Public Clickstream Data Analysis

Manaswi Kamila

Problem Statement

With online sales gaining popularity, tech companies are exploring ways to improve their sales by analyzing customer behavior and gaining insights about product trends. Furthermore, the websites make it easier for customers to find the products they require without much scavenging.

The objective of this case study is to extract data and gather insights from a real-life public clickstream dataset of a cosmetics store for the months of October and November 2019 which generally data engineers come up within an e-retail company by executing queries using Hive Query Language (HQL).

Clickstream data

Data which is collected by tracking our clicks on websites and searching for patterns within them. E-commerce companies make use of the data to give product recommendations.

Datasets provided

- 2019-Oct.csv- https://e-commerce-events-ml.s3.amazonaws.com/2019-Oct.csv
- 2019-Nov.csv - https://e-commerce-events-ml.s3.amazonaws.com/2019-Nov.csv
- Attribute description – Excel file which contains attribute details

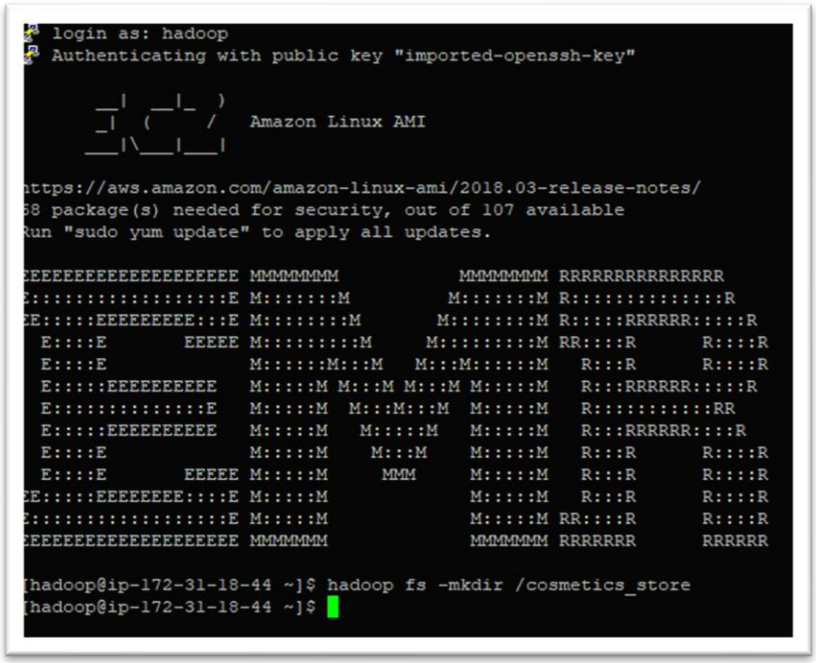
Implementation

Task 1: Importing the data from S3 to HDFS

Launched an EMR 5.29.0 cluster that utilizes the Hive services.

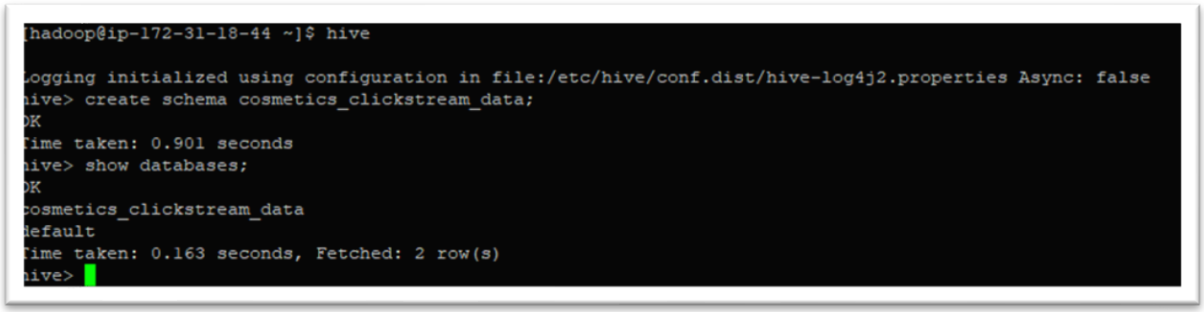
Created a HDFS directory “cosmetics_store” and imported data from public S3 bucket to HDFS directory using distcp command.

hadoop fs -mkdir /cosmetics_store
hadoop distcp s3n://e-commerce-events-ml/* /cosmetics_store
hadoop fs -ls /cosmetics_store

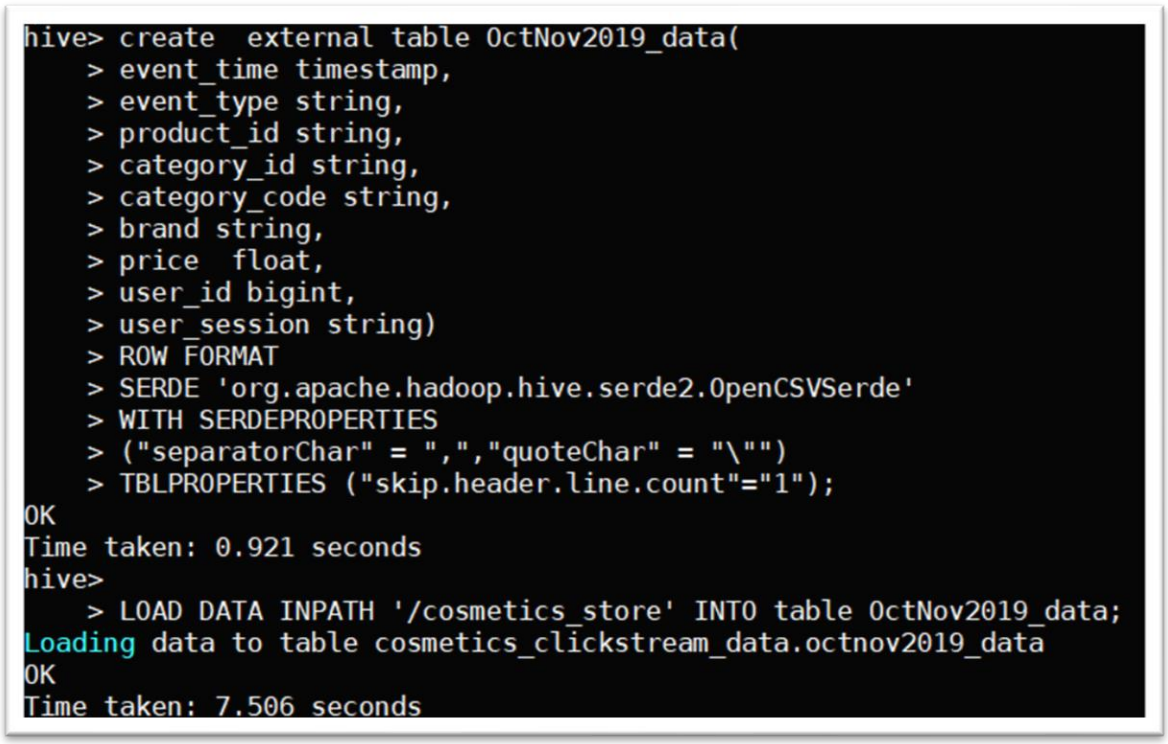


<u>Task 2:</u> Creating database and Hive tables

Logged in to Hive CLI and created database “cosmetics_clickstream_data”



Created a table “OctNov2019_data” using CSV Serde and loaded data from the CSV files on HDFS into the tables.



```
    > describe OctNov2019_data;
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: 0.453 seconds, Fetched: 9 row(s)
```

Created tables “cosmetic_OctNov2019_data”, “part_cosmetic_OctNov2019_data”, “bucket_part_cosmetic_OctNov2019_data”

- Derived a new column, "month", by extracting month from event_time column.
- Handled timestamp data by using SUBSTR function (to_date function caused loss of data – time of transaction was lost)

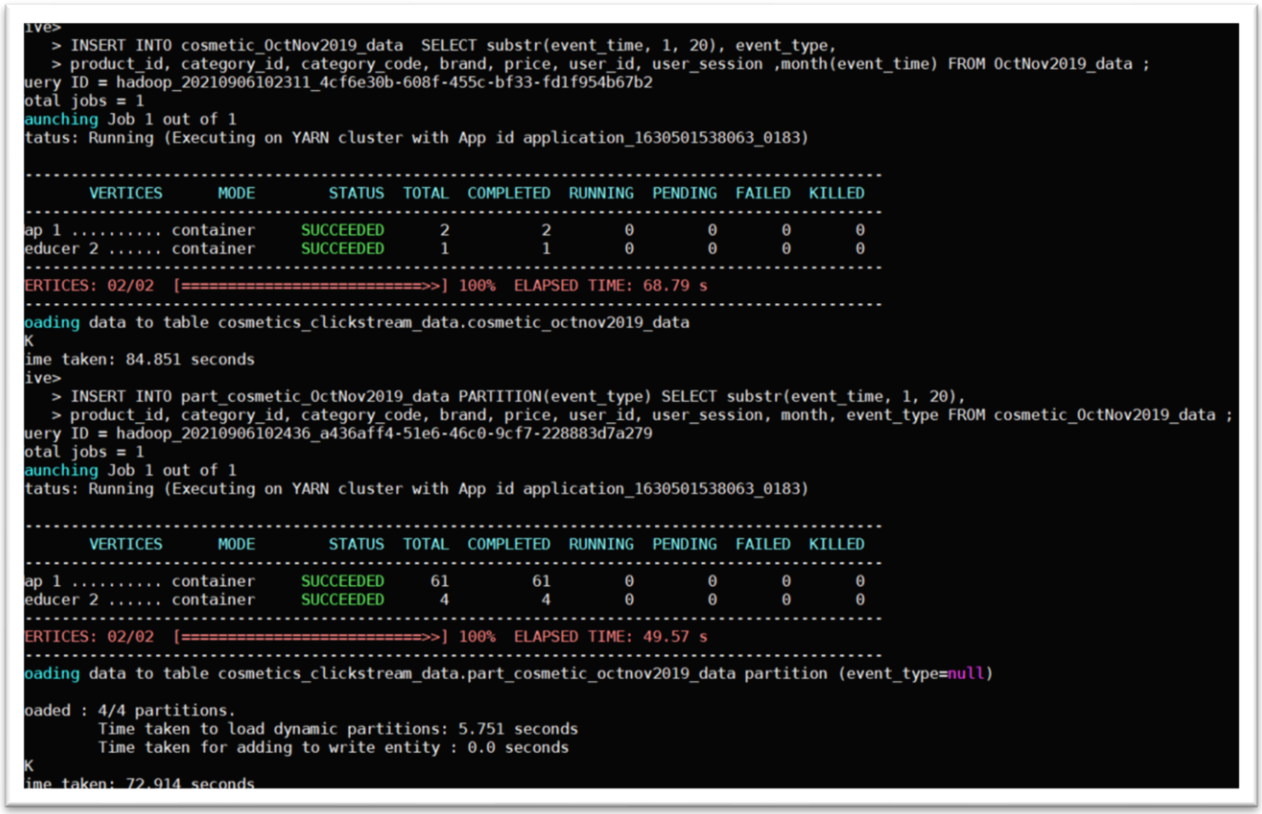
```
hive> create table cosmetic_OctNov2019_data(
    > event_time  timestamp,
    > event_type string,
    > product_id string,
    > category_id string,
    > category_code string,
    > brand string,
    > price  float,
    > user_id bigint,
    > user_session string,
    > month int)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ',' ;
OK
Time taken: 1.25 seconds
```

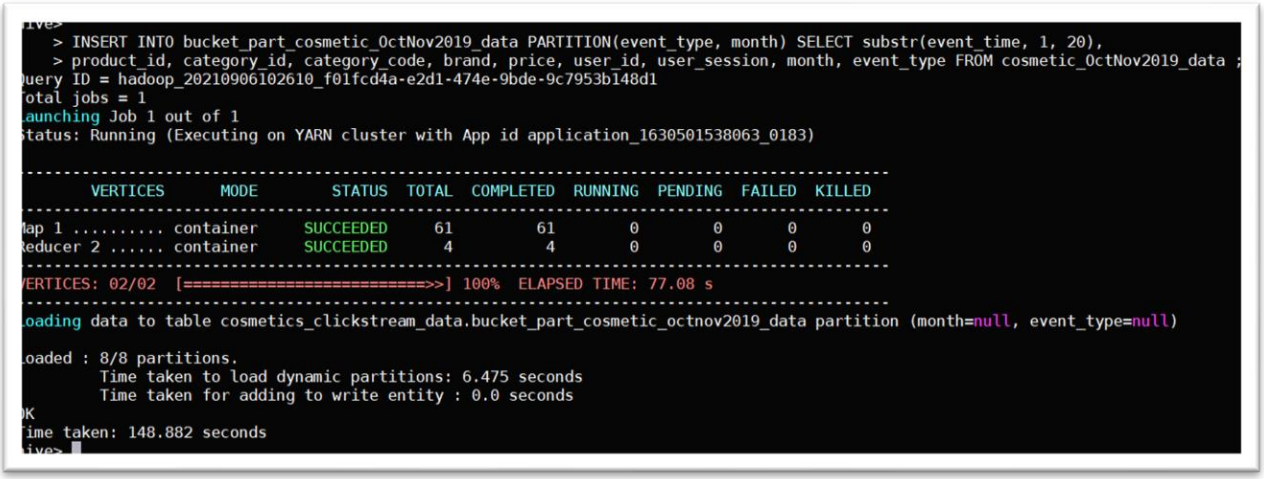
To create dynamic partitioned and bucketed tables, set the below properties in hive CLI.

```
set hive.exec.dynamic.partition=true;
set hive.exec.dynamic.partition.mode=nonstrict;
set hive.enforce.bucketing=true;
```

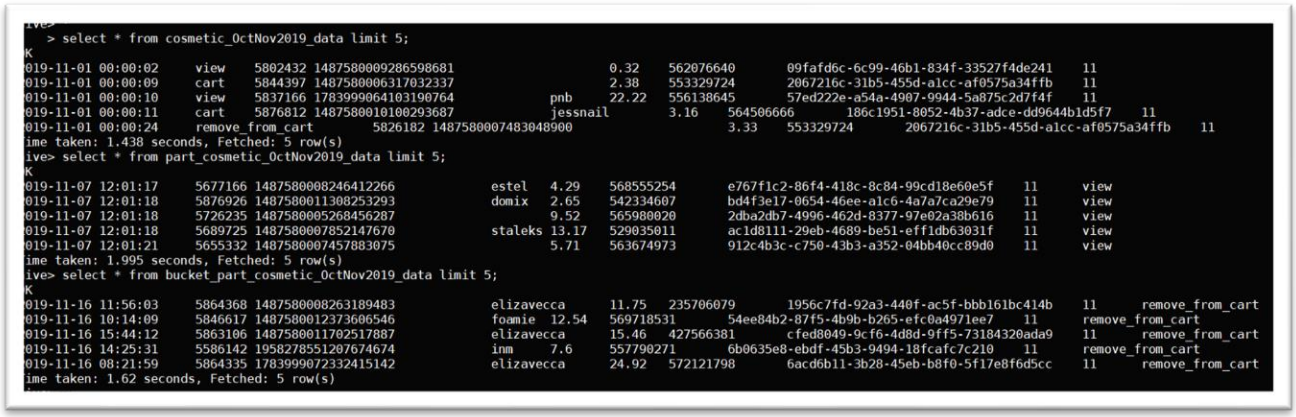
```
hive>
    > set hive.exec.dynamic.partition=true;
hive> set hive.exec.dynamic.partition.mode=nonstrict;
hive> set hive.enforce.bucketing=true;
hive>
    > create table  part_cosmetic_OctNov2019_data(
    > event_time timestamp,
    > product_id string,
    > category_id string,
    > category_code string,
    > brand string,
    > price  float,
    > user_id bigint,
    > user_session string,
    > month int)
    > PARTITIONED BY (event_type string)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ',' ;
OK
Time taken: 4.593 seconds
hive>
    > create external table bucket_part_cosmetic_OctNov2019_data(
    > event_time timestamp,
    > product_id string,
    > category_id string,
    > category_code string,
    > brand string,
    > price  float,
    > user_id bigint,
    > user_session string)
    > PARTITIONED BY (month int,event_type string )
    > CLUSTERED BY (brand) INTO 20 BUCKETS
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ',' ;
OK
Time taken: 1.008 seconds
hive>
```

Inserted records from OctNov2019_data and cosmetic_OctNov2019_data tables.

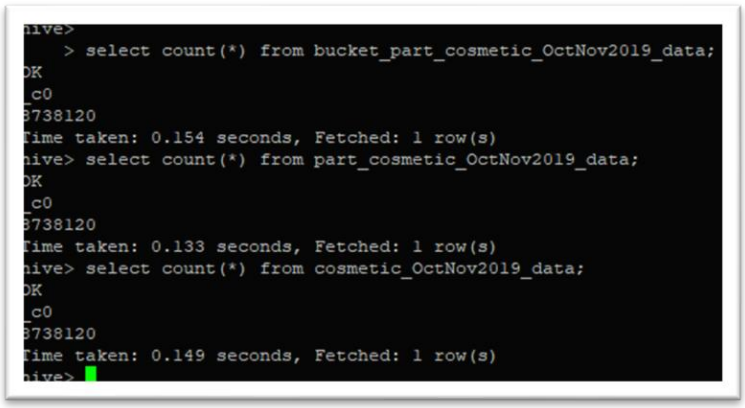




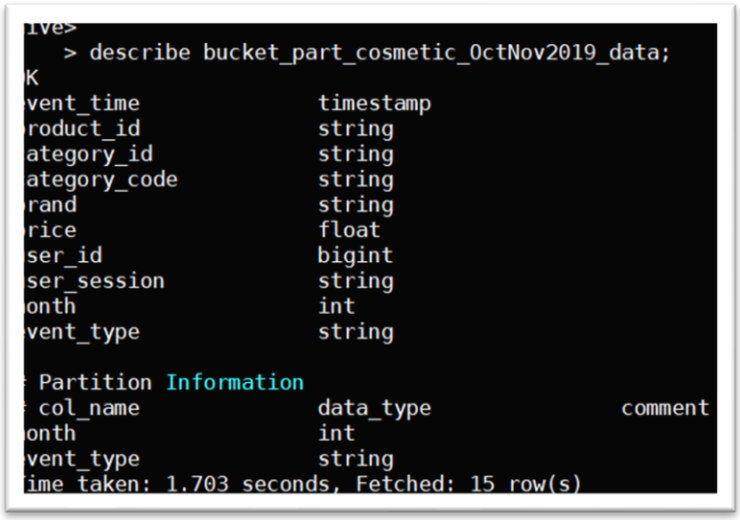
Checked the table data and the data format.



Checked the count of rows in each table.



Datatype of columns

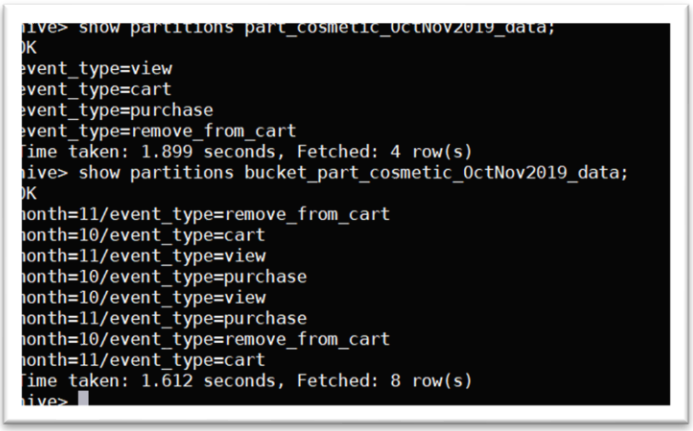


Checked the partitions in the tables created and the HDFS location of the database and the tables.

Partitions and sub-partitions

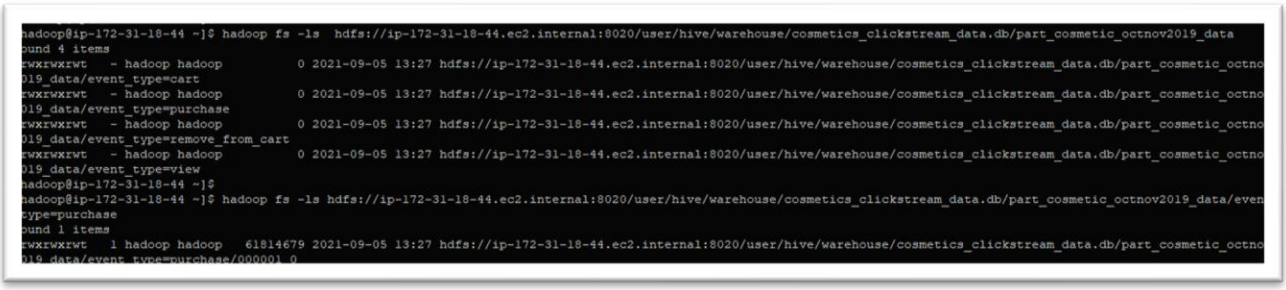
"part_cosmetic_OctNov2019_data" – There are 4 partitions with respect to event_type

"bucket_part_cosmetic_OctNov2019_data" – There are 8 partitions with respect to event_type and month columns.



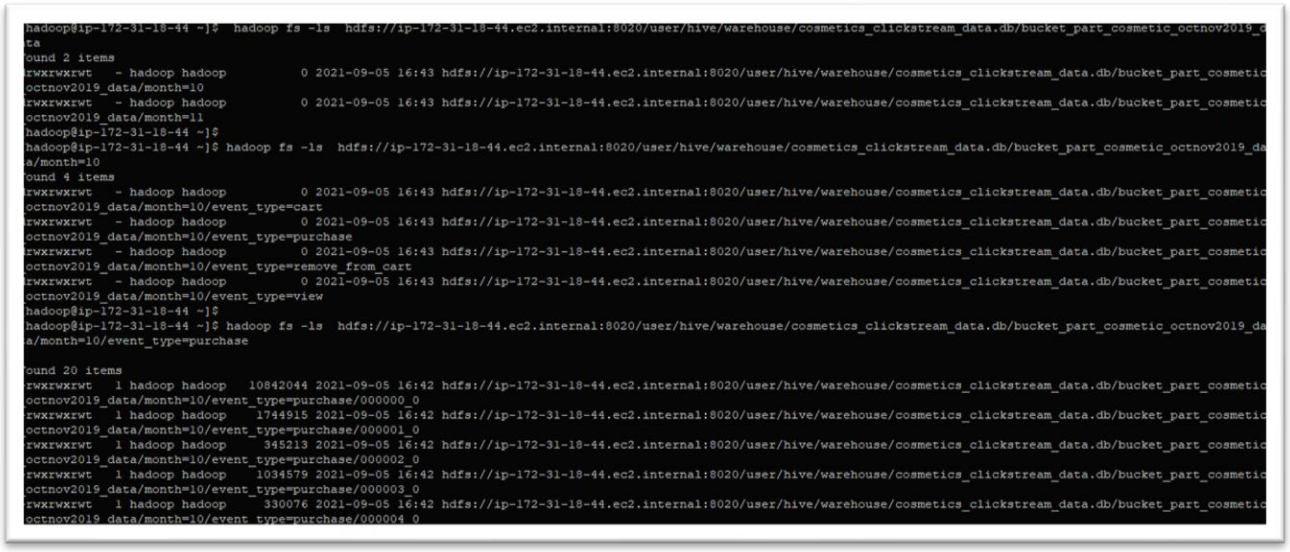
HDFS location of partitioned table

There are 4 directories, one for each partition and each partition has a single file.



HDFS location of bucketed table

There are 2 directories, one for each month. Inside each directory, there are 4 directories for each event_type. The directory has 20 files within since the rows are bucketed into 20 files.

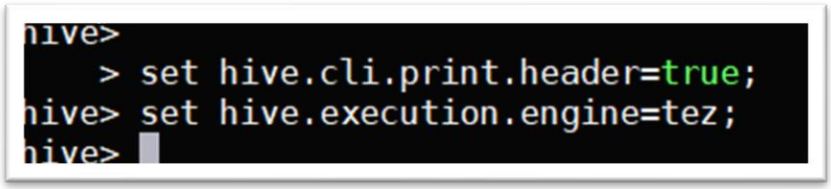


Task 3: Query Optimization

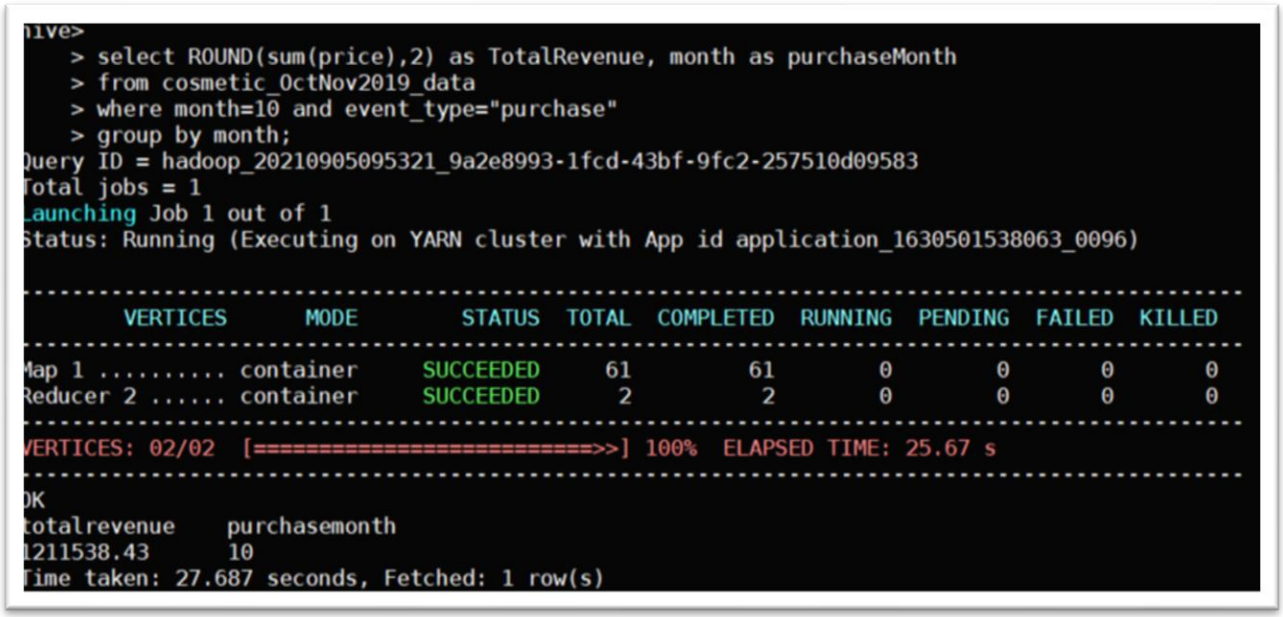
Used optimization techniques such as partitioning and bucketing and compared the query execution time on the three tables created.

Query: Check the total revenue generated due to purchases made in October

Set the below properties to execute the queries in “tez” mode and to display column headers in the queries being executed.



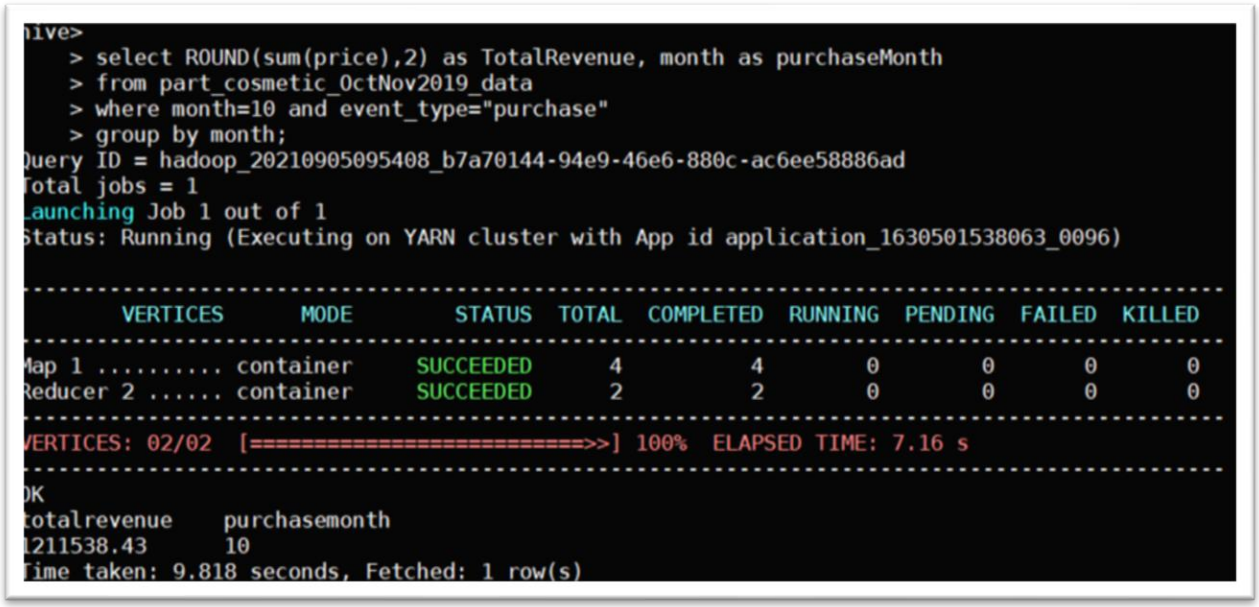
Non-bucketed table



```
hive>
    > select ROUND(sum(price),2) as TotalRevenue, month as purchaseMonth
    > from cosmetic_OctNov2019_data
    > where month=10 and event_type="purchase"
    > group by month;
Query ID = hadoop_20210905095321_9a2e8993-1fcd-43bf-9fc2-257510d09583
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1630501538063_0096)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED     61         61        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      2          2        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 25.67 s
--------------------------------------------------------------------------------
OK
totalrevenue    purchasemonth
1211538.43      10
Time taken: 27.687 seconds, Fetched: 1 row(s)
```

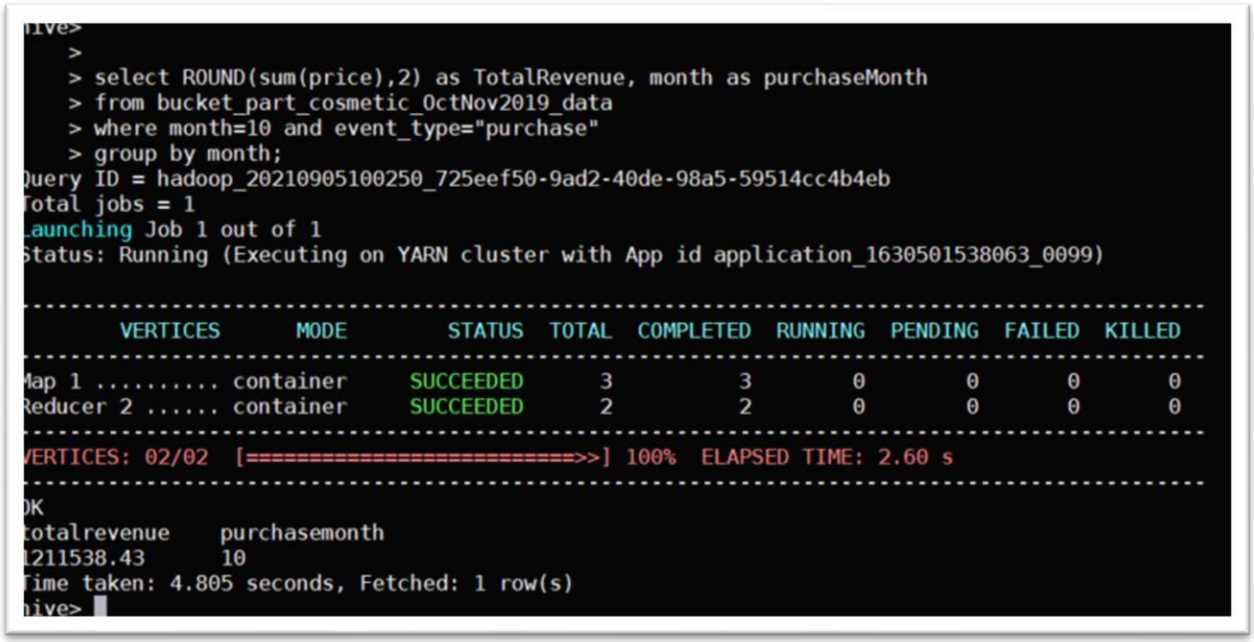
Partitioned table



```
hive>
    > select ROUND(sum(price),2) as TotalRevenue, month as purchaseMonth
    > from part_cosmetic_OctNov2019_data
    > where month=10 and event_type="purchase"
    > group by month;
Query ID = hadoop_20210905095408_b7a70144-94e9-46e6-880c-ac6ee58886ad
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1630501538063_0096)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      4          4        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      2          2        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 7.16 s
--------------------------------------------------------------------------------
OK
totalrevenue    purchasemonth
1211538.43      10
Time taken: 9.818 seconds, Fetched: 1 row(s)
```

Bucketed+ partitioned table



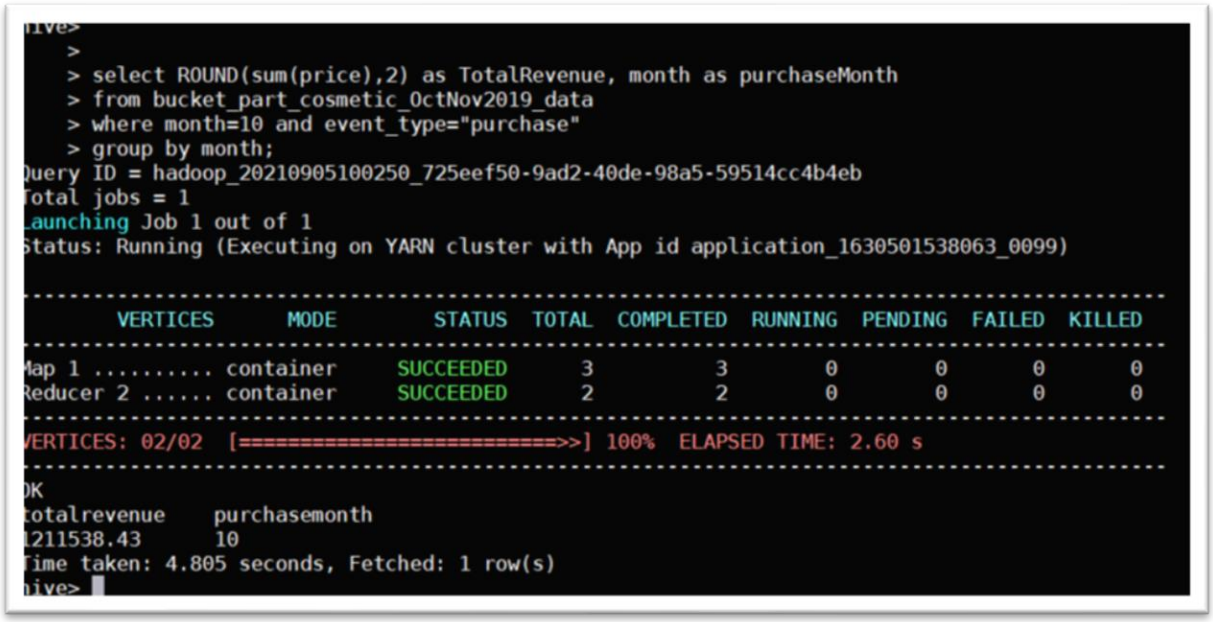
As seen from the above screenshots, the query executed faster for bucketed+ partitioned table followed by partitioned table and non-bucketed table.

Hence, the queries are executed on bucket_part_cosmetic_OctNov2019_data and the other tables created to check and compare the query optimization are deleted.

Task 4: Analysis using hive queries

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

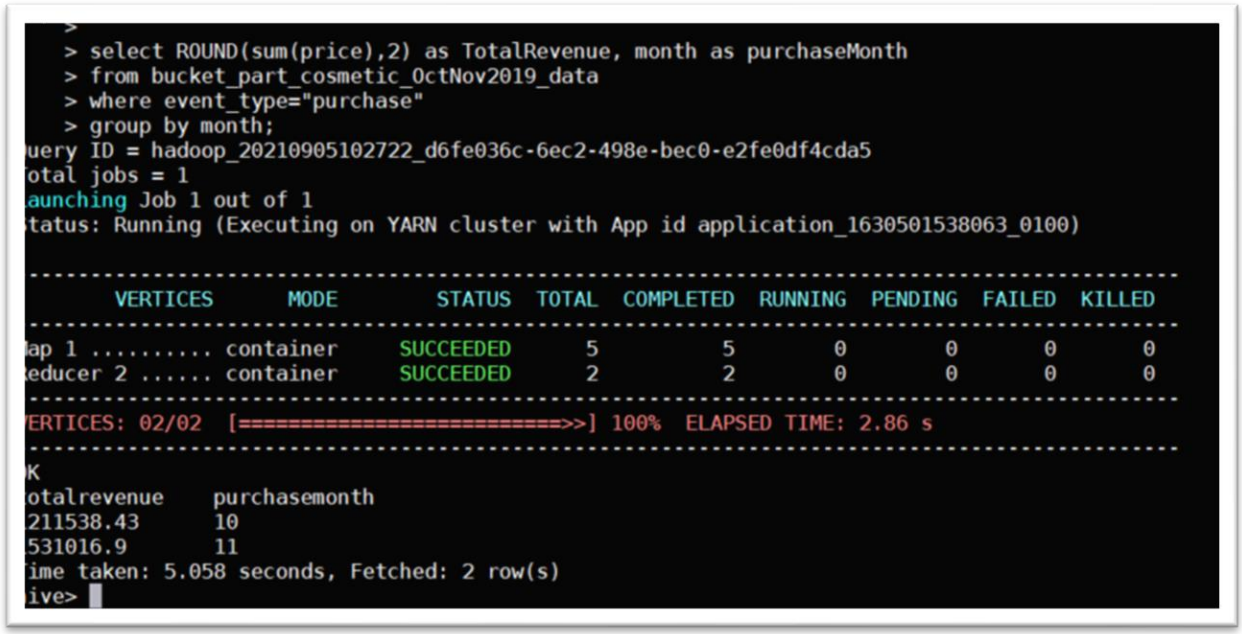
```
SELECT Round(Sum(price), 2) AS TotalRevenue,
       month                AS purchaseMonth
FROM   bucket_part_cosmetic_octnov2019_data
WHERE  month = 10
       AND event_type = "purchase"
GROUP  BY month;
```



Total revenue generated due to purchases made in October is 1211538.43

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

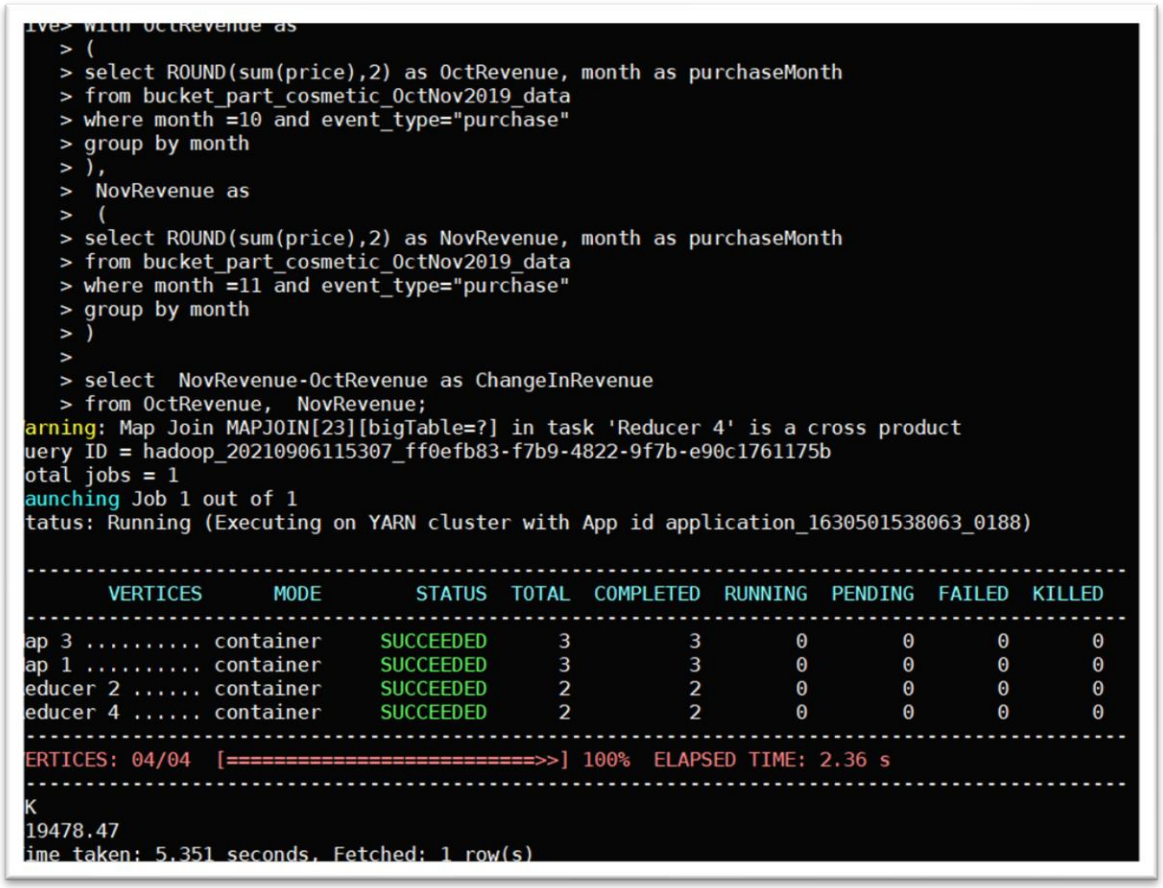
```
SELECT Round(Sum(price), 2) AS TotalRevenue,
       month                AS purchaseMonth
FROM   bucket_part_cosmetic_octnov2019_data
WHERE  event_type = "purchase"
GROUP  BY month;
```



Total revenue generated due to purchases made in October is 1211538.43 and that in November is 1531016.9.

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

```
WITH octrevenue
     AS (SELECT Round(Sum(price), 2) AS OctRevenue,
                month                AS purchaseMonth
         FROM   bucket_part_cosmetic_octnov2019_data
         WHERE  month = 10
                AND event_type = "purchase"
         GROUP  BY month),
     novrevenue
     AS (SELECT Round(Sum(price), 2) AS NovRevenue,
                month                AS purchaseMonth
         FROM   bucket_part_cosmetic_octnov2019_data
         WHERE  month = 11
                AND event_type = "purchase"
         GROUP  BY month)
SELECT novrevenue - octrevenue AS ChangeInRevenue
FROM   octrevenue, novrevenue;
```



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

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

```
SELECT DISTINCT( category_code )
FROM            bucket_part_cosmetic_octnov2019_data
WHERE           category_code IS NOT NULL
AND             category_code != "";Query: find the total number
of products available under EACH category.
SELECT    count(*),
          category_code
FROM      bucket_part_cosmetic_octnov2019_data
GROUP BY category_code;
```

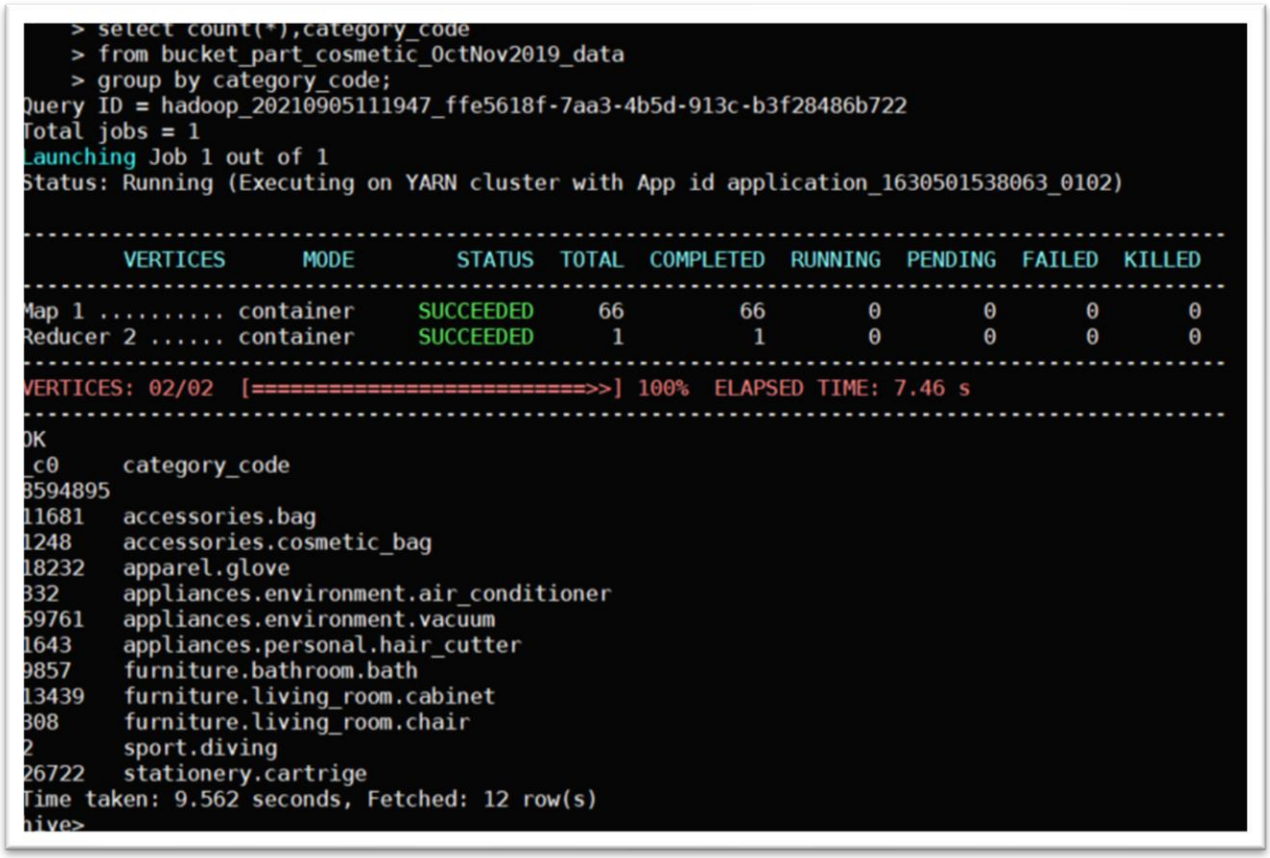
```
hive>
    > select DISTINCT(category_code)
    > from bucket_part_cosmetic_OctNov2019_data
    > where category_code IS NOT NULL AND category_code!= "";
Query ID = hadoop_20210905105724_b7aeff7a-2e8c-48be-92fb-92459dffdb1c
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1630501538063_0101)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED     66         66        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 8.18 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: 10.35 seconds, Fetched: 11 row(s)
hive>
```

There are 11 distinct categories of products.

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

```
SELECT Count(*),
       category_code
FROM   bucket_part_cosmetic_octnov2019_data
GROUP  BY category_code;
```

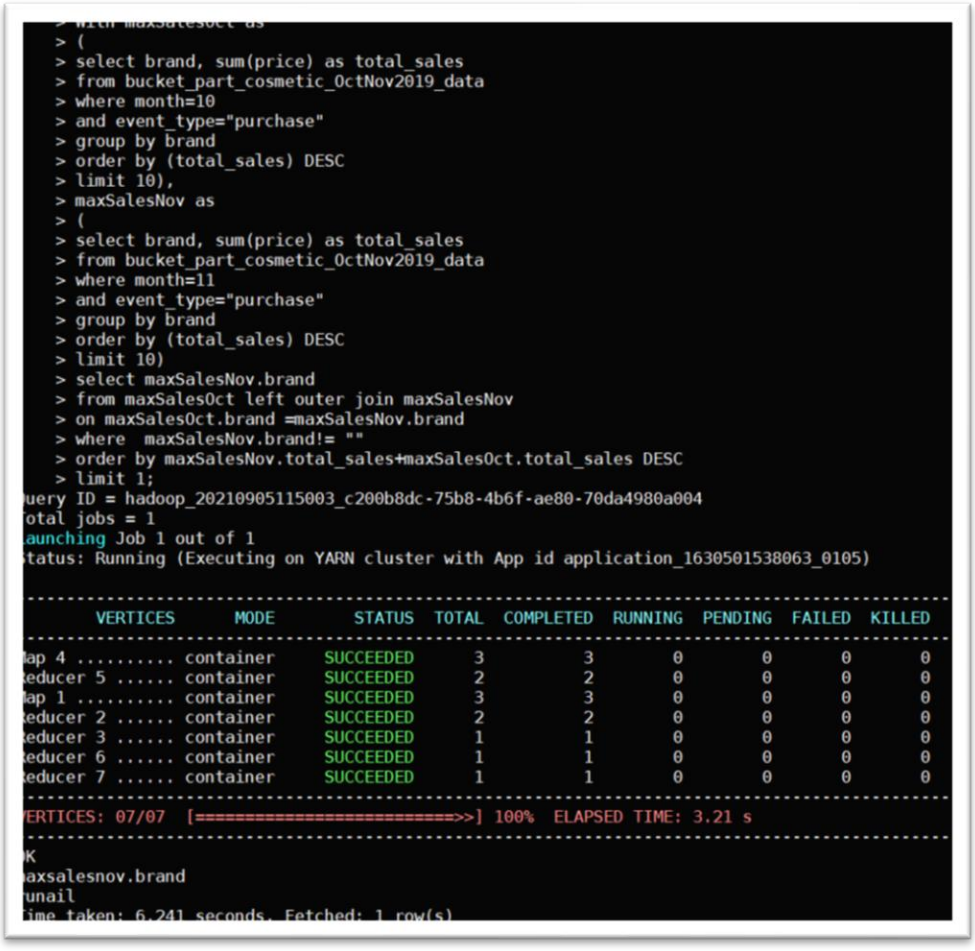


There are 8594895 entries with blank category code in the data.

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

```
WITH maxsalesoct
AS
  (
            SELECT   brand,
                     sum(price) AS total_sales
            FROM     bucket_part_cosmetic_octnov2019_data
            WHERE    month=10
            AND      event_type="purchase"
            GROUP BY brand
            ORDER BY (total_sales) DESC
            LIMIT    10),
  maxsalesnov
AS
  (
            SELECT   brand,
```

```
                    sum(price) AS total_sales
          FROM      bucket_part_cosmetic_octnov2019_data
          WHERE     month=11
          AND       event_type="purchase"
          GROUP BY brand
          ORDER BY (total_sales) DESC
          LIMIT     10)
 SELECT           maxsalesnov.brand
 FROM             maxsalesoct
 LEFT OUTER JOIN maxsalesnov
 ON               maxsalesoct.brand =maxsalesnov.brand
 WHERE            maxsalesnov.brand!= ""
 ORDER BY         maxsalesnov.total_sales+maxsalesoct.total_sales
DESC
 LIMIT            1;
```

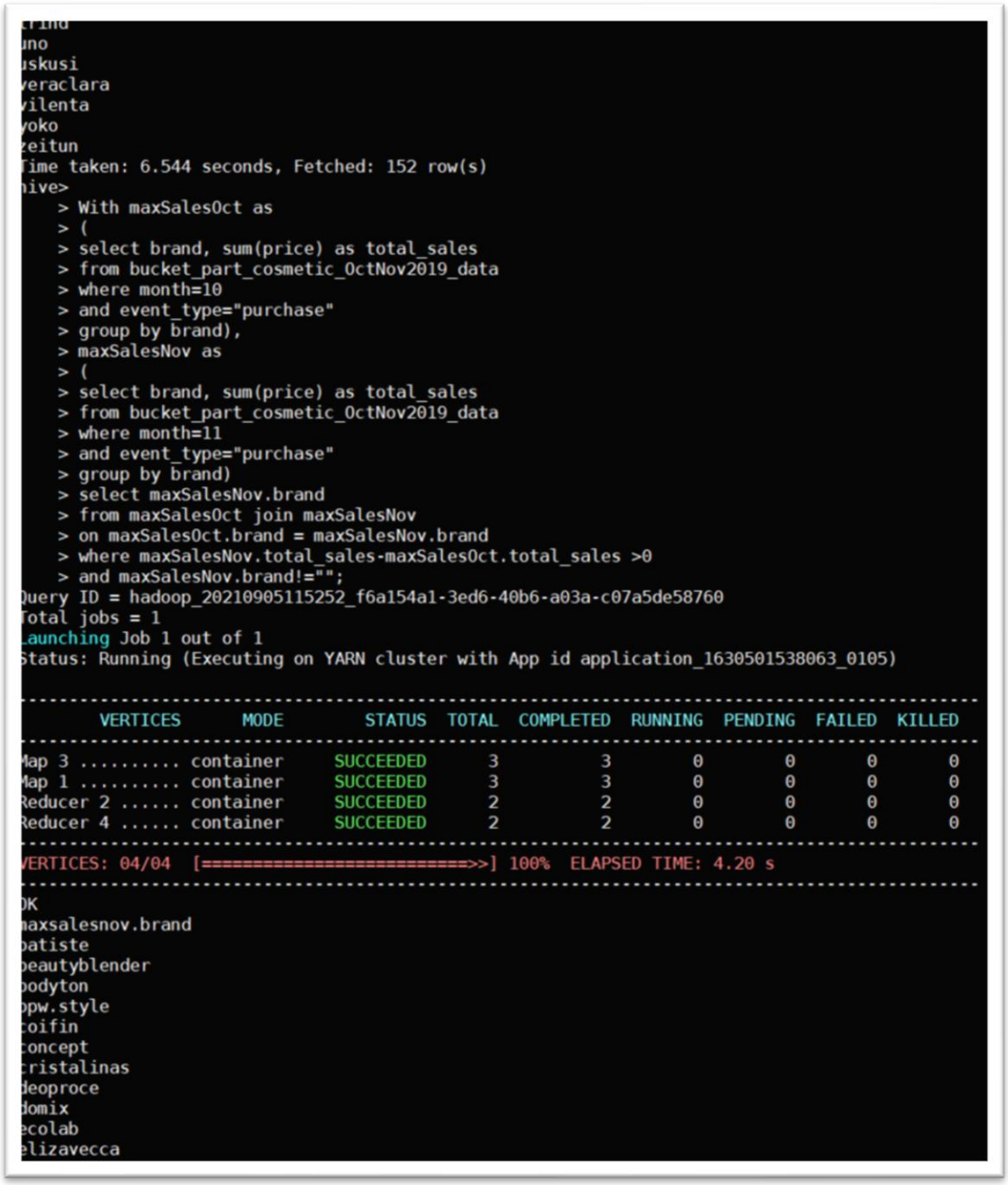


Runail had the maximum sales in October and November combined

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

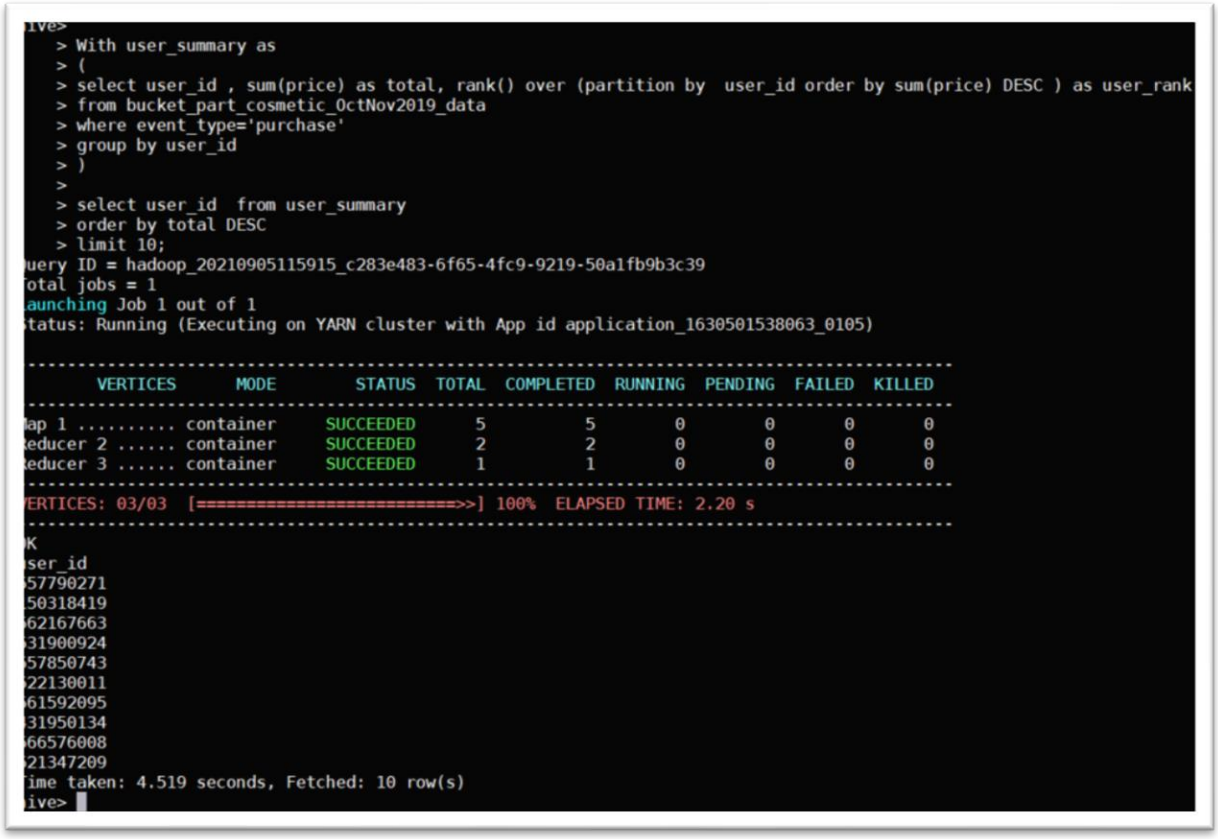
```
WITH maxsalesoct
     AS (SELECT brand,
                Sum(price) AS total_sales
         FROM   bucket_part_cosmetic_octnov2019_data
         WHERE  month = 10
                AND event_type = "purchase"
         GROUP  BY brand),
     maxsalesnov
     AS (SELECT brand,
                Sum(price) AS total_sales
         FROM   bucket_part_cosmetic_octnov2019_data
         WHERE  month = 11
                AND event_type = "purchase"
         GROUP  BY brand)
SELECT maxsalesnov.brand
FROM   maxsalesoct
       JOIN maxsalesnov
         ON maxsalesoct.brand = maxsalesnov.brand
WHERE  maxsalesnov.total_sales - maxsalesoct.total_sales > 0
       AND maxsalesnov.brand != "";
```

There are 152 brands which have increased their sales from October to November. First few and last few brands of the list are displayed in the screenshot



Query: 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.

```
WITH user_summary
AS
  (
            SELECT    user_id ,
                      sum(price)
                 AS total,
                      rank() over (partition BY user_id ORDER BY su
m(price) DESC ) AS user_rank
            FROM      bucket_part_cosmetic_octnov2019_data
            WHERE     event_type='purchase'
            GROUP BY user_id )
  SELECT    user_id
  FROM      user_summary
  ORDER BY total DESC
  LIMIT     10;
```



The user ids of top 10 users of the website who spend the most is displayed.