HIVE CASE STUDY

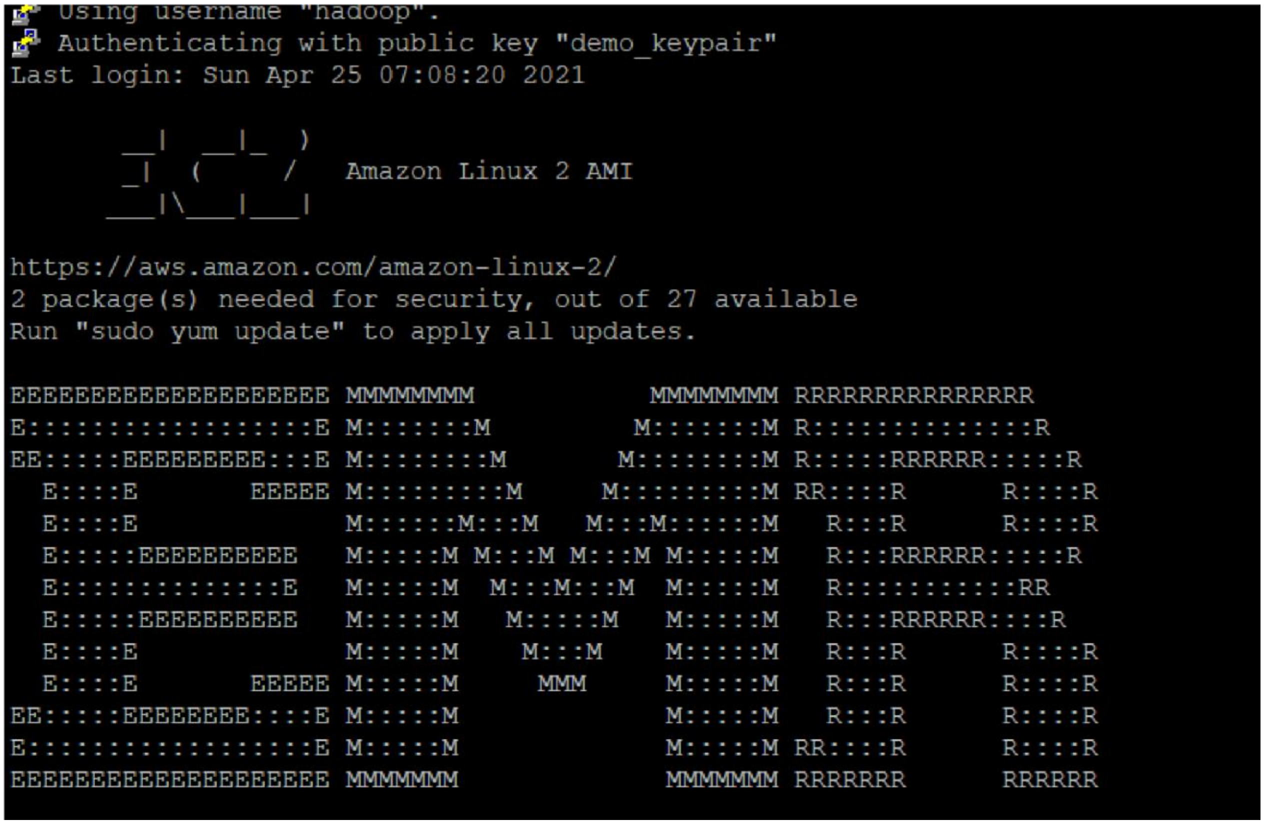
Click stream data analysis

Submitted by: Dyutimaya Das

Steps:

✞ Starting an EMR Cluster

- Select region as N. Virginia(us-east-1c).
- Create a key pair
- Go to AWS account and Create an EMR cluster with 1 master node and 1 core node having m4. large instance type.
  - ✦ Select the same key pair as created before.
- Enable SSH by editing an inbound rule and add SSH as port 22 to the rule.
- Open the putty terminal add the IP address of hostname and Put the .PPK file in the Auth section



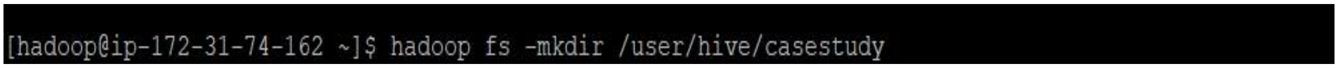
```
Using username "hadoop".
Authenticating with public key "demo_keypair"
Last login: Sun Apr 25 07:08:20 2021

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
2 package(s) needed for security, out of 27 available
Run "sudo yum update" to apply all updates.

EEEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR
```

✞ Hadoop Commands Using Putty Terminal

We have clickstream data from 2 months in form of 2 .csv files. Thus, we will create a directory in HDFS first.

Step 1: Create a directory in HDFS in to which we can copy the dataset which is present in the S3 bucket. By using the command

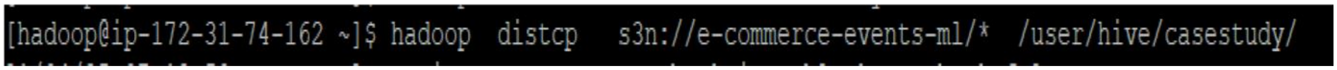
hadoop fs -mkdir /user/hive/casestudy



```
[hadoop@ip-172-31-74-162 ~]$ hadoop fs -mkdir /user/hive/casestudy
```

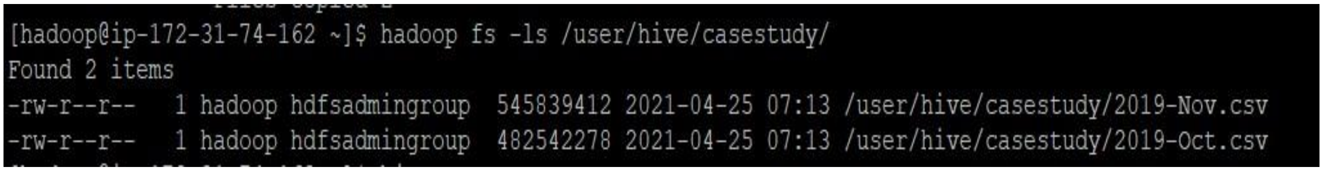
Step 2: Copy the data present in S3 bucket to the HDFS by using this command

hadoop distcp 's3://e-commerce-events-ml/ /user/hive/casestudy/*



```
[hadoop@ip-172-31-74-162 ~]$ hadoop  distcp   s3n://e-commerce-events-ml/*  /user/hive/casestudy/
```

Step 3: Check whether the data set has been moved to HDFS or not.

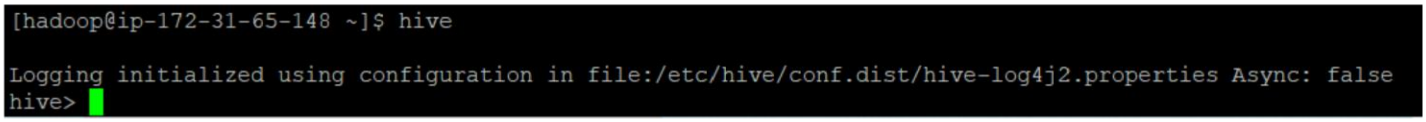


```
[hadoop@ip-172-31-74-162 ~]$ hadoop fs -ls /user/hive/casestudy/
Found 2 items
-rw-r--r--   1 hadoop hdfsadmingroup  545839412 2021-04-25 07:13 /user/hive/casestudy/2019-Nov.csv
-rw-r--r--   1 hadoop hdfsadmingroup  482542278 2021-04-25 07:13 /user/hive/casestudy/2019-Oct.csv
```

✞ Launching Hive Console

Step 1: Launch the Hive CLI using the command

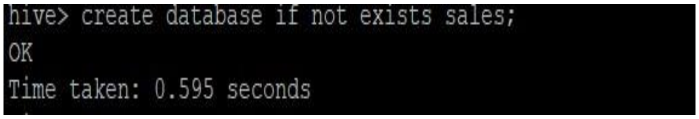
Hive



```
[hadoop@ip-172-31-65-148 ~]$ hive

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

Step 2: Create a data base for storing our tables metadata by using this command

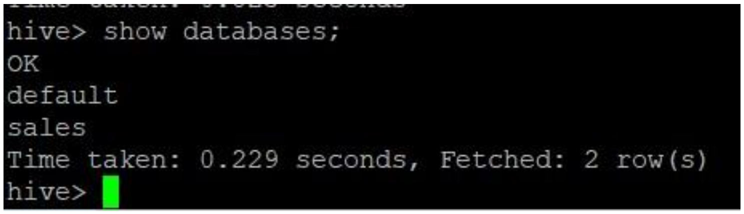
create database if not exists sales;



```
hive> create database if not exists sales;
OK
Time taken: 0.595 seconds
```

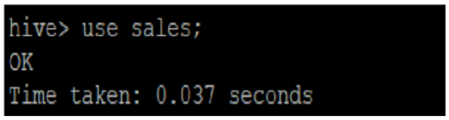
Step 3: Check the created database by using the command

Show databases;



```
hive> show databases;
OK
default
sales
Time taken: 0.229 seconds, Fetched: 2 row(s)
hive>
```

Salles database has been created perfectly.
Type *use sales;* to start working on this database.

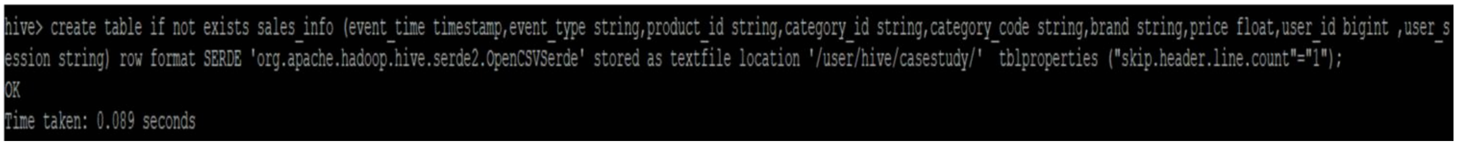


```
hive> use sales;
OK
Time taken: 0.037 seconds
```

step 4: Create a table to load all the data

- Create an Internal table here. We don't need to run any other applications on the top of this table so its better to create an internal table rather than an external table.
- Here we used OpenCSVSerde library to read .csv file
- We don't want to contain first line of csv file into the table because it contains the header file so we skip the line count as 1.

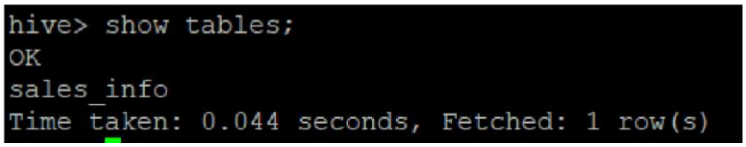
CREATE TABLE IF NOT EXISTS sales_info (event_time TIMESTAMP, event_type STRING, product_id STRING, category_id STRING, category_code STRING, brand STRING, price FLOAT, user_id BIGINT, user_session STRIN) ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde' STORED AS TEXTFILE LOCATION '/user/hive/casestudy/' TBLPROPERTIES ("skip.header.line.count"="1");



```
hive> create table if not exists sales_info (event_time timestamp,event_type string,product_id string,category_id string,category_code string,brand string,price float,user_id bigint ,user_s
ession string) row format SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde' stored as textfile location '/user/hive/casestudy/'  tblproperties ("skip.header.line.count"="1");
OK
Time taken: 0.089 seconds
```

Step 5: Check the table creation by using command

show tables;

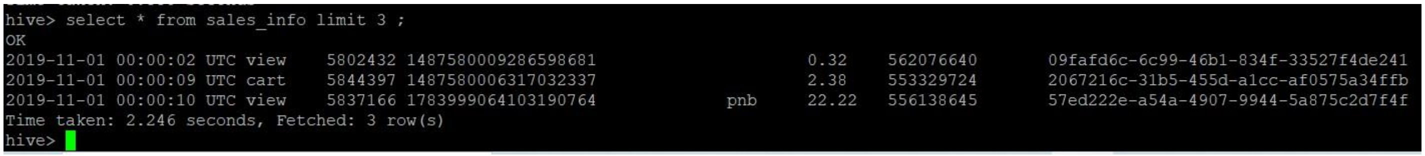


```
hive> show tables;
OK
sales_info
Time taken: 0.044 seconds, Fetched: 1 row(s)
```

The table sales_info is created.

Step 6: Check data in table. *select*

** from sales_info limit 3;*



```
hive> select * from sales_info limit 3 ;
OK
2019-11-01 00:00:02 UTC view    5802432 1487580009286598681                     0.32    562076640       09fafd6c-6c99-46b1-834f-33527f4de241
2019-11-01 00:00:09 UTC cart    5844397 1487580006317032337                     2.38    553329724       2067216c-31b5-455d-a1cc-af0575a34ffb
2019-11-01 00:00:10 UTC view    5837166 1783999064103190764             pnb     22.22   556138645       57ed222e-a54a-4907-9944-5a875c2d7f4f
Time taken: 2.246 seconds, Fetched: 3 row(s)
hive>
```

Took 2.246 s to fetch first 3 records from the non-optimized table sales_info.

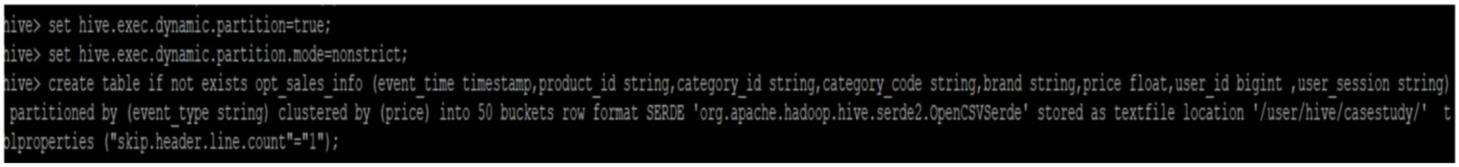
Step 7: Now we shall create our Optimized table.

- For creating Dynamic partitioned table, we need to first set the command as

  Set hive.exec.dynamic.partition=true;

  Set hive.exec.dynamic.partition.mode=nonstrict ;
- We will be using even_type as our field for partitioning. It contains 4 labels and most of our queries will be around purchases and 'purchase' is an event type. So, it's a better choice to use partitioning on event_type.
- We will be using bucketing and clustering simultaneously on price and will divide it into 50 buckets.
- This table is an internal table as all the data of previous table need to insert on the same table.

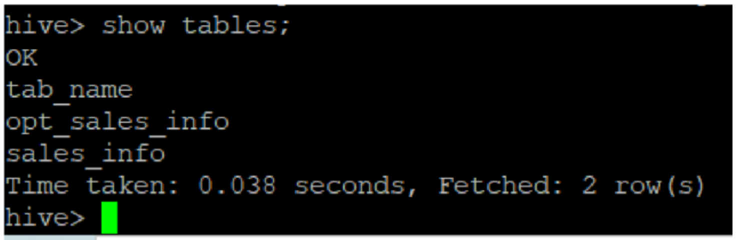
CREATE TABLE IF NOT EXISTS opt_sales_info (event_time TIMESTAMP, product_id STRING, category_id STRING, category_code STRING, brand STRING, price FLOAT, user_id BIGINT, user_session STRING) PARTITIONED BY (event_type string) CLUSTERED BY (price) INTO 50 BUCKETS ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde' STORED AS TEXTFILE LOCATION '/user/hive/casestudy/' TBLPROPERTIES ("skip.header.line.count"="1");



```
ive> set hive.exec.dynamic.partition=true;
ive> set hive.exec.dynamic.partition.mode=nonstrict;
ive> create table if not exists opt_sales_info (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 50 buckets row format SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde' stored as textfile location '/user/hive/casestudy/'  t
blproperties ("skip.header.line.count"="1");
```

Step 8: Now check the table creation by command

Show tables;

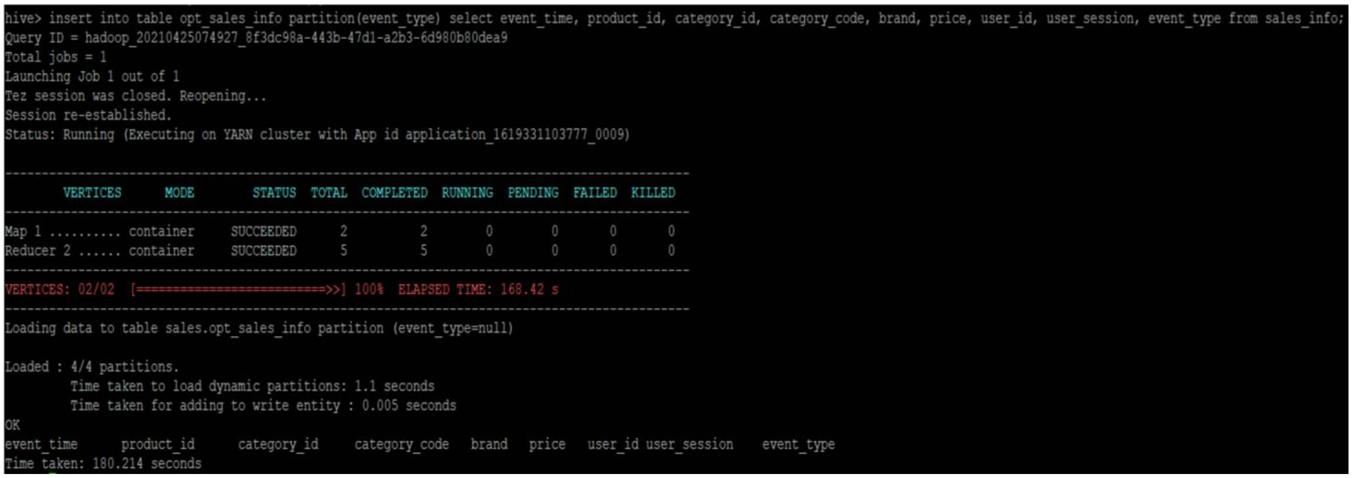


```
hive> show tables;
OK
tab_name
opt_sales_info
sales_info
Time taken: 0.038 seconds, Fetched: 2 row(s)
hive>
```

Both the tables are created base table (sales-info) and optimized table (opt_sales_info)

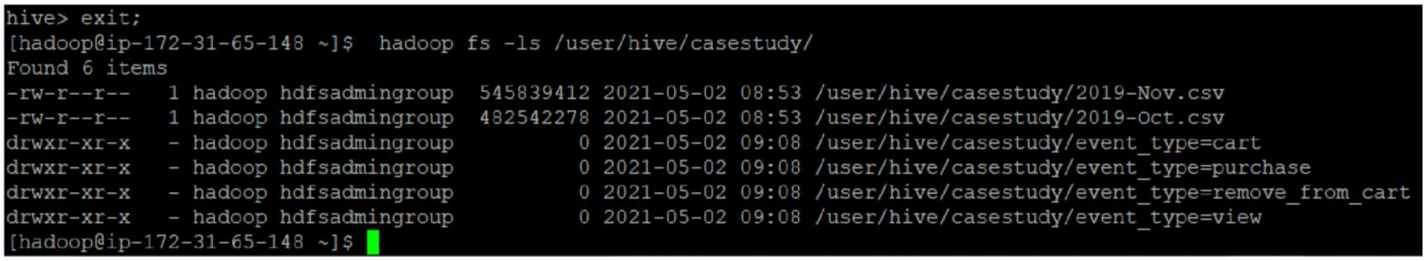
Step 9: Insert data into the optimize table by using the command.

```
INSERT INTO table opt_sales_info
PARTITION (event_type)
SELECT event time,
product_id,  category_id,
category_code,  brand,
price,  user_id,
user_session,
event_type  FROM
sales_info;
```



```
hive> insert into table opt_sales_info partition(event_type) select event_time, product_id, category_id, category_code, brand, price, user_id, user_session, event_type from sales_info;
Query ID = hadoop_20210425074927_8f3dc98a-443b-47d1-a2b3-6d980b80dea9
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_1619331103777_0009)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      2          2        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      5          5        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 168.42 s
----------------------------------------------------------------------------------------------
Loading data to table sales.opt_sales_info partition (event_type=null)

Loaded : 4/4 partitions.
         Time taken to load dynamic partitions: 1.1 seconds
         Time taken for adding to write entity : 0.005 seconds
OK
event_time      product_id      category_id     category_code   brand   price   user_id user_session    event_type
Time taken: 180.214 seconds
```

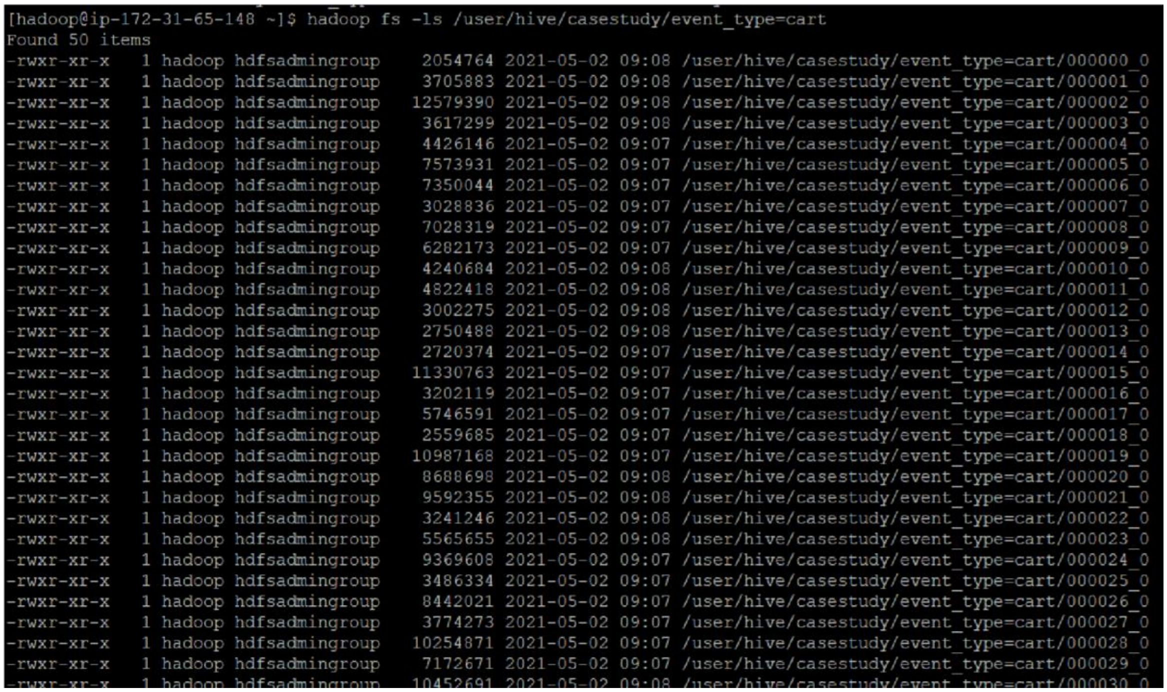
Step 10: Exit from the hive terminal and check the buckets and partitions in hdfs. *hadoop fs -ls '/user/hive/casestudy/'*



```
hive> exit;
[hadoop@ip-172-31-65-148 ~]$  hadoop fs -ls /user/hive/casestudy/
Found 6 items
-rw-r--r--   1 hadoop hdfsadmingroup  545839412 2021-05-02 08:53 /user/hive/casestudy/2019-Nov.csv
-rw-r--r--   1 hadoop hdfsadmingroup  482542278 2021-05-02 08:53 /user/hive/casestudy/2019-Oct.csv
drwxr-xr-x   - hadoop hdfsadmingroup          0 2021-05-02 09:08 /user/hive/casestudy/event_type=cart
drwxr-xr-x   - hadoop hdfsadmingroup          0 2021-05-02 09:08 /user/hive/casestudy/event_type=purchase
drwxr-xr-x   - hadoop hdfsadmingroup          0 2021-05-02 09:08 /user/hive/casestudy/event_type=remove_from_cart
drwxr-xr-x   - hadoop hdfsadmingroup          0 2021-05-02 09:08 /user/hive/casestudy/event_type=view
[hadoop@ip-172-31-65-148 ~]$
```

As we can see 4 partitions have been dynamically created on the field event_type in the form of directories with name format 'event_type=value'

Step 11: Check buckets of any one partition: *hadoop fs -ls '/user/hive/casestudy/event_type=cart'*

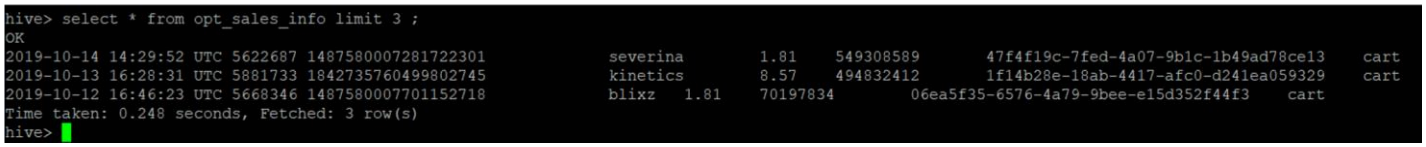


```
[hadoop@ip-172-31-65-148 ~]$ hadoop fs -ls /user/hive/casestudy/event_type=cart
Found 50 items
-rwxr-xr-x   1 hadoop hdfsadmingroup    2054764 2021-05-02 09:08 /user/hive/casestudy/event_type=cart/000000_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    3705883 2021-05-02 09:08 /user/hive/casestudy/event_type=cart/000001_0
-rwxr-xr-x   1 hadoop hdfsadmingroup   12579390 2021-05-02 09:08 /user/hive/casestudy/event_type=cart/000002_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    3617299 2021-05-02 09:08 /user/hive/casestudy/event_type=cart/000003_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    4426146 2021-05-02 09:07 /user/hive/casestudy/event_type=cart/000004_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    7573931 2021-05-02 09:07 /user/hive/casestudy/event_type=cart/000005_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    7350044 2021-05-02 09:07 /user/hive/casestudy/event_type=cart/000006_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    3028836 2021-05-02 09:07 /user/hive/casestudy/event_type=cart/000007_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    7028319 2021-05-02 09:07 /user/hive/casestudy/event_type=cart/000008_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    6282173 2021-05-02 09:07 /user/hive/casestudy/event_type=cart/000009_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    4240684 2021-05-02 09:08 /user/hive/casestudy/event_type=cart/000010_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    4822418 2021-05-02 09:08 /user/hive/casestudy/event_type=cart/000011_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    3002275 2021-05-02 09:08 /user/hive/casestudy/event_type=cart/000012_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    2750488 2021-05-02 09:08 /user/hive/casestudy/event_type=cart/000013_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    2720374 2021-05-02 09:07 /user/hive/casestudy/event_type=cart/000014_0
-rwxr-xr-x   1 hadoop hdfsadmingroup   11330763 2021-05-02 09:07 /user/hive/casestudy/event_type=cart/000015_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    3202119 2021-05-02 09:07 /user/hive/casestudy/event_type=cart/000016_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    5746591 2021-05-02 09:07 /user/hive/casestudy/event_type=cart/000017_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    2559685 2021-05-02 09:07 /user/hive/casestudy/event_type=cart/000018_0
-rwxr-xr-x   1 hadoop hdfsadmingroup   10987168 2021-05-02 09:07 /user/hive/casestudy/event_type=cart/000019_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    8688698 2021-05-02 09:08 /user/hive/casestudy/event_type=cart/000020_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    9592355 2021-05-02 09:08 /user/hive/casestudy/event_type=cart/000021_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    3241246 2021-05-02 09:08 /user/hive/casestudy/event_type=cart/000022_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    5565655 2021-05-02 09:08 /user/hive/casestudy/event_type=cart/000023_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    9369608 2021-05-02 09:07 /user/hive/casestudy/event_type=cart/000024_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    3486334 2021-05-02 09:07 /user/hive/casestudy/event_type=cart/000025_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    8442021 2021-05-02 09:07 /user/hive/casestudy/event_type=cart/000026_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    3774273 2021-05-02 09:07 /user/hive/casestudy/event_type=cart/000027_0
-rwxr-xr-x   1 hadoop hdfsadmingroup   10254871 2021-05-02 09:07 /user/hive/casestudy/event_type=cart/000028_0
-rwxr-xr-x   1 hadoop hdfsadmingroup    7172671 2021-05-02 09:07 /user/hive/casestudy/event_type=cart/000029_0
-rwxr-xr-x   1 hadoop hdfsadmingroup   10452691 2021-05-02 09:08 /user/hive/casestudy/event_type=cart/000030_0
```

50 buckets have been created on the ‘price’ field for each partition.

Step 12: Check data in table:

*SELECT * FROM opt_info_sales LIMIT 3;*



```
hive> select * from opt_sales_info limit 3 ;
OK
2019-10-14 14:29:52 UTC 5622687 1487580007281722301             severina        1.81    549308589       47f4f19c-7fed-4a07-9b1c-1b49ad78ce13    cart
2019-10-13 16:28:31 UTC 5881733 1842735760499802745             kinetics        8.57    494832412       1f14b28e-18ab-4417-afc0-d241ea059329    cart
2019-10-12 16:46:23 UTC 5668346 1487580007701152718             blixz   1.81    70197834        06ea5f35-6576-4a79-9bee-e15d352f44f3    cart
Time taken: 0.248 seconds, Fetched: 3 row(s)
hive> 
```

Took 0.248 s to fetch first 3 records from the non-optimized table sales_info whereas it took 2.246 s to fetch first 3 record from non-optimised table.

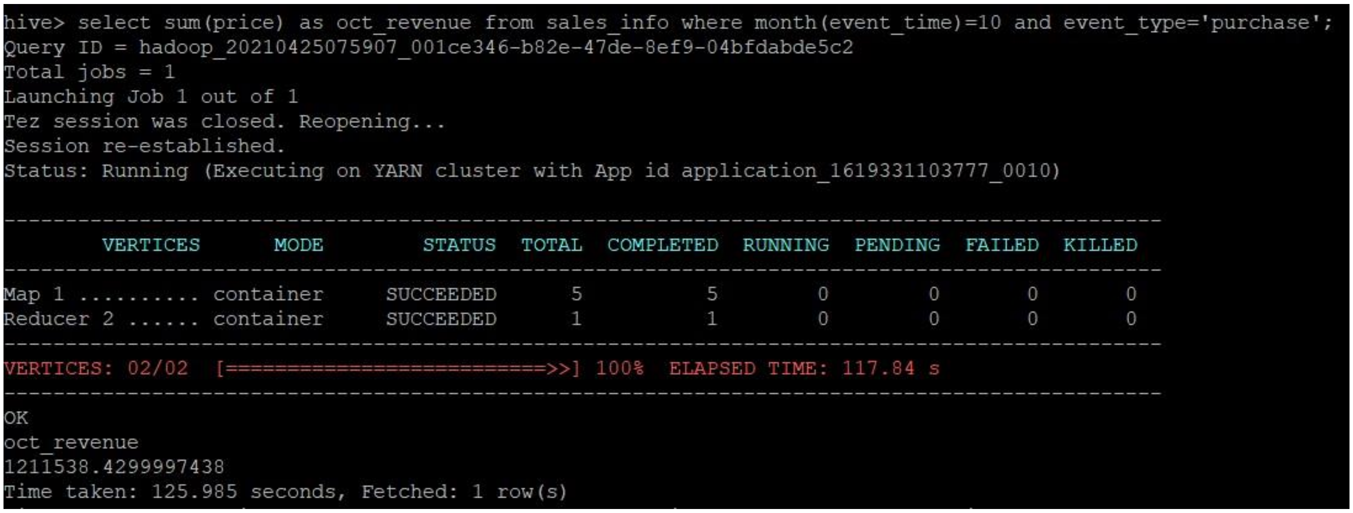
✞ Running Hive queries

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

We need to compare the performance of the non-optimized table and the optimized table.

Query on non-optimized table:

Select sum(price) as oct_revenue from
sales_info
where month(event_time)=10 and event_type=’purchase’; Screenshot:



```
hive> select sum(price) as oct_revenue from sales_info where month(event_time)=10 and event_type='purchase';
Query ID = hadoop_20210425075907_001ce346-b82e-47de-8ef9-04bfdabde5c2
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_1619331103777_0010)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      5          5        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 117.84 s
----------------------------------------------------------------------------------------------
OK
oct_revenue
1211538.4299997438
Time taken: 125.985 seconds, Fetched: 1 row(s)
```

Result:

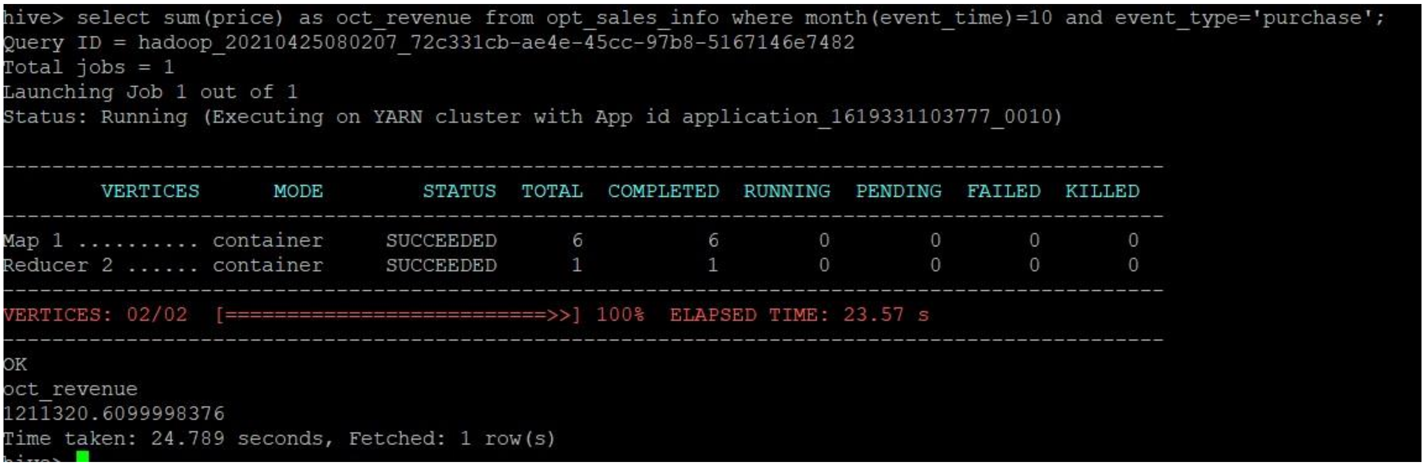
The total revenue generated due to purchases made in October is: 1211538.4299.

Here, the Map Reduce job has to run through the entire table to filter out purchase records from October and hence, the time taken is on the higher side i.e. 125.985 seconds

Query on optimized table:

Select sum(price) as oct_revenue from
opt_sales_info
where month (event_time) =10 and event_type='purchase';

Screenshot:



```
hive> select sum(price) as oct_revenue from opt_sales_info where month(event_time)=10 and event_type='purchase';
Query ID = hadoop_20210425080207_72c331cb-ae4e-45cc-97b8-5167146e7482
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1619331103777_0010)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      6          6        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 23.57 s
----------------------------------------------------------------------------------------------
OK
oct_revenue
1211320.6099998376
Time taken: 24.789 seconds, Fetched: 1 row(s)
```

Result:

Here we can see that the bucketing on ‘price’ field and the partitioning on the ‘event_type’ field helped speed up the querying by almost 5 folds. (24s vs 126s)

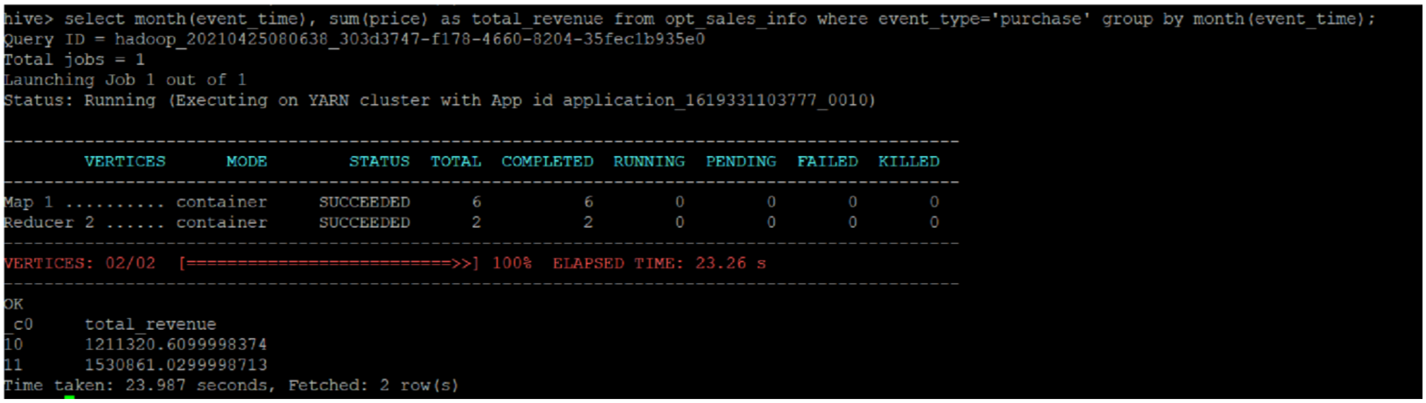
NOTE: As the performance on the optimized table (opt_sales-info) is much better, we will be running rest of our queries on this table itself.

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

Query:

select month(event_time), sum(price) as total_revenue
from opt_sales_info where event_type=’purchase’
group by month(event_time);

Screenshot:



```
hive> select month(event_time), sum(price) as total_revenue from opt_sales_info where event_type='purchase' group by month(event_time);
Query ID = hadoop_20210425080638_303d3747-f178-4660-8204-35fec1b935e0
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1619331103777_0010)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      6          6        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      2          2        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 23.26 s
----------------------------------------------------------------------------------------------
OK
_c0     total_revenue
10      1211320.6099998374
11      1530861.0299998713
Time taken: 23.987 seconds, Fetched: 2 row(s)
```

Results:

Here we can see that the total sum of purchases for October is 1211320.60 and November is 1530861.029.

We have used GROUP BY here to get the sums of both months in a single output. It can also be seen that the partitioning and bucketing have again helped to keep the query execution time just under 25s.

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

Query:

With revenue_table as (

Select sum (case when month(event_time) = 10

then price

else 0 end) as oct_total_sale,

sum (case when month(event_time) =11

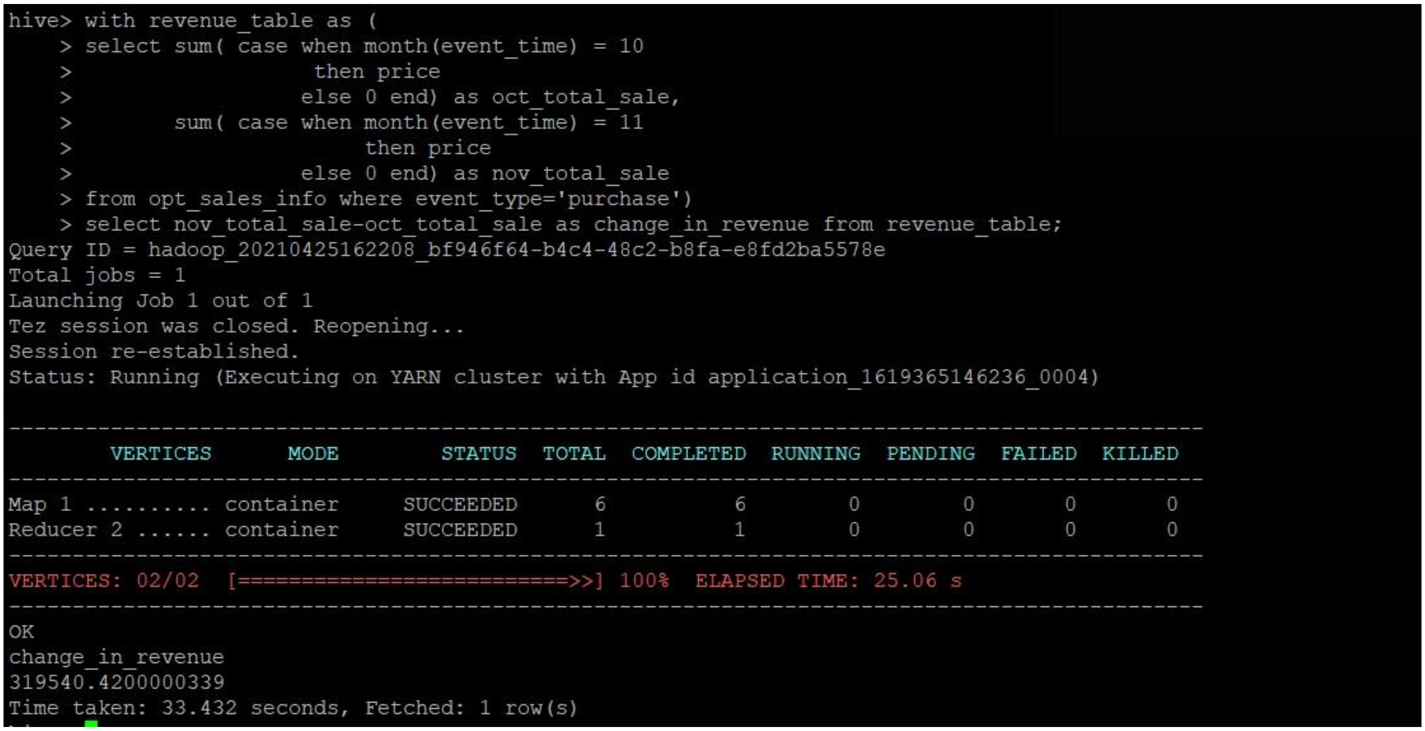
then price

else 0 end) as nov_total_sale

from opt_sales_info where event_type = 'purchase')

select nov_total_sale – oct_total_sale as change_in_revenue from revenue_table;

<u>Screenshot:</u>



```
hive> with revenue_table as (
    > select sum( case when month(event_time) = 10
    >                   then price
    >                  else 0 end) as oct_total_sale,
    >        sum( case when month(event_time) = 11
    >                       then price
    >                  else 0 end) as nov_total_sale
    > from opt_sales_info where event_type='purchase')
    > select nov_total_sale-oct_total_sale as change_in_revenue from revenue_table;
Query ID = hadoop_20210425162208_bf946f64-b4c4-48c2-b8fa-e8fd2ba5578e
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_1619365146236_0004)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      6          6        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 25.06 s
----------------------------------------------------------------------------------------------
OK
change_in_revenue
319540.4200000339
Time taken: 33.432 seconds, Fetched: 1 row(s)
```

<u>Results:</u>

In this query, firstly we have used a common table expression which contains 2 conditional sum variables, one is sum of the prices only when month is October and the other is sum of the prices only when its November. This is implemented using CASE. Once we have these 2 values, we are then going to query this CTE and fetch the difference in the 2 calculated sums. Prices are summed only when 'event_type' is a purchase.

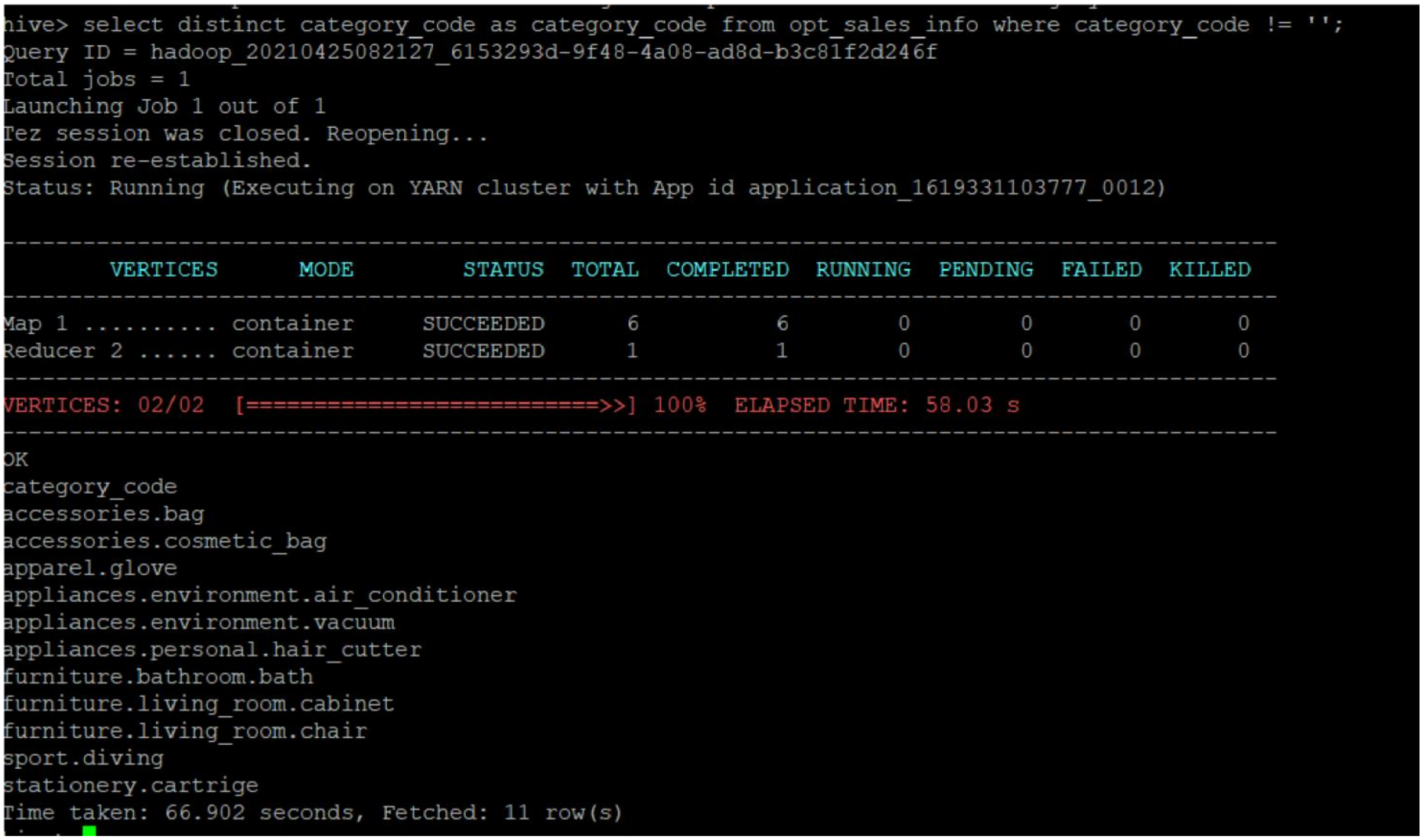
It can be seen that purchases in Nov have increased by an amount of 319540.42 over October.

<u>Query 4: Find the distinct categories of products. Categories with null category can be ignored.</u>

Query by using distinct function:

Select distinct category_code as category_code from opt_sales_info
where category_code != '';

Screenshot:



```
hive> select distinct category_code as category_code from opt_sales_info where category_code != '';
Query ID = hadoop_20210425082127_6153293d-9f48-4a08-ad8d-b3c81f2d246f
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_1619331103777_0012)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      6          6        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 58.03 s
----------------------------------------------------------------------------------------------
OK
category_code
accessories.bag
accessories.cosmetic_bag
apparel.glove
appliances.environment.air_conditioner
appliances.environment.vacuum
appliances.personal.hair_cutter
furniture.bathroom.bath
furniture.living_room.cabinet
furniture.living_room.chair
sport.diving
stationery.cartrige
Time taken: 66.902 seconds, Fetched: 11 row(s)
```

Result:

From the query, it can be seen that we used DISTINCT function to fetch distinct categories of the products. It took 66.9 s.

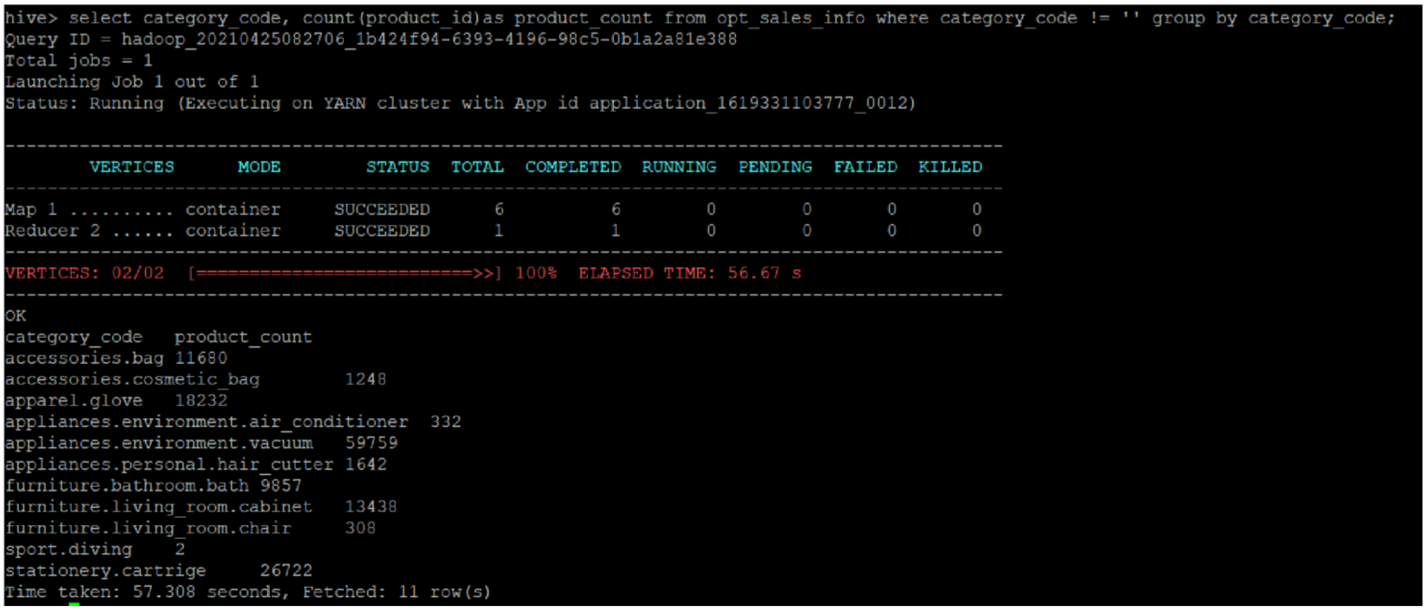
Here we got 11 distinct product categories as result.

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

Query:

```
select category_code, count(product_id) as product_count
from opt_sales_info where category_code != ‘’
group by category_code;
```

Screenshot:



```
hive> select category_code, count(product_id)as product_count from opt_sales_info where category_code != '' group by category_code;
Query ID = hadoop_20210425082706_1b424f94-6393-4196-98c5-0b1a2a81e388
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1619331103777_0012)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      6          6        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 56.67 s
----------------------------------------------------------------------------------------------
OK
category_code   product_count
accessories.bag 11680
accessories.cosmetic_bag        1248
apparel.glove   18232
appliances.environment.air_conditioner  332
appliances.environment.vacuum   59759
appliances.personal.hair_cutter 1642
furniture.bathroom.bath 9857
furniture.living_room.cabinet   13438
furniture.living_room.chair     308
sport.diving    2
stationery.cartrige     26722
Time taken: 57.308 seconds, Fetched: 11 row(s)
```

Result:

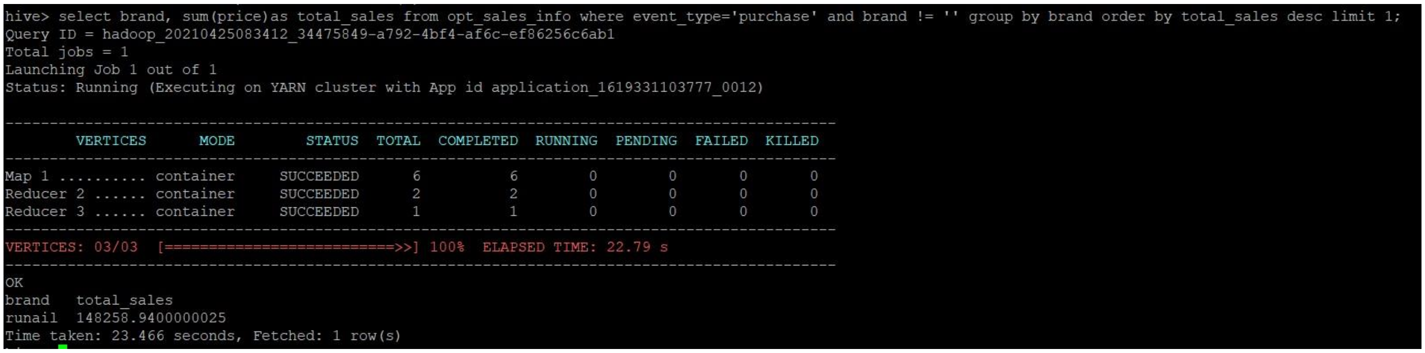
Here in this query, we used the COUNT function for counting the product_ids under each category_code here.

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

Query:

```
select brand, sum (price) as total_sales
from opt_sales_info
where event_type= ‘purchase’ and brand != ‘’
group by brand order by total_sales desc
limit 1;
```

Screenshot:



```
hive> select brand, sum(price)as total_sales from opt_sales_info where event_type='purchase' and brand != '' group by brand order by total_sales desc limit 1;
Query ID = hadoop_20210425083412_34475849-a792-4bf4-af6c-ef86256c6ab1
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1619331103777_0012)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      6          6        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      2          2        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 22.79 s
----------------------------------------------------------------------------------------------
OK
brand   total_sales
runail  148258.9400000025
Time taken: 23.466 seconds, Fetched: 1 row(s)
```

Result:

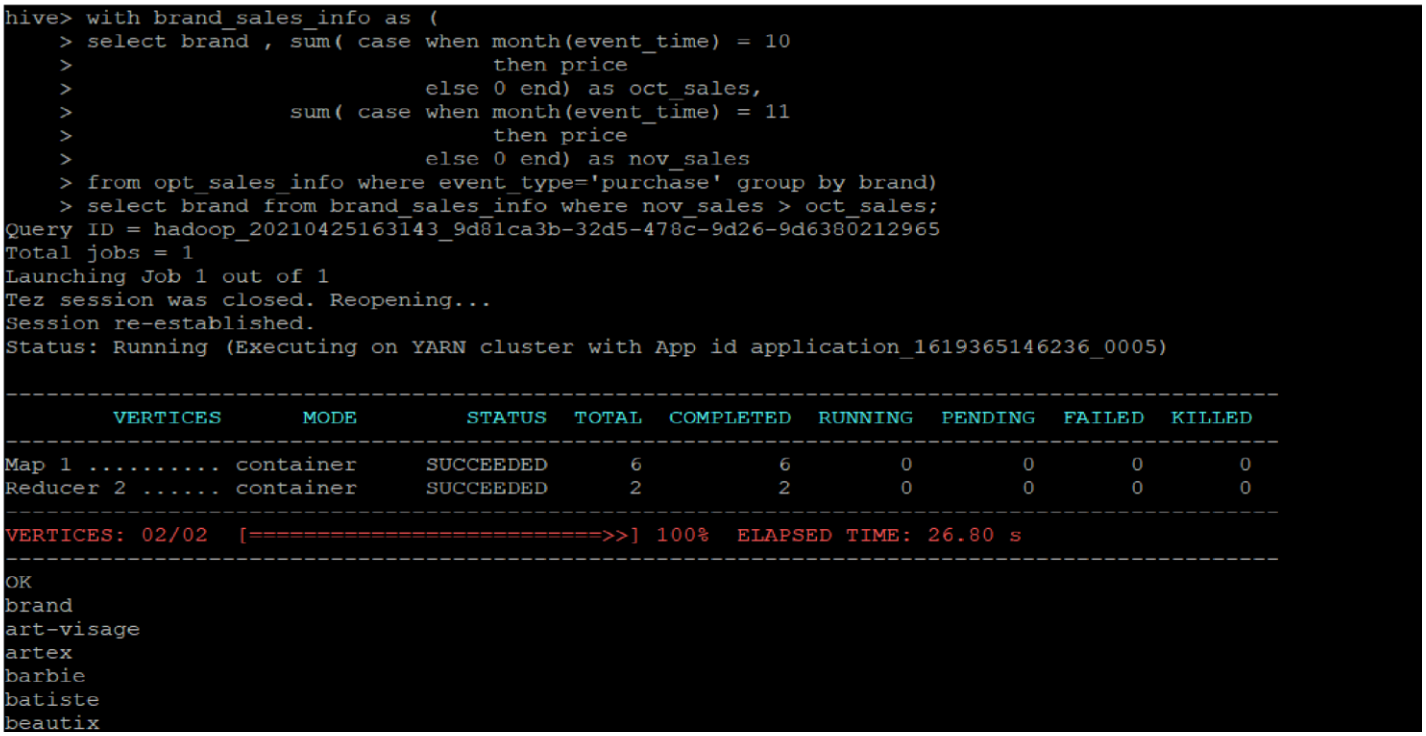
Runail has the most sales in both months combined. Sales value is 148258.94

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

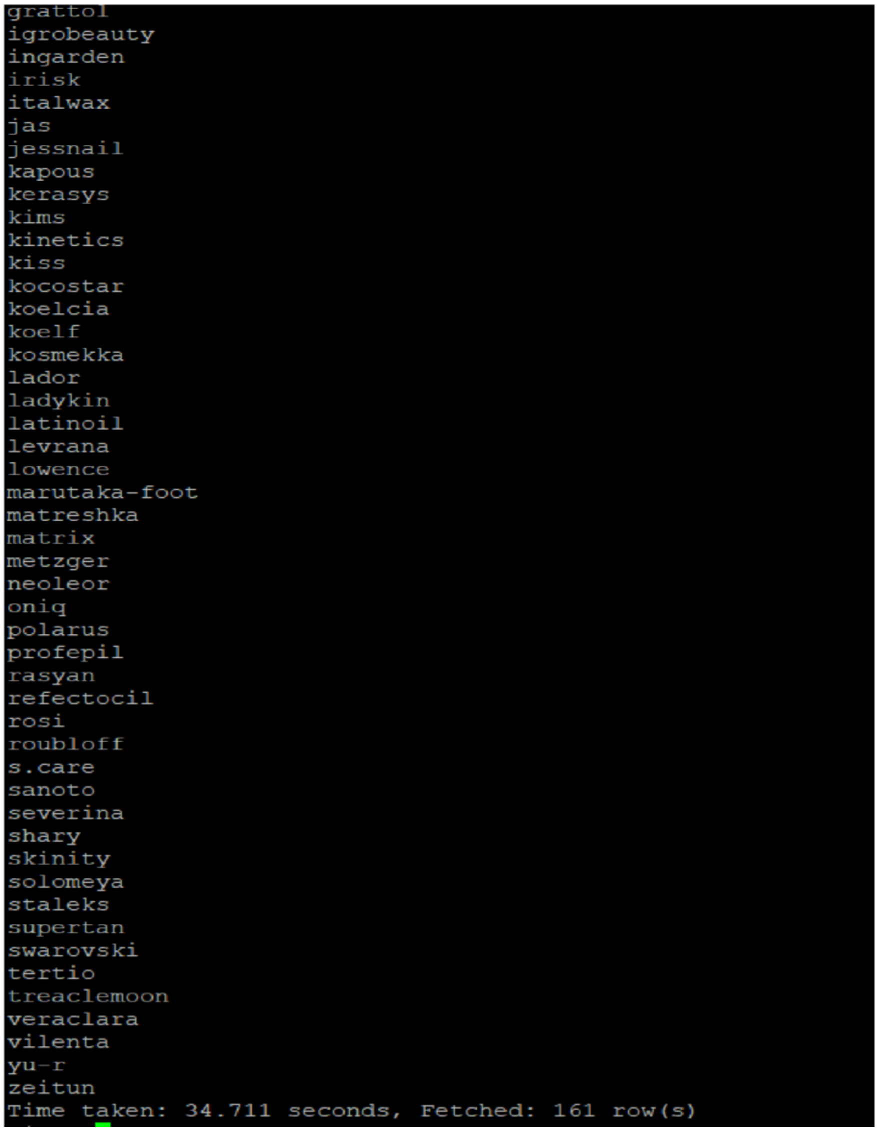
Query:

```
with brand_sales_info 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 opt_sales_info        where event_type =
'purchase'          group by brand)
select brand from brand_sales_info where nov_sales > oct_sales;
```

Screenshot:



```
hive> with brand_sales_info 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 opt_sales_info where event_type='purchase' group by brand)
    > select brand from brand_sales_info where nov_sales > oct_sales;
Query ID = hadoop_20210425163143_9d81ca3b-32d5-478c-9d26-9d6380212965
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_1619365146236_0005)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      6          6        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      2          2        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 26.80 s
----------------------------------------------------------------------------------------------
OK
brand
art-visage
artex
barbie
batiste
beautix
```



```
grattol
igrobeauty
ingarden
irisk
italwax
jas
jessnail
kapous
kerasys
kims
kinetics
kiss
kocostar
koelcia
koelf
kosmekka
lador
ladykin
latinoil
levrana
lowence
marutaka-foot
matreshka
matrix
metzger
neoleor
oniq
polarus
profepil
rasyan
refectocil
rosi
roubloff
s.care
sanoto
severina
shary
skinity
solomeya
staleks
supertan
swarovski
tertio
treaclemoon
veraclara
vilenta
yu-r
zeitun
Time taken: 34.711 seconds, Fetched: 161 row(s)
```

Result:

In this query, firstly we are using a common table expression which contains 2 conditional sum variables, one is the sum of prices only when month is October and the other sum of prices only when its November. This is implemented using CASE. Once we have these 2 values FOR EACH BRAND, we are then going to query this cpmmon table expression and fetch those brands whose NOV_SALES value is greater than their OCT_SALES value. Prices are summed only when ‘event_type’ is a purchase.

Here, we can see that total 161 brands increased their revenue from Oct to Nov

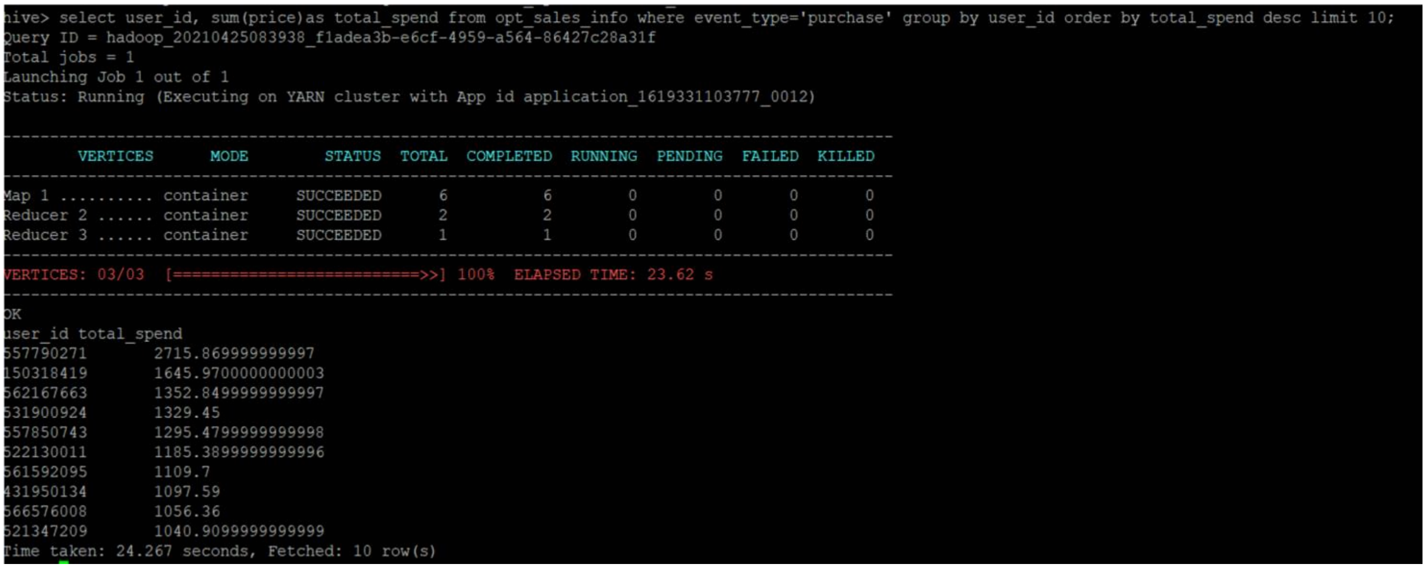
Query 8: Your company wants to reward the top 10 users of its website with a golden customer plan. Write a query to generate a list of top 10 users who spend the most.

Query:

select user_id, sum(price) as total_spend
from opt_sales-info where event_type =
'purchase' group by user_id order by
total_spend desc

limit 10;

Screenshot:



```
hive> select user_id, sum(price)as total_spend from opt_sales_info where event_type='purchase' group by user_id order by total_spend desc limit 10;
Query ID = hadoop_20210425083938_f1adea3b-e6cf-4959-a564-86427c28a31f
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1619331103777_0012)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      6          6        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      2          2        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 23.62 s
----------------------------------------------------------------------------------------------
OK
user_id total_spend
557790271       2715.869999999997
150318419       1645.9700000000003
562167663       1352.8499999999997
531900924       1329.45
557850743       1295.4799999999998
522130011       1185.3899999999996
561592095       1109.7
431950134       1097.59
566576008       1056.36
521347209       1040.9099999999999
Time taken: 24.267 seconds, Fetched: 10 row(s)
```

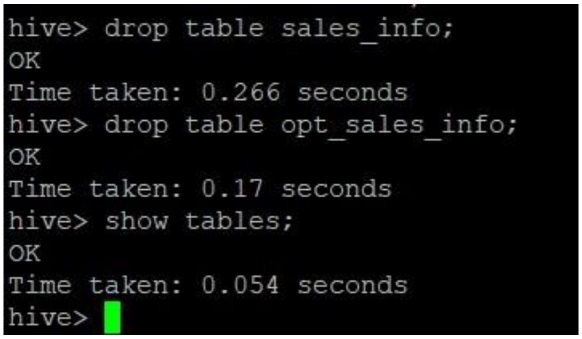
Result:

The Top 10 users are listed as above.

A simple order user by descending, Sum(price) and then doing LIMIT 10 did the job.

✞ Clean-up: Last Steps:

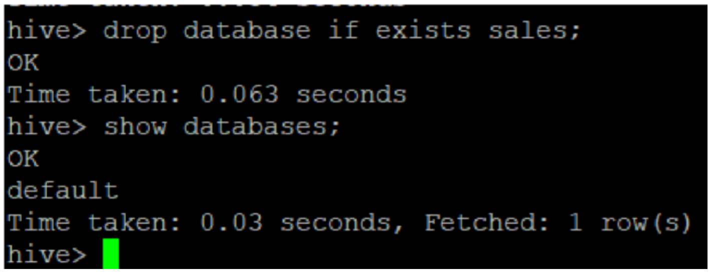
- Drop the tables:



```
hive> drop table sales_info;
OK
Time taken: 0.266 seconds
hive> drop table opt_sales_info;
OK
Time taken: 0.17 seconds
hive> show tables;
OK
Time taken: 0.054 seconds
hive>
```

No more tables left.

- Drop the database:

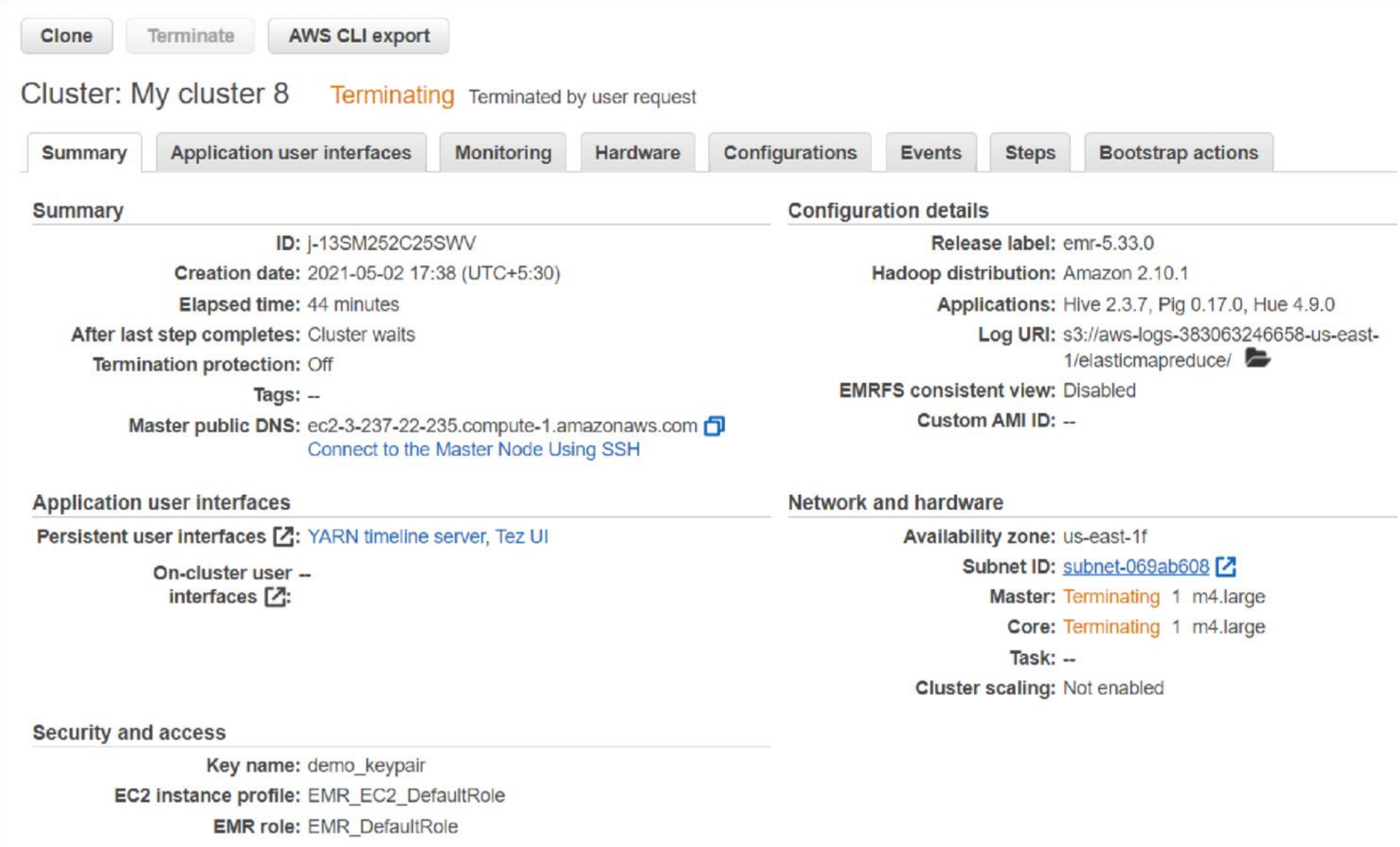


```
hive> drop database if exists sales;
OK
Time taken: 0.063 seconds
hive> show databases;
OK
default
Time taken: 0.03 seconds, Fetched: 1 row(s)
hive>
```

Our database is dropped now.

- Terminate the database:

  Go to the Cluster's page -> Click on Terminate -> Disable the Termination protection -> Terminate the cluster



Cluster: My cluster 8 Terminating Terminated by user request

Summary

ID: j-13SM252C25SWV
Creation date: 2021-05-02 17:38 (UTC+5:30)
Elapsed time: 44 minutes
After last step completes: Cluster waits
Termination protection: Off
Tags: --
Master public DNS: ec2-3-237-22-235.compute-1.amazonaws.com
Connect to the Master Node Using SSH

Configuration details

Release label: emr-5.33.0
Hadoop distribution: Amazon 2.10.1
Applications: Hive 2.3.7, Pig 0.17.0, Hue 4.9.0
Log URI: s3://aws-logs-383063246658-us-east-1/elasticmapreduce/
EMRFS consistent view: Disabled
Custom AMI ID: --

Application user interfaces

Persistent user interfaces: YARN timeline server, Tez UI
On-cluster user interfaces: --

Network and hardware

Availability zone: us-east-1f
Subnet ID: subnet-069ab608
Master: Terminating 1 m4.large
Core: Terminating 1 m4.large
Task: --
Cluster scaling: Not enabled

Security and access

Key name: demo_keypair
EC2 instance profile: EMR_EC2_DefaultRole
EMR role: EMR_DefaultRole