PYTHON PROJECT

CUSTOMER ANALYSIS FOR RETAIL

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import os
```

Import Dataset

In [2]:
```
customer=pd.read_csv('Customer.csv')
transaction=pd.read_csv('Transactions.csv')
pro_cat = pd.read_csv('prod_cat_info.csv')
```

Exploratory Data Analysis

In [3]:
```
customer.head()
```

Out[3]:

	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 [6]:
```
transaction.head()
```

Out[6]:

	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 [7]:
```
pro_cat.head()
```

Out[7]:

	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

1. Merge the datasets Customers, Product Hierarchy and Transactions as Customer_Final.

In [3]:
```
# Renaming 'prod_sub_cat_code' column in 'pro_cat' table to make it similar to 'transaction' table. To merge the both
# the tables without having repeated columns and null values.
pro_cat.rename(columns={"prod_sub_cat_code":"prod_subcat_code"},inplace=True)
```

In [4]:
```
prod_concat = pd.merge(left=transaction, right=pro_cat,on=["prod_cat_code","prod_subcat_code"],how="left")
```

In [5]:
```
prod_concat
```

Out[5]:

	transaction_id	cust_id	tran_date	prod_subcat_code	prod_cat_code	Qty	Rate	Tax	total_amt	Store_type	prod_cat	prod_subcat
0	80712190438	270351	28-02-2014	1	1	-5	-772	405.300	-4265.300	e-Shop	Clothing	Women
1	29258453508	270384	27-02-2014	5	3	-5	-1497	785.925	-8270.925	e-Shop	Electronics	Computers
2	51750724947	273420	24-02-2014	6	5	-2	-791	166.110	-1748.110	TeleShop	Books	DIY
3	93274880719	271509	24-02-2014	11	6	-3	-1363	429.345	-4518.345	e-Shop	Home and kitchen	Bath
4	51750724947	273420	23-02-2014	6	5	-2	-791	166.110	-1748.110	TeleShop	Books	DIY
...	...	...	...	...	...	...	...	...	...	...	...	...
23048	94340757522	274550	25-01-2011	12	5	1	1264	132.720	1396.720	e-Shop	Books	Academic
23049	89780862956	270022	25-01-2011	4	1	1	677	71.085	748.085	e-Shop	Clothing	Mens
23050	85115299378	271020	25-01-2011	2	6	4	1052	441.840	4649.840	MBR	Home and kitchen	Furnishing
23051	72870271171	270911	25-01-2011	11	5	3	1142	359.730	3785.730	TeleShop	Books	Children
23052	77960931771	271961	25-01-2011	11	5	1	447	46.935	493.935	TeleShop	Books	Children

23053 rows × 12 columns

In [6]:
```
prod_concat.isnull().sum()
```

Out[6]:
```
transaction_id      0
cust_id             0
tran_date           0
prod_subcat_code    0
prod_cat_code       0
Qty                 0
Rate                0
Tax                 0
total_amt           0
Store_type          0
prod_cat            0
prod_subcat         0
dtype: int64
```

In [8]:
```
customer_final = pd.merge(left=prod_concat, right=customer,right_on="customer_Id", left_on="cust_id", how="left")
```

In [9]:
```
customer_final
```

Out[9]:

	transaction_id	cust_id	tran_date	prod_subcat_code	prod_cat_code	Qty	Rate	Tax	total_amt	Store_type	prod_cat	prod_subcat	cı
0	80712190438	270351	28-02-2014	1	1	-5	-772	405.300	-4265.300	e-Shop	Clothing	Women	
1	29258453508	270384	27-02-2014	5	3	-5	-1497	785.925	-8270.925	e-Shop	Electronics	Computers	
2	51750724947	273420	24-02-2014	6	5	-2	-791	166.110	-1748.110	TeleShop	Books	DIY	
3	93274880719	271509	24-02-2014	11	6	-3	-1363	429.345	-4518.345	e-Shop	Home and kitchen	Bath	
4	51750724947	273420	23-02-2014	6	5	-2	-791	166.110	-1748.110	TeleShop	Books	DIY	
...	...	...	...	...	...	...	...	...	...	...	...	...	...
23048	94340757522	274550	25-01-2011	12	5	1	1264	132.720	1396.720	e-Shop	Books	Academic	
23049	89780862956	270022	25-01-2011	4	1	1	677	71.085	748.085	e-Shop	Clothing	Mens	
23050	85115299378	271020	25-01-2011	2	6	4	1052	441.840	4649.840	MBR	Home and kitchen	Furnishing	
23051	72870271171	270911	25-01-2011	11	5	3	1142	359.730	3785.730	TeleShop	Books	Children	
23052	77960931771	271961	25-01-2011	11	5	1	447	46.935	493.935	TeleShop	Books	Children	

23053 rows × 16 columns

In [10]:
```
customer_final.shape
```

Out[10]:
```
(23053, 16)
```

In [11]:
```
transaction.shape
```

Out[11]:
```
(23053, 10)
```

Rows of both the final data frame and transaction table has same number of rows.That means all the transaction done are present in final table

In [14]:
```
customer_final.isnull().sum()
```

Out[14]:
```
transaction_id      0
cust_id             0
tran_date           0
prod_subcat_code    0
prod_cat_code       0
Qty                 0
Rate                0
Tax                 0
total_amt           0
Store_type          0
prod_cat            0
prod_subcat         0
customer_Id         0
DOB                 0
Gender              9
city_code           8
dtype: int64
```

In [13]:
```
customer_final.dtypes
```

Out[13]:
```
transaction_id              int64
cust_id                     int64
tran_date                  object
prod_subcat_code            int64
prod_cat_code               int64
Qty                         int64
Rate                        int64
Tax                       float64
total_amt                 float64
Store_type                 object
prod_cat                   object
prod_subcat                object
customer_Id                 int64
DOB                datetime64[ns]
Gender                     object
city_code                 float64
dtype: object
```

In [15]:
```
# there are two columns whose data type are inconvinient...'tran_date' and 'DOB'
customer_final['DOB'] = pd.to_datetime(customer_final['DOB'], )
```

In [16]:
```
customer_final['DOB'].head()
```

Out[16]:
```
0   1981-09-26
1   1973-11-05
2   1992-07-27
3   1981-08-06
4   1992-07-27
Name: DOB, dtype: datetime64[ns]
```

In [17]:
```
customer_final['tran_date'] = pd.to_datetime(customer_final['tran_date'])
```

In [18]:
```
customer_final['tran_date'].head()
```

Out[18]:
```
0   2014-02-28
1   2014-02-27
2   2014-02-24
3   2014-02-24
4   2014-02-23
Name: tran_date, dtype: datetime64[ns]
```

In [19]:
```
# To check duplicate data in the final table
customer_final.duplicated().sum()
```

Out[19]:
```
13
```

In [20]:
```
# Dropping dupicate rows
customer_final.drop_duplicates(inplace=True)
```

In [21]:
```
customer_final.duplicated().sum()
```

Out[21]:
```
0
```

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

a. Get the column names and their corresponding data types

In [23]:
```
customer_final.dtypes
```

Out[23]:
```
transaction_id              int64
cust_id                     int64
tran_date           datetime64[ns]
prod_subcat_code            int64
prod_cat_code               int64
Qty                         int64
Rate                        int64
Tax                       float64
total_amt                 float64
Store_type                 object
prod_cat                   object
prod_subcat                object
customer_Id                 int64
DOB                 datetime64[ns]
Gender                     object
city_code                 float64
dtype: object
```

b. Top/Bottom 10 observations

In [22]:
```
# top 10 observation
customer_final.head(10)
```

Out[22]:

	transaction_id	cust_id	tran_date	prod_subcat_code	prod_cat_code	Qty	Rate	Tax	total_amt	Store_type	prod_cat	prod_subcat	custor
0	80712190438	270351	2014-02-28	1	1	-5	-772	405.300	-4265.300	e-Shop	Clothing	Women	2
1	29258453508	270384	2014-02-27	5	3	-5	-1497	785.925	-8270.925	e-Shop	Electronics	Computers	2
2	51750724947	273420	2014-02-24	6	5	-2	-791	166.110	-1748.110	TeleShop	Books	DIY	2
3	93274880719	271509	2014-02-24	11	6	-3	-1363	429.345	-4518.345	e-Shop	Home and kitchen	Bath	2
4	51750724947	273420	2014-02-23	6	5	-2	-791	166.110	-1748.110	TeleShop	Books	DIY	2
5	97439039119	272357	2014-02-23	8	3	-2	-824	173.040	-1821.040	TeleShop	Electronics	Personal Appliances	2
6	45649838090	273667	2014-02-22	11	6	-1	-1450	152.250	-1602.250	e-Shop	Home and kitchen	Bath	2
7	22643667930	271489	2014-02-22	12	6	-1	-1225	128.625	-1353.625	TeleShop	Home and kitchen	Tools	2
8	79792372943	275108	2014-02-22	3	1	-3	-908	286.020	-3010.020	MBR	Clothing	Kids	2
9	50076728598	269014	2014-02-21	8	3	-4	-581	244.020	-2568.020	e-Shop	Electronics	Personal Appliances	2

In [23]:
```
# bottom 10 observation
customer_final.tail(10)
```

Out[23]:

	transaction_id	cust_id	tran_date	prod_subcat_code	prod_cat_code	Qty	Rate	Tax	total_amt	Store_type	prod_cat	prod_subcat	cu
23043	49882891062	271982	2011-01-25	10	5	4	1330	558.600	5878.600	e-Shop	Books	Non-Fiction	
23044	14787475597	273982	2011-01-25	4	3	5	969	508.725	5353.725	e-Shop	Electronics	Mobiles	
23045	50691119572	273031	2011-01-25	6	5	1	1148	120.540	1268.540	TeleShop	Books	DIY	
23046	40893803228	272049	2011-01-25	11	6	3	1077	339.255	3570.255	e-Shop	Home and kitchen	Bath	
23047	30856003613	266866	2011-01-25	4	2	2	444	93.240	981.240	TeleShop	Footwear	Kids	
23048	94340757522	274550	2011-01-25	12	5	1	1264	132.720	1396.720	e-Shop	Books	Academic	
23049	89780862956	270022	2011-01-25	4	1	1	677	71.085	748.085	e-Shop	Clothing	Mens	
23050	85115299378	271020	2011-01-25	2	6	4	1052	441.840	4649.840	MBR	Home and kitchen	Furnishing	
23051	72870271171	270911	2011-01-25	11	5	3	1142	359.730	3785.730	TeleShop	Books	Children	
23052	77960931771	271961	2011-01-25	11	5	1	447	46.935	493.935	TeleShop	Books	Children	

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

In [24]:
```
customer_final.describe()
```

Out[24]:

	transaction_id	cust_id	prod_subcat_code	prod_cat_code	Qty	Rate	Tax	total_amt	customer_Id	23
count	2.304000e+04	23040.000000	23040.000000	23040.000000	23040.000000	23040.000000	23040.000000	23040.000000	23040.000000	
mean	5.006955e+10	271021.880252	6.148785	3.763498	2.435764	637.094965	248.677488	2109.865226	271021.880252	
std	2.898062e+10	2431.573668	3.726197	1.677091	2.264326	621.727374	187.188311	2505.610295	2431.573668	
min	3.268991e+06	266783.000000	1.000000	1.000000	-5.000000	-1499.000000	7.350000	-8270.925000	266783.000000	
25%	2.493315e+10	268935.000000	3.000000	2.000000	1.000000	312.000000	98.280000	762.450000	268935.000000	
50%	5.009188e+10	270980.500000	5.000000	4.000000	3.000000	710.000000	199.080000	1756.950000	270980.500000	
75%	7.532632e+10	273114.250000	10.000000	5.000000	4.000000	1109.000000	365.767500	3570.255000	273114.250000	
max	9.998755e+10	275265.000000	12.000000	6.000000	5.000000	1500.000000	787.500000	8287.500000	275265.000000	

d. Frequency tables for all the categorical variables

In [26]:
```
customer_final.loc[:,customer_final.dtypes=="object"].describe()
```

Out[26]:

	Store_type	prod_cat	prod_subcat	Gender
count	23040	23040	23040	23031
unique	4	6	18	2
top	e-Shop	Books	Women	M
freq	9304	6066	3046	11804

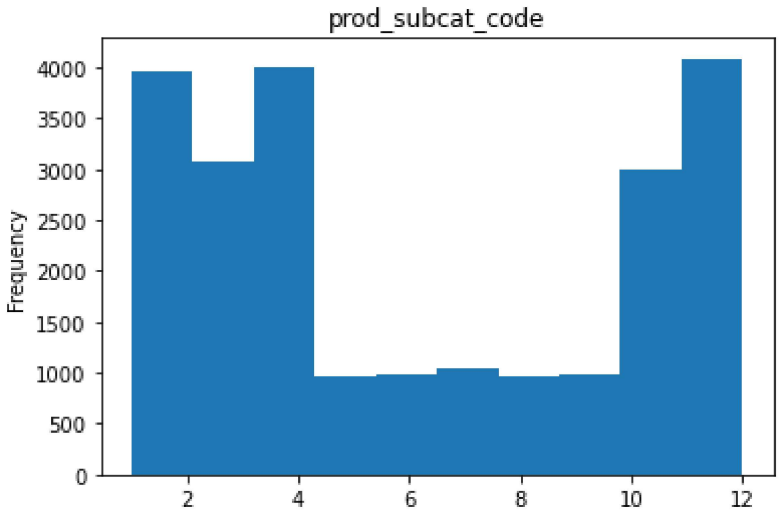
3. Generate histograms for all continuous variables and frequency bars for categorical variables.

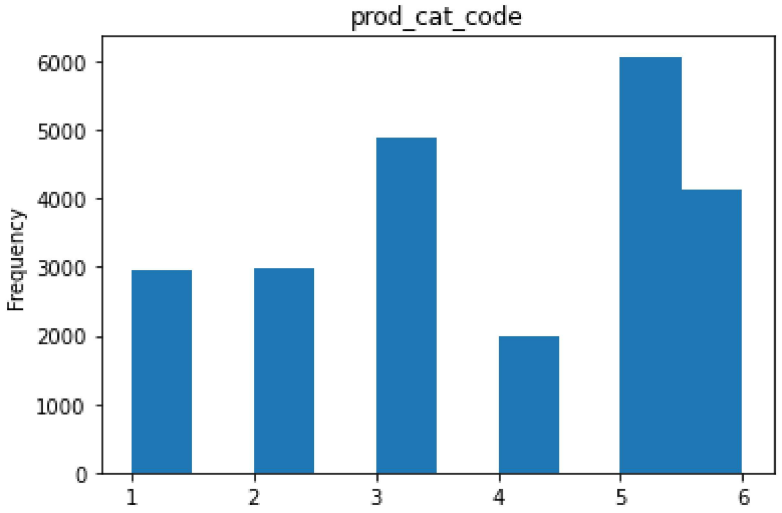
In [27]:
```
continuous_v = customer_final.loc[:,['prod_subcat_code','prod_cat_code', 'Qty', 'Rate', 'Tax', 'total_amt']]
```

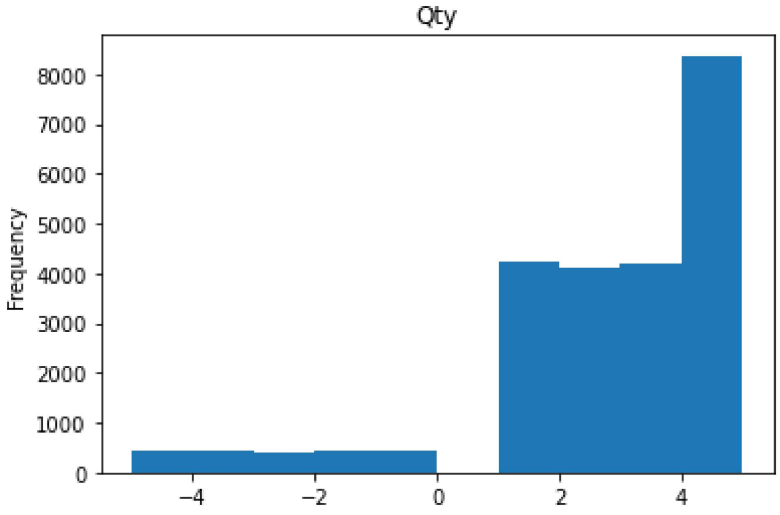
In [28]:
```
continuous_v.columns
```

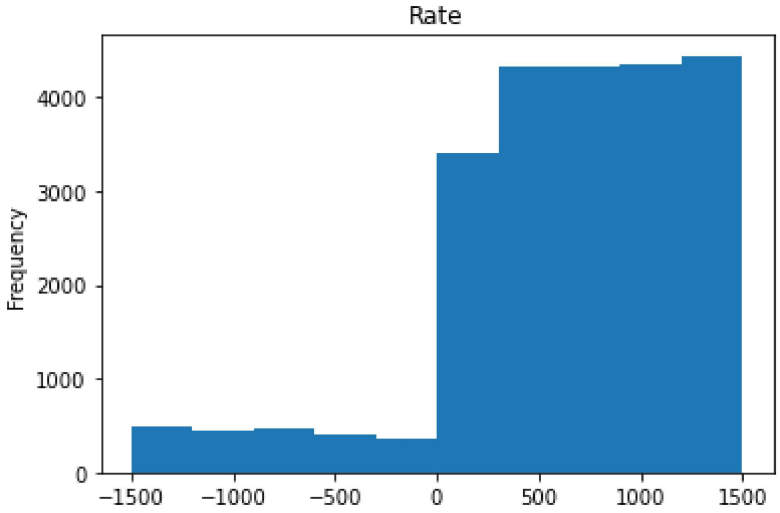
Out[28]:
```
Index(['prod_subcat_code', 'prod_cat_code', 'Qty', 'Rate', 'Tax', 'total_amt'], dtype='object')
```

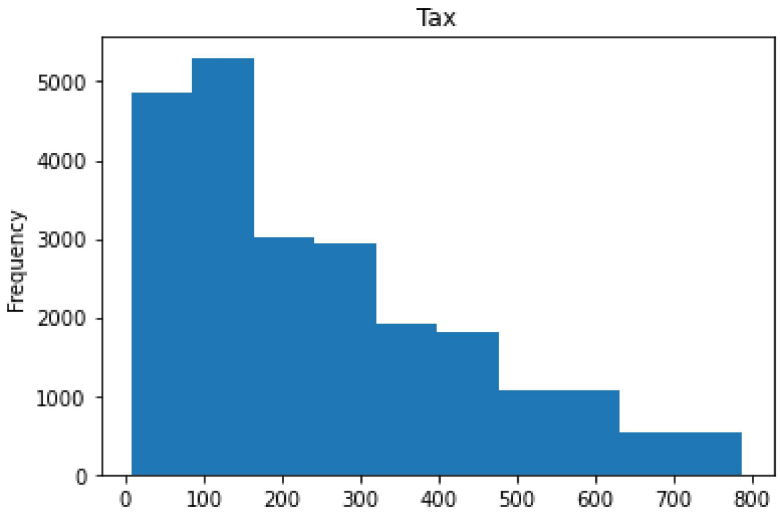
In [36]:
```
for var in continuous_v.columns:
    continuous_v[var].plot(kind='hist')
    plt.title(var)
    plt.show()
```

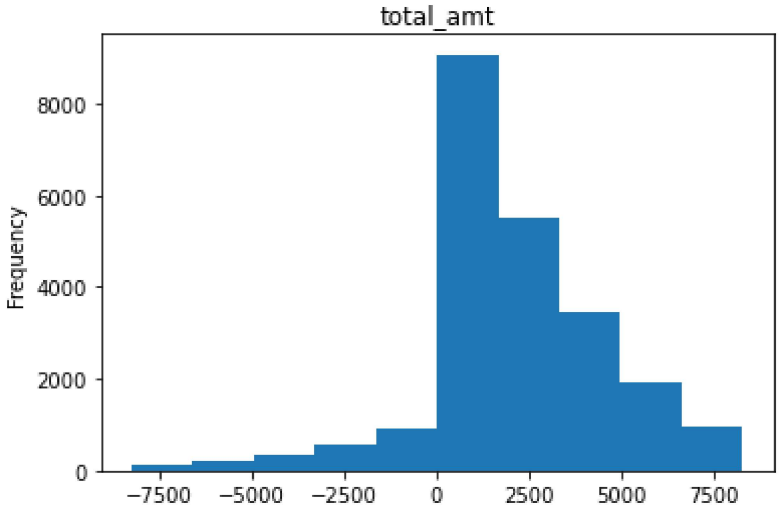





Qty
Frequency
8000
7000
6000
5000
4000
3000
2000
1000
0
−4
−2
0
2
4


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





Bar chart of categorical variables

In [37]:
```
category_customer = customer_final.loc[:,customer_final.dtypes=='object']
```

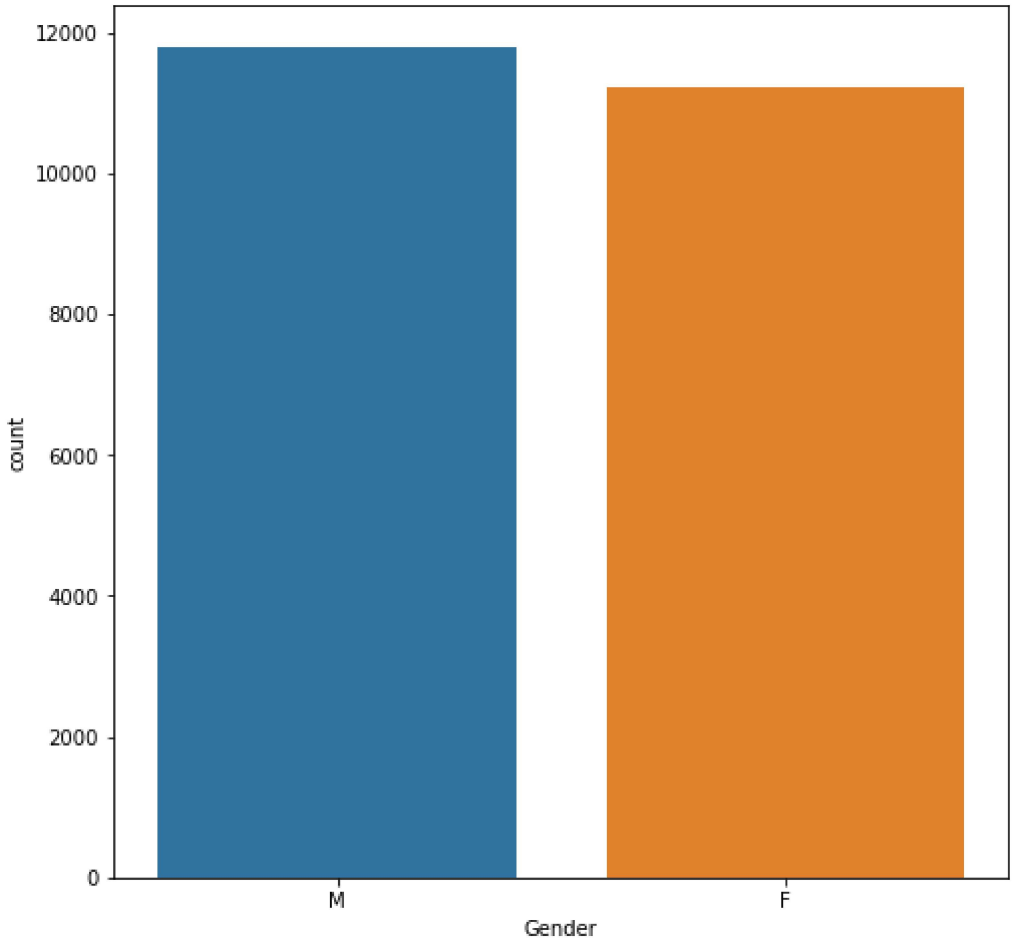
In [38]:
```
category_customer.head()
```

Out[38]:

	Store_type	prod_cat	prod_subcat	Gender
0	e-Shop	Clothing	Women	M
1	e-Shop	Electronics	Computers	F
2	TeleShop	Books	DIY	M
3	e-Shop	Home and kitchen	Bath	M
4	TeleShop	Books	DIY	M

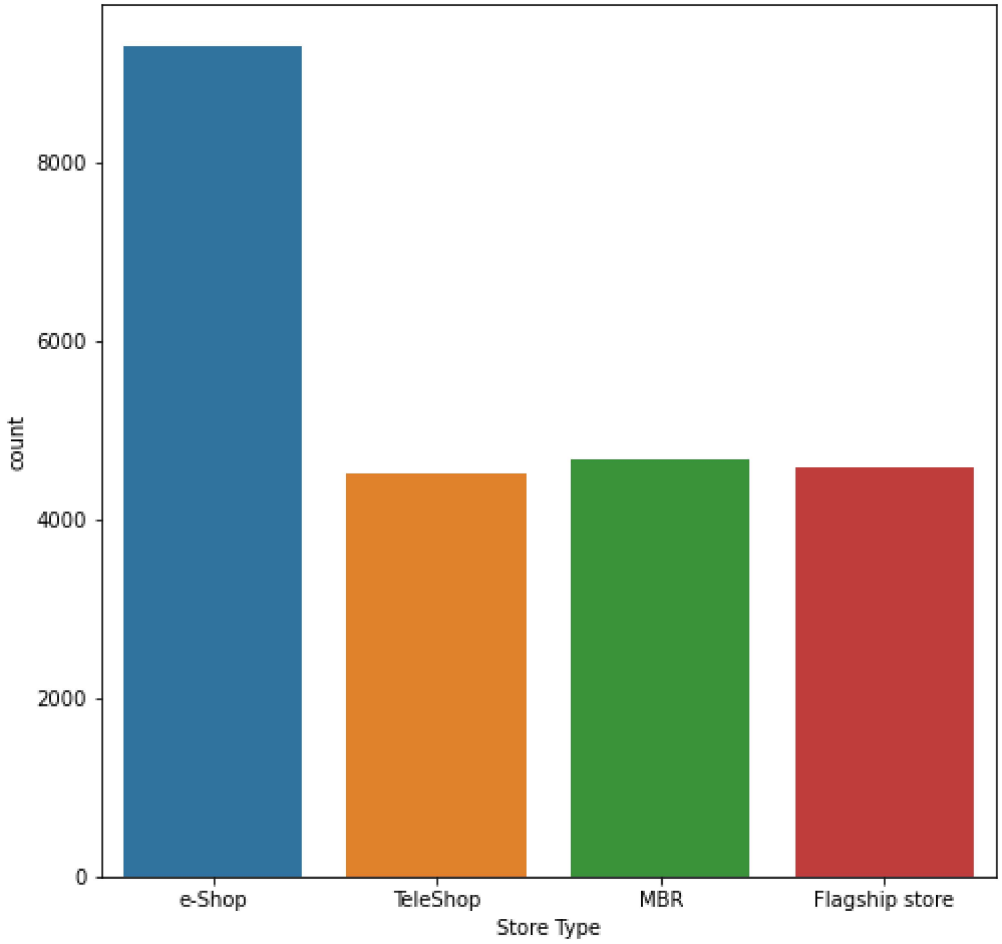
In [40]:
```
plt.figure(figsize=(8,8))
sns.countplot(category_customer['Gender'])
plt.show()
```

```
D:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explici
t keyword will result in an error or misinterpretation.
  warnings.warn(
```



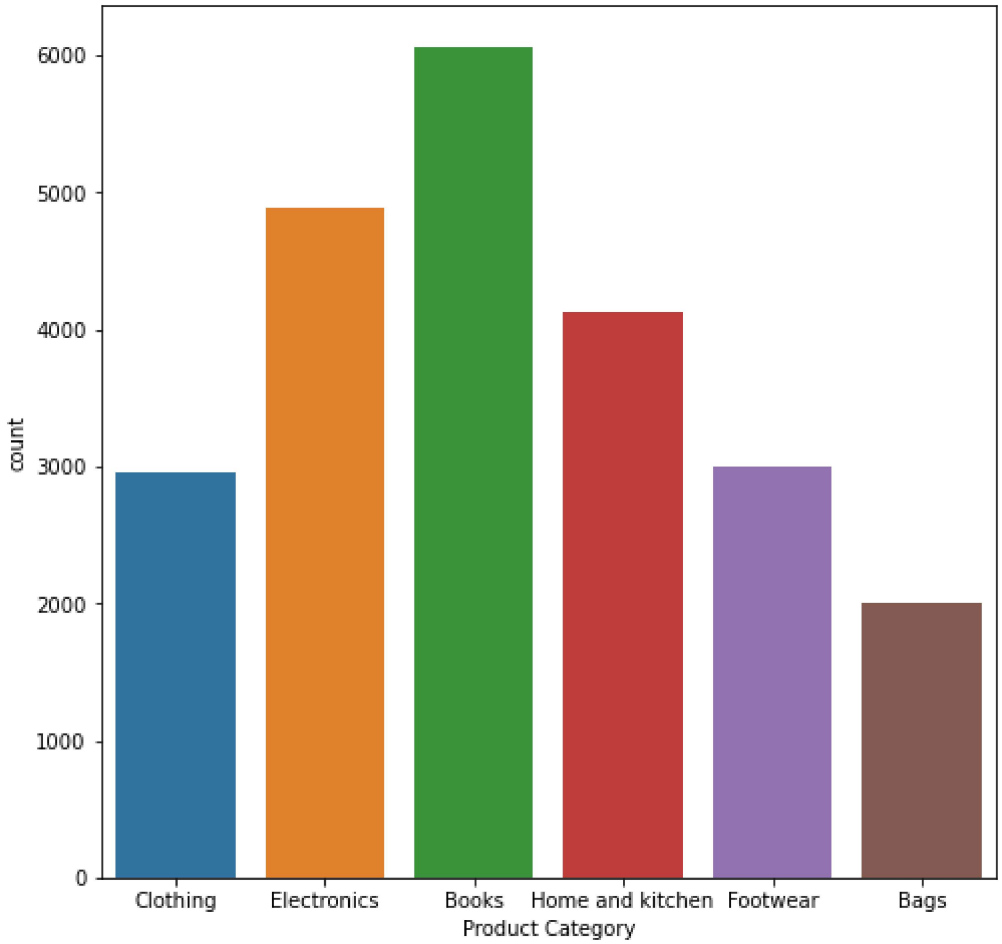
In [41]:
```
plt.figure(figsize=(8,8))
sns.countplot(category_customer['Store_type'])
plt.xlabel('Store Type')
plt.show()
```

```
D:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explici
t keyword will result in an error or misinterpretation.
  warnings.warn(
```

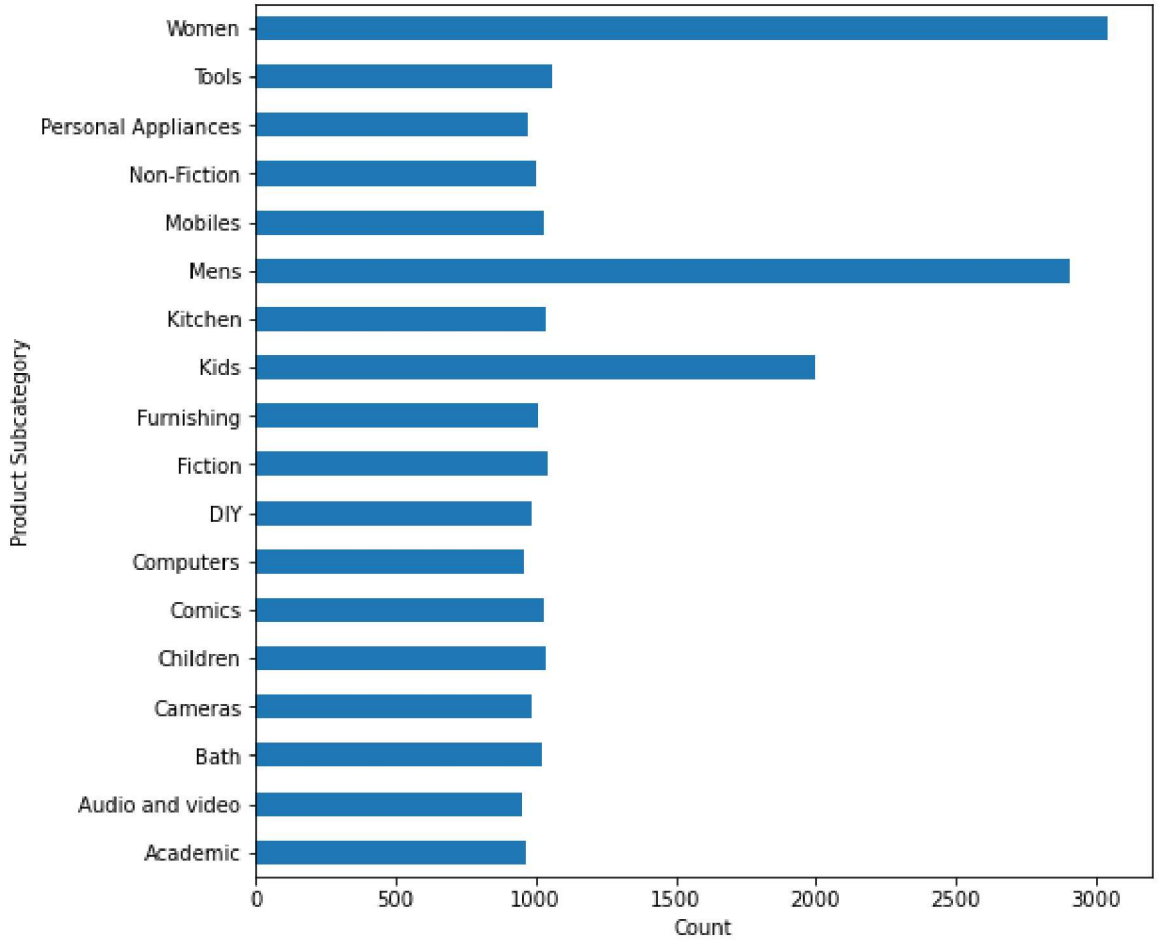


In [44]:
```
plt.figure(figsize=(8,8))
sns.countplot(category_customer['prod_cat'])
plt.xlabel('Product Category')
plt.ylabel('count')
plt.show()
```

```
D:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explici
t keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [43]:
```
plt.figure(figsize=(8,8))
category_customer.groupby('prod_subcat')['prod_subcat'].count().plot(kind='barh')
plt.xlabel('Count')
plt.ylabel('Product Subcategory')
plt.show()
```



4. Calculate the following information using the merged dataset :

a. Time period of the available transaction data

In [45]:
```
customer_final.sort_values(by='tran_date').head()
```

Out[45]:

	transaction_id	cust_id	tran_date	prod_subcat_code	prod_cat_code	Qty	Rate	Tax	total_amt	Store_type	prod_cat	prod_subcat	cu
22899	36332303449	268624	2011-01-02	10	6	-4	-295	123.900	-1303.900	Flagship store	Home and kitchen	Kitchen	
22893	25374972356	268904	2011-01-02	2	6	5	821	431.025	4536.025	MBR	Home and kitchen	Furnishing	
22894	15662366857	272756	2011-01-02	5	3	3	527	166.005	1747.005	e-Shop	Electronics	Computers	
22895	28972634039	275227	2011-01-02	9	3	-1	-334	35.070	-369.070	MBR	Electronics	Cameras	
22896	60041644943	267309	2011-01-02	3	2	1	392	41.160	433.160	Flagship store	Footwear	Women	

In [46]:
```
mi=customer_final['tran_date'].min()
mi
```

Out[46]:
```
Timestamp('2011-01-02 00:00:00')
```

In [47]:
```
ma=customer_final['tran_date'].max()
ma
```

Out[47]:
```
Timestamp('2014-12-02 00:00:00')
```

In [50]:
```
print('Time period of transaction data from: ',mi,'to',ma)
```

```
Time period of transaction data from:  2011-01-02 00:00:00 to 2014-12-02 00:00:00
```

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

In [51]:
```
customer_final.head()
```

Out[51]:

	transaction_id	cust_id	tran_date	prod_subcat_code	prod_cat_code	Qty	Rate	Tax	total_amt	Store_type	prod_cat	prod_subcat	custor
0	80712190438	270351	2014-02-28	1	1	-5	-772	405.300	-4265.300	e-Shop	Clothing	Women	2
1	29258453508	270384	2014-02-27	5	3	-5	-1497	785.925	-8270.925	e-Shop	Electronics	Computers	2
2	51750724947	273420	2014-02-24	6	5	-2	-791	166.110	-1748.110	TeleShop	Books	DIY	2
3	93274880719	271509	2014-02-24	11	6	-3	-1363	429.345	-4518.345	e-Shop	Home and kitchen	Bath	2
4	51750724947	273420	2014-02-23	6	5	-2	-791	166.110	-1748.110	TeleShop	Books	DIY	2

In [52]:
```
n=customer_final.loc[customer_final["total_amt"] < 0,"transaction_id"].count()
print('Count of transactions where the total amount of transaction was negative: ',n)
```

```
Count of transactions where the total amount of transaction was negative:  2164
```

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

In [53]:
```
#groupby the data set on the basis of "Gender" and "prod_cat"
product_gender = customer_final.groupby(["Gender","prod_cat"])[["Qty"]].sum().reset_index()
```

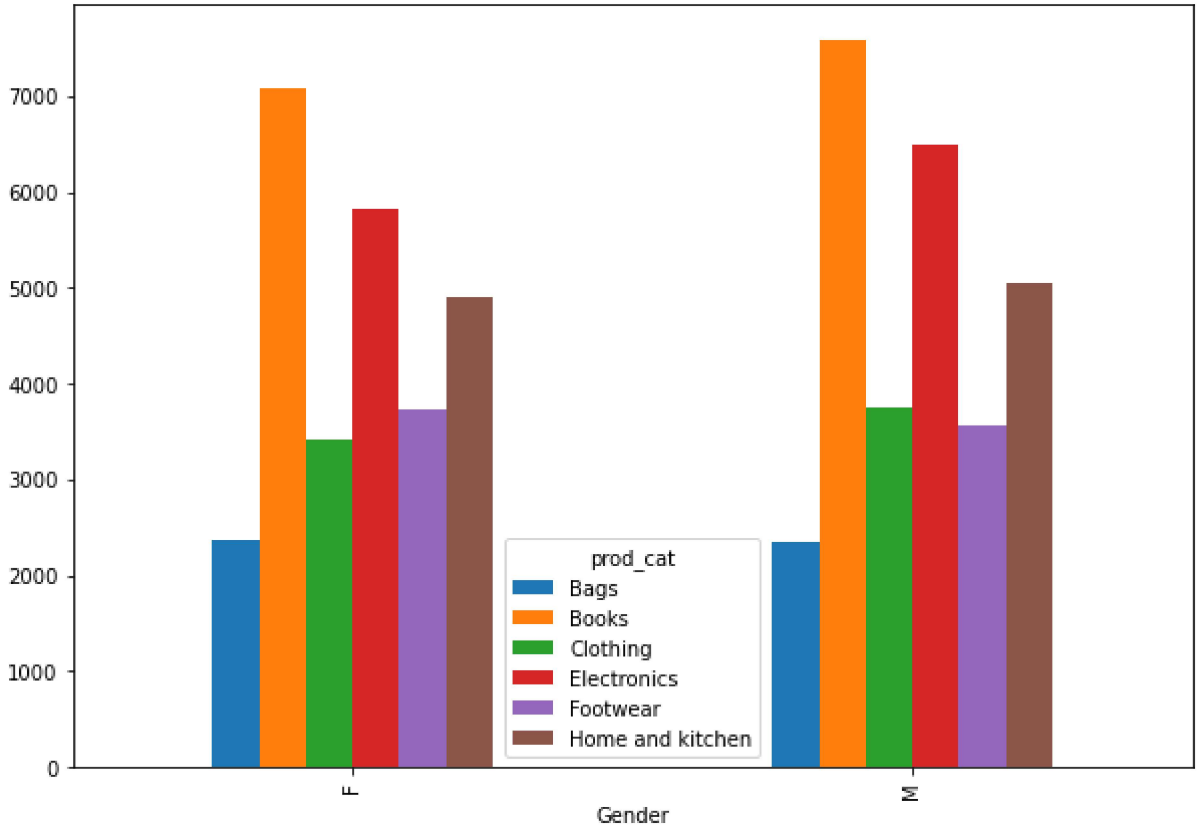
In [54]:
```
product_gender
```

Out[54]:

	Gender	prod_cat	Qty
0	F	Bags	2364
1	F	Books	7080
2	F	Clothing	3425
3	F	Electronics	5832
4	F	Footwear	3721
5	F	Home and kitchen	4898
6	M	Bags	2346
7	M	Books	7587
8	M	Clothing	3748
9	M	Electronics	6486
10	M	Footwear	3561
11	M	Home and kitchen	5051

In [62]:
```
#converting to pivot table for better view
product_gender.pivot(index="Gender",columns="prod_cat",values="Qty").plot(kind='bar',figsize=(10,7))
```

Out[62]: <AxesSubplot:xlabel='Gender'>



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

In [63]:
```
customer_final.head()
```

Out[63]:

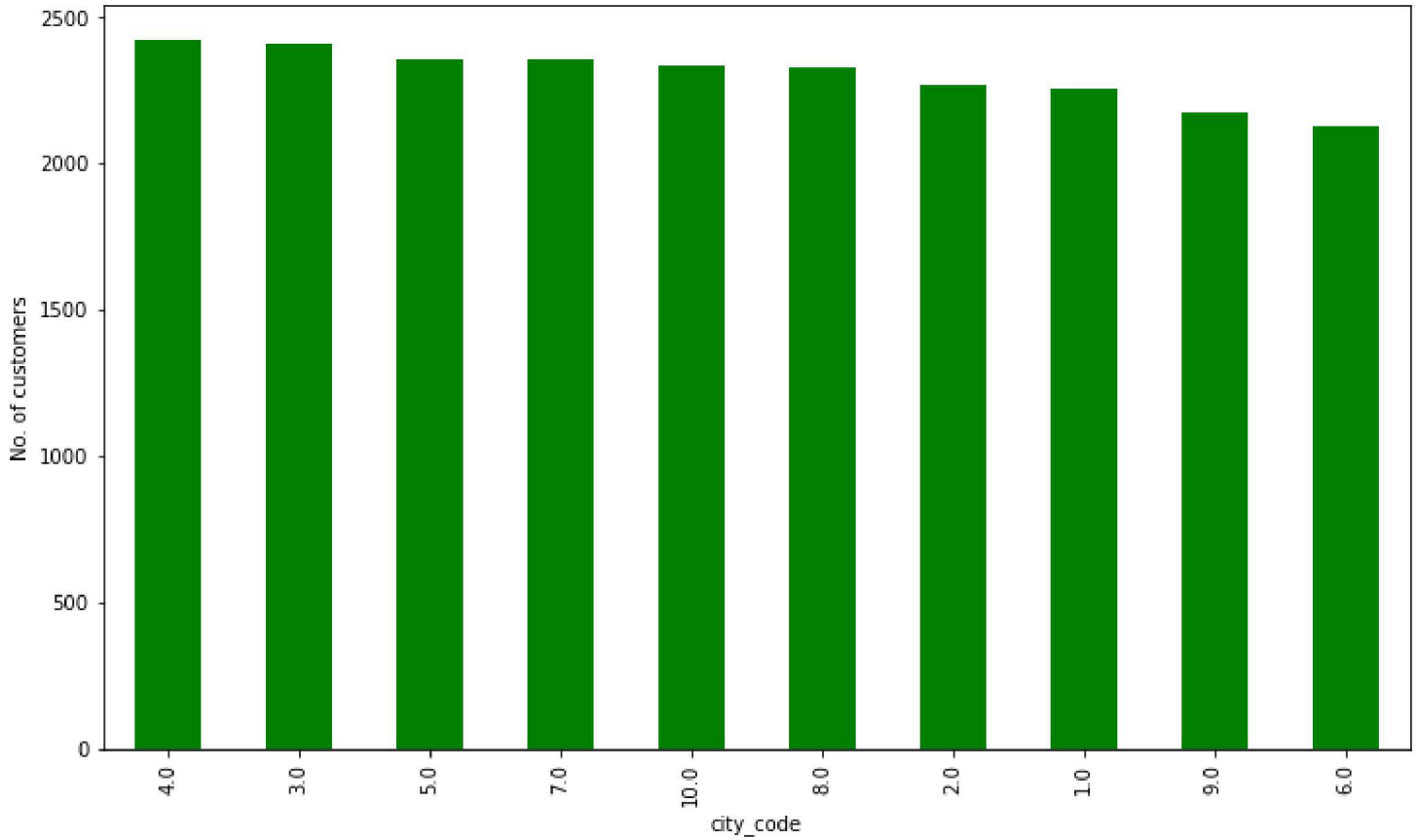
	transaction_id	cust_id	tran_date	prod_subcat_code	prod_cat_code	Qty	Rate	Tax	total_amt	Store_type	prod_cat	prod_subcat	custor
0	80712190438	270351	2014-02-28	1	1	-5	-772	405.300	-4265.300	e-Shop	Clothing	Women	2
1	29258453508	270384	2014-02-27	5	3	-5	-1497	785.925	-8270.925	e-Shop	Electronics	Computers	2
2	51750724947	273420	2014-02-24	6	5	-2	-791	166.110	-1748.110	TeleShop	Books	DIY	2
3	93274880719	271509	2014-02-24	11	6	-3	-1363	429.345	-4518.345	e-Shop	Home and kitchen	Bath	2
4	51750724947	273420	2014-02-23	6	5	-2	-791	166.110	-1748.110	TeleShop	Books	DIY	2

In [64]:
```
per = customer_final.groupby('city_code')['customer_Id'].count().sort_values(ascending =False)
per
```

Out[64]:
```
city_code
4.0     2422
3.0     2410
5.0     2357
7.0     2356
10.0    2333
8.0     2328
2.0     2268
1.0     2255
9.0     2176
6.0     2127
Name: customer_Id, dtype: int64
```

In [65]:
```
per.plot(kind='bar',color='green',figsize=(12,7))
plt.xlabel('city_code')
plt.ylabel('No. of customers')
```

Out[65]:
```
Text(0, 0.5, 'No. of customers')
```



In [73]:
```
percentage = round(per[4.0] / per.sum() * 100,2)
```

In [74]:
```
percentage
```

Out[74]:
```
10.52
```

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

In [76]:
```
customer_final.head()
```

Out[76]:

	transaction_id	cust_id	tran_date	prod_subcat_code	prod_cat_code	Qty	Rate	Tax	total_amt	Store_type	prod_cat	prod_subcat	custor
0	80712190438	270351	2014-02-28	1	1	-5	-772	405.300	-4265.300	e-Shop	Clothing	Women	2
1	29258453508	270384	2014-02-27	5	3	-5	-1497	785.925	-8270.925	e-Shop	Electronics	Computers	2
2	51750724947	273420	2014-02-24	6	5	-2	-791	166.110	-1748.110	TeleShop	Books	DIY	2
3	93274880719	271509	2014-02-24	11	6	-3	-1363	429.345	-4518.345	e-Shop	Home and kitchen	Bath	2
4	51750724947	273420	2014-02-23	6	5	-2	-791	166.110	-1748.110	TeleShop	Books	DIY	2

In [78]:
```
customer_final.groupby('Store_type')['Qty','Rate'].sum().sort_values(by='Qty',ascending=False)
```

```
C:\Users\HP\AppData\Local\Temp/ipykernel_24976/3389044182.py:1: FutureWarning: Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
  customer_final.groupby('Store_type')['Qty','Rate'].sum().sort_values(by='Qty',ascending=False)
```

Out[78]:

Store_type	Qty	Rate
e-Shop	22790	5945770
MBR	11195	2953665
Flagship store	11142	2942874
TeleShop	10993	2836359

In [79]:
```
print('e-Shop store sell the maximum products by value and by quantity')
```

```
e-Shop store sell the maximum products by value and by quantity
```

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

In [80]:
```
s=customer_final.pivot_table(index = 'prod_cat',columns='Store_type', values='total_amt', aggfunc='sum')
```

In [81]:
```
s
```

Out[81]:

prod_cat \ Store_type	Flagship store	MBR	TeleShop	e-Shop
Bags	870548.835	848678.675	789181.055	1617933.265
Books	2493677.810	2496039.195	2545714.470	5297161.155
Clothing	1194423.230	1287686.335	1241834.360	2527193.565
Electronics	2215136.040	2107969.825	1978457.195	4429142.770
Footwear	1234806.560	1112163.715	1235719.290	2643215.250
Home and kitchen	1713004.150	1822403.570	1581227.375	3327977.120

In [54]:
```
s.loc[['Clothing','Electronics'],'Flagship store']
```

Out[54]:
```
prod_cat
Clothing       1194423.23
Electronics    2215136.04
Name: Flagship store, dtype: float64
```

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

In [82]:
```
gender_group = round(customer_final.pivot_table(index = "prod_cat",columns="Gender", values="total_amt", aggfunc='sum'),
```

In [83]:
```
gender_group
```

Out[83]:

Gender	F	M
prod_cat		
Bags	2079618.84	2046722.99
Books	6174590.82	6645972.78
Clothing	3026750.80	3224079.50
Electronics	5019354.21	5711351.62
Footwear	3203155.22	3020200.36
Home and kitchen	4133702.24	4305169.50

In [84]:
```
male_earning = gender_group.loc["Electronics","M"]
```

In [85]:
```
print("The total amount earned from Male customers under the Electronics category is",male_earning)
```

```
The total amount earned from Male customers under the Electronics category is 5711351.62
```

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

In [86]:
```
#creating a new dataframe that does not contain transactions with negative values
positive=customer_final.loc[customer_final["total_amt"]>0]
positive
```

Out[86]:

	transaction_id	cust_id	tran_date	prod_subcat_code	prod_cat_code	Qty	Rate	Tax	total_amt	Store_type	prod_cat	prod_subcat	cu
10	29258453508	270384	2014-02-20	5	3	5	1497	785.925	8270.925	e-Shop	Electronics	Computers	
11	25455265351	267750	2014-02-20	12	6	3	1360	428.400	4508.400	e-Shop	Home and kitchen	Tools	
12	1571002198	275023	2014-02-20	6	5	4	587	246.540	2594.540	e-Shop	Books	DIY	
14	36554696014	269345	2014-02-20	3	5	3	1253	394.695	4153.695	e-Shop	Books	Comics	
15	56814940239	268799	2014-02-20	7	5	5	368	193.200	2033.200	e-Shop	Books	Fiction	
...	...	...	...	...	...	...	...	...	...	...	...	...	
23048	94340757522	274550	2011-01-25	12	5	1	1264	132.720	1396.720	e-Shop	Books	Academic	
23049	89780862956	270022	2011-01-25	4	1	1	677	71.085	748.085	e-Shop	Clothing	Mens	
23050	85115299378	271020	2011-01-25	2	6	4	1052	441.840	4649.840	MBR	Home and kitchen	Furnishing	
23051	72870271171	270911	2011-01-25	11	5	3	1142	359.730	3785.730	TeleShop	Books	Children	
23052	77960931771	271961	2011-01-25	11	5	1	447	46.935	493.935	TeleShop	Books	Children	

20876 rows × 16 columns

In [87]:
```
# creating a dataframe that contains unique transactions
unique_trans = positive.groupby(['customer_Id','prod_cat','prod_subcat'])['transaction_id'].count().reset_index()
```

In [88]:
```
unique_trans
```

Out[88]:

	customer_Id	prod_cat	prod_subcat	transaction_id
0	266783	Books	Non-Fiction	1
1	266783	Clothing	Mens	2
2	266783	Footwear	Mens	1
3	266784	Books	Fiction	1
4	266784	Books	Non-Fiction	1
...	...	...	...	...
19273	275264	Books	Non-Fiction	1
19274	275264	Home and kitchen	Tools	1
19275	275265	Bags	Mens	1
19276	275265	Books	Academic	1
19277	275265	Home and kitchen	Furnishing	1

19278 rows × 4 columns

In [89]:
```
# now finding the customers which have unique transactions greater than 10
unique_trans_count = unique_trans.groupby('customer_Id')['transaction_id'].count().reset_index()
```

In [90]:
```
unique_trans_count.head()
```

Out[90]:

	customer_Id	transaction_id
0	266783	3
1	266784	3
2	266785	5
3	266788	4
4	266794	8

In [91]:
```
unique_trans_count[unique_trans_count['transaction_id'] > 10]
```

Out[91]:

	customer_Id	transaction_id

There are no unique transactions greater than 10

(11) For all customers aged between 25-35, find out:

(a) What was the total amount spent for 'Electronics' and 'Books' product categories?

Adding new column 'age'

In [92]:
```
now = pd.Timestamp('now')
customer_final['DOB'] = pd.to_datetime(customer_final['DOB'], format='%m%d%y')    # 1
customer_final['DOB'] = customer_final['DOB'].where(customer_final['DOB'] < now, customer_final['DOB'] -  np.timedelta64
customer_final['AGE'] = (now - customer_final['DOB']).astype('<m8[Y]')
```

In [93]:
```
customer_final.head()
```

Out[93]:

	transaction_id	cust_id	tran_date	prod_subcat_code	prod_cat_code	Qty	Rate	Tax	total_amt	Store_type	prod_cat	prod_subcat	custor
0	80712190438	270351	2014-02-28	1	1	-5	-772	405.300	-4265.300	e-Shop	Clothing	Women	2
1	29258453508	270384	2014-02-27	5	3	-5	-1497	785.925	-8270.925	e-Shop	Electronics	Computers	2
2	51750724947	273420	2014-02-24	6	5	-2	-791	166.110	-1748.110	TeleShop	Books	DIY	2
3	93274880719	271509	2014-02-24	11	6	-3	-1363	429.345	-4518.345	e-Shop	Home and kitchen	Bath	2
4	51750724947	273420	2014-02-23	6	5	-2	-791	166.110	-1748.110	TeleShop	Books	DIY	2

as we have to deal with customers aged between 25-35, so creating new column 'Age_cat'

In [94]:
```
customer_final['Age_cat'] = pd.cut(customer_final['AGE'],bins=[24,35,46,57],labels=['25-35','36-46','47-57'],include_lowe
```

In [95]:
```
customer_final.head()
```

Out[95]:

	transaction_id	cust_id	tran_date	prod_subcat_code	prod_cat_code	Qty	Rate	Tax	total_amt	Store_type	prod_cat	prod_subcat	custor
0	80712190438	270351	2014-02-28	1	1	-5	-772	405.300	-4265.300	e-Shop	Clothing	Women	2
1	29258453508	270384	2014-02-27	5	3	-5	-1497	785.925	-8270.925	e-Shop	Electronics	Computers	2
2	51750724947	273420	2014-02-24	6	5	-2	-791	166.110	-1748.110	TeleShop	Books	DIY	2
3	93274880719	271509	2014-02-24	11	6	-3	-1363	429.345	-4518.345	e-Shop	Home and kitchen	Bath	2
4	51750724947	273420	2014-02-23	6	5	-2	-791	166.110	-1748.110	TeleShop	Books	DIY	2

In [96]:
```
# grouping the dataframe 'customer_final' on the basis of 'Age_cat' and 'prod_cat'
customer_25_35 = customer_final.groupby(['Age_cat','prod_cat'])['total_amt'].sum()
```

In [97]:
```
customer_25_35
```

Out[97]:
```
Age_cat  prod_cat
25-35    Bags                1119970.540
         Books               3423535.310
         Clothing            1937932.425
         Electronics         3102578.115
         Footwear            1864184.725
         Home and kitchen    2442989.250
36-46    Bags                1947991.240
         Books               6343244.765
         Clothing            2963099.490
         Electronics         5004305.215
         Footwear            2883974.860
         Home and kitchen    4091531.015
47-57    Bags                1058380.050
         Books               3065812.555
         Clothing            1350105.575
         Electronics         2623822.500
         Footwear            1477745.230
         Home and kitchen    1910091.950
Name: total_amt, dtype: float64
```

In [98]:
```
customer_25_35.loc['25-35',['Books','Electronics']]
```

Out[98]:
```
Age_cat  prod_cat
25-35    Books          3423535.310
         Electronics    3102578.115
Name: total_amt, dtype: float64
```

In [99]:
```
print("Total amount spent on 'Electronics' and 'Books' product categories is",
      customer_25_35.loc['25-35',['Books','Electronics']].sum().round(2))
```

```
Total amount spent on 'Electronics' and 'Books' product categories is 6526113.43
```

(b) What was the total amount spent by these customers between 1st Jan 2014 to 1st Mar 2014?

In [100]:
```
customer_final.head()
```

Out[100]:

	transaction_id	cust_id	tran_date	prod_subcat_code	prod_cat_code	Qty	Rate	Tax	total_amt	Store_type	prod_cat	prod_subcat	custor
0	80712190438	270351	2014-02-28	1	1	-5	-772	405.300	-4265.300	e-Shop	Clothing	Women	2
1	29258453508	270384	2014-02-27	5	3	-5	-1497	785.925	-8270.925	e-Shop	Electronics	Computers	2
2	51750724947	273420	2014-02-24	6	5	-2	-791	166.110	-1748.110	TeleShop	Books	DIY	2
3	93274880719	271509	2014-02-24	11	6	-3	-1363	429.345	-4518.345	e-Shop	Home and kitchen	Bath	2
4	51750724947	273420	2014-02-23	6	5	-2	-791	166.110	-1748.110	TeleShop	Books	DIY	2

In [101]:
```
# filtering out data that belongs to the 'age_cat' = 25-35
customer_total_amount_25_35 = customer_final[customer_final['Age_cat']=='25-35']
```

In [102]:
```
customer_total_amount_25_35.head()
```

Out[102]:

	transaction_id	cust_id	tran_date	prod_subcat_code	prod_cat_code	Qty	Rate	Tax	total_amt	Store_type	prod_cat	prod_subcat	custome
2	51750724947	273420	2014-02-24	6	5	-2	-791	166.11	-1748.11	TeleShop	Books	DIY	273
4	51750724947	273420	2014-02-23	6	5	-2	-791	166.11	-1748.11	TeleShop	Books	DIY	273
11	25455265351	267750	2014-02-20	12	6	3	1360	428.40	4508.40	e-Shop	Home and kitchen	Tools	267
23	91116291703	268509	2014-02-20	1	2	4	1243	522.06	5494.06	MBR	Footwear	Mens	268
28	88853694830	268444	2014-02-20	4	4	-3	-80	25.20	-265.20	MBR	Bags	Women	268

In [103]:
```
# getting all the data with transaction date between 1st Jan 2014 to 1st Mar 2014?
total_amount = customer_total_amount_25_35[(customer_total_amount_25_35['tran_date'] >='2014-01-01') & (customer_total_a
```

In [104]:
```
total_amount
```

Out[104]:

	transaction_id	cust_id	tran_date	prod_subcat_code	prod_cat_code	Qty	Rate	Tax	total_amt	Store_type	prod_cat	prod_subcat	custc
2	51750724947	273420	2014-02-24	6	5	-2	-791	166.110	-1748.110	TeleShop	Books	DIY	
4	51750724947	273420	2014-02-23	6	5	-2	-791	166.110	-1748.110	TeleShop	Books	DIY	
11	25455265351	267750	2014-02-20	12	6	3	1360	428.400	4508.400	e-Shop	Home and kitchen	Tools	
23	91116291703	268509	2014-02-20	1	2	4	1243	522.060	5494.060	MBR	Footwear	Mens	
28	88853694830	268444	2014-02-20	4	4	-3	-80	25.200	-265.200	MBR	Bags	Women	
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1048	14460826915	269348	2014-01-01	11	6	3	84	26.460	278.460	MBR	Home and kitchen	Bath	
1051	32889219128	269536	2014-01-01	10	5	5	1423	747.075	7862.075	e-Shop	Books	Non-Fiction	
1054	42711619809	271701	2014-01-01	1	2	5	336	176.400	1856.400	MBR	Footwear	Mens	
1059	67088172893	271877	2014-01-01	1	1	1	902	94.710	996.710	e-Shop	Clothing	Women	
1061	63635040022	268886	2014-01-01	3	2	5	652	342.300	3602.300	e-Shop	Footwear	Women	

202 rows × 18 columns

In [105]:
```
print('The total amount spent by customers aged 25-35 between 1st Jan 2014 to 1st Mar 2014 is',
      total_amount['total_amt'].sum())
```

```
The total amount spent by customers aged 25-35 between 1st Jan 2014 to 1st Mar 2014 is 449938.32
```

Conclusion

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. After performing an EDA had generated a histogram for the both continous and categorical variables. i have analyzed that there are more number of MALE customers then FEMALE in the dataset. E-SHOP has the more percent of market share holding in a store type section. In the product category books are the highest sold product. The time period of the available transaction data is between 02/01/2011 to 02/12/2014. The city code having 4.0 is having the more number of customers. E-SHOP store sells the maximum product by value as well asquantity.

In [ ]: