# Retail Case Study

A Retail store is required to analyze the day-to-day transactions and keep a track of its customers spread across various locations along with their purchases/returns across various categories. Create a report and display the below calculated metrics, reports and inferences.

## 1. Merge the datasets Customers, Product Hierarchy and Transactions as Customer_Final. Ensure to keep all customers who have done transactions with us and select the join type accordingly.

In [247…

```
import pandas as pd
import numpy as np
import datetime
import seaborn as sn
import matplotlib.pyplot as plt
```

In [89]:

```
Customer=pd.read_csv('Customer.csv')
Product_Hierarchy=pd.read_csv('Prod_cat_info.csv')
Transactions=pd.read_csv('Transactions.csv')
```

In [90]:

```
Customer.head(5)
```

Out[90]:

| | customer_Id | DOB | Gender | city_code |
|---|---|---|---|---|
| **0** | 268408 | 02-01-1970 | M | 4.0 |
| **1** | 269696 | 07-01-1970 | F | 8.0 |
| **2** | 268159 | 08-01-1970 | F | 8.0 |
| **3** | 270181 | 10-01-1970 | F | 2.0 |
| **4** | 268073 | 11-01-1970 | M | 1.0 |

In [91]:

```
Product_Hierarchy.head(5)
```

Out[91]:

| | prod_cat_code | prod_cat | prod_sub_cat_code | prod_subcat |
|---|---|---|---|---|

| | prod_cat_code | prod_cat | prod_sub_cat_code | prod_subcat |
|---|---|---|---|---|
| **0** | 1 | Clothing | 4 | Mens |
| **1** | 1 | Clothing | 1 | Women |
| **2** | 1 | Clothing | 3 | Kids |
| **3** | 2 | Footwear | 1 | Mens |
| **4** | 2 | Footwear | 3 | Women |

In [92]:
```
Transactions.head(5)
```

Out[92]:

| | transaction_id | cust_id | tran_date | prod_subcat_code | prod_cat_code | Qty | Rate | Tax | total_amt | Store_type |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 80712190438 | 270351 | 28-02-2014 | 1 | 1 | -5 | -772 | 405.300 | -4265.300 | e-Shop |
| **1** | 29258453508 | 270384 | 27-02-2014 | 5 | 3 | -5 | -1497 | 785.925 | -8270.925 | e-Shop |
| **2** | 51750724947 | 273420 | 24-02-2014 | 6 | 5 | -2 | -791 | 166.110 | -1748.110 | TeleShop |
| **3** | 93274880719 | 271509 | 24-02-2014 | 11 | 6 | -3 | -1363 | 429.345 | -4518.345 | e-Shop |
| **4** | 51750724947 | 273420 | 23-02-2014 | 6 | 5 | -2 | -791 | 166.110 | -1748.110 | TeleShop |

In [93]:
```
cust1=pd.merge(left = Customer, right = Transactions, left_on = 'customer_Id', right_on = 'cust_id', how='inner')
```

In [94]:
```
Customer_Final=pd.merge(left = cust1, right = Product_Hierarchy, left_on =['prod_cat_code','prod_subcat_code'] , \
                        right_on = ['prod_cat_code','prod_sub_cat_code'], how='inner')
```

In [312…
```
Customer_Final.head(5)
```

Out[312…

| | customer_Id | DOB | Gender | city_code | transaction_id | cust_id | tran_date | prod_subcat_code | prod_cat_code | Qty | Rate | Tax | total_amt | Store_type |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 268408 | 1970-02-01 | M | 4.0 | 87243835584 | 268408 | 2014-01-13 | 7 | 5 | 5 | 187 | 98.175 | 1033.175 | TeleShop |

| | customer_Id | DOB | Gender | city_code | transaction_id | cust_id | tran_date | prod_subcat_code | prod_cat_code | Qty | Rate | Tax | total_amt | Store_type |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 275152 | 1970-01-16 | M | 4.0 | 73109425404 | 275152 | 2011-03-25 | 7 | 5 | 2 | 464 | 97.440 | 1025.440 | e-Shop |
| **2** | 275034 | 1970-01-18 | F | 4.0 | 64777271023 | 275034 | 2011-05-23 | 7 | 5 | 2 | 197 | 41.370 | 435.370 | Flagship store |
| **3** | 270829 | 1970-01-22 | F | 8.0 | 87174343938 | 270829 | 2013-09-12 | 7 | 5 | 4 | 1141 | 479.220 | 5043.220 | e-Shop |
| **4** | 267657 | 1970-01-29 | F | 7.0 | 76242744953 | 267657 | 2013-05-23 | 7 | 5 | 4 | 1020 | 428.400 | 4508.400 | e-Shop |

## 2. Prepare a summary report for the merged data set.

## a. Get the column names and their corresponding data types

In [96]:
```
Customer_Final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 23053 entries, 0 to 23052
Data columns (total 17 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   customer_Id        23053 non-null  int64  
 1   DOB                23053 non-null  object 
 2   Gender             23044 non-null  object 
 3   city_code          23045 non-null  float64
 4   transaction_id     23053 non-null  int64  
 5   cust_id            23053 non-null  int64  
 6   tran_date          23053 non-null  object 
 7   prod_subcat_code   23053 non-null  int64  
 8   prod_cat_code      23053 non-null  int64  
 9   Qty                23053 non-null  int64  
 10  Rate               23053 non-null  int64  
 11  Tax                23053 non-null  float64
 12  total_amt          23053 non-null  float64
 13  Store_type         23053 non-null  object 
 14  prod_cat           23053 non-null  object 
 15  prod_sub_cat_code  23053 non-null  int64  
 16  prod_subcat        23053 non-null  object 
```

```
dtypes: float64(3), int64(8), object(6)
memory usage: 3.2+ MB
```

## b. Top/Bottom 10 observations

In [ ]:
```
Customer_Final.head(10)
```

In [ ]:
```
Customer_Final.tail(10)
```

In [314…
```
Customer_Final_Cont=Customer_Final[['tran_date','Rate','Tax','total_amt','age']]
```

In [309…
```
Customer_Final_Cat=Customer_Final[['Gender','city_code','Store_type','prod_cat','prod_subcat']]
```

## c. “Five-number summary” for continuous variables (min, Q1, median, Q3 and max)

In [244…
```
Customer_Final_Cont.describe().T
```

Out[244…

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **Qty** | 23053.0 | 2.432395 | 2.268406 | -5.000 | 1.00 | 3.00 | 4.000 | 5.0 |
| **Rate** | 23053.0 | 636.369713 | 622.363498 | -1499.000 | 312.00 | 710.00 | 1109.000 | 1500.0 |
| **Tax** | 23053.0 | 248.667192 | 187.177773 | 7.350 | 98.28 | 199.08 | 365.715 | 787.5 |
| **total_amt** | 23053.0 | 2107.308002 | 2507.561264 | -8270.925 | 762.45 | 1754.74 | 3569.150 | 8287.5 |
| **age** | 23053.0 | 40.966078 | 6.628164 | 30.000 | 35.00 | 41.00 | 47.000 | 52.0 |

## d. Frequency tables for all the categorical variables

In [313…
```
Customer_Final_Cat.apply(lambda x: x.value_counts()).T.stack()
```

Out[313…
```
Gender       F                 11233.0
             M                 11811.0
city_code    1.0                2258.0
             2.0                2270.0
```

```
                3.0                  2411.0
                4.0                  2422.0
                5.0                  2360.0
                6.0                  2127.0
                7.0                  2356.0
                8.0                  2330.0
                9.0                  2178.0
                10.0                 2333.0
Store_type      Flagship store       4577.0
                MBR                  4661.0
                TeleShop             4504.0
                e-Shop               9311.0
prod_cat        Bags                 1998.0
                Books                6069.0
                Clothing             2960.0
                Electronics          4898.0
                Footwear             2999.0
                Home and kitchen     4129.0
prod_subcat     Academic              967.0
                Audio and video       952.0
                Bath                 1023.0
                Cameras               985.0
                Children             1035.0
                Comics               1031.0
                Computers             958.0
                DIY                   989.0
                Fiction              1043.0
                Furnishing           1007.0
                Kids                 1997.0
                Kitchen              1037.0
                Mens                 2912.0
                Mobiles              1031.0
                Non-Fiction          1004.0
                Personal Appliances   972.0
                Tools                1062.0
                Women                3048.0
dtype: float64
```

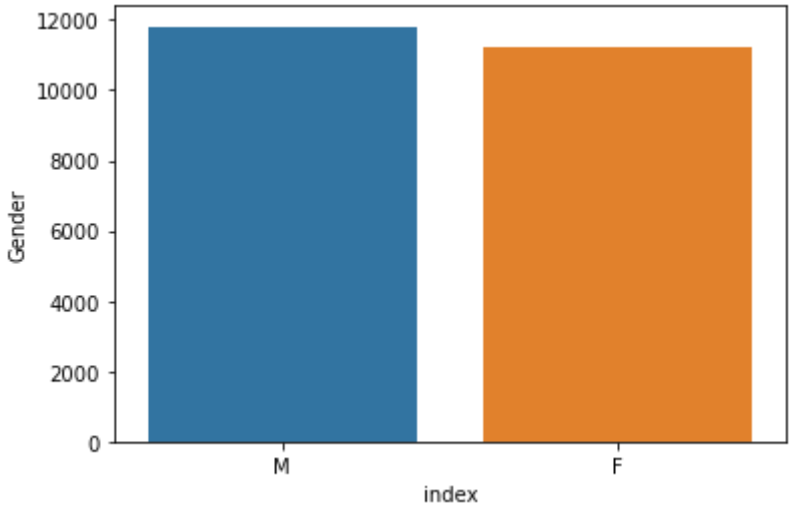
### 3. Generate histograms for all continuous variables and frequency bars for categorical variables.

In [274…

```
sn.barplot(x='index',y='Gender',data=Customer_Final_Cat.Gender.value_counts().reset_index())
```

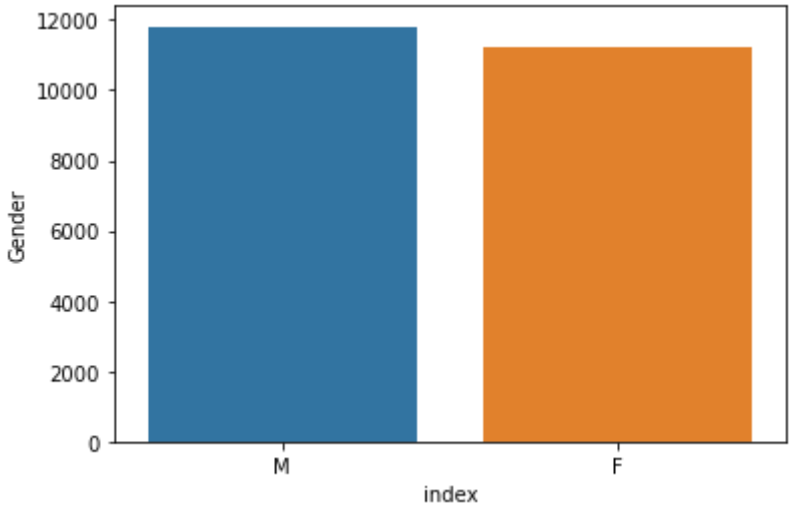
Out[274…

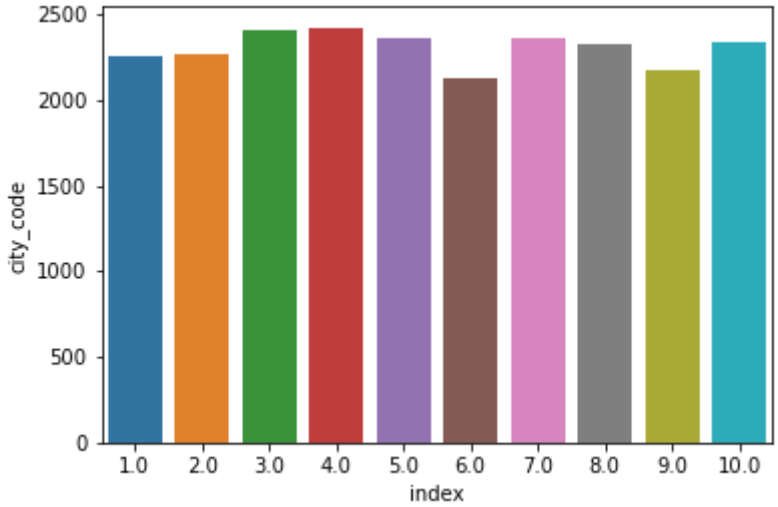
```
<AxesSubplot:xlabel='index', ylabel='Gender'>
```

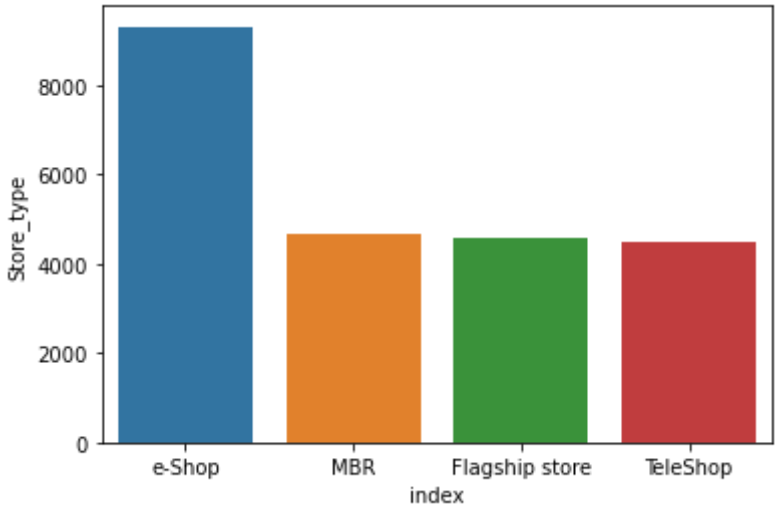


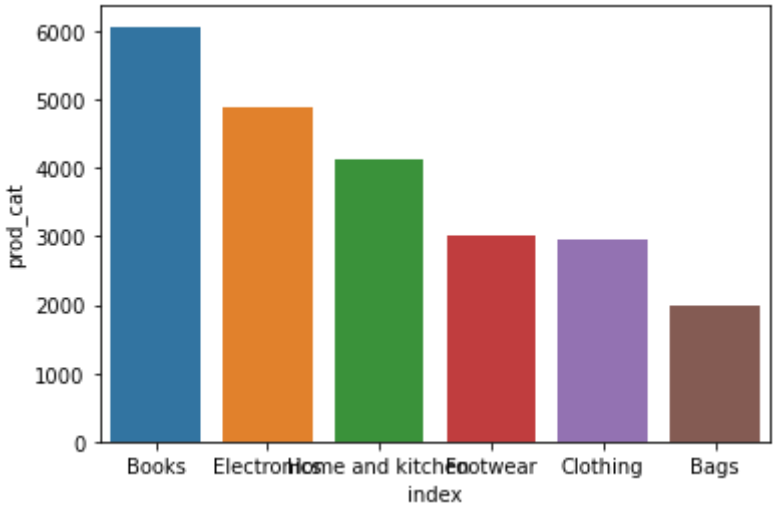

In [281…

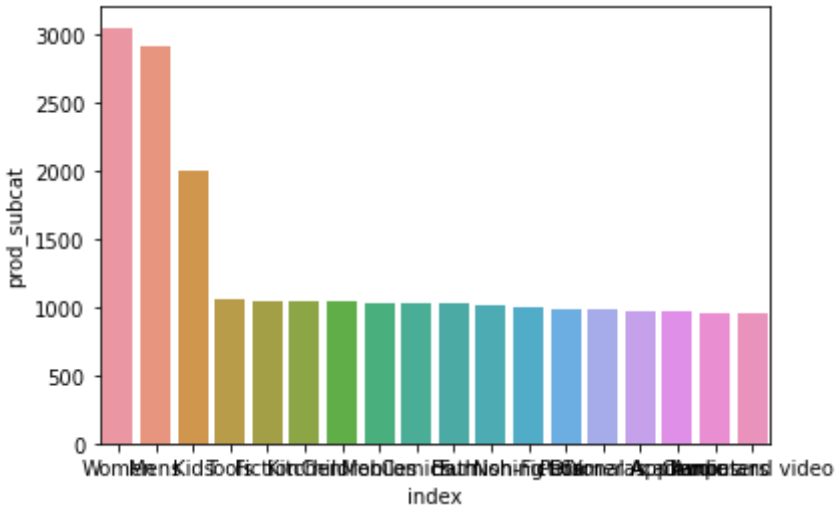
```
for i in Customer_Final_Cat.columns:
    #print (Customer_Final_Cat.loc[:,i])
    sn.barplot(x='index',y=i,data=Customer_Final_Cat.loc[:,i].value_counts().reset_index())
    plt.show()
```




2500
2000
1500
1000
500
0
city_code
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0
index


8000
6000
4000
2000
0
Store_type
e-Shop
MBR
Flagship store
TeleShop
index

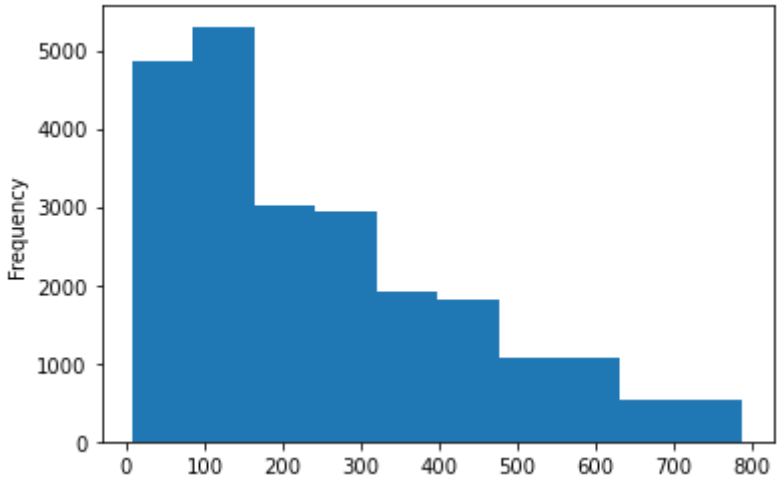



In [317…

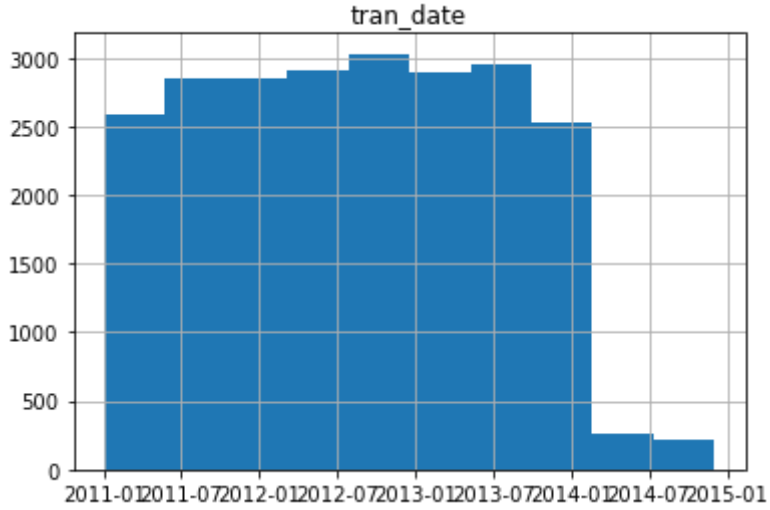
```
Customer_Final_Cont.Tax.plot(kind = 'hist')
```

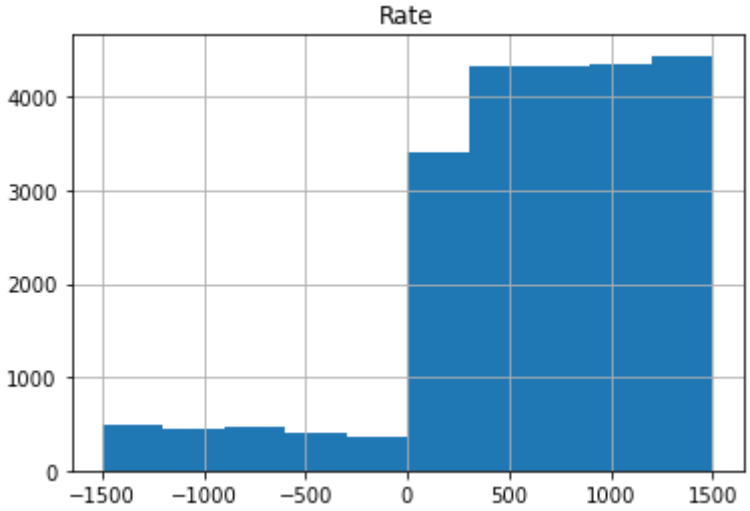
Out[317… <AxesSubplot:ylabel='Frequency'>

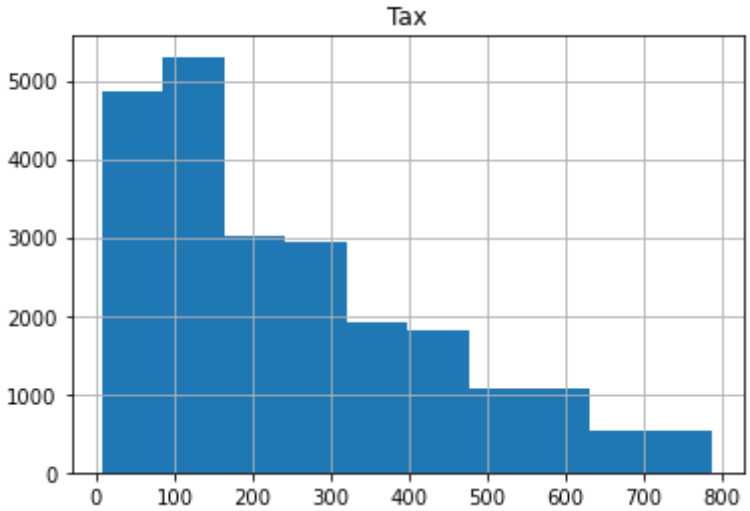


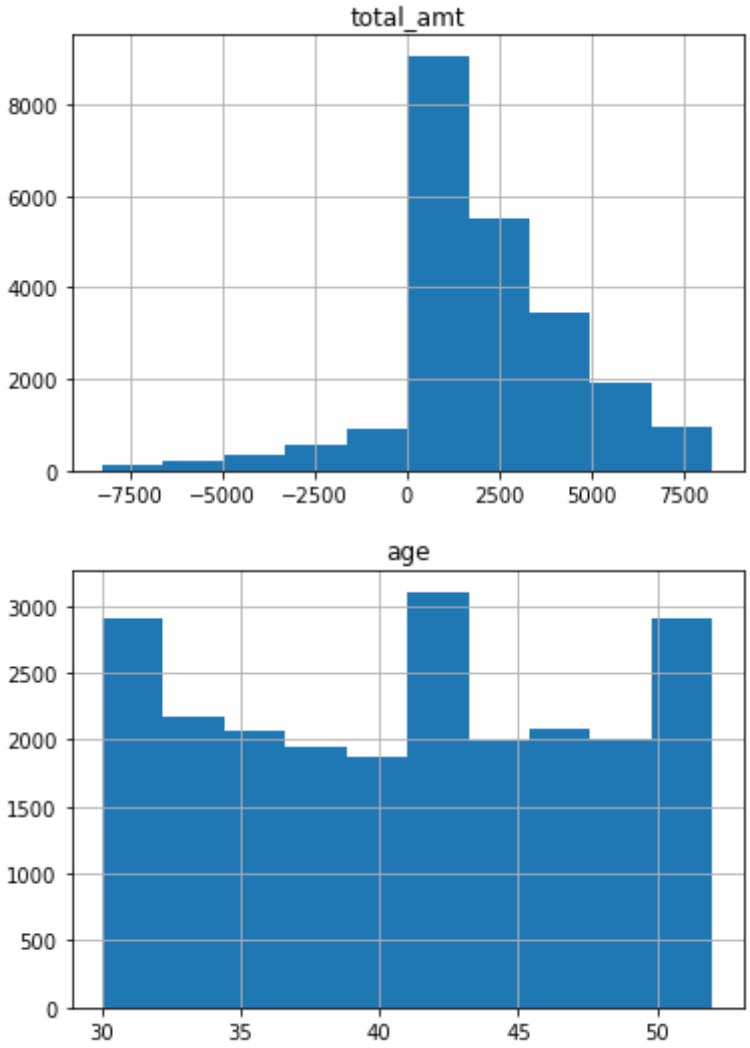

In [318…

```
for i in Customer_Final_Cont.columns:
    Customer_Final_Cont.hist(column=i)
```




Rate
4000
3000
2000
1000
0
−1500
−1000
−500
0
500
1000
1500


Tax
5000
4000
3000
2000
1000
0
0
100
200
300
400
500
600
700
800




## 4. Calculate the following information using the merged dataset :

### a. Time period of the available transaction data

In [100…
```
Customer_Final['tran_date']=pd.to_datetime(Customer_Final['tran_date'])
```

In [101…
```
Customer_Final['tran_date'].max()-Customer_Final['tran_date'].min()
```

Out[101… Timedelta('1430 days 00:00:00')

### b. Count of transactions where the total amount of transaction was negative

In [102…
```
Customer_Final.total_amt[(Customer_Final.total_amt<0)].count()
```

Out[102… 2177

### 5. Analyze which product categories are more popular among females vs male customers.

In [103…
```
Customer_Final.groupby(['Gender','prod_cat']).transaction_id.count()
```

Out[103…
```
Gender  prod_cat
F       Bags                994
        Books               2949
        Clothing            1439
        Electronics         2328
        Footwear            1529
        Home and kitchen    1994
M       Bags                1004
        Books               3116
        Clothing            1518
        Electronics         2570
        Footwear            1469
        Home and kitchen    2134
Name: transaction_id, dtype: int64
```

### 6. Which City code has the maximum customers and what was the percentage of customers from that city?

In [104…
```
Customer_Final.groupby('city_code').customer_Id.count().reset_index().sort_values(by='customer_Id').tail(1)
```

Out[104…

| | city_code | customer_Id |
|---|---|---|
| **3** | 4.0 | 2422 |

In [105…
```
Customer_Final.city_code.value_counts( normalize=True).reset_index().sort_values(by='city_code').tail(1)
```

Out[105…

| | index | city_code |
|---|---|---|

| | index | city_code |
|---|---|---|
| **0** | 4.0 | 0.105099 |

## 7. Which store type sells the maximum products by value and by quantity

In [120…
```
Customer_Final.pivot_table(index='Store_type', aggfunc = {'total_amt': 'sum', 'Qty':'sum'}).\
reset_index().sort_values(by='total_amt',ascending=False)
```

Out[120…

| | Store_type | Qty | total_amt |
|---|---|---|---|
| **3** | e-Shop | 22763 | 1.982482e+07 |
| **0** | Flagship store | 11133 | 9.715688e+06 |
| **1** | MBR | 11194 | 9.674486e+06 |
| **2** | TeleShop | 10984 | 9.364781e+06 |

## 8. What was the total amount earned from the "Electronics" and "Clothing" categories from Flagship Stores?

In [142…
```
Customer_Final[(Customer_Final.Store_type=='Flagship store') & \
               ((Customer_Final.prod_cat=='Electronics') | (Customer_Final.prod_cat=='Clothing'))].\
groupby('prod_cat').total_amt.sum()
```

Out[142…
```
prod_cat
Clothing       1194423.23
Electronics    2215136.04
Name: total_amt, dtype: float64
```

## 9. What was the total amount earned from "Male" customers under the "Electronics" category?

In [149…
```
Customer_Final[(Customer_Final.Gender=='M') & (Customer_Final.prod_cat=='Electronics')].total_amt.sum()
```

Out[149…
```
5703109.425
```

In [153…
```
Customer_Final.groupby(['Gender','prod_cat']).total_amt.sum()
```

Out[153…
```
Gender  prod_cat
F       Bags                2077985.650
        Books               6164692.235
        Clothing            3026750.805
        Electronics         5019354.210
        Footwear            3202552.990
        Home and kitchen    4132177.335
M       Bags                2046722.990
        Books               6645972.775
        Clothing            3224079.495
        Electronics         5703109.425
        Footwear            3014672.050
        Home and kitchen    4301075.480
Name: total_amt, dtype: float64
```

### 10. How many customers have more than 10 unique transactions, after removing all transactions which have any negative amounts?

In [183…
```
cust1=Customer_Final[Customer_Final.total_amt>=0].groupby('cust_id').transaction_id.count().reset_index()
```

In [184…
```
cust1[cust1.transaction_id>10].count()
```

Out[184…
```
cust_id           6
transaction_id    6
dtype: int64
```

### 11. For all customers aged between 25 - 35, find out:

### a. What was the total amount spent for “Electronics” and “Books” product categories?

In [187…
```
Customer_Final.DOB=pd.to_datetime(Customer_Final.DOB)
```

In [208…
```
Customer_Final['age']=pd.Timestamp.now().year-Customer_Final.DOB.dt.year
```

In [206…
```
pd.Timestamp.now().year
```

Out[206…
```
2022
```

In [215…
```
Customer_Final[(Customer_Final.age < 35) & (Customer_Final.age > 25) & \
               ((Customer_Final.prod_cat=='Electronics') | (Customer_Final.prod_cat=='books'))].\
groupby('prod_cat').total_amt.sum()
```

Out[215…
```
prod_cat
Electronics    2272147.41
Name: total_amt, dtype: float64
```

In [217…
```
Customer_Final[(Customer_Final.age < 35) & (Customer_Final.age > 25) & (Customer_Final.prod_cat=='books')].customer_Id.count()
```

Out[217…
```
0
```

### b. What was the total amount spent by these customers between 1st Jan, 2014 to 1st Mar, 2014

In [221…
```
Customer_Final[(Customer_Final.age < 35) & (Customer_Final.age > 25) & \
               (Customer_Final.tran_date<'1st Mar, 2014') & (Customer_Final.tran_date>'1st Jan, 2014')].total_amt.sum()
```

Out[221…
```
340788.63
```