Import Modules

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
%matplotlib inline
warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_csv('Black Friday Sales.csv')
```

In [3]:
```
# Descripotive analysis
```

In [4]:
```
df.head()
```

Out[4]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Prod
0	1000001	P00069042	F	0-17	10	A	2	0	
1	1000001	P00248942	F	0-17	10	A	2	0	
2	1000001	P00087842	F	0-17	10	A	2	0	
3	1000001	P00085442	F	0-17	10	A	2	0	
4	1000002	P00285442	M	55+	16	C	4+	0	

In [5]:
```
# Datatype info
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 [6]:
```
# statistical info
df.describe()
```

Out[6]:

	User_ID	Occupation	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3

count	5.500680e+05	550068.000000	550068.000000	550068.000000	376430.000000	166821.000000
mean	1.003029e+06	8.076707	0.409653	5.404270	9.842329	12.668243
std	1.727592e+03	6.522660	0.491770	3.936211	5.086590	4.125338
min	1.000001e+06	0.000000	0.000000	1.000000	2.000000	3.000000
25%	1.001516e+06	2.000000	0.000000	1.000000	5.000000	9.000000
50%	1.003077e+06	7.000000	0.000000	5.000000	9.000000	14.000000
75%	1.004478e+06	14.000000	1.000000	8.000000	15.000000	16.000000
max	1.006040e+06	20.000000	1.000000	20.000000	18.000000	18.000000

In [7]:
```
# Unique values
df.apply(lambda x: len(x.unique()))
```

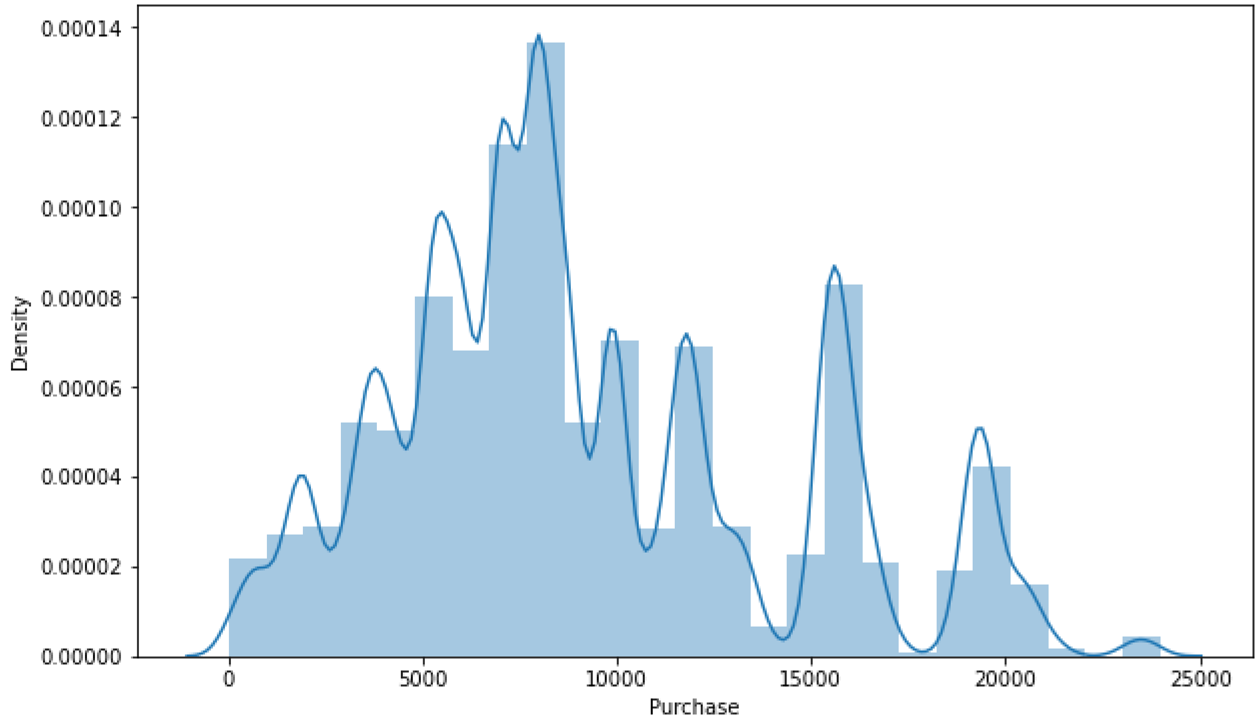
Out[7]:
```
User_ID                        5891
Product_ID                     3631
Gender                            2
Age                               7
Occupation                       21
City_Category                     3
Stay_In_Current_City_Years        5
Marital_Status                    2
Product_Category_1               20
Product_Category_2               18
Product_Category_3               16
Purchase                      18105
dtype: int64
```

In [8]:
```
# Null values
df.isnull().sum()
```

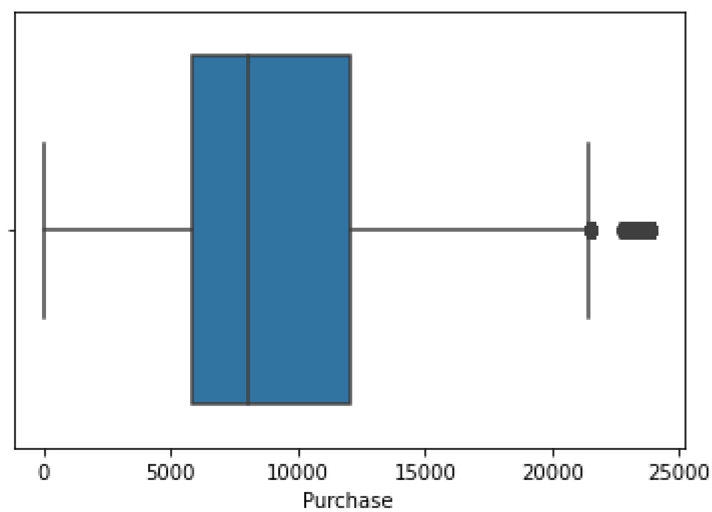
Out[8]:
```
User_ID                            0
Product_ID                         0
Gender                             0
Age                                0
Occupation                         0
City_Category                      0
Stay_In_Current_City_Years         0
Marital_Status                     0
Product_Category_1                 0
Product_Category_2            173638
Product_Category_3            383247
Purchase                           0
dtype: int64
```

Exploratory Data Ananalysis

In [9]:
```
plt.figure(figsize=(10,6))
sns.distplot(df['Purchase'] ,bins=25);
```

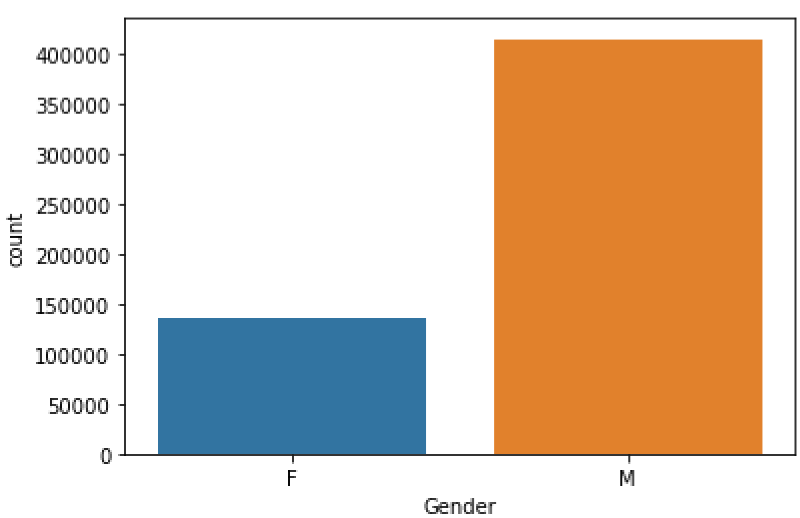


In [10]:
```
sns.boxplot(df['Purchase']);
```

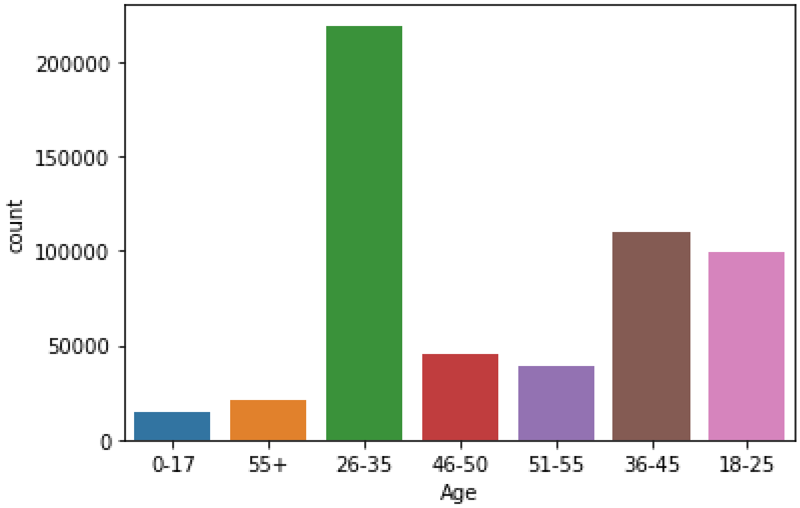


In [11]:
```
# We can see outliers in Purchase column
```

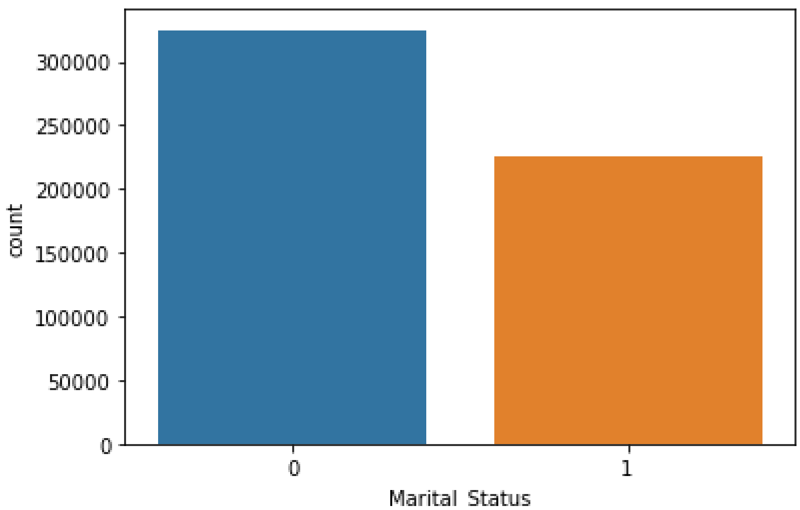
In [12]:
```
# dist of numeric variablres
sns.countplot(df.Gender);
```



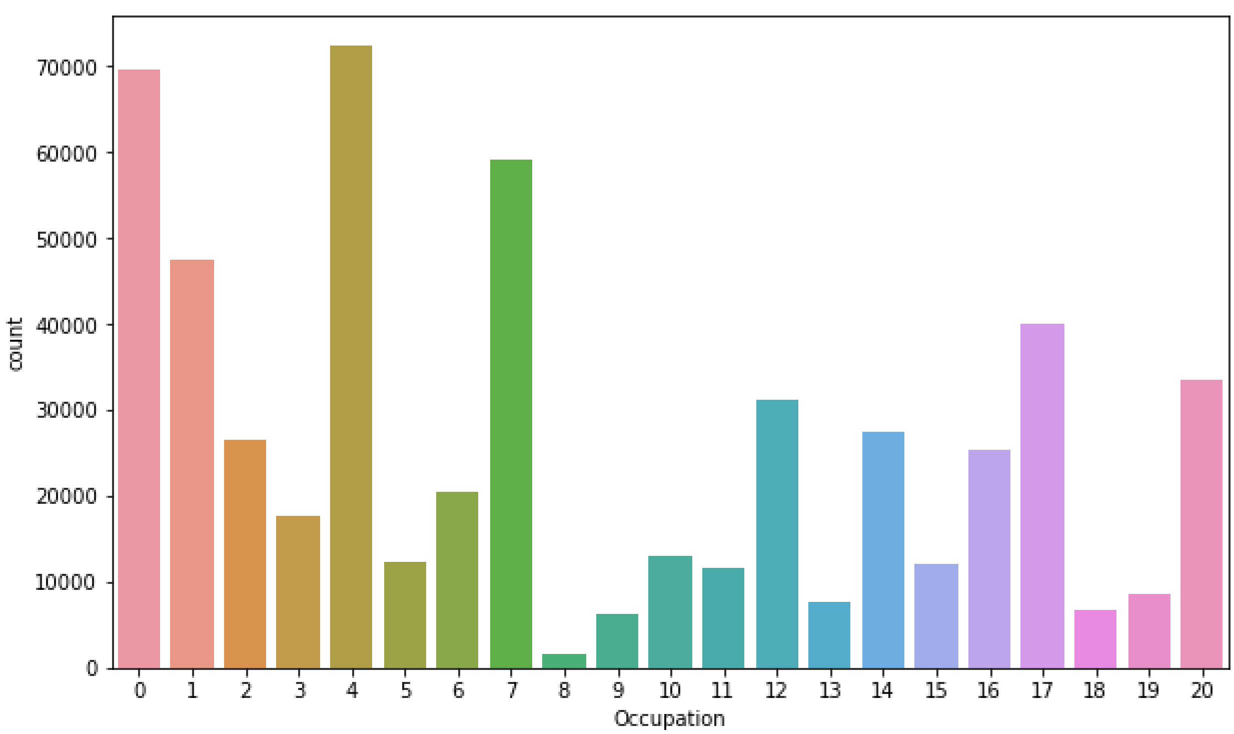
In [13]:
```
sns.countplot(df.Age);
```



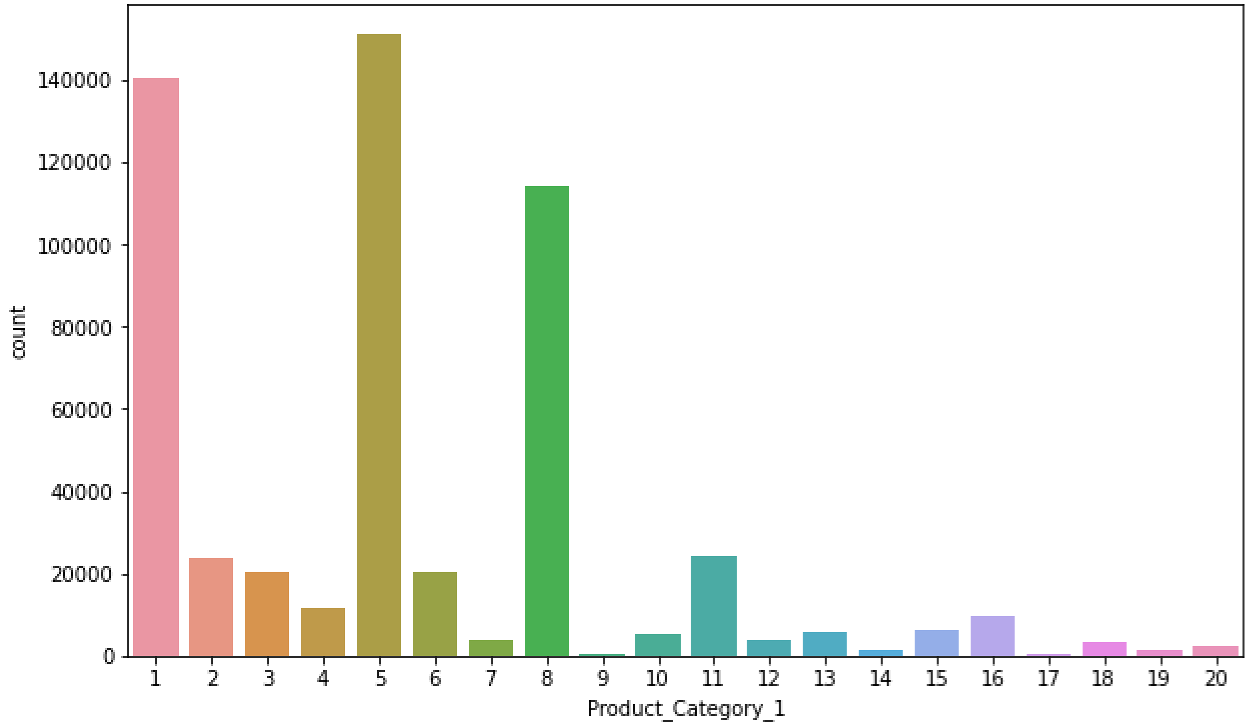
In [14]:
```
sns.countplot(df.Marital_Status);
```



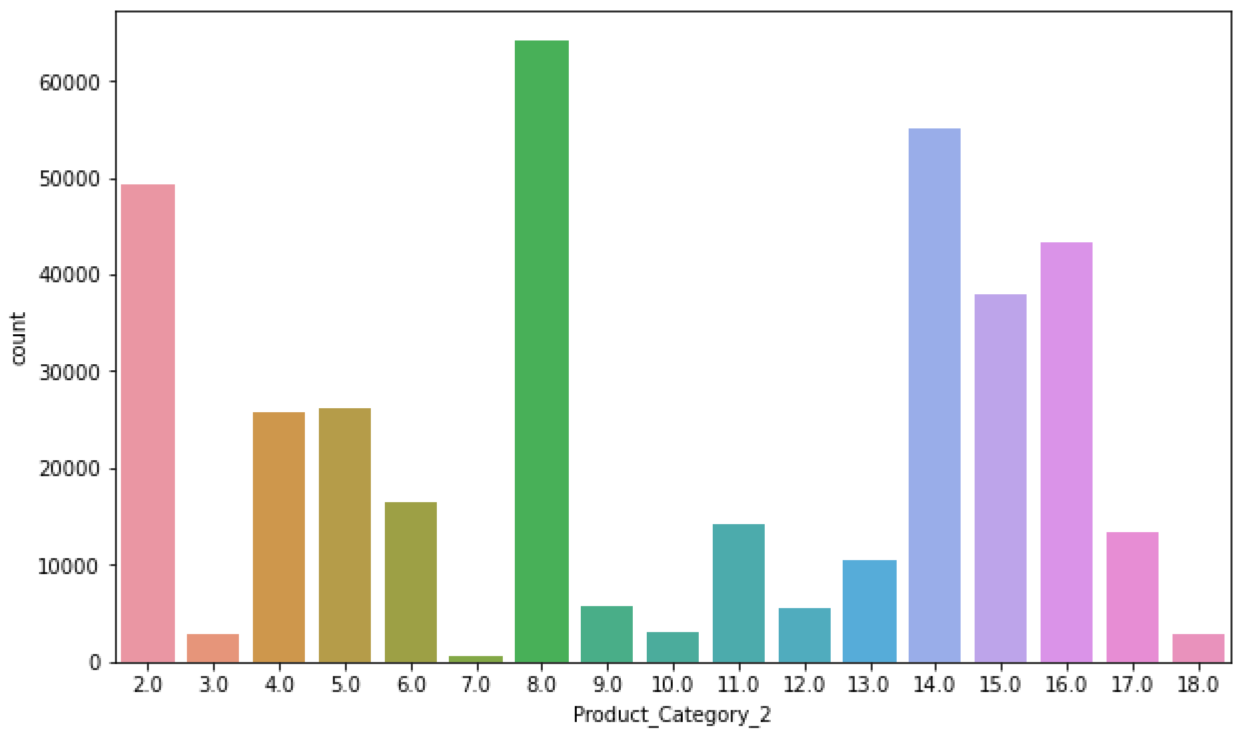
In [15]:
```
plt.figure(figsize=(10,6))
sns.countplot(df.Occupation);
```



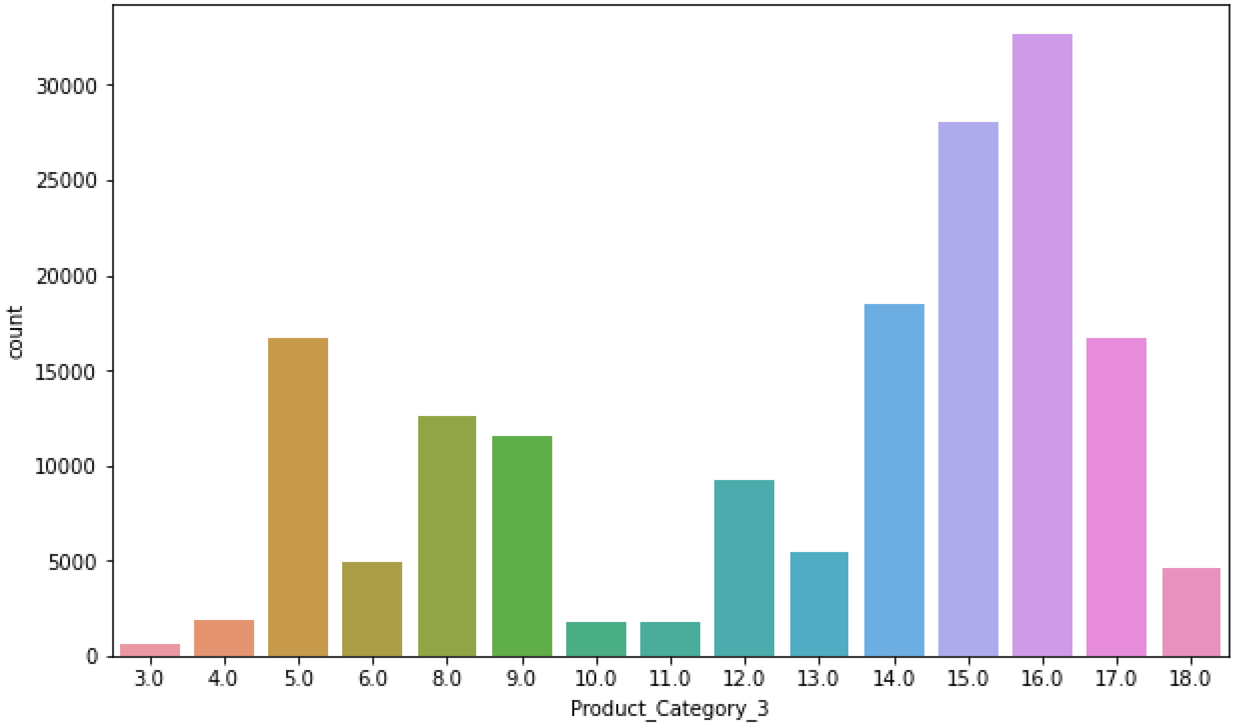
In [16]:
```python
plt.figure(figsize=(10,6))
sns.countplot(df.Product_Category_1);
```



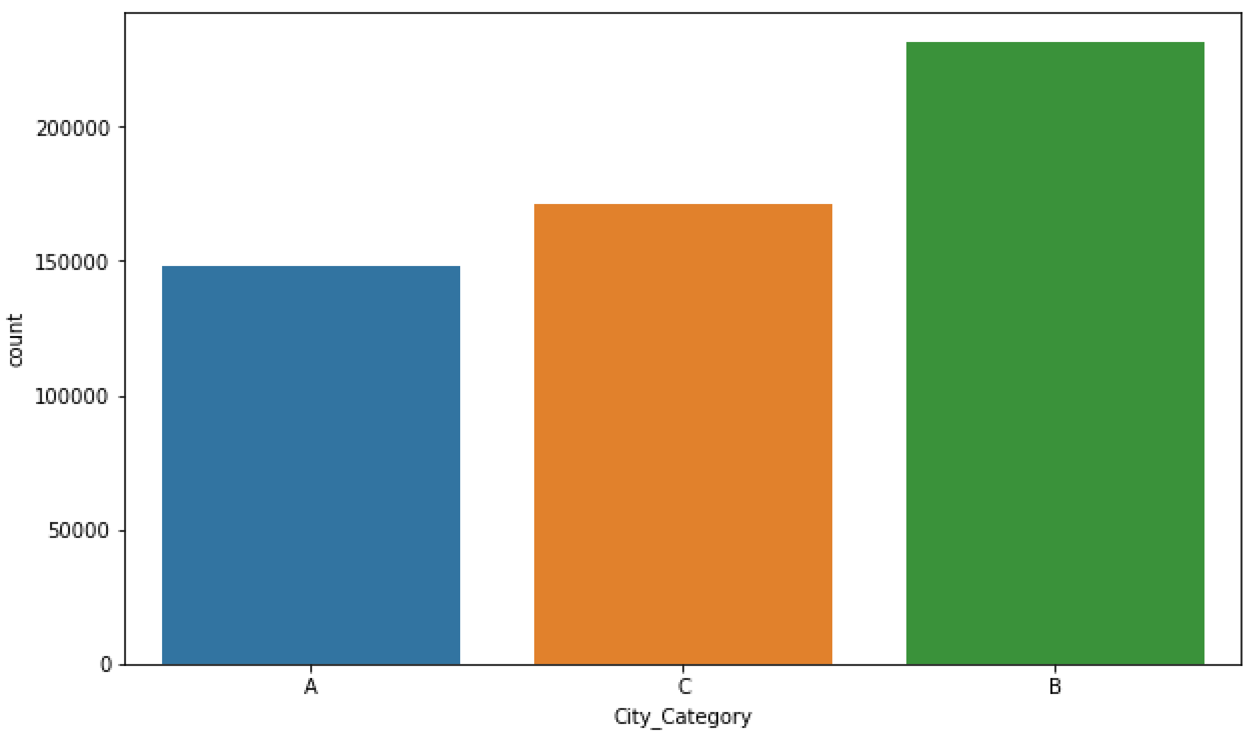
In [17]:
```python
plt.figure(figsize=(10,6))
sns.countplot(df.Product_Category_2);
```



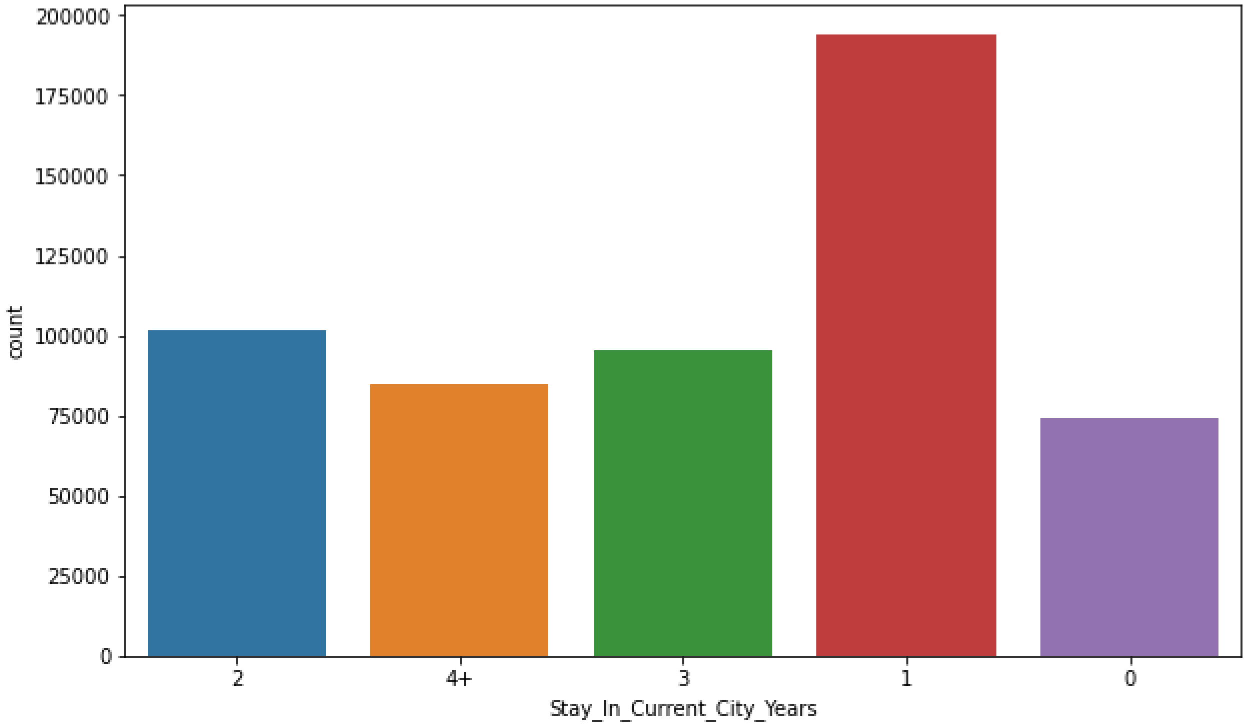
In [18]:
```python
plt.figure(figsize=(10,6))
sns.countplot(df.Product_Category_3);
```



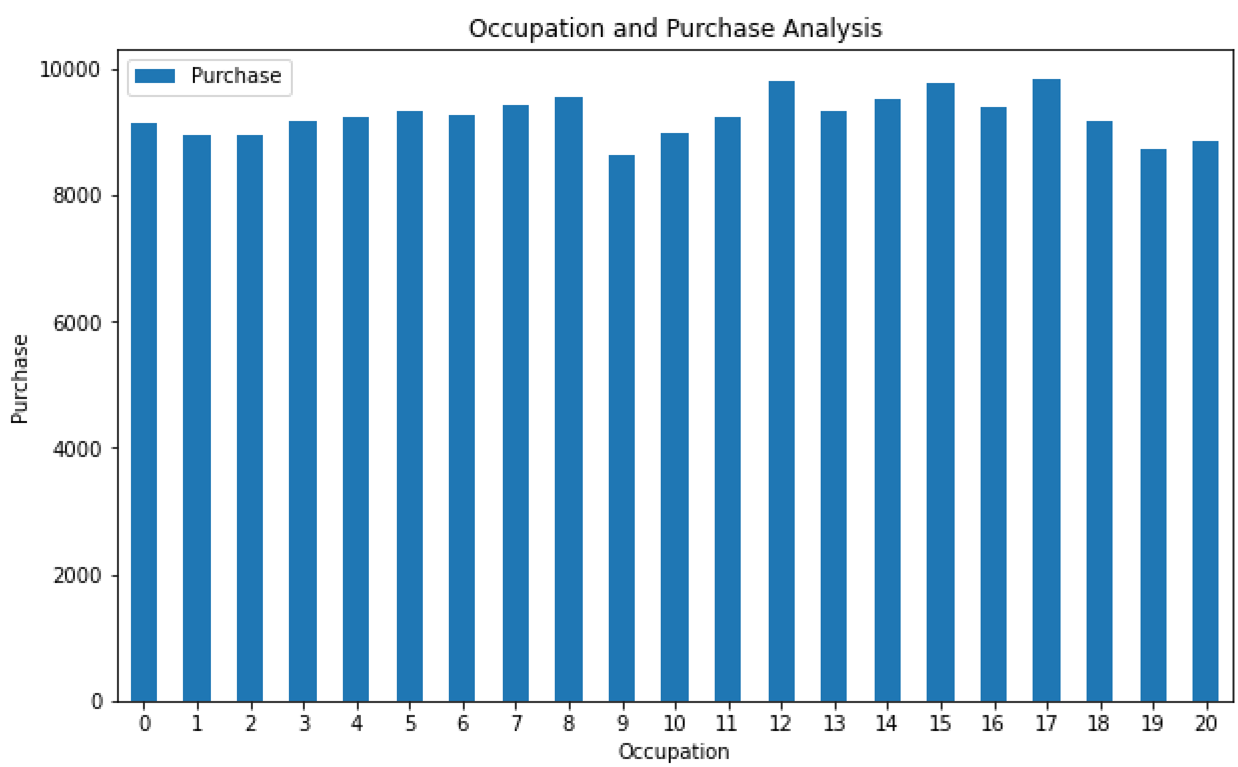
In [19]:
```
plt.figure(figsize=(10,6))
sns.countplot(df.City_Category);
```



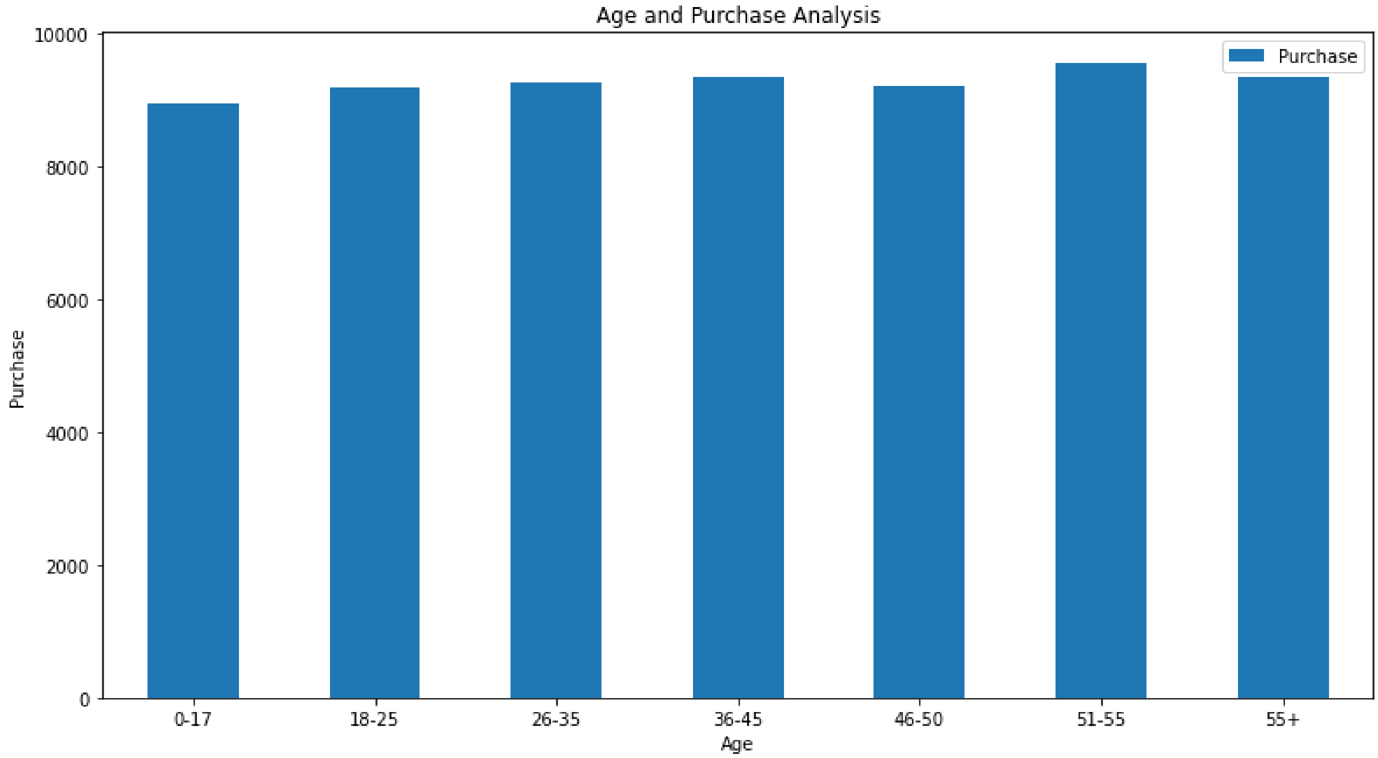
In [20]:
```
plt.figure(figsize=(10,6))
sns.countplot(df.Stay_In_Current_City_Years);
```



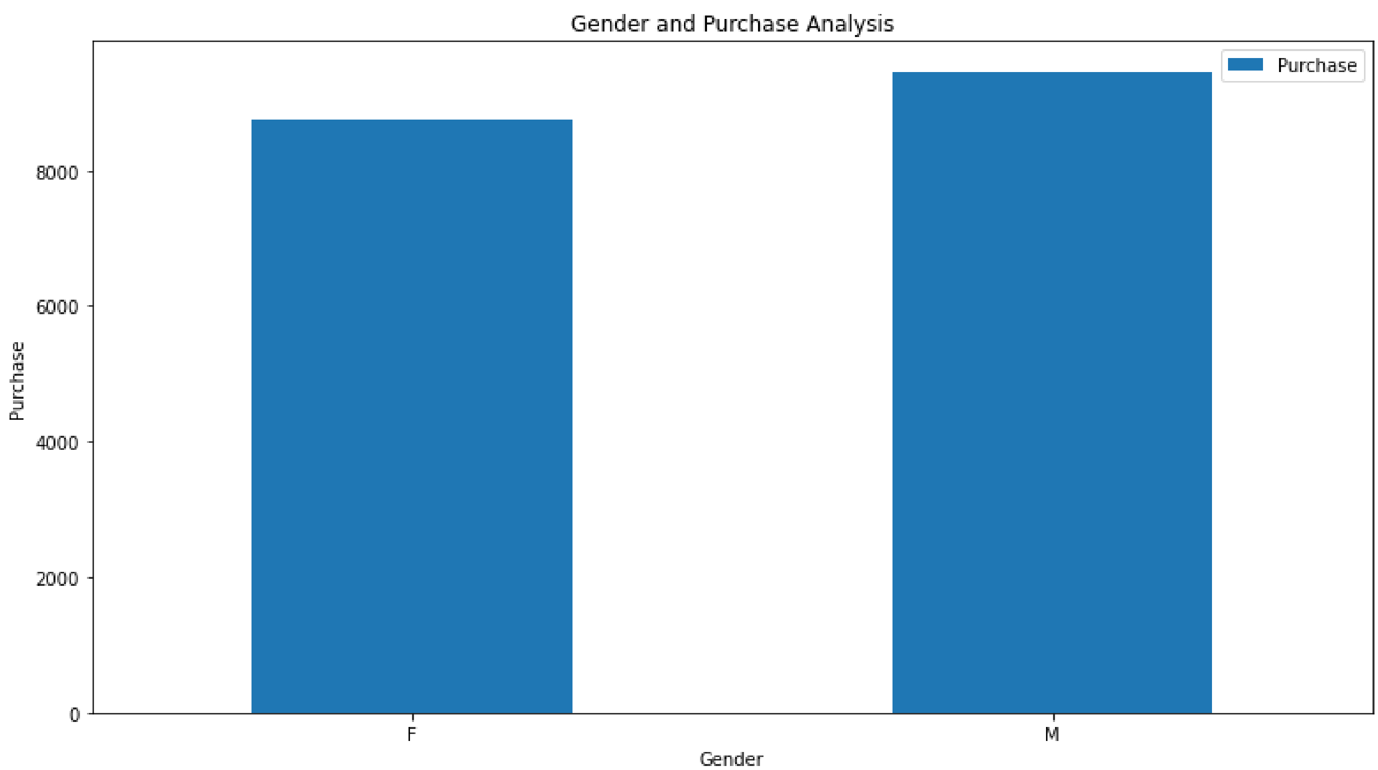
In [21]:
```
# Bivariate analysis
occupation_plot = df.pivot_table(index='Occupation', values='Purchase', aggfunc=np.mean)
occupation_plot.plot(kind='bar', figsize=(10, 6))
plt.xlabel('Occupation')
plt.ylabel("Purchase")
plt.title("Occupation and Purchase Analysis")
plt.xticks(rotation=0)
plt.show()
```



In [22]:
```
age_plot = df.pivot_table(index='Age', values='Purchase', aggfunc=np.mean)
age_plot.plot(kind='bar', figsize=(13, 7))
plt.xlabel('Age')
plt.ylabel("Purchase")
plt.title("Age and Purchase Analysis")
plt.xticks(rotation=0)
plt.show()
```



In [23]:
```
gender_plot = df.pivot_table(index='Gender', values='Purchase', aggfunc=np.mean)
gender_plot.plot(kind='bar', figsize=(13, 7))
plt.xlabel('Gender')
plt.ylabel("Purchase")
plt.title("Gender and Purchase Analysis")
plt.xticks(rotation=0)
plt.show()
```



In [24]:
```
df.head()
```

Out[24]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Prod
0	1000001	P00069042	F	0-17	10	A	2	0	

1	1000001	P00248942	F	0-17	10	A	2	0
2	1000001	P00087842	F	0-17	10	A	2	0
3	1000001	P00085442	F	0-17	10	A	2	0
4	1000002	P00285442	M	55+	16	C	4+	0

Preprocessing the Dataset

In [25]:
```
df.isnull().sum()
```

Out[25]:
```
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 [26]:
```
for i in df.columns:
    print(i," = " ,len(df[i].unique()))
```

```
User_ID  =  5891
Product_ID  =  3631
Gender  =  2
Age  =  7
Occupation  =  21
City_Category  =  3
Stay_In_Current_City_Years  =  5
Marital_Status  =  2
Product_Category_1  =  20
Product_Category_2  =  18
Product_Category_3  =  16
Purchase  =  18105
```

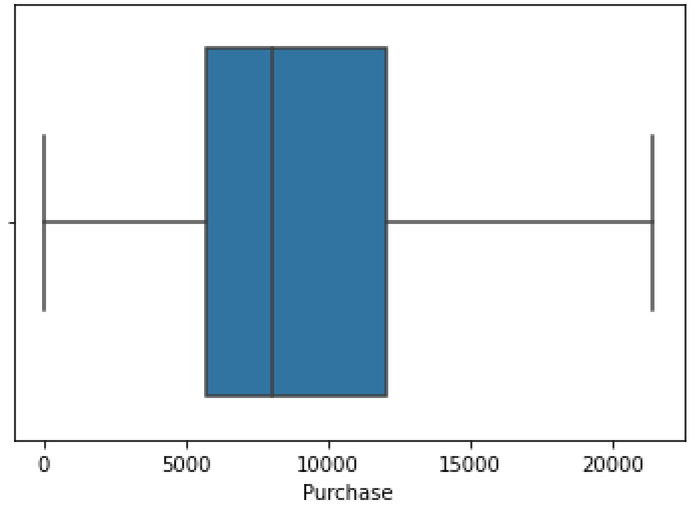
In [27]:
```
# Remove outliers using IQR technique
```

In [28]:
```
cols = ['Purchase']

Q1 = df[cols].quantile(0.25)
Q3 = df[cols].quantile(0.75)
IQR = Q3 - Q1

df = df[~((df[cols] < (Q1 - 1.5 * IQR)) |(df[cols] > (Q3 + 1.5 * IQR))).any(axis=1)]
```

In [ ]:

In [ ]:

In [29]:
```
sns.boxplot(df['Purchase']);
```

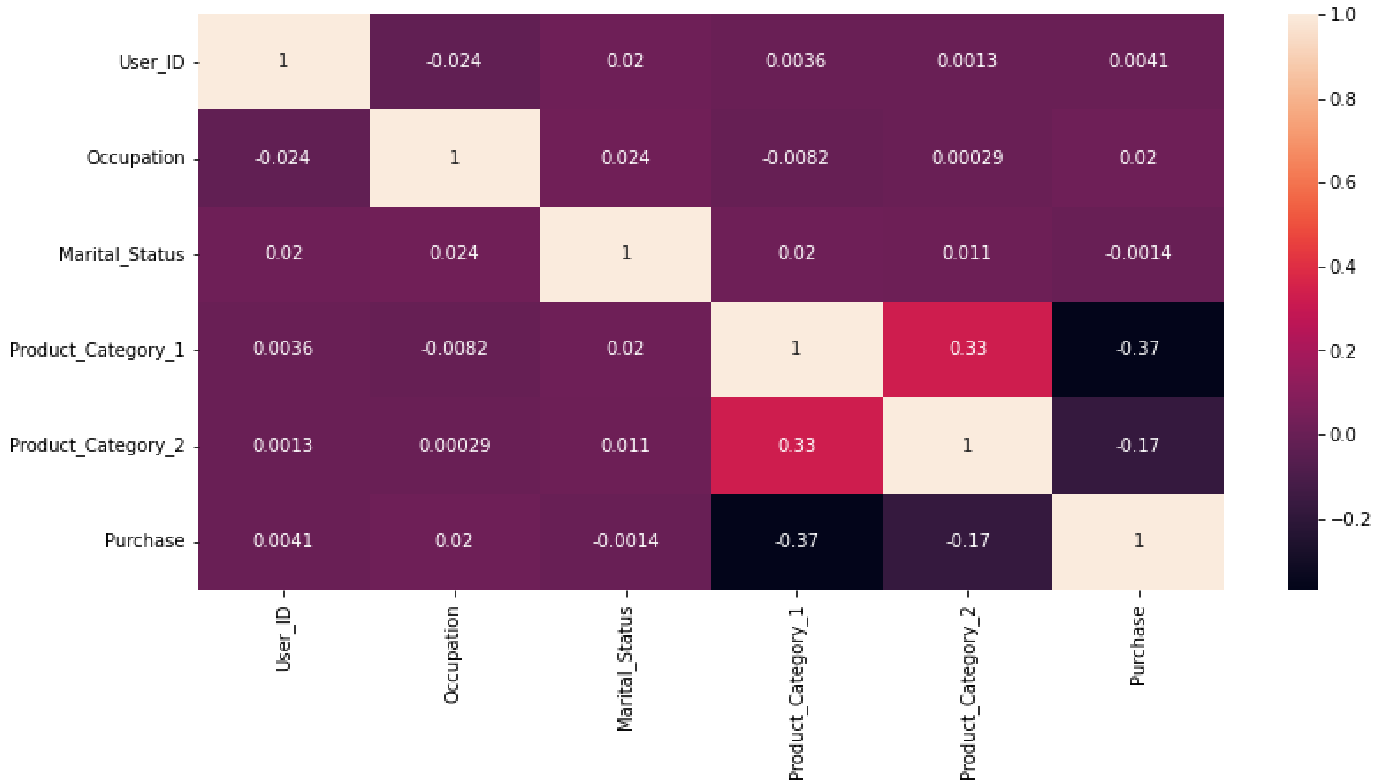


In [30]:
```
# Droping columns
```

In [31]:
```
df['Product_Category_2'] = df['Product_Category_2'].fillna((df['Product_Category_2']).me
df.drop(['Product_Category_3'], axis=1, inplace=True)
```

Corealtion Matrix

In [32]:
```
corr = df.corr()
plt.figure(figsize=(13,6))
sns.heatmap(corr, annot=True);
```



In [33]:
```
# df_Gender = pd.get_dummies(train['Gender'])
# df_Age = pd.get_dummies(train['Age'])
# df_City_Category = pd.get_dummies(train['City_Category'])
# df_Stay_In_Current_City_Years = pd.get_dummies(train['Stay_In_Current_City_Years'])

# data_final= pd.concat([train, df_Gender, df_Age, df_City_Category, df_Stay_In_Current_

# data_final.head()
```

In [34]:
```
# from sklearn.preprocessing import LabelEncoder
# LE= LabelEncoder()
```

In [35]:
```
# df['Gender'] = LE.fit_transform(df['Gender'])
# df['Age'] = LE.fit_transform(df['Age'])
# df['City_Category'] = LE.fit_transform(df['City_Category'])
# df['Stay_In_Current_City_Years'] = LE.fit_transform(df['Stay_In_Current_City_Years'])
```

In [36]:
```
df= pd.get_dummies(df, columns = ['Gender', 'Age','City_Category','Stay_In_Current_City_
```

In [37]:
```
df.head()
```

Out[37]:

	User_ID	Product_ID	Marital_Status	Purchase	Gender_F	Gender_M	Age_0-17	Age_18-25	Age_26-35	Age_36-45	...
0	1000001	P00069042	0	8370	1	0	1	0	0	0	...
1	1000001	P00248942	0	15200	1	0	1	0	0	0	...
2	1000001	P00087842	0	1422	1	0	1	0	0	0	...
3	1000001	P00085442	0	1057	1	0	1	0	0	0	...
4	1000002	P00285442	0	7969	0	1	0	0	0	0	...

5 rows × 79 columns

Input Split

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

Out[38]:

	User_ID	Product_ID	Marital_Status	Purchase	Gender_F	Gender_M	Age_0-17	Age_18-25	Age_26-35	Age_36-45	...
0	1000001	P00069042	0	8370	1	0	1	0	0	0	...
1	1000001	P00248942	0	15200	1	0	1	0	0	0	...
2	1000001	P00087842	0	1422	1	0	1	0	0	0	...
3	1000001	P00085442	0	1057	1	0	1	0	0	0	...
4	1000002	P00285442	0	7969	0	1	0	0	0	0	...

5 rows × 79 columns

In [39]:
```
X = df.drop(columns=['User_ID', 'Product_ID', 'Purchase'])
y = df['Purchase']
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 547391 entries, 0 to 550067
Data columns (total 76 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   Marital_Status              547391 non-null  int64
 1   Gender_F                    547391 non-null  uint8
 2   Gender_M                    547391 non-null  uint8
 3   Age_0-17                    547391 non-null  uint8
 4   Age_18-25                   547391 non-null  uint8
 5   Age_26-35                   547391 non-null  uint8
```

```
 6   Age_36-45                      547391 non-null  uint8
 7   Age_46-50                      547391 non-null  uint8
 8   Age_51-55                      547391 non-null  uint8
 9   Age_55+                        547391 non-null  uint8
 10  City_Category_A                547391 non-null  uint8
 11  City_Category_B                547391 non-null  uint8
 12  City_Category_C                547391 non-null  uint8
 13  Stay_In_Current_City_Years_0   547391 non-null  uint8
 14  Stay_In_Current_City_Years_1   547391 non-null  uint8
 15  Stay_In_Current_City_Years_2   547391 non-null  uint8
 16  Stay_In_Current_City_Years_3   547391 non-null  uint8
 17  Stay_In_Current_City_Years_4+  547391 non-null  uint8
 18  Occupation_0                   547391 non-null  uint8
 19  Occupation_1                   547391 non-null  uint8
 20  Occupation_2                   547391 non-null  uint8
 21  Occupation_3                   547391 non-null  uint8
 22  Occupation_4                   547391 non-null  uint8
 23  Occupation_5                   547391 non-null  uint8
 24  Occupation_6                   547391 non-null  uint8
 25  Occupation_7                   547391 non-null  uint8
 26  Occupation_8                   547391 non-null  uint8
 27  Occupation_9                   547391 non-null  uint8
 28  Occupation_10                  547391 non-null  uint8
 29  Occupation_11                  547391 non-null  uint8
 30  Occupation_12                  547391 non-null  uint8
 31  Occupation_13                  547391 non-null  uint8
 32  Occupation_14                  547391 non-null  uint8
 33  Occupation_15                  547391 non-null  uint8
 34  Occupation_16                  547391 non-null  uint8
 35  Occupation_17                  547391 non-null  uint8
 36  Occupation_18                  547391 non-null  uint8
 37  Occupation_19                  547391 non-null  uint8
 38  Occupation_20                  547391 non-null  uint8
 39  Product_Category_1_1           547391 non-null  uint8
 40  Product_Category_1_2           547391 non-null  uint8
 41  Product_Category_1_3           547391 non-null  uint8
 42  Product_Category_1_4           547391 non-null  uint8
 43  Product_Category_1_5           547391 non-null  uint8
 44  Product_Category_1_6           547391 non-null  uint8
 45  Product_Category_1_7           547391 non-null  uint8
 46  Product_Category_1_8           547391 non-null  uint8
 47  Product_Category_1_9           547391 non-null  uint8
 48  Product_Category_1_10          547391 non-null  uint8
 49  Product_Category_1_11          547391 non-null  uint8
 50  Product_Category_1_12          547391 non-null  uint8
 51  Product_Category_1_13          547391 non-null  uint8
 52  Product_Category_1_14          547391 non-null  uint8
 53  Product_Category_1_15          547391 non-null  uint8
 54  Product_Category_1_16          547391 non-null  uint8
 55  Product_Category_1_17          547391 non-null  uint8
 56  Product_Category_1_18          547391 non-null  uint8
 57  Product_Category_1_19          547391 non-null  uint8
 58  Product_Category_1_20          547391 non-null  uint8
 59  Product_Category_2_2           547391 non-null  uint8
 60  Product_Category_2_3           547391 non-null  uint8
 61  Product_Category_2_4           547391 non-null  uint8
 62  Product_Category_2_5           547391 non-null  uint8
 63  Product_Category_2_6           547391 non-null  uint8
 64  Product_Category_2_7           547391 non-null  uint8
 65  Product_Category_2_8           547391 non-null  uint8
 66  Product_Category_2_9           547391 non-null  uint8
 67  Product_Category_2_10          547391 non-null  uint8
 68  Product_Category_2_11          547391 non-null  uint8
 69  Product_Category_2_12          547391 non-null  uint8
 70  Product_Category_2_13          547391 non-null  uint8
 71  Product_Category_2_14          547391 non-null  uint8
```

```
 72  Product_Category_2_15          547391 non-null  uint8
 73  Product_Category_2_16          547391 non-null  uint8
 74  Product_Category_2_17          547391 non-null  uint8
 75  Product_Category_2_18          547391 non-null  uint8
dtypes: int64(1), uint8(75)
memory usage: 47.5 MB
```

In [40]:
```
X.head()
```

Out[40]:

	Marital_Status	Gender_F	Gender_M	Age_0-17	Age_18-25	Age_26-35	Age_36-45	Age_46-50	Age_51-55	Age_55+	...	Pro
0	0	1	0	1	0	0	0	0	0	0	...	
1	0	1	0	1	0	0	0	0	0	0	...	
2	0	1	0	1	0	0	0	0	0	0	...	
3	0	1	0	1	0	0	0	0	0	0	...	
4	0	0	1	0	0	0	0	0	0	1	...	

5 rows × 76 columns

In [ ]:

In [41]:
```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.30,random_state=42)
```

In [42]:
```
print(X.shape)
print(y.shape)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(547391, 76)
(547391,)
(383173, 76)
(383173,)
(164218, 76)
(164218,)
```

In [43]:
```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
X_train = ss.fit_transform(X_train)
X_test = ss.transform(X_test)
```

In [44]:
```
print(X.shape)
print(y.shape)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(547391, 76)
(547391,)
(383173, 76)
(383173,)
(164218, 76)
(164218,)
```

In [45]:
```
# Training Model selection
```

In [46]:
```
from sklearn.linear_model import LinearRegression
LR=LinearRegression()
```

In [47]:
```
LR.fit(X_train,y_train)
```

Out[47]:
```
LinearRegression()
```

In [48]:
```
y_pred=LR.predict(X_test)
```

In [49]:
```
y_pred
```

Out[49]:
```
array([11073.22265906, 13913.72265906,  7403.22265906, ...,
        7768.22265906,  5842.22265906,  7215.22265906])
```

In [50]:
```
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error
```

In [51]:
```
R2_score = r2_score(y_test,y_pred)
```

In [52]:
```
print("training score = ",LR.score(X_train, y_train))
print("Testing score = ",LR.score(X_test, y_test))
print("Mean Absolute error =",mean_absolute_error(y_test,y_pred))
print("Mean Squared error =",mean_squared_error(y_test,y_pred))
print("Root Mean Squared error=",np.sqrt(mean_squared_error(y_test,y_pred)))
print("R2score = " ,R2_score)
```

```
training score =  0.6347153750403496
Testing score =  0.6350016170765882
Mean Absolute error = 2260.2782310954826
Mean Squared error = 8934811.918436665
Root Mean Squared error= 2989.1155746201357
R2score =  0.6350016170765882
```

In [53]:
```
from sklearn.linear_model import Lasso
lasso = Lasso(alpha =0.0001)
lasso.fit(X_train, y_train)
y_pred = lasso.predict(X_test)
```

In [54]:
```
print("training score =", lasso.score(X_train, y_train))
print("Testing score =", lasso.score(X_test, y_test))
print("Mean Absolute error =",mean_absolute_error(y_test,y_pred))
print("Mean Squared error =",mean_squared_error(y_test,y_pred))
print("Root Mean Squared error=",np.sqrt(mean_squared_error(y_test,y_pred)))
print("R2score = " ,R2_score)
```

```
training score = 0.6347213760940762
Testing score = 0.6350018580884018
Mean Absolute error = 2260.399604181418
Mean Squared error = 8934806.018697584
Root Mean Squared error= 2989.1145877496206
R2score =  0.6350016170765882
```

In [55]:
```
from sklearn.linear_model import Ridge
Ridge = Ridge(alpha = 0.01)
Ridge.fit(X_train,y_train)
y_pred = Ridge.predict(X_test)
```

In [56]:
```
print("training score =", Ridge.score(X_train, y_train))
print("Testing score =", Ridge.score(X_test, y_test))
print("Mean Absolute error =",mean_absolute_error(y_test,y_pred))
print("Mean Squared error =",mean_squared_error(y_test,y_pred))
print("Root Mean Squared error=",np.sqrt(mean_squared_error(y_test,y_pred)))
print("R2score = " ,R2_score)
```

```
training score = 0.6347213760941086
Testing score = 0.635001857336509
Mean Absolute error = 2260.3995908922634
Mean Squared error = 8934806.037103202
Root Mean Squared error= 2989.1145908283947
R2score =  0.6350016170765882
```

In [57]:

```
from sklearn.ensemble import RandomForestRegressor
rf=RandomForestRegressor()
rf.fit(X_train,y_train)
```

Out[57]:

```
RandomForestRegressor()
```

In [58]:

```
print("training score =", rf.score(X_train, y_train))
print("Testing score =", rf.score(X_test, y_test))
print("Mean Absolute error =",mean_absolute_error(y_test,y_pred))
print("Mean Squared error =",mean_squared_error(y_test,y_pred))
print("Root Mean Squared error=",np.sqrt(mean_squared_error(y_test,y_pred)))
print("R2score = " ,R2_score)
```

```
training score = 0.7387019904768657
Testing score = 0.6314162482497752
Mean Absolute error = 2260.3995908922634
Mean Squared error = 8934806.037103202
Root Mean Squared error= 2989.1145908283947
R2score =  0.6350016170765882
```