In [1]:
```
import pandas
```

In [2]:
```
import pandas as pd
```

In [3]:
```
df = pd.read_csv("blackfriday.csv")
```

In [4]:
```
type(df)
```

Out[4]:
```
pandas.core.frame.DataFrame
```

In [5]:
```
df.shape
```

Out[5]:
```
(550068, 12)
```

In [6]:
```
df.ndim
```

Out[6]:
```
2
```

In [7]:
```
df.head(10) # top 10 rows
```

Out[7]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Mari
0	1000001	P00069042	F	0-17	10	A	2	
1	1000001	P00248942	F	0-17	10	A	2	
2	1000001	P00087842	F	0-17	10	A	2	
3	1000001	P00085442	F	0-17	10	A	2	
4	1000002	P00285442	M	55+	16	C	4+	
5	1000003	P00193542	M	26-35	15	A	3	
6	1000004	P00184942	M	46-50	7	B	2	
7	1000004	P00346142	M	46-50	7	B	2	
8	1000004	P0097242	M	46-50	7	B	2	
9	1000005	P00274942	M	26-35	20	A	1	

In [8]:
```
df.tail(10) # bottom 10 rows
```

Out[8]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
550058	1006024	P00372445	M	26-35	12	A	0
550059	1006025	P00370853	F	26-35	1	B	1
550060	1006026	P00371644	M	36-45	6	C	1
550061	1006029	P00372445	F	26-35	1	C	1
550062	1006032	P00372445	M	46-50	7	A	3
550063	1006033	P00372445	M	51-55	13	B	1
550064	1006035	P00375436	F	26-35	1	C	3
550065	1006036	P00375436	F	26-35	15	B	4+
550066	1006038	P00375436	F	55+	1	C	2
550067	1006039	P00371644	F	46-50	0	B	4+

In [9]:
```
df.duplicated() # check for duplicate values
```

Out[9]:
```
0         False
1         False
2         False
3         False
4         False
          ...
550063    False
550064    False
550065    False
550066    False
550067    False
Length: 550068, dtype: bool
```

In [10]:
```
sum(df.duplicated()) #count of duplicated values
```

Out[10]:
```
0
```

In [11]:
```
[i for i in df.columns if 'Product' in i] # product related columns
```

Out[11]:
```
['Product_ID',
 'Product_Category_1',
 'Product_Category_2',
 'Product_Category_3']
```

In [12]:
```
df_product = df[['Product_ID', 'Product_Category_1','Product_Category_2', 'Product_
```

In [13]:
```
df_product
```

Out[13]:

	Product_ID	Product_Category_1	Product_Category_2	Product_Category_3
0	P00069042	3	NaN	NaN
1	P00248942	1	6.0	14.0
2	P00087842	12	NaN	NaN
3	P00085442	12	14.0	NaN
4	P00285442	8	NaN	NaN
...	...	...	...	...
550063	P00372445	20	NaN	NaN
550064	P00375436	20	NaN	NaN
550065	P00375436	20	NaN	NaN
550066	P00375436	20	NaN	NaN
550067	P00371644	20	NaN	NaN

550068 rows × 4 columns

In [14]:
```
df.axes # axes returns row axis label as well as columns
```

Out[14]:
```
[RangeIndex(start=0, stop=550068, step=1),
 Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
        'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category_1',
        'Product_Category_2', 'Product_Category_3', 'Purchase'],
       dtype='object')]
```

In [15]:
```
df.Gender.values # values return the series of values as ndarray
```

Out[15]:
```
array(['F', 'F', 'F', ..., 'F', 'F', 'F'], dtype=object)
```

In [16]:
```
df.Gender.size
```

Out[16]:
```
550068
```

In [17]:
```
df.dtypes # type of columns
```

Out[17]:
```
User_ID                       int64
Product_ID                   object
Gender                       object
Age                          object
Occupation                    int64
City_Category                object
Stay_In_Current_City_Years   object
Marital_Status                int64
Product_Category_1            int64
Product_Category_2          float64
Product_Category_3          float64
Purchase                      int64
dtype: object
```

In [18]:
```
df.describe() #statistical values for numerical columns
```

Out[18]:

	User_ID	Occupation	Marital_Status	Product_Category_1	Product_Category_2	Prod
count	5.500680e+05	550068.000000	550068.000000	550068.000000	376430.000000	
mean	1.003029e+06	8.076707	0.409653	5.404270	9.842329	
std	1.727592e+03	6.522660	0.491770	3.936211	5.086590	
min	1.000001e+06	0.000000	0.000000	1.000000	2.000000	
25%	1.001516e+06	2.000000	0.000000	1.000000	5.000000	
50%	1.003077e+06	7.000000	0.000000	5.000000	9.000000	
75%	1.004478e+06	14.000000	1.000000	8.000000	15.000000	
max	1.006040e+06	20.000000	1.000000	20.000000	18.000000	

In [19]:
```
df.describe(include = ['object']) # for categorical columns
```

Out[19]:

	Product_ID	Gender	Age	City_Category	Stay_In_Current_City_Years
count	550068	550068	550068	550068	550068
unique	3631	2	7	3	5
top	P00265242	M	26-35	B	1
freq	1880	414259	219587	231173	193821

In [20]:
```
round(df["Product_ID"].value_counts(normalize= True)*100,3) # % of each pdocuct id
```

Out[20]:
```
P00265242    0.342
P00025442    0.294
P00110742    0.293
P00112142    0.284
P00057642    0.267
             ...
P00314842    0.000
P00298842    0.000
P00231642    0.000
P00204442    0.000
P00066342    0.000
Name: Product_ID, Length: 3631, dtype: float64
```

HANDLING MISSING VALUES

In [21]:
```
df.isnull()
```

Out[21]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
550063	False	False	False	False	False	False	False
550064	False	False	False	False	False	False	False
550065	False	False	False	False	False	False	False
550066	False	False	False	False	False	False	False
550067	False	False	False	False	False	False	False

550068 rows × 12 columns

In [22]:
```
df.isnull().sum()
```

Out[22]:
```
User_ID                           0
Product_ID                        0
Gender                            0
Age                               0
Occupation                        0
City_Category                     0
Stay_In_Current_City_Years        0
Marital_Status                    0
Product_Category_1                0
Product_Category_2           173638
Product_Category_3           383247
Purchase                          0
dtype: int64
```

In [23]:
```
df_temp = df.drop('Product_Category_2',axis =1,inplace= False)
```

In [24]:
```
df_temp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 11 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   User_ID                     550068 non-null  int64  
 1   Product_ID                  550068 non-null  object 
 2   Gender                      550068 non-null  object 
 3   Age                         550068 non-null  object 
 4   Occupation                  550068 non-null  int64  
 5   City_Category               550068 non-null  object 
 6   Stay_In_Current_City_Years  550068 non-null  object 
 7   Marital_Status              550068 non-null  int64  
 8   Product_Category_1          550068 non-null  int64  
 9   Product_Category_3          166821 non-null  float64
 10  Purchase                    550068 non-null  int64  
dtypes: float64(1), int64(5), object(5)
memory usage: 46.2+ MB
```

In [29]:
```
df_temp = df.drop(df_temp[df_temp.Product_Category_2.isnull()].index,axis=0,inplace
```

In [30]:
```
df_temp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 376430 entries, 1 to 545914
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   User_ID                     376430 non-null  int64  
 1   Product_ID                  376430 non-null  object 
 2   Gender                      376430 non-null  object 
 3   Age                         376430 non-null  object 
 4   Occupation                  376430 non-null  int64  
 5   City_Category               376430 non-null  object 
 6   Stay_In_Current_City_Years  376430 non-null  object 
 7   Marital_Status              376430 non-null  int64  
 8   Product_Category_1          376430 non-null  int64  
 9   Product_Category_2          376430 non-null  float64
 10  Product_Category_3          166821 non-null  float64
 11  Purchase                    376430 non-null  int64  
dtypes: float64(2), int64(5), object(5)
memory usage: 37.3+ MB
```

In [31]:
```
df.head(10)
```

Out[31]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Mari
0	1000001	P00069042	F	0-17	10	A	2	
1	1000001	P00248942	F	0-17	10	A	2	
2	1000001	P00087842	F	0-17	10	A	2	
3	1000001	P00085442	F	0-17	10	A	2	
4	1000002	P00285442	M	55+	16	C	4+	
5	1000003	P00193542	M	26-35	15	A	3	
6	1000004	P00184942	M	46-50	7	B	2	
7	1000004	P00346142	M	46-50	7	B	2	
8	1000004	P0097242	M	46-50	7	B	2	
9	1000005	P00274942	M	26-35	20	A	1	

In [32]:
```
df=df.fillna(method = 'pad') # previous value replaces missing value
```

In [33]:
```
df.isnull().sum()
```

Out[33]:
```
User_ID                     0
Product_ID                  0
Gender                      0
Age                         0
Occupation                  0
City_Category               0
Stay_In_Current_City_Years  0
Marital_Status              0
Product_Category_1          0
Product_Category_2          1
Product_Category_3          1
Purchase                    0
dtype: int64
```

In [34]:
```
df = df.fillna(method = 'backfill') # next value replaces missing value
```

In [35]:
```
df.isnull().sum()
```

Out[35]:
```
User_ID                     0
Product_ID                  0
Gender                      0
Age                         0
Occupation                  0
City_Category               0
Stay_In_Current_City_Years  0
Marital_Status              0
Product_Category_1          0
Product_Category_2          0
Product_Category_3          0
Purchase                    0
dtype: int64
```

In [36]:
```
df1= df.copy()
```

In [37]:
```
df1.head(10)
```

Out[37]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Mari
0	1000001	P00069042	F	0-17	10	A	2	
1	1000001	P00248942	F	0-17	10	A	2	
2	1000001	P00087842	F	0-17	10	A	2	
3	1000001	P00085442	F	0-17	10	A	2	
4	1000002	P00285442	M	55+	16	C	4+	
5	1000003	P00193542	M	26-35	15	A	3	
6	1000004	P00184942	M	46-50	7	B	2	
7	1000004	P00346142	M	46-50	7	B	2	
8	1000004	P0097242	M	46-50	7	B	2	
9	1000005	P00274942	M	26-35	20	A	1	

In [38]:
```
df1.loc[0,'Product_ID'] # first row of dataframe using loc
```

Out[38]:
```
'P00069042'
```

In [39]:
```
df1.loc[:,'Purchase'] # print purchase for all rows using loc
```

Out[39]:
```
0          8370
1         15200
2          1422
3          1057
4          7969
          ...
550063      368
550064      371
550065      137
550066      365
550067      490
Name: Purchase, Length: 550068, dtype: int64
```

In [40]:
```
df1.loc[:4,'Purchase']
```

Out[40]:
```
0     8370
1    15200
2     1422
3     1057
4     7969
Name: Purchase, dtype: int64
```

In [41]:
```
df1.loc[:,['Age', 'Occupation']]
```

Out[41]:

	Age	Occupation
0	0-17	10
1	0-17	10
2	0-17	10
3	0-17	10
4	55+	16
...	...	...
550063	51-55	13
550064	26-35	1
550065	26-35	15
550066	55+	1
550067	46-50	0

550068 rows × 2 columns

In [44]:
```
df1.loc[[0,1,2,3,4],['Age', 'Occupation']]
```

Out[44]:

	Age	Occupation
0	0-17	10
1	0-17	10
2	0-17	10
3	0-17	10
4	55+	16

In [47]:
```
df1.loc[2:4] # print all columns for 2nd, 3rd amd 4th row
```

Out[47]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Mari
2	1000001	P00087842	F	0-17	10	A	2	
3	1000001	P00085442	F	0-17	10	A	2	
4	1000002	P00285442	M	55+	16	C	4+	

In [48]:
```
df1.iloc[0]
```

Out[48]:
```
User_ID                     1000001
Product_ID                  P00069042
Gender                          F
Age                          0-17
Occupation                     10
City_Category                   A
Stay_In_Current_City_Years      2
Marital_Status                  0
Product_Category_1              3
Product_Category_2            6.0
Product_Category_3           14.0
Purchase                     8370
Name: 0, dtype: object
```

In [49]:
```
df1.iloc[:,0:5]
```

Out[49]:

	User_ID	Product_ID	Gender	Age	Occupation
0	1000001	P00069042	F	0-17	10
1	1000001	P00248942	F	0-17	10
2	1000001	P00087842	F	0-17	10
3	1000001	P00085442	F	0-17	10
4	1000002	P00285442	M	55+	16
...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13
550064	1006035	P00375436	F	26-35	1
550065	1006036	P00375436	F	26-35	15
550066	1006038	P00375436	F	55+	1
550067	1006039	P00371644	F	46-50	0

550068 rows × 5 columns

In [50]:
```
df1.iloc[[0,4,9],[0,3,6]] # select 1st, 5th and 10th row , 1st, 4th and 7th column
```

Out[50]:

	User_ID	Age	Stay_In_Current_City_Years
0	1000001	0-17	2
4	1000002	55+	4+
9	1000005	26-35	1

In [52]:
```
df1['Purchase'].idxmax()   # index of first occurance of max purchase value
```

Out[52]:
```
87440
```

In [53]:
```
df1.Purchase[df1['Purchase'].idxmax()]  # max purchase value
```

Out[53]:
```
23961
```

In [54]:
```
df1[df1["Purchase"]==23961]    # rows with maximum purchase
```

Out[54]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
87440	1001474	P00052842	M	26-35	4	A	2
93016	1002272	P00052842	M	26-35	0	C	1
370891	1003160	P00052842	M	26-35	17	C	3

In [55]:
```
df1.loc[df1[df1["Purchase"]==23961].index] # rows with maximum values using loc
```

Out[55]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
87440	1001474	P00052842	M	26-35	4	A	2
93016	1002272	P00052842	M	26-35	0	C	1
370891	1003160	P00052842	M	26-35	17	C	3

In [56]:
```
df1.at[2,'Purchase']
```

Out[56]:
```
1422
```

In [58]:
```
df1.iat[2,11] # value at 3rd row and 11th column
```

Out[58]:
```
1422
```

In [59]:
```
df1.loc[((df1['User_ID'] == 1006039) & (df1['Product_ID'] =='P00371644')), 'Purchas
```

Out[59]:
```
550067    490
Name: Purchase, dtype: int64
```

In [60]:
```
df1[(df1['City_Category'] == 'A') & (df1['Stay_In_Current_City_Years'] =='4+') & (d
#user who are in city A with more than 4 years and purchace greater than 10000
```

Out[60]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
98	1000022	P00351142	M	18-25	15	A	4+
100	1000022	P00195942	M	18-25	15	A	4+
102	1000022	P0098242	M	18-25	15	A	4+
103	1000022	P00262242	M	18-25	15	A	4+
416	1000073	P00351142	M	18-25	4	A	4+
...	...	...	...	...	...	...	...
545791	1006019	P00279442	M	26-35	0	A	4+
545792	1006019	P00262342	M	26-35	0	A	4+
545793	1006019	P00028842	M	26-35	0	A	4+
545794	1006019	P00070342	M	26-35	0	A	4+
545832	1006028	P0097242	M	18-25	4	A	4+

6947 rows × 12 columns

In [64]:
```
df1[(df1['Gender'] == 'F') & (df1['City_Category'] == 'B') & (df1['Stay_In_Current_
#users whose gender is female in city B with 3 years and purchase less than 5000
```

Out[64]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
70	1000018	P00366542	F	18-25	3	B	3
72	1000018	P00151842	F	18-25	3	B	3
80	1000018	P0094142	F	18-25	3	B	3
743	1000140	P00062842	F	36-45	1	B	3
744	1000140	P00084642	F	36-45	1	B	3
...	...	...	...	...	...	...	...
549774	1005618	P00370293	F	18-25	4	B	3
549785	1005635	P00370293	F	36-45	9	B	3
549826	1005687	P00370293	F	0-17	0	B	3
549866	1005752	P00375436	F	55+	0	B	3
549937	1005856	P00370293	F	26-35	7	B	3

1951 rows × 12 columns

In [65]:
```
newdf = df1.mask(df1['Occupation']!=10) # visualize record with value 10 & mask eve
```

In [66]:
```
newdf.head(10)
```

Out[66]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Ma
0	1000001.0	P00069042	F	0-17	10.0	A	2	
1	1000001.0	P00248942	F	0-17	10.0	A	2	
2	1000001.0	P00087842	F	0-17	10.0	A	2	
3	1000001.0	P00085442	F	0-17	10.0	A	2	
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

In [67]:
```
df1.sort_index()  # sort row wise
```

Out[67]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
0	1000001	P00069042	F	0-17	10	A	2
1	1000001	P00248942	F	0-17	10	A	2
2	1000001	P00087842	F	0-17	10	A	2
3	1000001	P00085442	F	0-17	10	A	2
4	1000002	P00285442	M	55+	16	C	4+
...	...	...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13	B	1
550064	1006035	P00375436	F	26-35	1	C	3
550065	1006036	P00375436	F	26-35	15	B	4+
550066	1006038	P00375436	F	55+	1	C	2
550067	1006039	P00371644	F	46-50	0	B	4+

550068 rows × 12 columns

In [68]:
```
df1.sort_index(axis =1)  # sort column wise
```

Out[68]:

	Age	City_Category	Gender	Marital_Status	Occupation	Product_Category_1	Product_Cat
0	0-17	A	F	0	10	3	
1	0-17	A	F	0	10	1	
2	0-17	A	F	0	10	12	
3	0-17	A	F	0	10	12	
4	55+	C	M	0	16	8	
...	...	...	...	...	...	...	
550063	51-55	B	M	1	13	20	
550064	26-35	C	F	0	1	20	
550065	26-35	B	F	1	15	20	
550066	55+	C	F	0	1	20	
550067	46-50	B	F	1	0	20	

550068 rows × 12 columns

In [69]:
```
df1.sort_index(ascending = False) # sort in desceding order row wise
```

Out[69]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
550067	1006039	P00371644	F	46-50	0	B	4+
550066	1006038	P00375436	F	55+	1	C	2
550065	1006036	P00375436	F	26-35	15	B	4+
550064	1006035	P00375436	F	26-35	1	C	3
550063	1006033	P00372445	M	51-55	13	B	1
...	...	...	...	...	...	...	...
4	1000002	P00285442	M	55+	16	C	4+
3	1000001	P00085442	F	0-17	10	A	2
2	1000001	P00087842	F	0-17	10	A	2
1	1000001	P00248942	F	0-17	10	A	2
0	1000001	P00069042	F	0-17	10	A	2

550068 rows × 12 columns

In [70]:
```
df1.sort_values(by = ['Purchase']) # sort by value
```

Out[70]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
549221	1004806	P00370293	M	26-35	17	C	2
549477	1005184	P00370293	M	18-25	20	B	4+
547819	1002802	P00370853	M	36-45	20	B	2
548027	1003105	P00370853	M	36-45	12	C	4+
547538	1002402	P00370853	M	46-50	17	B	4+
...	...	...	...	...	...	...	...
292083	1003045	P00052842	M	46-50	1	B	2
503697	1005596	P00117642	M	36-45	12	B	1
370891	1003160	P00052842	M	26-35	17	C	3
87440	1001474	P00052842	M	26-35	4	A	2
93016	1002272	P00052842	M	26-35	0	C	1

550068 rows × 12 columns

In [71]:
```
df1.sort_values(by = ['Age','Purchase']) # multiple column
```

Out[71]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
546045	1000194	P00370853	F	0-17	10	C	3
546449	1000775	P00370853	M	0-17	17	C	1
550024	1005973	P00370293	M	0-17	10	C	4+
545971	1000086	P00370853	F	0-17	10	C	3
549145	1004707	P00370293	M	0-17	0	C	4+
...	...	...	...	...	...	...	...
121808	1000837	P00085342	M	55+	7	C	1
56879	1002788	P00085342	M	55+	1	B	0
366333	1002359	P00085342	M	55+	13	C	1
7542	1001178	P00116142	M	55+	0	C	1
321782	1001577	P00052842	M	55+	0	C	1

550068 rows × 12 columns

In [72]:
```
df1.sort_values(by ='Purchase', ascending = False) # sort by descending order
```

Out[72]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
370891	1003160	P00052842	M	26-35	17	C	3
93016	1002272	P00052842	M	26-35	0	C	1
87440	1001474	P00052842	M	26-35	4	A	2
503697	1005596	P00117642	M	36-45	12	B	1
321782	1001577	P00052842	M	55+	0	C	1
...	...	...	...	...	...	...	...
546379	1000671	P00370853	M	18-25	4	C	0
546185	1000391	P00370293	M	46-50	11	C	2
547032	1001649	P00370293	M	18-25	19	C	2
546181	1000387	P00370293	F	36-45	7	C	0
549221	1004806	P00370293	M	26-35	17	C	2

550068 rows × 12 columns

In [73]:
```
# top 20 most revenue generated customersand their purchased product id
top20 = df1.sort_values(by ='Purchase', ascending = False).iloc[:20,:]
```

In [74]:
```
top20
```

Out[74]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
370891	1003160	P00052842	M	26-35	17	C	3
93016	1002272	P00052842	M	26-35	0	C	1
87440	1001474	P00052842	M	26-35	4	A	2
503697	1005596	P00117642	M	36-45	12	B	1
321782	1001577	P00052842	M	55+	0	C	1
349658	1005848	P00119342	M	51-55	20	A	0
292083	1003045	P00052842	M	46-50	1	B	2
298378	1003947	P00116142	M	26-35	0	C	3
437804	1001387	P00086242	F	51-55	13	B	1
229329	1005367	P00085342	M	18-25	4	A	1
416883	1004117	P00161842	M	18-25	4	B	4+
7542	1001178	P00116142	M	55+	0	C	1
373300	1003511	P00085342	M	51-55	0	C	2
33268	1005102	P00052842	M	26-35	12	C	2
388010	1005716	P00052842	M	0-17	10	C	4+
449656	1003301	P00086242	F	26-35	2	B	3
366333	1002359	P00085342	M	55+	13	C	1
54364	1002274	P00052842	M	18-25	2	B	3
56879	1002788	P00085342	M	55+	1	B	0
68926	1004520	P00116142	M	26-35	4	C	1

In [75]:
```
top20.User_ID.values # top 20 user id
```

Out[75]:
```
array([1003160, 1002272, 1001474, 1005596, 1001577, 1005848, 1003045,
       1003947, 1001387, 1005367, 1004117, 1001178, 1003511, 1005102,
       1005716, 1003301, 1002359, 1002274, 1002788, 1004520], dtype=int64)
```

In [79]:
```
top20.Product_ID.value_counts() # products included in top20
```

Out[79]:
```
P00052842    8
P00085342    4
P00116142    3
P00086242    2
P00117642    1
P00119342    1
P00161842    1
Name: Product_ID, dtype: int64
```

In [81]:
```
df1['Age'].value_counts(ascending= False) # age group much active for purchasing pr
```

Out[81]:
```
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: Age, dtype: int64
```

In [82]:
```
df1['Gender']= df1['Gender'].replace('F', 'Female') # replacing column values
df1['Gender']= df1['Gender'].replace('M', 'Male')
```

In [83]:
```
df1.head(10)
```

Out[83]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Mari
0	1000001	P00069042	Female	0-17	10	A	2	
1	1000001	P00248942	Female	0-17	10	A	2	
2	1000001	P00087842	Female	0-17	10	A	2	
3	1000001	P00085442	Female	0-17	10	A	2	
4	1000002	P00285442	Male	55+	16	C	4+	
5	1000003	P00193542	Male	26-35	15	A	3	
6	1000004	P00184942	Male	46-50	7	B	2	
7	1000004	P00346142	Male	46-50	7	B	2	
8	1000004	P0097242	Male	46-50	7	B	2	
9	1000005	P00274942	Male	26-35	20	A	1	

In [85]:
```
df1[['User_ID','Age']].value_counts() # count of purchase for all distinct user id
```

Out[85]:
```
User_ID  Age
1001680  26-35    1026
1004277  36-45     979
1001941  36-45     898
1001181  36-45     862
1000889  46-50     823
                  ...
1002111  55+         7
1005391  26-35       7
1002690  26-35       7
1005608  18-25       7
1000708  26-35       6
Length: 5891, dtype: int64
```

In [86]:
```
import numpy as np
```

In [87]:
```
df1['Purchase'].describe() # statistical values on purchase column
```

Out[87]:
```
count    550068.000000
mean       9263.968713
std        5023.065394
min          12.000000
25%        5823.000000
50%        8047.000000
75%       12054.000000
max       23961.000000
Name: Purchase, dtype: float64
```

In [88]:
```
df1['Purchase'].aggregate(np.sum) # total purchase amount using np.sum
```

Out[88]:
```
5095812742
```

In [90]:
```
df1['Purchase'].aggregate([np.sum, np.mean]) # sum and mean on purchase
```

Out[90]:
```
sum     5.095813e+09
mean    9.263969e+03
Name: Purchase, dtype: float64
```

In [92]:
```
df1[['Product_Category_1','Product_Category_2','Product_Category_3']].aggregate(np.
```

Out[92]:
```
Product_Category_1    2972716.0
Product_Category_2    5425425.0
Product_Category_3    6958758.0
dtype: float64
```

In [93]:
```
df1[['Product_Category_1','Product_Category_2','Product_Category_3']].aggregate([np
```

Out[93]:

	Product_Category_1	Product_Category_2	Product_Category_3
sum	2.972716e+06	5.425425e+06	6.958758e+06
mean	5.404270e+00	9.863190e+00	1.265072e+01

In [94]:
```
df1.Product_Category_1.apply(lambda x:x*10) # apply function on Product_Category_1
```

Out[94]:
```
0          30
1          10
2         120
3         120
4          80
          ...
550063    200
550064    200
550065    200
550066    200
550067    200
Name: Product_Category_1, Length: 550068, dtype: int64
```

In [95]:
```
# tag records to 'high focused' transaction where purchase amount is more than 5000
df1['Category'] = df.Purchase.apply(lambda x: 'High Focused' if x>5000 else 'Genera
```

In [96]:
```
df1.head()
```

Out[96]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Mari
0	1000001	P00069042	Female	0-17	10	A	2	
1	1000001	P00248942	Female	0-17	10	A	2	
2	1000001	P00087842	Female	0-17	10	A	2	
3	1000001	P00085442	Female	0-17	10	A	2	
4	1000002	P00285442	Male	55+	16	C	4+	

In [98]:
```
df1.Category.value_counts() # count of high focused and general in category
```

Out[98]:
```
High Focused    455145
General          94923
Name: Category, dtype: int64
```

In [99]:
```
df1.groupby('Gender').groups # group gender
```

Out[99]:
```
{'Female': [0, 1, 2, 3, 14, 15, 16, 17, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 3
9, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 65, 66, 70, 71, 72, 73, 74, 75, 76, 77,
78, 79, 80, 81, 82, 83, 84, 124, 125, 126, 147, 148, 149, 150, 151, 156, 157, 158,
163, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 219, 222, 223, 248, 249, 25
0, 251, 252, 253, 254, 255, 256, 257, 297, 298, 299, 355, 356, 357, 358, 359, 360,
361, 362, 363, 364, 365, 366, 367, 368, 369, 373, ...], 'Male': [4, 5, 6, 7, 8, 9,
10, 11, 12, 13, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 50, 51, 52, 53, 54, 5
5, 56, 57, 58, 59, 60, 61, 62, 63, 64, 67, 68, 69, 85, 86, 87, 88, 89, 90, 91, 92,
93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110,
111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 127, 128, 129, 13
0, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146,
152, 153, ...]}
```

In [100…
```
df1.groupby(['Gender', 'Age']).groups # group based on gender and age combination
```

Out[100]:
{('Female', '0-17'): [0, 1, 2, 3, 299, 423, 424, 425, 426, 427, 428, 429, 430, 43
1, 432, 467, 468, 539, 540, 541, 542, 543, 617, 618, 619, 620, 621, 1150, 1151, 13
04, 1305, 1306, 2905, 2907, 3010, 3715, 3804, 3805, 3806, 3807, 3808, 3835, 3836,
4551, 4552, 4553, 4554, 4555, 5453, 6431, 6759, 6760, 6761, 6762, 6763, 6764, 676
5, 6766, 6767, 6768, 6769, 6770, 6771, 6772, 6773, 6774, 6775, 6776, 6777, 6778, 6
779, 6780, 6781, 6782, 6783, 6784, 6785, 6786, 6787, 6788, 6789, 6790, 6791, 6792,
6793, 6794, 6795, 6796, 6797, 6798, 6799, 6800, 6801, 6802, 6803, 6804, 6805, 680
6, 6807, 6808, ...], ('Female', '18-25'): [70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
80, 81, 82, 83, 84, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 222, 223, 49
5, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 547, 548,
549, 550, 625, 910, 911, 912, 913, 914, 1046, 1228, 1267, 1268, 1269, 1490, 1491,
1492, 1493, 1494, 1495, 1496, 1497, 1498, 1499, 1500, 1552, 1553, 1554, 1555, 155
6, 1665, 1666, 1667, 1668, 1669, 1670, 1671, 1672, 1673, 1674, 1675, 1676, 1677, 1
678, 1822, 1903, 1904, 1905, 1947, 1948, 1949, 1950, 1951, 1952, 1953, 1954, 1959,
...], ('Female', '26-35'): [47, 48, 49, 124, 125, 126, 147, 148, 149, 150, 151, 16
3, 219, 297, 298, 406, 407, 454, 457, 458, 459, 460, 461, 529, 530, 585, 586, 691,
692, 693, 694, 695, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 74
1, 742, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854,
855, 856, 857, 858, 859, 860, 861, 862, 863, 1033, 1034, 1035, 1036, 1037, 1038, 1
039, 1040, 1041, 1042, 1043, 1044, 1045, 1085, 1086, 1087, 1088, 1364, 1365, 1369,
1565, 1627, 1628, 1629, 1630, 1631, 1632, 1633, 1634, 1635, ...], ('Female', '36-4
5'): [29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 65,
66, 156, 157, 158, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 38
5, 386, 387, 531, 532, 533, 534, 535, 536, 537, 538, 566, 567, 568, 743, 744, 757,
758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 77
4, 775, 776, 777, 778, 779, 780, 1187, 1188, 1189, 1190, 1191, 1229, 1230, 1231, 1
232, 1233, 1652, 1653, 1741, 1770, 1771, 1772, 1773, 1774, 2197, 2198, 2199, 2200,
2201, 2202, 2203, ...], ('Female', '46-50'): [248, 249, 250, 251, 252, 253, 254, 2
55, 256, 257, 414, 415, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 472, 47
3, 474, 654, 655, 656, 657, 658, 717, 718, 719, 720, 721, 722, 723, 724, 725, 879,
880, 881, 895, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1095, 1096, 1097, 1098, 1
099, 1100, 1101, 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1110, 1111, 1112,
1113, 1114, 1115, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1434, 143
5, 2549, 2550, 2551, 2552, 2553, 2554, 2555, 2556, 2557, 2558, 2559, 2560, 2561, 2
562, 2563, 2564, 2565, 2566, ...], ('Female', '51-55'): [14, 15, 16, 17, 355, 356,
357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 400, 401, 402, 99
7, 1467, 1468, 1469, 1470, 1471, 1472, 1473, 1474, 1475, 1476, 1477, 1478, 1479, 1
480, 1957, 1958, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993,
1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2026, 202
7, 2028, 2782, 2783, 2784, 3526, 3527, 3528, 3650, 3651, 3652, 3653, 3654, 3993, 3
994, 3995, 3996, 4177, 4178, 4179, 4180, 4755, 4901, 4902, 4903, 5625, 5626, 5630,
5631, 5632, 5633, 5884, 5885, 5886, 5887, 5888, 5889, ...], ('Female', '55+'): [47
5, 476, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971, 1972, 19
73, 1974, 1975, 1976, 1977, 1981, 1982, 2139, 2140, 2141, 2142, 2227, 2228, 2229,
2230, 2231, 2232, 3123, 3124, 3125, 3126, 3127, 3128, 3129, 3130, 3131, 3132, 313
3, 3134, 3655, 3656, 3657, 3658, 3659, 3660, 3687, 3688, 3689, 3690, 4693, 5444, 5
445, 5446, 5447, 5448, 5449, 5450, 5451, 5452, 5594, 5595, 5596, 5597, 5732, 5733,
5734, 5735, 5736, 5737, 5738, 5739, 5740, 5741, 5742, 5743, 5744, 5745, 5765, 576
6, 5767, 5768, 5769, 6077, 6078, 6404, 6405, 6406, 6407, 6408, 6409, 6931, 7780, 7
781, 7782, 7783, 8223, ...], ('Male', '0-17'): [85, 86, 87, 88, 89, 90, 91, 92, 9
3, 94, 95, 96, 865, 866, 867, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2243, 224
4, 2245, 2246, 2247, 2248, 2249, 2250, 2251, 3014, 3015, 3016, 3017, 3018, 3568, 3
569, 3570, 3571, 4375, 4526, 4527, 4528, 4529, 4530, 4531, 4532, 4647, 4648, 4649,
4650, 4651, 4652, 4653, 4654, 4655, 4656, 4698, 4699, 4771, 5059, 5060, 5061, 506
2, 5063, 5064, 5065, 5066, 5352, 5361, 5362, 5435, 5436, 5437, 5438, 5439, 5440, 5
441, 5624, 5725, 5821, 5822, 5823, 5824, 5919, 5920, 5921, 5922, 5923, 5924, 5925,
6032, 6033, 6112, 6113, 6114, 6115, 6520, 6521, ...], ('Male', '18-25'): [97, 98,
99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 127, 128, 129, 22
0, 221, 258, 259, 260, 261, 262, 263, 291, 292, 293, 294, 295, 296, 300, 301, 302,
303, 339, 340, 341, 388, 389, 390, 391, 392, 403, 404, 405, 408, 409, 416, 417, 41
8, 419, 420, 438, 439, 440, 462, 463, 464, 465, 466, 583, 584, 652, 653, 676, 677,
678, 679, 680, 681, 682, 683, 684, 685, 700, 701, 702, 703, 704, 705, 706, 707, 70
8, 709, 710, 711, 712, 713, 714, 715, 716, 726, 727, 728, 750, 751, 752, 753,
...], ('Male', '26-35'): [5, 9, 10, 11, 12, 13, 19, 20, 21, 22, 23, 24, 25, 26, 2

```
7, 28, 50, 51, 56, 57, 58, 59, 60, 61, 62, 63, 64, 130, 131, 132, 133, 134, 135, 1
36, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 196, 197, 198, 199, 200, 20
1, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217,
218, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 264, 265, 266, 267, 26
8, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284,
285, ...], ('Male', '36-45'): [18, 55, 112, 113, 114, 115, 116, 117, 118, 119, 12
0, 121, 122, 123, 152, 153, 154, 155, 335, 336, 337, 338, 393, 394, 395, 396, 397,
398, 421, 422, 433, 434, 435, 436, 437, 491, 492, 493, 494, 544, 545, 546, 551, 55
2, 553, 554, 555, 556, 557, 580, 581, 605, 606, 607, 608, 609, 610, 611, 612, 613,
614, 615, 616, 623, 624, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 63
7, 638, 639, 640, 641, 642, 643, 644, 665, 666, 667, 668, 669, 670, 671, 672, 673,
674, 675, 830, 831, 832, 833, 834, ...], ('Male', '46-50'): [6, 7, 8, 52, 53, 54,
164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 189, 19
0, 191, 192, 193, 194, 195, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245,
246, 247, 527, 528, 558, 559, 560, 561, 562, 563, 564, 565, 569, 570, 571, 572, 57
3, 574, 576, 577, 578, 646, 647, 648, 649, 650, 651, 898, 899, 900, 901, 902, 903,
904, 905, 906, 907, 908, 909, 1057, 1058, 1089, 1090, 1091, 1307, 1308, 1309, 132
3, 1324, 1325, 1326, 1327, 1328, 1329, 1330, 1331, 1332, 1333, 1334, 1335, 1336,
...], ('Male', '51-55'): [67, 68, 69, 333, 334, 370, 371, 788, 789, 790, 791, 792,
793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 868, 86
9, 870, 871, 1047, 1048, 1049, 1050, 1486, 1487, 1488, 1489, 1503, 1504, 1505, 150
6, 1681, 1682, 1683, 1738, 1739, 1740, 1775, 1776, 1777, 1778, 1779, 1780, 1781, 1
782, 1815, 1816, 1817, 1818, 1819, 1915, 1916, 1917, 1918, 1919, 1920, 1921, 1922,
2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 204
2, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2
056, 2124, 2175, ...], ('Male', '55+'): [4, 159, 160, 161, 162, 451, 452, 453, 47
7, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 645, 659, 893,
894, 1051, 1052, 1053, 1054, 1116, 1117, 1118, 1119, 1559, 1560, 1792, 1793, 1794,
1795, 1796, 1797, 1798, 1799, 1800, 1801, 1802, 1803, 1804, 1805, 1806, 1978, 197
9, 1980, 2016, 2017, 2018, 2096, 2097, 2322, 2510, 2614, 2615, 2766, 2767, 2768, 2
769, 2770, 2771, 2793, 2794, 2795, 2796, 2797, 2798, 2799, 3011, 3478, 3837, 3838,
3839, 3840, 3841, 3842, 3843, 3844, 3845, 4175, 4176, 4423, 4424, 4425, 4426, 469
4, 4695, 4696, 5052, 5053, 5083, 5084, ...]}
```

In [101…
```
df1.groupby('Gender').sum() # aggretaion function sum with group by
```

Out[101]:

Gender	User_ID	Occupation	Marital_Status	Product_Category_1	Product_Category_2	Produ
Female	136234060927	915426	56988	776517	1356094.0	
Male	415500008355	3527312	168349	2196199	4069331.0	

In [102…
```
df1.groupby('Gender')['Purchase'].aggregate(np.sum) # total purchased amount
```

Out[102]:
```
Gender
Female    1186232642
Male      3909580100
Name: Purchase, dtype: int64
```

In [103…
```
df1.groupby('Gender')['Purchase'].aggregate([np.sum, np.mean]) # sum and mean
```

Out[103]:

Gender	sum	mean
Female	1186232642	8734.565765
Male	3909580100	9437.526040

In [104…
```
df1[df1.groupby('Gender')['Purchase'].apply(lambda x :x>10000)] # apply function or
```

Out[104]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
1	1000001	P00248942	Female	0-17	10	A	2
5	1000003	P00193542	Male	26-35	15	A	3
6	1000004	P00184942	Male	46-50	7	B	2
7	1000004	P00346142	Male	46-50	7	B	2
8	1000004	P0097242	Male	46-50	7	B	2
...	...	...	...	...	...	...	...
545892	1006037	P00148642	Female	46-50	1	C	4+
545896	1006037	P00183142	Female	46-50	1	C	4+
545904	1006040	P00081142	Male	26-35	6	B	2
545908	1006040	P00127642	Male	26-35	6	B	2
545914	1006040	P00217442	Male	26-35	6	B	2

189450 rows × 13 columns

In [105…
```
df1.City_Category.value_counts() # values for city_category
```

Out[105]:
```
B    231173
C    171175
A    147720
Name: City_Category, dtype: int64
```

In [107…
```
df_dummy= pd.get_dummies(df1.City_Category, drop_first = True) # apply get dummies
```

In [108…
```
df_dummy.head()
```

Out[108]:

	B	C
0	0	0
1	0	0
2	0	0
3	0	0
4	0	1

In [109…
```
pd.concat([df1,df_dummy], axis = 1) #concatinate both dataframe
```

Out[109]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years
0	1000001	P00069042	Female	0-17	10	A	2
1	1000001	P00248942	Female	0-17	10	A	2
2	1000001	P00087842	Female	0-17	10	A	2
3	1000001	P00085442	Female	0-17	10	A	2
4	1000002	P00285442	Male	55+	16	C	4+
...	...	...	...	...	...	...	...
550063	1006033	P00372445	Male	51-55	13	B	1
550064	1006035	P00375436	Female	26-35	1	C	3
550065	1006036	P00375436	Female	26-35	15	B	4+
550066	1006038	P00375436	Female	55+	1	C	2
550067	1006039	P00371644	Female	46-50	0	B	4+

550068 rows × 15 columns

In [110…
```
df1.drop(['City_Category'],axis = 1, inplace = True) #drop original one
```

In [111…
```
df1.head()
```

Out[111]:

	User_ID	Product_ID	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Proc
0	1000001	P00069042	Female	0-17	10	2	0	
1	1000001	P00248942	Female	0-17	10	2	0	
2	1000001	P00087842	Female	0-17	10	2	0	
3	1000001	P00085442	Female	0-17	10	2	0	
4	1000002	P00285442	Male	55+	16	4+	0	

In [112…
```
df1.shape #verify shape
```

Out[112]:
```
(550068, 12)
```

In [ ]: