HIVE CASE STUDY REPORT

1 Problem Statement

For this assignment, we will be working with a public clickstream dataset of a cosmetics store. Using this dataset, we will extract valuable insights using optimized queries. In addition, we have shown the improvement of the performance after using the optimization techniques on first query.

We have used the data in the link given below:

https://e-commerce-events-ml.s3.amazonaws.com/2019-Oct.csv
https://e-commerce-events-ml.s3.amazonaws.com/2019-Nov.csv

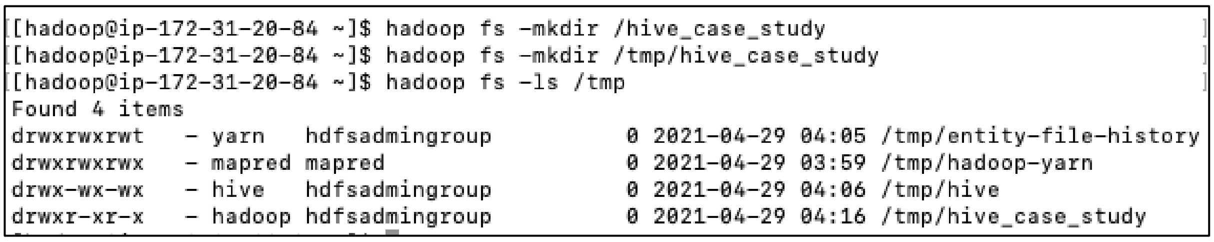
Both the Dataset, 2019-Oct.csv and 2019-Nov.csv consist of 9 attributes. Description and Data type of all the attributes are given in the below table.

Attribute_Name	Data_type	Description
event_time	timestamp	Time at which the event took place
event_type	string	Event type may be 'view', 'cart', 'remove_from_cart', 'purchase'
product_id	string	Unique identification of the product
category_id	string	Unique identification of the product category. Each product category contains several products
category_code	string	Name (if present) of the product category
brand	string	Name of the brand
price	float	Price of the product
user_id	bigint	Permanent user id
user_session	string	Identification for the user's session. Remains same for each user's session. It changes every time the user returns back the website after a long pause.

2 Implementation Phase

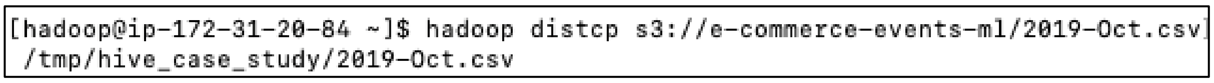
2.1 Copying the data set into the HDFS

We have launched an EMR cluster that utilizes the Hive services and connected our EMR cluster with master node. In the next step we have to copy the dataset into the HDFS, therefore, we made the folder ‘hive_case_study’ under ‘tmp’ folder.

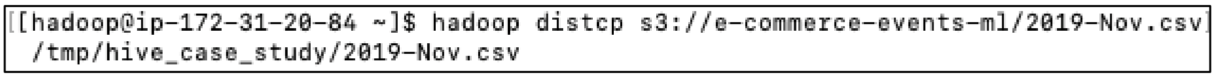


```
[hadoop@ip-172-31-20-84 ~]$ hadoop fs -mkdir /hive_case_study
[hadoop@ip-172-31-20-84 ~]$ hadoop fs -mkdir /tmp/hive_case_study
[hadoop@ip-172-31-20-84 ~]$ hadoop fs -ls /tmp
Found 4 items
drwxrwxrwt   - yarn   hdfsadmingroup          0 2021-04-29 04:05 /tmp/entity-file-history
drwxrwxrwx   - mapred mapred                  0 2021-04-29 03:59 /tmp/hadoop-yarn
drwx-wx-wx   - hive   hdfsadmingroup          0 2021-04-29 04:06 /tmp/hive
drwxr-xr-x   - hadoop hdfsadmingroup          0 2021-04-29 04:16 /tmp/hive_case_study
```

Copy the data from the S3 bucket into the HDFS.

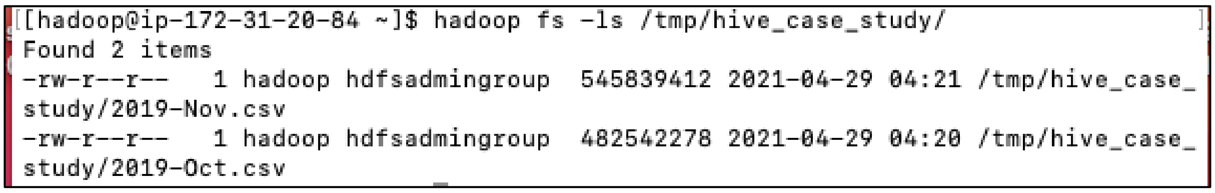


```
[hadoop@ip-172-31-20-84 ~]$ hadoop distcp s3://e-commerce-events-ml/2019-Oct.csv
 /tmp/hive_case_study/2019-Oct.csv
```



```
[hadoop@ip-172-31-20-84 ~]$ hadoop distcp s3://e-commerce-events-ml/2019-Nov.csv
 /tmp/hive_case_study/2019-Nov.csv
```

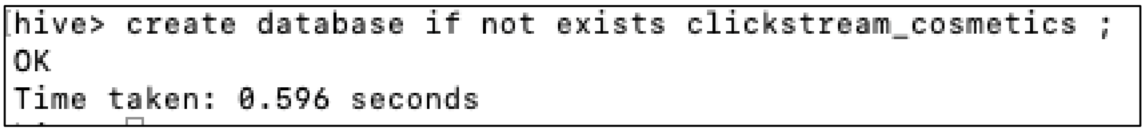
We can observe from the below snapshot that data has been successfully copied to the HDFS.



```
[hadoop@ip-172-31-20-84 ~]$ hadoop fs -ls /tmp/hive_case_study/
Found 2 items
-rw-r--r--   1 hadoop hdfsadmingroup  545839412 2021-04-29 04:21 /tmp/hive_case_
study/2019-Nov.csv
-rw-r--r--   1 hadoop hdfsadmingroup  482542278 2021-04-29 04:20 /tmp/hive_case_
study/2019-Oct.csv
```

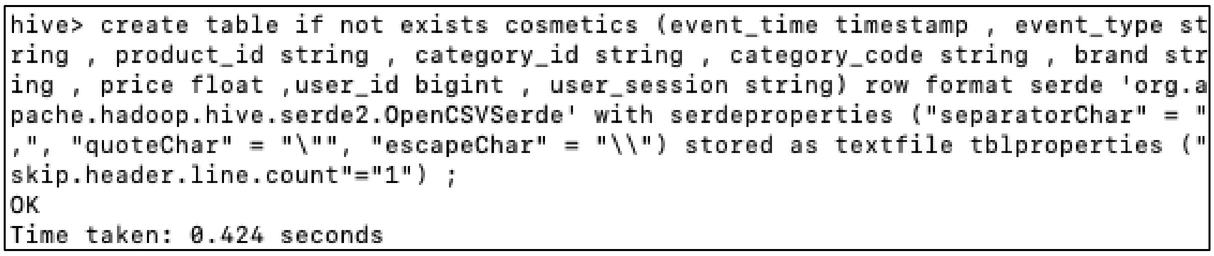
2.2 Creating the database and structure of the table

Create database 'clickstream_cosmetics'



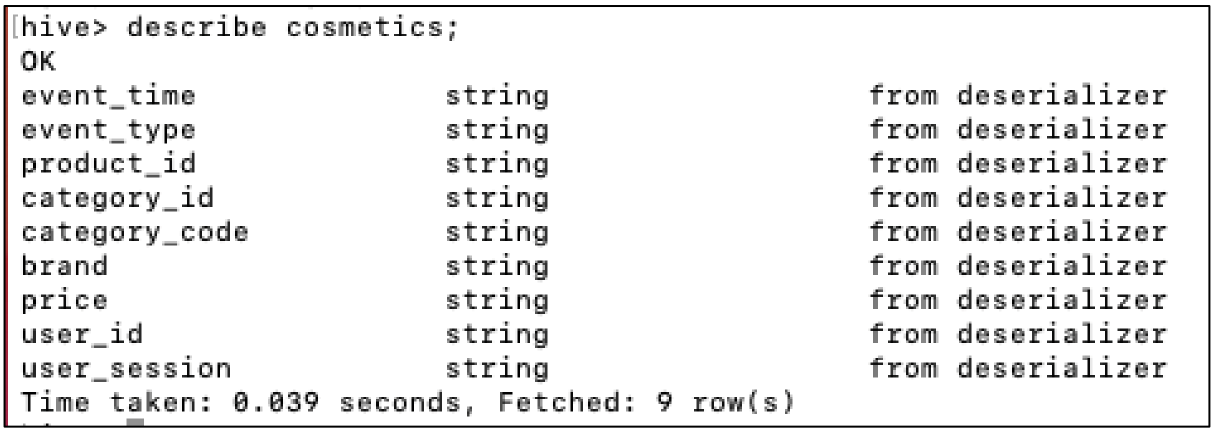
```
hive> create database if not exists clickstream_cosmetics ;
OK
Time taken: 0.596 seconds
```

Create the table 'cosmetics' with all the 9 attributes present in the dataset. We have used CSVSerde with the default properties value.



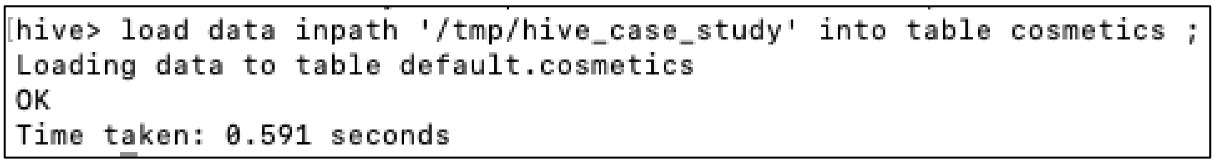
```
hive> create table if not exists cosmetics (event_time timestamp , event_type st
ring , product_id string , category_id string , category_code string , brand str
ing , price float ,user_id bigint , user_session string) row format serde 'org.a
pache.hadoop.hive.serde2.OpenCSVSerde' with serdeproperties ("separatorChar" = "
,", "quoteChar" = "\"", "escapeChar" = "\\") stored as textfile tblproperties ("
skip.header.line.count"="1") ;
OK
Time taken: 0.424 seconds
```

Let's look at the 'cosmetics' table schema.



```
hive> describe cosmetics;
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: 0.039 seconds, Fetched: 9 row(s)
```

Load the data into the table 'cosmetics'

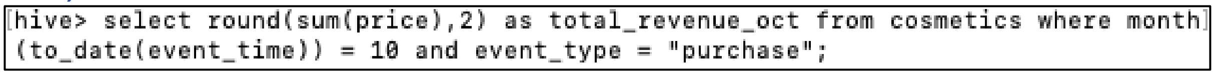


```
hive> load data inpath '/tmp/hive_case_study' into table cosmetics ;
Loading data to table default.cosmetics
OK
Time taken: 0.591 seconds
```

2.3 Launching Hive queries on the EMR cluster

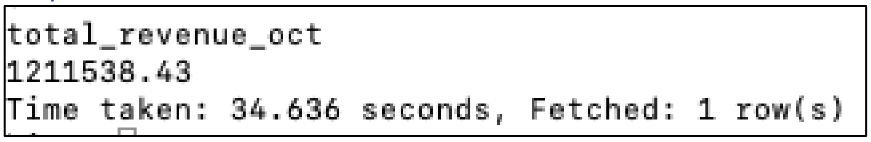
2.3.1 Find the total revenue generated due to purchases made in October

Query:



```
hive> select round(sum(price),2) as total_revenue_oct from cosmetics where month
(to_date(event_time)) = 10 and event_type = "purchase";
```

Output:

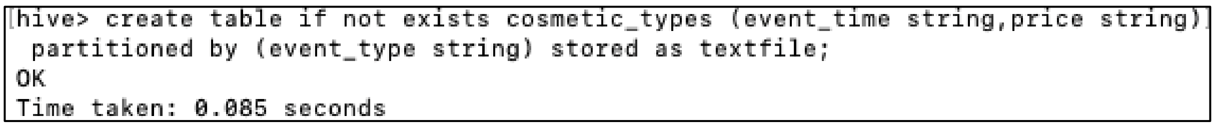


```
total_revenue_oct
1211538.43
Time taken: 34.636 seconds, Fetched: 1 row(s)
```

Time taken before optimizing: 34.636 seconds

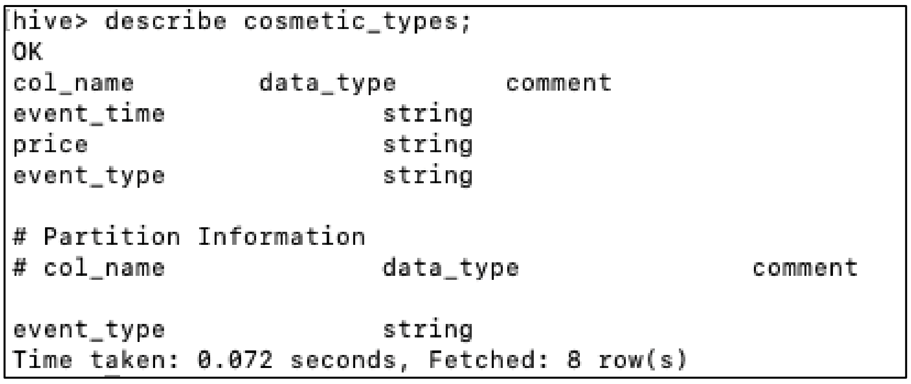
2.3.1.1 Optimizing the query

We will use optimization technique to improve the hive performance for the above query. Create Partitioned table 'cosmetic_types' based on 'event_type'.



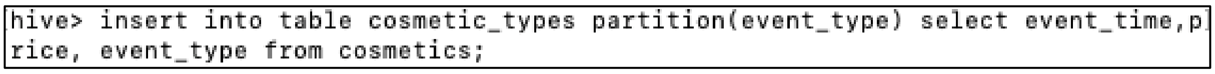
```
hive> create table if not exists cosmetic_types (event_time string,price string)
 partitioned by (event_type string) stored as textfile;
OK
Time taken: 0.085 seconds
```

Let's look at the partitioned table 'cosmetic_types' schema.



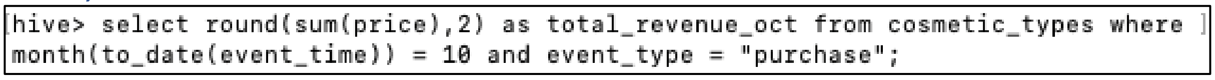
```
hive> describe cosmetic_types;
OK
col_name        data_type       comment
event_time              string
price                   string
event_type              string

# Partition Information
# col_name              data_type               comment

event_type              string
Time taken: 0.072 seconds, Fetched: 8 row(s)
```

Insert the data into partitioned table 'cosmetic_types'.



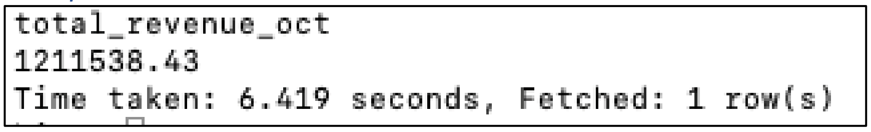
```
hive> insert into table cosmetic_types partition(event_type) select event_time,p
rice, event_type from cosmetics;
```

Query on Partitioned Table:



```
hive> select round(sum(price),2) as total_revenue_oct from cosmetic_types where 
month(to_date(event_time)) = 10 and event_type = "purchase";
```

Output:



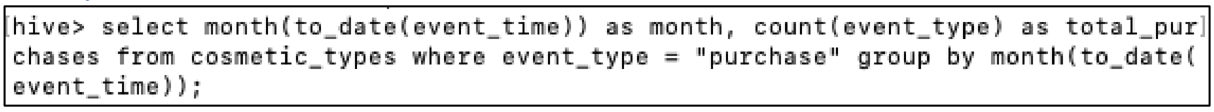
```
total_revenue_oct
1211538.43
Time taken: 6.419 seconds, Fetched: 1 row(s)
```

Time taken after Partitioning: 6.419 seconds

We can observe from the above two queries that hive performance is increased after using optimization technique. Time taken to run the query is decreased from 34.636 to 6.419 when we used partitioned table.

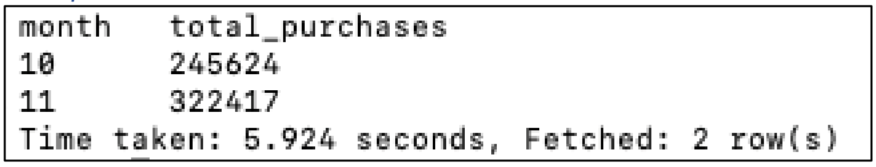
2.3.2 Write a query to yield the total sum of purchases per month in a single output.

Query:



```
hive> select month(to_date(event_time)) as month, count(event_type) as total_pur
chases from cosmetic_types where event_type = "purchase" group by month(to_date(
event_time));
```

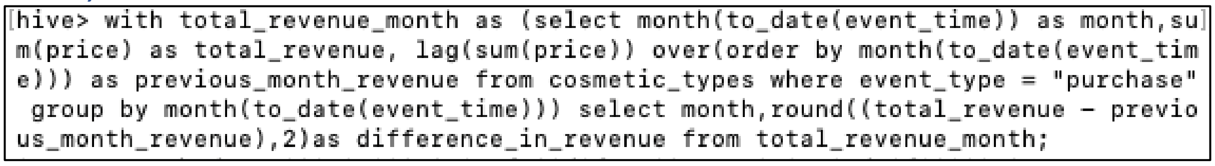
Output:



```
month   total_purchases
10      245624
11      322417
Time taken: 5.924 seconds, Fetched: 2 row(s)
```

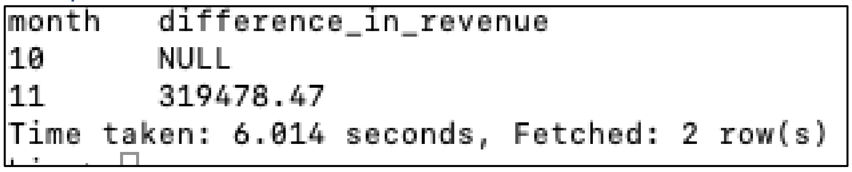
2.3.3 Write a query to find the change in revenue generated due to purchases from October to November

Query:



```
hive> with total_revenue_month as (select month(to_date(event_time)) as month,su
m(price) as total_revenue, lag(sum(price)) over(order by month(to_date(event_tim
e))) as previous_month_revenue from cosmetic_types where event_type = "purchase"
 group by month(to_date(event_time))) select month,round((total_revenue - previo
us_month_revenue),2)as difference_in_revenue from total_revenue_month;
```

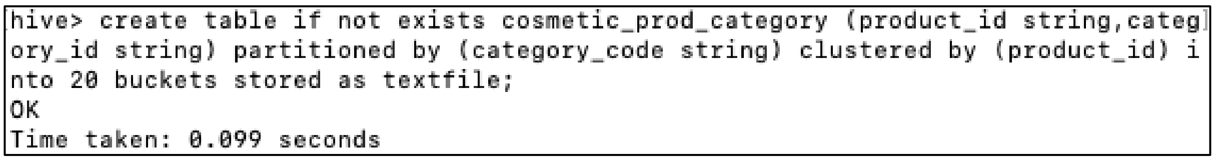
Output:



```
month   difference_in_revenue
10      NULL
11      319478.47
Time taken: 6.014 seconds, Fetched: 2 row(s)
```

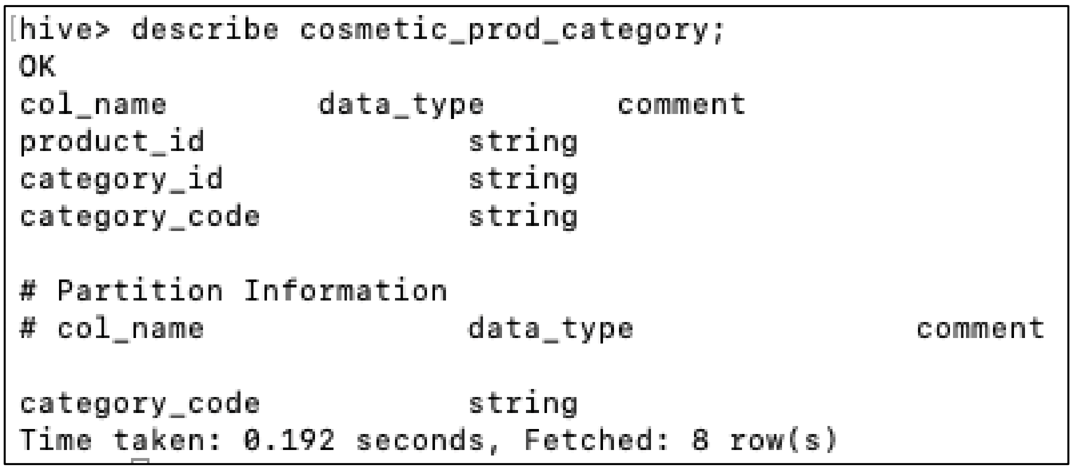
2.3.4 Find distinct categories of products. Categories with null category code can be ignored

For the above query we will create a table 'cosmetic_prod_category, partitioned based on 'category_code' and store the data into 20 buckets on the basis of 'product_id'.



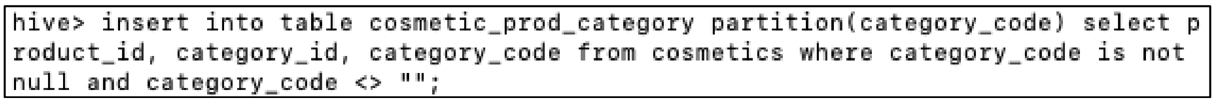
```
hive> create table if not exists cosmetic_prod_category (product_id string,categ
ory_id string) partitioned by (category_code string) clustered by (product_id) i
nto 20 buckets stored as textfile;
OK
Time taken: 0.099 seconds
```

Let's look at the 'cosmetic_prod_category' table schema.



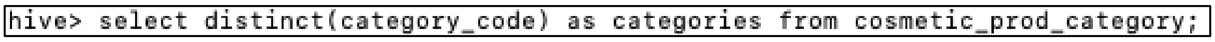
```
hive> describe cosmetic_prod_category;
OK
col_name        data_type       comment
product_id              string
category_id             string
category_code           string

# Partition Information
# col_name              data_type               comment

category_code           string
Time taken: 0.192 seconds, Fetched: 8 row(s)
```

Insert the data into table 'cosmetic_prod_category'.



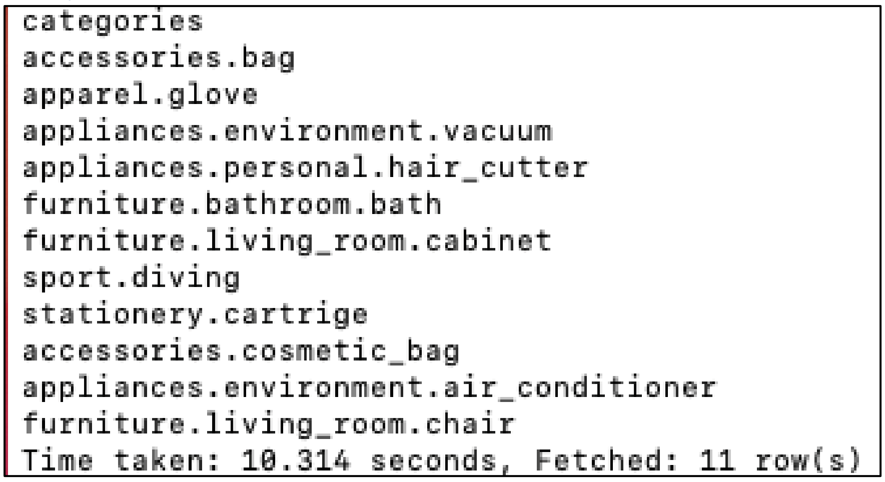
```
hive> insert into table cosmetic_prod_category partition(category_code) select p
roduct_id, category_id, category_code from cosmetics where category_code is not
null and category_code <> "";
```

Query:



```
hive> select distinct(category_code) as categories from cosmetic_prod_category;
```

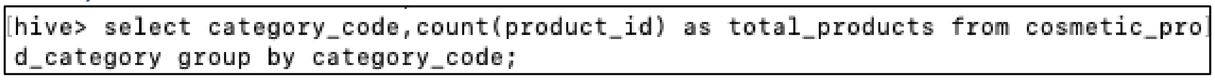
Output:



```
categories
accessories.bag
apparel.glove
appliances.environment.vacuum
appliances.personal.hair_cutter
furniture.bathroom.bath
furniture.living_room.cabinet
sport.diving
stationery.cartrige
accessories.cosmetic_bag
appliances.environment.air_conditioner
furniture.living_room.chair
Time taken: 10.314 seconds, Fetched: 11 row(s)
```

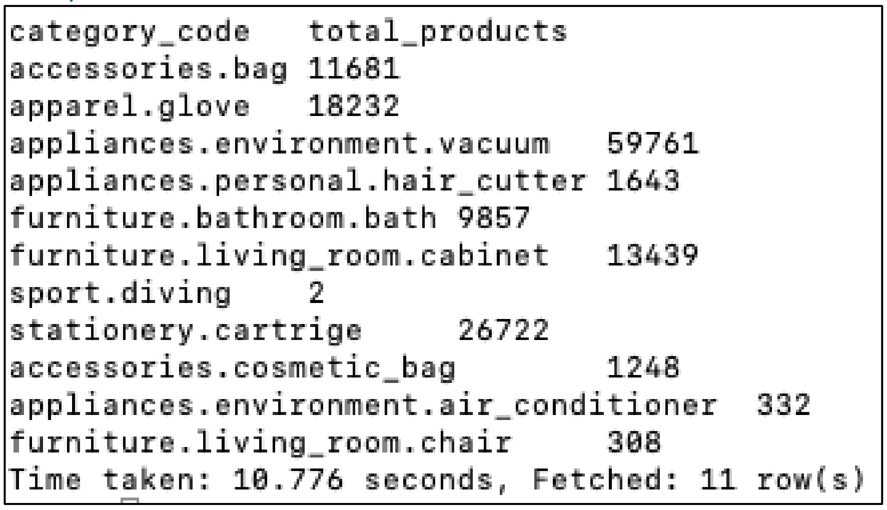
2.3.5 Find the total number of products available under each category

Query:



```
hive> select category_code,count(product_id) as total_products from cosmetic_pro
d_category group by category_code;
```

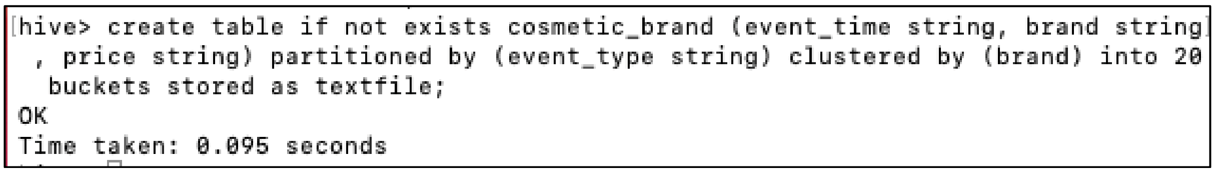
Output:



```
category_code   total_products
accessories.bag 11681
apparel.glove   18232
appliances.environment.vacuum   59761
appliances.personal.hair_cutter 1643
furniture.bathroom.bath 9857
furniture.living_room.cabinet   13439
sport.diving    2
stationery.cartrige     26722
accessories.cosmetic_bag        1248
appliances.environment.air_conditioner  332
furniture.living_room.chair     308
Time taken: 10.776 seconds, Fetched: 11 row(s)
```

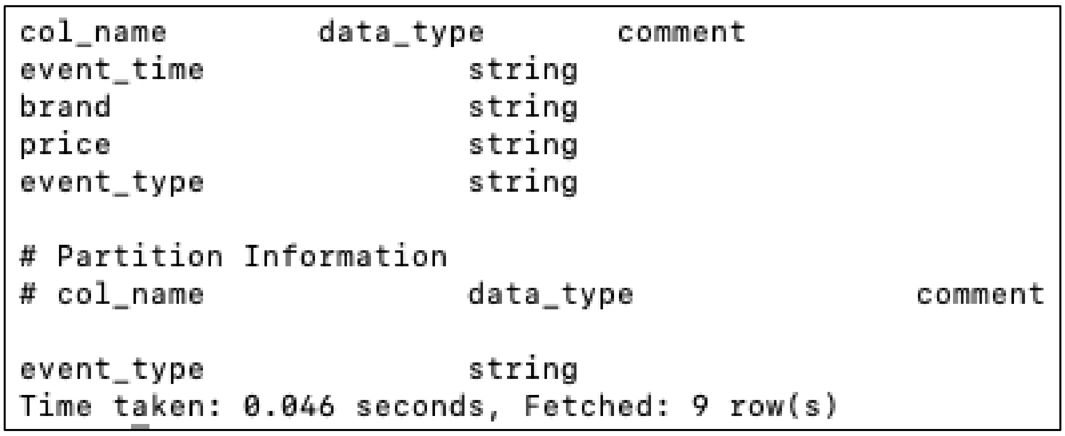
2.3.6 Which brand had the maximum sales in October and November combined?

For the above query we will create a table 'cosmetic_brand', partitioned by 'event_type' and bucketing on the basis of 'brand'.



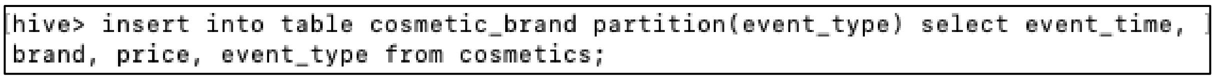
```
hive> create table if not exists cosmetic_brand (event_time string, brand string
 , price string) partitioned by (event_type string) clustered by (brand) into 20
 buckets stored as textfile;
OK
Time taken: 0.095 seconds
```

Let's look at the 'cosmetic_brand' table schema.



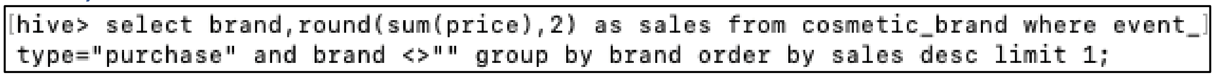
```
col_name            data_type           comment
event_time                  string
brand                       string
price                       string
event_type                  string

# Partition Information
# col_name                  data_type                       comment

event_type                  string
Time taken: 0.046 seconds, Fetched: 9 row(s)
```

Insert the data into 'cosmetic_brand' table.



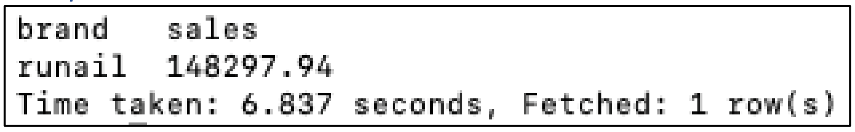
```
hive> insert into table cosmetic_brand partition(event_type) select event_time, 
brand, price, event_type from cosmetics;
```

Query:



```
hive> select brand,round(sum(price),2) as sales from cosmetic_brand where event_
type="purchase" and brand <>"" group by brand order by sales desc limit 1;
```

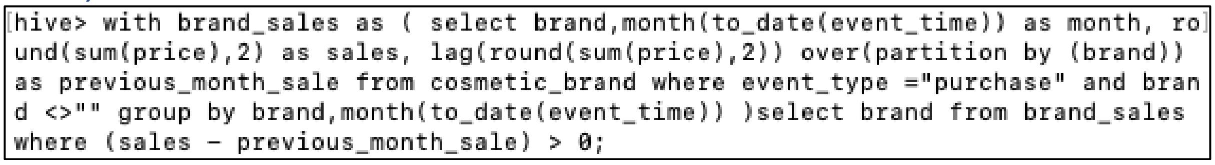
Output:



```
brand   sales
runail  148297.94
Time taken: 6.837 seconds, Fetched: 1 row(s)
```

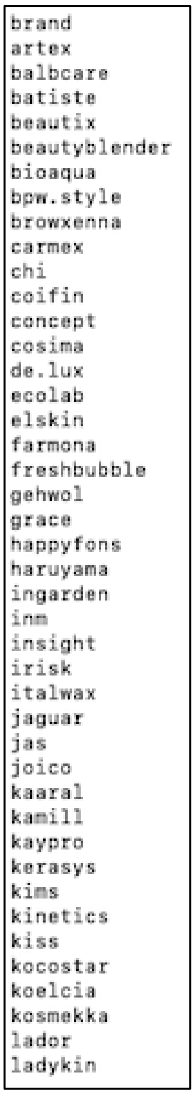
2.3.7 Which brands increased their sales from October to November?

Query:

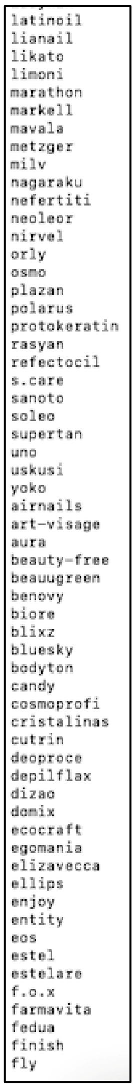


```
hive> with brand_sales as ( select brand,month(to_date(event_time)) as month, ro
und(sum(price),2) as sales, lag(round(sum(price),2)) over(partition by (brand))
as previous_month_sale from cosmetic_brand where event_type ="purchase" and bran
d <>"" group by brand,month(to_date(event_time)) )select brand from brand_sales
where (sales - previous_month_sale) > 0;
```

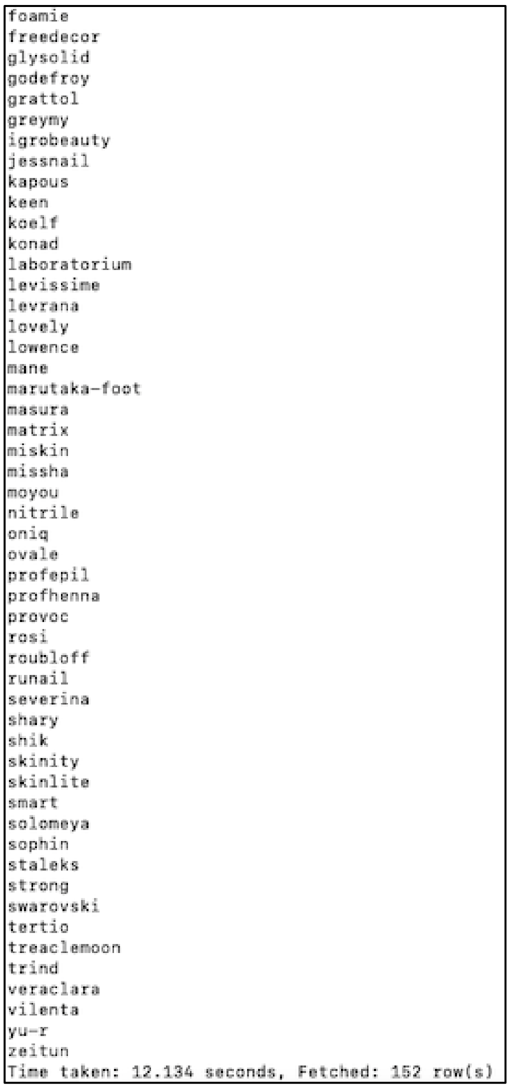
Output:



```
brand
artex
balbcare
batiste
beautix
beautyblender
bioaqua
bpw.style
browxenna
carmex
chi
coifin
concept
cosima
de.lux
ecolab
elskin
farmona
freshbubble
gehwol
grace
happyfons
haruyama
ingarden
inm
insight
irisk
italwax
jaguar
jas
joico
kaaral
kamill
kaypro
kerasys
kims
kinetics
kiss
kocostar
koelcia
kosmekka
lador
ladykin
```



```
latinoil
lianail
likato
limoni
marathon
markell
mavala
metzger
milv
nagaraku
nefertiti
neoleor
nirvel
orly
osmo
plazan
polarus
protokeratin
rasyan
refectocil
s.care
sanoto
soleo
supertan
uno
uskusi
yoko
airnails
art-visage
aura
beauty-free
beauugreen
benovy
biore
blixz
bluesky
bodyton
candy
cosmoprofi
cristalinas
cutrin
deoproce
depilflax
dizao
domix
ecocraft
egomania
elizavecca
ellips
enjoy
entity
eos
estel
estelare
f.o.x
farmavita
fedua
finish
fly
```

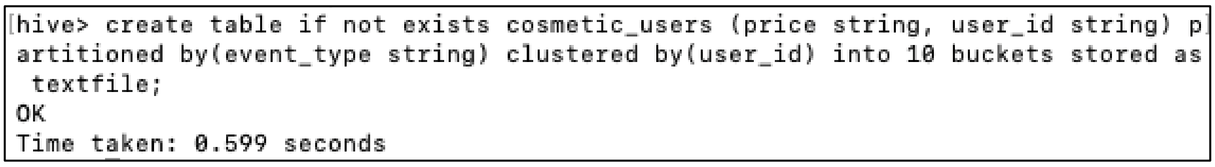


```
foamie
freedecor
glysolid
godefroy
grattol
greymy
igrobeauty
jessnail
kapous
keen
koelf
konad
laboratorium
levissime
levrana
lovely
lowence
mane
marutaka-foot
masura
matrix
miskin
missha
moyou
nitrile
oniq
ovale
profepil
profhenna
provoc
rosi
roubloff
runail
severina
shary
shik
skinity
skinlite
smart
solomeya
sophin
staleks
strong
swarovski
tertio
treaclemoon
trind
veraclara
vilenta
yu-r
zeitun
Time taken: 12.134 seconds, Fetched: 152 row(s)
```

152 out of 245 distinct brands increased their sales from October to November.

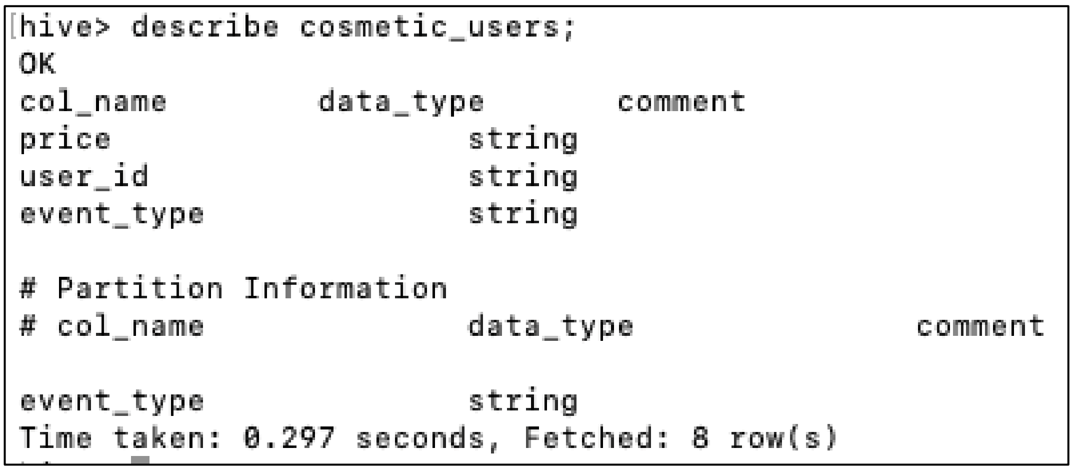
2.3.8 Your company wants to reward the top 10 users of its website with a Golden Customer plan. Write a query to generate a list of top 10 users who spend the most

For the above query we will create a table 'cosmetic_users', partitioned by 'event_type' and bucketed by 'user_id'.



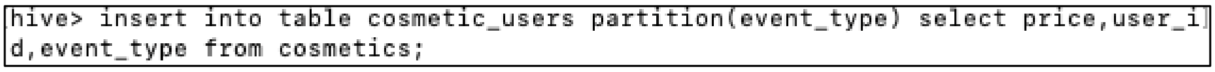
```
[hive> create table if not exists cosmetic_users (price string, user_id string) p
artitioned by(event_type string) clustered by(user_id) into 10 buckets stored as
 textfile;
OK
Time taken: 0.599 seconds
```

Let's look at the 'cosmetic_users' schema.



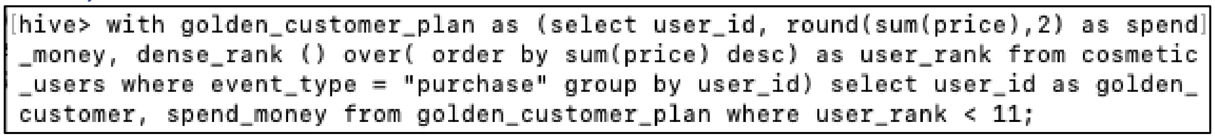
```
hive> describe cosmetic_users;
OK
col_name        data_type       comment
price                   string
user_id                 string
event_type              string

# Partition Information
# col_name              data_type               comment

event_type              string
Time taken: 0.297 seconds, Fetched: 8 row(s)
```

Insert the data into the ‘cosmetic_users’ table.



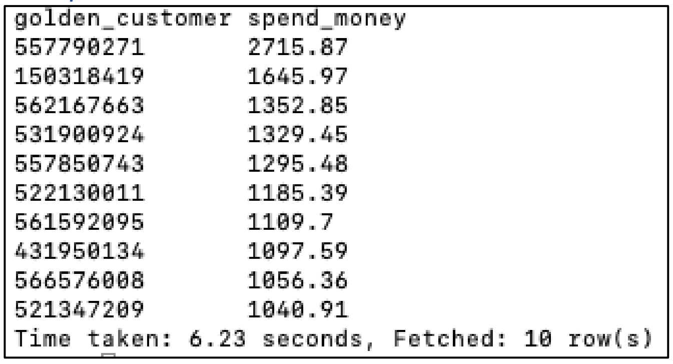
```
hive> insert into table cosmetic_users partition(event_type) select price,user_i
d,event_type from cosmetics;
```

Query:



```
hive> with golden_customer_plan as (select user_id, round(sum(price),2) as spend
_money, dense_rank () over( order by sum(price) desc) as user_rank from cosmetic
_users where event_type = "purchase" group by user_id) select user_id as golden_
customer, spend_money from golden_customer_plan where user_rank < 11;
```

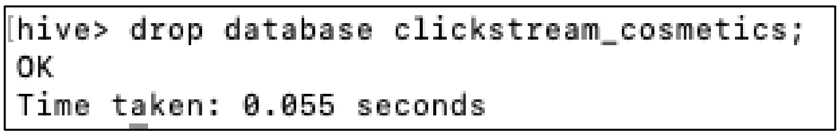
Output:



```
golden_customer spend_money
557790271       2715.87
150318419       1645.97
562167663       1352.85
531900924       1329.45
557850743       1295.48
522130011       1185.39
561592095       1109.7
431950134       1097.59
566576008       1056.36
521347209       1040.91
Time taken: 6.23 seconds, Fetched: 10 row(s)
```

2.4 Cleaning up

After completing the hive queries, we have dropped the database and terminated the cluster.



```
hive> drop database clickstream_cosmetics;
OK
Time taken: 0.055 seconds
```