**E-Commerce Sales Analysis (Amazon)**

**Group Members**

Khusroo Kalim Ansari: 300347313

Mohammed Ikram: 300343622

Newlin Tran: 300323900

**Introduction**

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. So, as a big data analyst, our job is to extract data and gather insights from a real-life data set of an e-commerce company.

Our Analysis will answer the following questions-

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

**Dataset Links**

* <https://registry.opendata.aws/>
* <https://e-commerce-events-ml.s3.amazonaws.com/2019-Oct.csv>
* <https://e-commerce-events-ml.s3.amazonaws.com/2019-Nov.csv>

**Knowledge & Software Used**

* Big Data & Cloud
* Hadoop
* Hive CLI (Used HQL to query)
* AWS EMR Cluster
* AWS S3 Bucket

**EMR CLUSTER CREATION**

**STEP 1:** Login to your AWS account & search EMR services. After the EMR home page appears click on **Create cluster** & follow the steps as mentioned. We have chosen cluster release version **emr-5.35.0** in our case study.

**Graphical user interface, text, application

Description automatically generated**



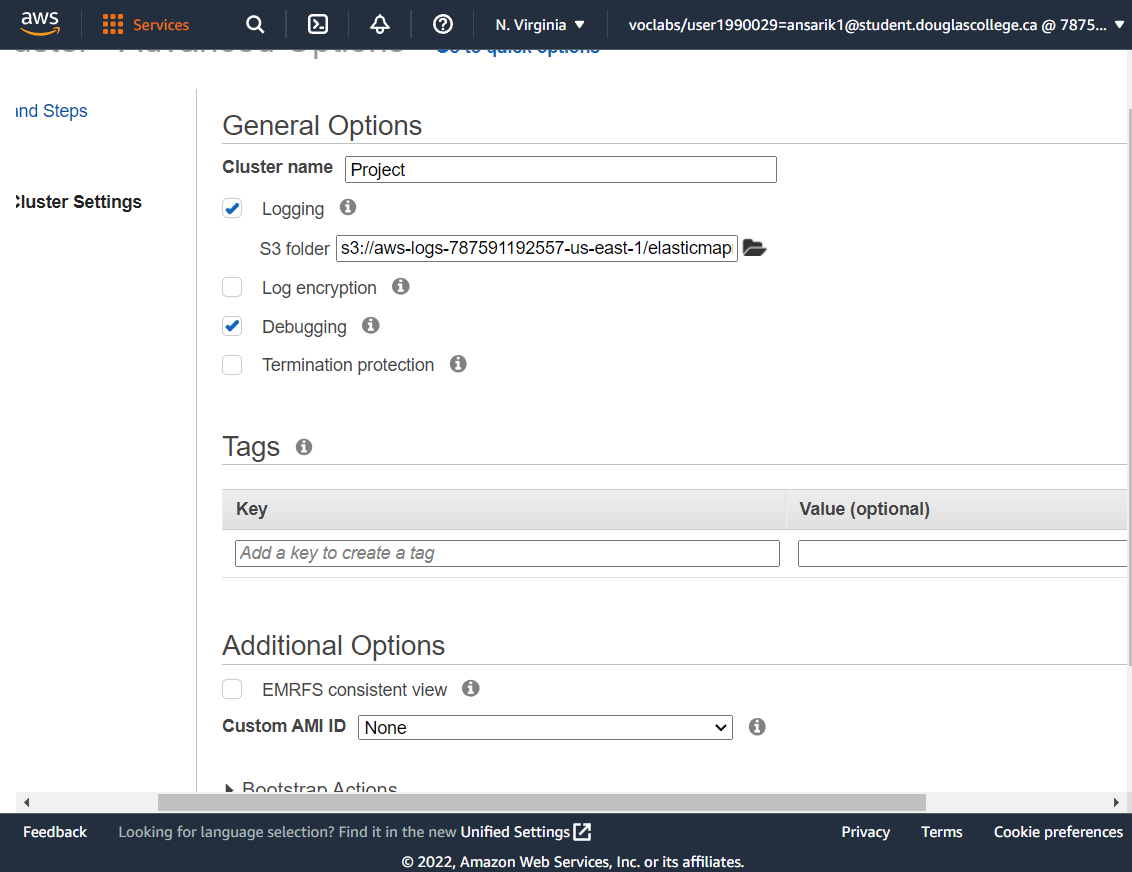
**STEP 2:** We will be going for a **2-node cluster** for our analysis & we will select **m5.large** instance type each for both master & core node.

A screenshot of a computer

Description automatically generated

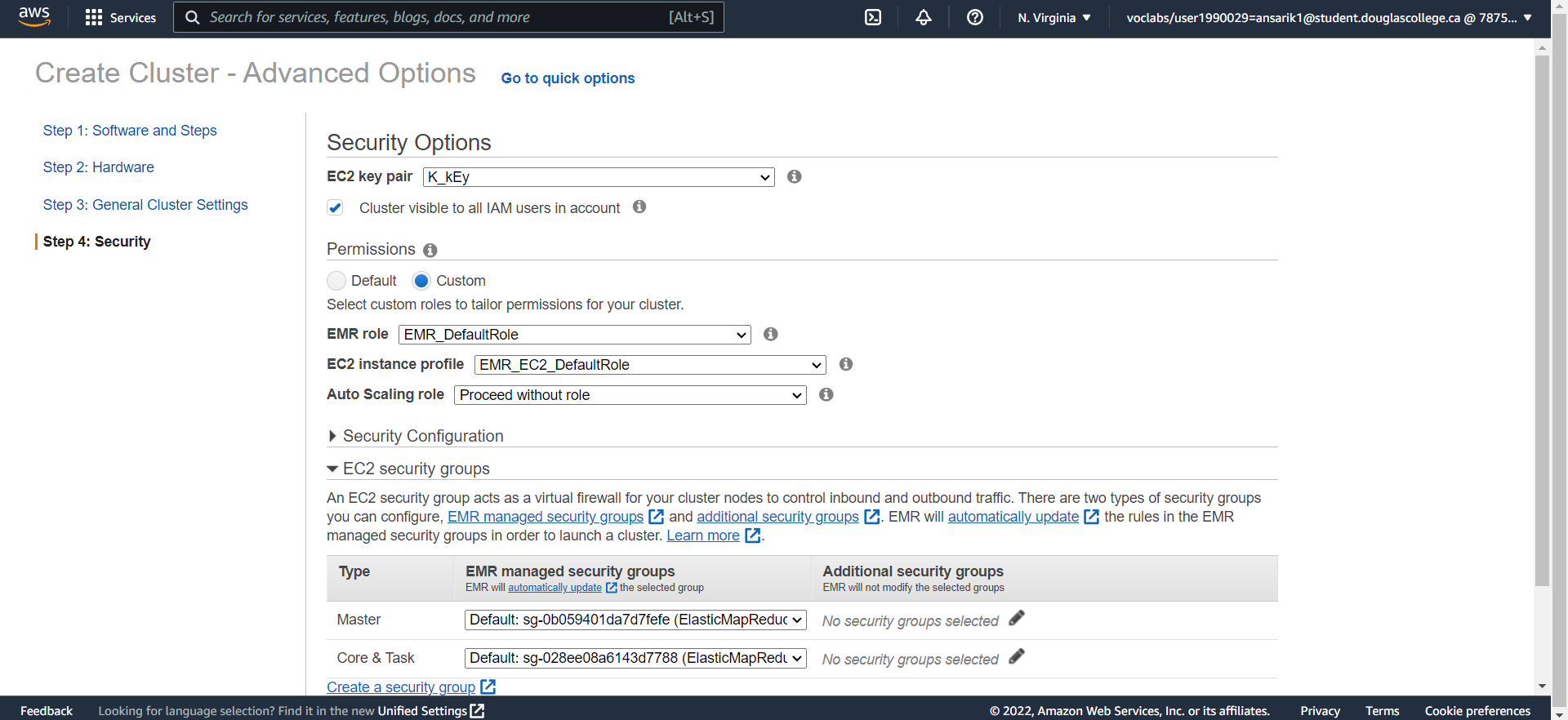


**Step 3:** Select a cluster name. Here we have taken the cluster name as **Project***.*

****

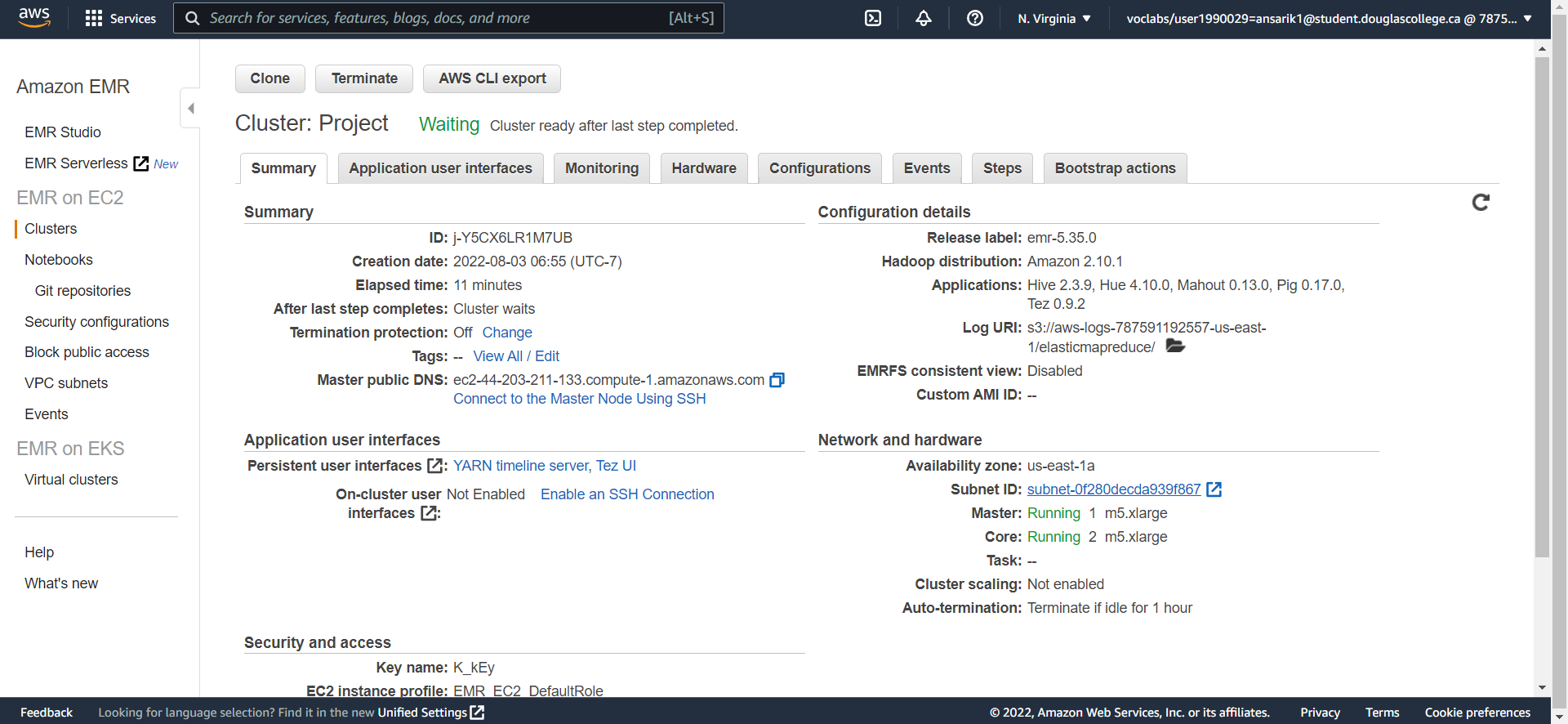


**Step 4:** Select an already created key-pair which will be used while connecting to master node and **K\_key** is the name of the key that we have used.

****

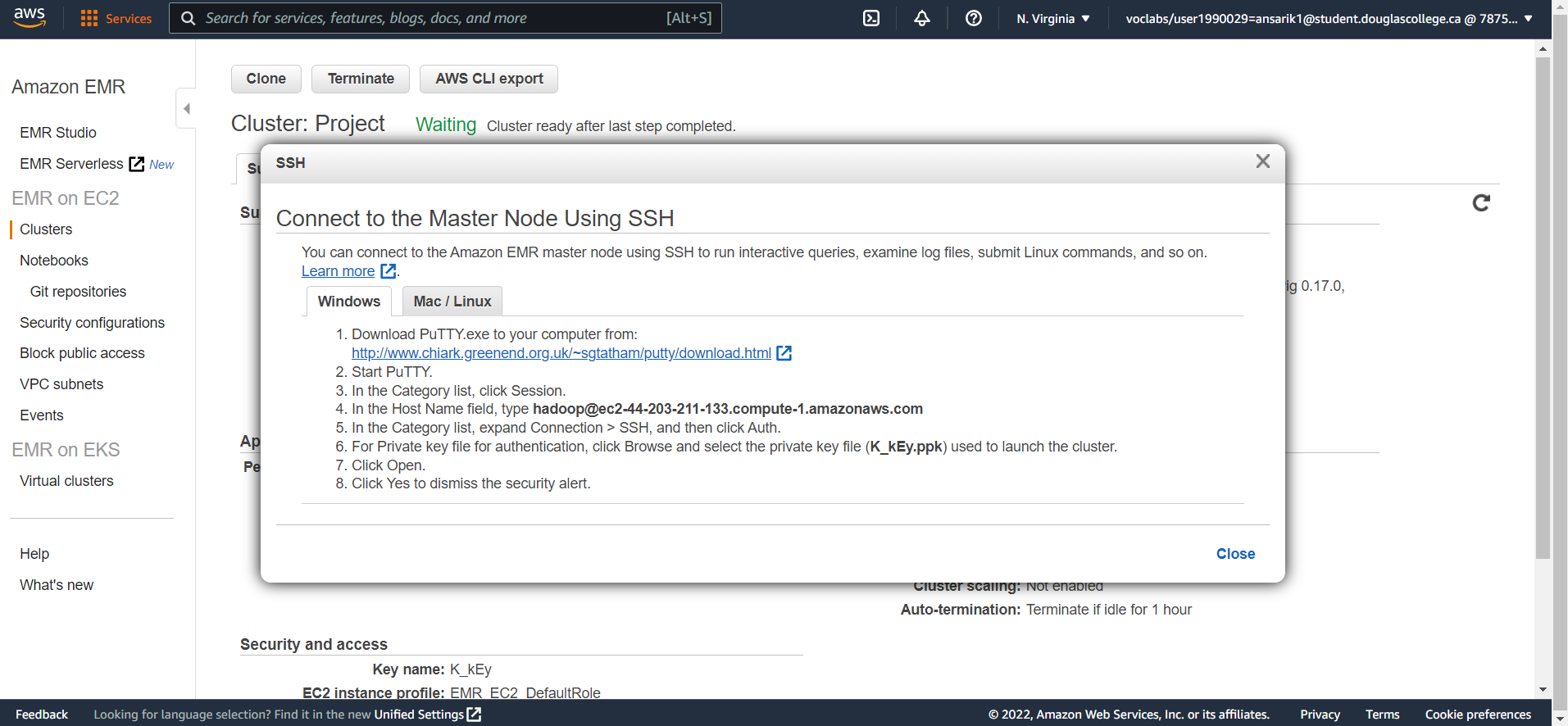


**Step 5:** Our cluster has been created successfully and is in **waiting state** which indicates its ready to be connected from the local system.

****

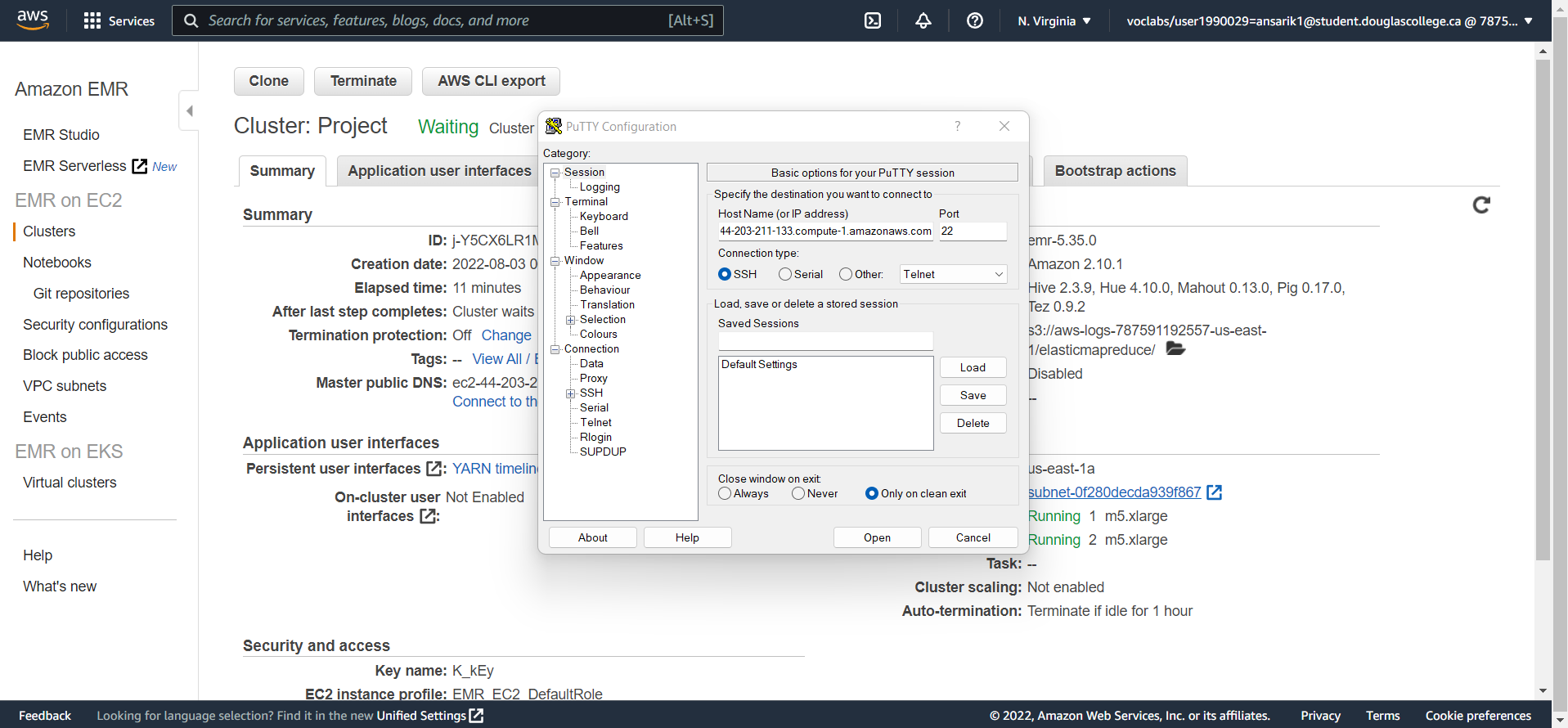


**Step 6:** Copy the highlighted link i.e., the master public DNS.

****



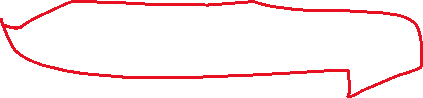
**Step 7:** Next, paste the address in the **Host Name** field.

** Step 7:** Click on **SSH** & then **Auth**. Give the location where the key pair is stored in the local system and click on **Open.**



**Graphical user interface, text, application

Description automatically generated**



**Step 8:** Next click on Accept which will open the SSH terminal.

**Graphical user interface, text, application

Description automatically generated**



After we have created an EMR cluster & successfully connected to it via putty we can begin to code in the SSH Terminal.

Text

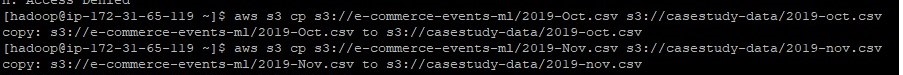
Description automatically generated

**DATA LOADING INTO S3 BUCKET**

*aws s3 cp s3://e-commerce-events-ml/2019-Oct.csv s3://casestudy-data/2019-oct.csv*

*aws s3 cp s3://e-commerce-events-ml/2019-Nov.csv s3://casestudy-data/2019-nov.csv*





using the above code, we could directly copy the data files into Hadoop without actually downloading the data files into our system, and the copying completes instantly.

**WORKING WITH HDFS**

* **Creating a folder in Hadoop**

*hadoop fs -mkdir /cstudy-folder*

*hadoop fs -ls /*

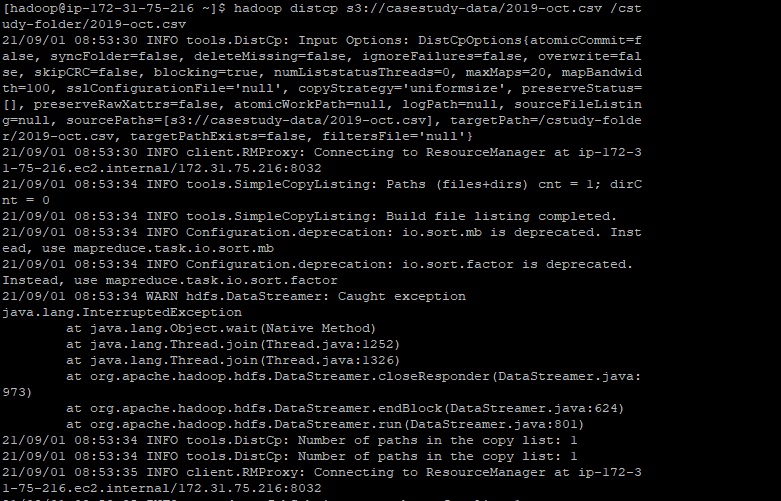
Graphical user interface, text

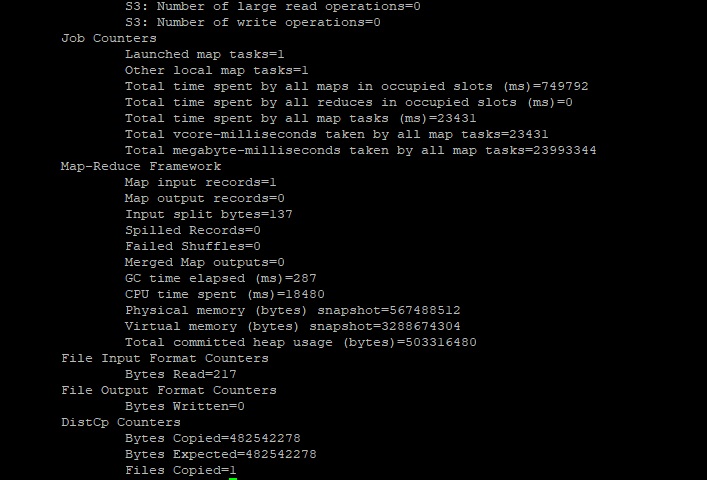
Description automatically generated

* **Copying October & November data from S3 bucket into HDFS**

*hadoop distcp s3://casestudy-data/2019-oct.csv /cstudy-folder/2019-oct.csv*

*hadoop distcp s3://casestudy-data/2019-nov.csv /cstudy-folder/2019-nov.csv*

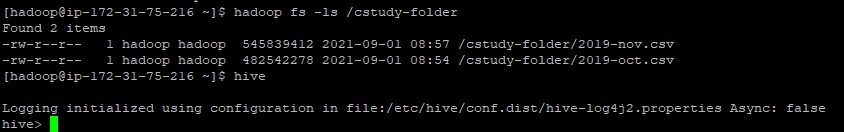




*Similarly, November data file was also copied, as shown in above image.*

* **Verifying if data has been copied successfully**

*hadoop fs -ls /cstudy-folder*

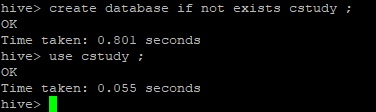


**WORKING ON HIVE**

* **Getting into Hive CLI – making a new database named *cstudy***

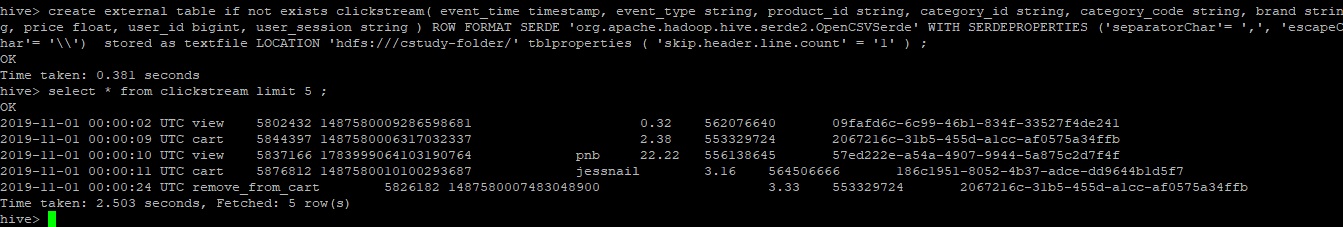
*create database if not exists cstudy ;*

*use cstudy ;*



* **Creating a common table named clickstream and storing both October & November data in it**

*create external table if not exists clickstream( 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'= ',', 'escapeChar'= '\\') stored as textfile LOCATION 'hdfs:///cstudy-folder/' tblproperties ( 'skip.header.line.count' = '1' ) ;*

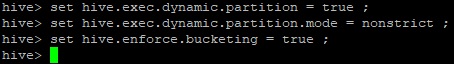
****

* **To create optimised table having partitions & buckets we need to enable some settings**

*set hive.exec.dynamic.partition = true ;*

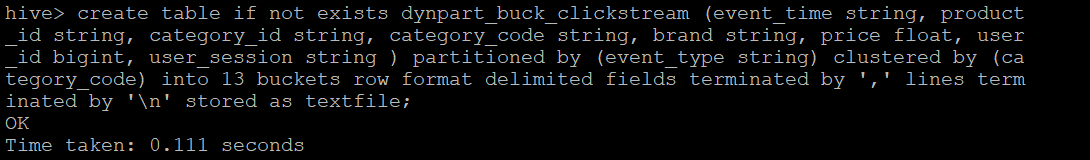
*set hive.exec.dynamic.partition.mode = nonstrict ;*

*set hive.enforce.bucketing = true ;*

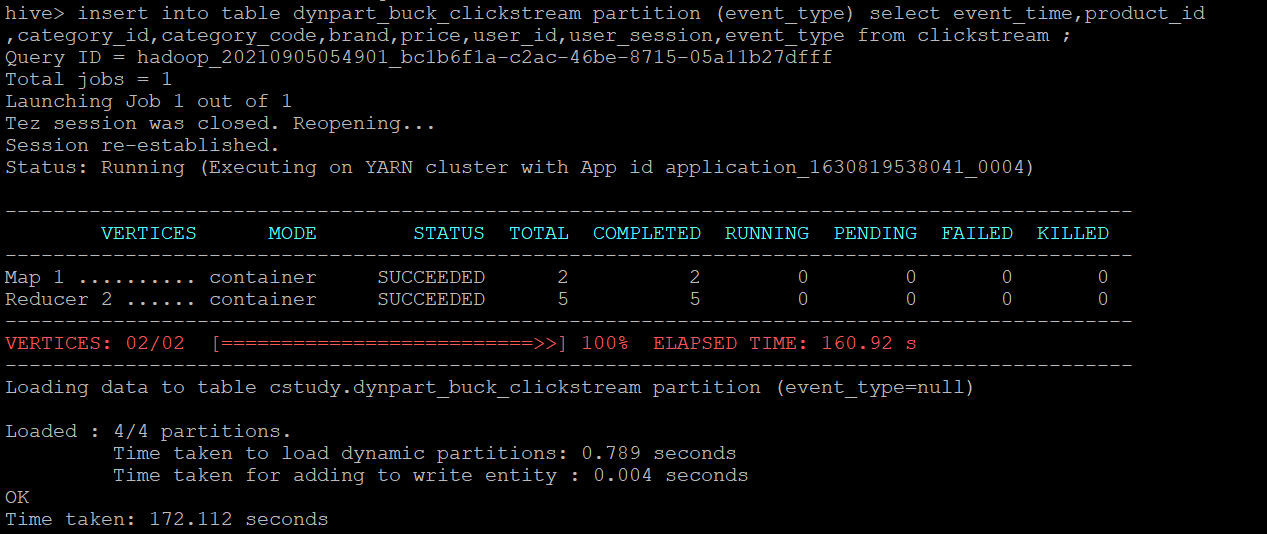
****

* **Creating table with dynamic partitions and buckets and inserting data into it**

*create table if not exists dynpart\_buck\_clickstream (event\_time string, 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 (category\_code) into 13 buckets row format delimited fields terminated by ',' lines terminated by '\n' stored as textfile;*

****

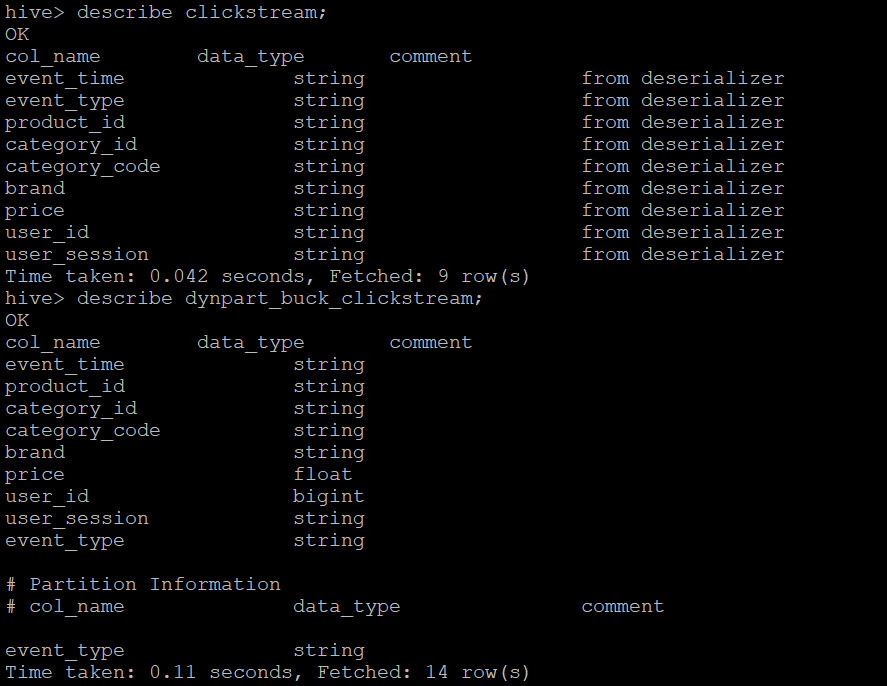
*insert into table dynpart\_buck\_clickstream partition (event\_type) select event\_time, product\_id,category\_id,category\_code,brand,price,user\_id,user\_session,event\_type from clickstream ;*

****

* **Describing both tables**

*describe clickstream;*

*describe dynpart\_buck\_clickstream;*

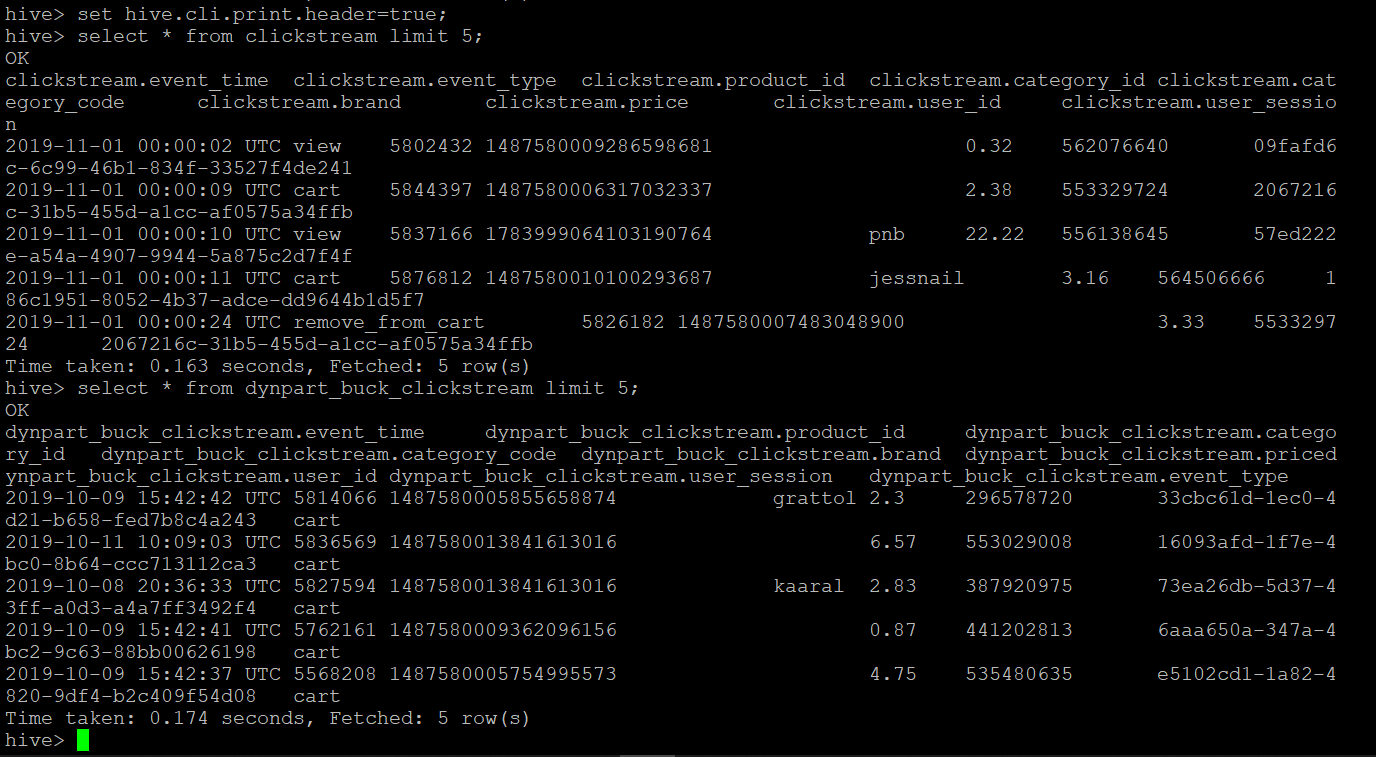
****

* **Checking data in both tables**

*set hive.cli.print.header=true;*

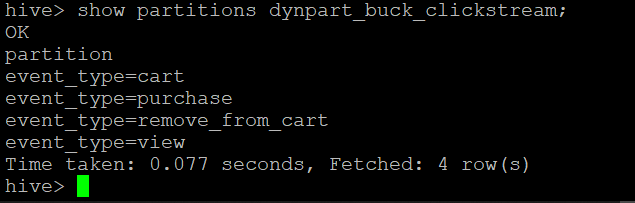
*select \* from clickstream limit 5 ;*

*select \* from dynpart\_buck\_clickstream limit 5 ;*

****

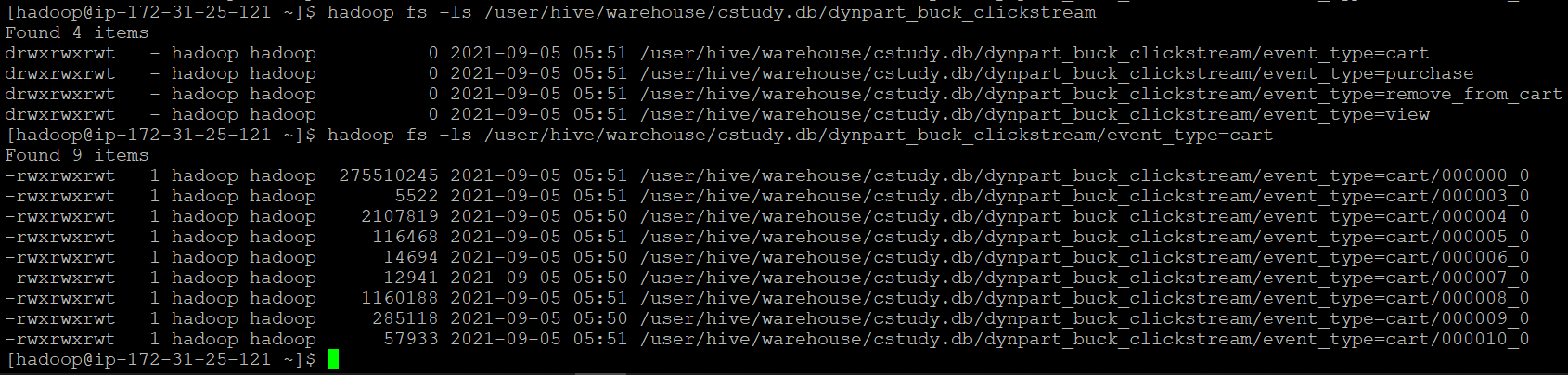
* **Checking if partitions were created successfully**

*show partitions dynpart\_buck\_clickstream;*

****

* **Checking in Hadoop if partitions were created successfully**

*hadoop fs -ls /user/hive/warehouse/cstudy.db/dynpart\_buck\_clickstream*

****

Overall, we have made two tables,

* One common table named **clickstream** which contains data of both October & November.
* One table with partitions & buckets named **dynpart\_buck\_clickstream** for optimised querying which also contains data of both October & November.

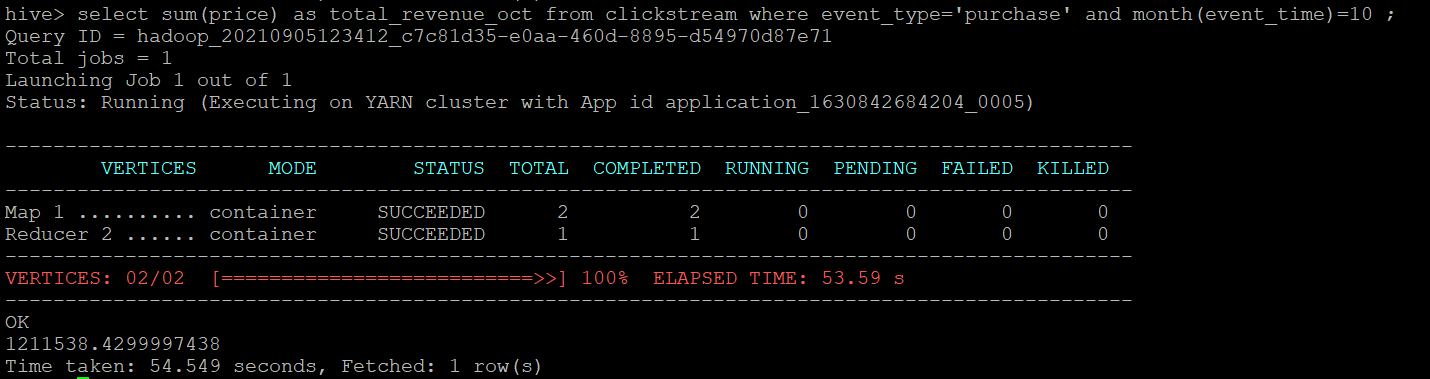
So, all the preparations are done & now we can move to query analysis-

**QUERY ANALYSIS**

1. **Find the total revenue generated due to purchases made in October.**
2. *Unoptimized query:*

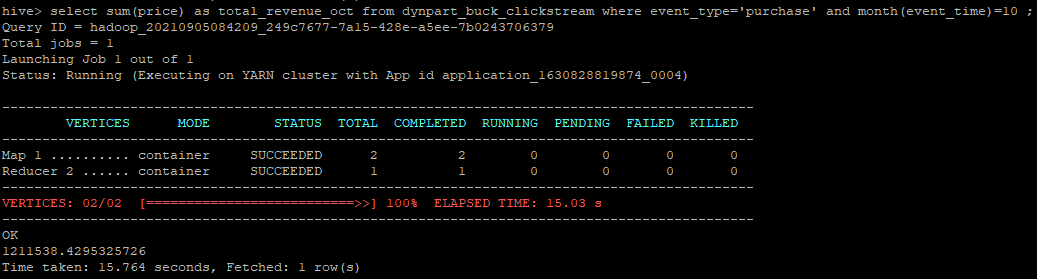
*select sum(price) as total\_revenue\_oct from clickstream where event\_type='purchase' and month(event\_time)=10 ;*

*and month(event\_time)=10 ;*



1. *Optimized query:*

*select sum(price) as total\_revenue\_oct from dynpart\_buck\_clickstream where event\_type='purchase' and month(event\_time)=10 ;*

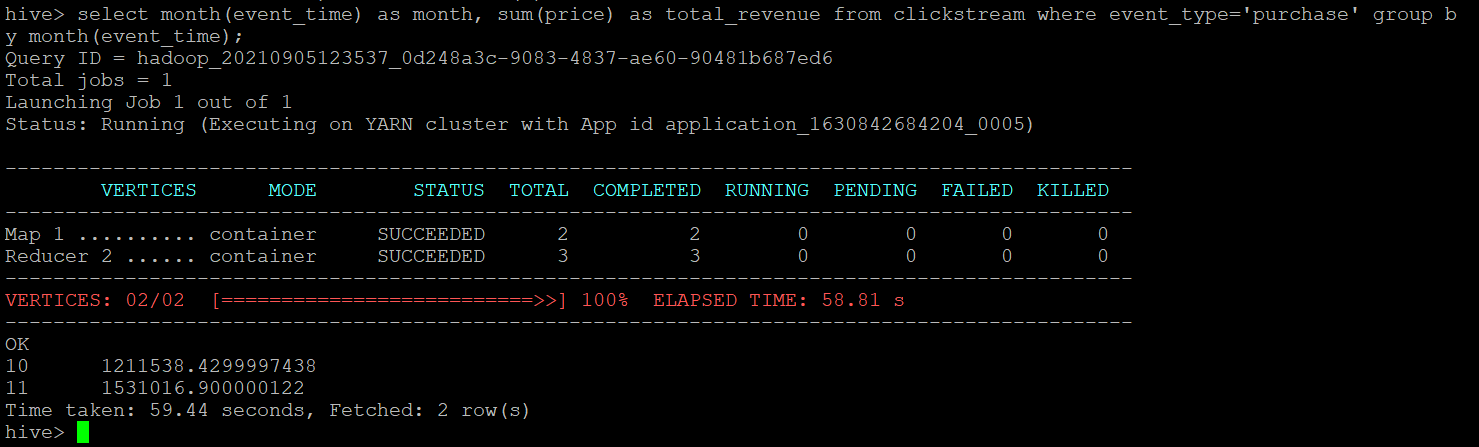


Optimized query using dynpart\_buck\_clickstream table

*The total revenue generated in October is* ***1211538.429.*** *Optimized query took* ***15.764 secs*** *while unoptimized query took* ***54.549 secs*** *to fetch the same result.*

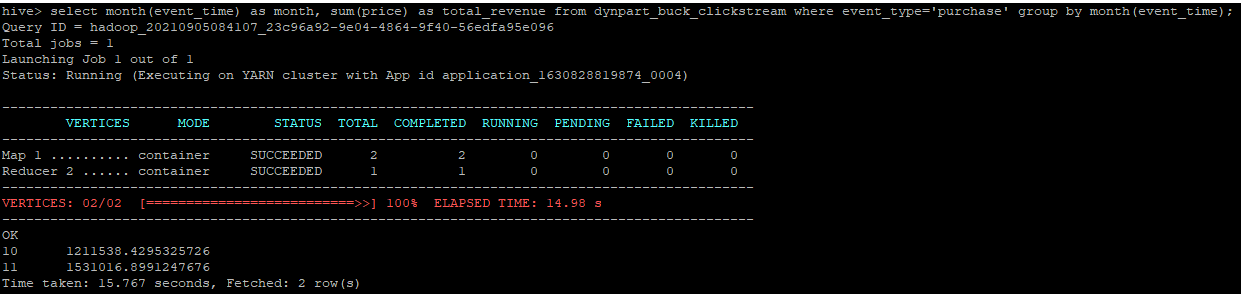
1. **Write a query to yield the total sum of purchases per month in a single output.**
2. *Unoptimized query:*

*select month(event\_time) as month, sum(price) as total\_revenue from clickstream where event\_type='purchase' group by month(event\_time);*



1. *Optimized query:*

*select month(event\_time) as month, sum(price) as total\_revenue from dynpart\_buck\_clickstream where event\_type='purchase' group by month(event\_time);*

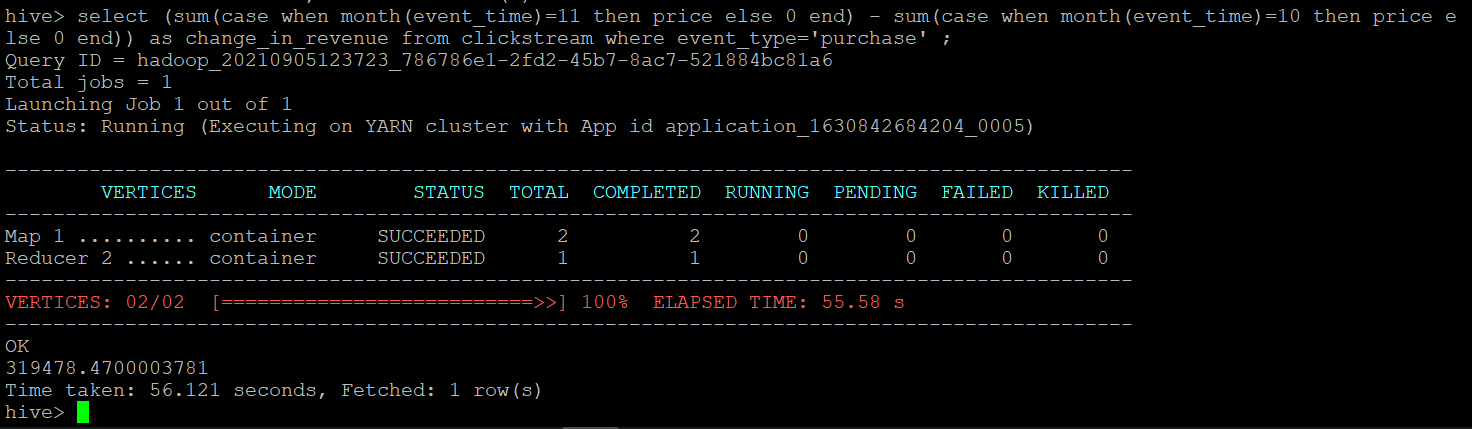


Optimized query using dynpart\_buck\_clickstream table

*Total sum of purchases for* ***October*** *is* ***1211538.429*** *while for the* ***November*** *it’s* ***1531016.899.*** *Optimized query took* ***15.767 secs*** *while unoptimized query took* ***59.44 secs****.*

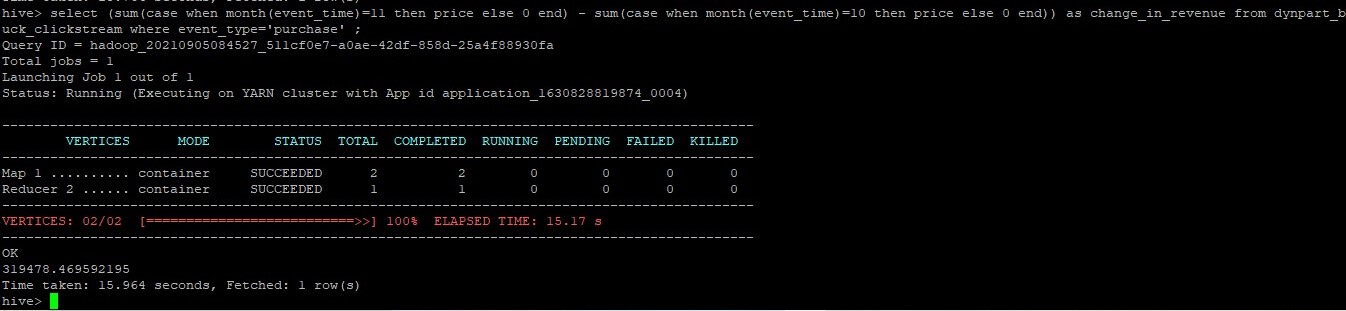
1. **Write a query to find the change in revenue due to purchases from October to November.**
2. *Unoptimized query:*

*select (sum(case when month(event\_time)=11 then price else 0 end) - sum(case when month(event\_time)=10 then price else 0 end)) as change\_in\_revenue from clickstream where event\_type='purchase' ;*

****

1. *Optimized query:*

*select (sum(case when month(event\_time)=11 then price else 0 end) - sum(case when month(event\_time)=10 then price else 0 end)) as change\_in\_revenue from dynpart\_buck\_clickstream where event\_type='purchase' ;*

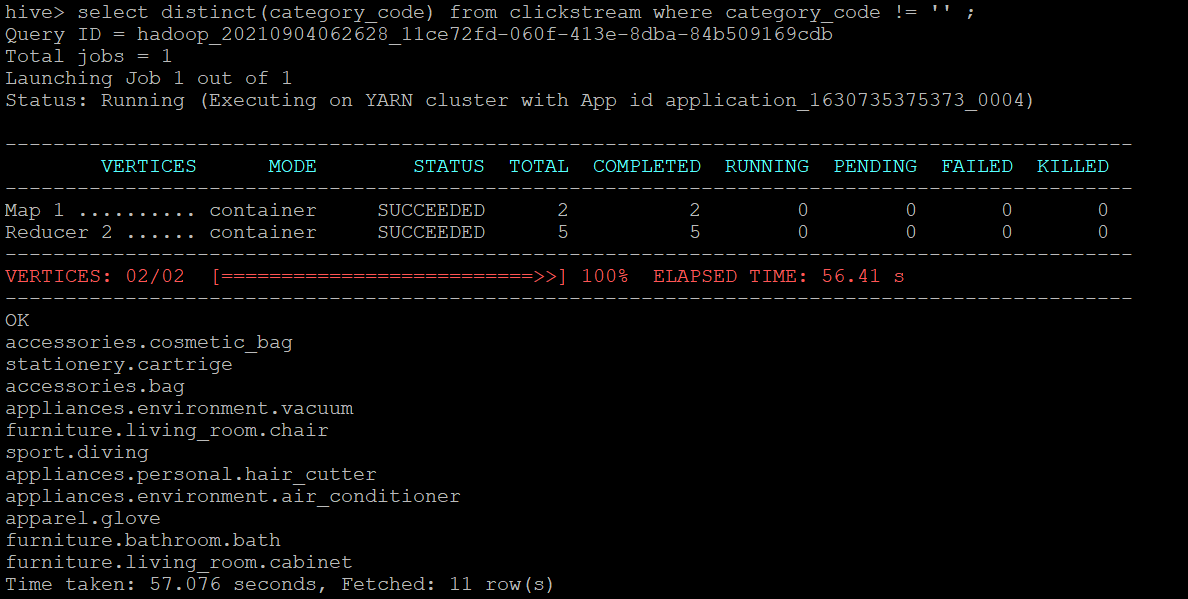


Optimized query using dynpart\_buck\_clickstream table

*Change in revenue is* ***319478.469****. Optimized query took* ***15.964 secs*** *while unoptimized query took* ***56.121 secs****.*

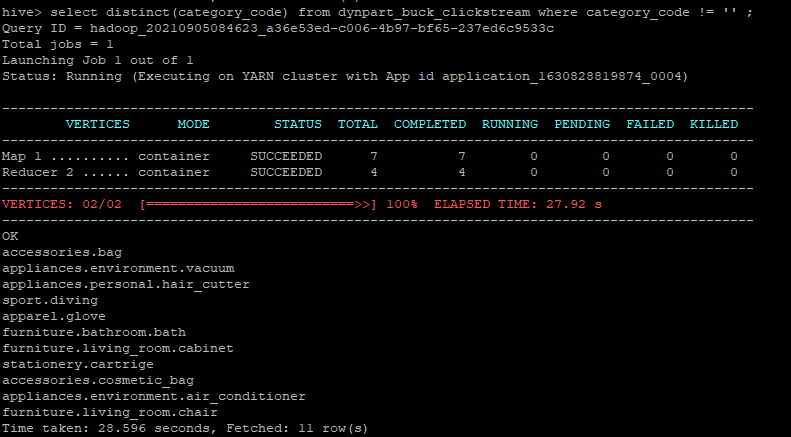
1. **Find distinct categories of products. Categories with null category code can be ignored.**
2. *Unoptimized query:*

*select distinct(category\_code) from clickstream where category\_code != '' ;*

**

1. *Optimized query:*

*select distinct(category\_code) from dynpart\_buck\_clickstream where category\_code != '' ;*

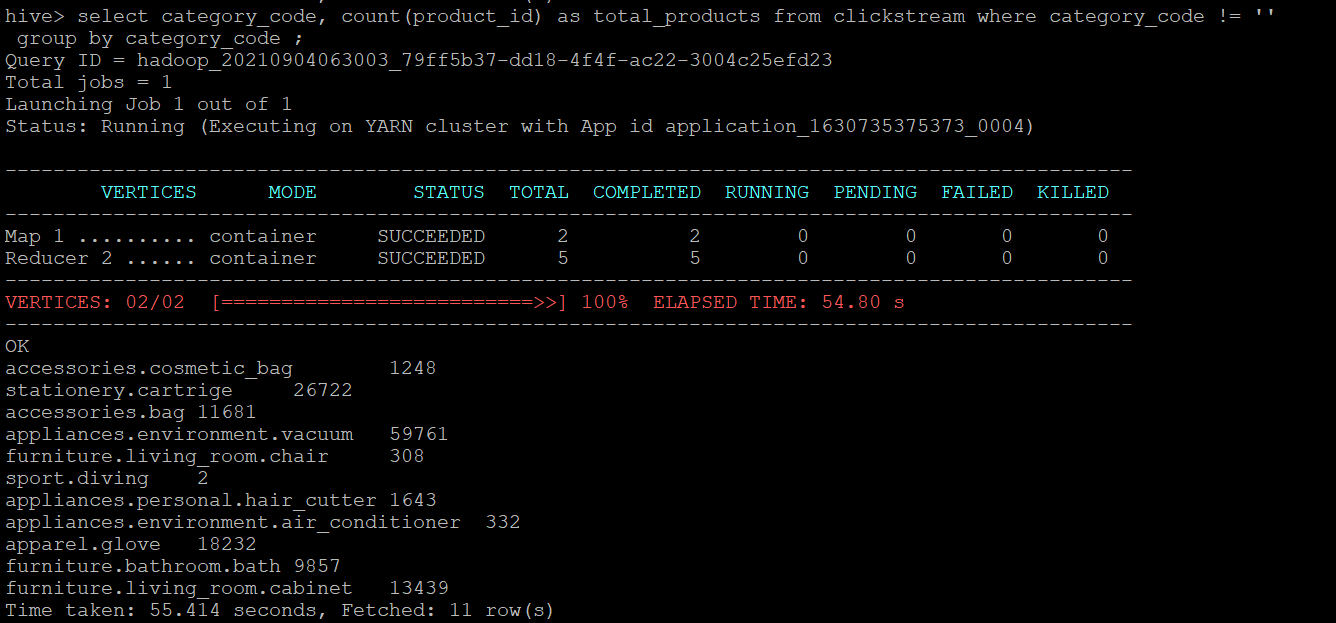
****

Optimized query using dynpart\_buck\_clickstream table

*There are* ***11 distinct categories****. Optimized query took* ***28.596 secs*** *while unoptimized query took* ***57.076 secs****.*

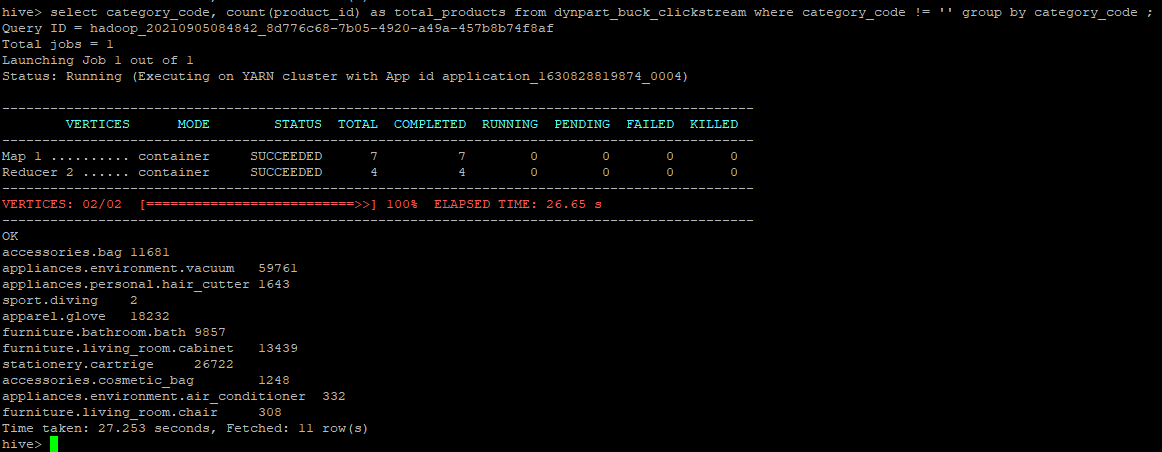
1. **Find the total number of products available under each category.**
2. *Unoptimized query:*

*select category\_code, count(product\_id) as total\_products from clickstream where category\_code != '' group by category\_code ;*

**

1. *Optimized query:*

*select category\_code, count(product\_id) as total\_products from dynpart\_buck\_clickstream where category\_code != '' group by category\_code ;*

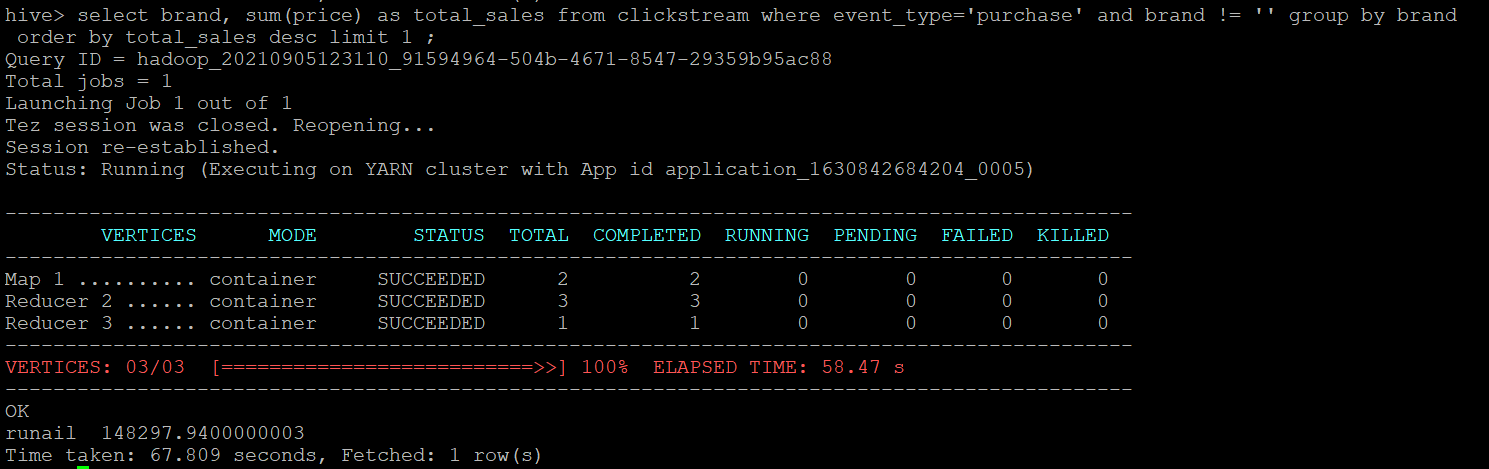
****

Optimized query using dynpart\_buck\_clickstream table

*Optimized query took* ***27.253 secs*** *while unoptimized query took* ***55.414 secs****.*

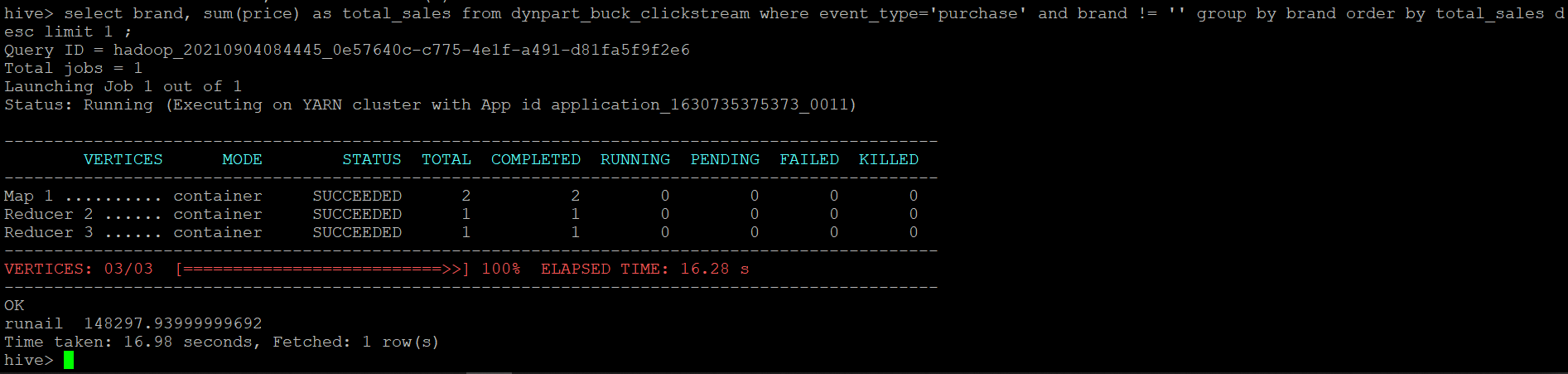
1. **Which brand had the maximum sales in October and November combined?**
2. *Unoptimized query:*

*select brand, sum(price) as total\_sales from clickstream where event\_type='purchase' and brand != '' group by brand order by total\_sales desc limit 1 ;*

**

1. *Optimized query:*

*select brand, sum(price) as total\_sales from dynpart\_buck\_clickstream where event\_type='purchase' and brand != '' group by brand order by total\_sales desc limit 1 ;*

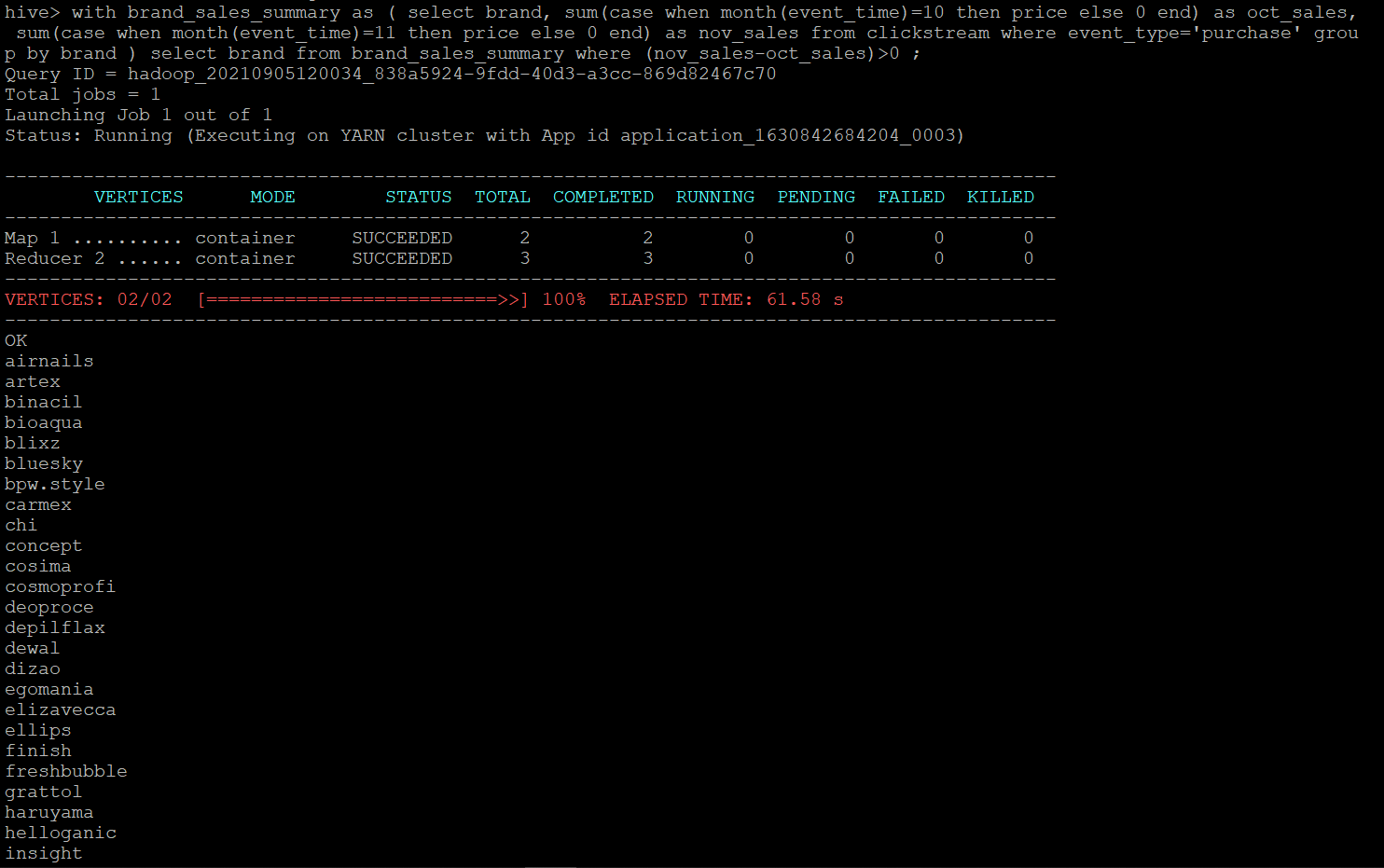


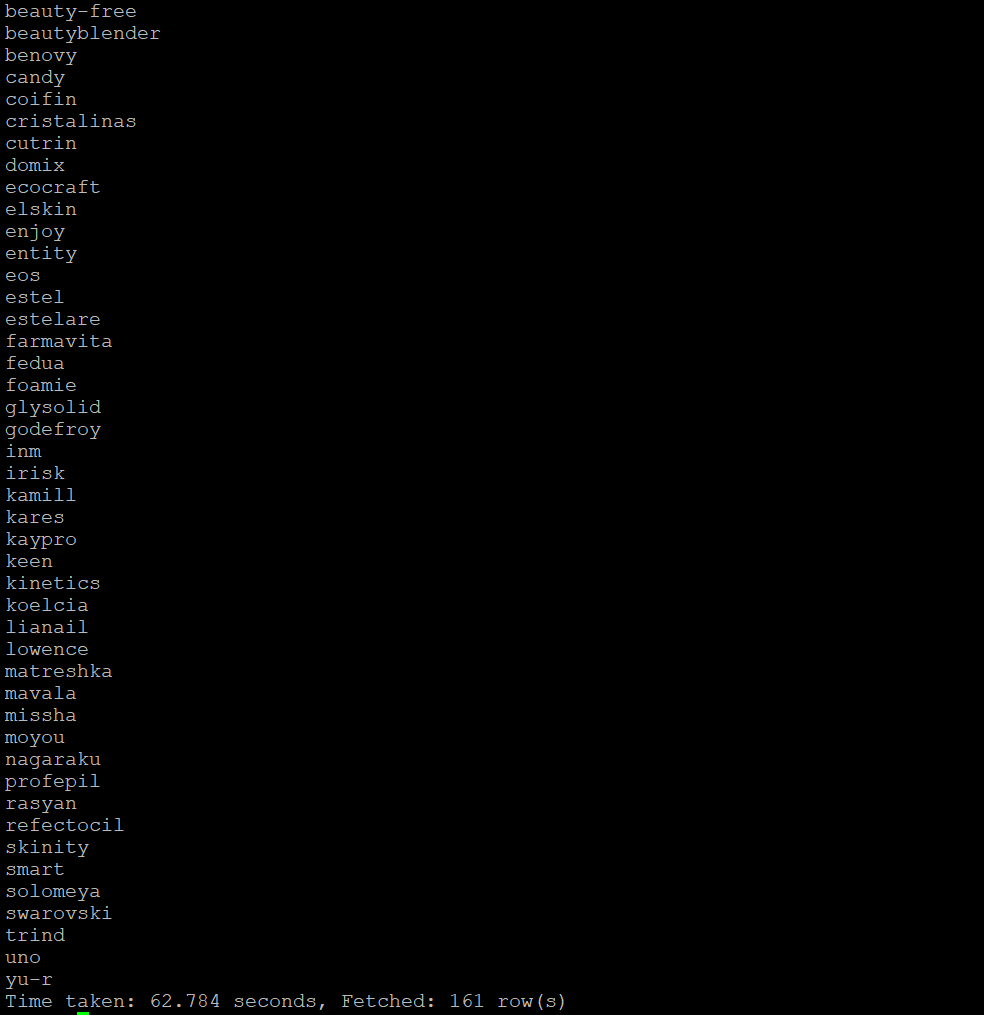
Optimized query using dynpart\_buck\_clickstream table

*The top brand is* ***runail*** *with* ***total\_sales 148297.939*** *& to show this optimized query took* ***16.98 secs*** *while unoptimized query took* ***67.809 secs.***

1. **Which brands increases their sales from October to November?**
2. *Unoptimized query:*

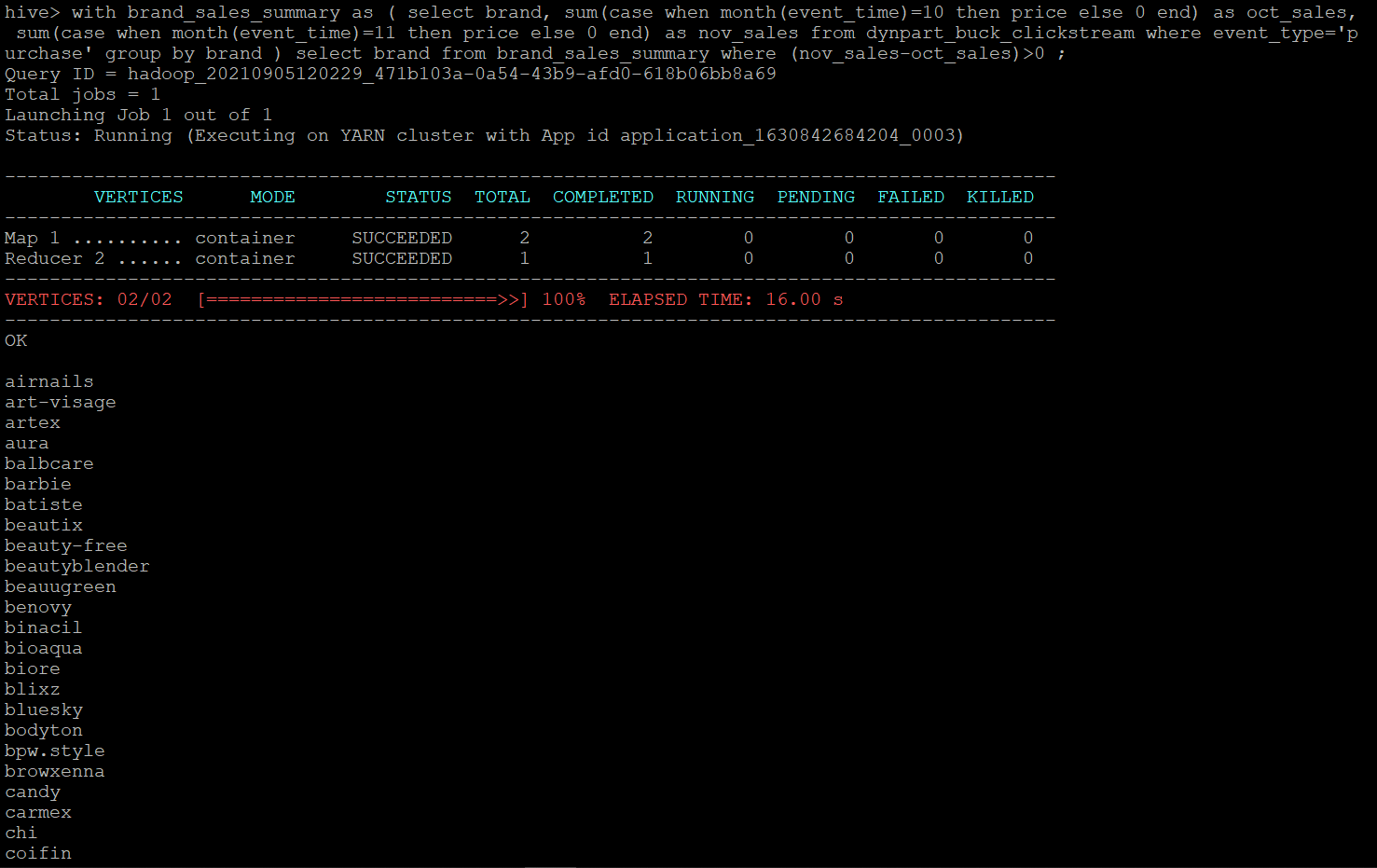
*with brand\_sales\_summary as ( select brand, sum(case when month(event\_time)=10 then price else 0 end) as oct\_sales, sum(case when month(event\_time)=11 then price else 0 end) as nov\_sales from clickstream where event\_type='purchase' group by brand ) select brand from brand\_sales\_summary where (nov\_sales-oct\_sales)>0 ;*

****

****

1. *Optimized query:*

*with brand\_sales\_summary as ( select brand, sum(case when month(event\_time)=10 then price else 0 end) as oct\_sales, sum(case when month(event\_time)=11 then price else 0 end) as nov\_sales from dynpart\_buck\_clickstream where event\_type='purchase' group by brand ) select brand from brand\_sales\_summary where (nov\_sales-oct\_sales)>0 ;*



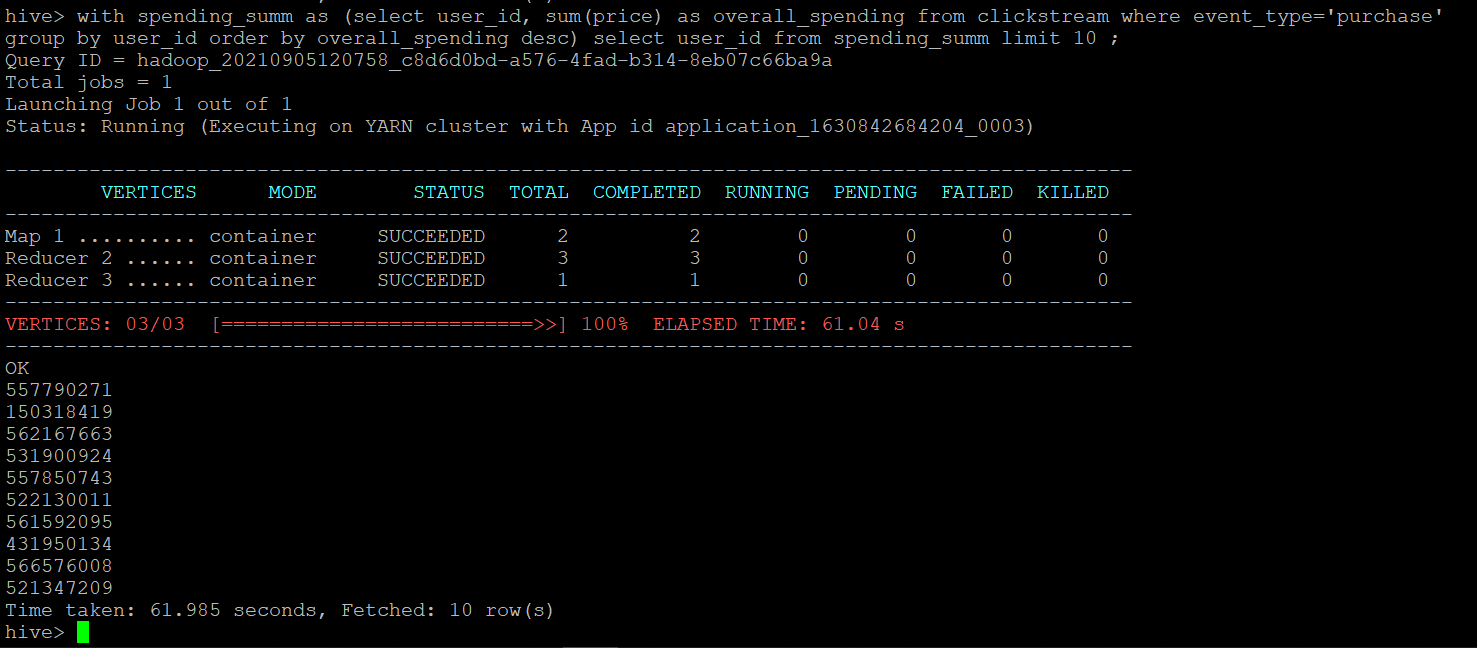


*There is a total of* ***161 brands*** *that had increased sales from October to November.*

*To show this optimized query took* ***17.222 secs*** *while unoptimized query took* ***62.784 secs.***

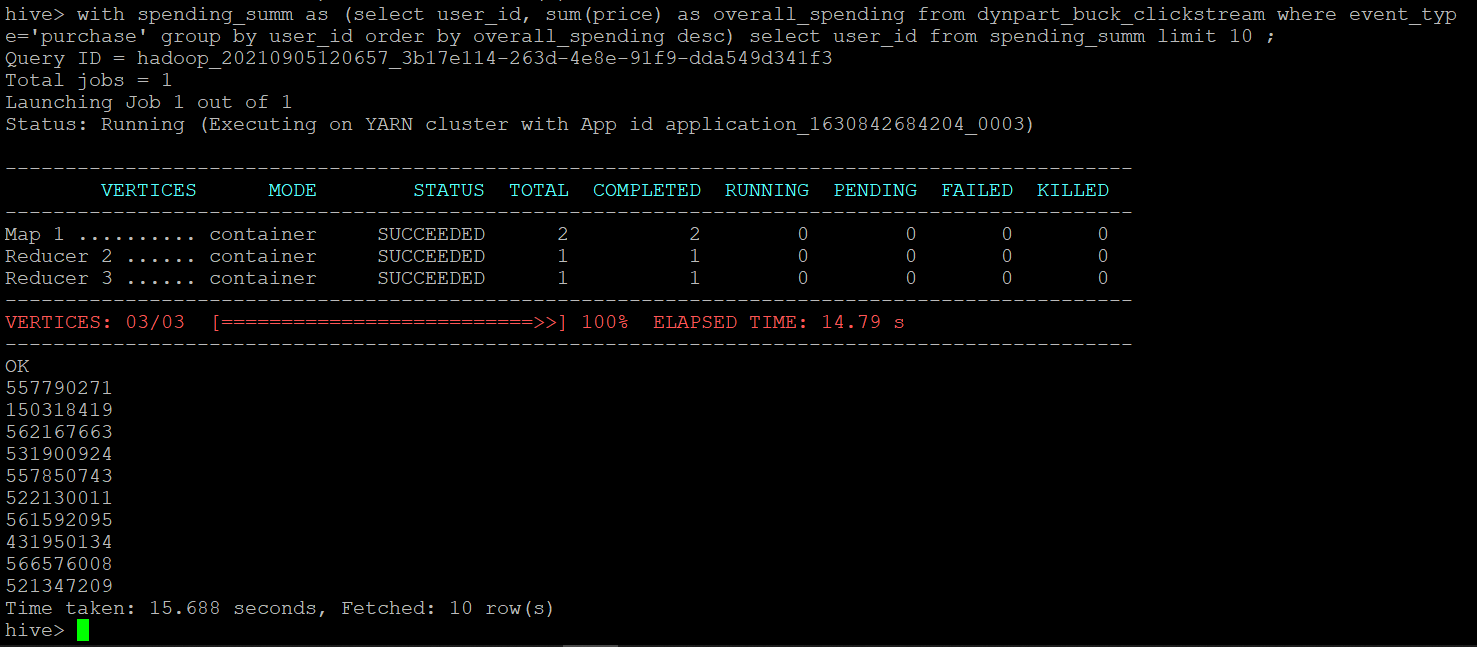
1. **Your company wants to reward the top 10 users of its website with a Golden Customer plan. Write a query to generate a list of top 10 users who spend the most.**
2. *Unoptimized query:*

*with spending\_summ as (select user\_id, sum(price) as overall\_spending from clickstream where event\_type='purchase' group by user\_id order by overall\_spending desc) select user\_id from spending\_summ limit 10 ;*

**

1. *Optimized query:*

*with spending\_summ as (select user\_id, sum(price) as overall\_spending from dynpart\_buck\_clickstream where event\_type='purchase' group by user\_id order by overall\_spending desc) select user\_id from spending\_summ limit 10 ;*

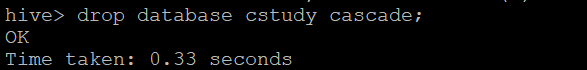


Optimized query using dynpart\_buck\_clickstream table

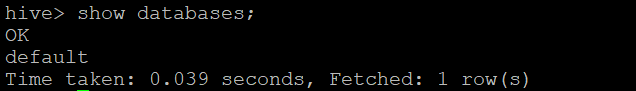
*So above are the top 10 customers who spend the most and should be awarded with golden customer plan. To show this the optimized query took* ***15.688 secs*** *while unoptimized query took* ***61.985 secs****.*

* **Dropping database**

*drop database cstudy cascade;*

****

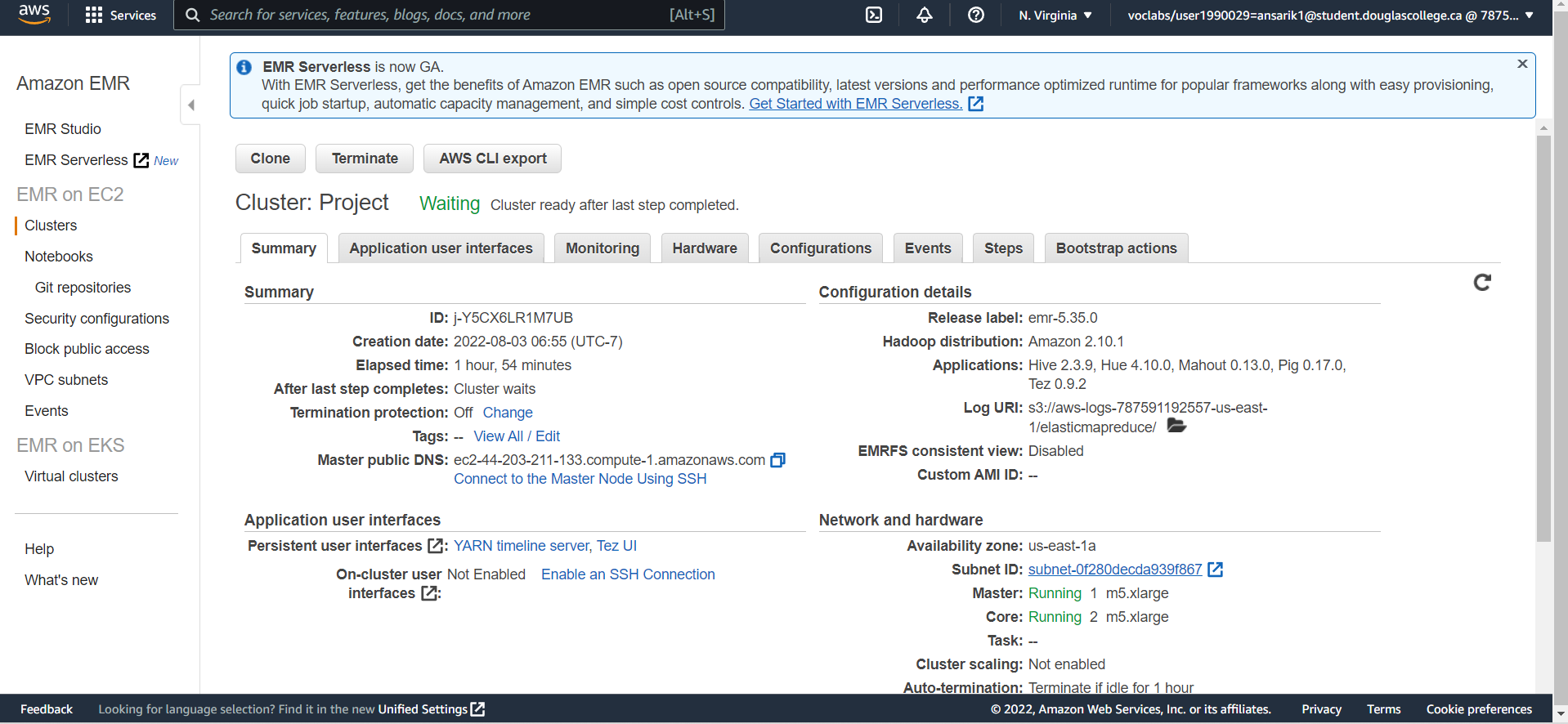
*show databases;*

****

We are done with our analysis. Finally, we will move towards terminating our cluster

**TERMINATING THE EMR CLUSTER**

**Step 1:** Click on Terminate.

****



**Step 2:** A security/confirmation message will pop up. Click on Terminate.

Graphical user interface, text, application, email

Description automatically generated

**Graphical user interface

Description automatically generated**

The Cluster has been terminated.