1)importing libraries

In [5]:
```
import pandas as pd
import numpy as np
```

2)Locating data frame 3)Locating pandas frame

In [18]:
```
df=pd.read_csv('train.csv')
```

4)Data Preprocessing- Checking missing values using isnull(),describe(), type of variables

In [19]:
```
df
```

Out[19]:

	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
0	1000001	P00069042	F	0-17	10	A	2	0	3	NaN	NaN	8370
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	14.0	15200
2	1000001	P00087842	F	0-17	10	A	2	0	12	NaN	NaN	1422
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	NaN	1057
4	1000002	P00285442	M	55+	16	C	4+	0	8	NaN	NaN	7969
...	...	...	...	...	...	...	...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13	B	1	1	20	NaN	NaN	368
550064	1006035	P00375436	F	26-35	1	C	3	0	20	NaN	NaN	371
550065	1006036	P00375436	F	26-35	15	B	4+	1	20	NaN	NaN	137
550066	1006038	P00375436	F	55+	1	C	2	0	20	NaN	NaN	365
550067	1006039	P00371644	F	46-50	0	B	4+	1	20	NaN	NaN	490

550068 rows × 12 columns

In [20]:
```
df.head(5)
```

Out[20]:

	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
0	1000001	P00069042	F	0-17	10	A	2	0	3	NaN	NaN	8370
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	14.0	15200
2	1000001	P00087842	F	0-17	10	A	2	0	12	NaN	NaN	1422
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	NaN	1057
4	1000002	P00285442	M	55+	16	C	4+	0	8	NaN	NaN	7969

In [21]:
```
df.tail(4)
```

Out[21]:

	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
550064	1006035	P00375436	F	26-35	1	C	3	0	20	NaN	NaN	371
550065	1006036	P00375436	F	26-35	15	B	4+	1	20	NaN	NaN	137
550066	1006038	P00375436	F	55+	1	C	2	0	20	NaN	NaN	365
550067	1006039	P00371644	F	46-50	0	B	4+	1	20	NaN	NaN	490

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

Out[22]:

	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
0	False	False	False	False	False	False	False	False	False	True	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	True	True	False
3	False	False	False	False	False	False	False	False	False	False	True	False
4	False	False	False	False	False	False	False	False	False	True	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...
550063	False	False	False	False	False	False	False	False	False	True	True	False
550064	False	False	False	False	False	False	False	False	False	True	True	False
550065	False	False	False	False	False	False	False	False	False	True	True	False
550066	False	False	False	False	False	False	False	False	False	True	True	False
550067	False	False	False	False	False	False	False	False	False	True	True	False

550068 rows × 12 columns

In [23]:
```
df.isnull().sum()
```

Out[23]:
```
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 [24]:
```
df.describe()
```

Out[24]:

	User_ID	Occupation	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
count	5.500680e+05	550068.000000	550068.000000	550068.000000	376430.000000	166821.000000	550068.000000
mean	1.003029e+06	8.076707	0.409653	5.404270	9.842329	12.668243	9263.968713
std	1.727592e+03	6.522660	0.491770	3.936211	5.086590	4.125338	5023.065394
min	1.000001e+06	0.000000	0.000000	1.000000	2.000000	3.000000	12.000000
25%	1.001516e+06	2.000000	0.000000	1.000000	5.000000	9.000000	5823.000000
50%	1.003077e+06	7.000000	0.000000	5.000000	9.000000	14.000000	8047.000000
75%	1.004478e+06	14.000000	1.000000	8.000000	15.000000	16.000000	12054.000000
max	1.006040e+06	20.000000	1.000000	20.000000	18.000000	18.000000	23961.000000

In [25]:
```
df.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 12 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_2          376430 non-null  float64
 10  Product_Category_3          166821 non-null  float64
 11  Purchase                    550068 non-null  int64  
dtypes: float64(2), int64(5), object(5)
memory usage: 50.4+ MB
```

In [26]:
```
df.shape
```

Out[26]:
```
(550068, 12)
```

5)Data formatting and data normalization-

a)Checking data types

In [27]:
```
df.dtypes
```

Out[27]:
```
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
```

b)Type conversion-from float to int

In [28]:
```
df['Product_Category_1'].astype(float)
```

Out[28]:
```
0         3.0
1         1.0
2        12.0
3        12.0
4         8.0
         ... 
550063    20.0
550064    20.0
550065    20.0
550066    20.0
550067    20.0
Name: Product_Category_1, Length: 550068, dtype: float64
```

6)Turn categorical values into quantitative variables-

a)Column City category contains A,B,C as values. Let us convert them to 1,2,3 using replace method

In [ ]:
```
Before converting
```

In [29]:
```
df['City_Category']
```

Out[29]:
```
0         A
1         A
2         A
3         A
4         C
         ..
550063    B
550064    C
550065    B
550066    C
550067    B
Name: City_Category, Length: 550068, dtype: object
```

In [36]:
```
a=df['City_Category'].replace(to_replace='A',value=1)
```

In [37]:
```
a
```

Out[37]:
```
0         1
1         1
2         1
3         1
4         C
         ..
550063    B
550064    C
550065    B
550066    C
550067    B
Name: City_Category, Length: 550068, dtype: object
```

Similarly can be done with value B and C

b)Using get dummies

In [44]:
```
b=pd.get_dummies(df,columns=['City_Category'])
b
```

Out[44]:

	User_ID	Product_ID	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase	City_Category_A	City_Category_B	City_Category_C
0	1000001	P00069042	F	0-17	10	2	0	3	NaN	NaN	8370	True	False	False
1	1000001	P00248942	F	0-17	10	2	0	1	6.0	14.0	15200	True	False	False
2	1000001	P00087842	F	0-17	10	2	0	12	NaN	NaN	1422	True	False	False
3	1000001	P00085442	F	0-17	10	2	0	12	14.0	NaN	1057	True	False	False
4	1000002	P00285442	M	55+	16	4+	0	8	NaN	NaN	7969	False	False	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13	1	1	20	NaN	NaN	368	False	True	False
550064	1006035	P00375436	F	26-35	1	3	0	20	NaN	NaN	371	False	False	True
550065	1006036	P00375436	F	26-35	15	4+	1	20	NaN	NaN	137	False	True	False
550066	1006038	P00375436	F	55+	1	2	0	20	NaN	NaN	365	False	False	True
550067	1006039	P00371644	F	46-50	0	4+	1	20	NaN	NaN	490	False	True	False

550068 rows × 14 columns

c)Using label encoding

In [47]:
```
from sklearn import preprocessing
l=preprocessing.LabelEncoder()
```

In [48]:
```
l.fit_transform(df['Gender'])
```

Out[48]:
```
array([0, 0, 0, ..., 0, 0, 0])
```