**Case study approach:**

1. **Input files**

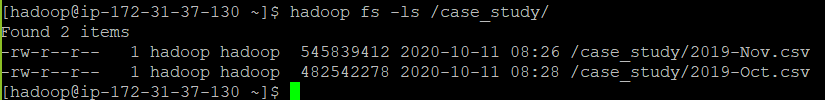
* 2019-Nov.csv
* 2019-Oct.csv

1. **Store given CSV input files inside folder “casestudyinput” in S3 bucket**

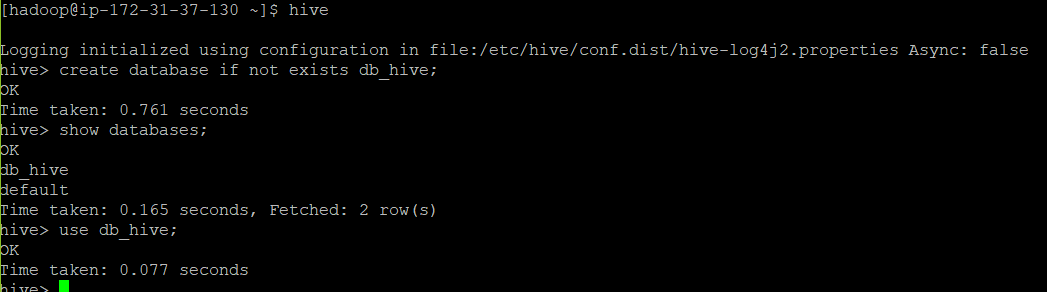
* s3://hivecasestudy/casestudyinput/

1. **Create folder “case\_study” in HDFS and copy files from S3 into case\_study folder**

* hadoop fs -mkdir /case\_study
* hadoop distcp s3://hivecasestudy/casestudyinput/2019-Nov.csv /case\_study/2019-Nov.csv
* hadoop distcp s3://hivecasestudy/casestudyinput/2019-Oct.csv /case\_study/2019-Oct.csv

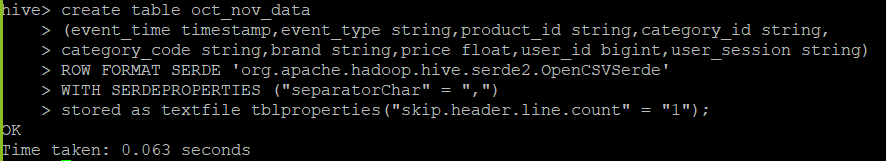


1. **Launch Hive and create database with name “db\_hive”**

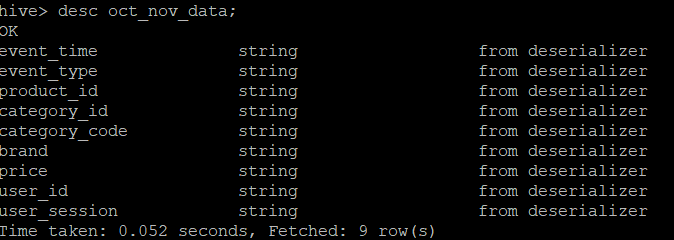


1. **Create table with name “oct\_nov\_data” using CSVserde**

* create table oct\_nov\_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");

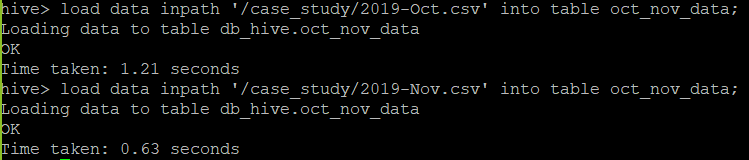


* desc oct\_nov\_data;

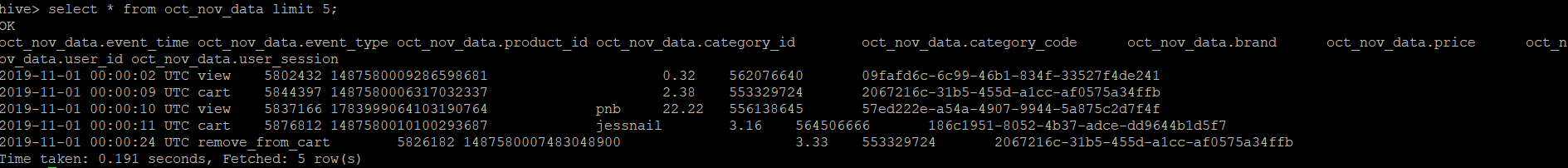


1. **Loading data from HDFS into Hive table “oct\_nov\_data”**

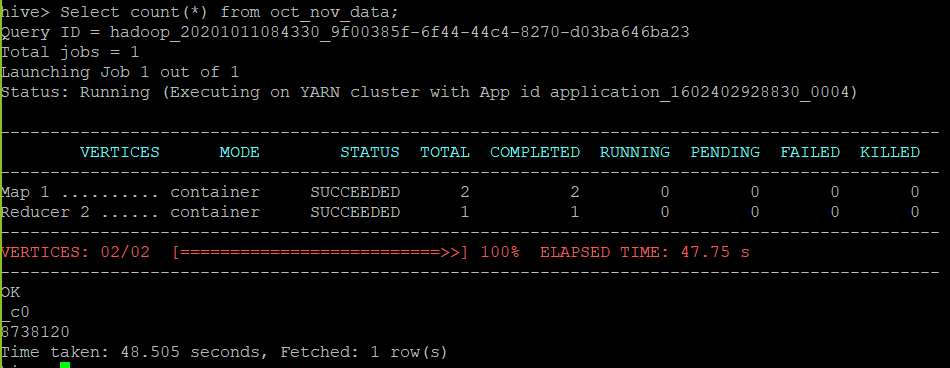
* load data inpath '/case\_study/2019-Oct.csv' into table oct\_nov\_data;
* load data inpath '/case\_study/2019-Nov.csv' into table oct\_nov\_data;



* **Make headers visible by using below command**
  + set hive.cli.print.header=true;
* **Check data**
  + select \* from oct\_nov\_data limit 5;

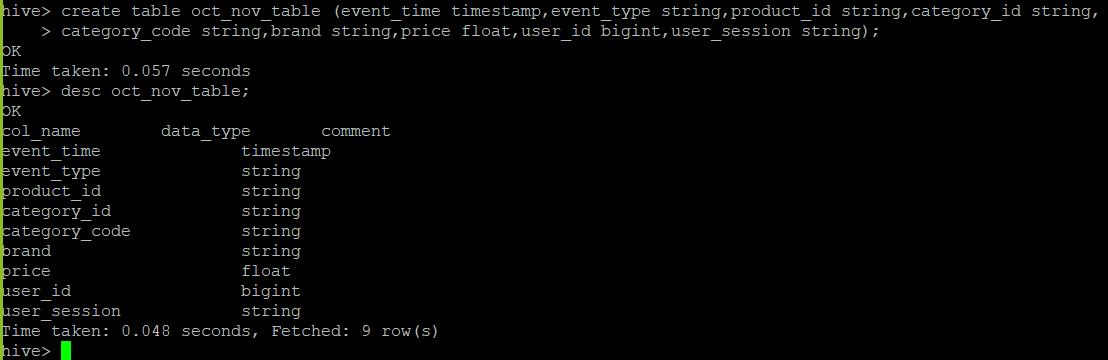


* **Count rows in data**
  + Select count(\*) from oct\_nov\_data;



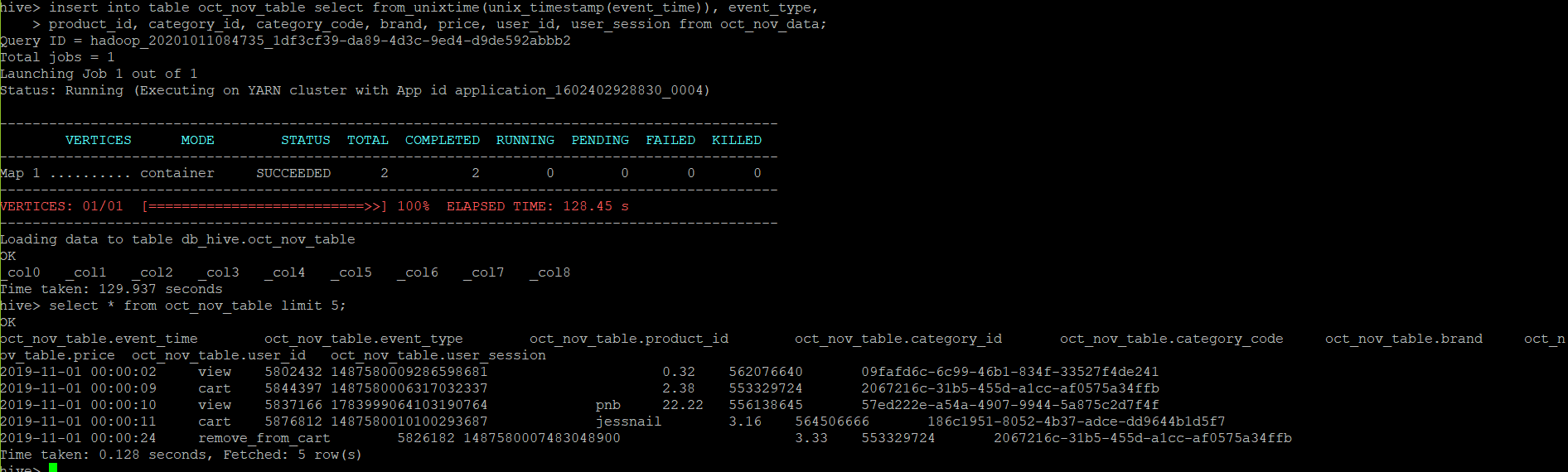
1. **Create new table “oct\_nov\_table” to get the desired datatype for (event\_time, price and user\_id)**

* create table oct\_nov\_table (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);
* desc oct\_nov\_table



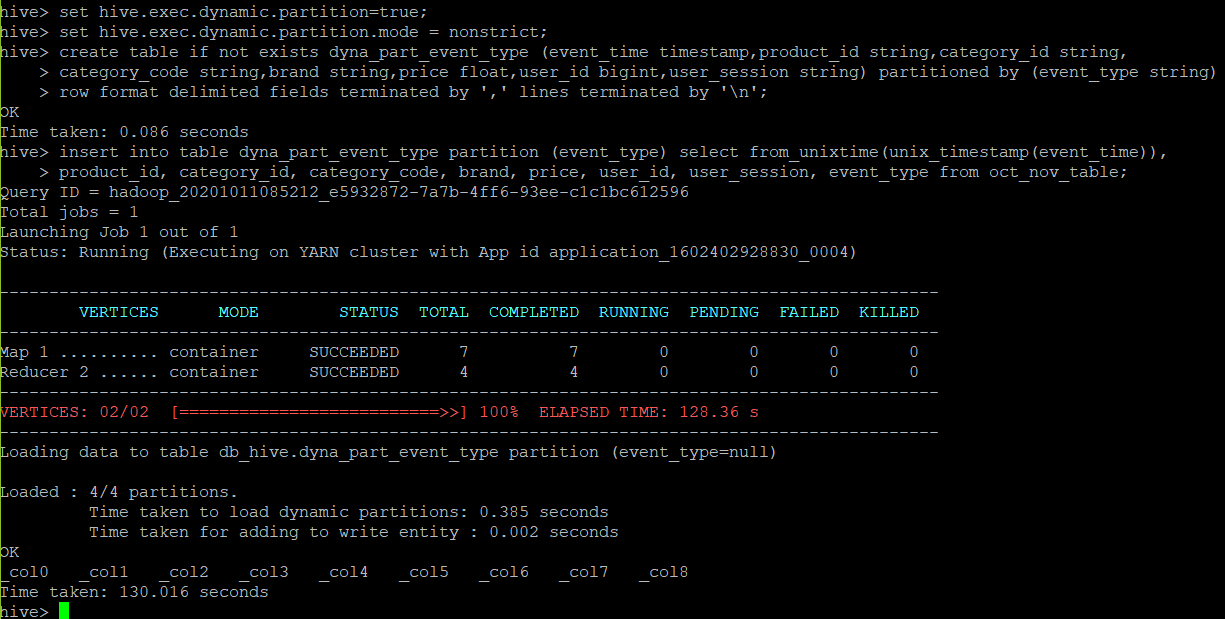
1. **Insert data from “oct\_nov\_data” into “oct\_nov\_table” with desired data type**

* insert into table oct\_nov\_table select from\_unixtime(unix\_timestamp(event\_time)), event\_type, product\_id, category\_id, category\_code, brand, price, user\_id, user\_session from oct\_nov\_data;
* select \* from oct\_nov\_table limit 5;

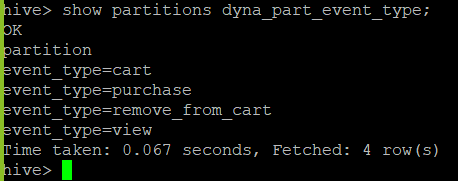


1. **Create table “dyna\_part\_event\_type” with dynamic partitioning on event\_type to optimize the query performance**

* **Enable the dynamic partitioning in Hive** 
  + set hive.exec.dynamic.partition=true;
  + set hive.exec.dynamic.partition.mode = nonstrict;
* **Create table dyna\_part\_event\_type**
  + create table if not exists dyna\_part\_event\_type (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) row format delimited fields terminated by ',' lines terminated by '\n';
* **Insert data into table dyna\_part\_event\_type from oct\_nov\_table**
  + insert into table dyna\_part\_event\_type partition (event\_type) select from\_unixtime(unix\_timestamp(event\_time)), product\_id, category\_id, category\_code, brand, price, user\_id, user\_session, event\_type from oct\_nov\_table;



* **show partitions dyna\_part\_event\_type;**



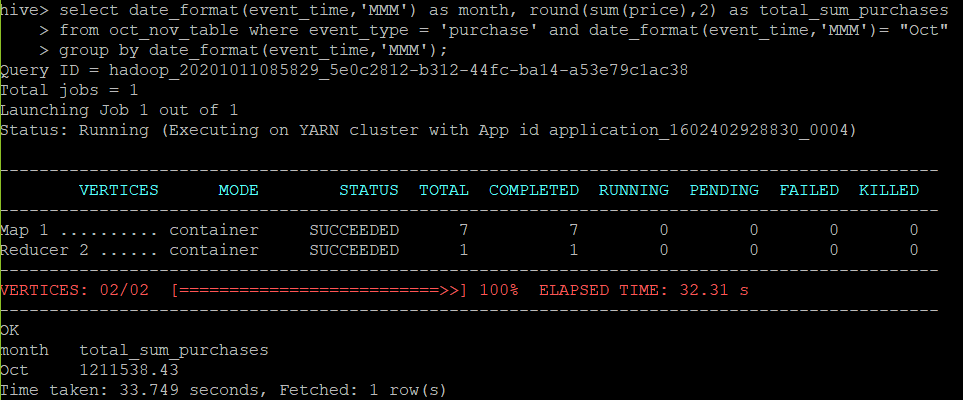
1. **Query 1: Find the total revenue generated due to purchases made in October**

* **Without optimization:**
  + select date\_format(event\_time,'MMM') as month, round(sum(price),2) as total\_sum\_purchases

from oct\_nov\_table

where event\_type = 'purchase' and date\_format(event\_time,'MMM')= "Oct"

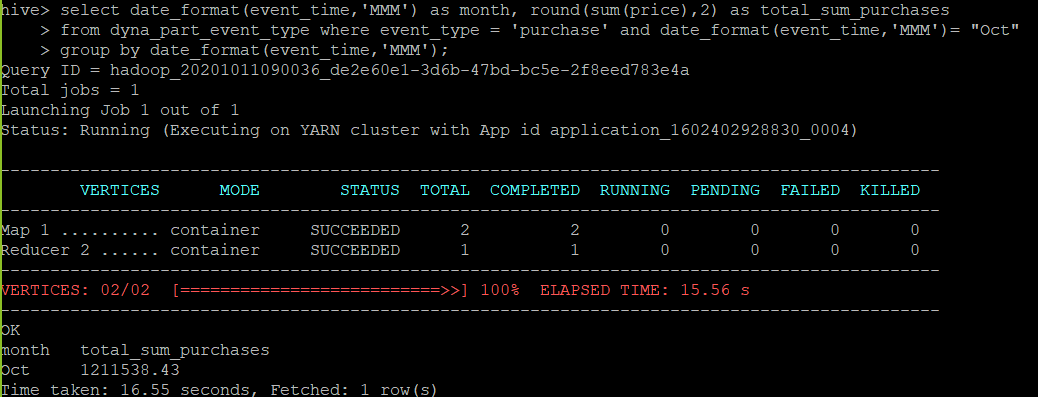
group by date\_format(event\_time,'MMM');



* **With optimization:**
  + select date\_format(event\_time,'MMM') as month, round(sum(price),2) as total\_sum\_purchases

from dyna\_part\_event\_type where event\_type = 'purchase' and date\_format(event\_time,'MMM')= "Oct"

group by date\_format(event\_time,'MMM');



Insights:

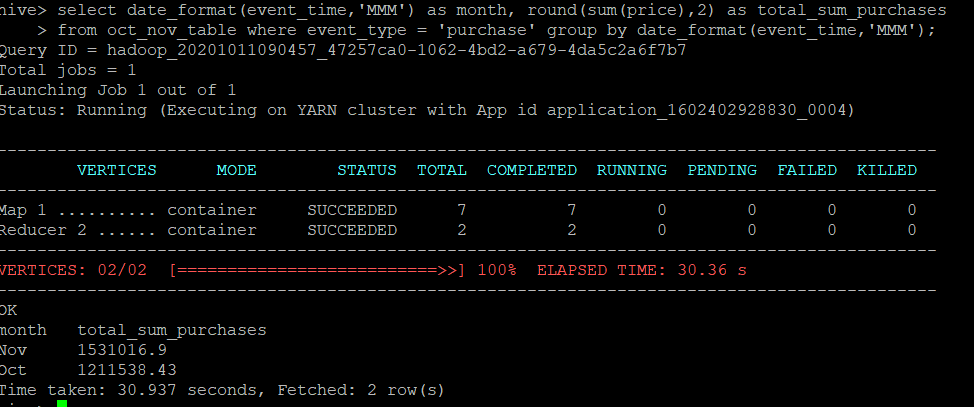
Total revenue generated due to purchases made in October is 1211538.43

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

* **Without optimization:**
  + select date\_format(event\_time,'MMM') as month, round(sum(price),2) as total\_sum\_purchases

from oct\_nov\_table where event\_type = 'purchase'

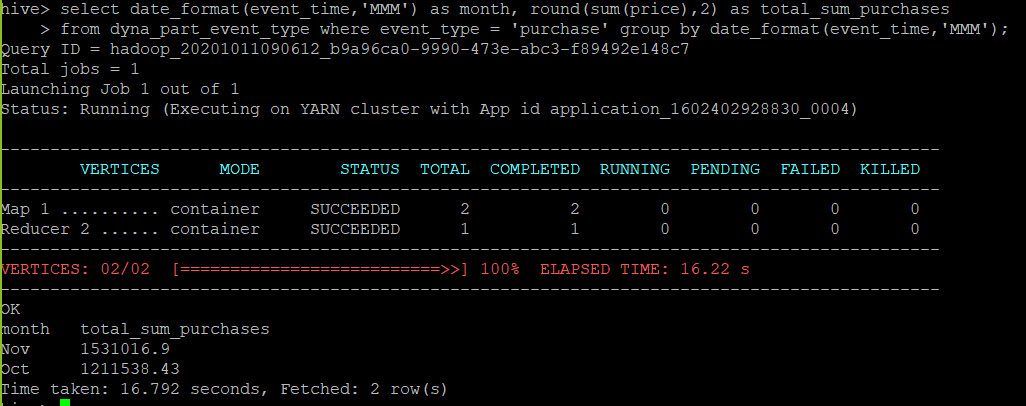
group by date\_format(event\_time,'MMM');



* **With optimization:**
  + select date\_format(event\_time,'MMM') as month, round(sum(price),2) as total\_sum\_purchases

from dyna\_part\_event\_type where event\_type = 'purchase'

group by date\_format(event\_time,'MMM');



Insights:

Total sum of purchases per month:

Oct - 1211538.43

Nov - 1531016.90

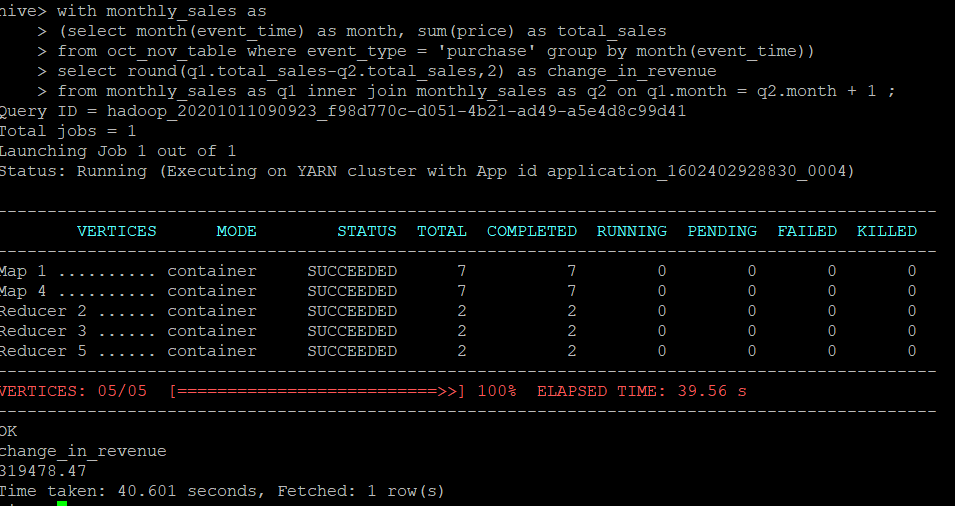
1. **Query 3: Write a query to find the change in revenue generated due to purchases from October to November.**

* **Without optimization:**
  + with monthly\_sales as

(select month(event\_time) as month, sum(price) as total\_sales

from oct\_nov\_table where event\_type = 'purchase' group by month(event\_time))

select round(q1.total\_sales-q2.total\_sales,2) as change\_in\_revenue from monthly\_sales as q1 inner join monthly\_sales as q2 on q1.month = q2.month + 1 ;

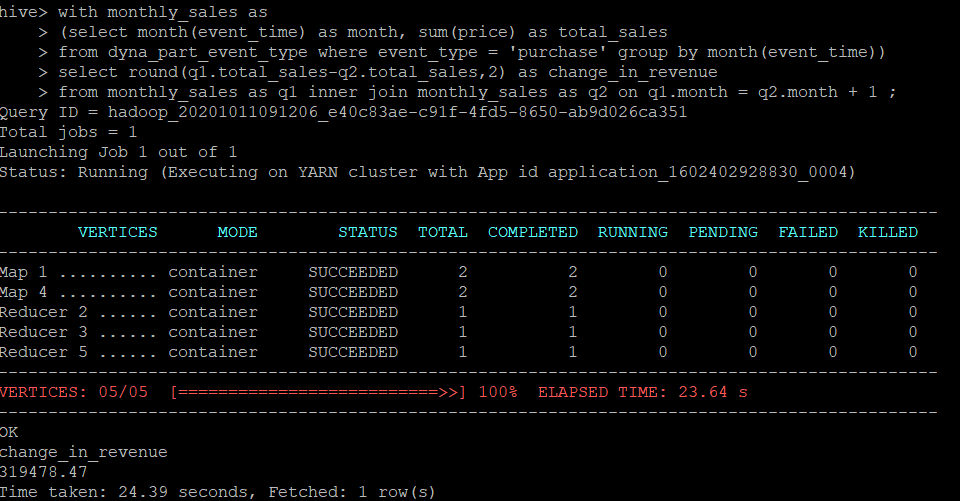


* **With optimization:**
  + with monthly\_sales as

(select month(event\_time) as month, sum(price) as total\_sales

from dyna\_part\_event\_type where event\_type = 'purchase' group by month(event\_time))

select round(q1.total\_sales-q2.total\_sales,2) as change\_in\_revenue from monthly\_sales as q1 inner join monthly\_sales as q2 on q1.month = q2.month + 1 ;

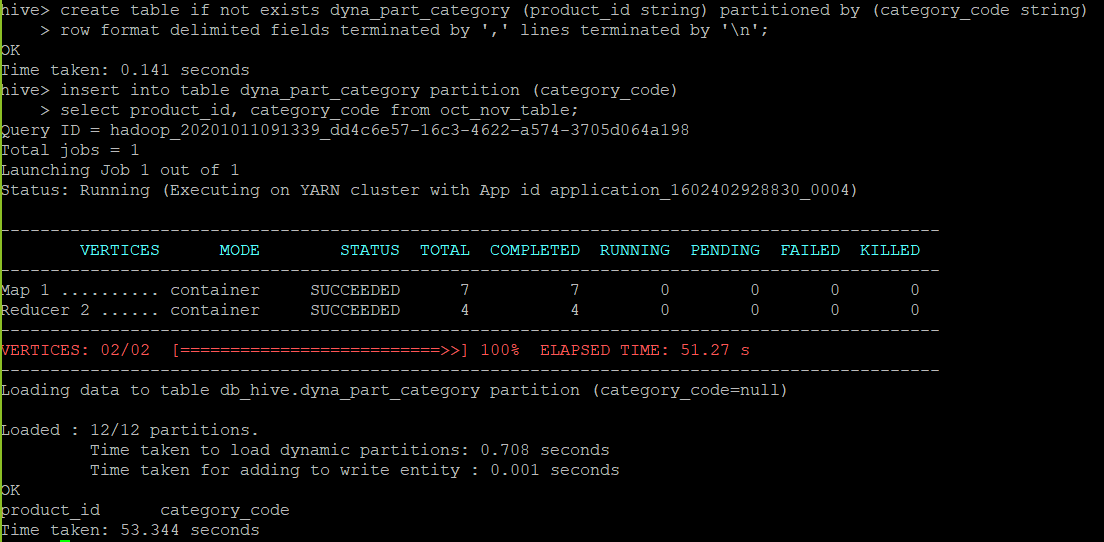


Insights:

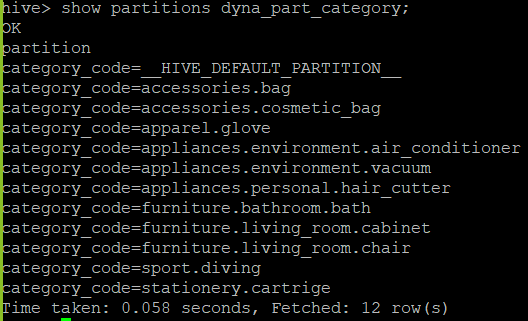
change in revenue generated due to purchases from October to November is 319478.47

1. **Create table “dyna\_part\_category” with dynamic partitioning on category\_code to optimize the queries.**

* create table if not exists dyna\_part\_category (product\_id string) partitioned by (category\_code string) row format delimited fields terminated by ',' lines terminated by '\n';
* insert into table dyna\_part\_category partition (category\_code) select product\_id, category\_code from oct\_nov\_table;

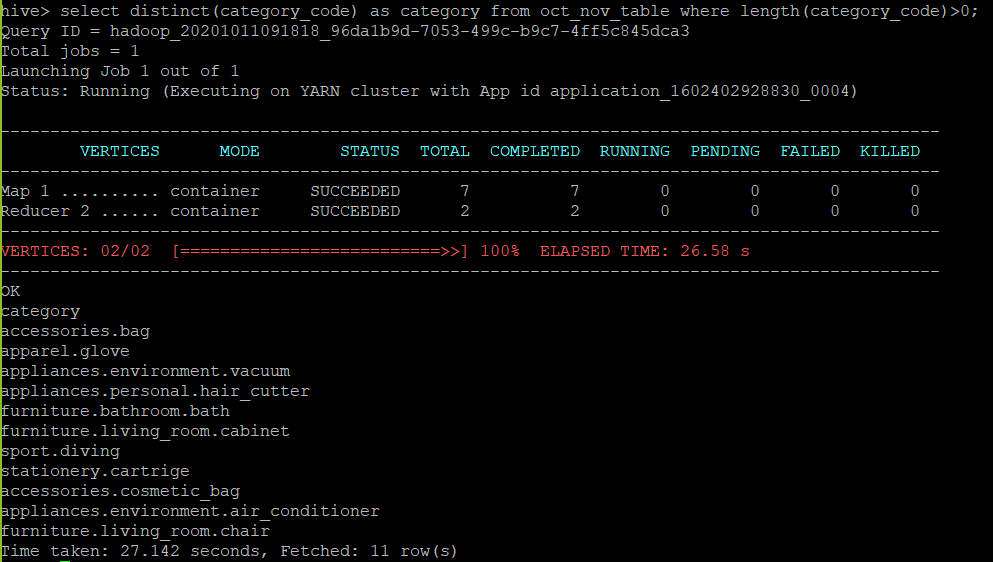


* show partitions dyna\_part\_category;

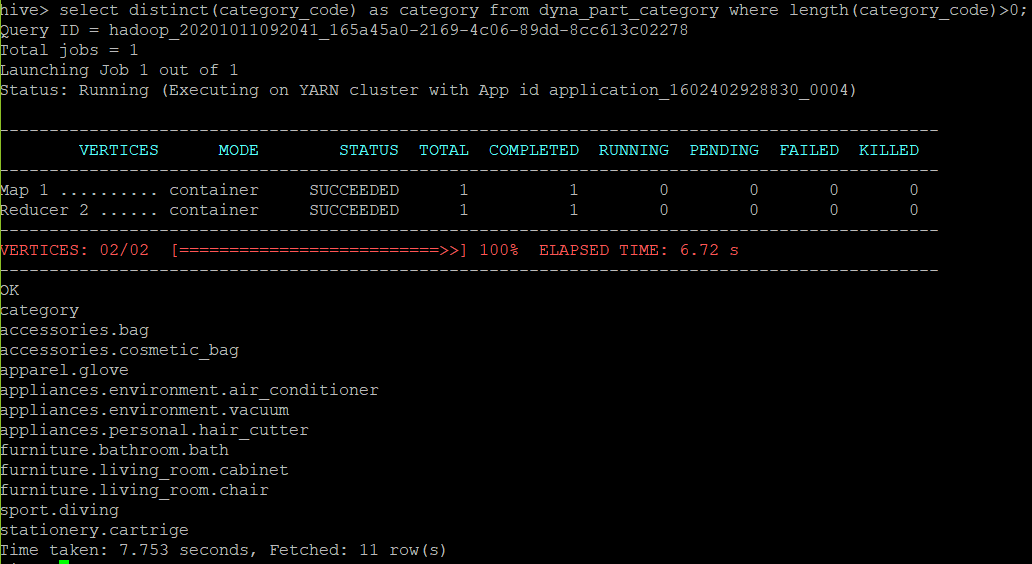


1. **Query 4: Find distinct categories of products. Categories with null category code can be ignored.**

* **Without optimization:**
  + select distinct(category\_code) as category from oct\_nov\_table where length(category\_code)>0;



* **With optimization:**
  + select distinct(category\_code) as category from dyna\_part\_category where length(category\_code)>0;



Insights:

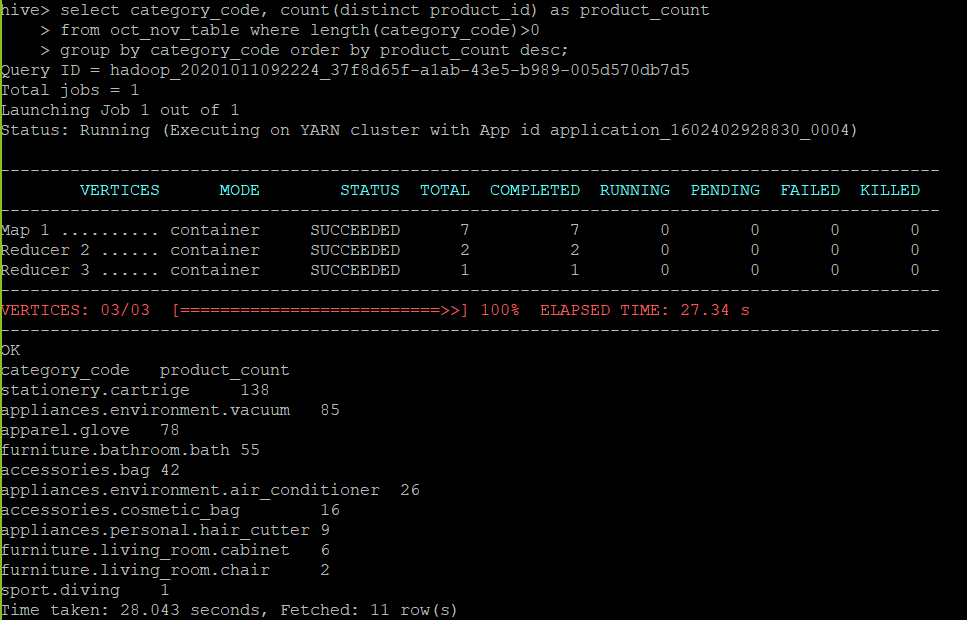
List of different product categories shown in above result.

1. **Query 5: Find the total number of products available under each category.**

* **Without optimization:**
  + select category\_code, count(distinct product\_id) as product\_count

from oct\_nov\_table where length(category\_code)>0

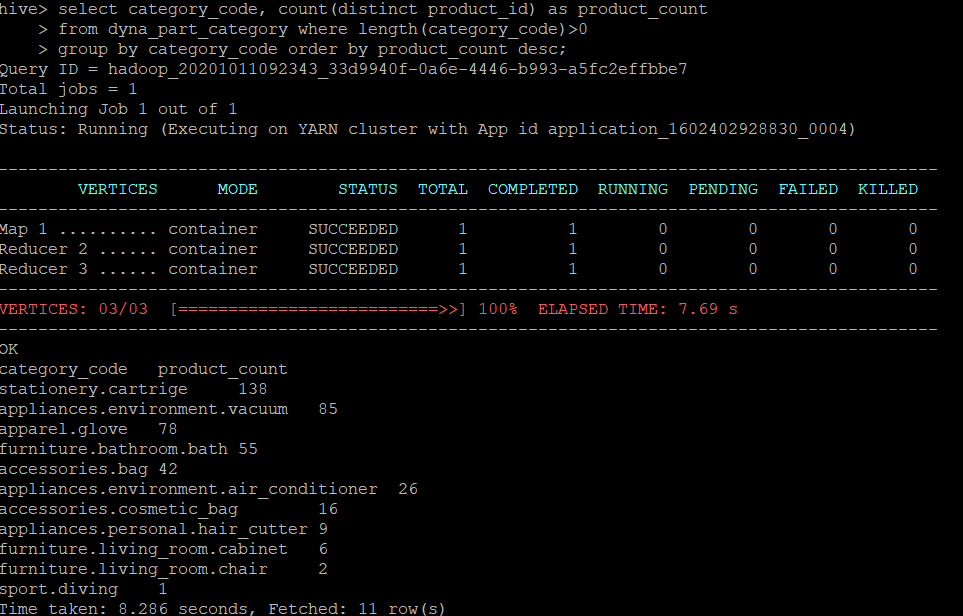
group by category\_code order by product\_count desc;



* **With optimization:**
  + select category\_code, count(distinct product\_id) as product\_count

from dyna\_part\_category where length(category\_code)>0

group by category\_code order by product\_count desc;



Insights:

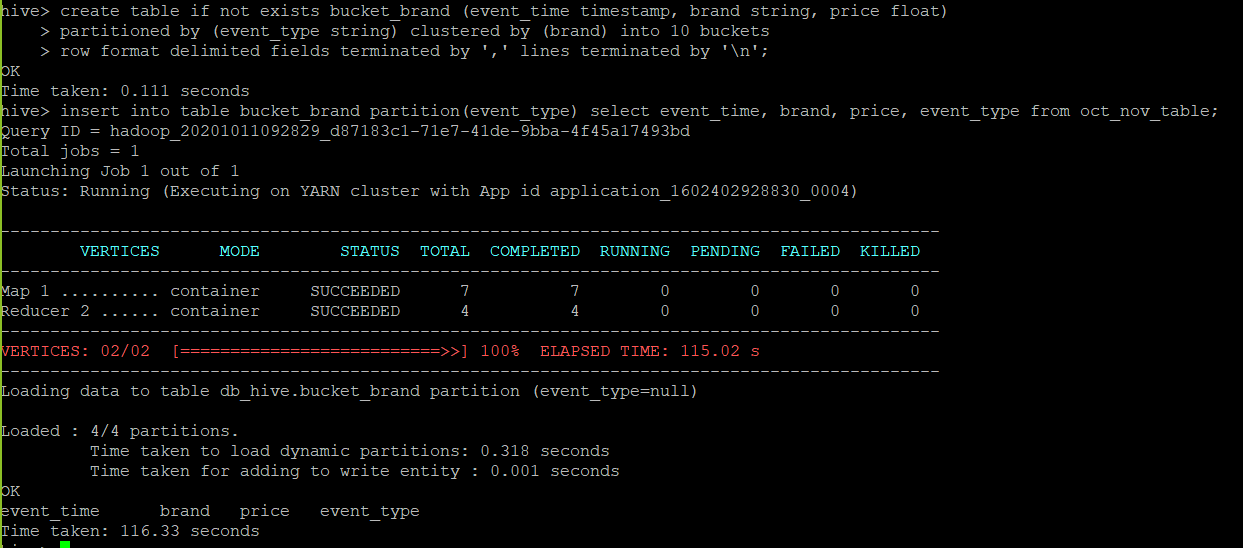
Total number of products available under each category is shown in above result.

1. **Create table “bucket\_brand” with dynamic partitioning on event\_type and bucketing on brand to optimize the queries.**

* create table if not exists bucket\_brand (event\_time timestamp, brand string, price float) partitioned by (event\_type string) clustered by (brand) into 10 buckets

row format delimited fields terminated by ',' lines terminated by '\n';

* insert into table bucket\_brand partition(event\_type) select event\_time, brand, price, event\_type from oct\_nov\_table;

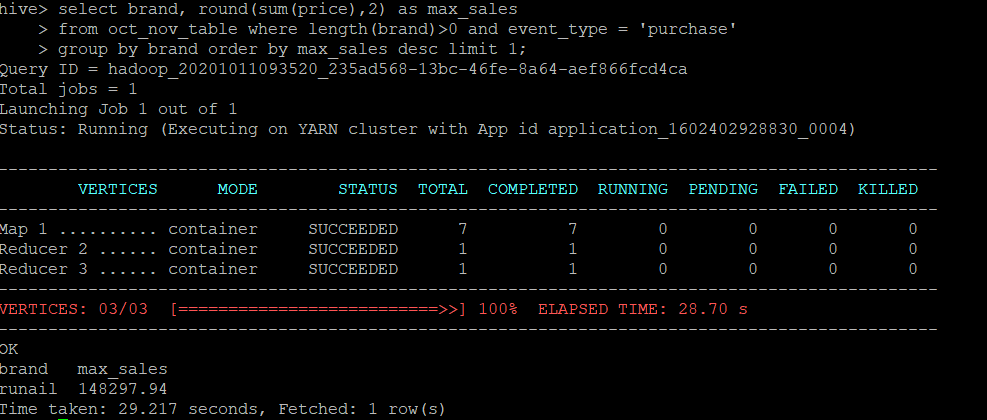


1. **Query 6: Which brand had the maximum sales in October and November combined.**

* **Without optimization:**
  + select brand, round(sum(price),2) as max\_sales

from oct\_nov\_table where length(brand)>0 and event\_type = 'purchase'

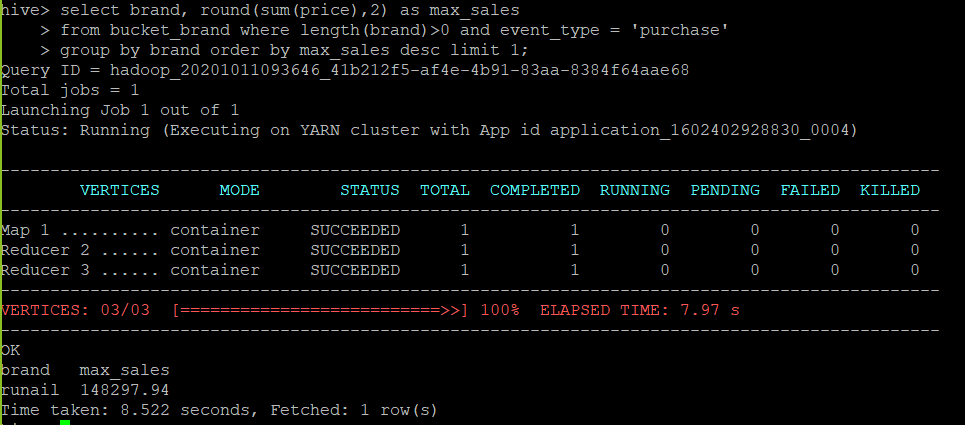
group by brand order by max\_sales desc limit 1;



* **With optimization:**
  + select brand, round(sum(price),2) as max\_sales

from bucket\_brand where length(brand)>0 and event\_type = 'purchase'

group by brand order by max\_sales desc limit 1;



Insights:

Runail brand had the maximum sales in October and November combined which is 18297.94.

1. **Query 7: Which brands increased their sales from October to November.**

* **Without optimization:**
  + with com\_data as

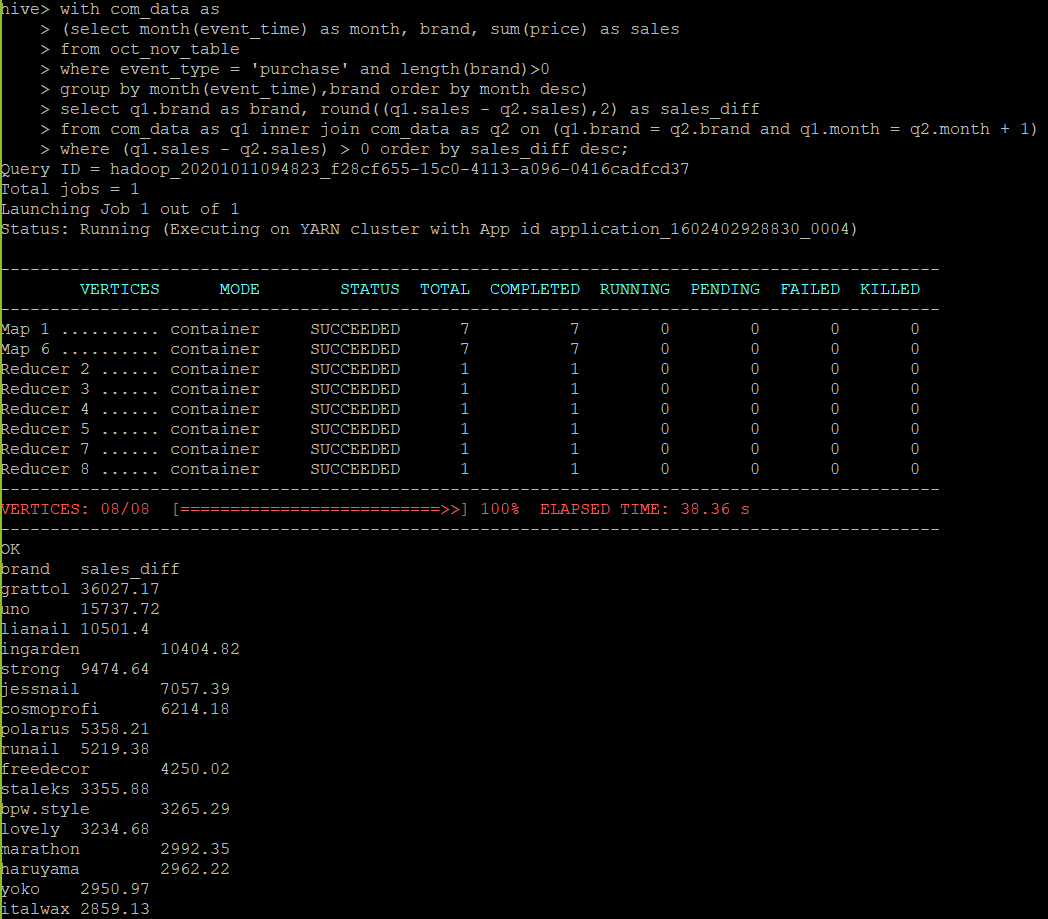
(select month(event\_time) as month, brand, sum(price) as sales

from oct\_nov\_table where event\_type = 'purchase' and length(brand)>0

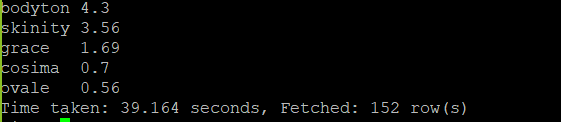
group by month(event\_time),brand order by month desc)

select q1.brand as brand, round((q1.sales - q2.sales),2) as sales\_diff

from com\_data as q1 inner join com\_data as q2 on (q1.brand = q2.brand and q1.month = q2.month + 1) where (q1.sales - q2.sales) > 0 order by sales\_diff desc;



Continued….



* **With optimization:**
  + with com\_data as

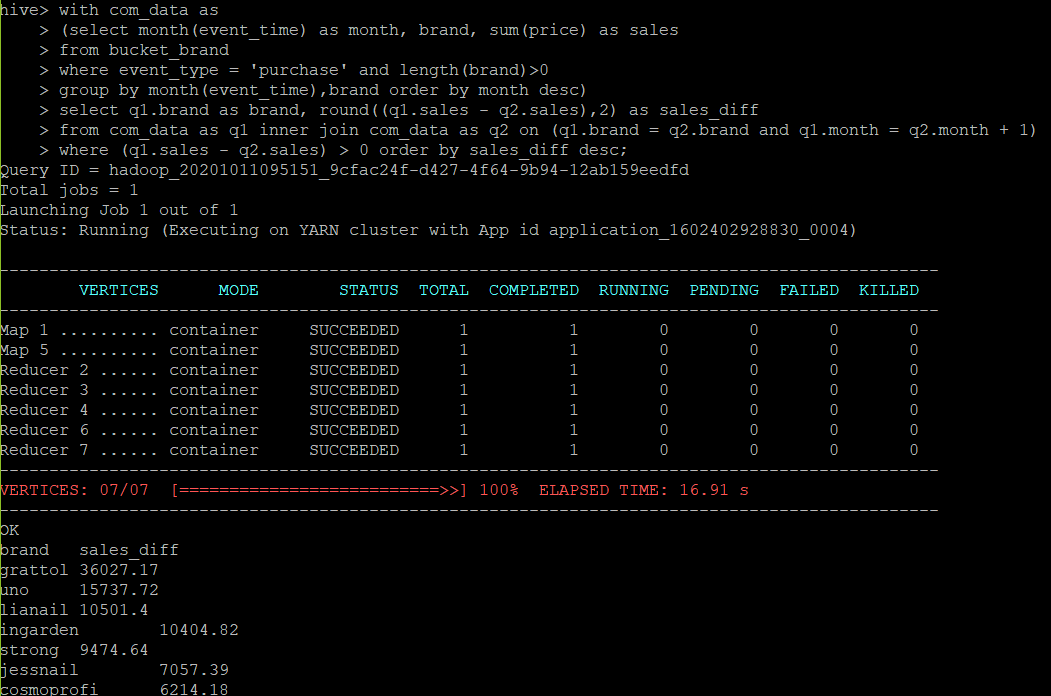
(select month(event\_time) as month, brand, sum(price) as sales

from bucket\_brand where event\_type = 'purchase' and length(brand)>0

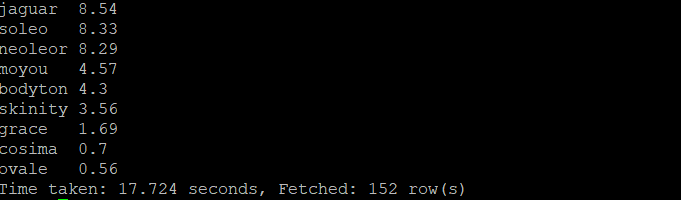
group by month(event\_time),brand order by month desc)

select q1.brand as brand, round((q1.sales - q2.sales),2) as sales\_diff

from com\_data as q1 inner join com\_data as q2 on (q1.brand = q2.brand and q1.month = q2.month + 1) where (q1.sales - q2.sales) > 0 order by sales\_diff desc;



Continued….



Insights:

Approximately 152 brands increased their sales from October to November

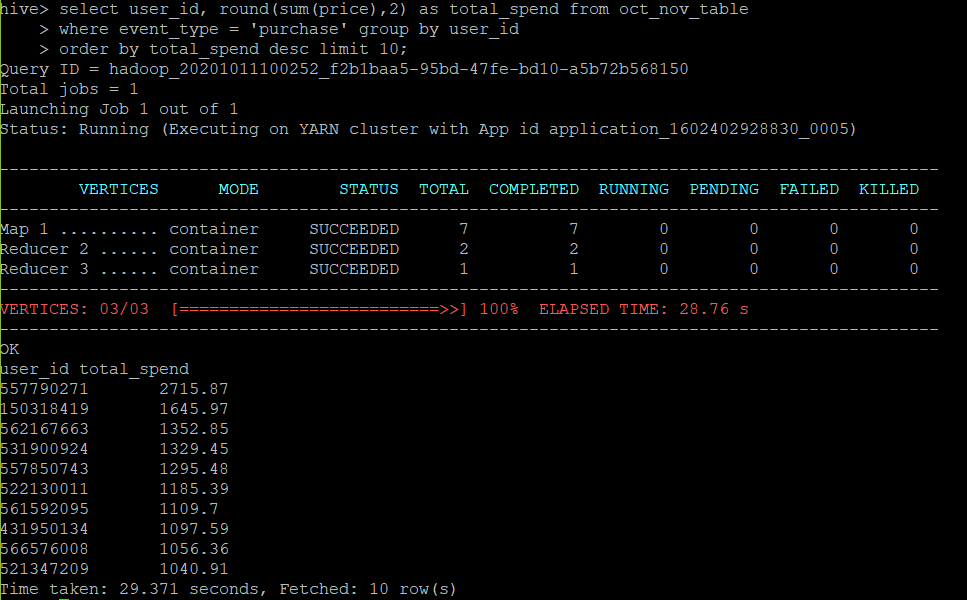


1. **Query 8: Your company wants to reward the top 10 users of its website with a Golden Customer plan. Write a query to generate a list of top 10 users who spend the most.**

* **Without optimization:**
  + select user\_id, round(sum(price),2) as total\_spend from oct\_nov\_table

where event\_type = 'purchase' group by user\_id

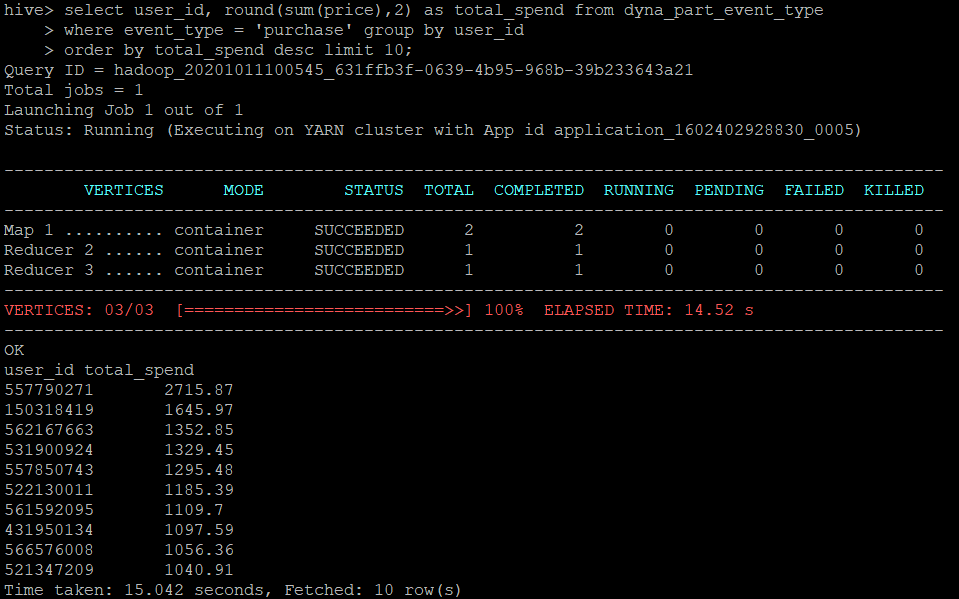
order by total\_spend desc limit 10;



* **With optimization:**
  + select user\_id, round(sum(price),2) as total\_spend from dyna\_part\_event\_type

where event\_type = 'purchase' group by user\_id

order by total\_spend desc limit 10;



Insights:

List of top 10 users who spend the most are shown in above result.

**Overall comparison of execution time for each Query with and without optimization:**

