DS8003 – Final Project Report

(By:- Akshdeep Kaler: 501009212, Li Gong: 501077874)

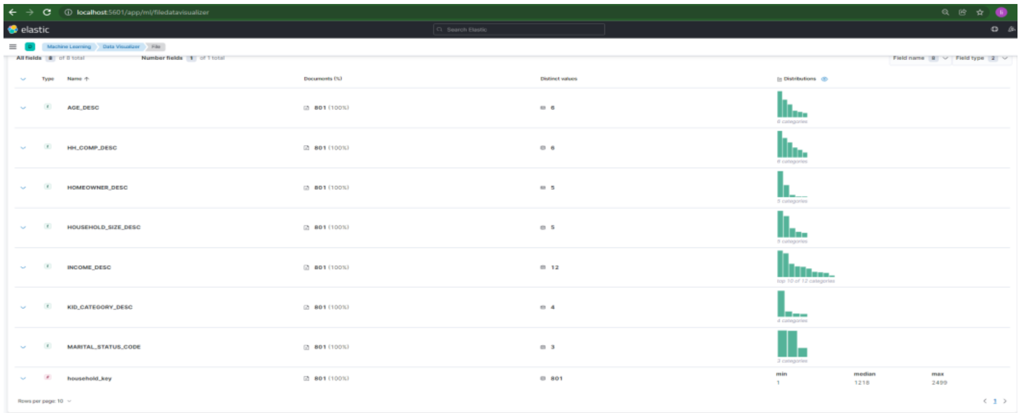
Problem Definition:

The retail stores sell products to customers and they would like to retain their customers and make them buy more products for increased profitability. For this purpose, various marketing campaigns are also run, in addition to improved service delivery and better pricing. To know if the employed strategy is working, it is important to know that what elements of the strategy steer the business in right directions and which of them are not producing the desired output.

The Dataset:

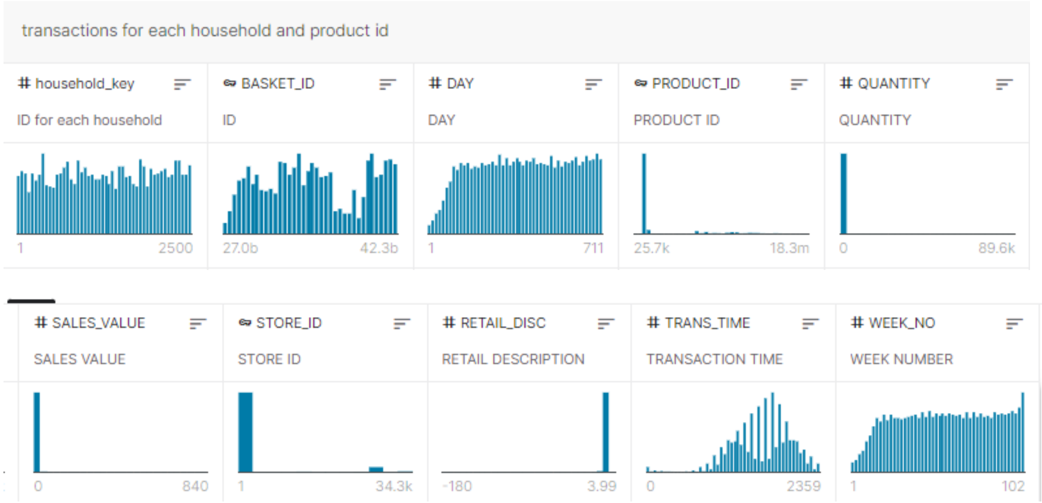
We use the open-sourced dataset, titled The Complete Journey, made available by dunhumby. The dataset contains household level, anonymized, transaction data (2500 households) including the demographics and marketing campaigns (30 campaigns). The transactional data include over 90 thousand products categorized in 44 departments. The dataset is available at https://www.kaggle.com/frtgnn/dunnhumby-the-complete-journey.The following is the summary of each datasets:

1. hh_demographic: The table contains demographic information for a portion of households.



Type	Name ↑	Documents (%)	Distinct values	Distributions
t	AGE_DESC	801 (100%)	6	6 categories
t	HH_COMP_DESC	801 (100%)	6	6 categories
t	HOMEOWNER_DESC	801 (100%)	5	5 categories
t	HOUSEHOLD_SIZE_DESC	801 (100%)	5	5 categories
t	INCOME_DESC	801 (100%)	12	top 10 of 12 categories
t	KID_CATEGORY_DESC	801 (100%)	4	4 categories
t	MARITAL_STATUS_CODE	801 (100%)	3	3 categories
#	household_key	801 (100%)	801	min 1, median 1218, max 2499

2. transaction_data: This contains all products purchased by households.



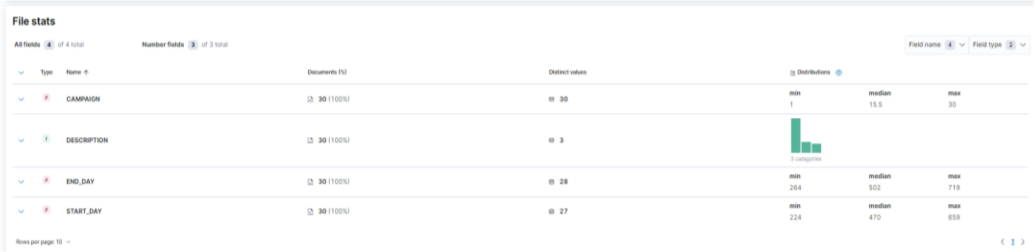
transactions for each household and product id

Column	Description	Min	Max
# household_key	ID for each household	1	2500
⇔ BASKET_ID	ID	27.0b	42.3b
# DAY	DAY	1	711
⇔ PRODUCT_ID	PRODUCT ID	25.7k	18.3m
# QUANTITY	QUANTITY	0	89.6k
# SALES_VALUE	SALES VALUE	0	840
⇔ STORE_ID	STORE ID	1	34.3k
# RETAIL_DISC	RETAIL DESCRIPTION	-180	3.99
# TRANS_TIME	TRANSACTION TIME	0	2359
# WEEK_NO	WEEK NUMBER	1	102

3. campaign_table **(Not Used During Project)**

   This table lists the campaigns received by each household in the study.

4. campaign_desc: This table gives the length of time for which a campaign runs. Any coupons received as part of a campaign are valid within dates contained in this table.

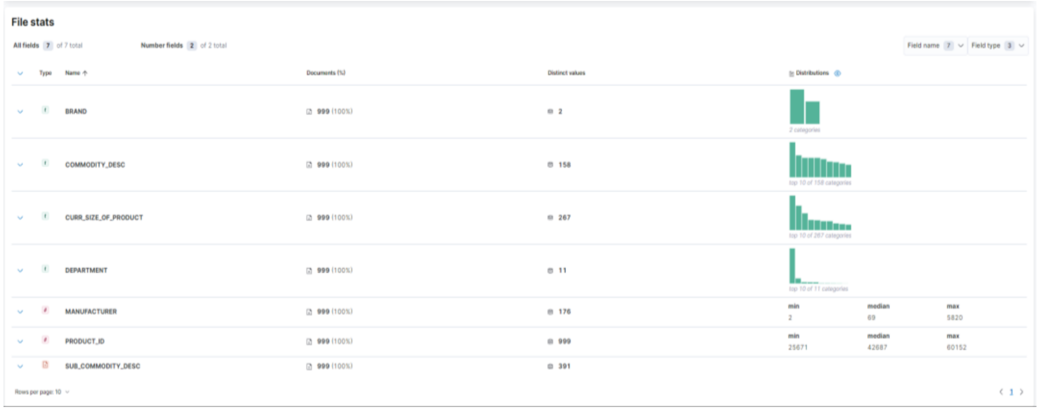


File stats

All fields 4 of 4 total   Number fields 3 of 3 total

Type	Name ↑	Documents (%)	Distinct values	Distributions
#	CAMPAIGN	30 (100%)	30	min 1 median 15.5 max 30
t	DESCRIPTION	30 (100%)	3	3 categories
#	END_DAY	30 (100%)	28	min 264 median 502 max 719
#	START_DAY	30 (100%)	27	min 224 median 470 max 659

5. product: This table contains information on each product sold such as type of product, national or private label and a brand identifier.

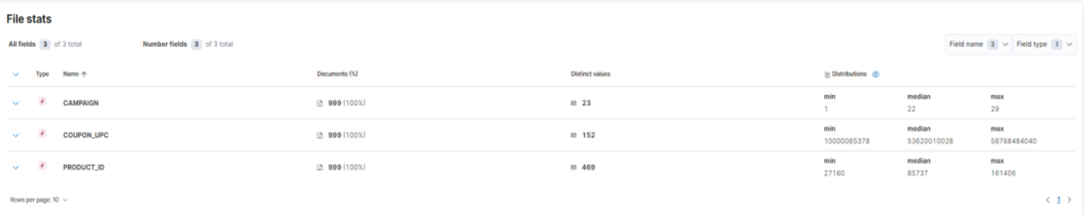


File stats

All fields 7 of 7 total   Number fields 2 of 2 total

Type	Name ↑	Documents (%)	Distinct values	Distributions
t	BRAND	999 (100%)	2	2 categories
t	COMMODITY_DESC	999 (100%)	158	top 10 of 158 categories
t	CURR_SIZE_OF_PRODUCT	999 (100%)	267	top 10 of 267 categories
t	DEPARTMENT	999 (100%)	11	top 10 of 11 categories
#	MANUFACTURER	999 (100%)	176	min 2 median 69 max 5820
#	PRODUCT_ID	999 (100%)	999	min 25671 median 42687 max 60152
🗋	SUB_COMMODITY_DESC	999 (100%)	391	

6. coupon: This table list all the coupons sent to customers as part of a campaign, as well as the products for which each coupon is redeemable.

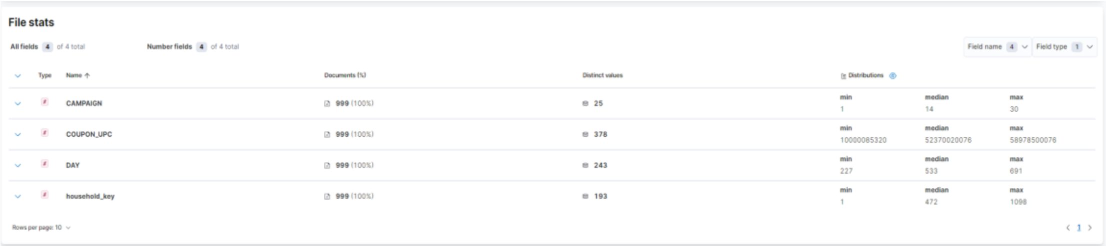


File stats

All fields 3 of 3 total   Number fields 3 of 3 total

Type	Name ↑	Documents (%)	Distinct values	Distributions
#	CAMPAIGN	999 (100%)	23	min 1 median 22 max 29
#	COUPON_UPC	999 (100%)	152	min 10000085378 median 53620010028 max 58768484040
#	PRODUCT_ID	999 (100%)	469	min 27160 median 85737 max 161406

7. coupon_redempt: This table identifies the coupons that each household redeemed.



File stats

All fields 4 of 4 total   Number fields 4 of 4 total

Type	Name ↑	Documents (%)	Distinct values	Distributions
#	CAMPAIGN	999 (100%)	25	min 1 median 14 max 30
#	COUPON_UPC	999 (100%)	378	min 10000085320 median 52370020076 max 58978500076
#	DAY	999 (100%)	243	min 227 median 533 max 691
#	household_key	999 (100%)	193	min 1 median 472 max 1098

8. casual_data **(Not Used During Project)**

This table signifies whether a given product was featured in the weekly mailer or was part of an in-store display (other than regular product placement).

Work Distribution:

To get the solution to the problem mentioned above, the questions were divided among the group members:

1. Akshdeep Kaler:
    - What are the characteristics of customers whose spending at the store is increasing?
    - What are the categories of the products that are seeing increased/decreased sales?
    - What are the most profitable categories of the products over time?
    - Which day has the highest sales?
2. Li Gong:
    - Are the marketing campaigns effective?
    - Which of the marketing campaigns was the most successful one?
    - What are the characteristics of customers who were attracted by each marketing campaign?

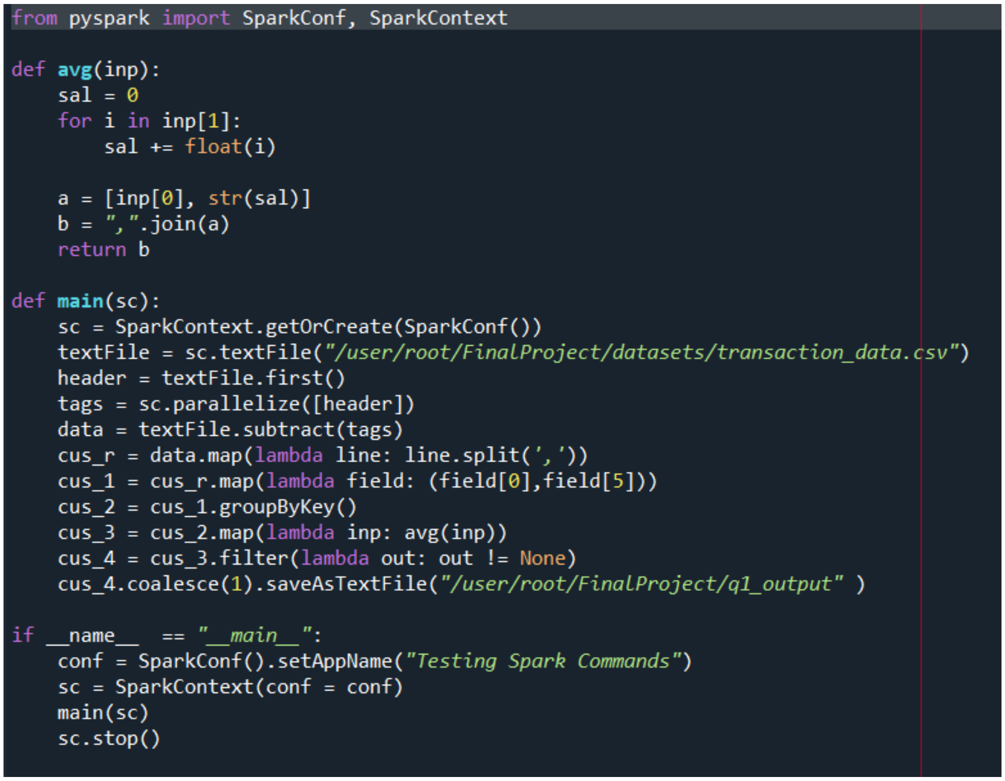
Q1. What are the characteristics of customers whose spending at the store is increasing?

Dataset : **transaction_data.csv** and **hh_demographic.csv** are used.

The tools used for extracting the information are:

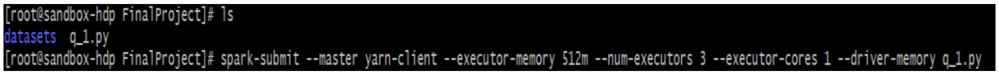
- Pyspark
- Pyspark SQL
- Hadoop Distributed File System (HDFS)

In the data file ‘transaction_data.csv,’ each customer or household details of spending is given. For the required question, we are interested in household_key and sales attributes/columns. The Pyspark API is used to extract the columns and store them on the Hadoop Distributed File System. The following code extracts the total sales according to each ‘household_key’ and stores in the HDFS, where it can be accessed for further analysis.



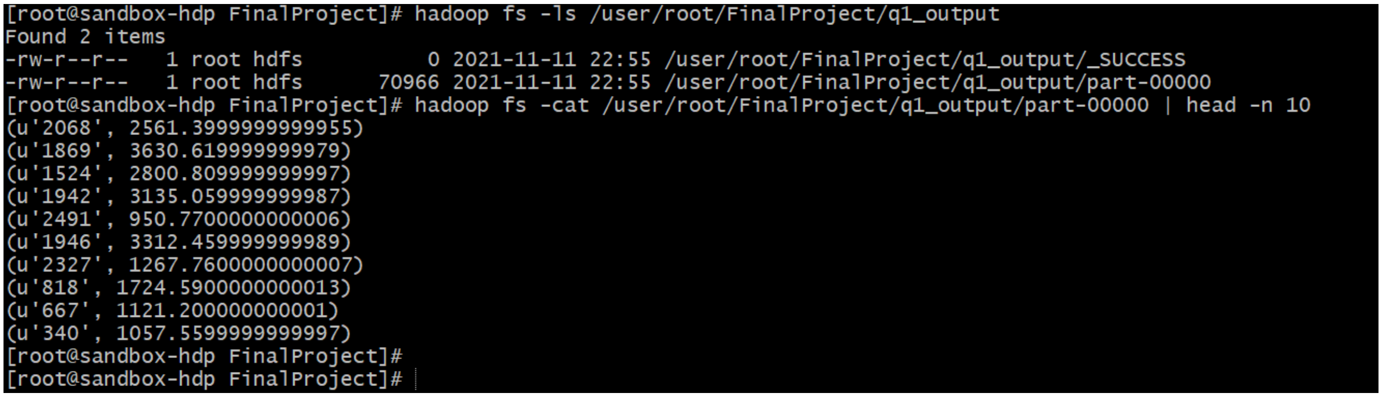
```python
from pyspark import SparkConf, SparkContext

def avg(inp):
    sal = 0
    for i in inp[1]:
        sal += float(i)

    a = [inp[0], str(sal)]
    b = ",".join(a)
    return b

def main(sc):
    sc = SparkContext.getOrCreate(SparkConf())
    textFile = sc.textFile("/user/root/FinalProject/datasets/transaction_data.csv")
    header = textFile.first()
    tags = sc.parallelize([header])
    data = textFile.subtract(tags)
    cus_r = data.map(lambda line: line.split(','))
    cus_1 = cus_r.map(lambda field: (field[0],field[5]))
    cus_2 = cus_1.groupByKey()
    cus_3 = cus_2.map(lambda inp: avg(inp))
    cus_4 = cus_3.filter(lambda out: out != None)
    cus_4.coalesce(1).saveAsTextFile("/user/root/FinalProject/q1_output" )

if __name__  == "__main__":
    conf = SparkConf().setAppName("Testing Spark Commands")
    sc = SparkContext(conf = conf)
    main(sc)
    sc.stop()
```

Execution of ‘q_1.py’ file in the command shell.



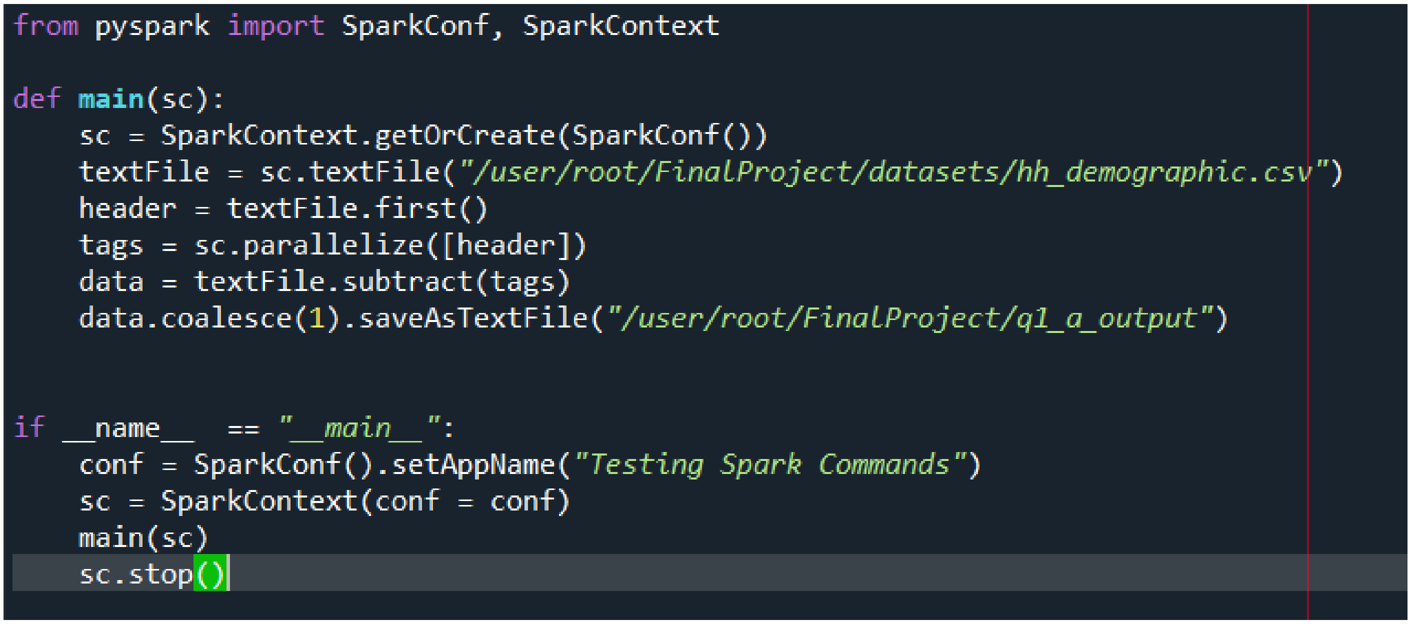
```
[root@sandbox-hdp FinalProject]# ls
datasets  q_1.py
[root@sandbox-hdp FinalProject]# spark-submit --master yarn-client --executor-memory 512m --num-executors 3 --executor-cores 1 --driver-memory q_1.py
```

Preview of extracted file in a format where the first tuple is household key and the second is total money spent in-store during two-year data.



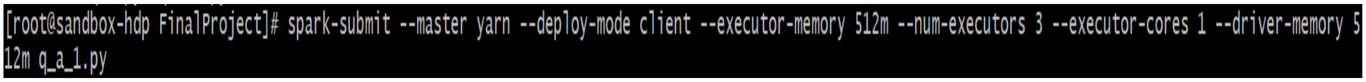
```
[root@sandbox-hdp FinalProject]# hadoop fs -ls /user/root/FinalProject/q1_output
Found 2 items
-rw-r--r--   1 root hdfs          0 2021-11-11 22:55 /user/root/FinalProject/q1_output/_SUCCESS
-rw-r--r--   1 root hdfs      70966 2021-11-11 22:55 /user/root/FinalProject/q1_output/part-00000
[root@sandbox-hdp FinalProject]# hadoop fs -cat /user/root/FinalProject/q1_output/part-00000 | head -n 10
(u'2068', 2561.3999999999955)
(u'1869', 3630.619999999979)
(u'1524', 2800.809999999997)
(u'1942', 3135.059999999987)
(u'2491', 950.7700000000006)
(u'1946', 3312.459999999989)
(u'2327', 1267.7600000000007)
(u'818', 1724.5900000000013)
(u'667', 1121.200000000001)
(u'340', 1057.5599999999997)
[root@sandbox-hdp FinalProject]#
[root@sandbox-hdp FinalProject]#
```

The avoid the first row of the data file ‘hh_demographic.csv’ to be included during the Pyspark SQL implementation. Pyspark API in python file is used to remove the first row or header row and store it in HDFS for further processing.



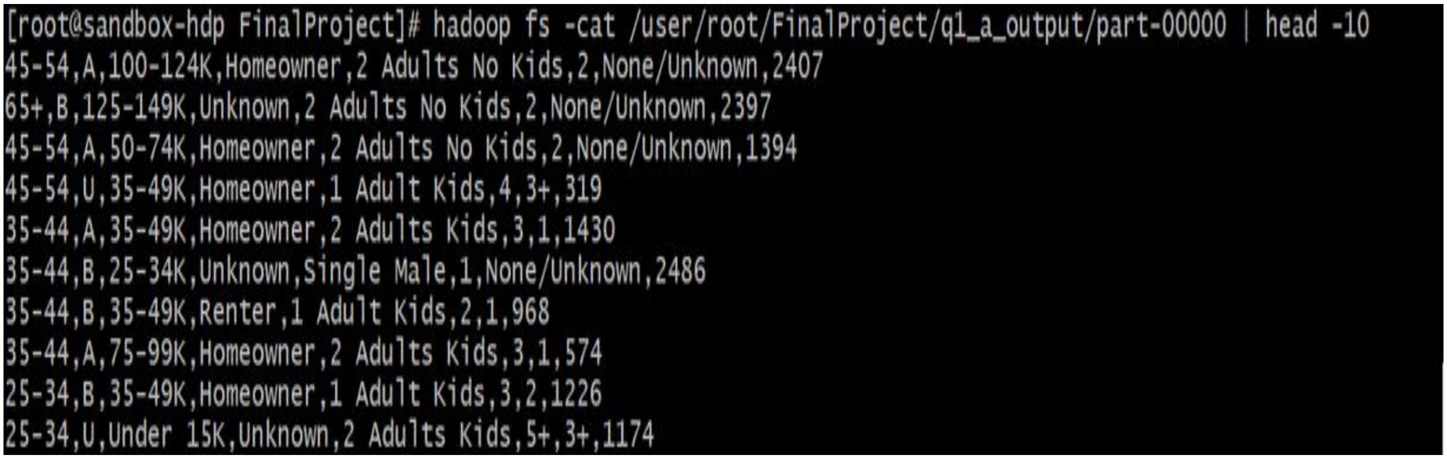
```python
from pyspark import SparkConf, SparkContext

def main(sc):
    sc = SparkContext.getOrCreate(SparkConf())
    textFile = sc.textFile("/user/root/FinalProject/datasets/hh_demographic.csv")
    header = textFile.first()
    tags = sc.parallelize([header])
    data = textFile.subtract(tags)
    data.coalesce(1).saveAsTextFile("/user/root/FinalProject/q1_a_output")


if __name__  == "__main__":
    conf = SparkConf().setAppName("Testing Spark Commands")
    sc = SparkContext(conf = conf)
    main(sc)
    sc.stop()
```

Execution of above-written python code file ‘q_a_1.py’ in the command shell.



```
[root@sandbox-hdp FinalProject]# spark-submit --master yarn --deploy-mode client --executor-memory 512m --num-executors 3 --executor-cores 1 --driver-memory 5
12m q_a_1.py
```

The output of file ‘q_a_1.py’ is stored on the Hadoop file distribution system. To check the output, the following commands are used on the command shell.



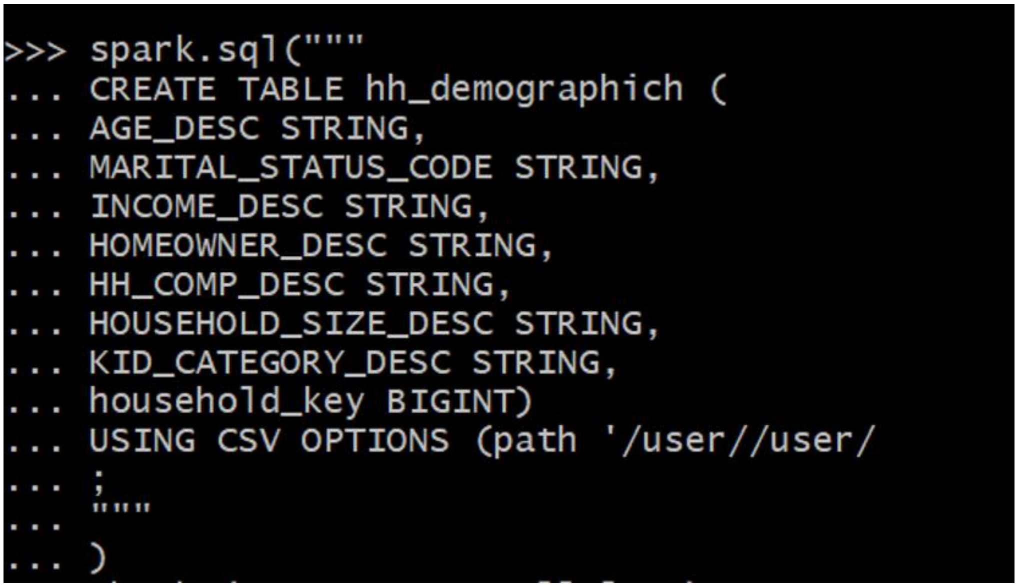
```
[root@sandbox-hdp FinalProject]# hadoop fs -cat /user/root/FinalProject/q1_a_output/part-00000 | head -10
45-54,A,100-124K,Homeowner,2 Adults No Kids,2,None/Unknown,2407
65+,B,125-149K,Unknown,2 Adults No Kids,2,None/Unknown,2397
45-54,A,50-74K,Homeowner,2 Adults No Kids,2,None/Unknown,1394
45-54,U,35-49K,Homeowner,1 Adult Kids,4,3+,319
35-44,A,35-49K,Homeowner,2 Adults Kids,3,1,1430
35-44,B,25-34K,Unknown,Single Male,1,None/Unknown,2486
35-44,B,35-49K,Renter,1 Adult Kids,2,1,968
35-44,A,75-99K,Homeowner,2 Adults Kids,3,1,574
25-34,B,35-49K,Homeowner,1 Adult Kids,3,2,1226
25-34,U,Under 15K,Unknown,2 Adults Kids,5+,3+,1174
```

The two outputs from the files ‘q_1.py’ and ‘q_a_1.py’ are stored on the Hadoop files Distributed system. Further Psypark SQL tool is used for extracting the information of spending of each household/customer. The first two tables are created from the extracted files, and then SQL commands are used to inner join the two tables and extract the information in the end. Following are the procedure followed to get information about the customers.

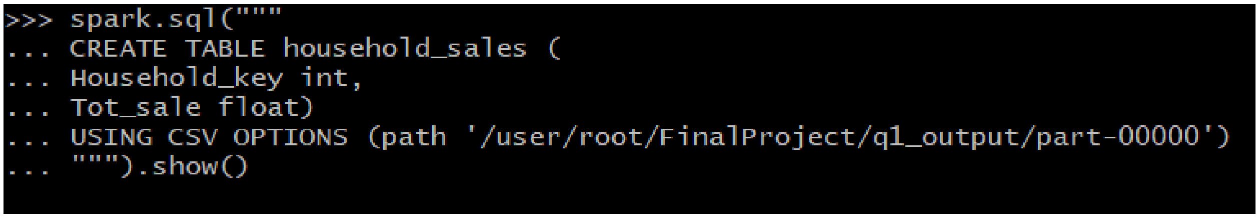
Run the Pyspark on the command shell.

Following commands are used to create a new table ‘hh_demographic’ and ‘house_holdsales’.

The output paths from ‘q_1.py’ and ‘q_a_1.py’ are used to generate the tables in Pyspark SQL with the help of the following codes:

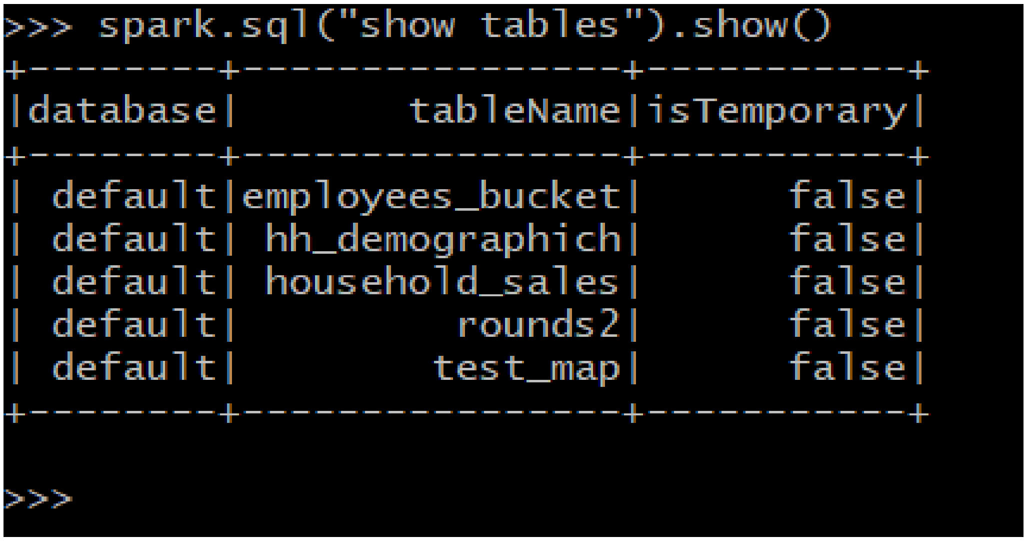


```
>>> spark.sql("""
... CREATE TABLE hh_demographich (
... AGE_DESC STRING,
... MARITAL_STATUS_CODE STRING,
... INCOME_DESC STRING,
... HOMEOWNER_DESC STRING,
... HH_COMP_DESC STRING,
... HOUSEHOLD_SIZE_DESC STRING,
... KID_CATEGORY_DESC STRING,
... household_key BIGINT)
... USING CSV OPTIONS (path '/user//user/
... ;
... """
... )
```



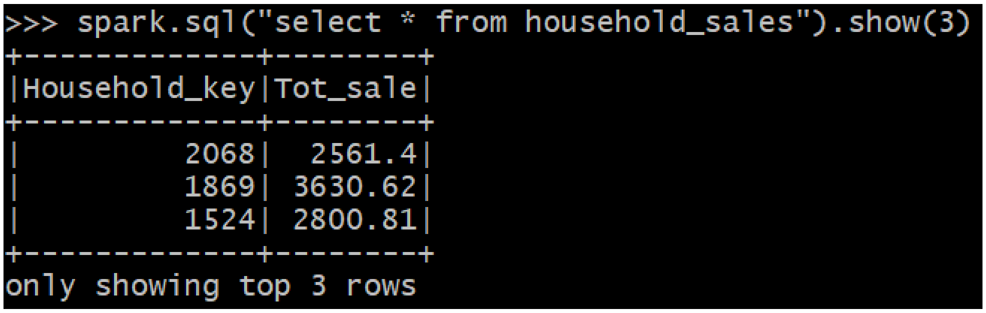
```
>>> spark.sql("""
... CREATE TABLE household_sales (
... Household_key int,
... Tot_sale float)
... USING CSV OPTIONS (path '/user/root/FinalProject/q1_output/part-00000')
... """).show()
```

The generated tables are stored in the ‘default’ database.

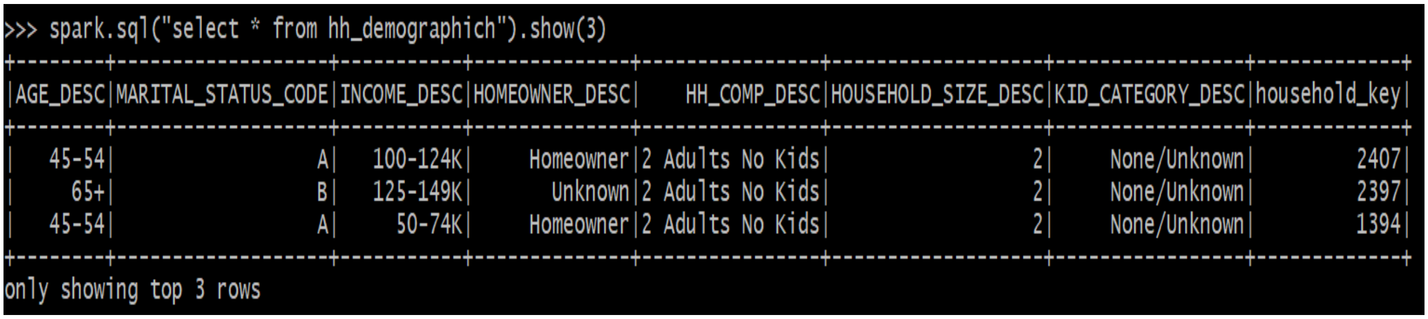


```
>>> spark.sql("show tables").show()
+--------+----------------+-----------+
|database|       tableName|isTemporary|
+--------+----------------+-----------+
| default|employees_bucket|      false|
| default| hh_demographich|      false|
| default| household_sales|      false|
| default|         rounds2|      false|
| default|        test_map|      false|
+--------+----------------+-----------+


>>>
```

The top 3 rows of each table.

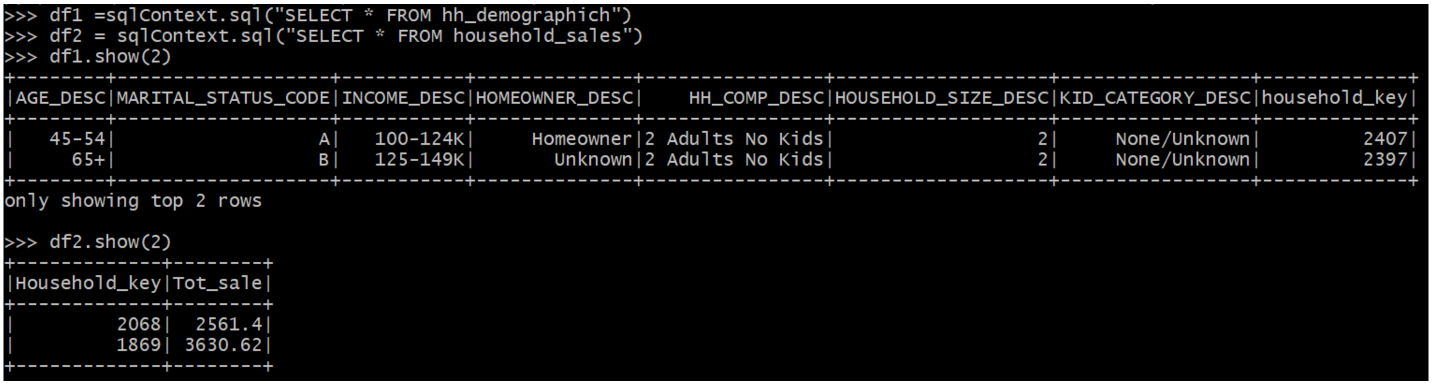


```
>>> spark.sql("select * from household_sales").show(3)
+-------------+--------+
|Household_key|Tot_sale|
+-------------+--------+
|         2068|  2561.4|
|         1869| 3630.62|
|         1524| 2800.81|
+-------------+--------+
only showing top 3 rows
```



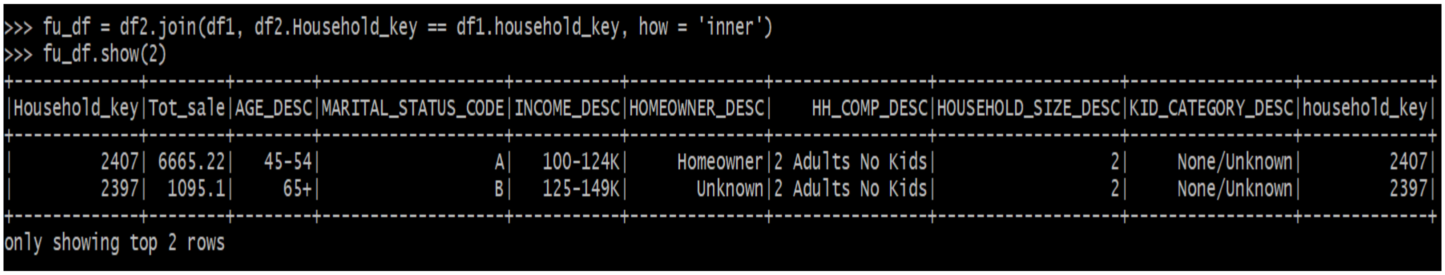
```
>>> spark.sql("select * from hh_demographich").show(3)
+--------+-------------------+-----------+--------------+----------------+-------------------+-----------------+-------------+
|AGE_DESC|MARITAL_STATUS_CODE|INCOME_DESC|HOMEOWNER_DESC|    HH_COMP_DESC|HOUSEHOLD_SIZE_DESC|KID_CATEGORY_DESC|household_key|
+--------+-------------------+-----------+--------------+----------------+-------------------+-----------------+-------------+
|   45-54|                  A|   100-124K|     Homeowner|2 Adults No Kids|                  2|     None/Unknown|         2407|
|     65+|                  B|   125-149K|       Unknown|2 Adults No Kids|                  2|     None/Unknown|         2397|
|   45-54|                  A|     50-74K|     Homeowner|2 Adults No Kids|                  2|     None/Unknown|         1394|
+--------+-------------------+-----------+--------------+----------------+-------------------+-----------------+-------------+
only showing top 3 rows
```

DataFrames are generated from the two tables and stored under separate variables.



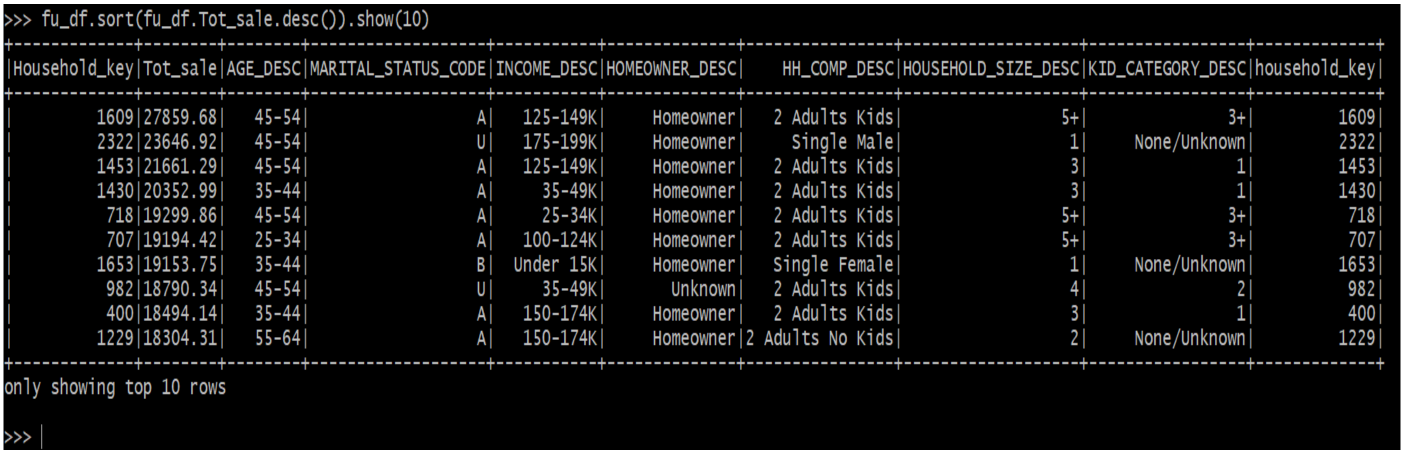
```
>>> df1 =sqlContext.sql("SELECT * FROM hh_demographich")
>>> df2 = sqlContext.sql("SELECT * FROM household_sales")
>>> df1.show(2)
+--------+-------------------+-----------+--------------+----------------+-------------------+-----------------+-------------+
|AGE_DESC|MARITAL_STATUS_CODE|INCOME_DESC|HOMEOWNER_DESC|    HH_COMP_DESC|HOUSEHOLD_SIZE_DESC|KID_CATEGORY_DESC|household_key|
+--------+-------------------+-----------+--------------+----------------+-------------------+-----------------+-------------+
|   45-54|                  A|   100-124K|     Homeowner|2 Adults No Kids|                  2|     None/Unknown|         2407|
|     65+|                  B|   125-149K|       Unknown|2 Adults No Kids|                  2|     None/Unknown|         2397|
+--------+-------------------+-----------+--------------+----------------+-------------------+-----------------+-------------+
only showing top 2 rows

>>> df2.show(2)
+-------------+--------+
|Household_key|Tot_sale|
+-------------+--------+
|         2068|  2561.4|
|         1869| 3630.62|
+-------------+--------+
```

Pyspark join function combines the two tables on the common column of Household_key and stores them in the final data frame ‘fu_df.’



```
>>> fu_df = df2.join(df1, df2.Household_key == df1.household_key, how = 'inner')
>>> fu_df.show(2)
+-------------+--------+--------+-------------------+-----------+--------------+----------------+-------------------+-----------------+-------------+
|Household_key|Tot_sale|AGE_DESC|MARITAL_STATUS_CODE|INCOME_DESC|HOMEOWNER_DESC|    HH_COMP_DESC|HOUSEHOLD_SIZE_DESC|KID_CATEGORY_DESC|household_key|
+-------------+--------+--------+-------------------+-----------+--------------+----------------+-------------------+-----------------+-------------+
|         2407| 6665.22|   45-54|                  A|   100-124K|     Homeowner|2 Adults No Kids|                  2|     None/Unknown|         2407|
|         2397|  1095.1|     65+|                  B|   125-149K|       Unknown|2 Adults No Kids|                  2|     None/Unknown|         2397|
+-------------+--------+--------+-------------------+-----------+--------------+----------------+-------------------+-----------------+-------------+
only showing top 2 rows
```

The columns in the final data frame are sorted according to ‘Tot_sale’ or total sales done by each customer/ household in descending order. The top 10 customer demographic details are extracted.



```
>>> fu_df.sort(fu_df.Tot_sale.desc()).show(10)
+-------------+--------+--------+-------------------+-----------+--------------+----------------+-------------------+-----------------+-------------+
|Household_key|Tot_sale|AGE_DESC|MARITAL_STATUS_CODE|INCOME_DESC|HOMEOWNER_DESC|    HH_COMP_DESC|HOUSEHOLD_SIZE_DESC|KID_CATEGORY_DESC|household_key|
+-------------+--------+--------+-------------------+-----------+--------------+----------------+-------------------+-----------------+-------------+
|         1609|27859.68|   45-54|                  A|   125-149K|     Homeowner|   2 Adults Kids|                 5+|               3+|         1609|
|         2322|23646.92|   45-54|                  U|   175-199K|     Homeowner|     Single Male|                  1|     None/Unknown|         2322|
|         1453|21661.29|   45-54|                  A|   125-149K|     Homeowner|   2 Adults Kids|                  3|                1|         1453|
|         1430|20352.99|   35-44|                  A|     35-49K|     Homeowner|   2 Adults Kids|                  3|                1|         1430|
|          718|19299.86|   45-54|                  A|     25-34K|     Homeowner|   2 Adults Kids|                 5+|               3+|          718|
|          707|19194.42|   25-34|                  A|   100-124K|     Homeowner|   2 Adults Kids|                 5+|               3+|          707|
|         1653|19153.75|   35-44|                  B|  Under 15K|     Homeowner|   Single Female|                  1|     None/Unknown|         1653|
|          982|18790.34|   45-54|                  U|     35-49K|       Unknown|   2 Adults Kids|                  4|                2|          982|
|          400|18494.14|   35-44|                  A|   150-174K|     Homeowner|   2 Adults Kids|                  3|                1|          400|
|         1229|18304.31|   55-64|                  A|   150-174K|     Homeowner|2 Adults No Kids|                  2|     None/Unknown|         1229|
+-------------+--------+--------+-------------------+-----------+--------------+----------------+-------------------+-----------------+-------------+
only showing top 10 rows

>>> 
```

The final output concluded that among the top 10 customers whose spending is increasing are 45-54 years of age group. Almost everyone is Homeowner and mainly earns above 100K dollars annually.

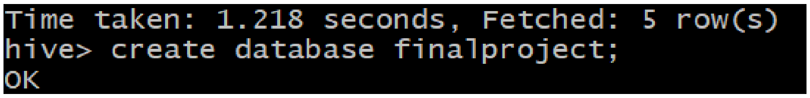
Q2. What are the categories of the products that are seeing increased/ decreased sales?

The tools used to answer the question are:

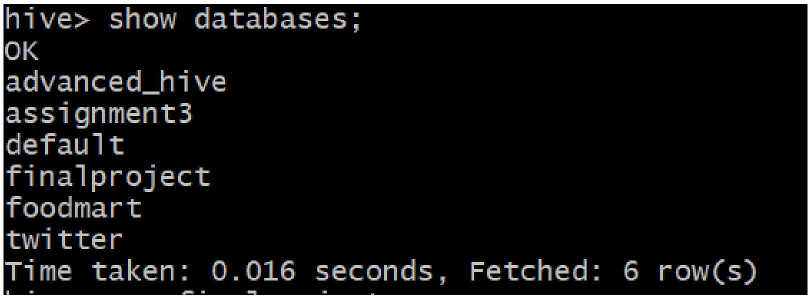
- Hive (for querying)
- Apache Hadoop for storage

The data files used for extracting the information are 'transaction_data.csv' and 'coupon.csv'. First data is stored at Hadoop Distributed File System. Hive is opened by passing command 'hive' on command shell. Then following procedure is followed:

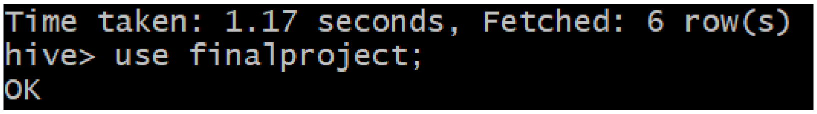
1. Create Database 'finalproject' to store generated tables.



```
Time taken: 1.218 seconds, Fetched: 5 row(s)
hive> create database finalproject;
OK
```

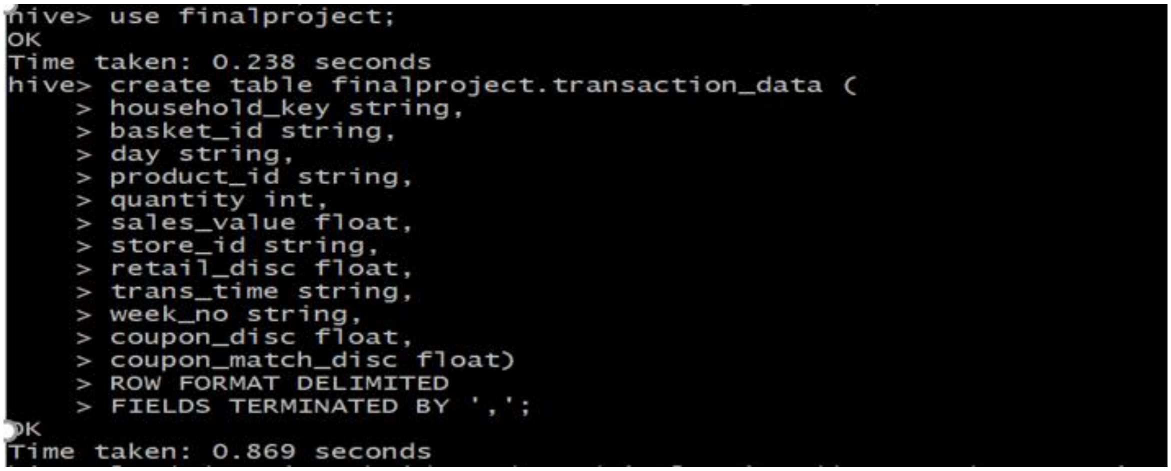


```
hive> show databases;
OK
advanced_hive
assignment3
default
finalproject
foodmart
twitter
Time taken: 0.016 seconds, Fetched: 6 row(s)
```

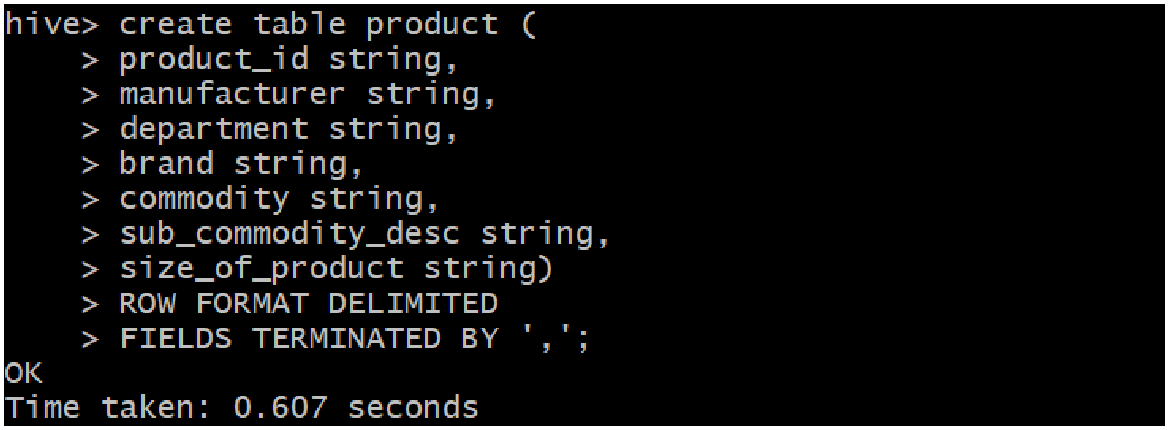


```
Time taken: 1.17 seconds, Fetched: 6 row(s)
hive> use finalproject;
OK
```

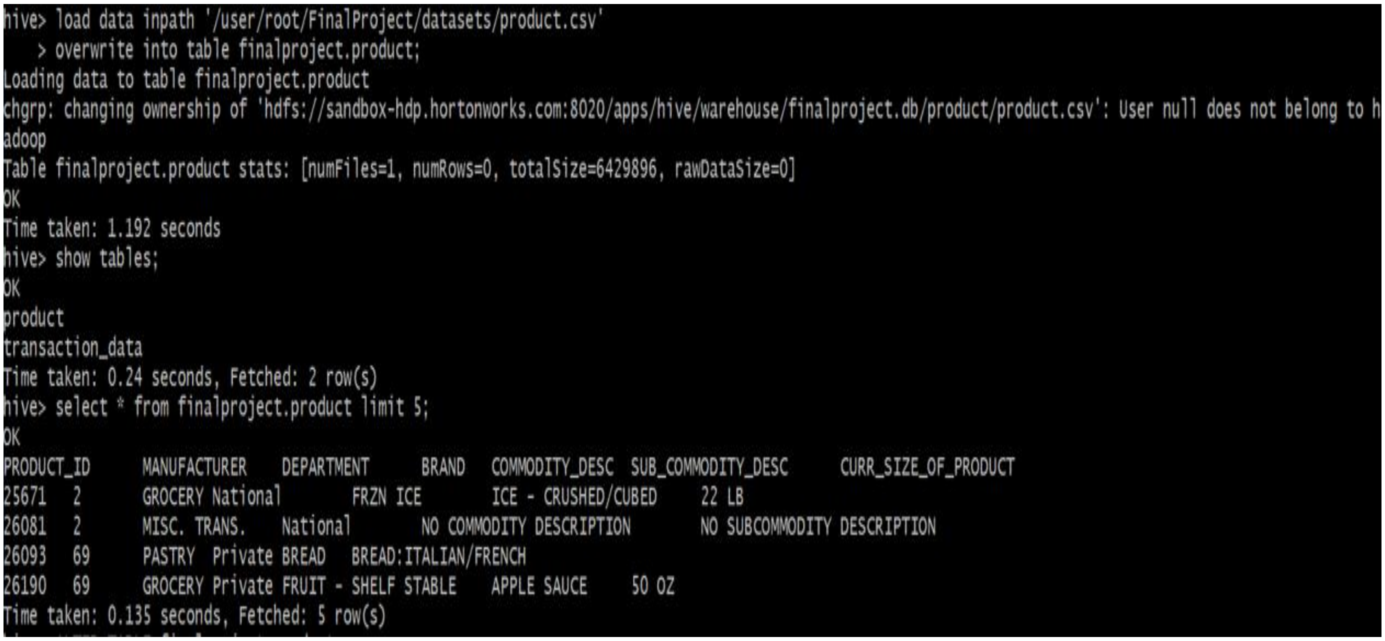
2. Create the tables ‘transaction_data’ and ‘product’ then load the data into table from the HDFS.



```
hive> use finalproject;
OK
Time taken: 0.238 seconds
hive> create table finalproject.transaction_data (
    > household_key string,
    > basket_id string,
    > day string,
    > product_id string,
    > quantity int,
    > sales_value float,
    > store_id string,
    > retail_disc float,
    > trans_time string,
    > week_no string,
    > coupon_disc float,
    > coupon_match_disc float)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ',';
OK
Time taken: 0.869 seconds
```

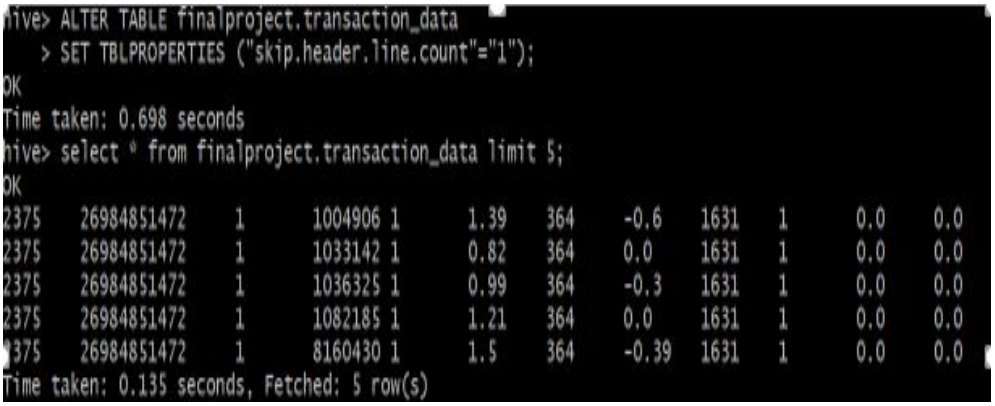


```
hive> create table product (
    > product_id string,
    > manufacturer string,
    > department string,
    > brand string,
    > commodity string,
    > sub_commodity_desc string,
    > size_of_product string)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ',';
OK
Time taken: 0.607 seconds
```

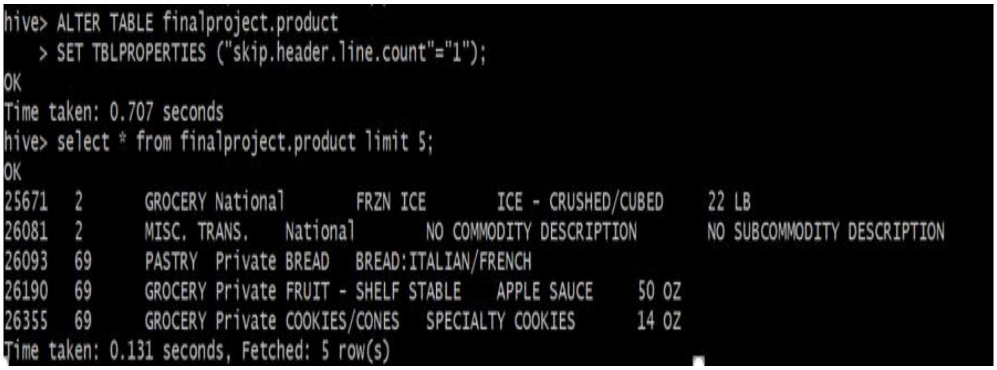


```
hive> load data inpath '/user/root/FinalProject/datasets/product.csv'
    > overwrite into table finalproject.product;
Loading data to table finalproject.product
chgrp: changing ownership of 'hdfs://sandbox-hdp.hortonworks.com:8020/apps/hive/warehouse/finalproject.db/product/product.csv': User null does not belong to h
adoop
Table finalproject.product stats: [numFiles=1, numRows=0, totalSize=6429896, rawDataSize=0]
OK
Time taken: 1.192 seconds
hive> show tables;
OK
product
transaction_data
Time taken: 0.24 seconds, Fetched: 2 row(s)
hive> select * from finalproject.product limit 5;
OK
PRODUCT_ID      MANUFACTURER    DEPARTMENT      BRAND   COMMODITY_DESC  SUB_COMMODITY_DESC      CURR_SIZE_OF_PRODUCT
25671   2       GROCERY National        FRZN ICE        ICE - CRUSHED/CUBED     22 LB
26081   2       MISC. TRANS.    National        NO COMMODITY DESCRIPTION        NO SUBCOMMODITY DESCRIPTION
26093   69      PASTRY  Private BREAD   BREAD:ITALIAN/FRENCH
26190   69      GROCERY Private FRUIT - SHELF STABLE    APPLE SAUCE     50 OZ
Time taken: 0.135 seconds, Fetched: 5 row(s)
```

3. Remove the first row of the tables.

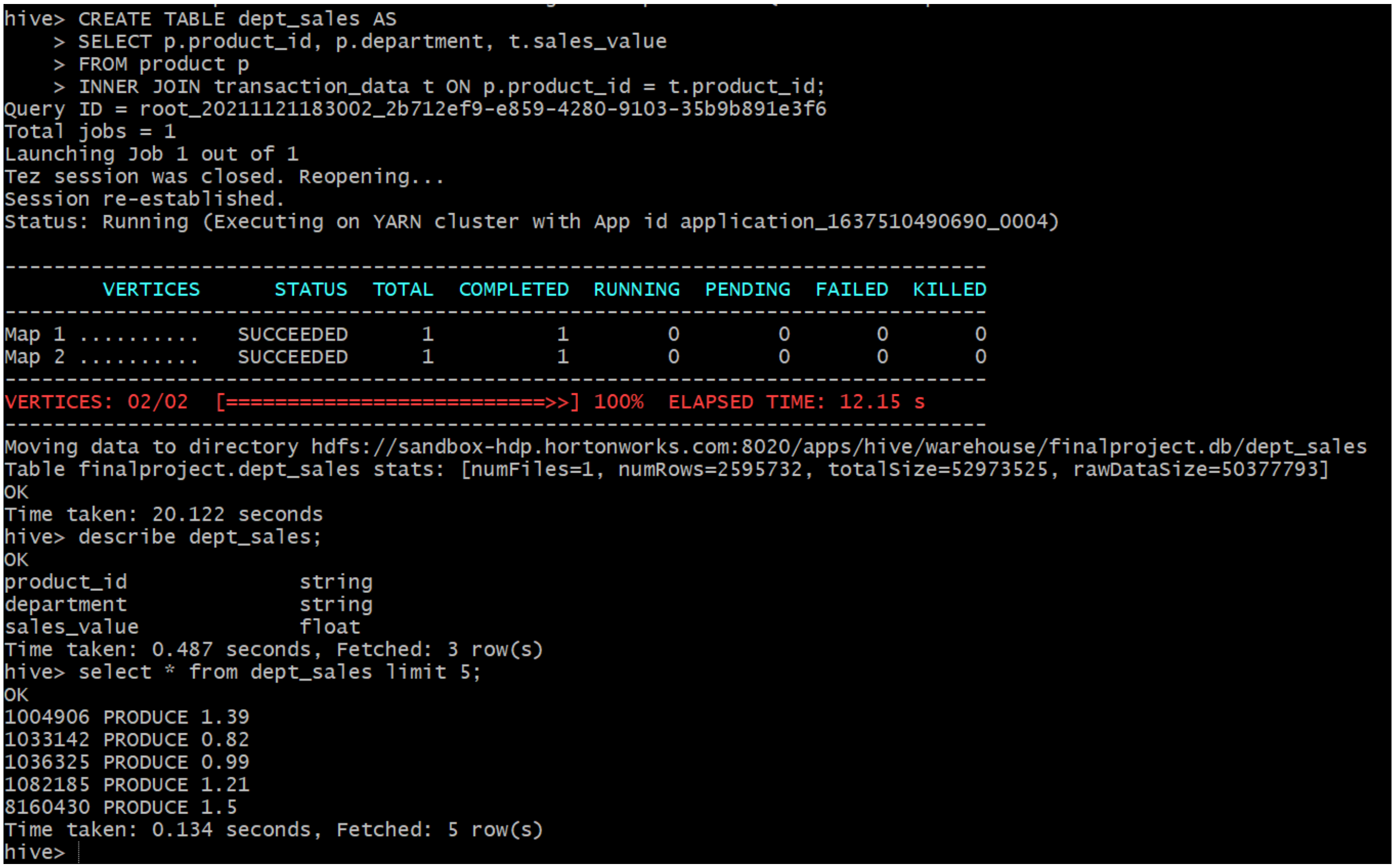


```
hive> ALTER TABLE finalproject.transaction_data
    > SET TBLPROPERTIES ("skip.header.line.count"="1");
OK
Time taken: 0.698 seconds
hive> select * from finalproject.transaction_data limit 5;
OK
2375    26984851472     1       1004906 1       1.39    364     -0.6    1631    1       0.0     0.0
2375    26984851472     1       1033142 1       0.82    364     0.0     1631    1       0.0     0.0
2375    26984851472     1       1036325 1       0.99    364     -0.3    1631    1       0.0     0.0
2375    26984851472     1       1082185 1       1.21    364     0.0     1631    1       0.0     0.0
375     26984851472     1       8160430 1       1.5     364     -0.39   1631    1       0.0     0.0
Time taken: 0.135 seconds, Fetched: 5 row(s)
```



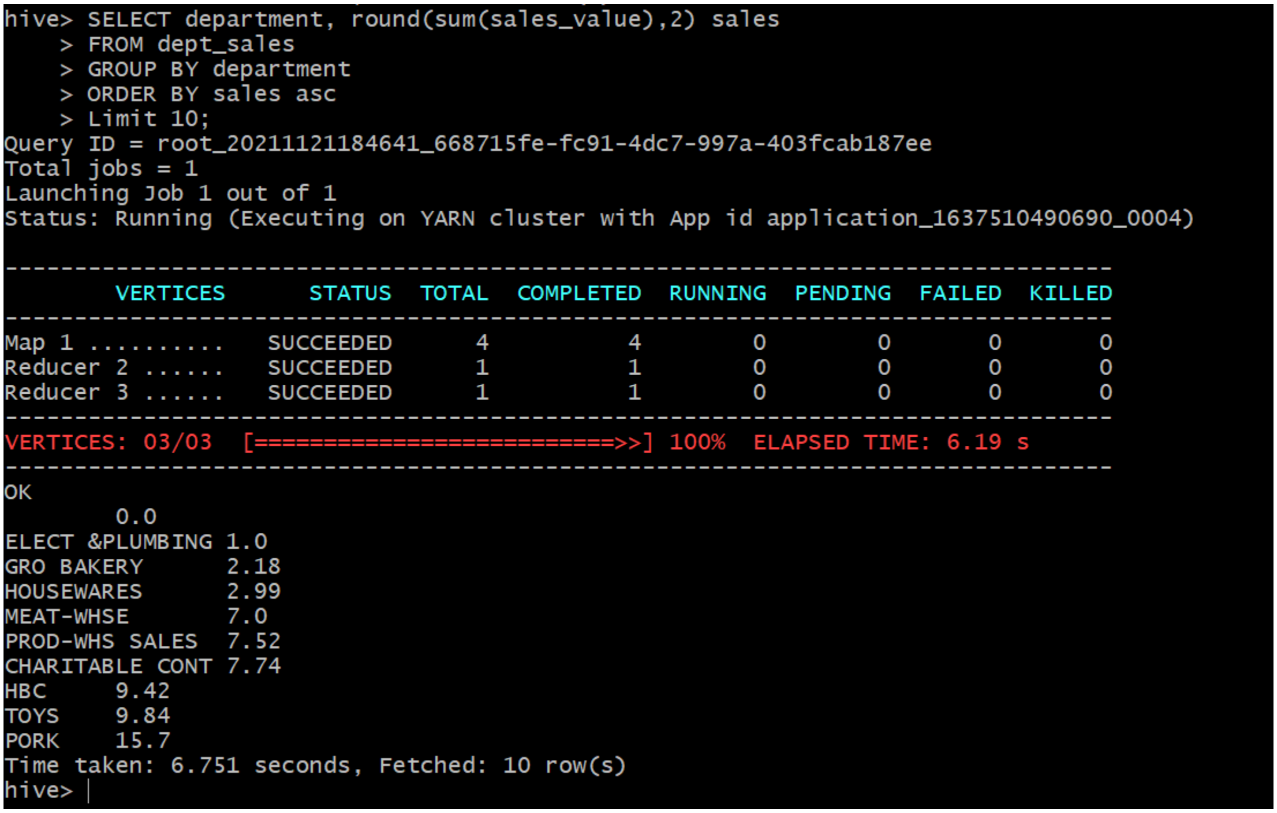
```
hive> ALTER TABLE finalproject.product
    > SET TBLPROPERTIES ("skip.header.line.count"="1");
OK
Time taken: 0.707 seconds
hive> select * from finalproject.product limit 5;
OK
25671   2       GROCERY National        FRZN ICE        ICE - CRUSHED/CUBED     22 LB
26081   2       MISC. TRANS.    National        NO COMMODITY DESCRIPTION        NO SUBCOMMODITY DESCRIPTION
26093   69      PASTRY  Private BREAD   BREAD:ITALIAN/FRENCH
26190   69      GROCERY Private FRUIT - SHELF STABLE    APPLE SAUCE     50 OZ
26355   69      GROCERY Private COOKIES/CONES   SPECIALTY COOKIES       14 OZ
Time taken: 0.131 seconds, Fetched: 5 row(s)
```

4. Create new table ‘dept_sales’ by inner joining the tables ‘transaction_data’ and ‘product’.



```
hive> CREATE TABLE dept_sales AS
    > SELECT p.product_id, p.department, t.sales_value
    > FROM product p
    > INNER JOIN transaction_data t ON p.product_id = t.product_id;
Query ID = root_20211121183002_2b712ef9-e859-4280-9103-35b9b891e3f6
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_1637510490690_0004)

----------------------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      1          1        0        0       0       0
Map 2 ..........   SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 12.15 s
----------------------------------------------------------------------------------------------
Moving data to directory hdfs://sandbox-hdp.hortonworks.com:8020/apps/hive/warehouse/finalproject.db/dept_sales
Table finalproject.dept_sales stats: [numFiles=1, numRows=2595732, totalSize=52973525, rawDataSize=50377793]
OK
Time taken: 20.122 seconds
hive> describe dept_sales;
OK
product_id              string
department              string
sales_value             float
Time taken: 0.487 seconds, Fetched: 3 row(s)
hive> select * from dept_sales limit 5;
OK
1004906 PRODUCE 1.39
1033142 PRODUCE 0.82
1036325 PRODUCE 0.99
1082185 PRODUCE 1.21
8160430 PRODUCE 1.5
Time taken: 0.134 seconds, Fetched: 5 row(s)
hive>
```

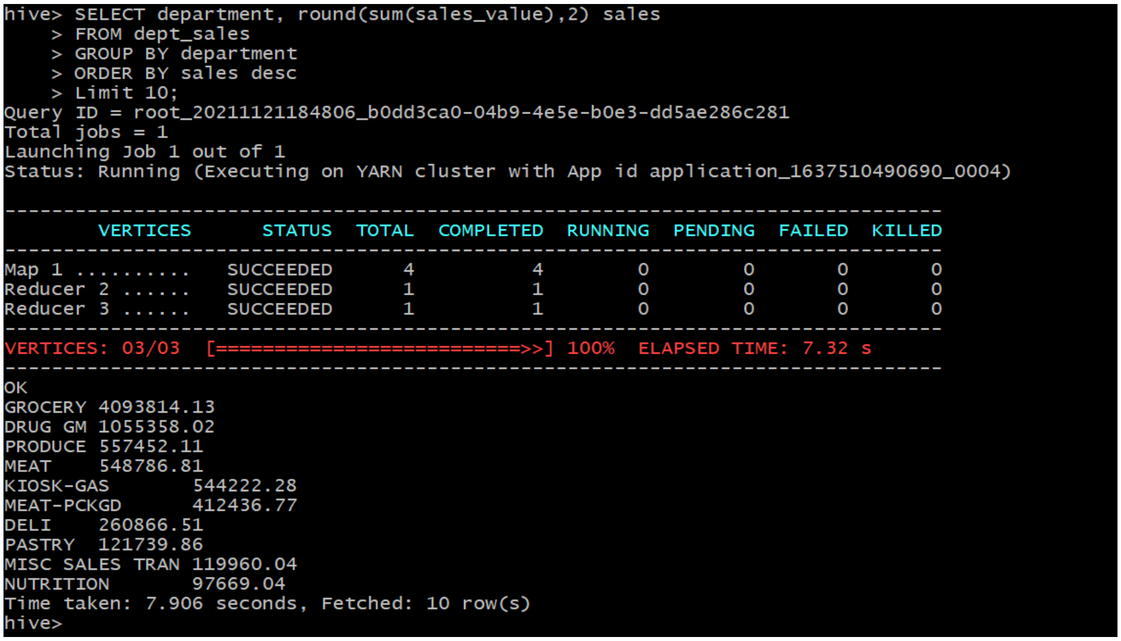
5. Outputs are generated by summing the sales value of each item category in descending and ascending order.



```
hive> SELECT department, round(sum(sales_value),2) sales
    > FROM dept_sales
    > GROUP BY department
    > ORDER BY sales asc
    > Limit 10;
Query ID = root_20211121184641_668715fe-fc91-4dc7-997a-403fcab187ee
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1637510490690_0004)

----------------------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      4          4        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
Reducer 3 ......   SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 6.19 s
----------------------------------------------------------------------------------------------
OK
        0.0
ELECT &PLUMBING 1.0
GRO BAKERY      2.18
HOUSEWARES      2.99
MEAT-WHSE       7.0
PROD-WHS SALES  7.52
CHARITABLE CONT 7.74
HBC     9.42
TOYS    9.84
PORK    15.7
Time taken: 6.751 seconds, Fetched: 10 row(s)
hive> 
```

Figure 1.

In figure 1, the output generated the sales of bottom 10 categories which had the lowest sales. These categories Elect&Plumbing, Gro Bakery, Housewares, Meat-WHSE, Prod-WHS Sales, HBC, Toys and Pork.



```
hive> SELECT department, round(sum(sales_value),2) sales
    > FROM dept_sales
    > GROUP BY department
    > ORDER BY sales desc
    > Limit 10;
Query ID = root_20211121184806_b0dd3ca0-04b9-4e5e-b0e3-dd5ae286c281
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1637510490690_0004)

----------------------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      4          4        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
Reducer 3 ......   SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 7.32 s
----------------------------------------------------------------------------------------------
OK
GROCERY 4093814.13
DRUG GM 1055358.02
PRODUCE 557452.11
MEAT    548786.81
KIOSK-GAS       544222.28
MEAT-PCKGD      412436.77
DELI    260866.51
PASTRY  121739.86
MISC SALES TRAN 119960.04
NUTRITION       97669.04
Time taken: 7.906 seconds, Fetched: 10 row(s)
hive>
```

Figure 2.

In figure 2, the output generated shows the top 10 categories which had higher sales. These categories are Grocery, Drug GM, Produce, Meat, Kiosk-Gas, Meat-Pckgd, Deli, Pastry, Misc Sales Tran and Nutrition.

Q3. What are the most profitable categories of the products overtime?

From the Figure 2., it is evident the **Groceries** are most profitable category in the stores with sale of around 4,093,814 dollars which is around 50 percent of total sales of the store in 2 years period.

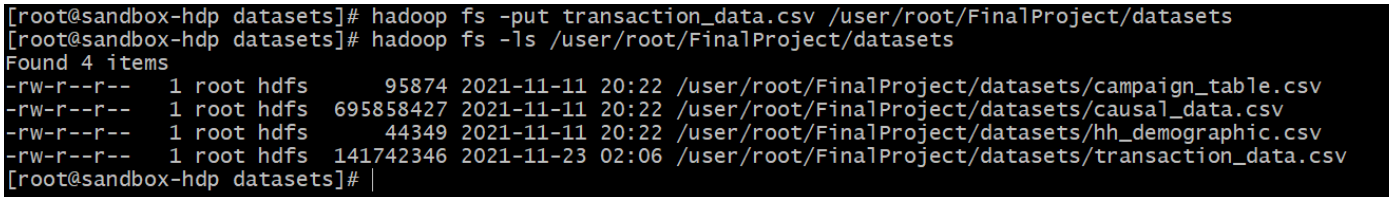
Q3 a. Which day in two years period has the highest sales?

The tools used for extracting the information are:

- Hadoop map-reduce (processing)
- HDFS (Hadoop Distributed File System)

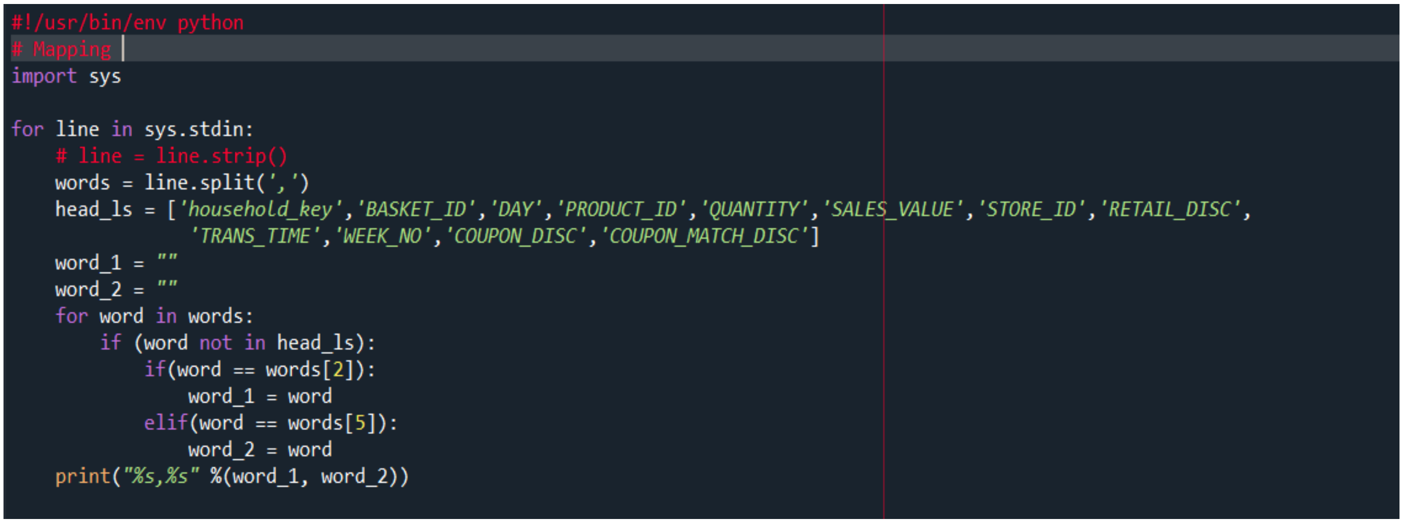
The following is procedure followed to get the desired output:

1. Copy the required file data files to the HDFS. In this case, it is 'transaction_data.csv'.



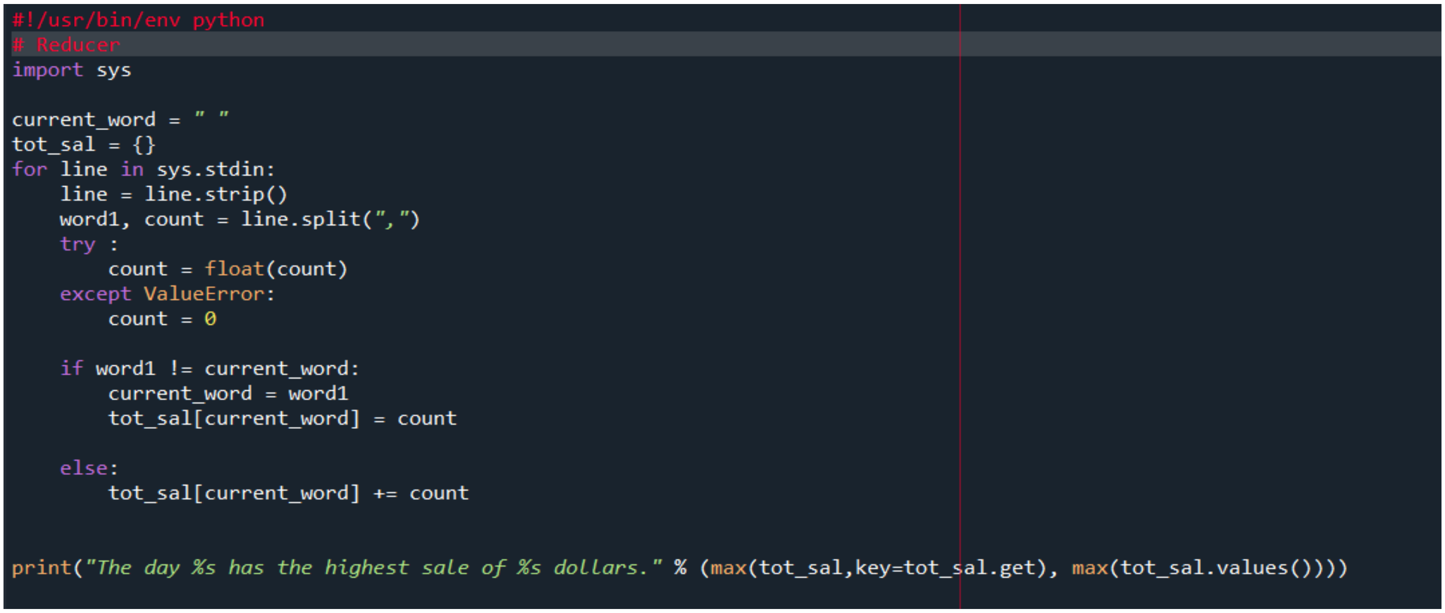
```
[root@sandbox-hdp datasets]# hadoop fs -put transaction_data.csv /user/root/FinalProject/datasets
[root@sandbox-hdp datasets]# hadoop fs -ls /user/root/FinalProject/datasets
Found 4 items
-rw-r--r--   1 root hdfs      95874 2021-11-11 20:22 /user/root/FinalProject/datasets/campaign_table.csv
-rw-r--r--   1 root hdfs  695858427 2021-11-11 20:22 /user/root/FinalProject/datasets/causal_data.csv
-rw-r--r--   1 root hdfs      44349 2021-11-11 20:22 /user/root/FinalProject/datasets/hh_demographic.csv
-rw-r--r--   1 root hdfs  141742346 2021-11-23 02:06 /user/root/FinalProject/datasets/transaction_data.csv
[root@sandbox-hdp datasets]# |
```

2. Created the Mapping python file, in which first row is skipped and output the two columns value in key, value pair. In first part of pair 'word_1' refers to the 'day' column and in second part 'word_2' refers to the 'sales_value' column.



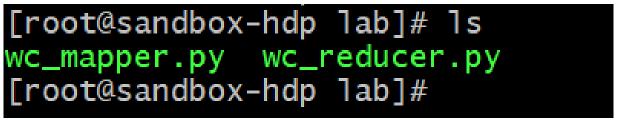
```python
#!/usr/bin/env python
# Mapping
import sys

for line in sys.stdin:
    # line = line.strip()
    words = line.split(',')
    head_ls = ['household_key','BASKET_ID','DAY','PRODUCT_ID','QUANTITY','SALES_VALUE','STORE_ID','RETAIL_DISC',
               'TRANS_TIME','WEEK_NO','COUPON_DISC','COUPON_MATCH_DISC']
    word_1 = ""
    word_2 = ""
    for word in words:
        if (word not in head_ls):
            if(word == words[2]):
                word_1 = word
            elif(word == words[5]):
                word_2 = word
    print("%s,%s" %(word_1, word_2))
```

3. The reducer python file ‘wc_reducer.py’ is created to read through the output of the mapping python file ‘wc_mapper.py’ and performs the operations and extract the day which has the highest sales value.



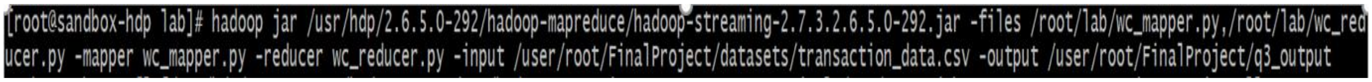
```
#!/usr/bin/env python
# Reducer
import sys

current_word = " "
tot_sal = {}
for line in sys.stdin:
    line = line.strip()
    word1, count = line.split(",")
    try :
        count = float(count)
    except ValueError:
        count = 0

    if word1 != current_word:
        current_word = word1
        tot_sal[current_word] = count

    else:
        tot_sal[current_word] += count


print("The day %s has the highest sale of %s dollars." % (max(tot_sal,key=tot_sal.get), max(tot_sal.values())))
```

4. The mapper and reducer files are copied into the local storage of sandbox.

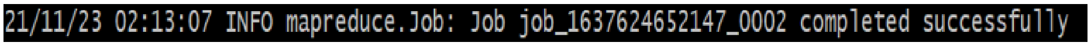


```
[root@sandbox-hdp lab]# ls
wc_mapper.py  wc_reducer.py
[root@sandbox-hdp lab]#
```

5. Python streaming is executed and it uses wc_mapper.py and wc_reducer.py files to get the output.

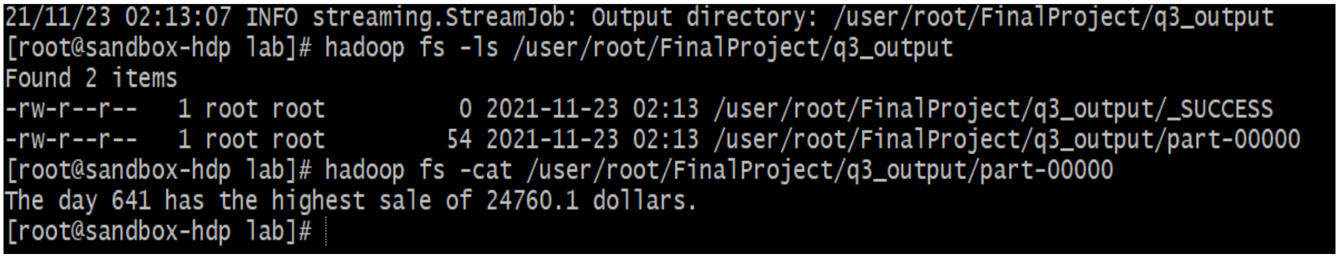


```
[root@sandbox-hdp lab]# hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming-2.7.3.2.6.5.0-292.jar -files /root/lab/wc_mapper.py,/root/lab/wc_red
ucer.py -mapper wc_mapper.py -reducer wc_reducer.py -input /user/root/FinalProject/datasets/transaction_data.csv -output /user/root/FinalProject/q3_output
```



```
21/11/23 02:13:07 INFO mapreduce.Job: Job job_1637624652147_0002 completed successfully
```

6. The output files are generated.



```
21/11/23 02:13:07 INFO streaming.StreamJob: Output directory: /user/root/FinalProject/q3_output
[root@sandbox-hdp lab]# hadoop fs -ls /user/root/FinalProject/q3_output
Found 2 items
-rw-r--r--   1 root root          0 2021-11-23 02:13 /user/root/FinalProject/q3_output/_SUCCESS
-rw-r--r--   1 root root         54 2021-11-23 02:13 /user/root/FinalProject/q3_output/part-00000
[root@sandbox-hdp lab]# hadoop fs -cat /user/root/FinalProject/q3_output/part-00000
The day 641 has the highest sale of 24760.1 dollars.
[root@sandbox-hdp lab]#
```

The output shows that 641th day of 2 year period had the highest sales of 24740.1 dollars. The 641th day falls in the month of October. From the value we infer the reason behind the sales rise.

Q4: Are the marketing campaigns effective?

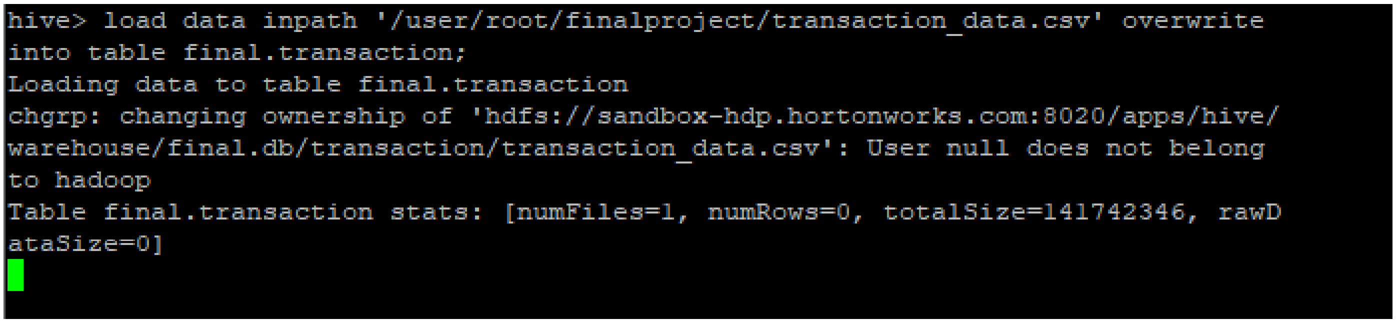
Datasets : **transaction_data.csv** , **coupon_redempt.csv**

Tools:

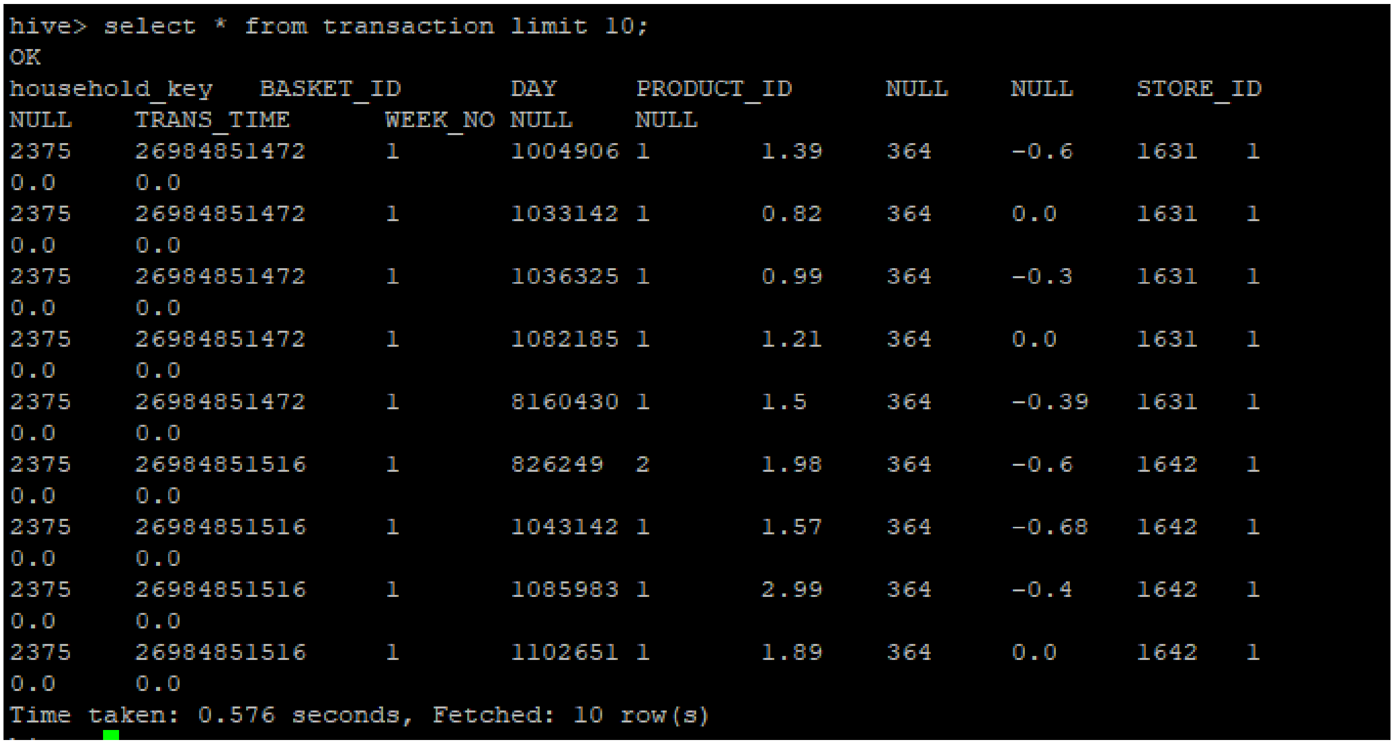
- Hive: create tables, query tables
- Hadoop Distributed File System (HDFS) : store distributed data

Aiming to prove the effects of marketing campaigns, whether the coupons distributed during campaigns can increase the sales by comparing the sales values and quantities between purchase with redeemed coupons and without coupons.

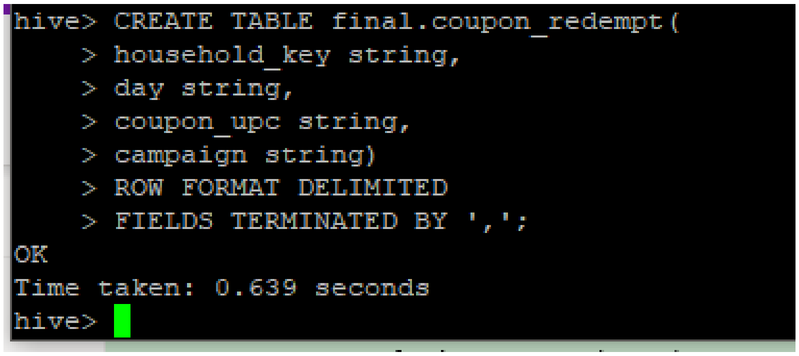
1. Create tables: **transaction_data, coupon_redempt**



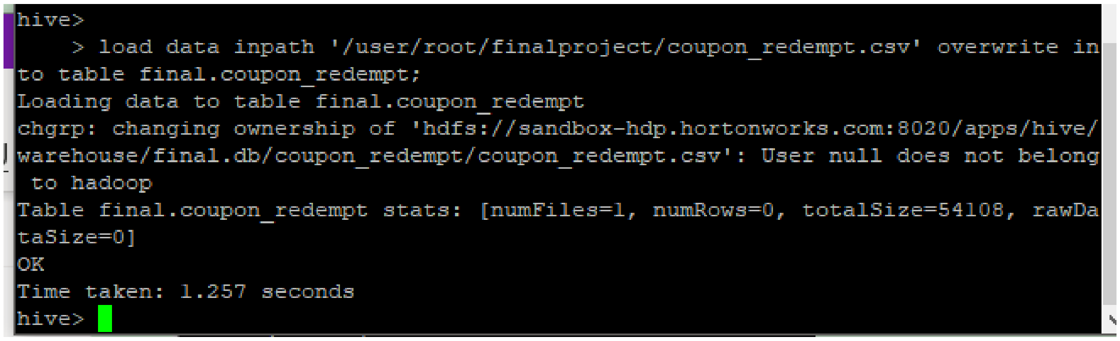
```
hive> load data inpath '/user/root/finalproject/transaction_data.csv' overwrite
into table final.transaction;
Loading data to table final.transaction
chgrp: changing ownership of 'hdfs://sandbox-hdp.hortonworks.com:8020/apps/hive/
warehouse/final.db/transaction/transaction_data.csv': User null does not belong
to hadoop
Table final.transaction stats: [numFiles=1, numRows=0, totalSize=141742346, rawD
ataSize=0]
```



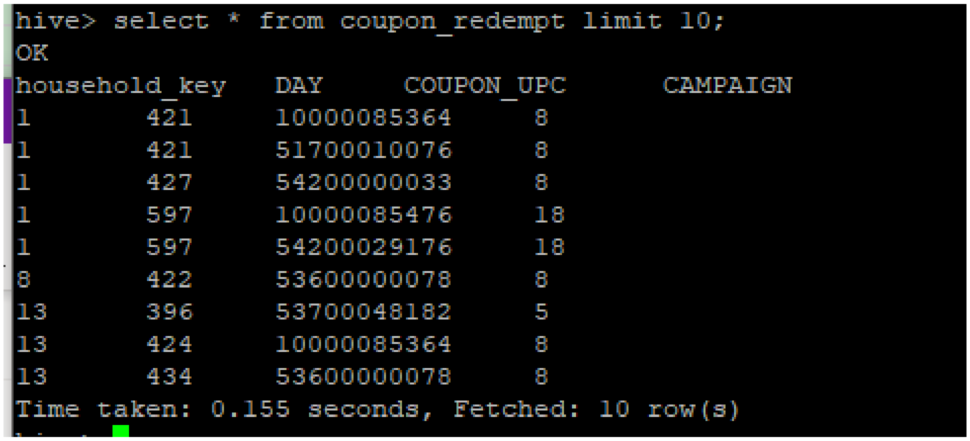
```
hive> select * from transaction limit 10;
OK
household_key   BASKET_ID       DAY     PRODUCT_ID      NULL    NULL    STORE_ID
NULL    TRANS_TIME      WEEK_NO NULL    NULL
2375    26984851472     1       1004906 1       1.39    364     -0.6    1631    1
0.0     0.0
2375    26984851472     1       1033142 1       0.82    364     0.0     1631    1
0.0     0.0
2375    26984851472     1       1036325 1       0.99    364     -0.3    1631    1
0.0     0.0
2375    26984851472     1       1082185 1       1.21    364     0.0     1631    1
0.0     0.0
2375    26984851472     1       8160430 1       1.5     364     -0.39   1631    1
0.0     0.0
2375    26984851516     1       826249  2       1.98    364     -0.6    1642    1
0.0     0.0
2375    26984851516     1       1043142 1       1.57    364     -0.68   1642    1
0.0     0.0
2375    26984851516     1       1085983 1       2.99    364     -0.4    1642    1
0.0     0.0
2375    26984851516     1       1102651 1       1.89    364     0.0     1642    1
0.0     0.0
Time taken: 0.576 seconds, Fetched: 10 row(s)
```



```
hive> CREATE TABLE final.coupon_redempt(
    > household_key string,
    > day string,
    > coupon_upc string,
    > campaign string)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ',';
OK
Time taken: 0.639 seconds
hive> 
```



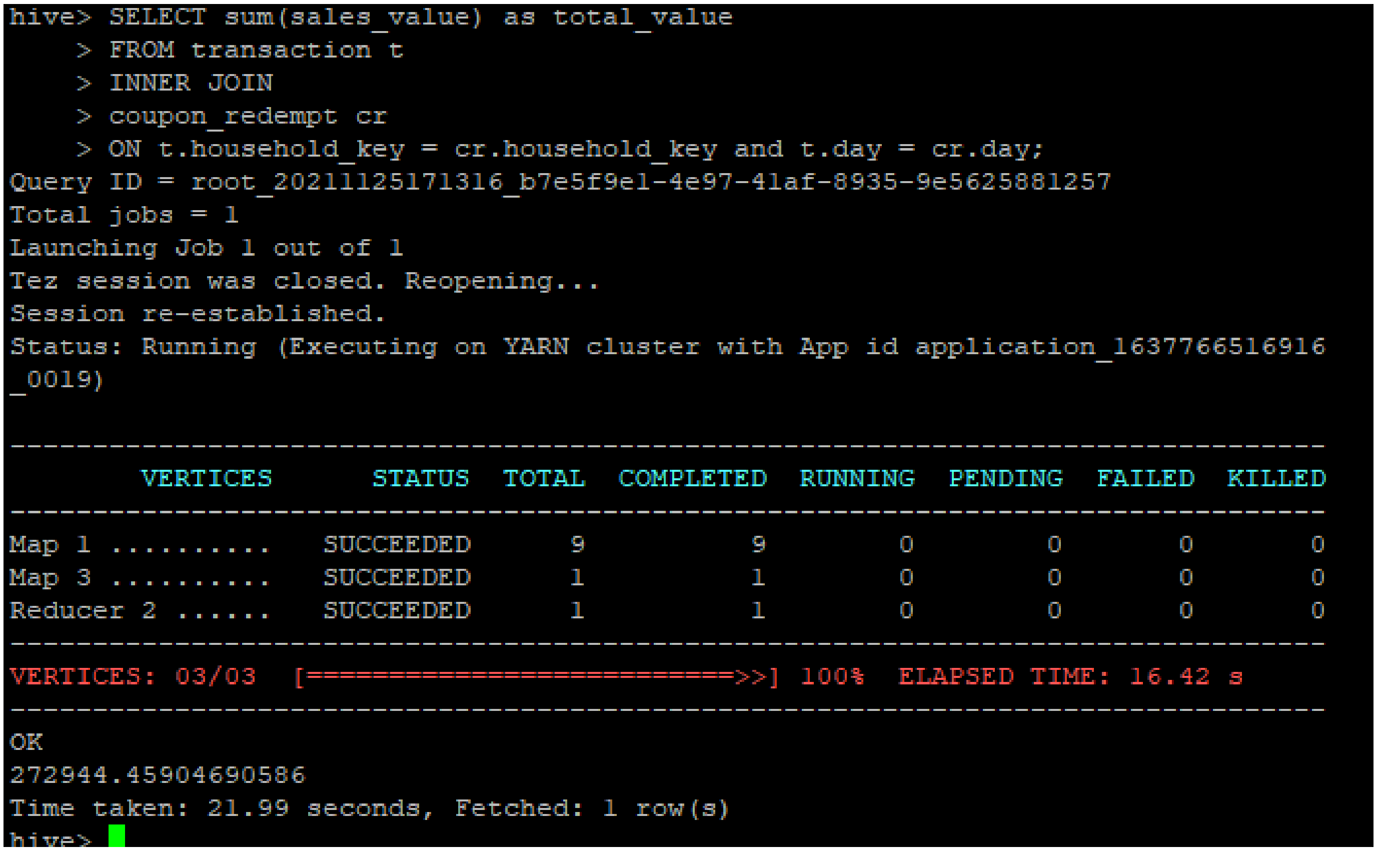
```
hive>
    > load data inpath '/user/root/finalproject/coupon_redempt.csv' overwrite in
to table final.coupon_redempt;
Loading data to table final.coupon_redempt
chgrp: changing ownership of 'hdfs://sandbox-hdp.hortonworks.com:8020/apps/hive/
warehouse/final.db/coupon_redempt/coupon_redempt.csv': User null does not belong
 to hadoop
Table final.coupon_redempt stats: [numFiles=1, numRows=0, totalSize=54108, rawDa
taSize=0]
OK
Time taken: 1.257 seconds
hive> 
```



```
hive> select * from coupon_redempt limit 10;
OK
household_key   DAY     COUPON_UPC      CAMPAIGN
1       421     10000085364     8
1       421     51700010076     8
1       427     54200000033     8
1       597     10000085476     18
1       597     54200029176     18
8       422     53600000078     8
13      396     53700048182     5
13      424     10000085364     8
13      434     53600000078     8
Time taken: 0.155 seconds, Fetched: 10 row(s)
```

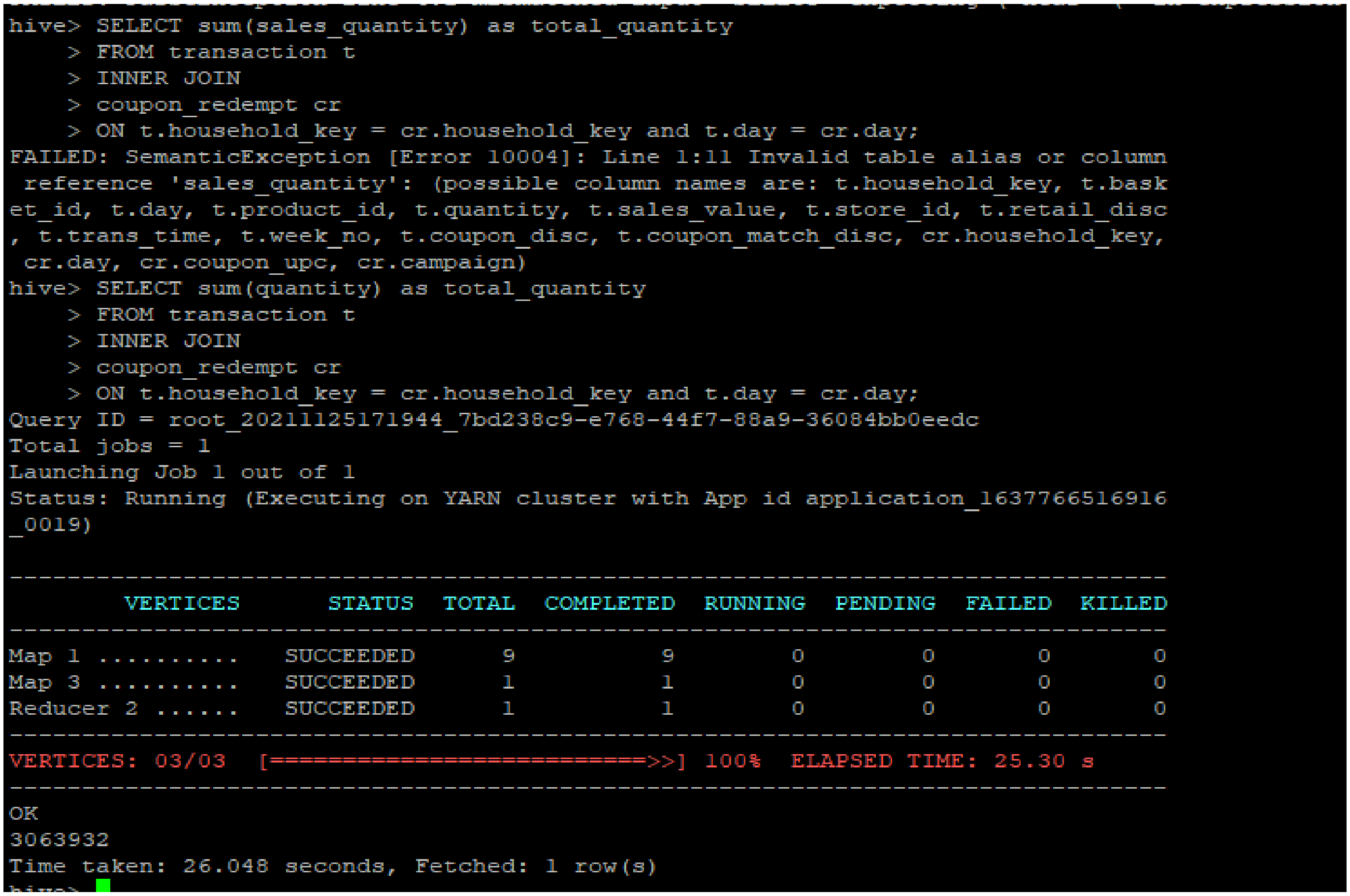
2. Calculate the total values for the purchase with redeemed coupons

After checking the values in transaction table and coupon_redempt tables, we found that the “household_key” is a n-to-n matching key. As “household_key” in both tables are not unique. We can understand as one household had several purchases (in transactions table) while the same household had multiple records of using coupons (in coupon_redempt table). Therefore, in order not to expand the total sales amount after the join, we add another value - "day" - to the join key to restrict the matching condition.



```
hive> SELECT sum(sales_value) as total_value
    > FROM transaction t
    > INNER JOIN
    > coupon_redempt cr
    > ON t.household_key = cr.household_key and t.day = cr.day;
Query ID = root_20211125171316_b7e5f9e1-4e97-41af-8935-9e5625881257
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_1637766516916
_0019)

----------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      9          9        0        0       0       0
Map 3 ..........   SUCCEEDED      1          1        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 16.42 s
----------------------------------------------------------------------------------
OK
272944.45904690586
Time taken: 21.99 seconds, Fetched: 1 row(s)
hive>
```

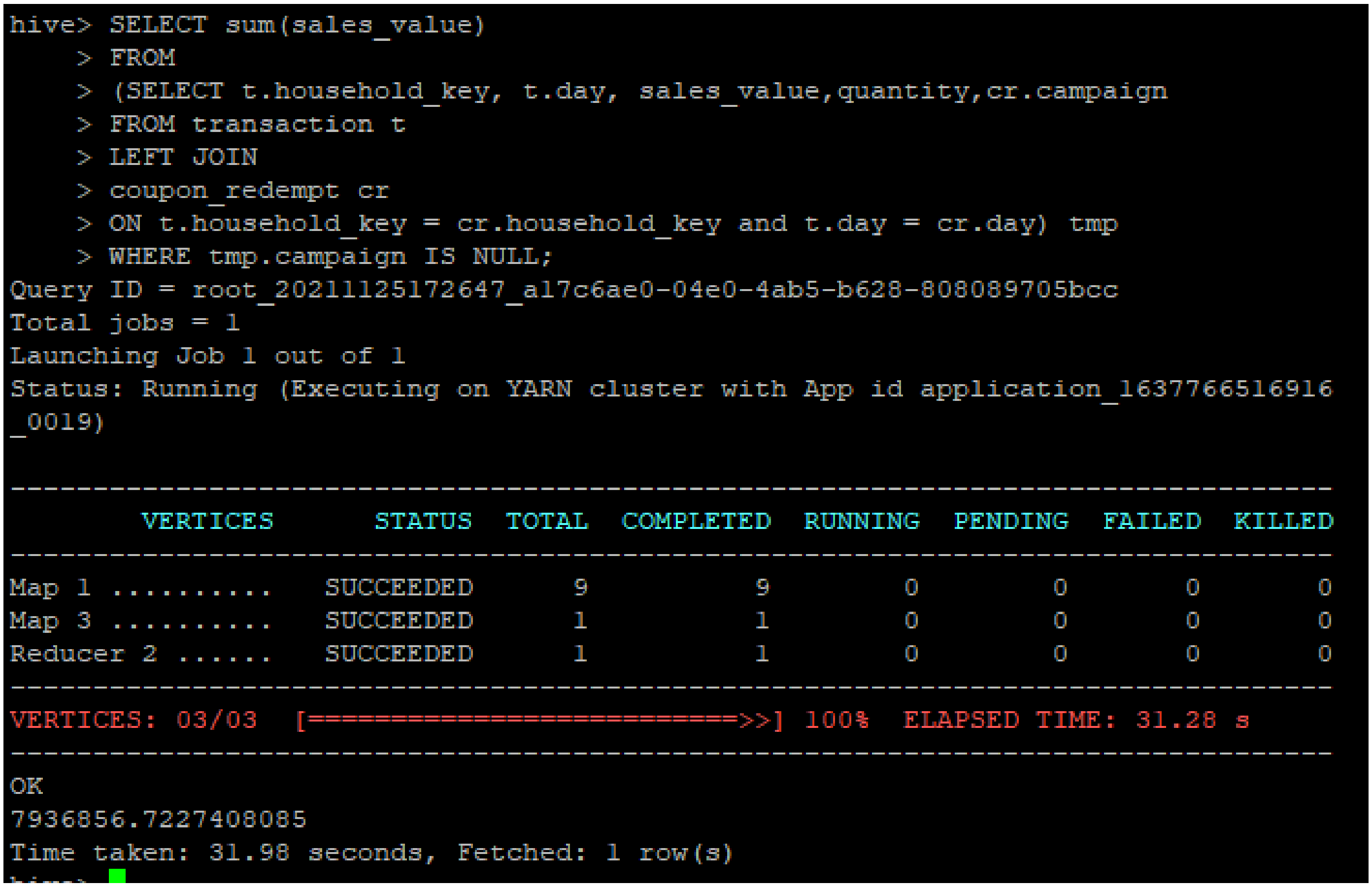
3. Calculate the total quantities for the purchase with redeemed coupons



```
hive> SELECT sum(sales_quantity) as total_quantity
    > FROM transaction t
    > INNER JOIN
    > coupon_redempt cr
    > ON t.household_key = cr.household_key and t.day = cr.day;
FAILED: SemanticException [Error 10004]: Line 1:11 Invalid table alias or column
 reference 'sales_quantity': (possible column names are: t.household_key, t.bask
et_id, t.day, t.product_id, t.quantity, t.sales_value, t.store_id, t.retail_disc
, t.trans_time, t.week_no, t.coupon_disc, t.coupon_match_disc, cr.household_key,
 cr.day, cr.coupon_upc, cr.campaign)
hive> SELECT sum(quantity) as total_quantity
    > FROM transaction t
    > INNER JOIN
    > coupon_redempt cr
    > ON t.household_key = cr.household_key and t.day = cr.day;
Query ID = root_20211125171944_7bd238c9-e768-44f7-88a9-36084bb0eedc
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1637766516916
_0019)

----------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      9          9        0        0       0       0
Map 3 ..........   SUCCEEDED      1          1        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 25.30 s
----------------------------------------------------------------------------------
OK
3063932
Time taken: 26.048 seconds, Fetched: 1 row(s)
hive>
```

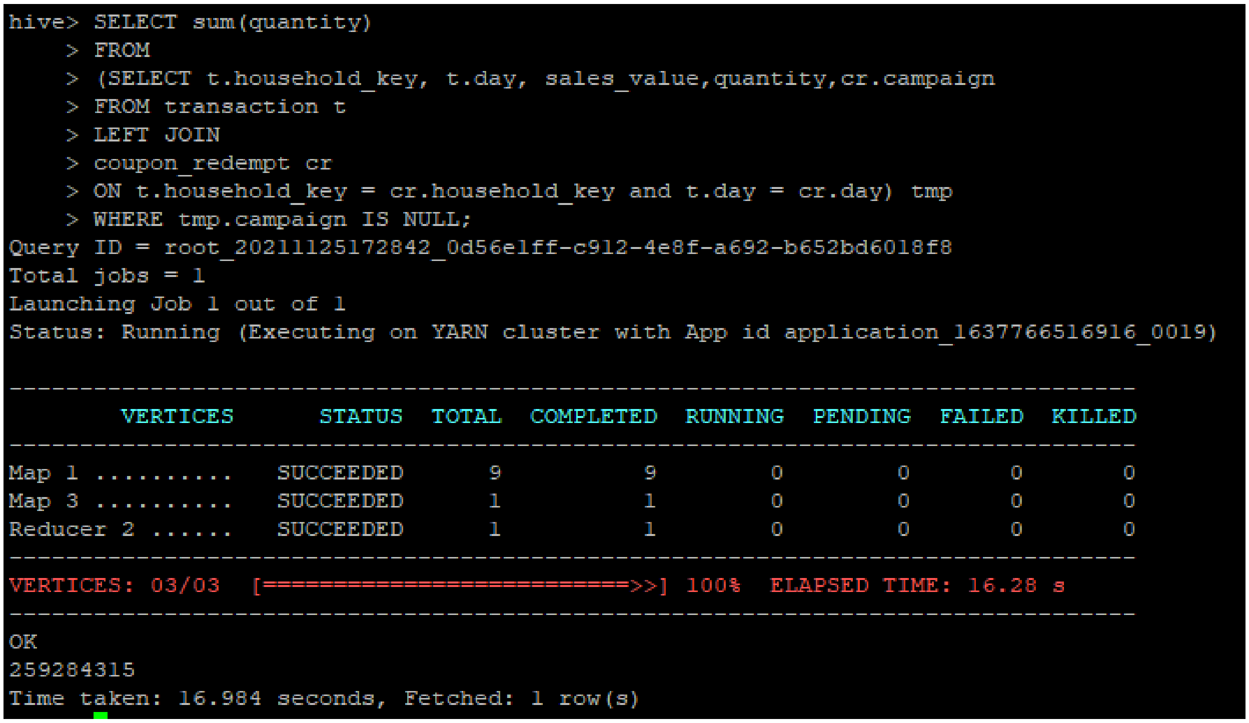
4. Calculate the total values for the purchase without redeemed coupons

This time we use left join to keep all the purchase records from transaction table. The ones which are not matched by the records from coupon_redempt table are the purchases without redeeming coupons.



```
hive> SELECT sum(sales_value)
    > FROM
    > (SELECT t.household_key, t.day, sales_value,quantity,cr.campaign
    > FROM transaction t
    > LEFT JOIN
    > coupon_redempt cr
    > ON t.household_key = cr.household_key and t.day = cr.day) tmp
    > WHERE tmp.campaign IS NULL;
Query ID = root_20211125172647_a17c6ae0-04e0-4ab5-b628-808089705bcc
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1637766516916
_0019)

--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      9          9        0        0       0       0
Map 3 ..........   SUCCEEDED      1          1        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 31.28 s
--------------------------------------------------------------------------------
OK
7936856.7227408085
Time taken: 31.98 seconds, Fetched: 1 row(s)
```

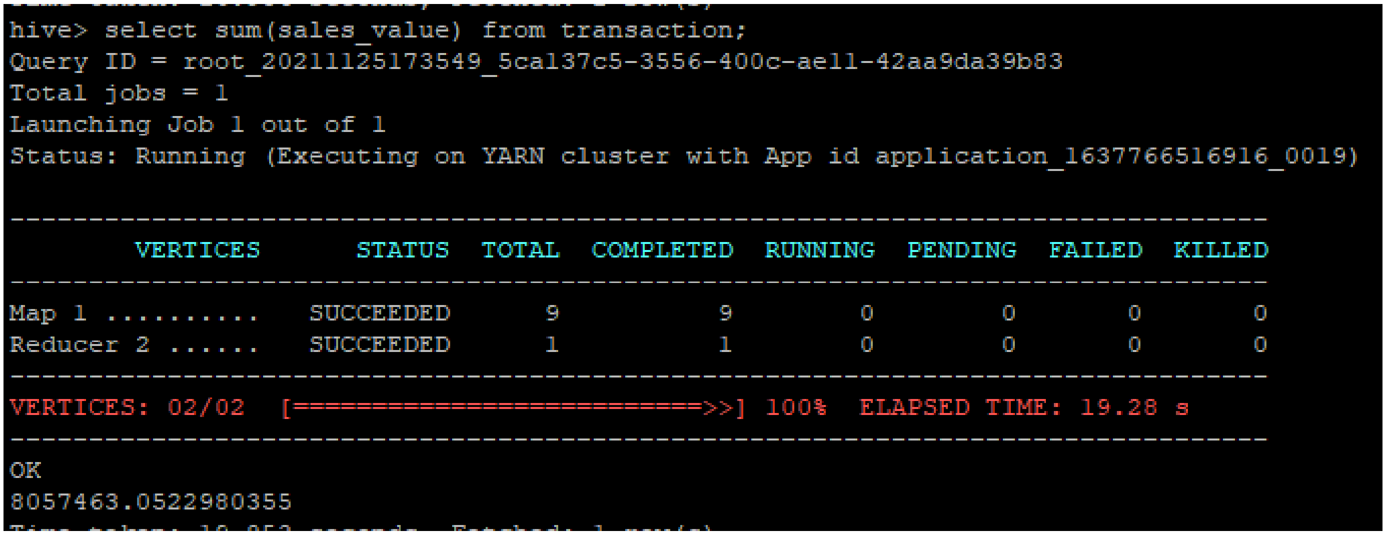
5. Calculate the total quantities for the purchase without redeemed coupons



```
hive> SELECT sum(quantity)
    > FROM
    > (SELECT t.household_key, t.day, sales_value,quantity,cr.campaign
    > FROM transaction t
    > LEFT JOIN
    > coupon_redempt cr
    > ON t.household_key = cr.household_key and t.day = cr.day) tmp
    > WHERE tmp.campaign IS NULL;
Query ID = root_20211125172842_0d56e1ff-c912-4e8f-a692-b652bd6018f8
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1637766516916_0019)

--------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      9          9        0        0       0       0
Map 3 ..........   SUCCEEDED      1          1        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 16.28 s
--------------------------------------------------------------------------------
OK
259284315
Time taken: 16.984 seconds, Fetched: 1 row(s)
```

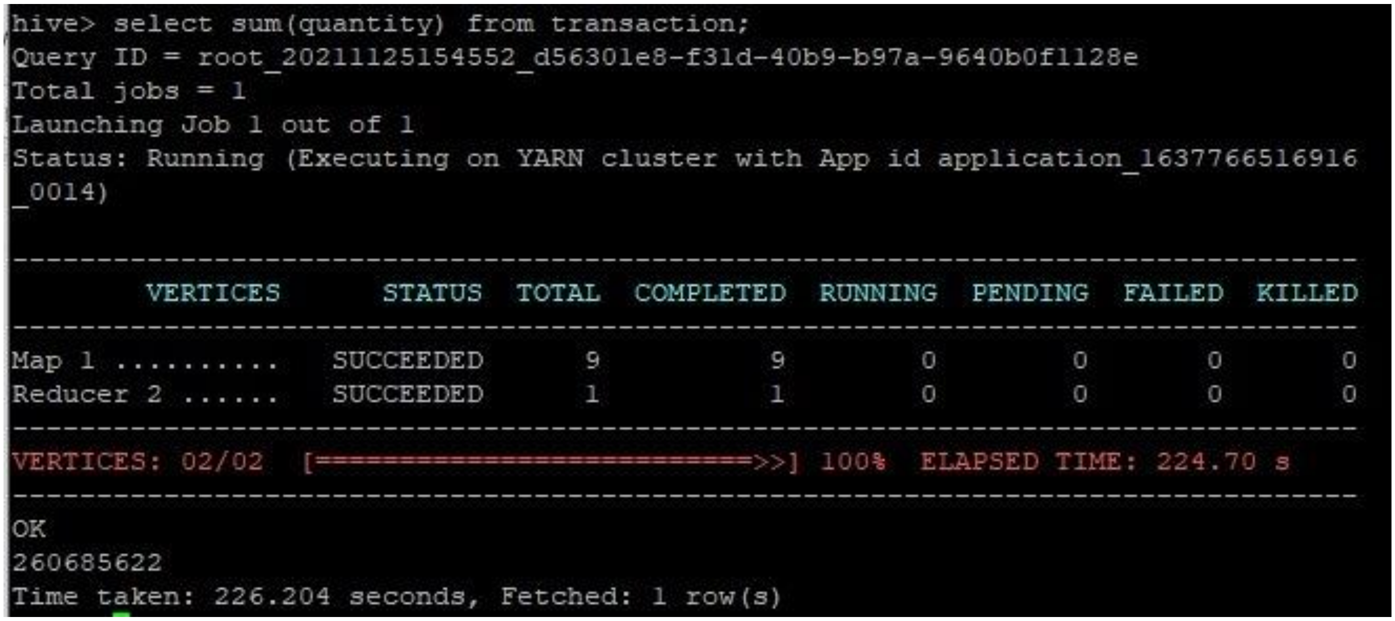
6. Use another method to validate the total amounts

Calculate the total sales value by selecting total “sales_values” from transaction table.



```
hive> select sum(sales_value) from transaction;
Query ID = root_20211125173549_5ca137c5-3556-400c-ae11-42aa9da39b83
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1637766516916_0019)

----------------------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      9          9        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 19.28 s
----------------------------------------------------------------------------------------------
OK
8057463.0522980355
Time taken: 19.053 seconds, Fetched: 1 row(s)
```

Calculate the total sales value by selecting total “quantities” from transaction table.



```
hive> select sum(quantity) from transaction;
Query ID = root_20211125154552_d56301e8-f31d-40b9-b97a-9640b0f1128e
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1637766516916
_0014)

----------------------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      9          9        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 224.70 s
----------------------------------------------------------------------------------------------
OK
260685622
Time taken: 226.204 seconds, Fetched: 1 row(s)
```

7. Calculate the promotion rate by sales and quantities:

Sales with redeemed coupons / (sales with redeemed coupons + sales without redeemed coupons):

272944/ (272944+7936856) * 100 = 3.3%

Sales with redeemed coupons / total sales:

272944/8057463 * 100 = 3.3%

Quantities with redeemed coupons / (quantities with redeemed coupons + quantities without redeemed coupons):

3063932/ (3063932+259284315) * 100 = 1.2%

Quantities with redeemed coupons / total quantities:

3063932/260685622 * 100 = 1.2%

The marketing campaign has increased sales by 3.3%, 1,2% of quantities. Obviously, the rate of increase in volume is less than half of the increase in sales. We can conclude that marketing campaigns are more effective on higher price products.

Q5. Which of the marketing campaigns was the most successful one?

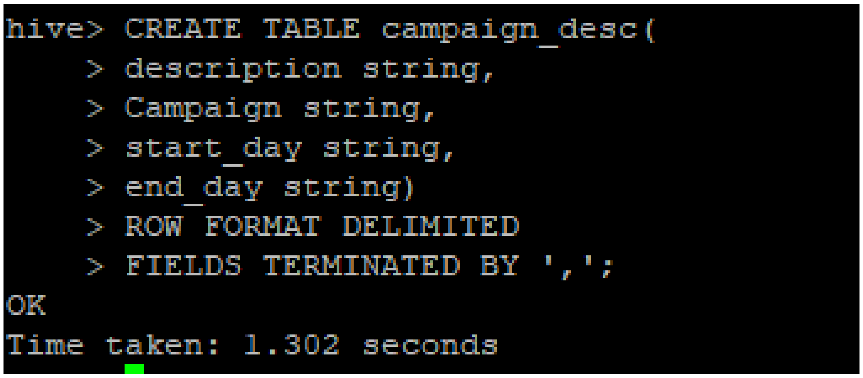
Datasets : **transaction_data.csv** , **coupon_redempt.csv**, **campaign_desc.csv**

Tools:

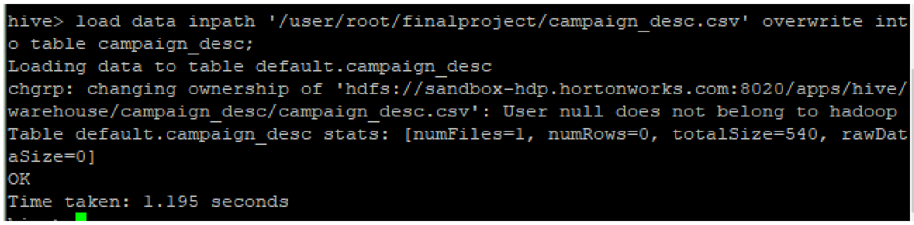
- Hive: create tables, query tables
- Hadoop Distributed File System (HDFS) : store distributed data

To evaluate the success of the campaign, we can take clues from the coupons redeemed during the campaign by adding up the total sales and quantities relating the redeemed coupons for each campagin. Then check which one has the largest volume.

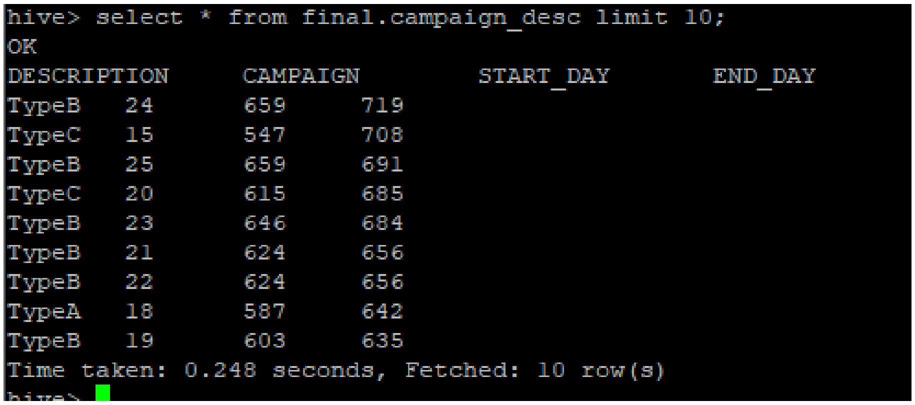
1. Create tables: **campaign_desc**



```
hive> CREATE TABLE campaign_desc(
    > description string,
    > Campaign string,
    > start_day string,
    > end_day string)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ',';
OK
Time taken: 1.302 seconds
```



```
hive> load data inpath '/user/root/finalproject/campaign_desc.csv' overwrite int
o table campaign_desc;
Loading data to table default.campaign_desc
chgrp: changing ownership of 'hdfs://sandbox-hdp.hortonworks.com:8020/apps/hive/
warehouse/campaign_desc/campaign_desc.csv': User null does not belong to hadoop
Table default.campaign_desc stats: [numFiles=1, numRows=0, totalSize=540, rawDat
aSize=0]
OK
Time taken: 1.195 seconds
```

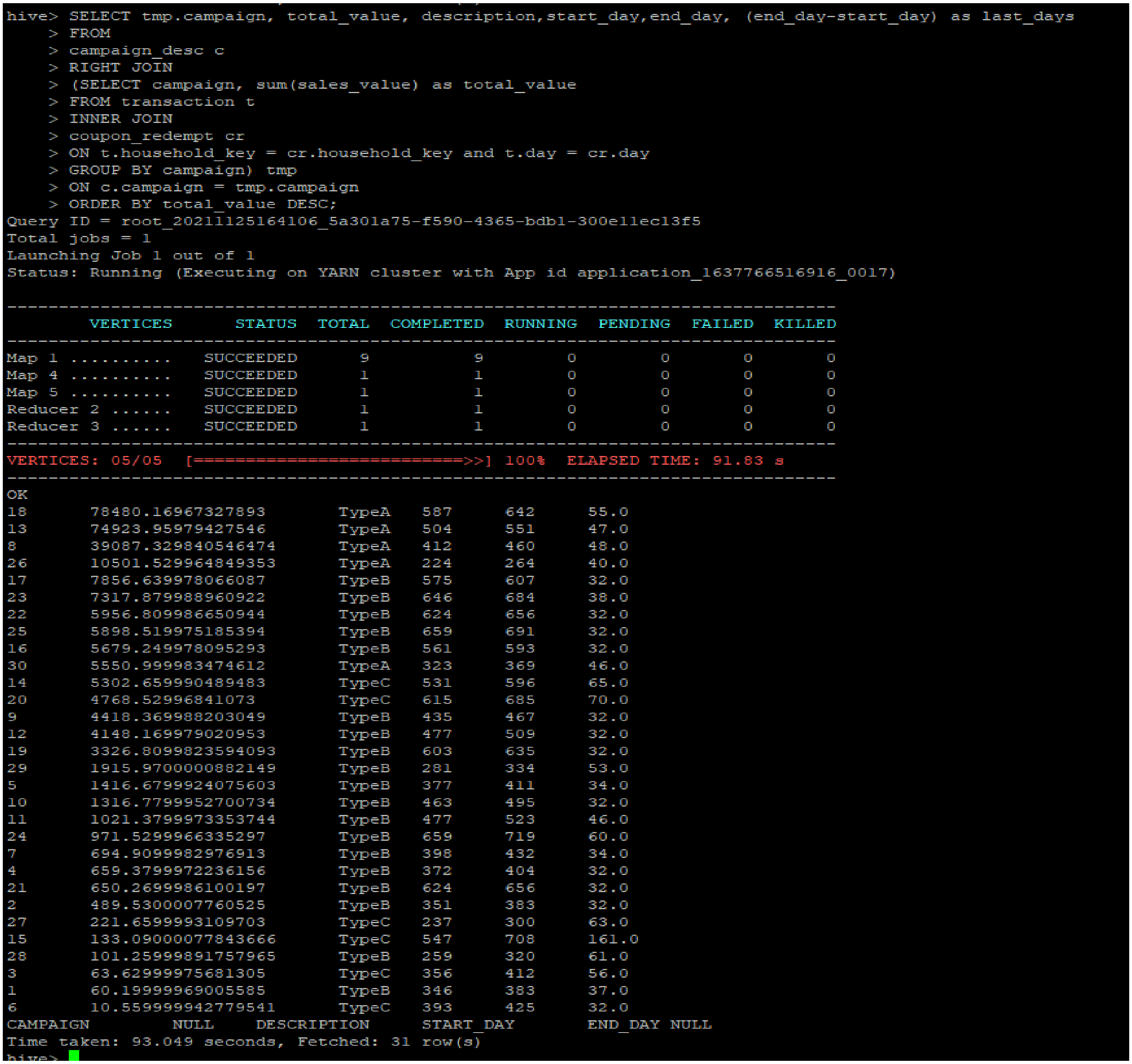


```
hive> select * from final.campaign_desc limit 10;
OK
DESCRIPTION     CAMPAIGN        START_DAY       END_DAY
TypeB   24      659     719
TypeC   15      547     708
TypeB   25      659     691
TypeC   20      615     685
TypeB   23      646     684
TypeB   21      624     656
TypeB   22      624     656
TypeA   18      587     642
TypeB   19      603     635
Time taken: 0.248 seconds, Fetched: 10 row(s)
```

2. Find total sales value/quantities of each campaign by descending order

Calculate total sales value per campaign by descending order.

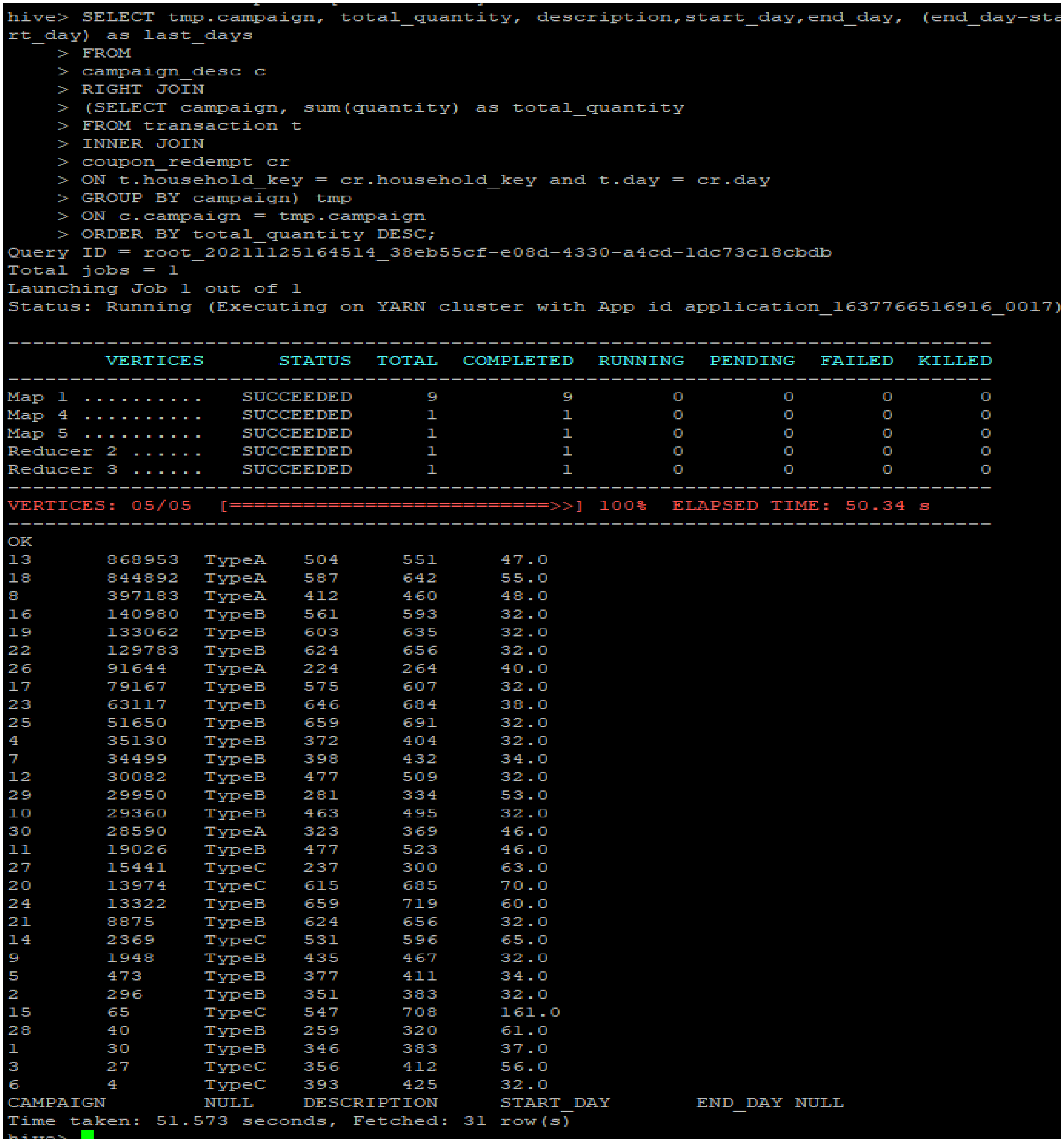
After inner joining transaction table and coupon_redempt table (as described in Q4), the query results are grouped by campaign number to get the total sales per campaign. This is then joined to the campaign_desc table to link with the campaign information while keeping the whole campaign info from the previous query (right join).



```
hive> SELECT tmp.campaign, total_value, description,start_day,end_day, (end_day-start_day) as last_days
    > FROM
    > campaign_desc c
    > RIGHT JOIN
    > (SELECT campaign, sum(sales_value) as total_value
    > FROM transaction t
    > INNER JOIN
    > coupon_redempt cr
    > ON t.household_key = cr.household_key and t.day = cr.day
    > GROUP BY campaign) tmp
    > ON c.campaign = tmp.campaign
    > ORDER BY total_value DESC;
Query ID = root_20211125164106_5a301a75-f590-4365-bdb1-300e11ec13f5
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1637766516916_0017)

----------------------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      9          9        0        0       0       0
Map 4 ..........   SUCCEEDED      1          1        0        0       0       0
Map 5 ..........   SUCCEEDED      1          1        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
Reducer 3 ......   SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 05/05  [==========================>>] 100%  ELAPSED TIME: 91.83 s
----------------------------------------------------------------------------------------------
OK
18      78480.16967327893       TypeA   587     642     55.0
13      74923.95979427546       TypeA   504     551     47.0
8       39087.329840546474      TypeA   412     460     48.0
26      10501.529964849353      TypeA   224     264     40.0
17      7856.639978066087       TypeB   575     607     32.0
23      7317.879988960922       TypeB   646     684     38.0
22      5956.809986650944       TypeB   624     656     32.0
25      5898.519975185394       TypeB   659     691     32.0
16      5679.249978095293       TypeB   561     593     32.0
30      5550.999983474612       TypeA   323     369     46.0
14      5302.659990489483       TypeC   531     596     65.0
20      4768.52996841073        TypeC   615     685     70.0
9       4418.369988203049       TypeB   435     467     32.0
12      4148.169979020953       TypeB   477     509     32.0
19      3326.8099823594093      TypeB   603     635     32.0
29      1915.9700000882149      TypeB   281     334     53.0
5       1416.6799924075603      TypeB   377     411     34.0
10      1316.7799952700734      TypeB   463     495     32.0
11      1021.3799973353744      TypeB   477     523     46.0
24      971.5299966335297       TypeB   659     719     60.0
7       694.9099982976913       TypeB   398     432     34.0
4       659.3799972236156       TypeB   372     404     32.0
21      650.2699986100197       TypeB   624     656     32.0
2       489.5300007760525       TypeB   351     383     32.0
27      221.6599993109703       TypeC   237     300     63.0
15      133.09000077843666      TypeC   547     708     161.0
28      101.25999891757965      TypeB   259     320     61.0
3       63.62999975681305       TypeC   356     412     56.0
1       60.19999969005585       TypeB   346     383     37.0
6       10.559999942779541      TypeC   393     425     32.0
CAMPAIGN        NULL    DESCRIPTION     START_DAY       END_DAY NULL
Time taken: 93.049 seconds, Fetched: 31 row(s)
hive>
```

According to the total sales value, No.18 campaign has the best performance. Top 4 campaigns are mostly type A which last around 45-50 days. Type B campaigns which have length of around 30 days performed in the medium level. Campaigns longer than 55 days (Type C) are not recommended.

Calculate total quantities per campaign by descending order.



```
hive> SELECT tmp.campaign, total_quantity, description,start_day,end_day, (end_day-sta
rt_day) as last_days
    > FROM
    > campaign_desc c
    > RIGHT JOIN
    > (SELECT campaign, sum(quantity) as total_quantity
    > FROM transaction t
    > INNER JOIN
    > coupon_redempt cr
    > ON t.household_key = cr.household_key and t.day = cr.day
    > GROUP BY campaign) tmp
    > ON c.campaign = tmp.campaign
    > ORDER BY total_quantity DESC;
Query ID = root_20211125164514_38eb55cf-e08d-4330-a4cd-1dc73c18cbdb
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1637766516916_0017)

----------------------------------------------------------------------------------------------
        VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 ..........   SUCCEEDED      9          9        0        0       0       0
Map 4 ..........   SUCCEEDED      1          1        0        0       0       0
Map 5 ..........   SUCCEEDED      1          1        0        0       0       0
Reducer 2 ......   SUCCEEDED      1          1        0        0       0       0
Reducer 3 ......   SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 05/05  [==========================>>] 100%  ELAPSED TIME: 50.34 s
----------------------------------------------------------------------------------------------
OK
13      868953  TypeA   504     551     47.0
18      844892  TypeA   587     642     55.0
8       397183  TypeA   412     460     48.0
16      140980  TypeB   561     593     32.0
19      133062  TypeB   603     635     32.0
22      129783  TypeB   624     656     32.0
26      91644   TypeA   224     264     40.0
17      79167   TypeB   575     607     32.0
23      63117   TypeB   646     684     38.0
25      51650   TypeB   659     691     32.0
4       35130   TypeB   372     404     32.0
7       34499   TypeB   398     432     34.0
12      30082   TypeB   477     509     32.0
29      29950   TypeB   281     334     53.0
10      29360   TypeB   463     495     32.0
30      28590   TypeA   323     369     46.0
11      19026   TypeB   477     523     46.0
27      15441   TypeC   237     300     63.0
20      13974   TypeC   615     685     70.0
24      13322   TypeB   659     719     60.0
21      8875    TypeB   624     656     32.0
14      2369    TypeC   531     596     65.0
9       1948    TypeB   435     467     32.0
5       473     TypeB   377     411     34.0
2       296     TypeB   351     383     32.0
15      65      TypeC   547     708     161.0
28      40      TypeB   259     320     61.0
1       30      TypeB   346     383     37.0
3       27      TypeC   356     412     56.0
6       4       TypeC   393     425     32.0
CAMPAIGN        NULL    DESCRIPTION     START_DAY       END_DAY NULL
Time taken: 51.573 seconds, Fetched: 31 row(s)
```

As for the total quantities, No. 13 has the largest volume among all the campaign. By comparing the top campaigns between sales and quantities. Some campaigns, e.g. no. 18, 26, 17,23,25, are more effective on valuable products (with higher price). As these are also top campaigns, we could conclude products with higher price are sensitive to campaigns.

In conclusion, No. 18 is the most successful campaign due to its largest sales contribution. In addition, it may consist of premium products that are sensitive to the marketing campaign.

Q6. What are the characteristics of customers who were attracted by each marketing campaign?

Datasets : **hh_demographic.csv**, **coupon_redempt.csv**

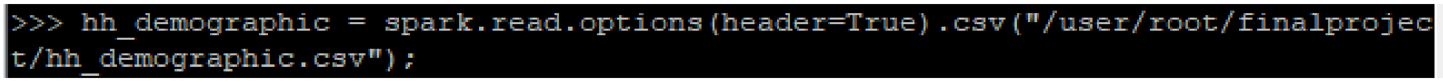
Tools:

- Pysark : create tables, query tables, output csv file
- Hadoop Distributed File System (HDFS) : store distributed data, output query result
- Elasticsearch and Kibana: visualize query result

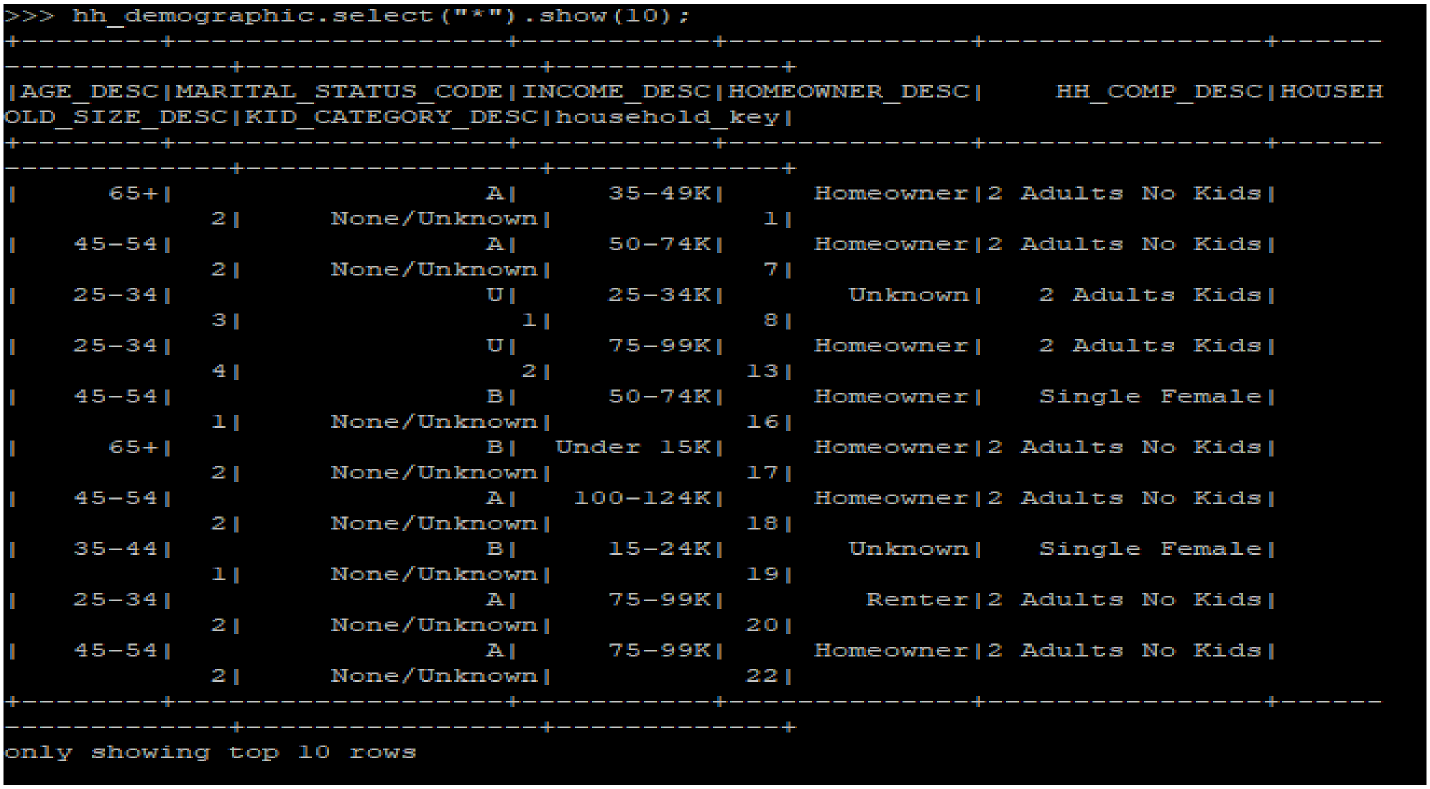
In order to get the information of customers for each marketing campaign, we can look into the coupon_redempt table, then link these customers with their demographic info from the hh_demographic table.

1. Create RDDs in Pyspark:

Create hh_demographic RDD:

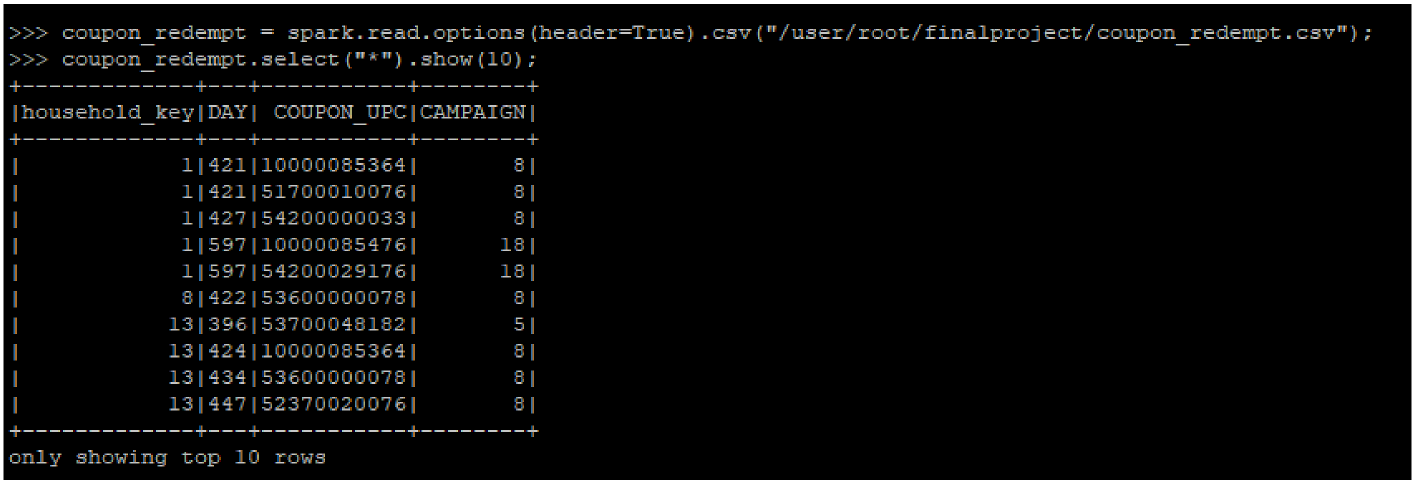


```
>>> hh_demographic = spark.read.options(header=True).csv("/user/root/finalprojec
t/hh_demographic.csv");
```



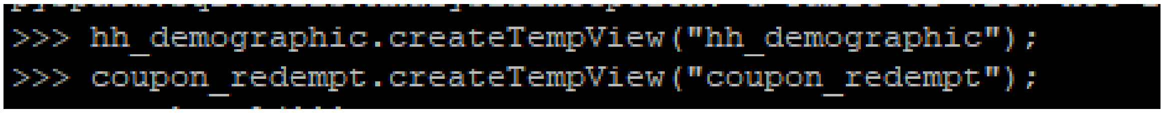
```
>>> hh_demographic.select("*").show(10);
+--------+-------------------+-----------+--------------+----------------+------
-------------+-----------------+-------------+
|AGE_DESC|MARITAL_STATUS_CODE|INCOME_DESC|HOMEOWNER_DESC|    HH_COMP_DESC|HOUSEH
OLD_SIZE_DESC|KID_CATEGORY_DESC|household_key|
+--------+-------------------+-----------+--------------+----------------+------
-------------+-----------------+-------------+
|     65+|                  A|     35-49K|     Homeowner|2 Adults No Kids|
            2|     None/Unknown|            1|
|   45-54|                  A|     50-74K|     Homeowner|2 Adults No Kids|
            2|     None/Unknown|            7|
|   25-34|                  U|     25-34K|       Unknown|   2 Adults Kids|
            3|                1|            8|
|   25-34|                  U|     75-99K|     Homeowner|   2 Adults Kids|
            4|                2|           13|
|   45-54|                  B|     50-74K|     Homeowner|   Single Female|
            1|     None/Unknown|           16|
|     65+|                  B|  Under 15K|     Homeowner|2 Adults No Kids|
            2|     None/Unknown|           17|
|   45-54|                  A|   100-124K|     Homeowner|2 Adults No Kids|
            2|     None/Unknown|           18|
|   35-44|                  B|     15-24K|       Unknown|   Single Female|
            1|     None/Unknown|           19|
|   25-34|                  A|     75-99K|        Renter|2 Adults No Kids|
            2|     None/Unknown|           20|
|   45-54|                  A|     75-99K|     Homeowner|2 Adults No Kids|
            2|     None/Unknown|           22|
+--------+-------------------+-----------+--------------+----------------+------
-------------+-----------------+-------------+
only showing top 10 rows
```

Create coupon_redempt RDD:



```
>>> coupon_redempt = spark.read.options(header=True).csv("/user/root/finalproject/coupon_redempt.csv");
>>> coupon_redempt.select("*").show(10);
+-------------+---+-----------+--------+
|household_key|DAY| COUPON_UPC|CAMPAIGN|
+-------------+---+-----------+--------+
|            1|421|10000085364|       8|
|            1|421|51700010076|       8|
|            1|427|54200000033|       8|
|            1|597|10000085476|      18|
|            1|597|54200029176|      18|
|            8|422|53600000078|       8|
|           13|396|53700048182|       5|
|           13|424|10000085364|       8|
|           13|434|53600000078|       8|
|           13|447|52370020076|       8|
+-------------+---+-----------+--------+
only showing top 10 rows
```

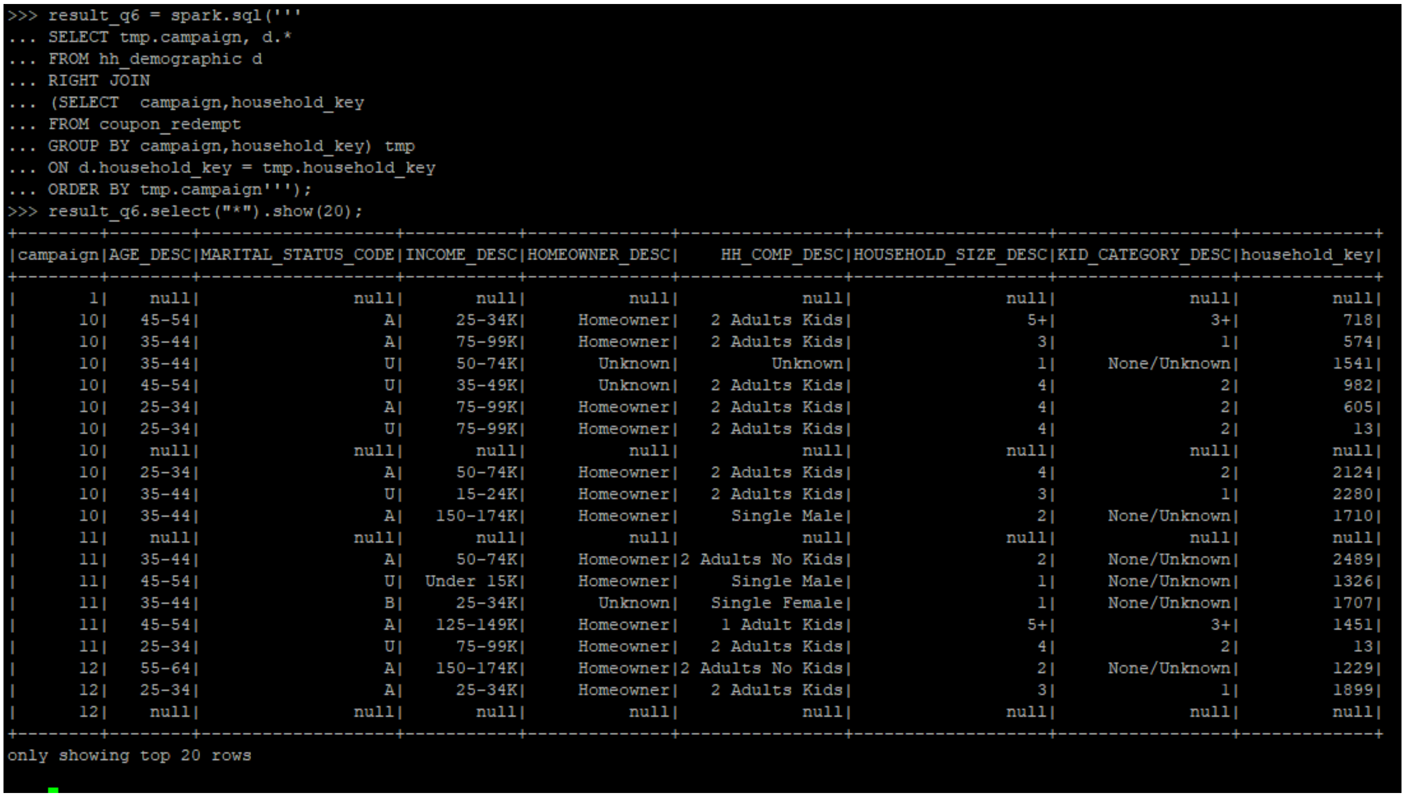
Before querying, pyspark needs to convert RDD to View:



```
>>> hh_demographic.createTempView("hh_demographic");
>>> coupon_redempt.createTempView("coupon_redempt");
```

2. Find characteristics of customers in each marketing campaign.

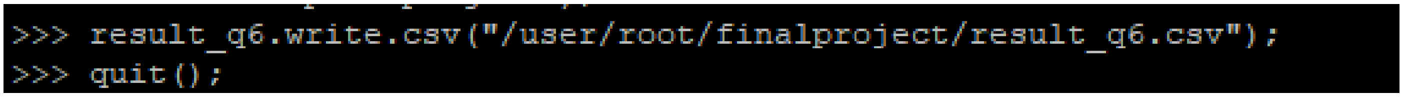
To get the information of customers for each campaign, we group the data by campaign number and household ID in coupon_redempt table. Then join the customer demographic info with keeping all the campaign number from previous query result.



```
>>> result_q6 = spark.sql('''
... SELECT tmp.campaign, d.*
... FROM hh_demographic d
... RIGHT JOIN
... (SELECT  campaign,household_key
... FROM coupon_redempt
... GROUP BY campaign,household_key) tmp
... ON d.household_key = tmp.household_key
... ORDER BY tmp.campaign''');
>>> result_q6.select("*").show(20);
+--------+--------+-------------------+-----------+--------------+----------------+-------------------+-----------------+-------------+
|campaign|AGE_DESC|MARITAL_STATUS_CODE|INCOME_DESC|HOMEOWNER_DESC|    HH_COMP_DESC|HOUSEHOLD_SIZE_DESC|KID_CATEGORY_DESC|household_key|
+--------+--------+-------------------+-----------+--------------+----------------+-------------------+-----------------+-------------+
|       1|    null|               null|       null|          null|            null|               null|             null|         null|
|      10|   45-54|                  A|     25-34K|     Homeowner|   2 Adults Kids|                 5+|               3+|          718|
|      10|   35-44|                  A|     75-99K|     Homeowner|   2 Adults Kids|                  3|                1|          574|
|      10|   35-44|                  U|     50-74K|       Unknown|         Unknown|                  1|     None/Unknown|         1541|
|      10|   45-54|                  U|     35-49K|       Unknown|   2 Adults Kids|                  4|                2|          982|
|      10|   25-34|                  A|     75-99K|     Homeowner|   2 Adults Kids|                  4|                2|          605|
|      10|   25-34|                  U|     75-99K|     Homeowner|   2 Adults Kids|                  4|                2|           13|
|      10|    null|               null|       null|          null|            null|               null|             null|         null|
|      10|   25-34|                  A|     50-74K|     Homeowner|   2 Adults Kids|                  4|                2|         2124|
|      10|   35-44|                  U|     15-24K|     Homeowner|   2 Adults Kids|                  3|                1|         2280|
|      10|   35-44|                  A|   150-174K|     Homeowner|     Single Male|                  2|     None/Unknown|         1710|
|      11|    null|               null|       null|          null|            null|               null|             null|         null|
|      11|   35-44|                  A|     50-74K|     Homeowner|2 Adults No Kids|                  2|     None/Unknown|         2489|
|      11|   45-54|                  U|  Under 15K|     Homeowner|     Single Male|                  1|     None/Unknown|         1326|
|      11|   35-44|                  B|     25-34K|       Unknown|   Single Female|                  1|     None/Unknown|         1707|
|      11|   45-54|                  A|   125-149K|     Homeowner|    1 Adult Kids|                 5+|               3+|         1451|
|      11|   25-34|                  U|     75-99K|     Homeowner|   2 Adults Kids|                  4|                2|           13|
|      12|   55-64|                  A|   150-174K|     Homeowner|2 Adults No Kids|                  2|     None/Unknown|         1229|
|      12|   25-34|                  A|     25-34K|     Homeowner|   2 Adults Kids|                  3|                1|         1899|
|      12|    null|               null|       null|          null|            null|               null|             null|         null|
+--------+--------+-------------------+-----------+--------------+----------------+-------------------+-----------------+-------------+
only showing top 20 rows
```

3. Output query result in Pysark

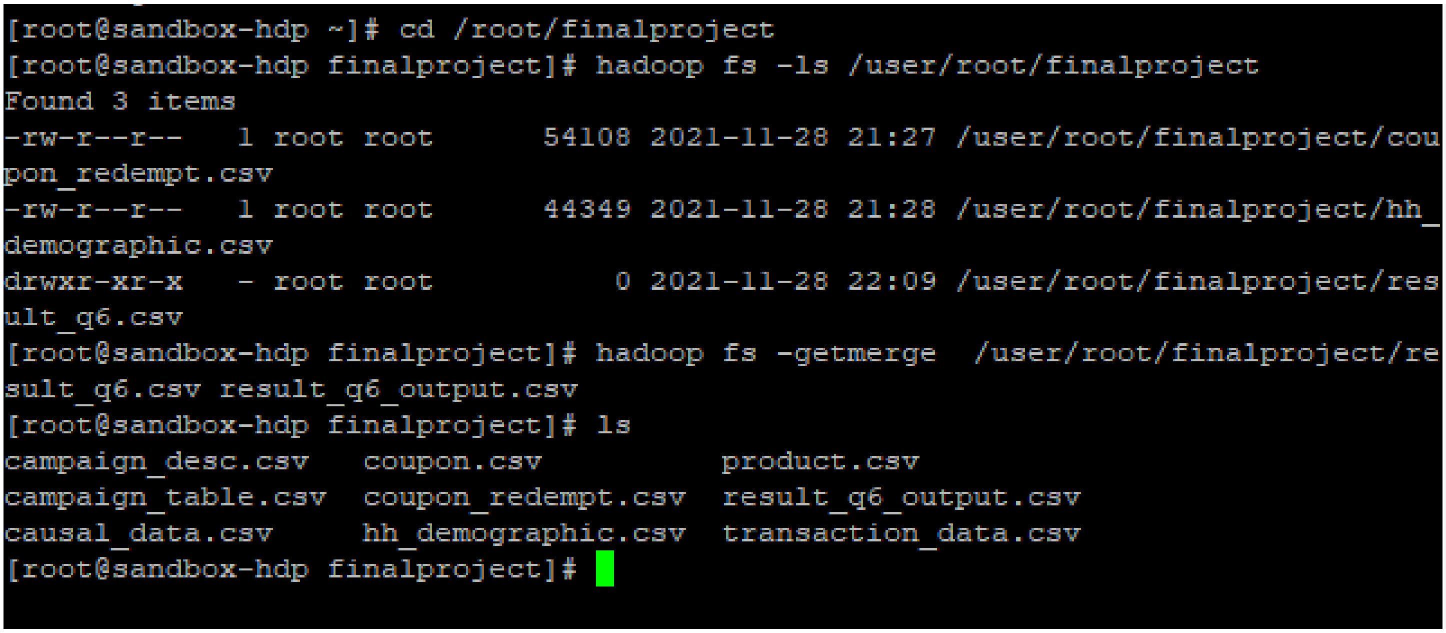
Write the result into csv file.



```
>>> result_q6.write.csv("/user/root/finalproject/result_q6.csv");
>>> quit();
```

4. Output result from HDFS

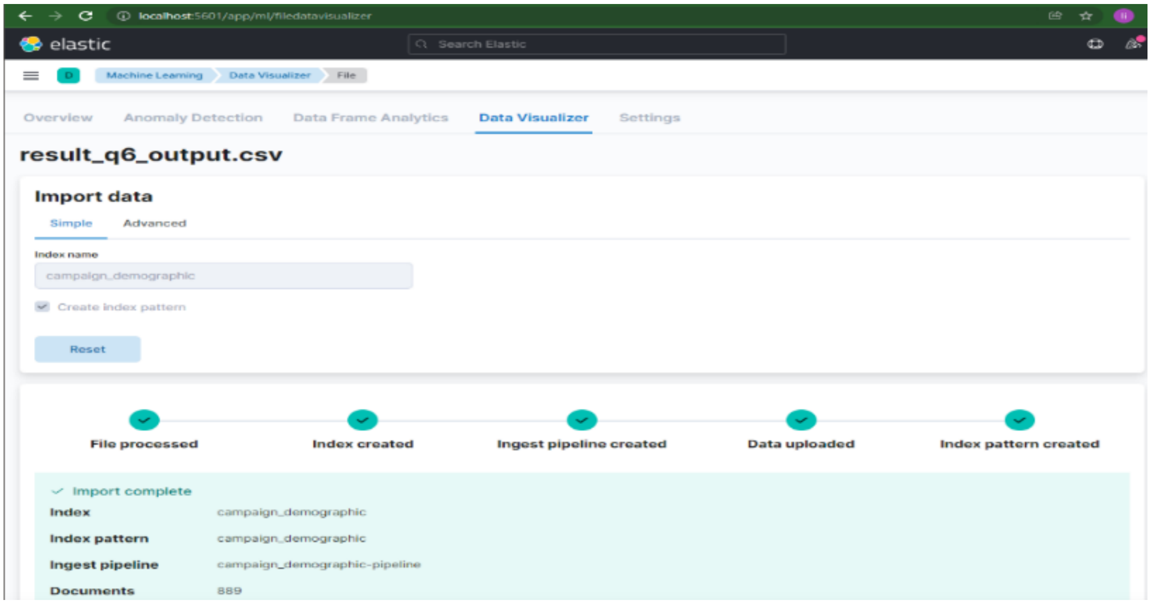
In order to use csv file in other tool, we merge all the output csv files into one csv file, and ouput it from HDFS directory to local directory.



```
[root@sandbox-hdp ~]# cd /root/finalproject
[root@sandbox-hdp finalproject]# hadoop fs -ls /user/root/finalproject
Found 3 items
-rw-r--r--   1 root root      54108 2021-11-28 21:27 /user/root/finalproject/cou
pon_redempt.csv
-rw-r--r--   1 root root      44349 2021-11-28 21:28 /user/root/finalproject/hh_
demographic.csv
drwxr-xr-x   - root root          0 2021-11-28 22:09 /user/root/finalproject/res
ult_q6.csv
[root@sandbox-hdp finalproject]# hadoop fs -getmerge  /user/root/finalproject/re
sult_q6.csv result_q6_output.csv
[root@sandbox-hdp finalproject]# ls
campaign_desc.csv   coupon.csv          product.csv
campaign_table.csv  coupon_redempt.csv  result_q6_output.csv
causal_data.csv     hh_demographic.csv  transaction_data.csv
[root@sandbox-hdp finalproject]# 
```

5. Data visualization in Kibana

Import query result into Kibana.



result_q6_output.csv

Import data

Index name
campaign_demographic

☑ Create index pattern

✓ Import complete

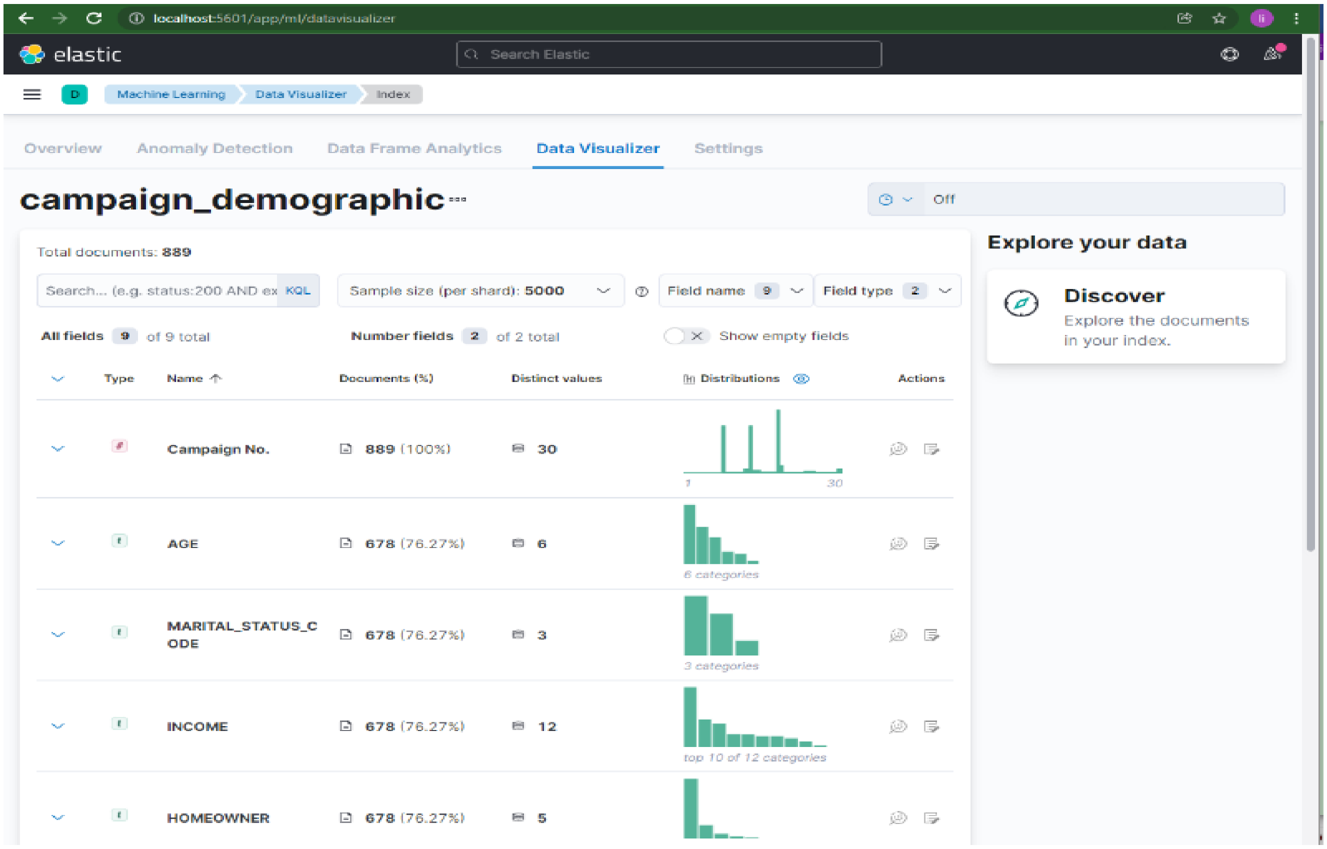
Index campaign_demographic

Index pattern campaign_demographic

Ingest pipeline campaign_demographic-pipeline

Documents 889

Set up Elasticsearch index pattern named campaign_demographic.



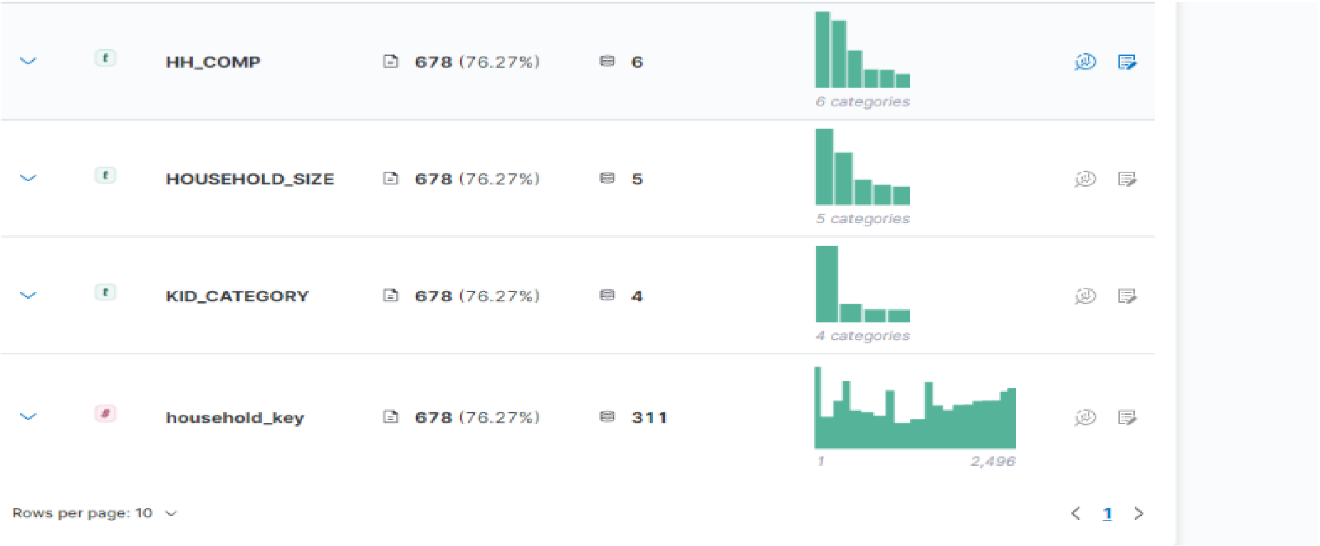
campaign_demographic

Total documents: 889

Search... (e.g. status:200 AND ex KQL | Sample size (per shard): 5000 | Field name 9 | Field type 2

All fields 9 of 9 total | Number fields 2 of 2 total | Show empty fields

Type	Name	Documents (%)	Distinct values	Distributions	Actions
#	Campaign No.	889 (100%)	30	1 – 30	
t	AGE	678 (76.27%)	6	6 categories	
t	MARITAL_STATUS_CODE	678 (76.27%)	3	3 categories	
t	INCOME	678 (76.27%)	12	top 10 of 12 categories	
t	HOMEOWNER	678 (76.27%)	5		



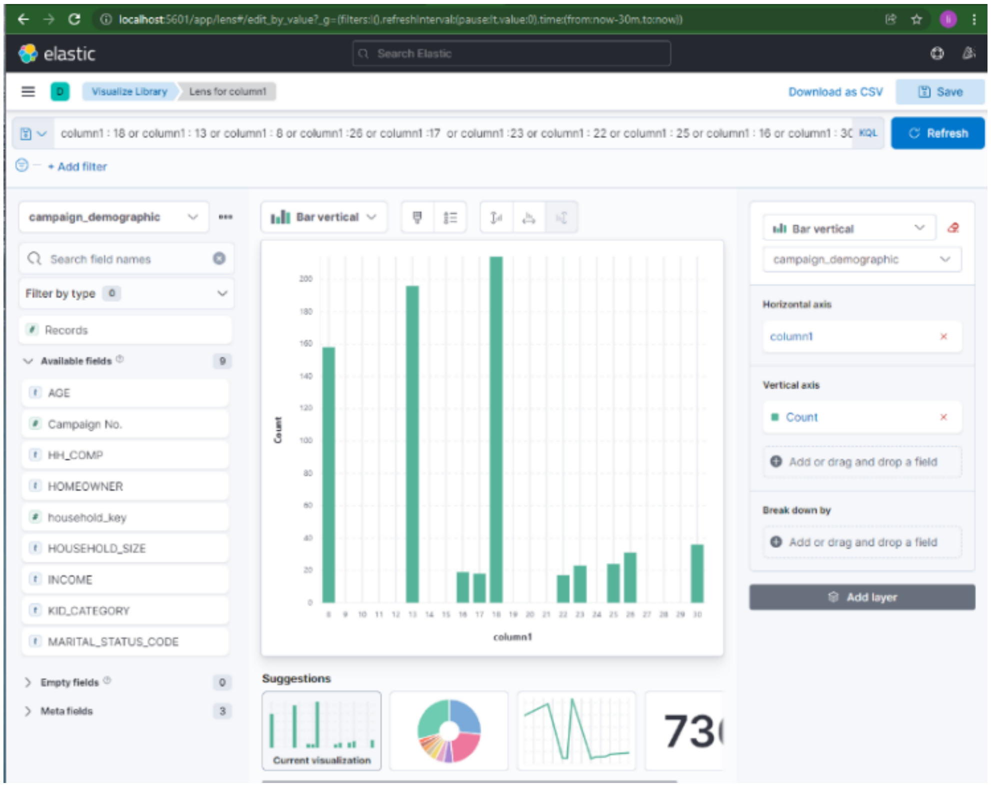
Type	Name	Documents (%)	Distinct values	Distributions	Actions
t	HH_COMP	678 (76.27%)	6	6 categories	
t	HOUSEHOLD_SIZE	678 (76.27%)	5	5 categories	
t	KID_CATEGORY	678 (76.27%)	4	4 categories	
#	household_key	678 (76.27%)	311	1 – 2,496	

Rows per page: 10

Filter data by doing following search:

column1: 18 or column1: 13 or column1: 8 or column1 :26 or column1 :17 or column1 :23 or column1:22 or column1: 25 or column1: 16 or column1: 30

These are top 10 campaigns as we found in Q5.



localhost:5601/app/lens#/edit_by_value?_g=(filters:!(),refreshInterval:(pause:!t,value:0),time:(from:now-30m,to:now))

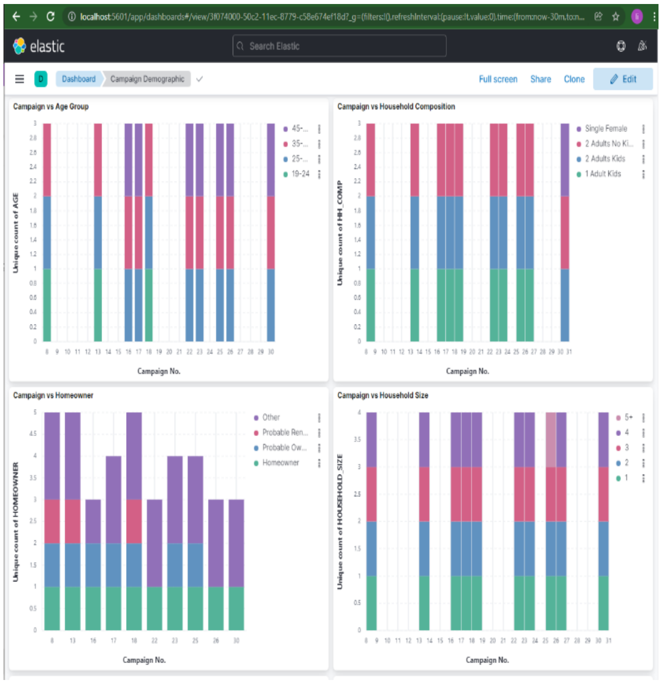
column1 : 18 or column1 : 13 or column1 : 8 or column1 :26 or column1 :17 or column1 :23 or column1 : 22 or column1 : 25 or column1 : 16 or column1 : 30

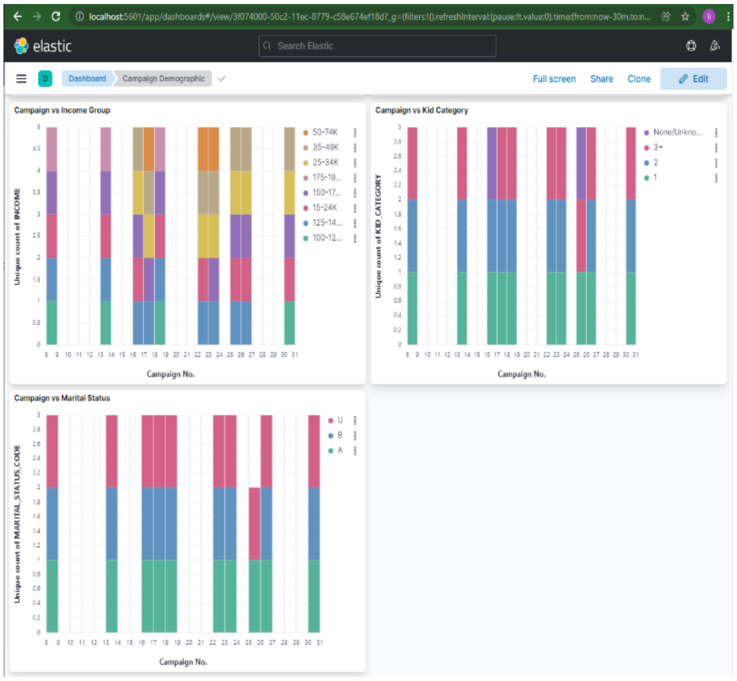
Horizontal axis: column1

Vertical axis: Count

Create a dashboard by illustrating 7 customer characters of top 10 campaigns.

Here is the shared link: http://localhost:5601/goto/52b4ff7b5e9ca80e8999de952e404d27





As we can see from the bar chart, most of the campaign fans are between the ages of 45-54, have families consisting of two adults with no children, rent rather than own, and have incomes between $350,000 and $490,000, if with more than three children.

Insights Description:

- Mostly the customers who spend more on the store are of 45-54 year age group have kids and are married. It could be the reason behind the higher sales of the grocery items in the stores.
- As the grocery covers around 50% of total sales and store has $24,740.1 dollar of highest sale in a day, the store need to originate up with the more ideas to increase the sales of other product.
- Marketing campaigns are more effective on higher price products due to the larger increase of sales than quantities.
- The campaigns with optimal performance mostly last about 45-50 days. We don't recommend more than 55 days.
- Customers between age above 45, with three more children, and income between 35k and 49k are more likely to enrolled in the marketing campaigns.

Future work:

- We will consider causal dataset to exclude the effects of other occurring events, such as in-store display, weekly mailer, for each campaign.

References:

- Datasets: https://www.kaggle.com/frtgnn/dunnhumby-the-complete-journey
- Complete Kibana Tutorial to Visualize and Query Data
  https://phoenixnap.com/kb/kibana-tutorial