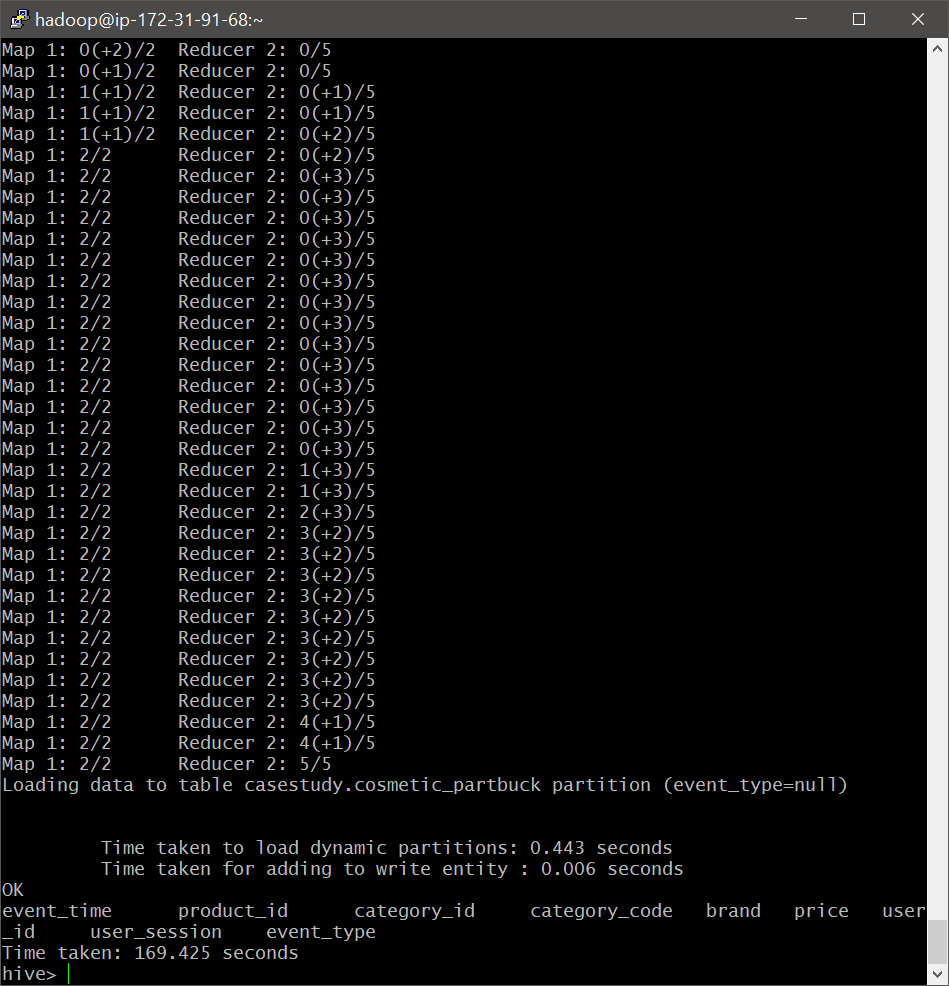
Command: INSERT INTO TABLE cosmetic\_partbuck partition(event\_type)

SELECT event\_time, product\_id, category\_id, category\_code, brand, price, user\_id, user\_session, event\_type

FROM cosmeticdata;

Output:

|  |
| --- |
| hive> INSERT INTO TABLE cosmetic\_partbuck partition(event\_type)  > SELECT event\_time, product\_id, category\_id, category\_code, brand, price, user\_id, user\_session, event\_type  > FROM cosmeticdata;  Query ID = hadoop\_20210704181233\_01a34636-31b8-481e-8140-16689bdcc97b  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\_1625416651349\_0006)  Map 1: 0/2 Reducer 2: 0/5  Map 1: 0/2 Reducer 2: 0/5  Map 1: 0(+1)/2 Reducer 2: 0/5  Map 1: 0(+2)/2 Reducer 2: 0/5  Map 1: 0(+2)/2 Reducer 2: 0/5  Map 1: 0(+2)/2 Reducer 2: 0/5  Map 1: 0(+2)/2 Reducer 2: 0/5  Map 1: 0(+2)/2 Reducer 2: 0/5  Map 1: 0(+2)/2 Reducer 2: 0/5  Map 1: 0(+2)/2 Reducer 2: 0/5  Map 1: 0(+2)/2 Reducer 2: 0/5  Map 1: 0(+2)/2 Reducer 2: 0/5  Map 1: 0(+2)/2 Reducer 2: 0/5  Map 1: 0(+2)/2 Reducer 2: 0/5  Map 1: 0(+2)/2 Reducer 2: 0/5  Map 1: 0(+2)/2 Reducer 2: 0/5  Map 1: 0(+2)/2 Reducer 2: 0/5  Map 1: 0(+2)/2 Reducer 2: 0/5  Map 1: 0(+2)/2 Reducer 2: 0/5  Map 1: 0(+2)/2 Reducer 2: 0/5  Map 1: 0(+2)/2 Reducer 2: 0/5  Map 1: 0(+2)/2 Reducer 2: 0/5  Map 1: 0(+2)/2 Reducer 2: 0/5  Map 1: 0(+2)/2 Reducer 2: 0/5  Map 1: 0(+2)/2 Reducer 2: 0/5  Map 1: 0(+2)/2 Reducer 2: 0/5  Map 1: 0(+1)/2 Reducer 2: 0/5  Map 1: 1(+1)/2 Reducer 2: 0(+1)/5  Map 1: 1(+1)/2 Reducer 2: 0(+1)/5  Map 1: 1(+1)/2 Reducer 2: 0(+2)/5  Map 1: 2/2 Reducer 2: 0(+2)/5  Map 1: 2/2 Reducer 2: 0(+3)/5  Map 1: 2/2 Reducer 2: 0(+3)/5  Map 1: 2/2 Reducer 2: 0(+3)/5  Map 1: 2/2 Reducer 2: 0(+3)/5  Map 1: 2/2 Reducer 2: 0(+3)/5  Map 1: 2/2 Reducer 2: 0(+3)/5  Map 1: 2/2 Reducer 2: 0(+3)/5  Map 1: 2/2 Reducer 2: 0(+3)/5  Map 1: 2/2 Reducer 2: 0(+3)/5  Map 1: 2/2 Reducer 2: 0(+3)/5  Map 1: 2/2 Reducer 2: 0(+3)/5  Map 1: 2/2 Reducer 2: 0(+3)/5  Map 1: 2/2 Reducer 2: 0(+3)/5  Map 1: 2/2 Reducer 2: 0(+3)/5  Map 1: 2/2 Reducer 2: 1(+3)/5  Map 1: 2/2 Reducer 2: 1(+3)/5  Map 1: 2/2 Reducer 2: 2(+3)/5  Map 1: 2/2 Reducer 2: 3(+2)/5  Map 1: 2/2 Reducer 2: 3(+2)/5  Map 1: 2/2 Reducer 2: 3(+2)/5  Map 1: 2/2 Reducer 2: 3(+2)/5  Map 1: 2/2 Reducer 2: 3(+2)/5  Map 1: 2/2 Reducer 2: 3(+2)/5  Map 1: 2/2 Reducer 2: 3(+2)/5  Map 1: 2/2 Reducer 2: 3(+2)/5  Map 1: 2/2 Reducer 2: 3(+2)/5  Map 1: 2/2 Reducer 2: 4(+1)/5  Map 1: 2/2 Reducer 2: 4(+1)/5  Map 1: 2/2 Reducer 2: 5/5  Loading data to table casestudy.cosmetic\_partbuck partition (event\_type=null)  Time taken to load dynamic partitions: 0.443 seconds  Time taken for adding to write entity : 0.006 seconds  OK  event\_time product\_id category\_id category\_code brand price user\_id user\_session event\_type  Time taken: 169.425 seconds |

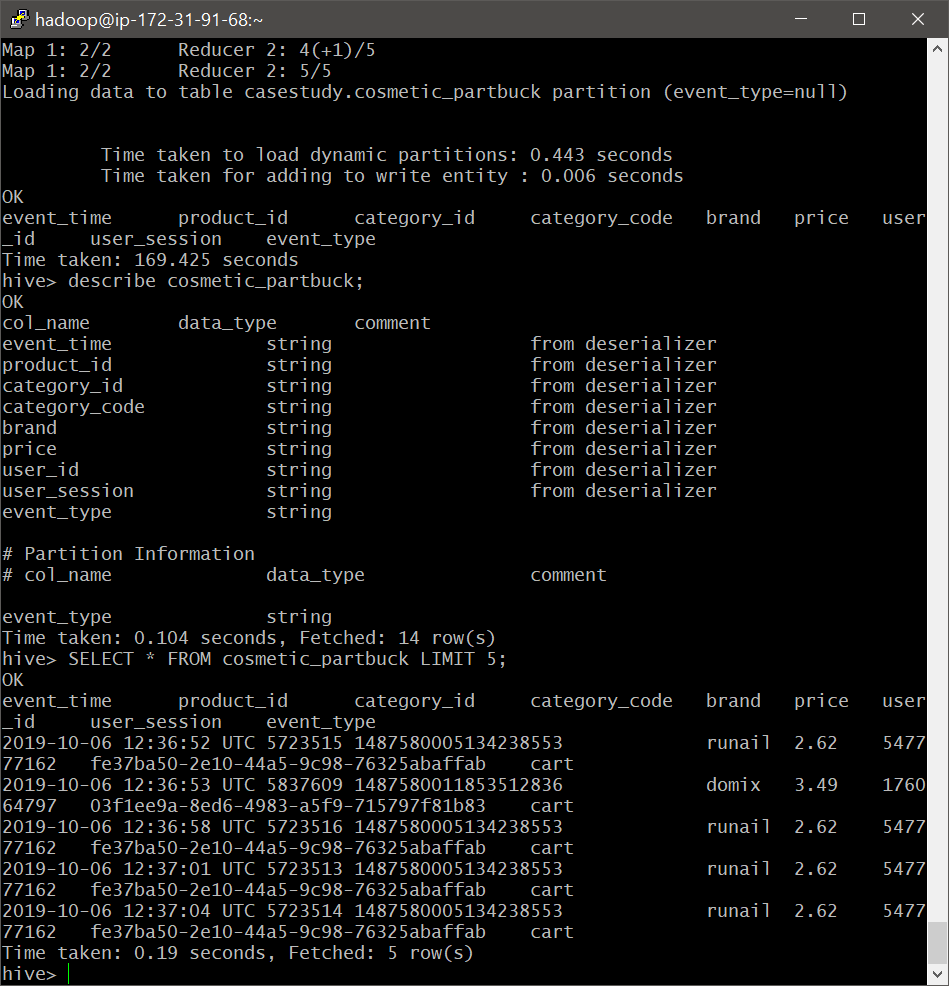
 

- Checking the properties of the new table

Command: describe cosmetic\_partbuck;

Output:

|  |
| --- |
| hive> describe cosmetic\_partbuck;  OK  col\_name data\_type comment  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: 0.104 seconds, Fetched: 14 row(s) |

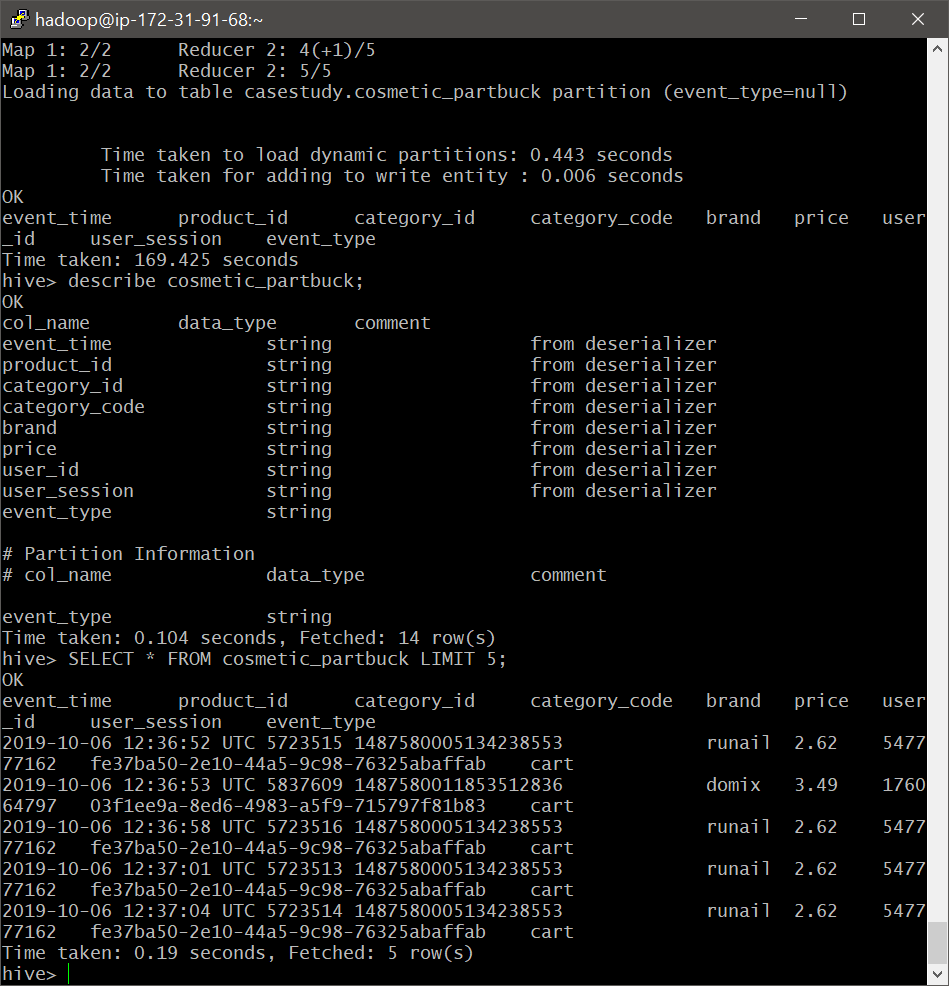


- Checking the correct loading of data into the new table

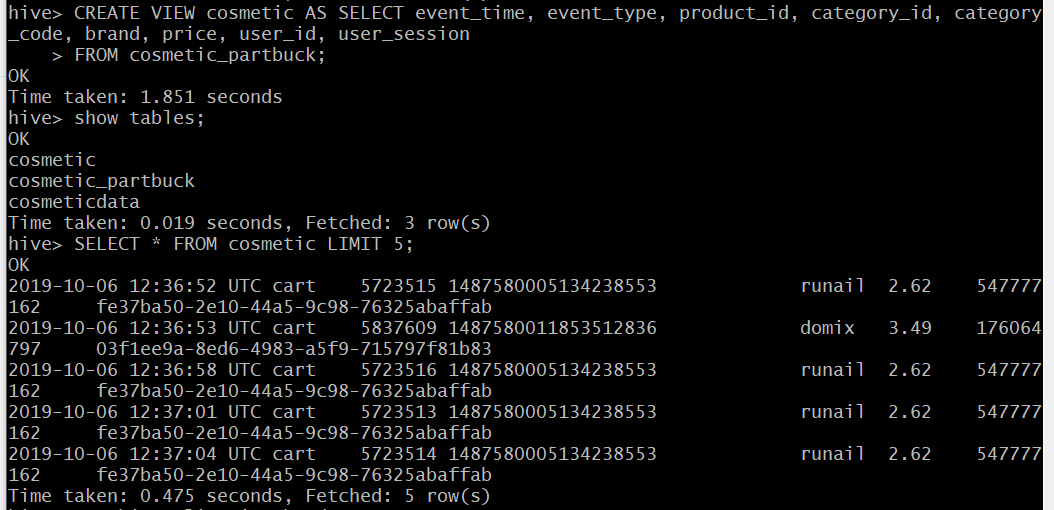
Command: SELECT \* FROM cosmetic\_partbuck LIMIT 5;

Output:

|  |
| --- |
| hive> SELECT \* FROM cosmetic\_partbuck LIMIT 5;  OK  event\_time product\_id category\_id category\_code brand price user\_id user\_session event\_type  2019-10-06 12:36:52 UTC 5723515 1487580005134238553 runail 2.62 547777162 fe37ba50-2e10-44a5-9c98-76325abaffab cart  2019-10-06 12:36:53 UTC 5837609 1487580011853512836 domix 3.49 176064797 03f1ee9a-8ed6-4983-a5f9-715797f81b83 cart  2019-10-06 12:36:58 UTC 5723516 1487580005134238553 runail 2.62 547777162 fe37ba50-2e10-44a5-9c98-76325abaffab cart  2019-10-06 12:37:01 UTC 5723513 1487580005134238553 runail 2.62 547777162 fe37ba50-2e10-44a5-9c98-76325abaffab cart  2019-10-06 12:37:04 UTC 5723514 1487580005134238553 runail 2.62 547777162 fe37ba50-2e10-44a5-9c98-76325abaffab cart  Time taken: 0.19 seconds, Fetched: 5 row(s) |



- Use of ‘view’ command (Unsucessful)



We created a new table “cosmetic” using the ‘view’ command. This table is exactly the same as the underlying table (cosmetic\_partbuck).

Please note: We will be henceforth using “cosmetic” table, which is a partitioned and bucketed table.

- Dropping unnecessary table ‘cosmetic\_partbuck’

We will be dropping the ‘cosmetic\_partbuck’ table as it is essentially a duplicate of ‘cosmetic’ table.

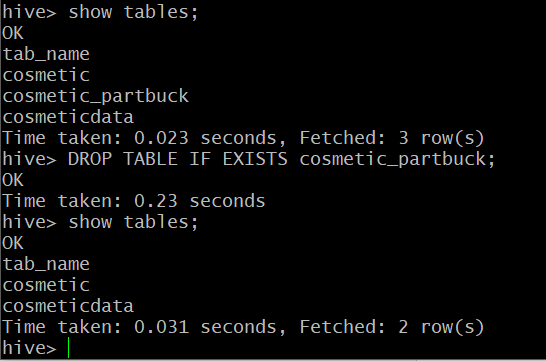
Command: show tables;

DROP TABLE IF EXISTS cosmetic\_partbuck;

show tables;

Output:

|  |
| --- |
| hive> show tables;  OK  tab\_name  cosmetic  cosmetic\_partbuck  cosmeticdata  Time taken: 0.023 seconds, Fetched: 3 row(s)  hive> DROP TABLE IF EXISTS cosmetic\_partbuck;  OK  Time taken: 0.23 seconds  hive> show tables;  OK  tab\_name  cosmetic  cosmeticdata  Time taken: 0.031 seconds, Fetched: 2 row(s) |



# Querying – Assignment Questions

|  |
| --- |
| [Question1](#_Find_the_total) [Question2](#_Write_a_query) [Question3](#_Write_a_query_1) [Question4](#_Find_distinct_categories) [Question5](#_Find_the_total_1) [Question6](#_Which_brand_had) [Question7](#_Which_brands_increased) [Question8](#_Your_company_wants) |

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

Command: SELECT ROUND(SUM(price),2) as TotalRev\_October

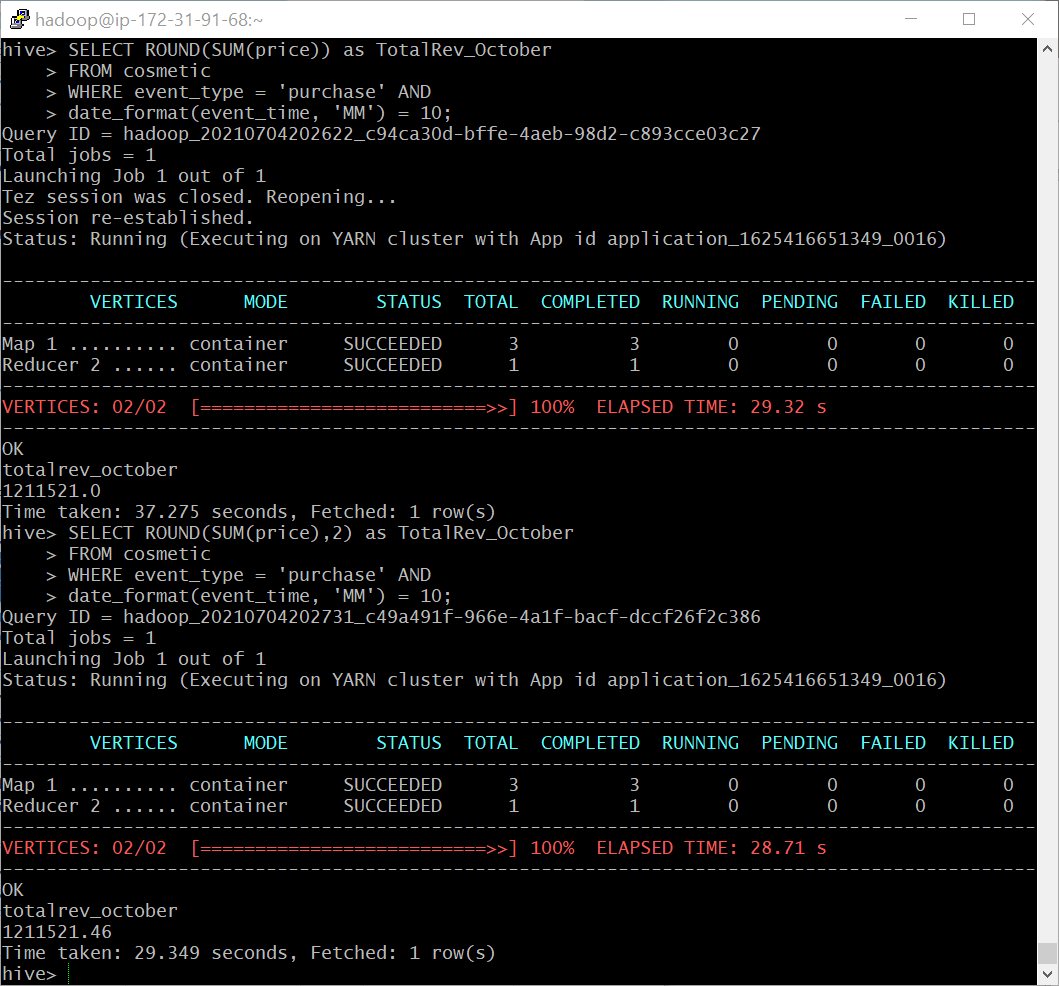
FROM cosmetic

WHERE event\_type = 'purchase' AND

date\_format(event\_time, 'MM') = 10;

Output:

|  |
| --- |
| hive> SELECT ROUND(SUM(price),2) as TotalRev\_October  > FROM cosmetic  > WHERE event\_type = 'purchase' AND  > date\_format(event\_time, 'MM') = 10;  Query ID = hadoop\_20210704202731\_c49a491f-966e-4a1f-bacf-dccf26f2c386  Total jobs = 1  Launching Job 1 out of 1  Status: Running (Executing on YARN cluster with App id application\_1625416651349\_0016)  ----------------------------------------------------------------------------------------------  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: 28.71 s  ----------------------------------------------------------------------------------------------  OK  totalrev\_october  1211521.46  Time taken: 29.349 seconds, Fetched: 1 row(s) |



Comment: We see that the Total Revenue in the month of ‘October 2019’ is Rs. 1211521.46 /-.

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

Command: SELECT date\_format(event\_time, 'MM') as Month, ROUND(SUM(price),2) as Sum\_Purchases

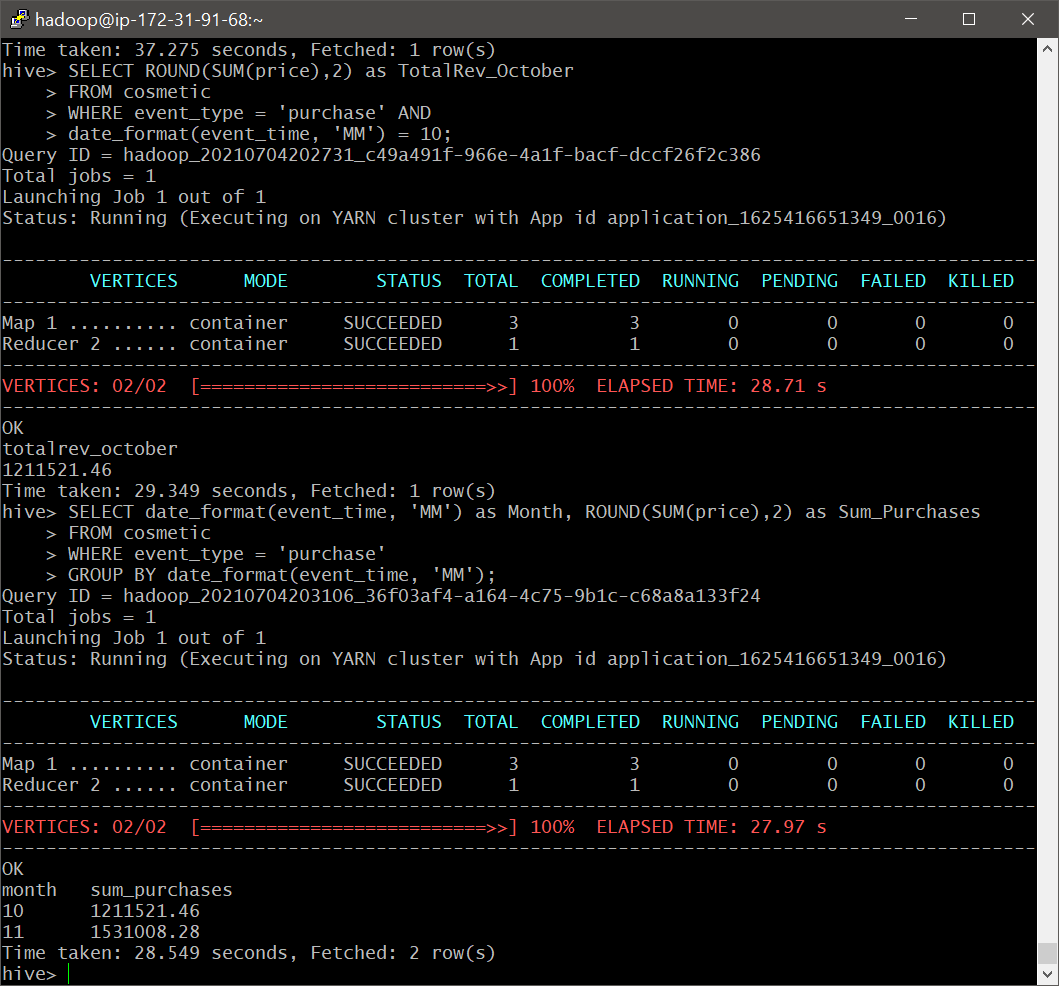
FROM cosmetic

WHERE event\_type = 'purchase'

GROUP BY date\_format(event\_time, 'MM');

Output:

|  |
| --- |
| hive> SELECT date\_format(event\_time, 'MM') as Month, ROUND(SUM(price),2) as Sum\_Purchases  > FROM cosmetic  > WHERE event\_type = 'purchase'  > GROUP BY date\_format(event\_time, 'MM');  Query ID = hadoop\_20210704203106\_36f03af4-a164-4c75-9b1c-c68a8a133f24  Total jobs = 1  Launching Job 1 out of 1  Status: Running (Executing on YARN cluster with App id application\_1625416651349\_0016)  ----------------------------------------------------------------------------------------------  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: 27.97 s  ----------------------------------------------------------------------------------------------  OK  month sum\_purchases  10 1211521.46  11 1531008.28  Time taken: 28.549 seconds, Fetched: 2 row(s) |



Comment:

* We observe that the sum of purchases (Revenue) in the month of ‘November 2019’ is higher than that of ‘October 2019’.
* We can infer that the month of November has performed better than October.

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

Command: SELECT ROUND((Nov\_Rev - Oct\_Rev), 2) AS Change\_in\_Revenue

FROM

(

SELECT SUM(case when date\_format(event\_time,'MM')=10 then price else 0 end) AS Oct\_Rev,

SUM(case when date\_format(event\_time,'MM')=11 then price else 0 end) AS Nov\_Rev

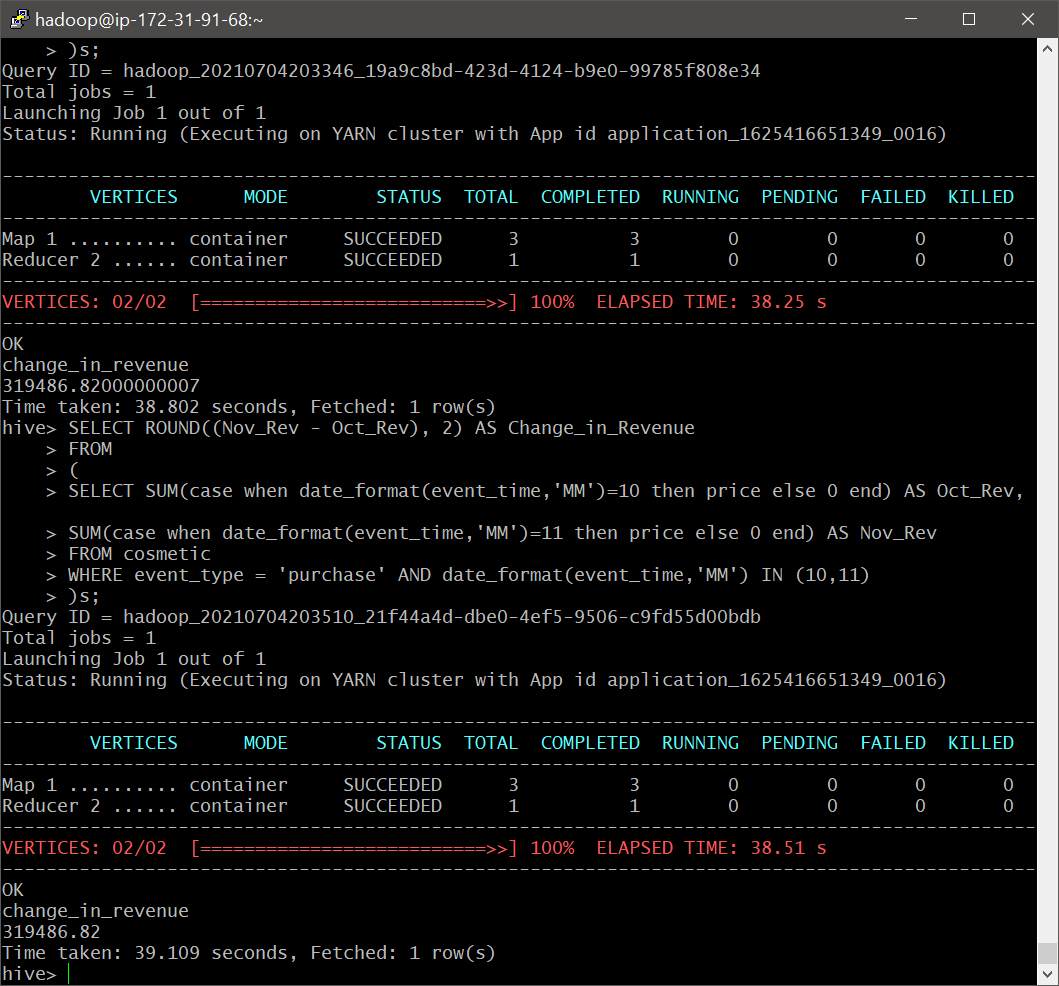
FROM cosmetic

WHERE event\_type = 'purchase' AND date\_format(event\_time,'MM') IN (10,11)

)s;

Output:

|  |
| --- |
| hive> SELECT ROUND((Nov\_Rev - Oct\_Rev), 2) AS Change\_in\_Revenue  > FROM  > (  > SELECT SUM(case when date\_format(event\_time,'MM')=10 then price else 0 end) AS Oct\_Rev,  > SUM(case when date\_format(event\_time,'MM')=11 then price else 0 end) AS Nov\_Rev  > FROM cosmetic  > WHERE event\_type = 'purchase' AND date\_format(event\_time,'MM') IN (10,11)  > )s;  Query ID = hadoop\_20210704203510\_21f44a4d-dbe0-4ef5-9506-c9fd55d00bdb  Total jobs = 1  Launching Job 1 out of 1  Status: Running (Executing on YARN cluster with App id application\_1625416651349\_0016)  ----------------------------------------------------------------------------------------------  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: 38.51 s  ----------------------------------------------------------------------------------------------  OK  change\_in\_revenue  319486.82  Time taken: 39.109 seconds, Fetched: 1 row(s) |
|  |



Comment: In continuation with our previous inference, we can see that Revenue generated in November is higher than October by Rs. 3,19,486.82 /-.

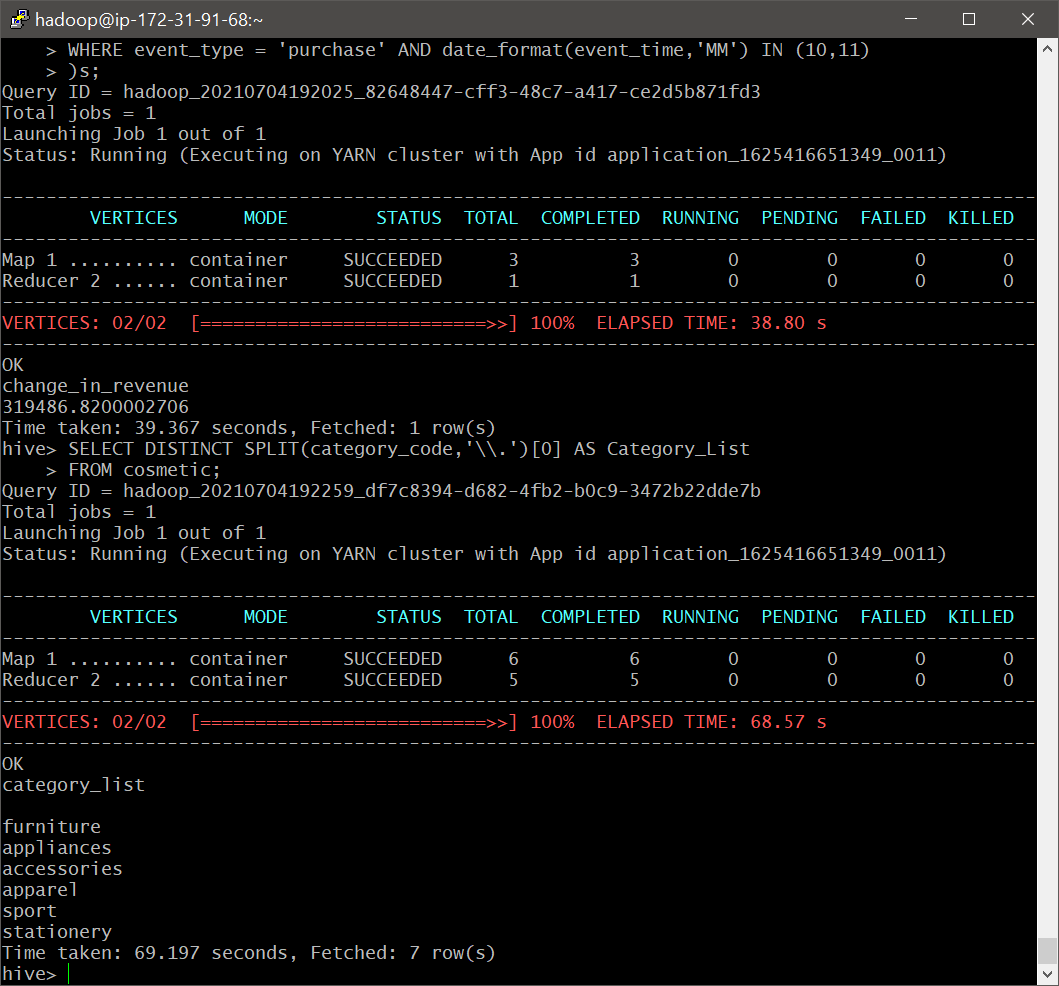
## **Find distinct categories of products. Categories with null category code can be ignored.**

Command: SELECT DISTINCT SPLIT(category\_code,'\\.')[0] AS Category\_List

FROM cosmetic;

Output:

|  |
| --- |
| hive> SELECT DISTINCT SPLIT(category\_code,'\\.')[0] AS Category\_List  > FROM cosmetic;  Query ID = hadoop\_20210704192259\_df7c8394-d682-4fb2-b0c9-3472b22dde7b  Total jobs = 1  Launching Job 1 out of 1  Status: Running (Executing on YARN cluster with App id application\_1625416651349\_0011)  ----------------------------------------------------------------------------------------------  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: 68.57 s  ----------------------------------------------------------------------------------------------  OK  category\_list  furniture  appliances  accessories  apparel  sport  stationery  Time taken: 69.197 seconds, Fetched: 7 row(s) |



Comment:

* We observe 6 different categories present namely; furniture, appliances, accessories, apparel, sport, stationery.
* The category\_code column contained values, which were delimited by ‘.’. We use the SPLIT command to split and located the first index alone, which contained the main Category.

## **Find the total number of products available under each category.**

Command: SELECT SPLIT(category\_code,'\\.')[0] AS Category, COUNT(product\_id) AS Total\_Product\_count

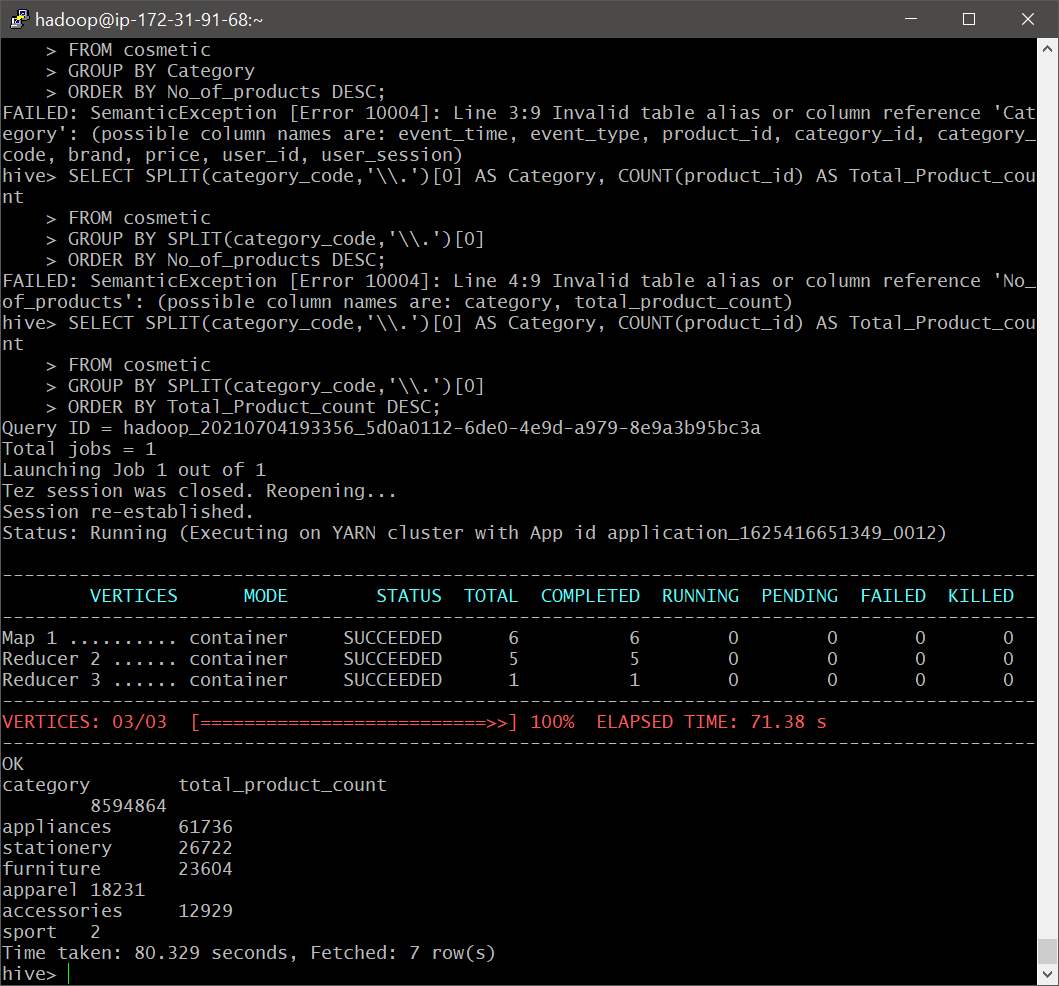
FROM cosmetic

GROUP BY SPLIT(category\_code,'\\.')[0]

ORDER BY Total\_Product\_count DESC;

Output:

|  |
| --- |
| hive> SELECT SPLIT(category\_code,'\\.')[0] AS Category, COUNT(product\_id) AS Total\_Product\_count  > FROM cosmetic  > GROUP BY SPLIT(category\_code,'\\.')[0]  > ORDER BY Total\_Product\_count DESC;  Query ID = hadoop\_20210704193356\_5d0a0112-6de0-4e9d-a979-8e9a3b95bc3a  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\_1625416651349\_0012)  ----------------------------------------------------------------------------------------------  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: 71.38 s  ----------------------------------------------------------------------------------------------  OK  category total\_product\_count  8594864  appliances 61736  stationery 26722  furniture 23604  apparel 18231  accessories 12929  sport 2  Time taken: 80.329 seconds, Fetched: 7 row(s) |



Comment:

* We observe that, ‘appliances’ category has the highest number of cosmetic products available under it.
* We can see ‘sports’ category has the least cosmetic products under it. This make sense as sports category would not contain many cosmetic products.

## **Which brand had the maximum sales in October and November combined?**

Command: SELECT brand, ROUND(SUM(price),2) as Total\_sales

FROM cosmetic

WHERE event\_type = 'purchase'

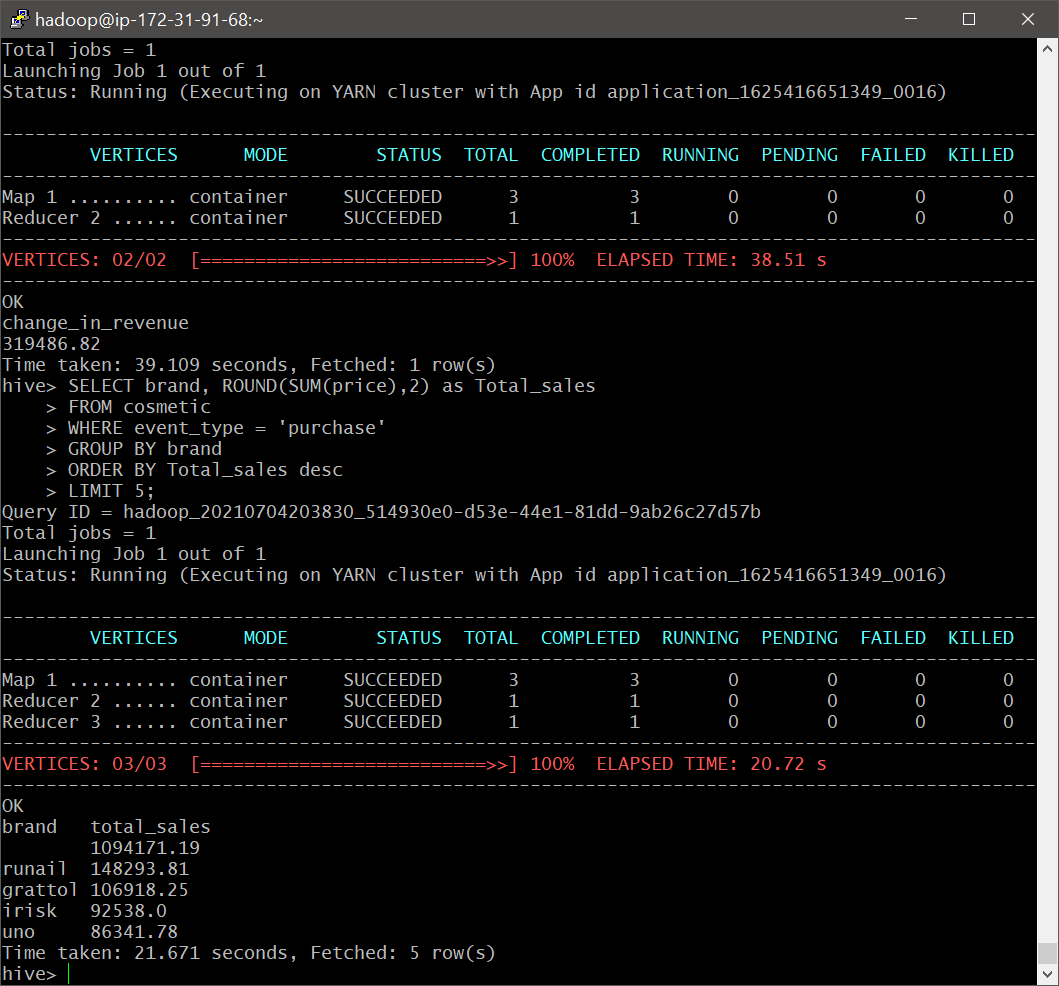
GROUP BY brand

ORDER BY Total\_sales desc

LIMIT 5;

Output:

|  |
| --- |
| hive> SELECT brand, ROUND(SUM(price),2) as Total\_sales  > FROM cosmetic  > WHERE event\_type = 'purchase'  > GROUP BY brand  > ORDER BY Total\_sales desc  > LIMIT 5;  Query ID = hadoop\_20210704203830\_514930e0-d53e-44e1-81dd-9ab26c27d57b  Total jobs = 1  Launching Job 1 out of 1  Status: Running (Executing on YARN cluster with App id application\_1625416651349\_0016)  ----------------------------------------------------------------------------------------------  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: 20.72 s  ----------------------------------------------------------------------------------------------  OK  brand total\_sales  1094171.19  runail 148293.81  grattol 106918.25  irisk 92538.0  uno 86341.78  Time taken: 21.671 seconds, Fetched: 5 row(s) |



Comment:

* We observe that, ‘runail’ brand is a popular brand with high sales in both months.
* We see that the brand ‘gruttol’ is not far behind with a difference of hardly Rs 40,000/-

## **Which brands increased their sales from October to November?**

Command: WITH OctNov\_Sales AS (

SELECT brand,

SUM(CASE WHEN date\_format(event\_time, 'MM')=10 THEN price ELSE 0 END) AS Oct\_Sales,

SUM(CASE WHEN date\_format(event\_time, 'MM')=11 THEN price ELSE 0 END) AS Nov\_Sales

FROM cosmetic

WHERE event\_type = 'purchase' AND

date\_format(event\_time, 'MM') 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

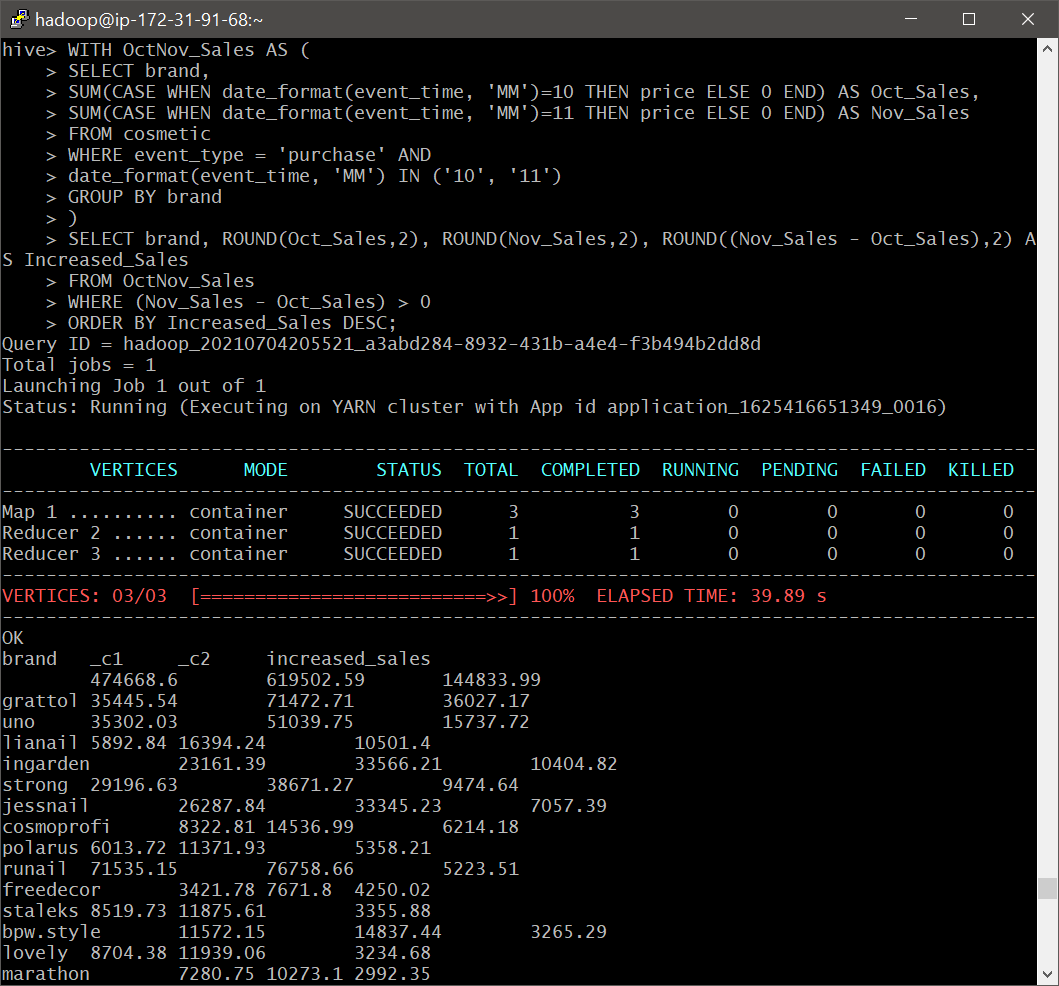
FROM OctNov\_Sales

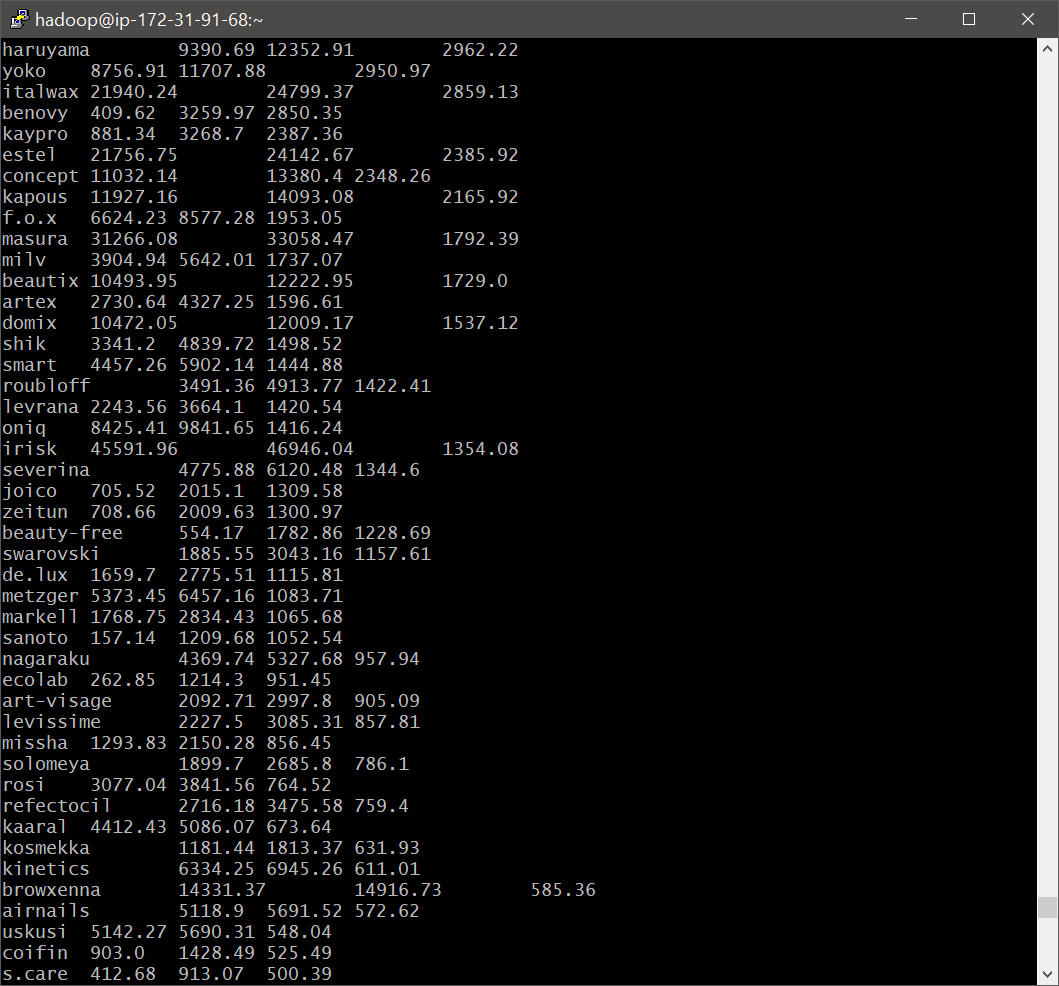
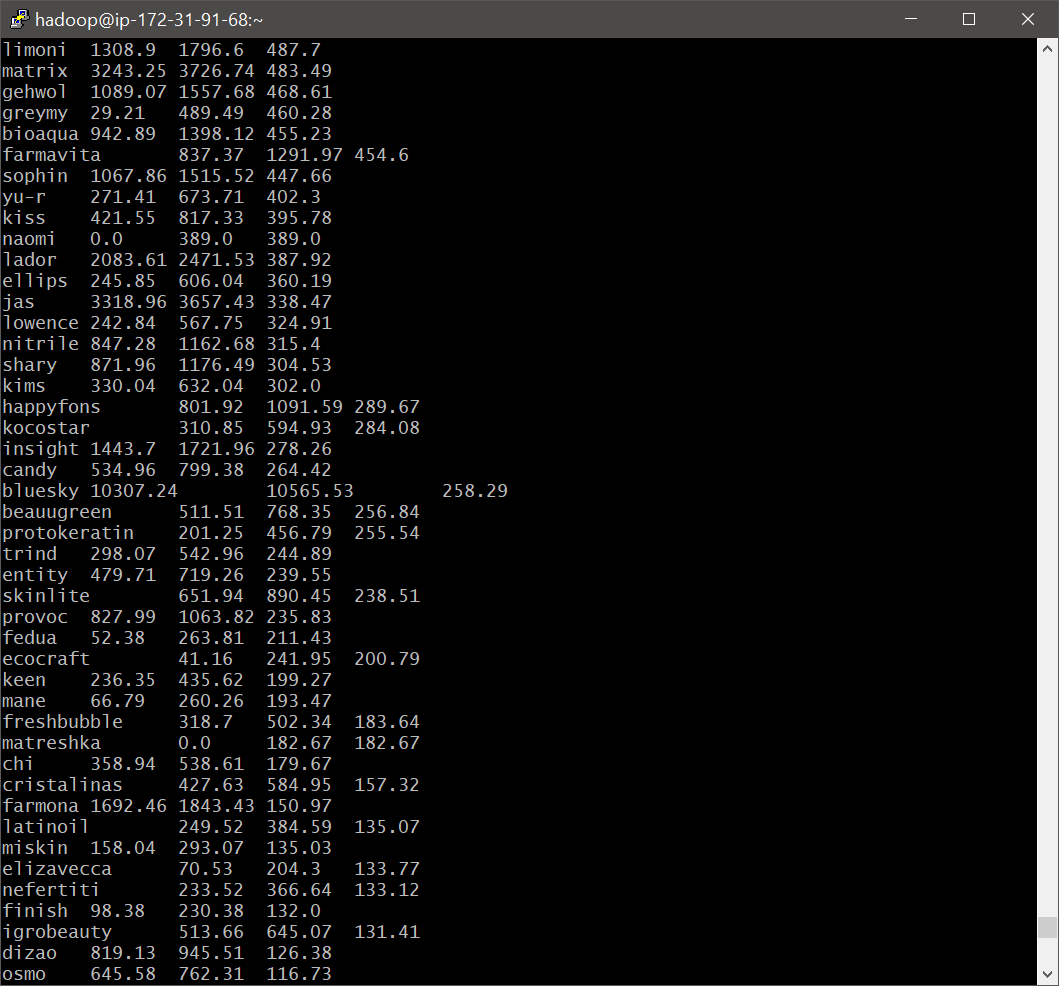
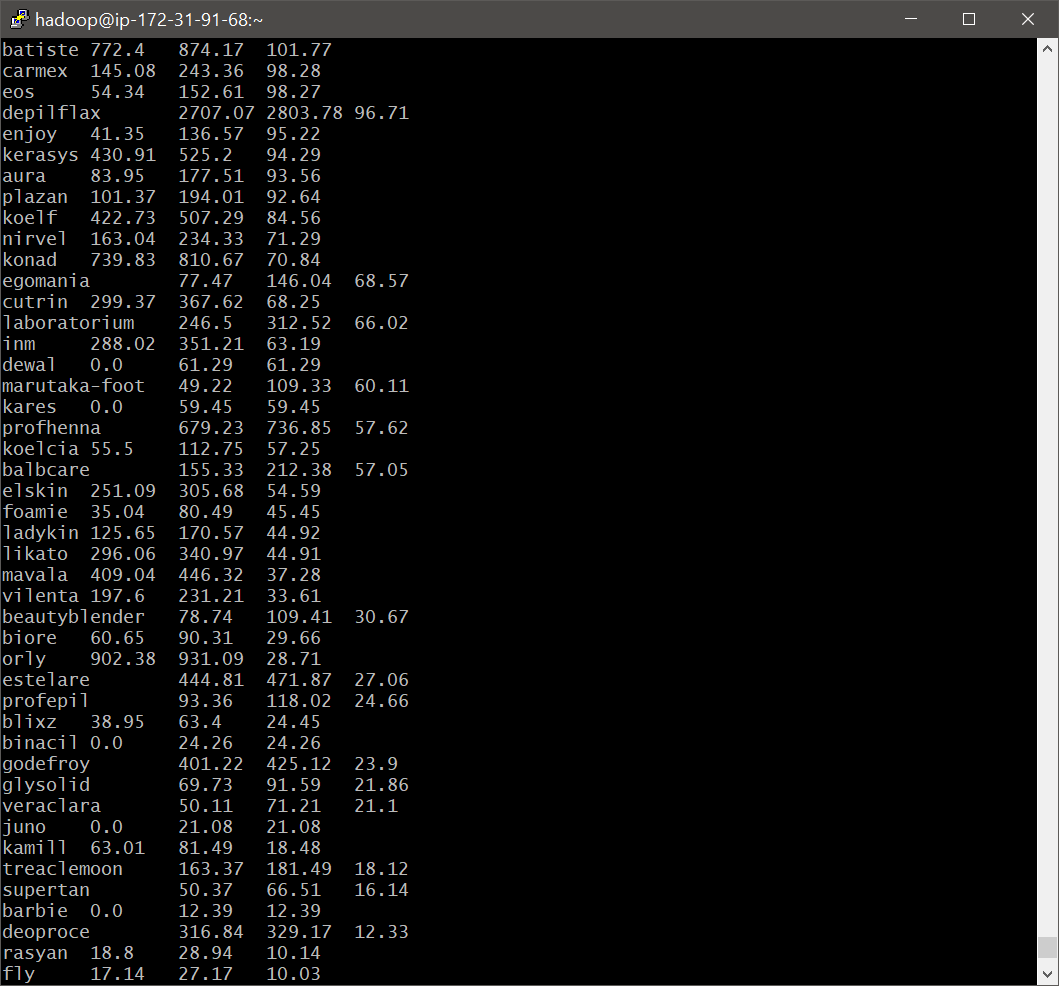
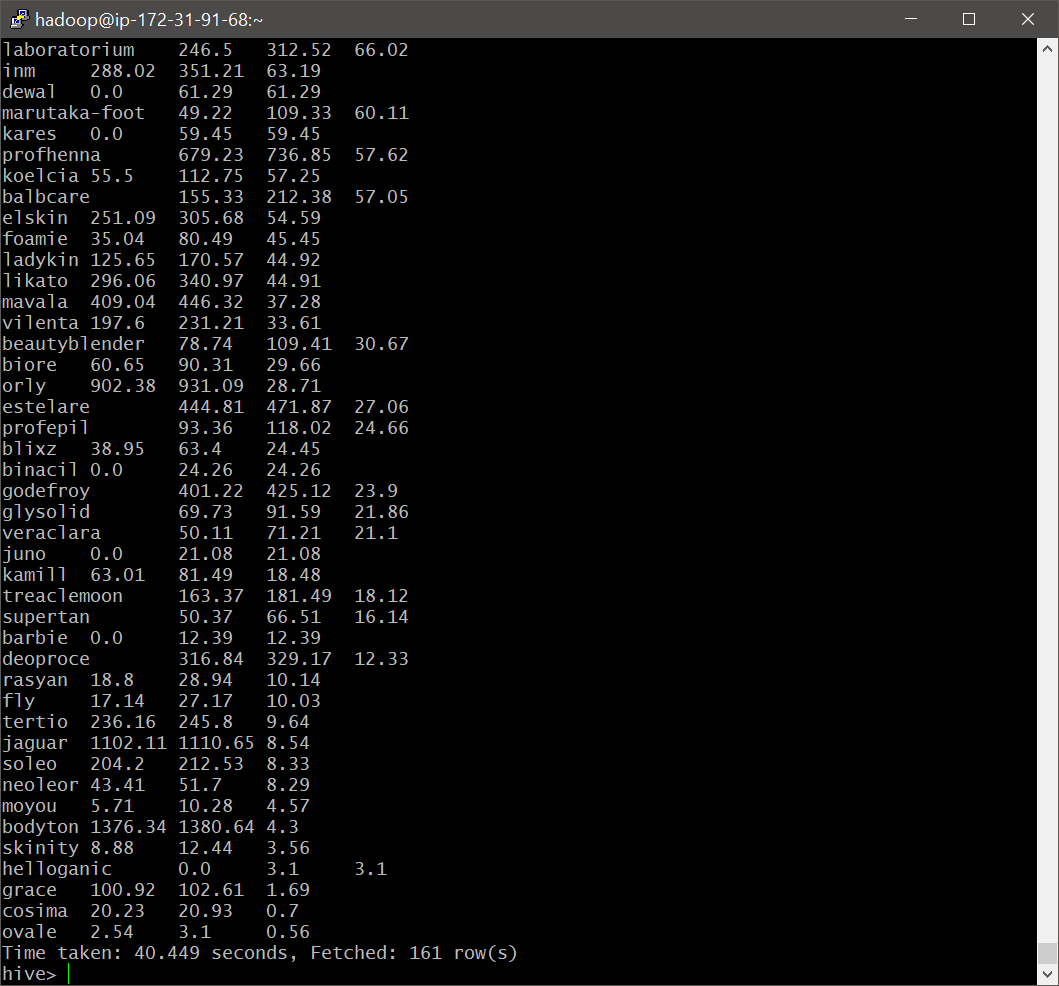
WHERE (Nov\_Sales - Oct\_Sales) > 0

ORDER BY Increased\_Sales DESC;

Output:

|  |
| --- |
| hive> WITH OctNov\_Sales AS (  > SELECT brand,  > SUM(CASE WHEN date\_format(event\_time, 'MM')=10 THEN price ELSE 0 END) AS Oct\_Sales,  > SUM(CASE WHEN date\_format(event\_time, 'MM')=11 THEN price ELSE 0 END) AS Nov\_Sales  > FROM cosmetic  > WHERE event\_type = 'purchase' AND  > date\_format(event\_time, 'MM') 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 OctNov\_Sales  > WHERE (Nov\_Sales - Oct\_Sales) > 0  > ORDER BY Increased\_Sales DESC;  Query ID = hadoop\_20210704205521\_a3abd284-8932-431b-a4e4-f3b494b2dd8d  Total jobs = 1  Launching Job 1 out of 1  Status: Running (Executing on YARN cluster with App id application\_1625416651349\_0016)  ----------------------------------------------------------------------------------------------  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: 39.89 s  ----------------------------------------------------------------------------------------------  OK  brand \_c1 \_c2 increased\_sales  474668.6 619502.59 144833.99  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 71535.15 76758.66 5223.51  freedecor 3421.78 7671.8 4250.02  staleks 8519.73 11875.61 3355.88  bpw.style 11572.15 14837.44 3265.29  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 31266.08 33058.47 1792.39  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 1885.55 3043.16 1157.61  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  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  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 251.09 305.68 54.59  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  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: 40.449 seconds, Fetched: 161 row(s) |



Comment:

* We observe that, ‘gruttol’ brand has seen the highest per month increase of Rs 36027.17/- . In the previous query we observed that it was a brand with second highest total sales.
* We see ‘runail’ brand is in the 9th position.
* The brand with lowest per month increase of ‘0.56’ is ‘ovale’; followed by ‘cosima’ with a ‘0.7’ difference in monthly sales.

## **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 Total\_money\_spent

FROM cosmetic

WHERE event\_type = 'purchase'

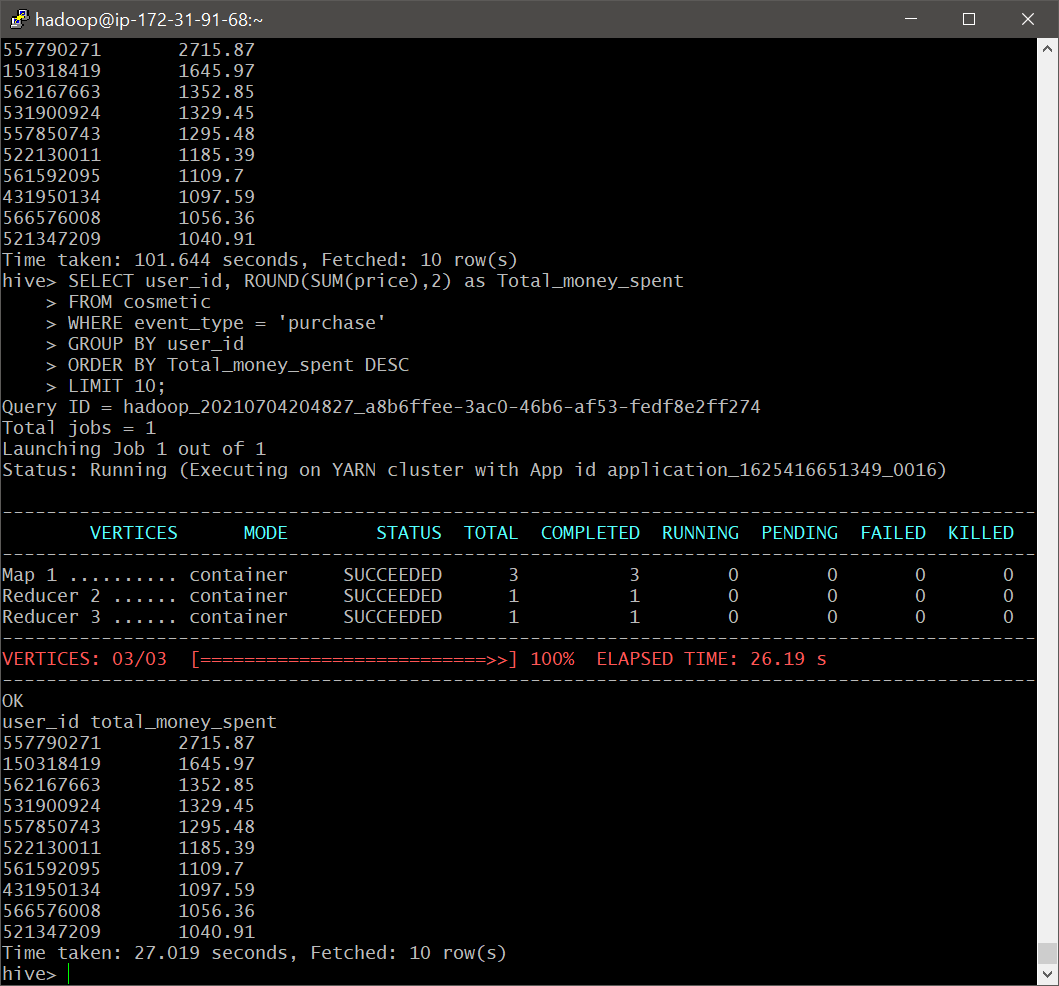
GROUP BY user\_id

ORDER BY Total\_money\_spent DESC

LIMIT 10;

Output:

|  |
| --- |
| hive> SELECT user\_id, ROUND(SUM(price),2) as Total\_money\_spent  > FROM cosmetic  > WHERE event\_type = 'purchase'  > GROUP BY user\_id  > ORDER BY Total\_money\_spent DESC  > LIMIT 10;  Query ID = hadoop\_20210704204827\_a8b6ffee-3ac0-46b6-af53-fedf8e2ff274  Total jobs = 1  Launching Job 1 out of 1  Status: Running (Executing on YARN cluster with App id application\_1625416651349\_0016)  ----------------------------------------------------------------------------------------------  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: 26.19 s  ----------------------------------------------------------------------------------------------  OK  user\_id total\_money\_spent  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: 27.019 seconds, Fetched: 10 row(s) |



# Demonstration - Improvement of performance after using optimization on tables

Hive optimization techniques are used to reduce the query execution time. We had created an optimized table “cosmetic” 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 “cosmeticdata”. We have selected the Question 6 & Question 8 for the demonstration.

## **Which brand had the maximum sales in October and November combined? *(Not optimized Querying)***

Command: SELECT brand, ROUND(SUM(price),2) as Total\_sales

FROM cosmeticdata

WHERE event\_type = 'purchase'

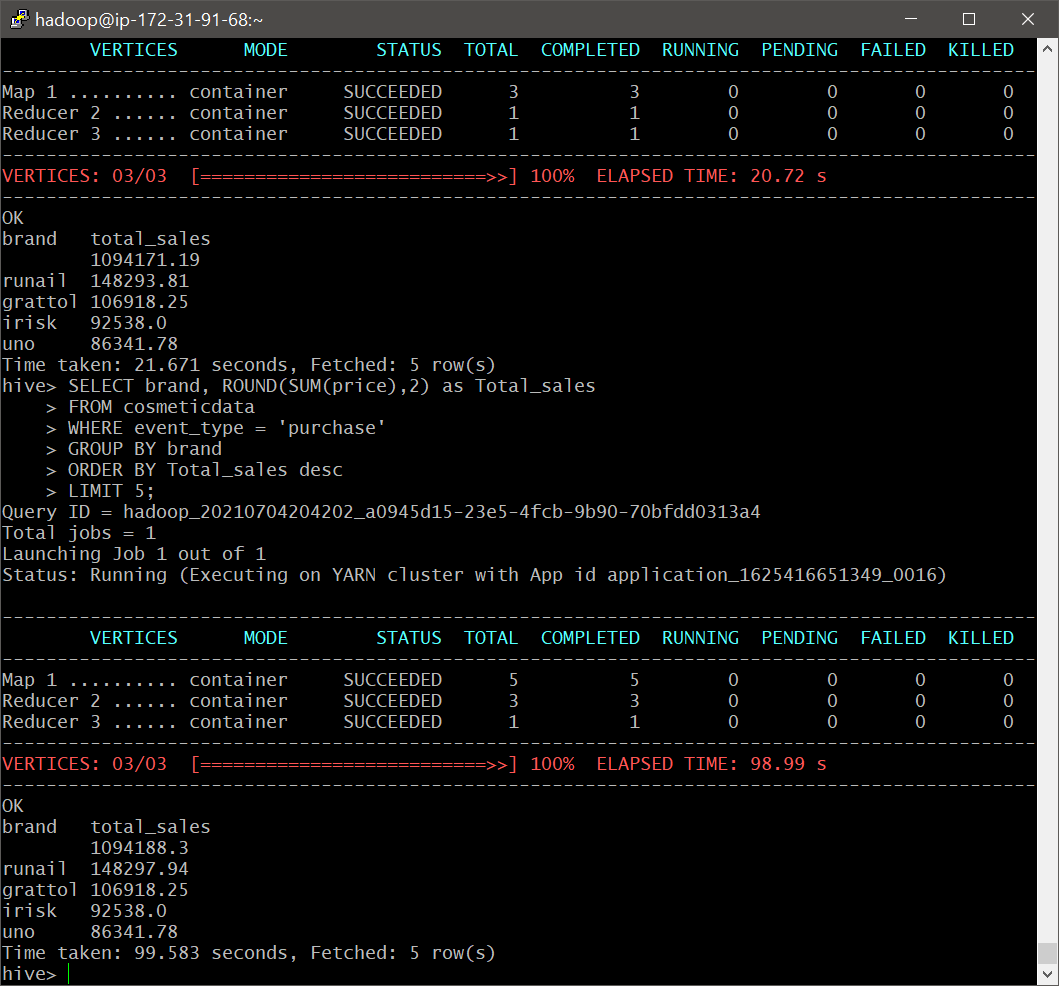
GROUP BY brand

ORDER BY Total\_sales desc

LIMIT 5;

Output:

|  |
| --- |
| hive> SELECT brand, ROUND(SUM(price),2) as Total\_sales  > FROM cosmeticdata  > WHERE event\_type = 'purchase'  > GROUP BY brand  > ORDER BY Total\_sales desc  > LIMIT 5;  Query ID = hadoop\_20210704204202\_a0945d15-23e5-4fcb-9b90-70bfdd0313a4  Total jobs = 1  Launching Job 1 out of 1  Status: Running (Executing on YARN cluster with App id application\_1625416651349\_0016)  ----------------------------------------------------------------------------------------------  VERTICES MODE STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED  ----------------------------------------------------------------------------------------------  Map 1 .......... container SUCCEEDED 5 5 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: 98.99 s  ----------------------------------------------------------------------------------------------  OK  brand total\_sales  1094188.3  runail 148297.94  grattol 106918.25  irisk 92538.0  uno 86341.78  Time taken: 99.583 seconds, Fetched: 5 row(s) |



Comment:

* We see no change in results obtained through both the ways as the data used is the same.
* We observed that, the query execution time previously was 21.671 seconds. We see the query execution time for non optimized query is 99.583 seconds, with a difference of 77.912 seconds. This is a huge difference, as the data size increase the increase in time will be substantially higher.

## **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. *(Not optimized Querying)***

Command: SELECT user\_id, ROUND(SUM(price),2) as Total\_money\_spent

FROM cosmeticdata

WHERE event\_type = 'purchase'

GROUP BY user\_id

ORDER BY Total\_money\_spent DESC

LIMIT 10;

Output:

|  |
| --- |
| hive> SELECT user\_id, ROUND(SUM(price),2) as Total\_money\_spent  > FROM cosmeticdata  > WHERE event\_type = 'purchase'  > GROUP BY user\_id  > ORDER BY Total\_money\_spent DESC  > LIMIT 10;  Query ID = hadoop\_20210704204534\_5ffd3077-8a33-4396-870b-06559f044c0f  Total jobs = 1  Launching Job 1 out of 1  Status: Running (Executing on YARN cluster with App id application\_1625416651349\_0016)  ----------------------------------------------------------------------------------------------  VERTICES MODE STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED  ----------------------------------------------------------------------------------------------  Map 1 .......... container SUCCEEDED 5 5 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: 101.13 s  ----------------------------------------------------------------------------------------------  OK  user\_id total\_money\_spent  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: 101.644 seconds, Fetched: 10 row(s) |



Comment:

* We see no change in results obtained through both the ways as the data used is the same.
* Similar to previous query we observe that, the query execution time has increased significantly. The previous query execution time was 27.019 seconds and the query execution time for non optimized query is 101.644 seconds, with a difference of 74.625 seconds.
* We can conclude that, partitioning and bucketing are essential to reduce query execution time.

# Cleaning-up

Since we have loaded the database directly from S3 to HDFS, terminating the EMR cluster will automatically drop the database.

We remove the termination protection enabled & and terminate our cluster.

