retail-case-study-python1report

May 29, 2023

1 Customer Analysis for Retail

May 1, 2023

1.0.1 Problem Statement:

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.is

Description: With the retail market getting more and more competitive by the day, there has never been anything more important than the ability for optimizing service business processes when trying to satisfy the expectations of customers. Channelizing and managing data with the aim of working in favor of the customer as well as generating profits is very significant for survival. Ideally, a retailer's customer data reflects the company's success in reaching and nurturing its customers. Retailers built reports summarizing customer behavior using metrics such as conversion rate, average order value, recency of purchase and total amount spent in recent transactions. These measurements provided general insight into the behavioral tendencies of customers. Customer intelligence is the practice of determining and delivering data-driven insights into past and predicted future customer behavior.To be effective, customer intelligence must combine raw transactional and behavioral data to generate derived measures. In a nutshell, for big retail players all over the world, data analytics is applied more these days at all stages of the retail process – taking track of popular products that are emerging, doing forecasts of sales and future demand via predictive simulation, optimizing placements of products and offers through heat-mapping of customers and many others.

1.0.2 Importing Libraries

[2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import datetime as dt
import seaborn as sns
import re



# set the graphs to show in the jupyter notebook
%matplotlib inline
```

```
# set seaborn graphs to a better style
sns.set(style="ticks")
```

1.0.3 The Datasets:-

Customer

Transaction

Product Heirarchy

- Customer: Customers information including demographics
- Transaction: Transactions of customers
- Product Heirarchy: Product information (cateogry, sub category etc…)

1.0.4 Reading and merging datasets

[60]:
```
input_file_path = "C:\\Projects\\EDA\\data\\input_files//"
input_file_name = "Customer.csv"
input_file_path + input_file_name
Customer = pd.read_csv(input_file_path + input_file_name )
```

[61]:
```
input_file_path = "C:\\Projects\\EDA\\data\\input_files//"
input_file_name = "prod_cat_info.csv"
input_file_path + input_file_name
prod_cat_info = pd.read_csv(input_file_path + input_file_name )
```

[62]:
```
input_file_path = "C:\\Projects\\EDA\\data\\input_files//"
input_file_name = "Transactions.csv"
input_file_path + input_file_name
Transactions = pd.read_csv(input_file_path + input_file_name )
```

[63]:
```
prod_cat_info = prod_cat_info.rename(columns={'prod_sub_cat_code':␣
  ↪'prod_subcat_code'})
```

[64]:
```
Trans_and_prod = pd.merge (left = Transactions, right = prod_cat_info, on =␣
  ↪['prod_cat_code','prod_subcat_code'], how ='left')
```

[65]:
```
Customer = Customer.rename(columns={'customer_Id': 'cust_id'})
```

[66]:
```
Customer_Final = pd.merge (left = Trans_and_prod, right = Customer, on =␣
  ↪'cust_id', how ='left')
```

1.0.5 Data Exploration

[67]:
```
Customer_Final.dtypes
```

```
[67]: 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
      DOB                  object
      Gender               object
      city_code           float64
      dtype: object
```

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

```
[68]:    transaction_id  cust_id   tran_date  prod_subcat_code  prod_cat_code  Qty  \
      0     80712190438   270351  28-02-2014                 1              1   -5
      1     29258453508   270384  27-02-2014                 5              3   -5
      2     51750724947   273420  24-02-2014                 6              5   -2
      3     93274880719   271509  24-02-2014                11              6   -3
      4     51750724947   273420  23-02-2014                 6              5   -2
      5     97439039119   272357  23-02-2014                 8              3   -2
      6     45649838090   273667  22-02-2014                11              6   -1
      7     22643667930   271489  22-02-2014                12              6   -1
      8     79792372943   275108  22-02-2014                 3              1   -3
      9     50076728598   269014  21-02-2014                 8              3   -4

         Rate      Tax  total_amt Store_type          prod_cat          prod_subcat  \
      0  -772  405.300  -4265.300     e-Shop          Clothing                Women
      1 -1497  785.925  -8270.925     e-Shop       Electronics            Computers
      2  -791  166.110  -1748.110   TeleShop             Books                  DIY
      3 -1363  429.345  -4518.345     e-Shop  Home and kitchen                 Bath
      4  -791  166.110  -1748.110   TeleShop             Books                  DIY
      5  -824  173.040  -1821.040   TeleShop       Electronics  Personal Appliances
      6 -1450  152.250  -1602.250     e-Shop  Home and kitchen                 Bath
      7 -1225  128.625  -1353.625   TeleShop  Home and kitchen                Tools
      8  -908  286.020  -3010.020        MBR          Clothing                 Kids
      9  -581  244.020  -2568.020     e-Shop       Electronics  Personal Appliances

                DOB Gender  city_code
      0  26-09-1981      M        5.0
      1  11-05-1973      F        8.0
      2  27-07-1992      M        8.0
```

```
3  08-06-1981      M        3.0
4  27-07-1992      M        8.0
5  09-10-1982      F        6.0
6  29-05-1981      M        9.0
7  21-04-1971      M        9.0
8  04-11-1971      F        8.0
9  27-11-1979      F        3.0
```

[69]:

```
Customer_Final['transaction_id'] = Customer_Final['transaction_id'].
  ↪astype('object')
Customer_Final['cust_id'] = Customer_Final['cust_id'].astype('object')
Customer_Final['prod_subcat_code'] = Customer_Final['prod_subcat_code'].
  ↪astype('object')
Customer_Final['prod_cat_code'] = Customer_Final['prod_cat_code'].
  ↪astype('object')
Customer_Final['city_code'] = Customer_Final['city_code'].astype('object')
```

[70]:

```
Customer_Final.describe(include = np.number)
```

[70]:

```
                Qty           Rate            Tax     total_amt
count  23053.000000   23053.000000   23053.000000  23053.000000
mean       2.432395     636.369713     248.667192   2107.308002
std        2.268406     622.363498     187.177773   2507.561264
min       -5.000000   -1499.000000       7.350000  -8270.925000
25%        1.000000     312.000000      98.280000    762.450000
50%        3.000000     710.000000     199.080000   1754.740000
75%        4.000000    1109.000000     365.715000   3569.150000
max        5.000000    1500.000000     787.500000   8287.500000
```

[71]:

```
cat_vars = Customer_Final.select_dtypes(include=['object']).columns
for col in cat_vars:
    freq_table = Customer_Final[col].value_counts()
    print('\nFrequency Table for', col)
    print(freq_table)
```

```

Frequency Table for transaction_id
4170892941     4
32263938079    4
426787191      4
91377906980    3
44125492691    3
              ..
88791150012    1
17648795819    1
25673128667    1
14616200775    1
```

```
77960931771    1
Name: transaction_id, Length: 20878, dtype: int64

Frequency Table for cust_id
269449    13
268819    13
272286    12
270831    12
272415    12
          ..
270876     1
272472     1
273867     1
274139     1
273723     1
Name: cust_id, Length: 5506, dtype: int64

Frequency Table for tran_date
13-07-2011    35
21-12-2013    33
23-10-2011    33
22-11-2011    33
25-09-2011    33
              ..
23-02-2014     2
24-02-2014     2
27-02-2014     1
21-02-2014     1
28-02-2014     1
Name: tran_date, Length: 1129, dtype: int64

Frequency Table for prod_subcat_code
4     4002
3     3067
10    2993
1     2950
11    2058
12    2029
7     1043
2     1007
6      989
9      985
8      972
5      958
Name: prod_subcat_code, dtype: int64

Frequency Table for prod_cat_code
5    6069
```

```
3    4898
6    4129
2    2999
1    2960
4    1998
Name: prod_cat_code, dtype: int64

Frequency Table for Store_type
e-Shop            9311
MBR               4661
Flagship store    4577
TeleShop          4504
Name: Store_type, dtype: int64

Frequency Table for prod_cat
Books               6069
Electronics         4898
Home and kitchen    4129
Footwear            2999
Clothing            2960
Bags                1998
Name: prod_cat, dtype: int64

Frequency Table for prod_subcat
Women                  3048
Mens                   2912
Kids                   1997
Tools                  1062
Fiction                1043
Kitchen                1037
Children               1035
Mobiles                1031
Comics                 1031
Bath                   1023
Furnishing             1007
Non-Fiction            1004
DIY                     989
Cameras                 985
Personal Appliances     972
Academic                967
Computers               958
Audio and video         952
Name: prod_subcat, dtype: int64

Frequency Table for DOB
27-12-1988    32
17-09-1982    32
25-02-1974    27
```

```
20-03-1972    25
18-11-1991    24
              ..
29-01-1976     1
01-05-1980     1
23-06-1988     1
25-06-1985     1
10-06-1972     1
Name: DOB, Length: 3987, dtype: int64

Frequency Table for Gender
M    11811
F    11233
Name: Gender, dtype: int64

Frequency Table for city_code
4.0     2422
3.0     2411
5.0     2360
7.0     2356
10.0    2333
8.0     2330
2.0     2270
1.0     2258
9.0     2178
6.0     2127
Name: city_code, dtype: int64
```

1.0.6 Handling Null values

[72]:

```
Customer_Final.isnull().sum()
```

[72]:

```
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
DOB                 0
Gender              9
city_code           8
```

```
dtype: int64
```

[73]:

```
for col in Customer_Final.columns:
    if Customer_Final[col].dtype == 'float64':
        Customer_Final[col].fillna(Customer_Final[col].mean(), inplace=True)
    elif Customer_Final[col].dtype == 'object':
        Customer_Final[col].fillna(Customer_Final[col].mode()[0], inplace=True)
```

[74]:

```
Customer_Final.isnull().sum()
```

[74]:

```
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
DOB                 0
Gender              0
city_code           0
dtype: int64
```

[76]:

```
Customer_Final['transaction_id'] = Customer_Final['transaction_id'].
 ↪astype('object')
Customer_Final['cust_id'] = Customer_Final['cust_id'].astype('object')
Customer_Final['prod_subcat_code'] = Customer_Final['prod_subcat_code'].
 ↪astype('object')
Customer_Final['prod_cat_code'] = Customer_Final['prod_cat_code'].
 ↪astype('object')
Customer_Final['city_code'] = Customer_Final['city_code'].astype('object')
```

[77]:

```
Customer_Final.dtypes
```

[77]:

```
transaction_id       object
cust_id              object
tran_date            object
prod_subcat_code     object
prod_cat_code        object
Qty                   int64
Rate                  int64
Tax                 float64
total_amt           float64
```

```
Store_type          object
prod_cat            object
prod_subcat         object
DOB                 object
Gender              object
city_code           object
dtype: object
```

1.0.7 Data Analysis

Time period of the available transaction data

[29]:

```
#changing the tran_date column to datetime data type
Customer_Final['tran_date'] = pd.to_datetime(Customer_Final['tran_date'],␣
  ↪format = '%d-%m-%Y')
#finding the earliest and the latest date using max and min func
earliest_date = Customer_Final['tran_date'].min()
latest_date = Customer_Final['tran_date'].max()
#getting the difference to get the time period
time_period = latest_date - earliest_date
print(f'The available transaction data spans {time_period.days} days, from␣
  ↪{earliest_date} to {latest_date}.')
```

```
The available transaction data spans 1430 days, from 2011-01-02 00:00:00 to
2014-12-02 00:00:00.
```

Return transaction — Count of transactions where the total amount of transaction was negative

[33]:

```
negative_transactions = Customer_Final[Customer_Final['total_amt'] <0].shape[0]

print(f'The number of transactions where the total amount was negative is␣
  ↪{negative_transactions}.')
```

```
The number of transactions where the total amount was negative is 2177.
```

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

[38]:

```
grouped = Customer_Final.groupby(['Gender', 'prod_cat']).agg({'total_amt':␣
  ↪'sum'})
```

[41]:

```
totals =grouped.groupby(level=0).transform('sum')
grouped['Percent'] = 100 * grouped['total_amt'] / totals['total_amt']
print(grouped)
```

```
                           total_amt    Percent
Gender prod_cat
F      Bags              2077985.650   8.796260
       Books             6164692.235  26.095578
```

```
        Clothing          3026750.805  12.812450
        Electronics       5019354.210  21.247281
        Footwear          3202552.990  13.556633
        Home and kitchen  4132177.335  17.491799
M       Bags              2046722.990   8.208025
        Books             6645972.775  26.652514
        Clothing          3224079.495  12.929608
        Electronics       5703109.425  22.871325
        Footwear          3014672.050  12.089816
        Home and kitchen  4301075.480  17.248712
```

Analyzing Which City code has the maximum customers and the percentage of customers from that city

[47]:

```
#grouping the data
Grouped = Customer_Final.groupby('city_code').agg({'cust_id': 'nunique'})
#Getting the percentage
total_customers = Grouped['cust_id'].sum()
Grouped['Percent'] = 100 * Grouped['cust_id'] / total_customers
#Getting the max city code
max_customers = Grouped['cust_id'].idxmax()
max_percent = Grouped.loc[max_customers, 'Percent']
print(f"The city code with the maximum customers is {max_customers} with a␣
 ↪percentage of {max_percent:.2f}%")
```

```
The city code with the maximum customers is 3.0 with a percentage of 10.47%
```

Analyzing Which store type sells the maximum products by value and by quantity

[74]:

```
#Grouping the data
Quantity_grouped = Customer_Final.groupby(['prod_cat'])[['Qty']].sum()
Value_grouped = Customer_Final.groupby(['prod_cat'])[['total_amt']].sum()
#getting the max
Max_Qty = Quantity_grouped.idxmax()['Qty']
Max_Value = Value_grouped.idxmax()['total_amt']
print(f"The maximum product sold by quantity is {Max_Qty} and by value is␣
 ↪{Max_Value}")
```

```
The maximum product sold by quantity is Books and by value is Books
```

Analzying the total amount earned from the "Electronics" and "Clothing" categories from Flagship Stores?

[94]:

```
#filtering the data
flagship_elec_cloth = Customer_Final[(Customer_Final['Store_type'] == 'Flagship␣
 ↪store') & (Customer_Final['prod_cat'].isin(['Electronics', 'Clothing']))]
#grouping the data
Amount_grouped = flagship_elec_cloth.groupby(['prod_cat'])[['total_amt']].sum()
```

```
print(f" the total amount earned from the Clothing and Electronics categories␣
 ↪from Flagship Stores resp are {Amount_grouped.to_string(index = False,␣
 ↪header = False)}")
```

```
 the total amount earned from the Clothing and Electronics categories from
Flagship Stores resp are 1194423.23
2215136.04
```

Analyzing total amount earned from "Male" customers under the "Electronics" category?

[84]:
```
#filering the data
male_Electronics = Customer_Final[(Customer_Final['Gender'] == 'M') &␣
 ↪(Customer_Final['prod_cat'] == 'Electronics')]
#grouping the data
Amount_grouped1 = male_Electronics.groupby(['prod_cat'])[['total_amt']].sum()
print(f" the total amount earned from Male customers under the Electronics␣
 ↪category is {Amount_grouped1.to_string(header=False, index=False)}")
```

```
 the total amount earned from Male customers under the Electronics category is
5703109.425
```

Analyzing customers who have more than 10 unique transactions, after removing all transactions that have any negative amounts

[102]:
```
#Filtering the data
Positive_transactions = Customer_Final[(Customer_Final['total_amt'] > 0)]
#Counting the data by grouping
Count_Ptransaction = Positive_transactions.groupby(['cust_id'])[['total_amt']].
 ↪count()
#Filtering the data again
count10_transactions =Count_Ptransaction[(Count_Ptransaction['total_amt'] > 10)]
#counting the cust
no_of_customers = count10_transactions.count()['total_amt']
print(f" The count of customers who have more than 10 unique transaction with␣
 ↪us is {no_of_customers}")
```

```
 The count of customers who have more than 10 unique transaction with us is 6
```

Analyzing for all customers aged between 25 - 35, the total amount spent for "Electronics" and "Books" product categories

[127]:
```
#a
#Changing the data type of the DOB column to datetime data type
Customer_Final['DOB'] = pd.to_datetime(Customer_Final['DOB'], format =␣
 ↪'%d-%m-%Y')
#Getting todays date using now
now = pd.to_datetime('now')
#Getting the differnce
```

```
Customer_Final['Age'] = (now - Customer_Final['DOB'])/ pd.Timedelta(days=365.
 ↪2425)
#Filtering the data
cust_25to35 = Customer_Final[(Customer_Final['Age'] >= 25) &␣
 ↪(Customer_Final['Age'] <= 35)]
cust_25to35 = Customer_Final[(Customer_Final['prod_cat'].isin(['Electronics',␣
 ↪'Books']))]
#Getting the total amount
total_spent = cust_25to35['total_amt'].sum()
print (f'the total amount spent by the 25-35 years customers for "Electronics"␣
 ↪and "Books" product categories  {total_spent}')
```

```
the total amount spent by the 25-35 years customers for "Electronics" and
"Books" product categories  23545157.675
```

```
C:\Users\dell\anaconda3\lib\site-packages\pandas\core\arrays\datetimes.py:2224:
FutureWarning: The parsing of 'now' in pd.to_datetime without `utc=True` is
deprecated. In a future version, this will match Timestamp('now') and
Timestamp.now()
  result, tz_parsed = tslib.array_to_datetime(
```

Analyzing for all customers aged between 25 - 35, the total amount spent by these customers between 1st Jan, 2014 to 1st Mar, 2014

[134]:

```
#b
#Filtering the data
cust1_25to35 = Customer_Final[(Customer_Final['Age'] >= 25) &␣
 ↪(Customer_Final['Age'] <= 35)]
date_filter = Customer_Final[(Customer_Final['tran_date'] >= '2014-01-01') &␣
 ↪(Customer_Final['tran_date'] < '2014-03-01')]
#Getting the total amount
total_amountspent  = date_filter['total_amt'].sum()
print(f'the total amount spent by these customers between 1st Jan, 2014 to 1st␣
 ↪Mar, 2014 is {total_amountspent}')
```

```
the total amount spent by these customers between 1st Jan, 2014 to 1st Mar, 2014
is 1366271.725
```

1.0.8 Data Visualization

[78]:

```
Continuous_customer = Customer_Final.select_dtypes (exclude = object)
Categorical_customer = Customer_Final.select_dtypes (include = object)
```

[79]:

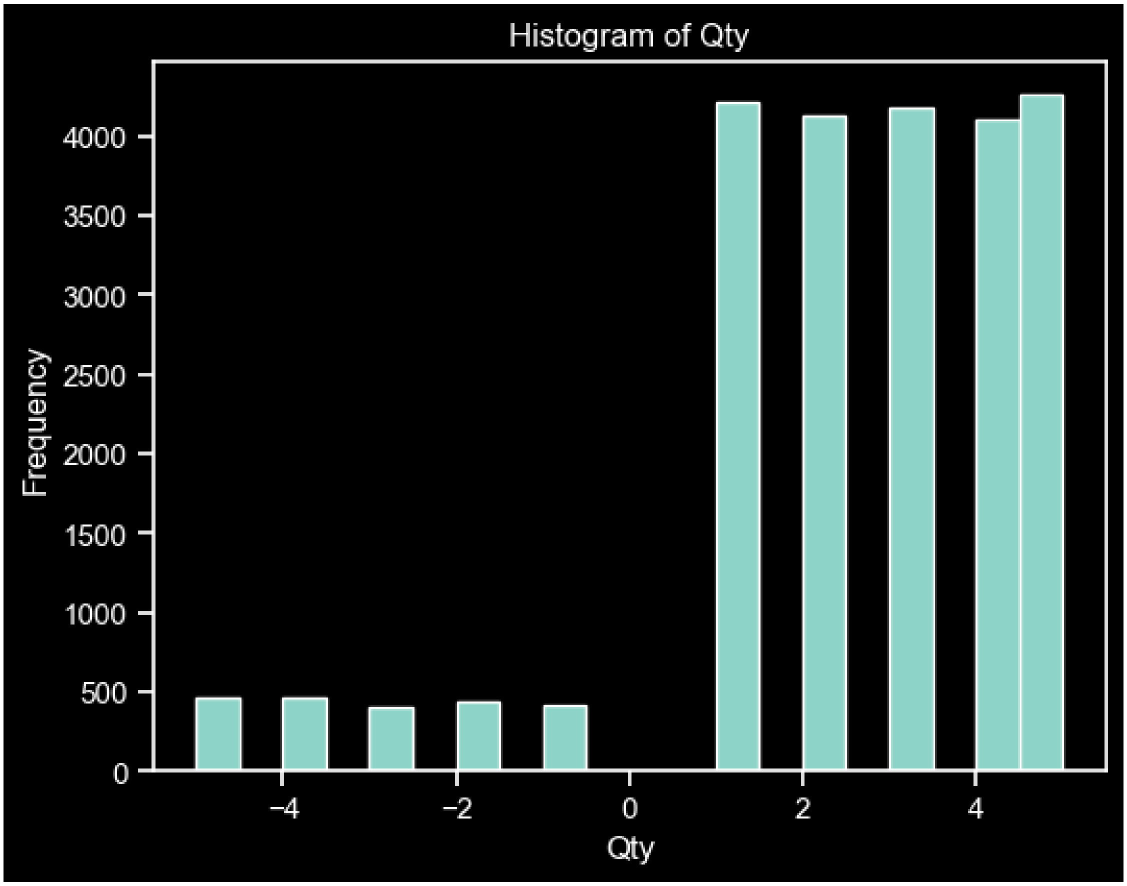
```
Customer_Final.select_dtypes (include = object)
```

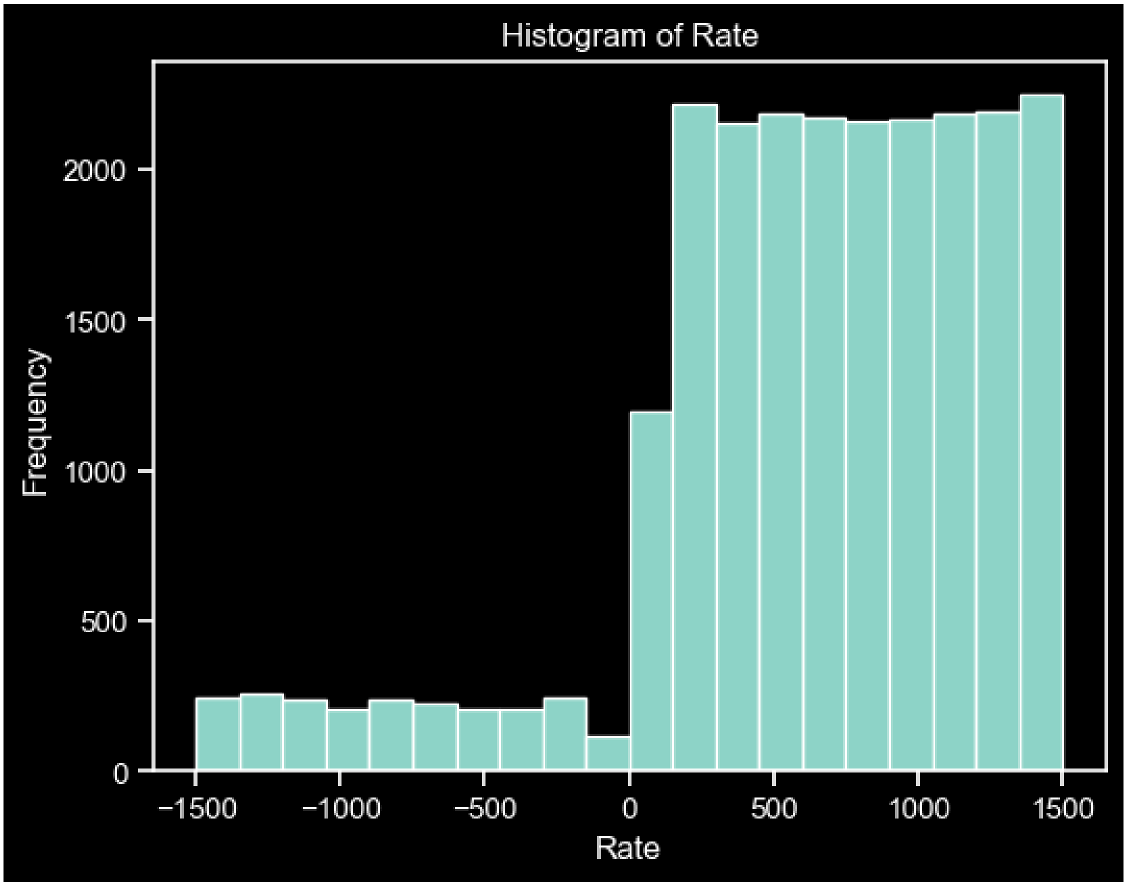
[79]:

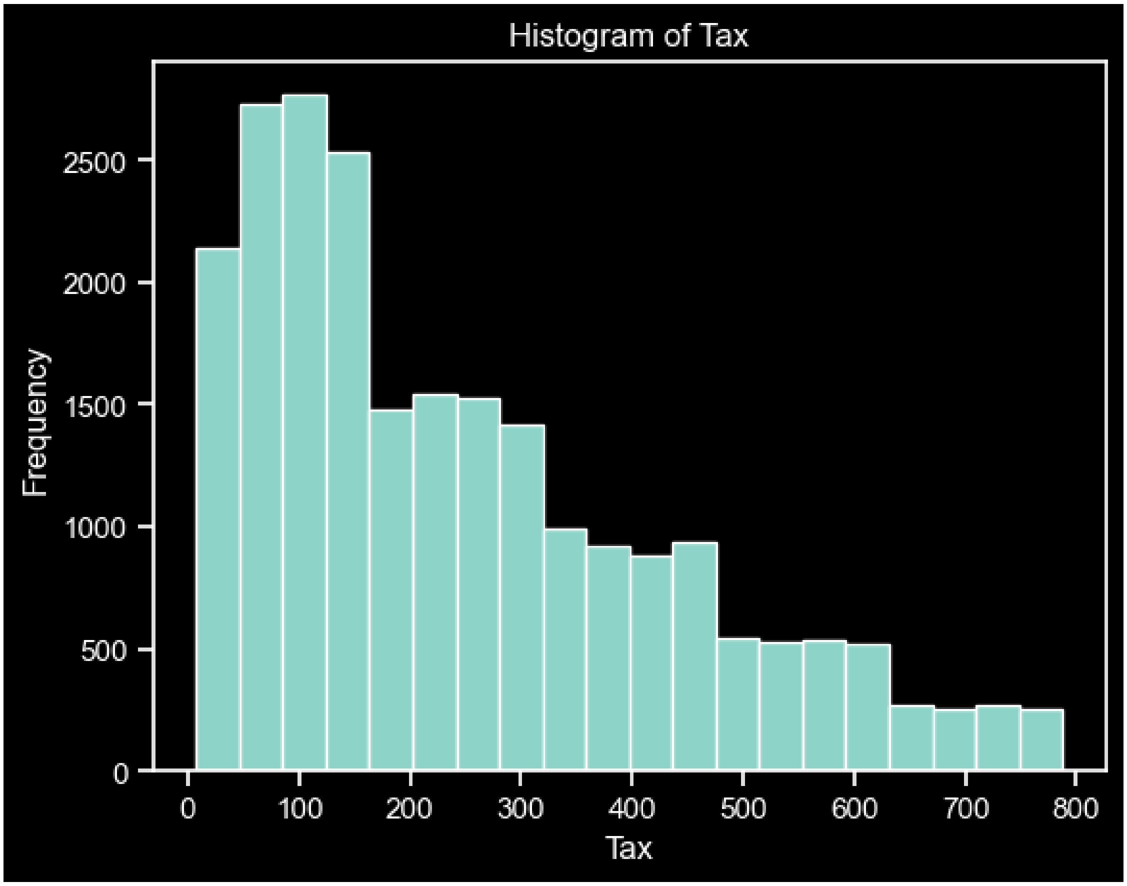
```
      transaction_id cust_id   tran_date prod_subcat_code prod_cat_code  \
0        80712190438  270351  28-02-2014                1             1
1        29258453508  270384  27-02-2014                5             3
```

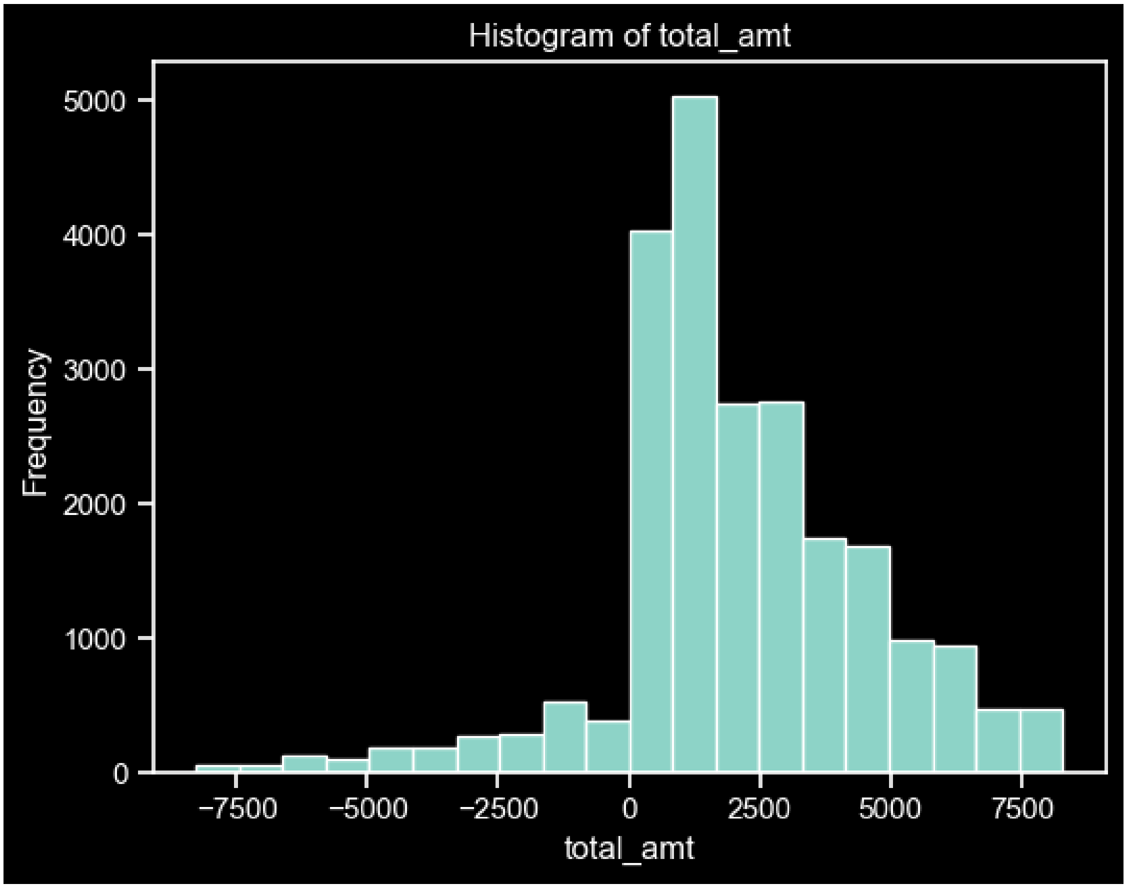
```
2        51750724947  273420  24-02-2014                6             5
3        93274880719  271509  24-02-2014               11             6
4        51750724947  273420  23-02-2014                6             5
…              …     …         …              …           …
23048    94340757522  274550  25-01-2011               12             5
23049    89780862956  270022  25-01-2011                4             1
23050    85115299378  271020  25-01-2011                2             6
23051    72870271171  270911  25-01-2011               11             5
23052    77960931771  271961  25-01-2011               11             5

      Store_type          prod_cat prod_subcat         DOB Gender city_code
0         e-Shop          Clothing       Women  26-09-1981      M       5.0
1         e-Shop       Electronics   Computers  11-05-1973      F       8.0
2       TeleShop             Books         DIY  27-07-1992      M       8.0
3         e-Shop  Home and kitchen        Bath  08-06-1981      M       3.0
4       TeleShop             Books         DIY  27-07-1992      M       8.0
…          …               …         …         …    …       …
23048     e-Shop             Books    Academic  21-02-1972      M       7.0
23049     e-Shop          Clothing        Mens  27-04-1984      M       9.0
23050        MBR  Home and kitchen  Furnishing  20-06-1976      M       8.0
23051   TeleShop             Books    Children  22-05-1970      M       2.0
23052   TeleShop             Books    Children  15-01-1982      M       1.0

[23053 rows x 11 columns]
```

[86]:
```
#putting continuous columns in a conttainer
continuous_vars = ['Qty', 'Rate', 'Tax', 'total_amt']
#using for loop
for var in continuous_vars:
    plt.style.use('dark_background')
    plt.rcParams.update({'text.color': 'white'})
    plt.hist(Continuous_customer [var], bins=20)
    plt.xlabel(var)
    plt.ylabel('Frequency')
    plt.title(f'Histogram of {var}')
    plt.show()
```


Histogram of Qty
Frequency
0
500
1000
1500
2000
2500
3000
3500
4000
−4
−2
0
2
4
Qty


Histogram of Rate
Frequency
0
500
1000
1500
2000
−1500
−1000
−500
0
500
1000
1500
Rate


Histogram of Tax
Frequency
0
500
1000
1500
2000
2500
Tax
0
100
200
300
400
500
600
700
800



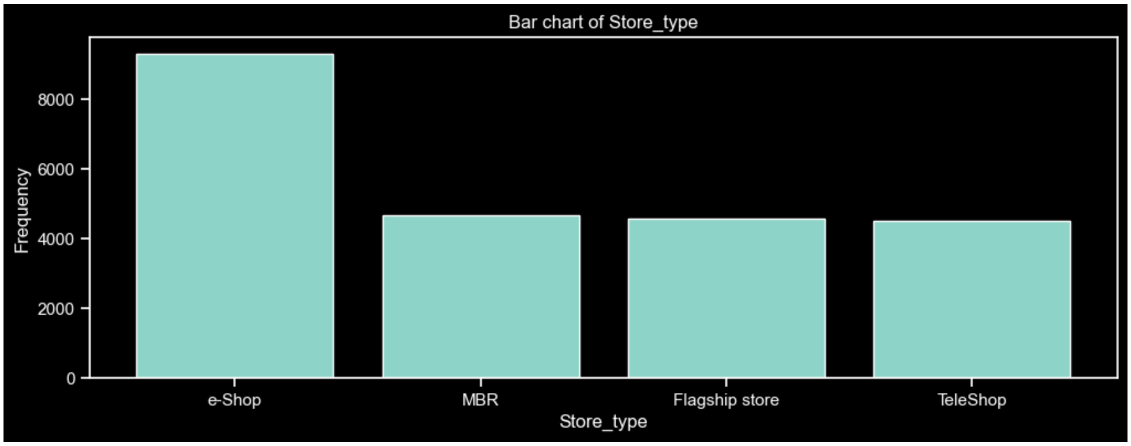
The impression I get from the

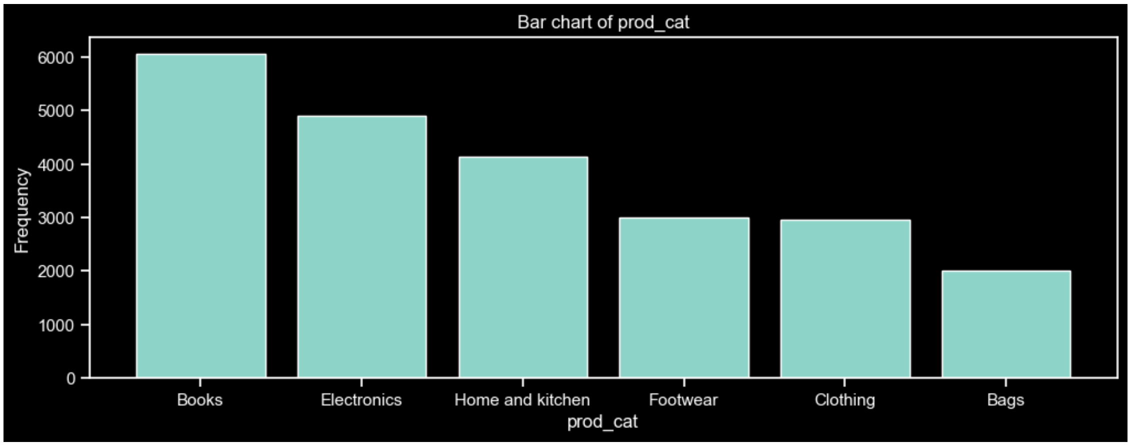
Histogram of Qty and Rate(Frequency wise) shows the maximum quantity purchased is five and also most of the people have bought five items followed by one and three items. Some people have also retured products.

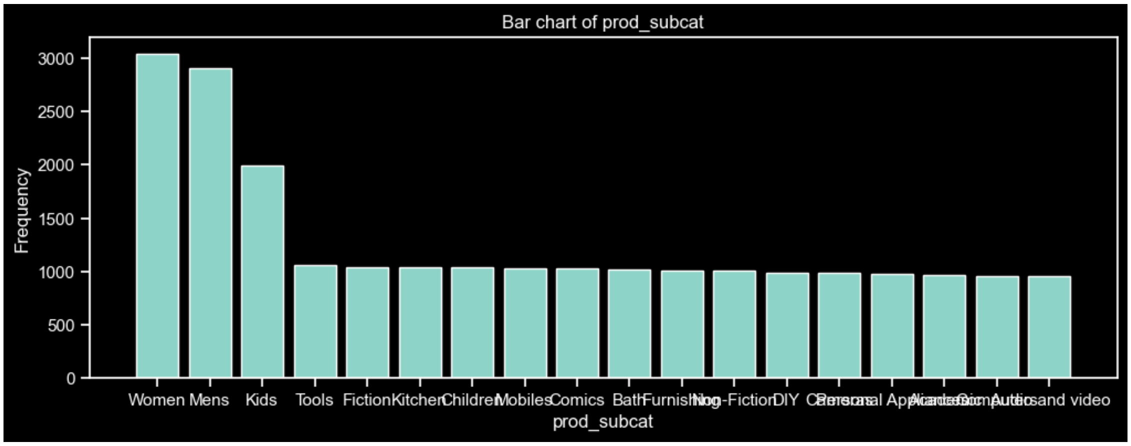
Histogram of Tax and total amount (Frequency wise) shows the items with low tax are puerchased often than with the high tax rate.

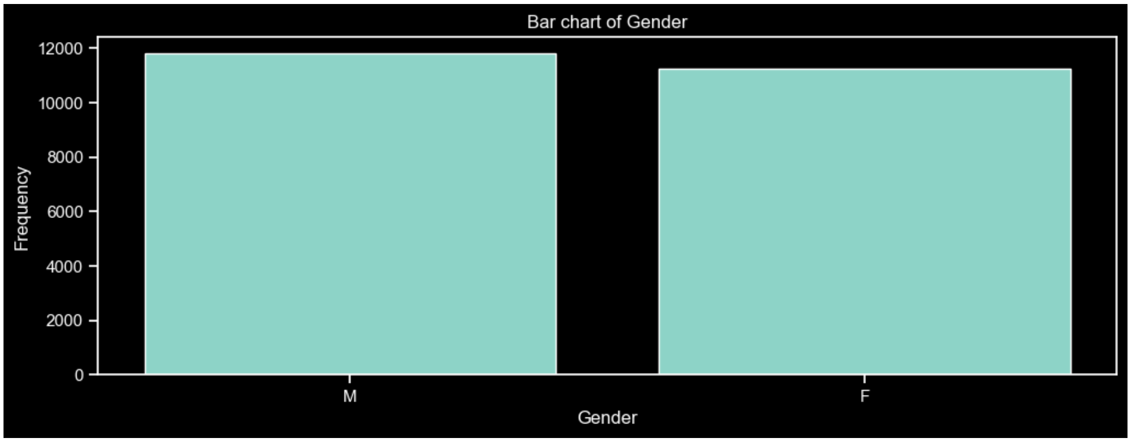
[87]:

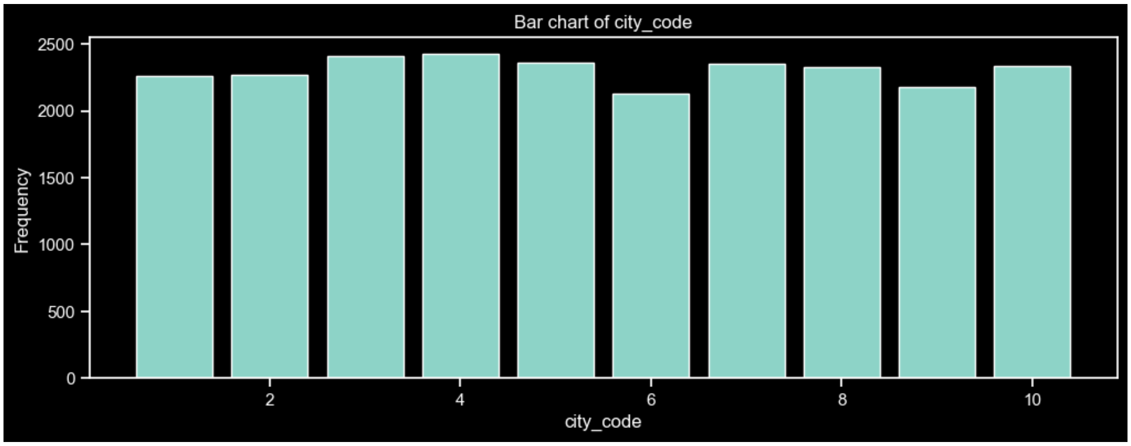
```
#putting categoricals columns in a container
categorical_vars = [ 'Store_type', 'prod_cat', 'prod_subcat','Gender',␣
 ↪'city_code']
#using for loop
for var in categorical_vars:
    plt.style.use('dark_background')
    plt.rcParams.update({'text.color': 'white'})
    plt.figure(figsize=(12,4 ))
    counts = Categorical_customer[var].value_counts()
    plt.bar(counts.index, counts.values)
    plt.xlabel(var)
    plt.ylabel('Frequency')
    plt.title(f'Bar chart of {var}')
    plt.show()
```











The impression I get from the

Frequency histogram of the store type shows the most of the sale happened by E-shop and the reamaining contribution done by the MBR, Flagship Store and Teleshop are equally same.

Frequency histogram of the Product category shows the most purchased category is book followed by Electronics, Home and Kitchen, Footwear, Clothing and Bags- Least bought category

Frequency histogram of the gender shows males have purchased slightly higher than females

Frequency histogram of the city code shows the purchased made by the all the cities are more or less same.