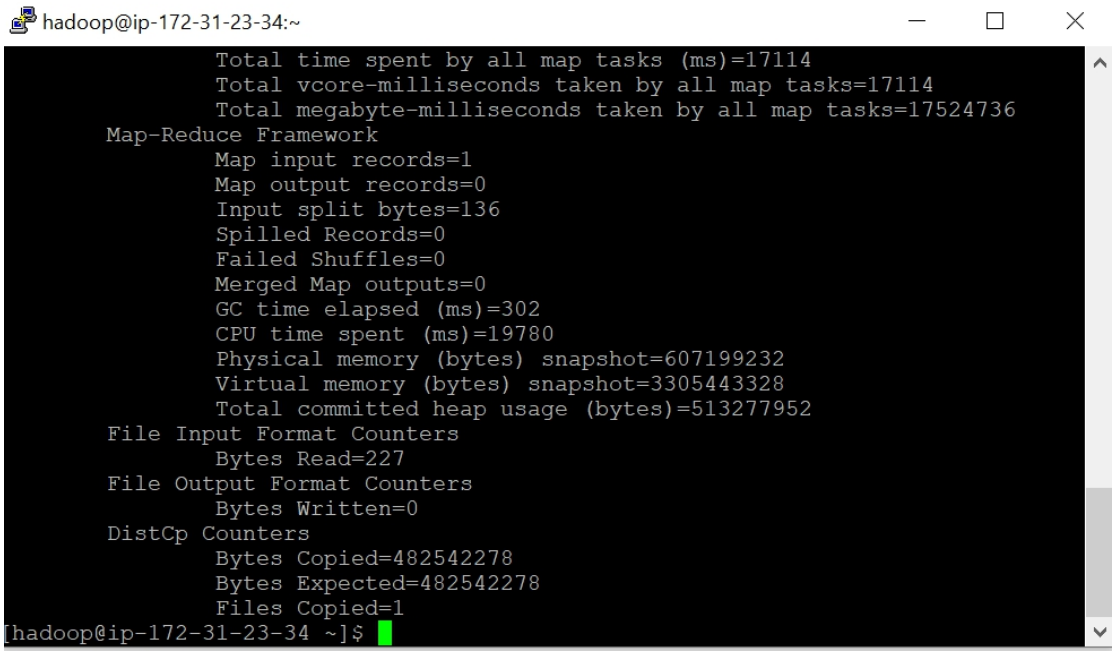
Hive Query Case Study

1. Create a folder in HDFS to store the data files
   hadoop fs -mkdir /tmp/test-folder

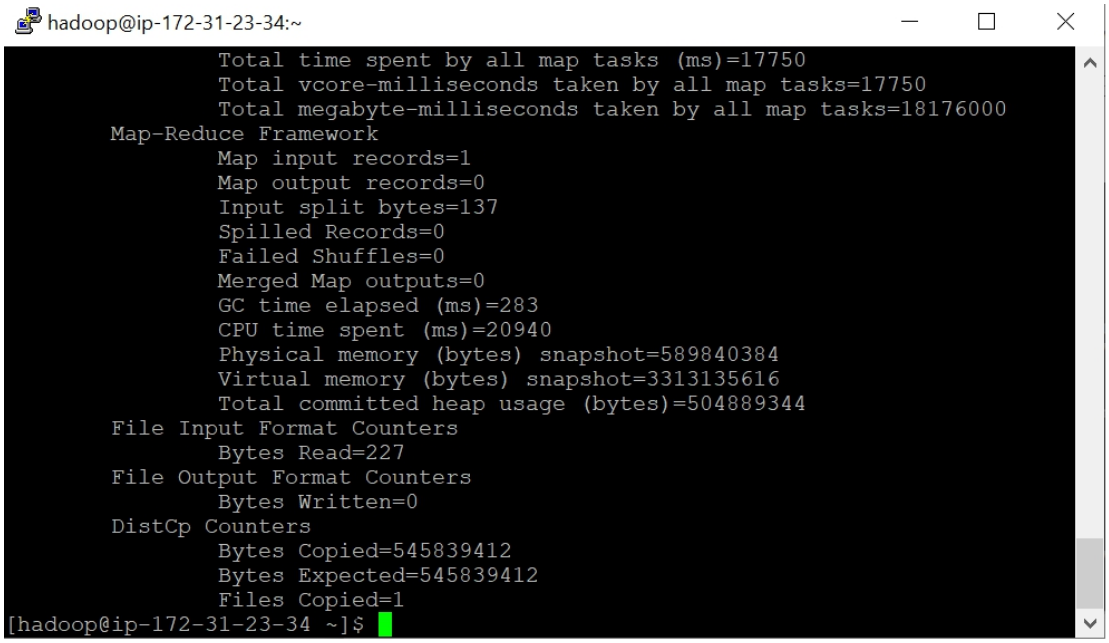
```
[hadoop@ip-172-31-23-34 ~]$ hadoop fs -mkdir /tmp/test-folder
[hadoop@ip-172-31-23-34 ~]$ hadoop fs -ls /tmp/
Found 3 items
drwxrwxrwx   - mapred mapred          0 2021-05-03 13:55 /tmp/hadoop-yarn
drwx-wx-wx   - hive   hadoop          0 2021-05-03 13:57 /tmp/hive
drwxr-xr-x   - hadoop hadoop          0 2021-05-03 14:17 /tmp/test-folder
[hadoop@ip-172-31-23-34 ~]$
```

2. Copy the data files provided to "/tmp/test-folder folder" folder created on HDFS

   hadoop distcp s3://bucketvijay/2019-Oct.csv /tmp/test-folder/

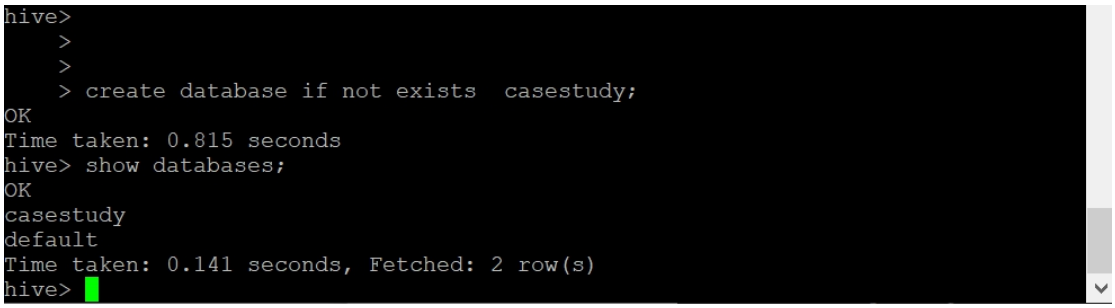


hadoop distcp s3://bucketvijay/2019-Nov.csv /tmp/test-folder/



3. Create a database called “casestudy”

   create database if not exists casestudy;



4. Create a table, load the CSV data files and skip the header row.

```
create table if not exists ecom (
event_time timestamp,
event_type string,
product_id string,
category_id string,
category_code string,
brand string,
price decimal(10,3),
user_id bigint,
user_session string
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
   "separatorChar" = ",",
   "quoteChar" = "\"",
   "escapeChar" = "\\")
```

STORED AS TEXTFILE
LOCATION '/tmp/test-folder/'
TBLPROPERTIES ("skip.header.line.count" = "1");

```
hive> create table if not exists ecom (
    > event_time timestamp,
    > event_type string,
    > product_id string,
    > category_id string,
    > category_code string,
    > brand string,
    > price decimal(10,3),
    > user_id bigint,
    > user_session string
    > )
    > ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
    > WITH SERDEPROPERTIES (
    >     "separatorChar" = ",",
    >     "quoteChar" = "\"",
    >     "escapeChar" = "\\")
    > STORED AS TEXTFILE
    > LOCATION '/tmp/test-folder/'
    > TBLPROPERTIES ("skip.header.line.count" = "1");
OK
Time taken: 0.723 seconds
hive> show tables;
OK
ecom
Time taken: 0.049 seconds, Fetched: 1 row(s)
hive>
```

5. Show 5 rows from “ecom” table;

Select * from ecom limit 5;

```
hive> Select * from ecom limit 5;
OK
ecom.event_time ecom.event_type ecom.product_id ecom.category_id        ecom.cat
egory_code      ecom.brand      ecom.price      ecom.user_id    ecom.user_sessio
n
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    2
2.22    556138645       57ed222e-a54a-4907-9944-5a875c2d7f4f
2019-11-01 00:00:11 UTC cart    5876812 1487580010100293687             jessnail
3.16    564506666       186c1951-8052-4b37-adce-dd9644b1d5f7
2019-11-01 00:00:24 UTC remove_from_cart        5826182 14875800074830489000    3
.33     553329724       2067216c-31b5-455d-a1cc-af0575a34ffb
Time taken: 0.149 seconds, Fetched: 5 row(s)
hive>
```

6. Describe “ecom” table

describe ecom;

```
hive> describe ecom;
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: 0.737 seconds, Fetched: 9 row(s)
hive>
```

7. Count the number of rows inserted into the table.
   Select count(*) from ecom;

```
hive> select count(*) from ecom;
Query ID = hadoop_20210503142858_6a8518de-2731-42ad-8230-9e46584e551c
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1620050213727_0003)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      2          2        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 51.17 s
--------------------------------------------------------------------------------
OK
8738120
Time taken: 56.563 seconds, Fetched: 1 row(s)
hive>
```

8. Set the parameters to create the dynamic partition.

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

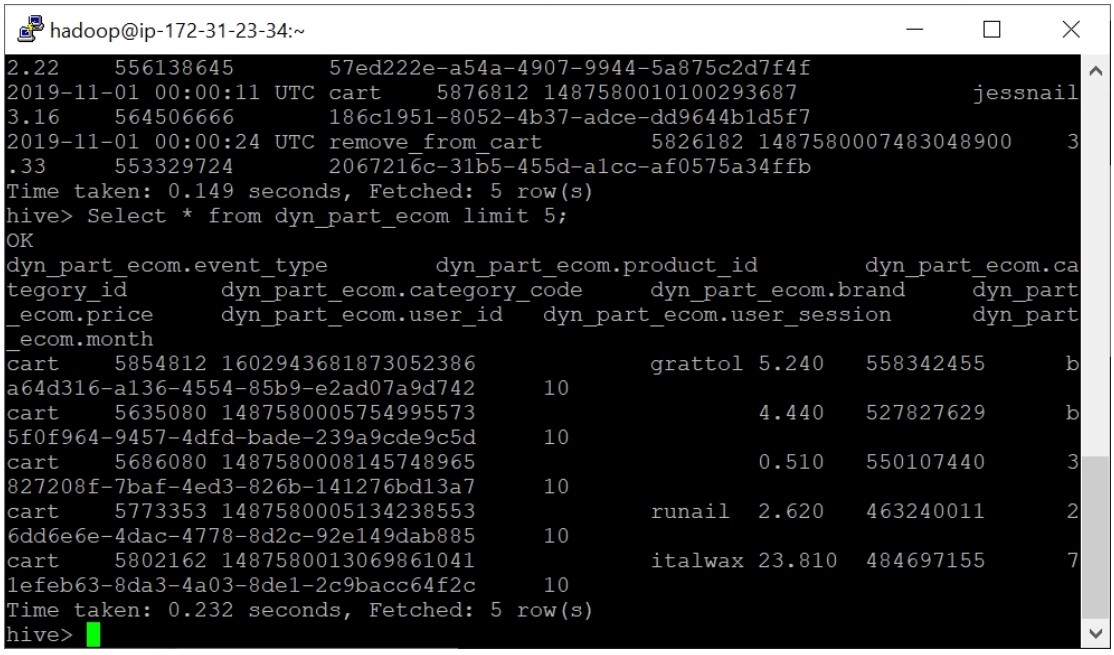
```
hive>   set hive.exec.dynamic.partition = true;
hive> set hive.exec.dynamic.partition.mode = nonstrict;
hive>
```

9. Create a partition table on “event month” and buckets based on “event type”.

   create table if not exists dyn_part_ecom
   (
   event_type string,
   product_id string,
   category_id string,
   category_code string,
   brand string,
   price decimal(10,3),
   user_id bigint,
   user_session string
)
   partitioned by (month int) clustered by (event_type) into 4 buckets;

```
hive> create table if not exists dyn_part_ecom
    > (
    > event_type string,
    > product_id string,
    > category_id string,
    > category_code string,
    > brand string,
    > price decimal(10,3),
    > user_id bigint,
    > user_session string
    > )
    > partitioned by (month int) clustered by (event_type) into 4 buckets;
OK
Time taken: 0.09 seconds
hive>
```

10. Show 5 rows from “dyn_part_ecom” table;
    Select * from dyn_part_ecom limit 5;



11. Describe dyn_part_ecom table
    describe dyn_part_ecom;

```
hive> describe dyn_part_ecom;
OK
event_type              string
product_id              string
category_id             string
category_code           string
brand                   string
price                   decimal(10,3)
user_id                 bigint
user_session            string
month                   int

# Partition Information
# col_name              data_type               comment

month                   int
Time taken: 0.354 seconds, Fetched: 14 row(s)
hive>
```

12. Populate the partitioned table based on initial data load completed in “ecom” table.

```
insert into table dyn_part_ecom partition (month)
select
event_type ,
product_id ,
category_id ,
category_code ,
brand ,
price ,
user_id ,
user_session,
month(event_time) as month
from ecom;
```

```
Map 1: 2/2      Reducer 2: 1(+2)/5
Map 1: 2/2      Reducer 2: 1(+3)/5
Map 1: 2/2      Reducer 2: 1(+3)/5
Map 1: 2/2      Reducer 2: 1(+3)/5
Map 1: 2/2      Reducer 2: 1(+3)/5
Map 1: 2/2      Reducer 2: 1(+3)/5
Map 1: 2/2      Reducer 2: 1(+3)/5
Map 1: 2/2      Reducer 2: 1(+3)/5
Map 1: 2/2      Reducer 2: 2(+2)/5
Map 1: 2/2      Reducer 2: 2(+3)/5
Map 1: 2/2      Reducer 2: 3(+2)/5
Map 1: 2/2      Reducer 2: 3(+2)/5
Map 1: 2/2      Reducer 2: 3(+2)/5
Map 1: 2/2      Reducer 2: 3(+2)/5
Map 1: 2/2      Reducer 2: 4(+1)/5
Map 1: 2/2      Reducer 2: 5/5
Loading data to table default.dyn_part_ecom partition (month=null)


        Time taken to load dynamic partitions: 0.378 seconds
        Time taken for adding to write entity : 0.002 seconds
OK
Time taken: 168.734 seconds
hive>
```

13. Confirm number of rows in original table (“ecom”) matches with number of rows inserted into partitioned table (“dyn_part_ecom”)

```
Select count(*) from dyn_part_ecom;
```

```
hive> Select count(*) from dyn_part_ecom;
OK
8738120
Time taken: 0.289 seconds, Fetched: 1 row(s)
hive>
```

14. Set the parameter to allow hive to perform Cartesian product

```
set hive.strict.checks.cartesian.product=false;
set hive.mapred.mode=nonstrict;
```

```
hive> set hive.strict.checks.cartesian.product=false;
hive> set hive.mapred.mode=nonstrict;
hive>
```

Assessment Questions:

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

```
select sum(price) as OCT_Revenue
from dyn_part_ecom
where month= 10 and event_type = 'purchase';
```

```
Map 1: 0/5      Reducer 2: 0/1
Map 1: 0/5      Reducer 2: 0/1
Map 1: 0/5      Reducer 2: 0/1
Map 1: 0(+1)/5  Reducer 2: 0/1
Map 1: 0(+3)/5  Reducer 2: 0/1
Map 1: 0(+3)/5  Reducer 2: 0/1
Map 1: 0(+3)/5  Reducer 2: 0/1
Map 1: 0(+3)/5  Reducer 2: 0/1
Map 1: 1(+3)/5  Reducer 2: 0/1
Map 1: 2(+2)/5  Reducer 2: 0/1
Map 1: 3(+2)/5  Reducer 2: 0/1
Map 1: 3(+2)/5  Reducer 2: 0(+1)/1
Map 1: 4(+1)/5  Reducer 2: 0(+1)/1
Map 1: 5/5      Reducer 2: 0(+1)/1
Map 1: 5/5      Reducer 2: 1/1
OK
oct_revenue
1211538.430
Time taken: 21.872 seconds, Fetched: 1 row(s)
hive>
```

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

```
select month, sum(price) Revenue
from dyn_part_ecom
where event_type = 'purchase'
group by month;
```

```
Map 1: 0/7      Reducer 2: 0/2
Map 1: 0/7      Reducer 2: 0/2
Map 1: 0(+2)/7  Reducer 2: 0/2
Map 1: 0(+3)/7  Reducer 2: 0/2
Map 1: 0(+3)/7  Reducer 2: 0/2
Map 1: 0(+3)/7  Reducer 2: 0/2
Map 1: 0(+3)/7  Reducer 2: 0/2
Map 1: 1(+2)/7  Reducer 2: 0/2
Map 1: 1(+3)/7  Reducer 2: 0/2
Map 1: 2(+3)/7  Reducer 2: 0/2
Map 1: 3(+2)/7  Reducer 2: 0/2
Map 1: 3(+3)/7  Reducer 2: 0/2
Map 1: 4(+3)/7  Reducer 2: 0/2
Map 1: 5(+2)/7  Reducer 2: 0/2
Map 1: 5(+2)/7  Reducer 2: 0(+1)/2
Map 1: 6(+1)/7  Reducer 2: 0(+2)/2
Map 1: 7/7      Reducer 2: 0(+2)/2
Map 1: 7/7      Reducer 2: 2/2
OK
month   revenue
10      1211538.430
11      1531016.900
Time taken: 26.745 seconds, Fetched: 2 row(s)
hive>
```

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

Before using mapjoin (query time taken = 33 secs)**:**

```
SELECT OCT_Revenue, NOV_Revenue, (NOV_Revenue - OCT_Revenue) as Revenue_Growth
From
(select sum(price) as OCT_Revenue
from dyn_part_ecom
where month= 10 and event_type = 'purchase') t1
FULL OUTER JOIN
(select sum(price) as NOV_Revenue
from dyn_part_ecom
where month= 11 and event_type = 'purchase') t2 ;
```

```
Map 1: 4(+1)/5  Map 4: 5/5      Reducer 2: 0/1  Reducer 3: 0/1  Reducer 5: 0(+1)
/1
Map 1: 4(+1)/5  Map 4: 5/5      Reducer 2: 0(+1)/1      Reducer 3: 0/1  Reducer
5: 0(+1)/1
Map 1: 4(+1)/5  Map 4: 5/5      Reducer 2: 0(+1)/1      Reducer 3: 0/1  Reducer
5: 1/1
Map 1: 5/5      Map 4: 5/5      Reducer 2: 0(+1)/1      Reducer 3: 0/1  Reducer
5: 1/1
Map 1: 5/5      Map 4: 5/5      Reducer 2: 1/1  Reducer 3: 0(+1)/1      Reducer
5: 1/1
Map 1: 5/5      Map 4: 5/5      Reducer 2: 1/1  Reducer 3: 1/1  Reducer 5: 1/1
OK
oct_revenue     nov_revenue     revenue_growth
1211538.430     1531016.900     319478.470
Time taken: 33.542 seconds, Fetched: 1 row(s)
hive>
```

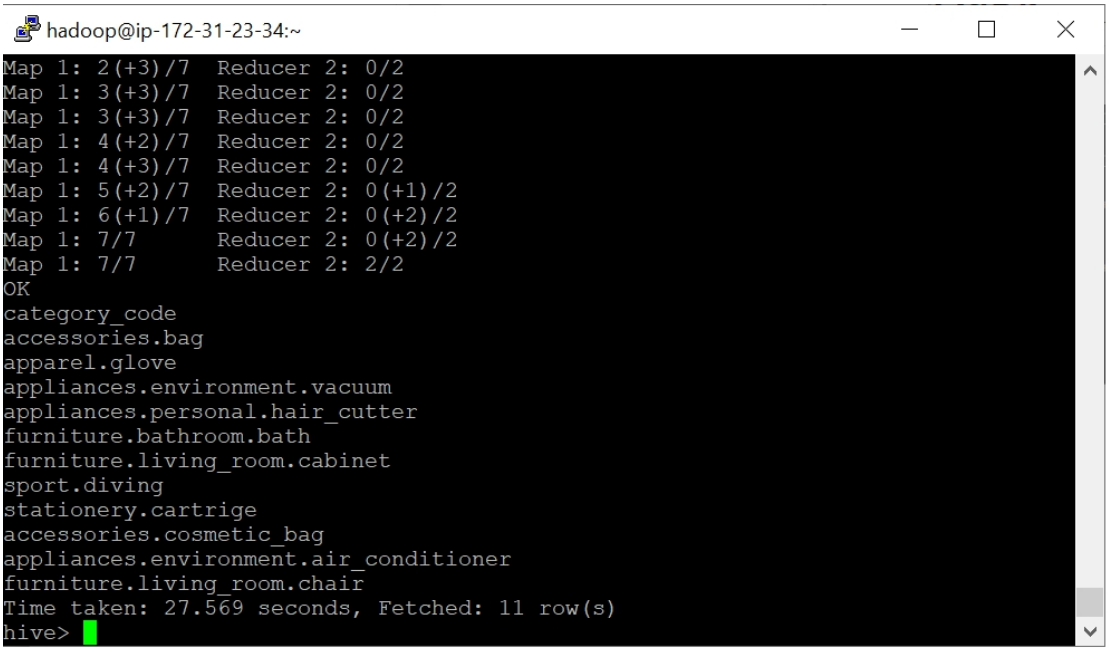
After using mapjoin (query time taken = 29 secs)**:**

```
SELECT /*+ MAPJOIN(OCT_Revenue) */ OCT_Revenue, NOV_Revenue, (NOV_Revenue -
OCT_Revenue) as Revenue_Growth
From
(select sum(price) as OCT_Revenue
from dyn_part_ecom
where month= 10 and event_type = 'purchase') t1
FULL OUTER JOIN
(select sum(price) as NOV_Revenue
from dyn_part_ecom
where month= 11 and event_type = 'purchase') t2 ;
```

```
Map 1: 4(+1)/5  Map 4: 5/5      Reducer 2: 0(+1)/1      Reducer 3: 0/1  Reducer
5: 0(+1)/1
Map 1: 4(+1)/5  Map 4: 5/5      Reducer 2: 0(+1)/1      Reducer 3: 0/1  Reducer
5: 1/1
Map 1: 5/5      Map 4: 5/5      Reducer 2: 1/1  Reducer 3: 0/1  Reducer 5: 1/1
Map 1: 5/5      Map 4: 5/5      Reducer 2: 1/1  Reducer 3: 0(+1)/1      Reducer
5: 1/1
Map 1: 5/5      Map 4: 5/5      Reducer 2: 1/1  Reducer 3: 1/1  Reducer 5: 1/1
OK
oct_revenue     nov_revenue     revenue_growth
1211538.430     1531016.900     319478.470
Time taken: 29.737 seconds, Fetched: 1 row(s)
hive>
```

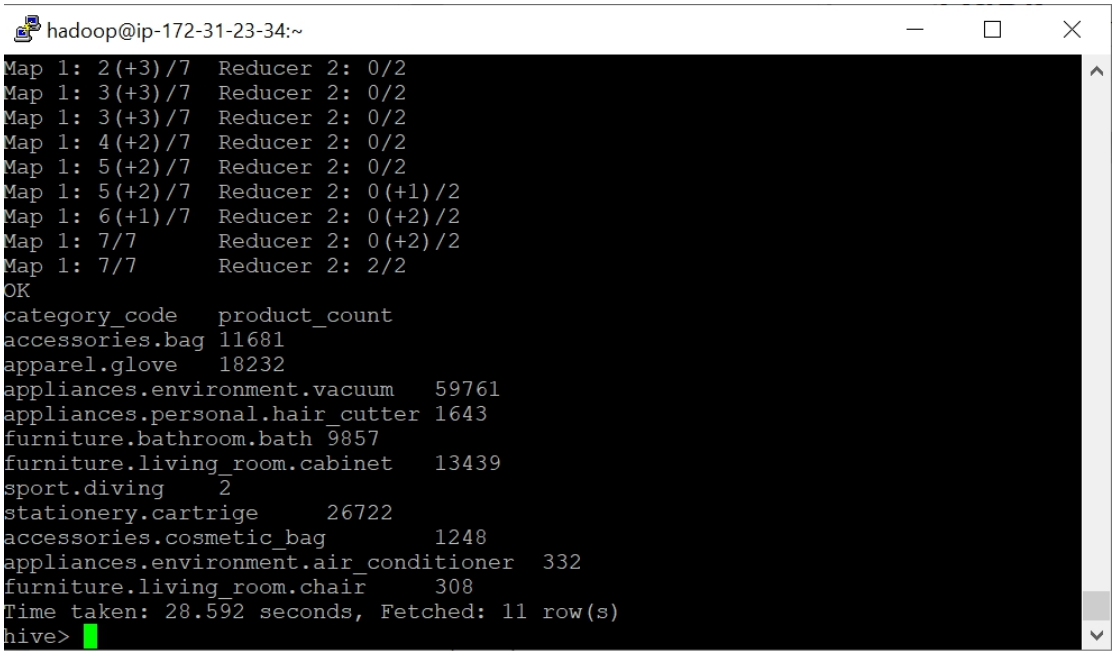
4. Find distinct categories of products. Categories with null category code can be ignored.

```
select distinct category_code
from dyn_part_ecom
where LENGTH(category_code) > 0 ;
```



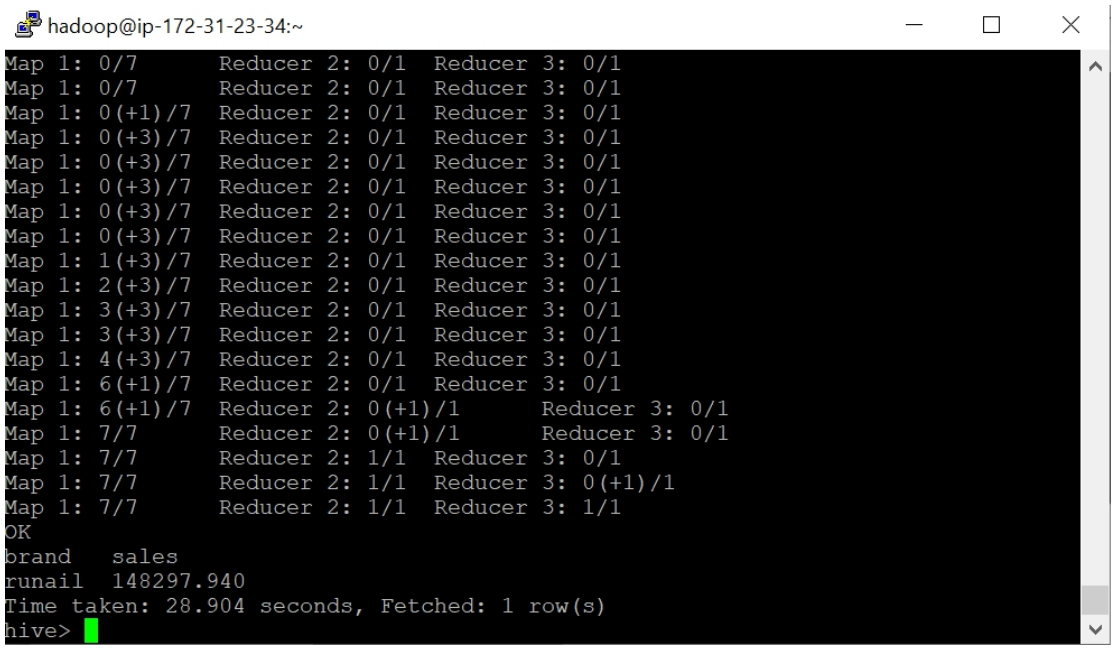
5. Find the total number of products available under each category.

```
select category_code, count(product_id) as product_count
from dyn_part_ecom
where LENGTH(category_code) > 0
group by category_code;
```



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

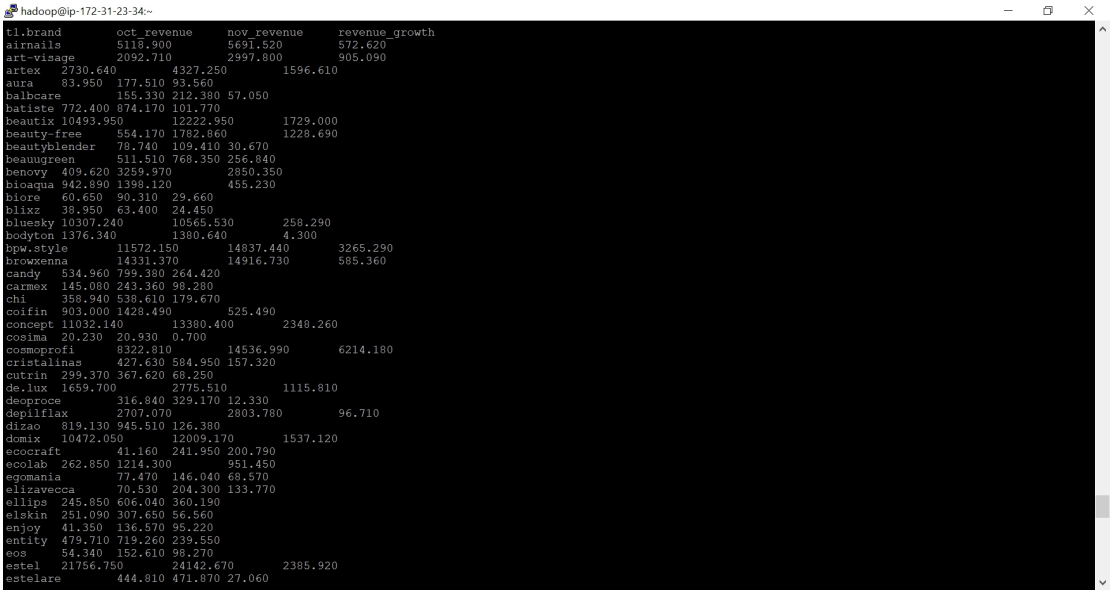
```
select brand, sum(price) as sales
from dyn_part_ecom
where LENGTH(brand) > 0
and event_type = 'purchase'
group by brand
order by sales desc limit 1;
```

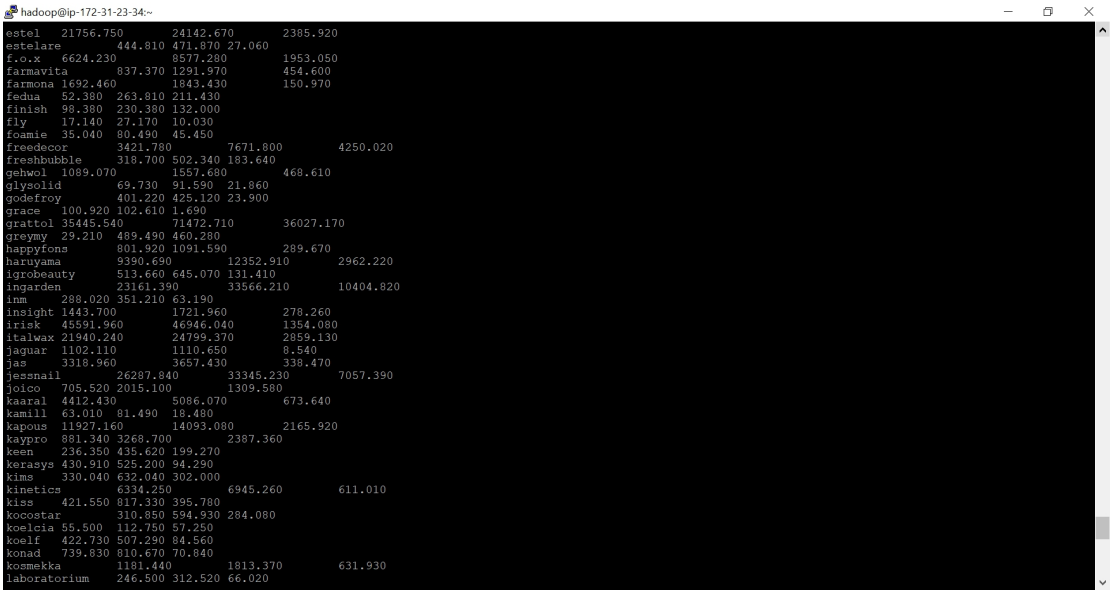


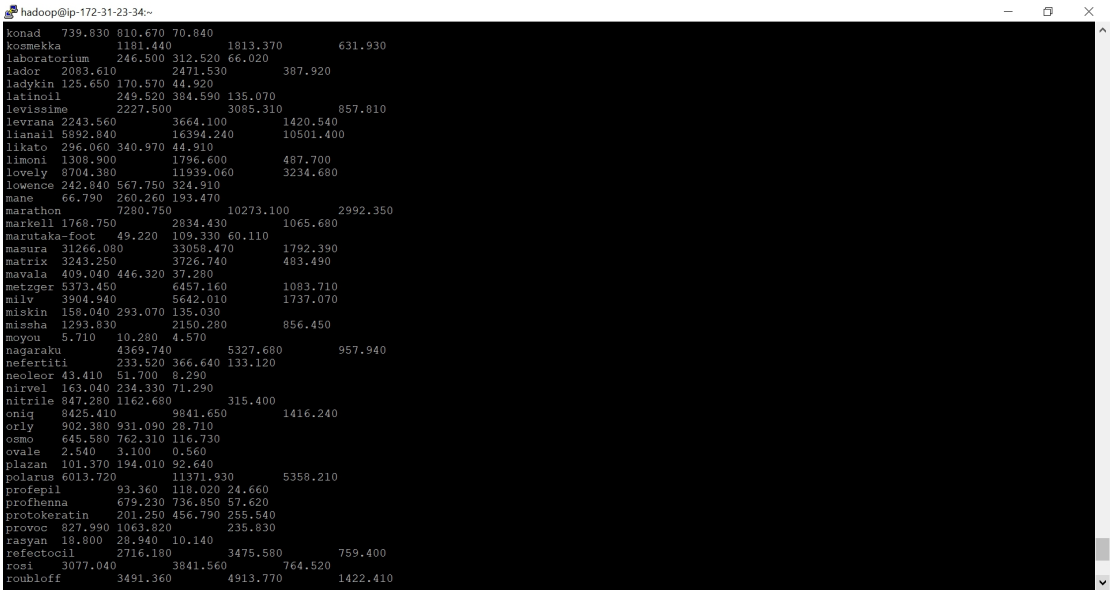
7. Which brands increased their sales from October to November?

```
SELECT t1.brand, OCT_Revenue, NOV_Revenue, (NOV_Revenue - OCT_Revenue) as
Revenue_Growth
From
(select brand, sum(price) as OCT_Revenue
```

```
from dyn_part_ecom
where month= 10 and event_type = 'purchase' and LENGTH(brand) > 0
group by brand) t1
LEFT JOIN
(select brand, sum(price) as NOV_Revenue
from dyn_part_ecom
where month= 11 and event_type = 'purchase' and LENGTH(brand) > 0 group by brand) t2
WHERE t1.brand = t2.brand
AND (NOV_Revenue - OCT_Revenue) > 0;
```








```
roubloff         3491.360        4913.770        1422.410
runail   71539.280       76758.660       5219.380
s.care   412.680 913.070 500.390
sanoto   157.140 1209.680        1052.540
severina         4775.880        6120.480        1344.600
shary    871.960 1176.490        304.530
shik     3341.200        4839.720        1498.520
skinity 8.880    12.440  3.560
skinlite         651.940 890.450 238.510
smart    4457.260        5902.140        1444.880
soleo    204.200 212.530 8.330
solomeya         1899.700        2685.800        786.100
sophin   1067.860        1515.520        447.660
staleks 8519.730         11875.610       3355.880
strong   29196.630       38671.270       9474.640
supertan         50.370  66.510  16.140
swarovski        1887.930        3043.160        1155.230
tertio   236.160 245.800 9.640
treaclemoon      163.370 181.490 18.120
trind    298.070 542.960 244.890
uno      35302.030       51039.750       15737.720
uskusi   5142.270        5690.310        548.040
veraclara        50.110  71.210  21.100
vilenta 197.600 231.210 33.610
yoko     8756.910        11707.880       2950.970
yu-r     271.410 673.710 402.300
zeitun   708.660 2009.630        1300.970
Time taken: 31.957 seconds, Fetched: 152 row(s)
hive>
```

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.

```
select user_id, sum(price) total_spend
from dyn_part_ecom
where event_type = 'purchase'
group by user_id
order by total_spend desc
limit 10;
```

```
--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      7          7        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: 29.28 s
--------------------------------------------------------------------------------
OK
user_id total_spend
557790271       2715.870
150318419       1645.970
562167663       1352.850
531900924       1329.450
557850743       1295.480
522130011       1185.390
561592095       1109.700
431950134       1097.590
566576008       1056.360
521347209       1040.910
Time taken: 29.872 seconds, Fetched: 10 row(s)
hive>
```