Mini-Project

Black Friday: How much will a customer spend?

Introduction

A retail company “ABC Private Limited” wants to understand the customer purchase behaviour (specifically, purchase amount) against various products of different categories. They have shared purchase summary of various customers for a selected high volume products from last month.

They want to build a model to predict the purchase amount of customer against various products which will help them to create personalized offer for customers against different products.

Performance Measure

Usually for regression problems the typical performance measure is the Root Mean Square Error (RMSE). This function gives an idea of how much error the system makes in its predictions with higher weight for large errors.

Make Assumptions

Before even looking at the available data is good to make some assumption on the expected results. Therefore, let’s start to think about possible parameters that might influence the amount a client spends on Black Friday.

Available data

This is the current data tavailable:

In [1]:
```
# data.png
```

If we analyse it individually we see that we do not have any information regarding the stores. Moreover, there is some information related to the customer such as age group, sex, occupation and marital status. On the other hand, we have data on the city's size and how many years the customer has lived in it whereas on the product's side there is only information regarding the categories and the amount spent. It is my belief that Gender , Age , City_Category , Product_Category_1 are the predictors that will influence more the amount spent by a customer on this day.

The target variable is Purchase .

In [2]:
```
# target.png
```

Take a quick look at the Data Structure

Let's start by importing some libraries and our data.

In [3]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [4]:
```
train = pd.read_csv("train.csv")
test = pd.read_csv("test-comb.csv")
train.head()
```

Out[4]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_
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+

In [5]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 12 columns):
User_ID                       550068 non-null int64
Product_ID                    550068 non-null object
Gender                        550068 non-null object
Age                           550068 non-null object
Occupation                    550068 non-null int64
City_Category                 550068 non-null object
Stay_In_Current_City_Years    550068 non-null object
Marital_Status                550068 non-null int64
Product_Category_1            550068 non-null int64
Product_Category_2            376430 non-null float64
Product_Category_3            166821 non-null float64
Purchase                      550068 non-null int64
dtypes: float64(2), int64(5), object(5)
memory usage: 50.4+ MB
```

In [6]:
```
train.describe()
```

Out[6]:

	User_ID	Occupation	Marital_Status	Product_Category_1	Product_Cate
count	5.500680e+05	550068.000000	550068.000000	550068.000000	376430.00000
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

Organisation of our analysis

Our goal as a Data Scientist is to identify the most important variables and to define the best regression model for predicting out target variable. Hence, this analysis will be divided into five stages:

1. Exploratory data analysis (EDA);

2. Data Pre-processing;

3. Feature engineering;

4. Feature Transformation;

5. Modeling;

The following is a workflow chart illustrating the five stages:

In [7]:
```
#workflow.png
```

1. Exploratory Data Analysis (EDA)

We've made our first assumptions on the data and now we are ready to perform some basic data exploration and come up with some inference. Hence, the goal for this section is to take a glimpse on the data as well as any irregularities so that we can correct on the next section, Data Pre-Processing.

1.1. Univariate Analysis

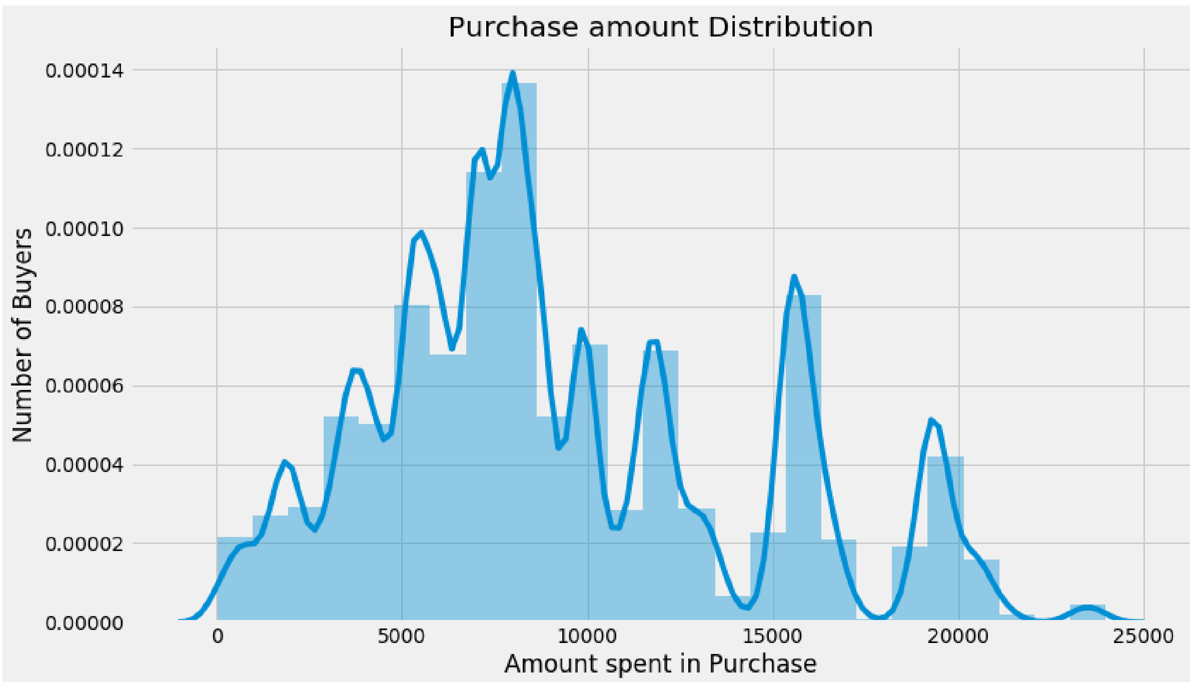
To get an idea of the distribution of numerical variables, histograms are an excellent starting point. Let's begin by generating one for Purchase, our target variable.

1.1.1. Distribution of the target variable: Purchase

In [8]:
```
plt.style.use('fivethirtyeight')
plt.figure(figsize=(12,7))
sns.distplot(train.Purchase, bins = 25)
plt.xlabel("Amount spent in Purchase")
plt.ylabel("Number of Buyers")
plt.title("Purchase amount Distribution")
```

```
C:\Users\Mayank\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: Future
Warning: Using a non-tuple sequence for multidimensional indexing is deprecat
ed; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be i
nterpreted as an array index, `arr[np.array(seq)]`, which will result either
in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[8]:
```
Text(0.5, 1.0, 'Purchase amount Distribution')
```



In [9]:
```
print ("Skew is:", train.Purchase.skew())
print("Kurtosis: %f" % train.Purchase.kurt())
```

```
Skew is: 0.6001400037087128
Kurtosis: -0.338378
```

1.1.2. Numerical Variables

In [10]:
```
numeric_features = train.select_dtypes(include=[np.number])
numeric_features.dtypes
```

Out[10]:
```
User_ID                 int64
Occupation              int64
Marital_Status          int64
Product_Category_1      int64
Product_Category_2    float64
Product_Category_3    float64
Purchase                int64
dtype: object
```

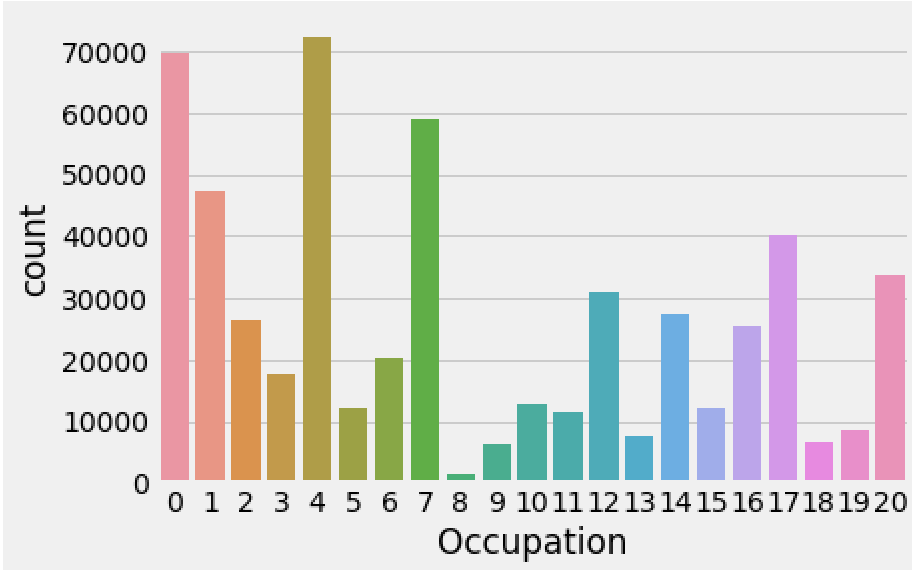
1.1.2.1. Distribution of the Occupation variable

In [11]:
```
train.Occupation.value_counts()
```

Out[11]:
```
4     72308
0     69638
7     59133
1     47426
17    40043
20    33562
12    31179
14    27309
2     26588
16    25371
6     20355
3     17650
10    12930
5     12177
15    12165
11    11586
19     8461
13     7728
18     6622
9      6291
8      1546
Name: Occupation, dtype: int64
```

In [12]:
```
sns.countplot(train.Occupation)
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x1dfedfb14e0>



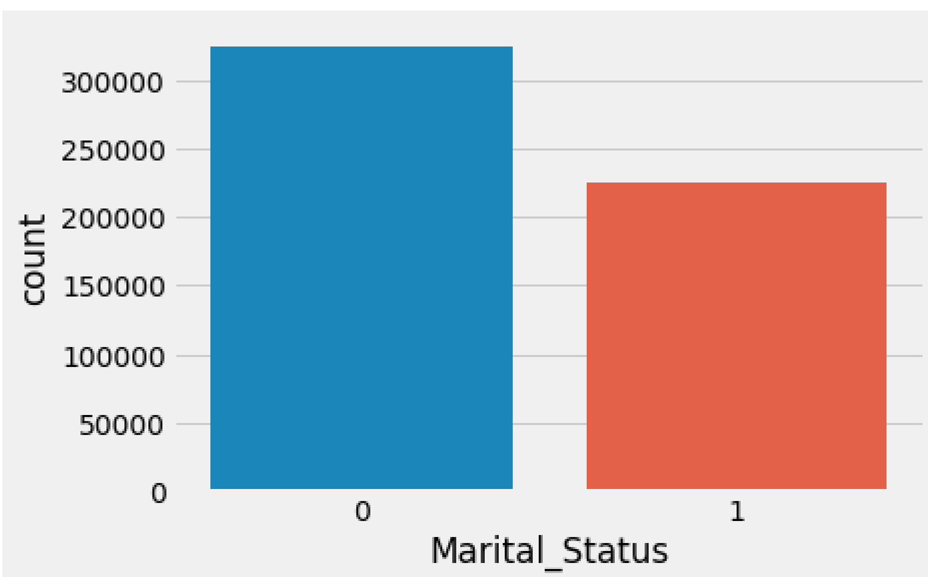
1.1.2.2. Distribution of the Marital_Status variable

In [13]:
```
train.Marital_Status.value_counts()
```

Out[13]:
```
0    324731
1    225337
Name: Marital_Status, dtype: int64
```

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

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x1dfedad3400>



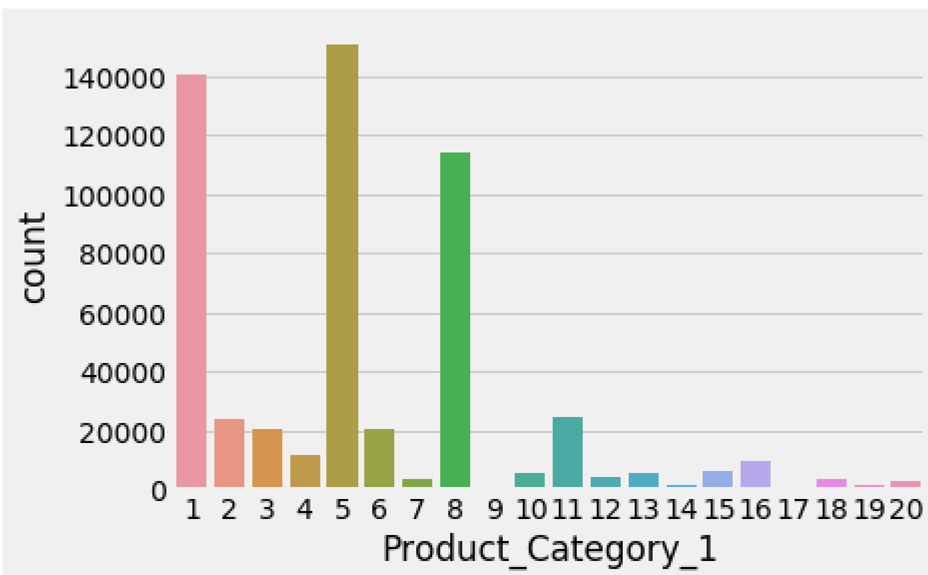
1.1.2.3. Distribution of the Product_Category_1 variable

In [15]:
```
train.Product_Category_1.value_counts()
```

Out[15]:
```
5     150933
1     140378
8     113925
11     24287
2      23864
6      20466
3      20213
4      11753
16      9828
15      6290
13      5549
10      5125
12      3947
7       3721
18      3125
20      2550
19      1603
14      1523
17       578
9        410
Name: Product_Category_1, dtype: int64
```

In [16]:
```
sns.countplot(train.Product_Category_1)
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1dfedbd3400>
```



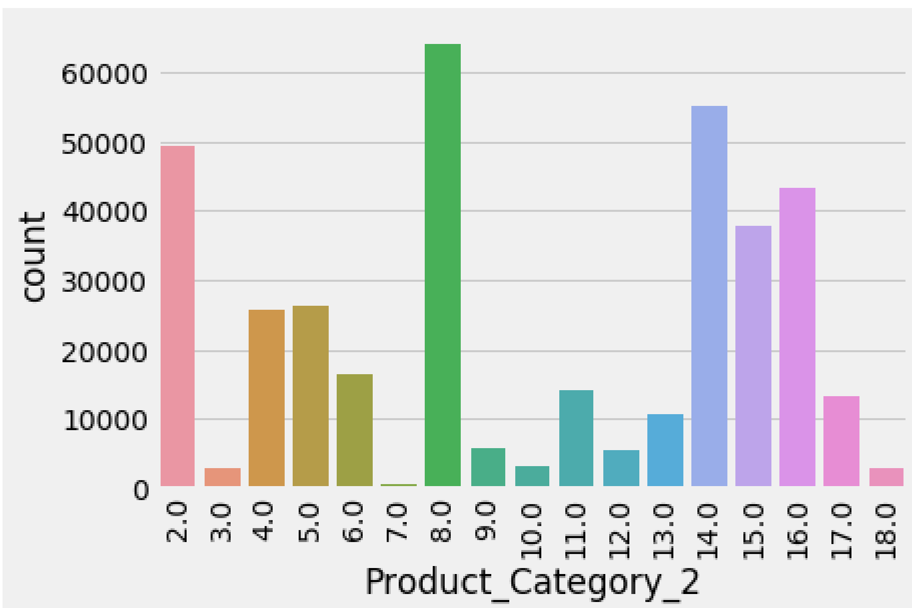
1.1.2.4. Distribution of the Product_Category_2 variable

In [17]:
```
train.Product_Category_2.value_counts()
```

Out[17]:
```
8.0     64088
14.0    55108
2.0     49217
16.0    43255
15.0    37855
5.0     26235
4.0     25677
6.0     16466
11.0    14134
17.0    13320
13.0    10531
9.0      5693
12.0     5528
10.0     3043
3.0      2884
18.0     2770
7.0       626
Name: Product_Category_2, dtype: int64
```

In [18]:
```
sns.countplot(train.Product_Category_2)
plt.xticks(rotation=90)
```

Out[18]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16]),
 <a list of 17 Text xticklabel objects>)
```



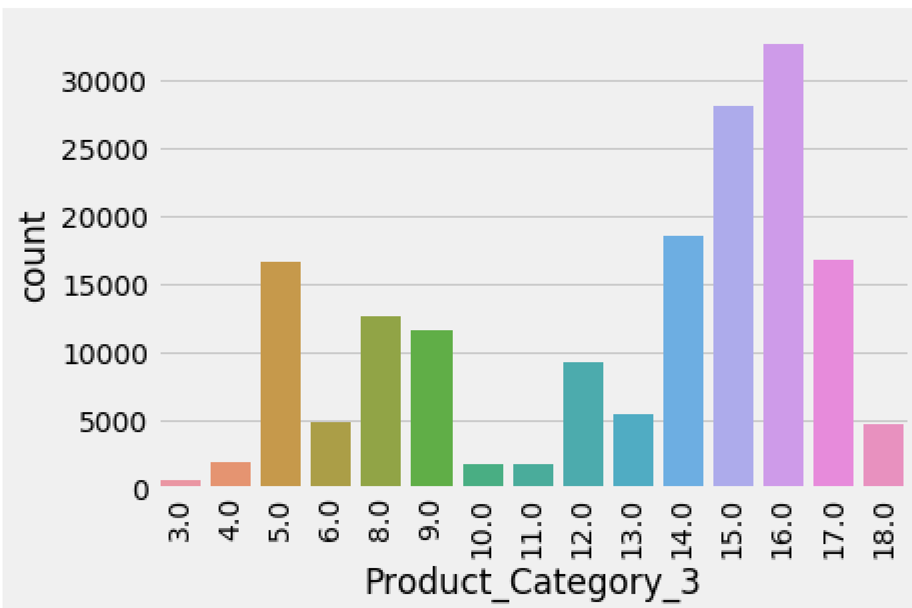
1.1.2.5. Distribution of the Product_Category_3 variable

In [19]:
```
train.Product_Category_3.value_counts()
```

Out[19]:
```
16.0    32636
15.0    28013
14.0    18428
17.0    16702
5.0     16658
8.0     12562
9.0     11579
12.0     9246
13.0     5459
6.0      4890
18.0     4629
4.0      1875
11.0     1805
10.0     1726
3.0       613
Name: Product_Category_3, dtype: int64
```

In [20]:
```
sns.countplot(train.Product_Category_3)
plt.xticks(rotation=90)
```

Out[20]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14]),
 <a list of 15 Text xticklabel objects>)
```



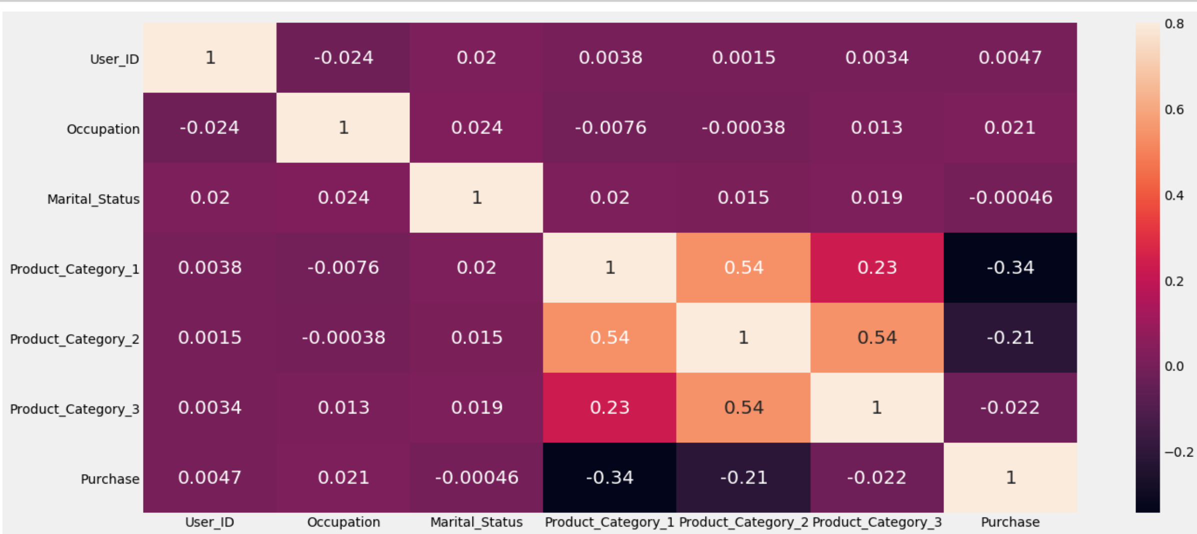
1.1.2.6. Correlation between Numerical Predictors and Target variable

In [21]:
```
corr = numeric_features.corr()

print (corr['Purchase'].sort_values(ascending=False)[:10], '\n')
print (corr['Purchase'].sort_values(ascending=False)[-10:])
```

```
Purchase              1.000000
Occupation            0.020833
User_ID               0.004716
Marital_Status       -0.000463
Product_Category_3   -0.022006
Product_Category_2   -0.209918
Product_Category_1   -0.343703
Name: Purchase, dtype: float64

Purchase              1.000000
Occupation            0.020833
User_ID               0.004716
Marital_Status       -0.000463
Product_Category_3   -0.022006
Product_Category_2   -0.209918
Product_Category_1   -0.343703
Name: Purchase, dtype: float64
```

In [22]:
```
#correlation matrix
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corr, vmax=.9, square=True);
```

In [23]:
```
#correlation matrix
f, ax = plt.subplots(figsize=(20, 9))
sns.heatmap(corr, vmax=.8,annot_kws={'size': 20}, annot=True);
```



In [24]:
```
#Correlations Between Attributes
#Pearson’s Correlation
#Coefficient, that assumes a normal distribution of the attributes involved"""

s = corr.unstack()
#s.sort_values(kind="quicksort")
s
```

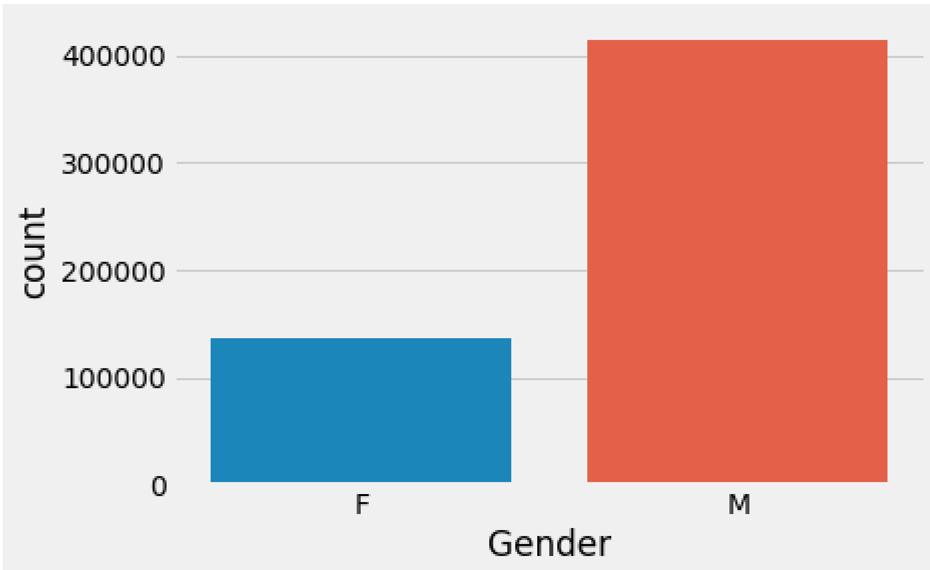
```
Out[24]: User_ID            User_ID               1.000000
                            Occupation           -0.023971
                            Marital_Status        0.020443
                            Product_Category_1    0.003825
                            Product_Category_2    0.001529
                            Product_Category_3    0.003419
                            Purchase              0.004716
         Occupation         User_ID              -0.023971
                            Occupation            1.000000
                            Marital_Status        0.024280
                            Product_Category_1   -0.007618
                            Product_Category_2   -0.000384
                            Product_Category_3    0.013263
                            Purchase              0.020833
         Marital_Status     User_ID               0.020443
                            Occupation            0.024280
                            Marital_Status        1.000000
                            Product_Category_1    0.019888
                            Product_Category_2    0.015138
                            Product_Category_3    0.019473
                            Purchase             -0.000463
         Product_Category_1 User_ID               0.003825
                            Occupation           -0.007618
                            Marital_Status        0.019888
                            Product_Category_1    1.000000
                            Product_Category_2    0.540583
                            Product_Category_3    0.229678
                            Purchase             -0.343703
         Product_Category_2 User_ID               0.001529
                            Occupation           -0.000384
                            Marital_Status        0.015138
                            Product_Category_1    0.540583
                            Product_Category_2    1.000000
                            Product_Category_3    0.543649
                            Purchase             -0.209918
         Product_Category_3 User_ID               0.003419
                            Occupation            0.013263
                            Marital_Status        0.019473
                            Product_Category_1    0.229678
                            Product_Category_2    0.543649
                            Product_Category_3    1.000000
                            Purchase             -0.022006
         Purchase           User_ID               0.004716
                            Occupation            0.020833
                            Marital_Status       -0.000463
                            Product_Category_1   -0.343703
                            Product_Category_2   -0.209918
                            Product_Category_3   -0.022006
                            Purchase              1.000000
         dtype: float64
```

1.1.3. Categorical Variables

1.1.3.1. Distribution of the variable Gender

In [25]:
```
sns.countplot(train.Gender)
```

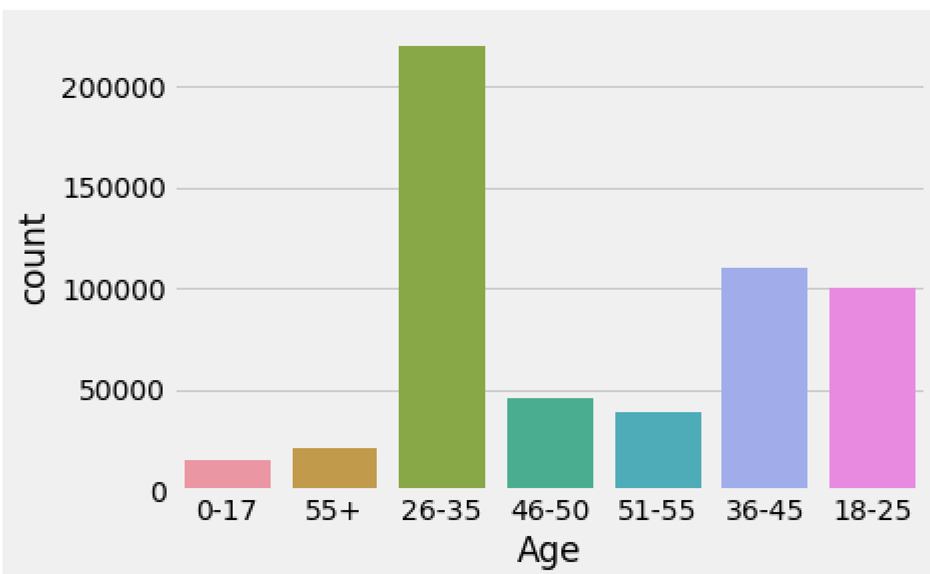
Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x1dfeef1bf60>



1.1.3.2. Distribution of the variable Age

In [26]:
```
sns.countplot(train.Age)
```

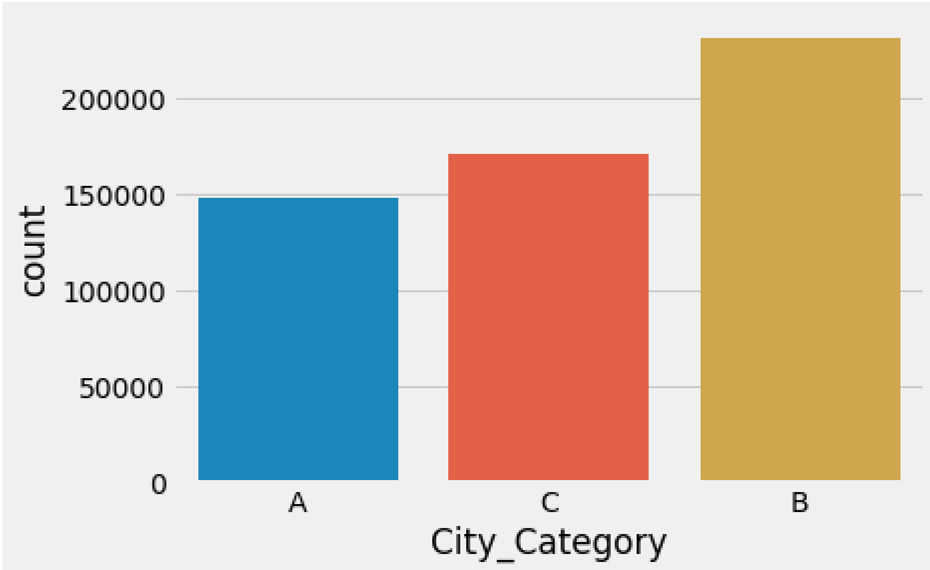
Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x1dfeef89c50>



1.1.3.3. Distribution of the variable City_Category

In [27]:
```
sns.countplot(train.City_Category)
```

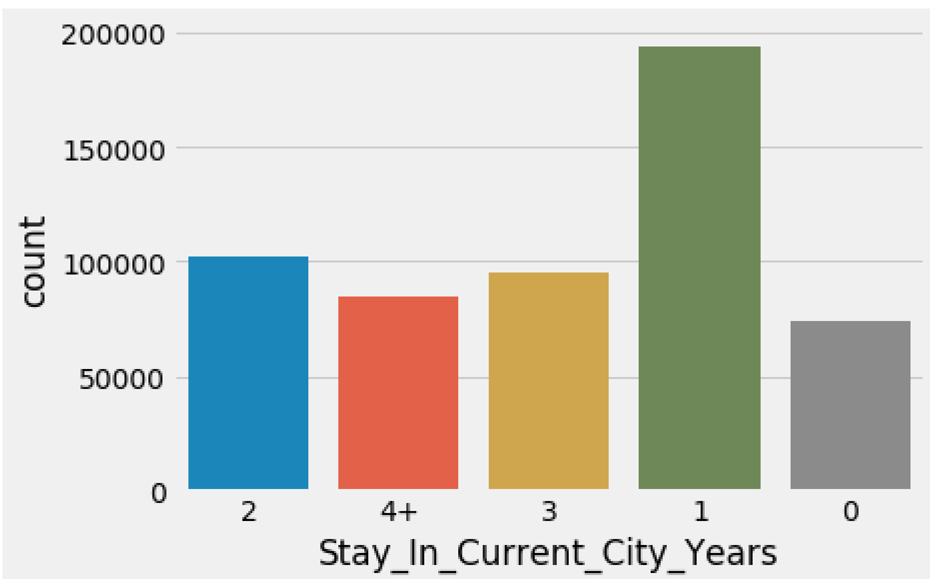
Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x1dfeefdeb00>



1.1.3.4. Distribution of the variable Stay_In_Current_City_Years

In [28]:
```
sns.countplot(train.Stay_In_Current_City_Years)
```

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x1dfef016860>

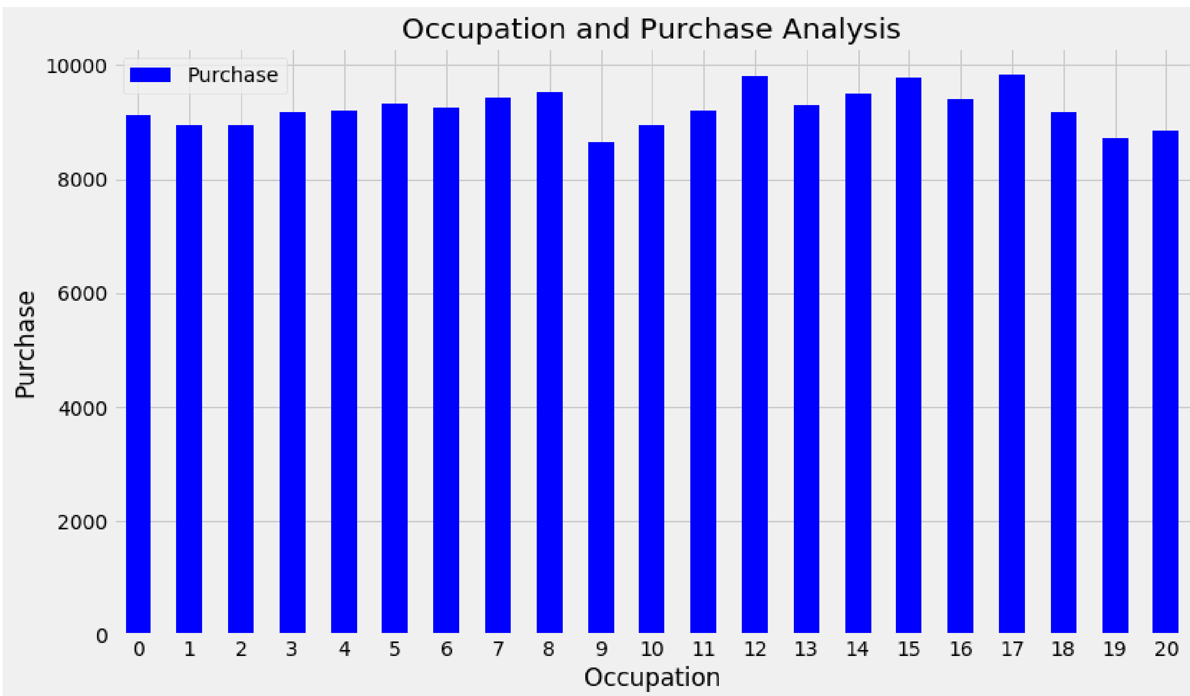


1.2. Bivariate Distribution

1.2.1. Numerical Variables

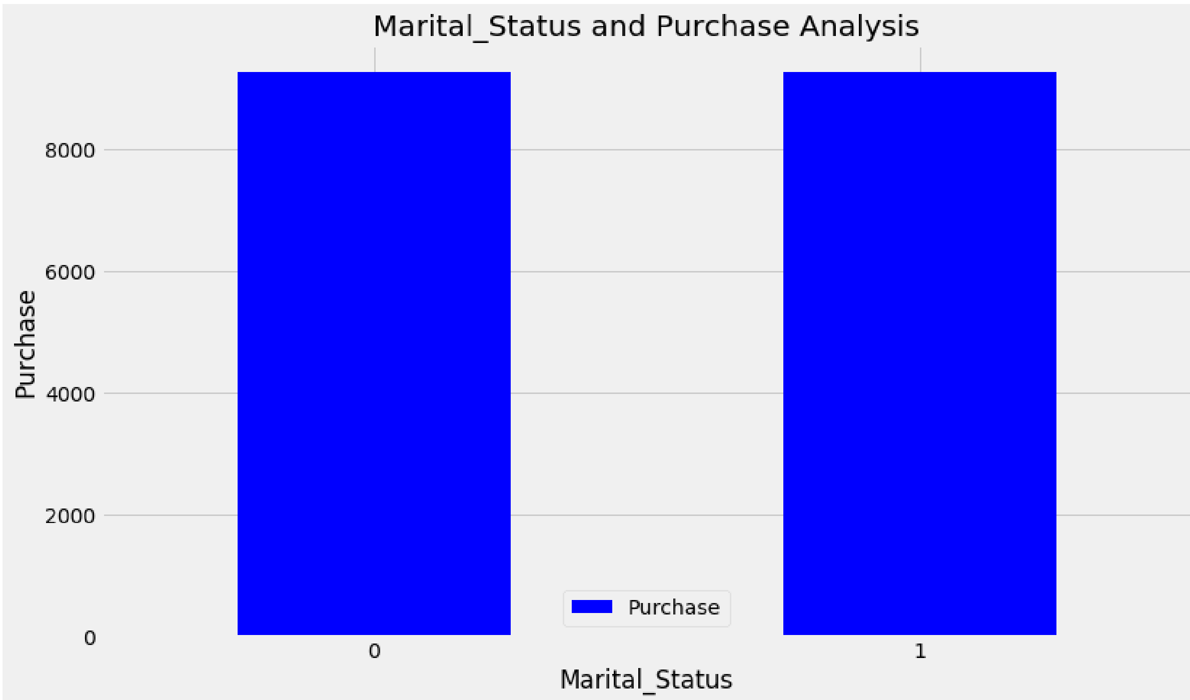
1.2.1.1. Occupation and Purchase Analysis

In [29]:
```
Occupation_pivot = \
train.pivot_table(index='Occupation', values="Purchase", aggfunc=np.mean)

Occupation_pivot.plot(kind='bar', color='blue',figsize=(12,7))
plt.xlabel("Occupation")
plt.ylabel("Purchase")
plt.title("Occupation and Purchase Analysis")
plt.xticks(rotation=0)
plt.show()
```



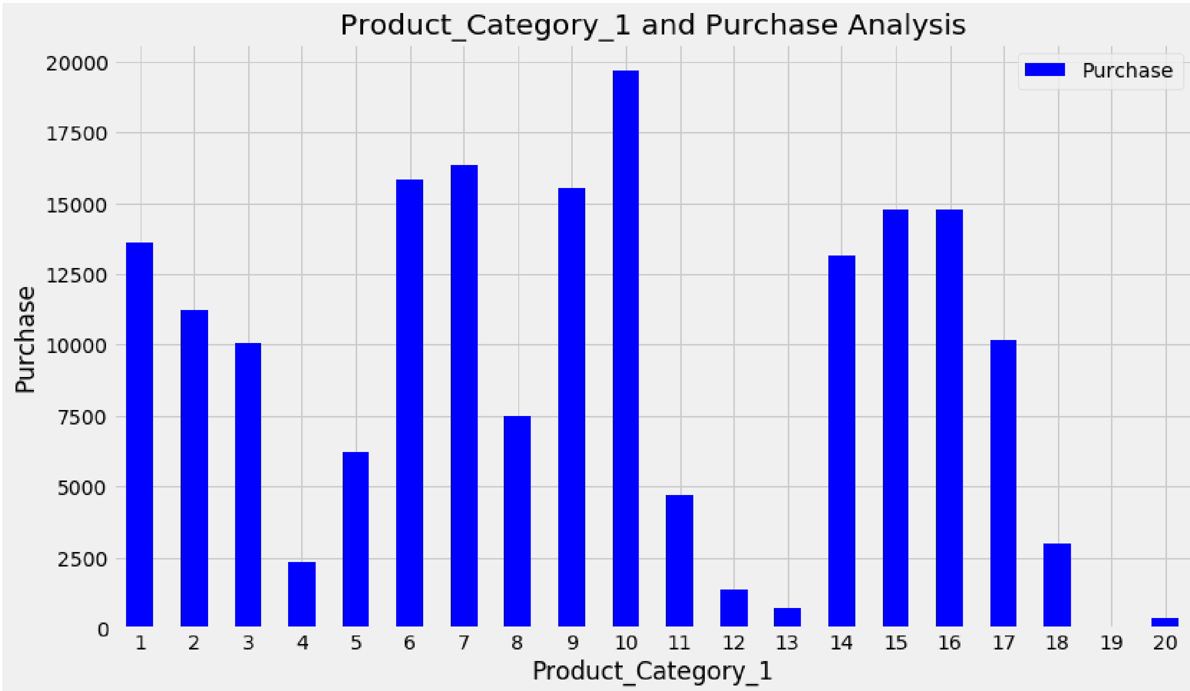
1.2.1.2. Marital_Status and Purchase Analysis

In [30]:
```
Marital_Status_pivot = \
train.pivot_table(index='Marital_Status', values="Purchase", aggfunc=np.mean)

Marital_Status_pivot.plot(kind='bar', color='blue',figsize=(12,7))
plt.xlabel("Marital_Status")
plt.ylabel("Purchase")
plt.title("Marital_Status and Purchase Analysis")
plt.xticks(rotation=0)
plt.show()
```

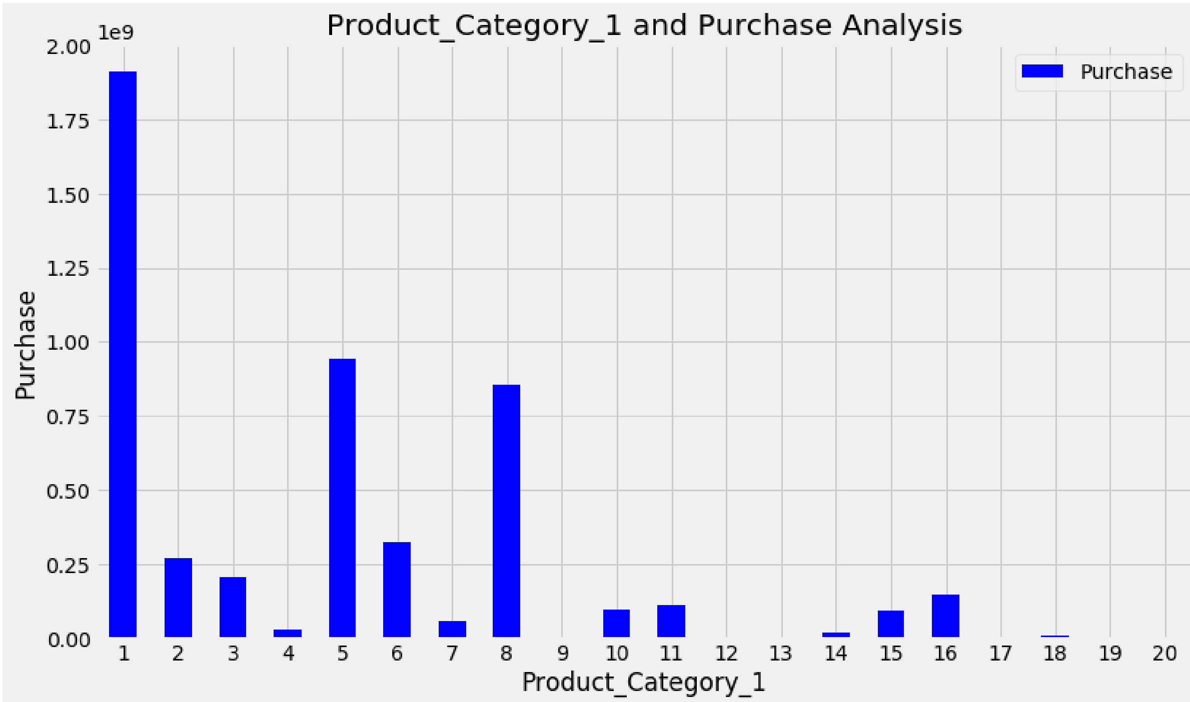


1.2.1.3. Product_Category_1 and Purchase Analysis

In [31]:
```
Product_category_1_pivot = \
train.pivot_table(index='Product_Category_1', values="Purchase", aggfunc=np.me
an)

Product_category_1_pivot.plot(kind='bar', color='blue',figsize=(12,7))
plt.xlabel("Product_Category_1")
plt.ylabel("Purchase")
plt.title("Product_Category_1 and Purchase Analysis")
plt.xticks(rotation=0)
plt.show()
```

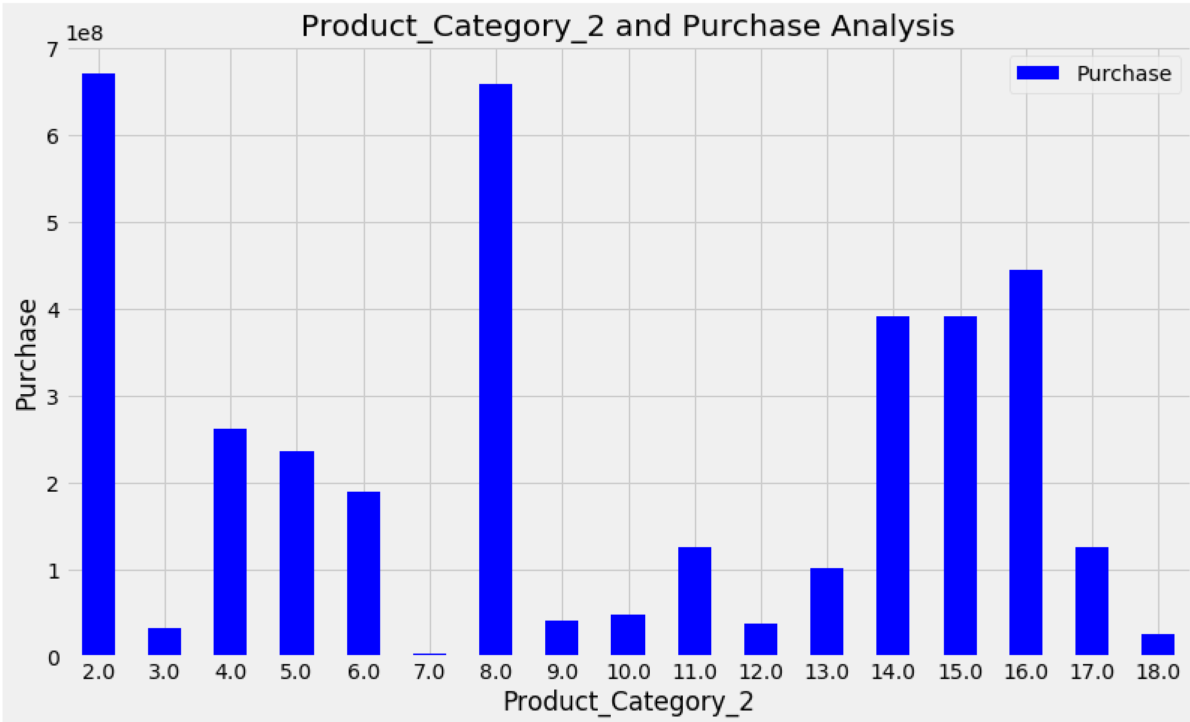


In [32]:
```
Product_category_1_pivot = \
train.pivot_table(index='Product_Category_1', values="Purchase", aggfunc=np.su
m)

Product_category_1_pivot.plot(kind='bar', color='blue',figsize=(12,7))
plt.xlabel("Product_Category_1")
plt.ylabel("Purchase")
plt.title("Product_Category_1 and Purchase Analysis")
plt.xticks(rotation=0)
plt.show()
```

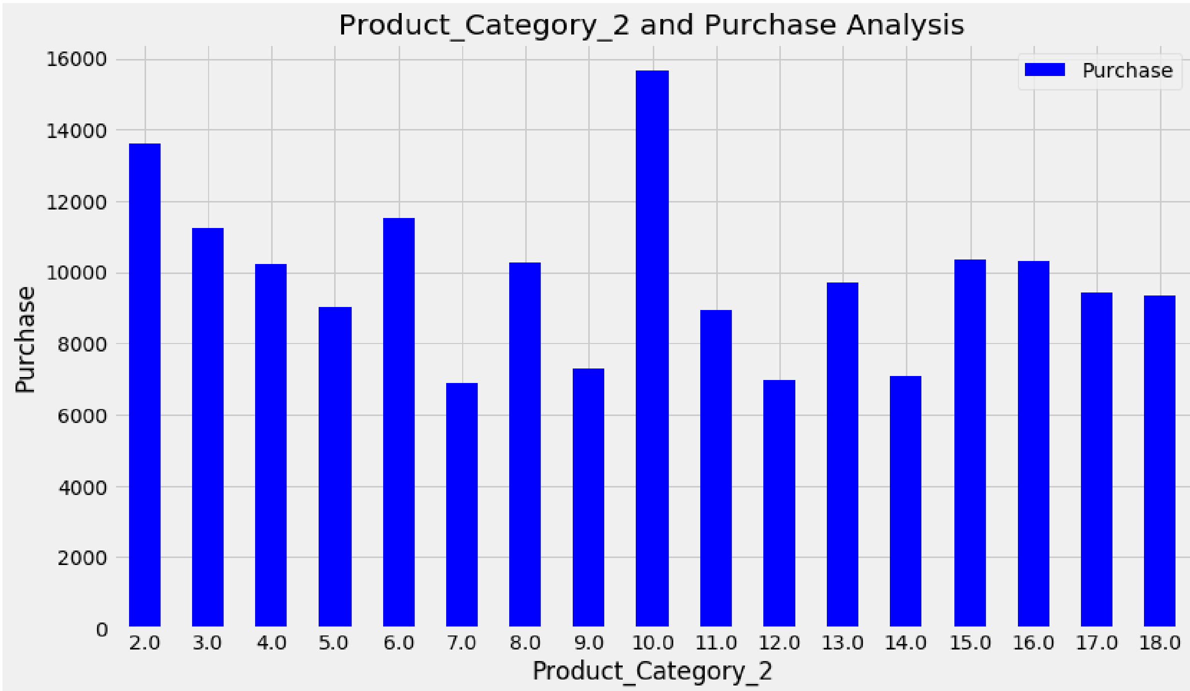


1.2.1.4. Product_Category_2 and Purchase Analysis

In [33]:
```
Product_category_1_pivot = \
train.pivot_table(index='Product_Category_2', values="Purchase", aggfunc=np.su
m)

Product_category_1_pivot.plot(kind='bar', color='blue',figsize=(12,7))
plt.xlabel("Product_Category_2")
plt.ylabel("Purchase")
plt.title("Product_Category_2 and Purchase Analysis")
plt.xticks(rotation=0)
plt.show()
```

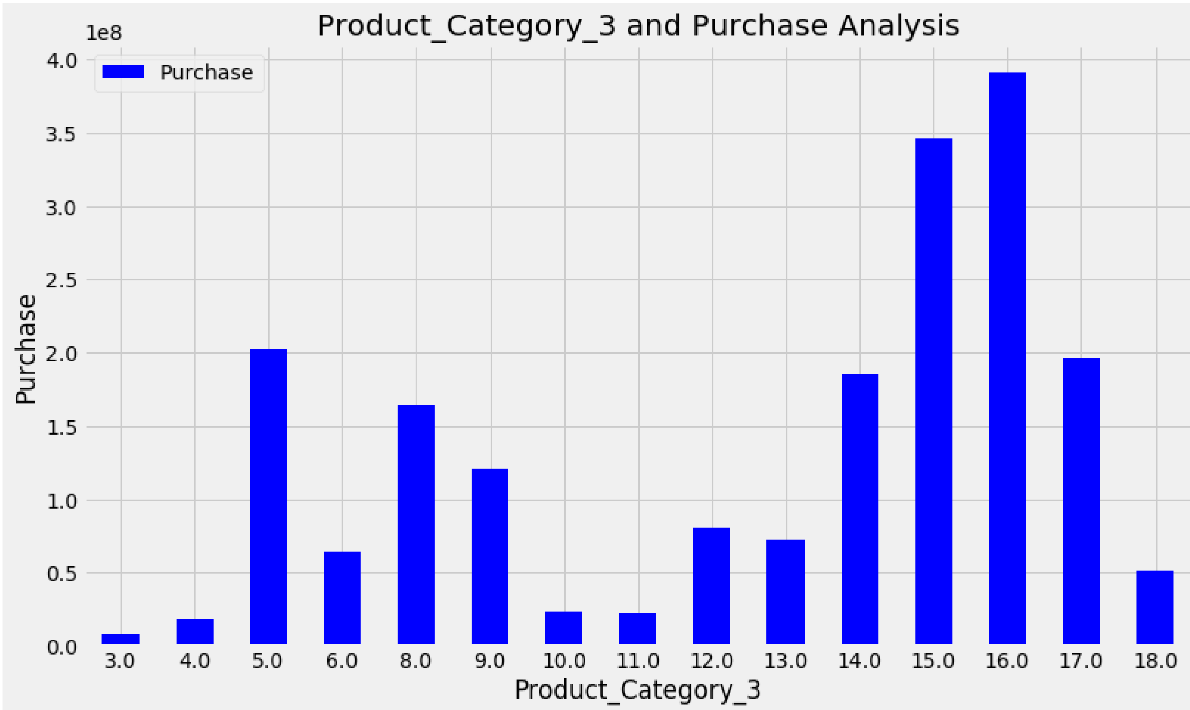


In [34]:
```
Product_category_1_pivot = \
train.pivot_table(index='Product_Category_2', values="Purchase", aggfunc=np.me
an)

Product_category_1_pivot.plot(kind='bar', color='blue',figsize=(12,7))
plt.xlabel("Product_Category_2")
plt.ylabel("Purchase")
plt.title("Product_Category_2 and Purchase Analysis")
plt.xticks(rotation=0)
plt.show()
```

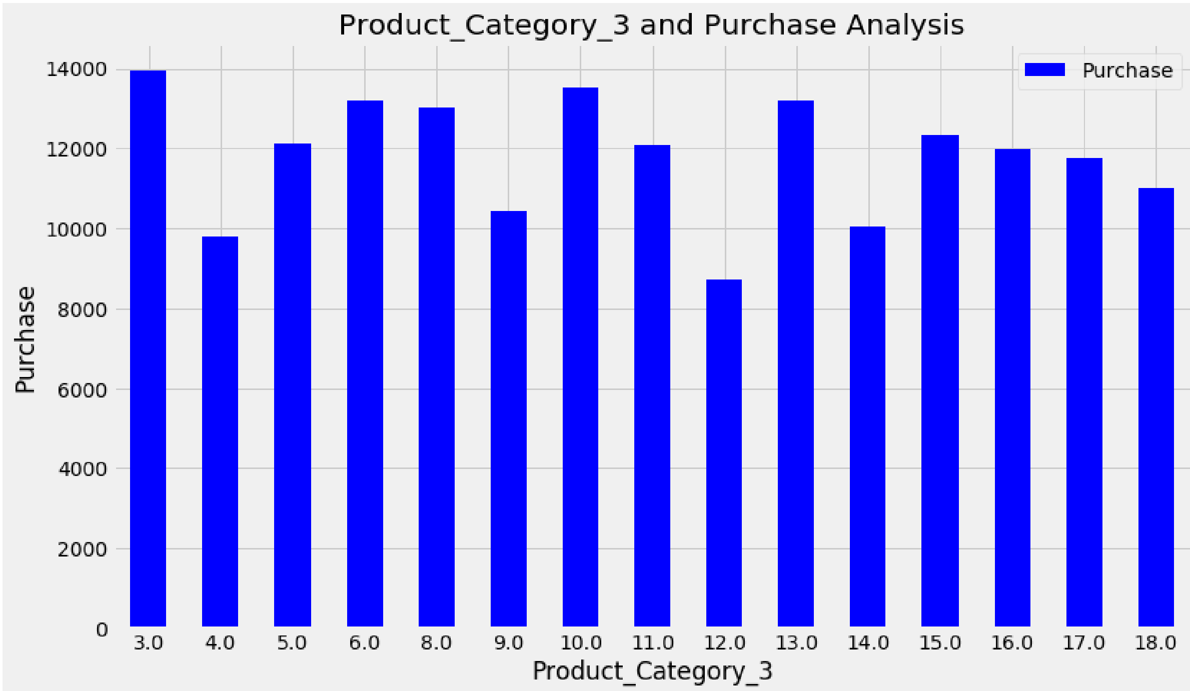


1.2.1.4. Product_Category_3 and Purchase Analysis

In [35]:
```
Product_category_1_pivot = \
train.pivot_table(index='Product_Category_3', values="Purchase", aggfunc=np.su
m)

Product_category_1_pivot.plot(kind='bar', color='blue',figsize=(12,7))
plt.xlabel("Product_Category_3")
plt.ylabel("Purchase")
plt.title("Product_Category_3 and Purchase Analysis")
plt.xticks(rotation=0)
plt.show()
```



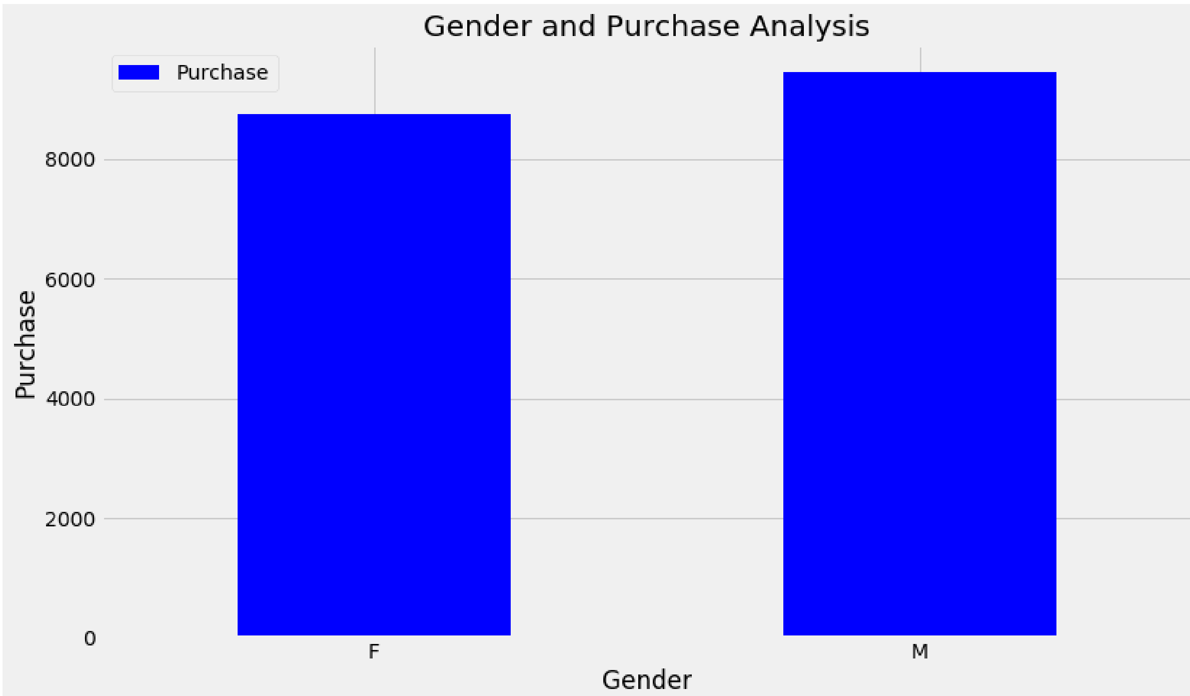
In [36]:
```
Product_category_1_pivot = \
train.pivot_table(index='Product_Category_3', values="Purchase", aggfunc=np.me
an)

Product_category_1_pivot.plot(kind='bar', color='blue',figsize=(12,7))
plt.xlabel("Product_Category_3")
plt.ylabel("Purchase")
plt.title("Product_Category_3 and Purchase Analysis")
plt.xticks(rotation=0)
plt.show()
```



1.2.2. Categorical Variables

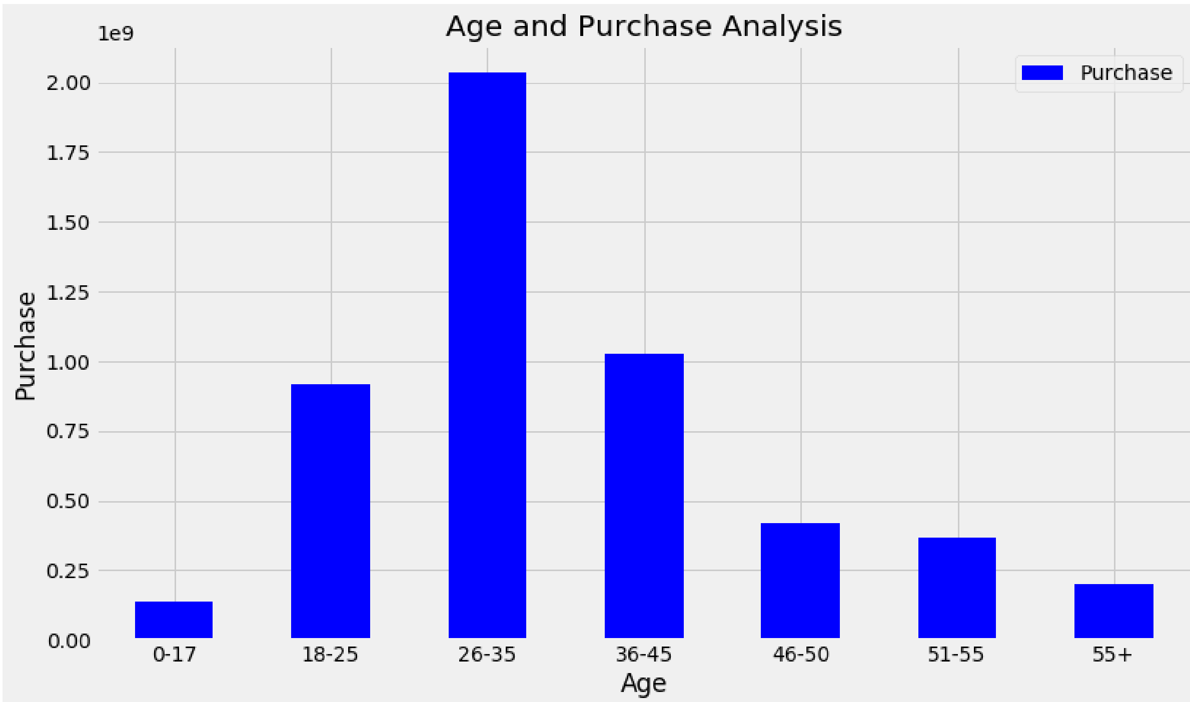
1.2.2.1. Gender and Purchase Analysis

In [37]:
```
Product_category_1_pivot = \
train.pivot_table(index='Gender', values="Purchase", aggfunc=np.mean)

Product_category_1_pivot.plot(kind='bar', color='blue',figsize=(12,7))
plt.xlabel("Gender")
plt.ylabel("Purchase")
plt.title("Gender and Purchase Analysis")
plt.xticks(rotation=0)
plt.show()
```

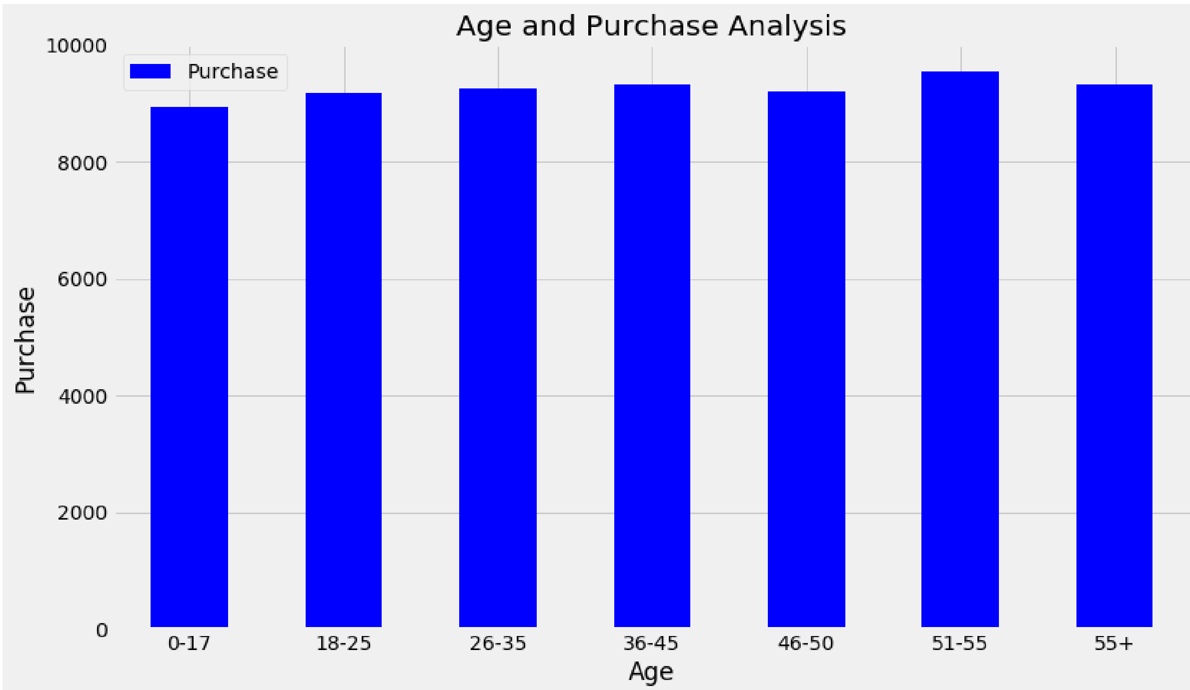


1.2.2.2. Age and Purchase Analysis

In [38]:
```
Product_category_1_pivot = \
train.pivot_table(index='Age', values="Purchase", aggfunc=np.sum)

Product_category_1_pivot.plot(kind='bar', color='blue',figsize=(12,7))
plt.xlabel("Age")
plt.ylabel("Purchase")
plt.title("Age and Purchase Analysis")
plt.xticks(rotation=0)
plt.show()
```

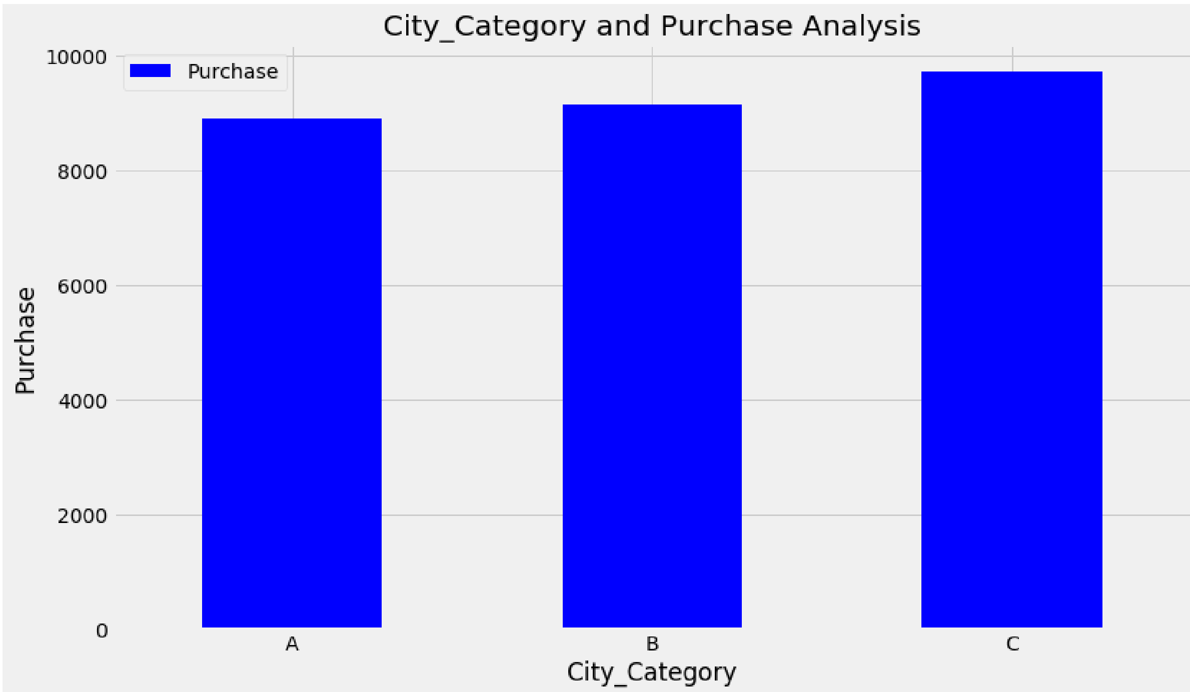


In [39]:
```
Product_category_1_pivot = \
train.pivot_table(index='Age', values="Purchase", aggfunc=np.mean)

Product_category_1_pivot.plot(kind='bar', color='blue',figsize=(12,7))
plt.xlabel("Age")
plt.ylabel("Purchase")
plt.title("Age and Purchase Analysis")
plt.xticks(rotation=0)
plt.show()
```



1.2.2.3. City_Category and Purchase Analysis

In [40]:
```
Product_category_1_pivot = \
train.pivot_table(index='City_Category', values="Purchase", aggfunc=np.mean)

Product_category_1_pivot.plot(kind='bar', color='blue',figsize=(12,7))
plt.xlabel("City_Category")
plt.ylabel("Purchase")
plt.title("City_Category and Purchase Analysis")
plt.xticks(rotation=0)
plt.show()
```



1.2.2.4. Stay_in_Current_City_Years and Purchase Analysis

In [41]:
```
Product_category_1_pivot = \
train.pivot_table(index='Stay_In_Current_City_Years', values="Purchase", aggfu
nc=np.mean)

Product_category_1_pivot.plot(kind='bar', color='blue',figsize=(12,7))
plt.xlabel("Stay_in_Current_City_Years")
plt.ylabel("Purchase")
plt.title("Stay_in_Current_City_Years and Purchase Analysis")
plt.xticks(rotation=0)
plt.show()
```

2. Data Preprocessing

In [42]:
```
train.isnull().sum()
```

Out[42]:
```
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 [43]:
```
test.isnull().sum()
```

Out[43]:
```
Unnamed: 0                         0
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             72344
Product_Category_3            162562
Comb                               0
dtype: int64
```

In [44]:
```
print(train['Age'].value_counts())
print(test['Age'].value_counts())
```

```
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: Age, dtype: int64
26-35    93428
36-45    46711
18-25    42293
46-50    19577
51-55    16283
55+       9075
0-17      6232
Name: Age, dtype: int64
```

In [45]:
```
train['Occupation'].value_counts() # 21
```

Out[45]:
```
4     72308
0     69638
7     59133
1     47426
17    40043
20    33562
12    31179
14    27309
2     26588
16    25371
6     20355
3     17650
10    12930
5     12177
15    12165
11    11586
19     8461
13     7728
18     6622
9      6291
8      1546
Name: Occupation, dtype: int64
```

In [46]:
```
train['City_Category'].value_counts()
```

Out[46]:
```
B    231173
C    171175
A    147720
Name: City_Category, dtype: int64
```

In [47]:
```
train['Stay_In_Current_City_Years'].value_counts()
```

Out[47]:
```
1     193821
2     101838
3      95285
4+     84726
0      74398
Name: Stay_In_Current_City_Years, dtype: int64
```

In [48]:
```
train['Marital_Status'].value_counts()
```

Out[48]:
```
0    324731
1    225337
Name: Marital_Status, dtype: int64
```

In [49]:
```
train['Product_Category_1'].value_counts() # 20
```

Out[49]:
```
5     150933
1     140378
8     113925
11     24287
2      23864
6      20466
3      20213
4      11753
16      9828
15      6290
13      5549
10      5125
12      3947
7       3721
18      3125
20      2550
19      1603
14      1523
17       578
9        410
Name: Product_Category_1, dtype: int64
```

In [50]:
```
train['Product_Category_2'].value_counts() # 17
```

Out[50]:
```
8.0     64088
14.0    55108
2.0     49217
16.0    43255
15.0    37855
5.0     26235
4.0     25677
6.0     16466
11.0    14134
17.0    13320
13.0    10531
9.0      5693
12.0     5528
10.0     3043
3.0      2884
18.0     2770
7.0       626
Name: Product_Category_2, dtype: int64
```

In [51]:
```
train['Product_Category_3'].value_counts() # 15
```

Out[51]:
```
16.0    32636
15.0    28013
14.0    18428
17.0    16702
5.0     16658
8.0     12562
9.0     11579
12.0     9246
13.0     5459
6.0      4890
18.0     4629
4.0      1875
11.0     1805
10.0     1726
3.0       613
Name: Product_Category_3, dtype: int64
```

In [52]:
```
train.columns
```

Out[52]:
```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Categor
y',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category_1',
       'Product_Category_2', 'Product_Category_3', 'Purchase'],
      dtype='object')
```

In [53]:
```
test.columns
```

Out[53]:
```
Index(['Unnamed: 0', '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',
       'Comb'],
      dtype='object')
```

In [54]:
```
train_join_test = train[['User_ID', 'Product_ID']]
train = train[['Gender', 'Age', 'Occupation', 'City_Category', 'Stay_In_Curren
t_City_Years', 'Marital_Status', 'Product_Category_1',
        'Purchase']]
train.head()
```

Out[54]:

	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status
0	F	0-17	10	A	2	0
1	F	0-17	10	A	2	0
2	F	0-17	10	A	2	0
3	F	0-17	10	A	2	0
4	M	55+	16	C	4+	0

In [55]:
```
test = test[['Gender', 'Age', 'Occupation', 'City_Category', 'Stay_In_Current_
City_Years', 'Marital_Status', 'Product_Category_1']]
test.head()
```

Out[55]:

	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status
0	M	46-50	7	B	2	0
1	M	26-35	17	C	0	0
2	F	36-45	1	B	4+	0
3	F	36-45	1	B	4+	0
4	F	26-35	1	C	1	0

3. Feature Engineering

In [56]:
```
from sklearn.preprocessing import LabelEncoder
```

In [57]:
```
le = LabelEncoder()
```

In [58]:
```
train['Gender'] = le.fit_transform(train['Gender'])
test['Gender'] = le.fit_transform(test['Gender'])
```

In [59]:
```
train.head()
```

Out[59]:

	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status
0	0	0-17	10	A	2	0
1	0	0-17	10	A	2	0
2	0	0-17	10	A	2	0
3	0	0-17	10	A	2	0
4	1	55+	16	C	4+	0

In [60]:
```
test.head()
```

Out[60]:

	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status
0	1	46-50	7	B	2	0
1	1	26-35	17	C	0	0
2	0	36-45	1	B	4+	0
3	0	36-45	1	B	4+	0
4	0	26-35	1	C	1	0

In [61]:
```
train['Age'] = le.fit_transform(train['Age'])
test['Age'] = le.fit_transform(test['Age'])
```

In [62]:
```
train['City_Category'] = le.fit_transform(train['City_Category'])
```

In [63]:
```
test['City_Category'] = le.fit_transform(test['City_Category'])
```

In [64]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 8 columns):
Gender                        550068 non-null int32
Age                           550068 non-null int32
Occupation                    550068 non-null int64
City_Category                 550068 non-null int32
Stay_In_Current_City_Years    550068 non-null object
Marital_Status                550068 non-null int64
Product_Category_1            550068 non-null int64
Purchase                      550068 non-null int64
dtypes: int32(3), int64(4), object(1)
memory usage: 27.3+ MB
```

In [65]:
```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 233599 entries, 0 to 233598
Data columns (total 7 columns):
Gender                        233599 non-null int32
Age                           233599 non-null int32
Occupation                    233599 non-null int64
City_Category                 233599 non-null int32
Stay_In_Current_City_Years    233599 non-null object
Marital_Status                233599 non-null int64
Product_Category_1            233599 non-null int64
dtypes: int32(3), int64(3), object(1)
memory usage: 9.8+ MB
```

In [66]:
```
mask = train['Stay_In_Current_City_Years'] == '4+'
train.loc[mask, 'Stay_In_Current_City_Years'] = 4
```

In [67]:
```
mask1 = test['Stay_In_Current_City_Years'] == '4+'
test.loc[mask1, 'Stay_In_Current_City_Years'] = 4
```

In [68]:
```
train.head()
```

Out[68]:

	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status
0	0	0	10	0	2	0
1	0	0	10	0	2	0
2	0	0	10	0	2	0
3	0	0	10	0	2	0
4	1	6	16	2	4	0

In [69]:
```
test.head()
```

Out[69]:

	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status
0	1	4	7	1	2	0
1	1	2	17	2	0	0
2	0	3	1	1	4	0
3	0	3	1	1	4	0
4	0	2	1	2	1	0

In [70]:
```
train['Stay_In_Current_City_Years'] = train['Stay_In_Current_City_Years'].asty
pe('int')
test['Stay_In_Current_City_Years'] = test['Stay_In_Current_City_Years'].astype
('int')
```

In [71]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 8 columns):
Gender                        550068 non-null int32
Age                           550068 non-null int32
Occupation                    550068 non-null int64
City_Category                 550068 non-null int32
Stay_In_Current_City_Years    550068 non-null int32
Marital_Status                550068 non-null int64
Product_Category_1            550068 non-null int64
Purchase                      550068 non-null int64
dtypes: int32(4), int64(4)
memory usage: 25.2 MB
```

In [72]:
```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 233599 entries, 0 to 233598
Data columns (total 7 columns):
Gender                        233599 non-null int32
Age                           233599 non-null int32
Occupation                    233599 non-null int64
City_Category                 233599 non-null int32
Stay_In_Current_City_Years    233599 non-null int32
Marital_Status                233599 non-null int64
Product_Category_1            233599 non-null int64
dtypes: int32(4), int64(3)
memory usage: 8.9 MB
```

4. Model

4.1 Multiple Linear Regression

In [75]:
```
X_train = train.drop('Purchase',axis=1)
y_train = train['Purchase']
X_test = test
```

In [76]:
```
from sklearn.linear_model import LinearRegression
```

In [77]:
```
reg = LinearRegression()
```

In [78]:
```
reg.fit(X_train, y_train)
```

Out[78]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [89]:
```
prediction1 = reg.predict(X_test)
```

In [90]:
```
prediction1
```

Out[90]:
```
array([11492.07110442, 10785.26962272,  9091.44149746, ...,
       10815.25902604,  7372.75994218,  9638.41030116])
```

In [91]:
```
pred = pd.DataFrame(prediction1, columns = ['Purchase'])
pred.head()
```

Out[91]:

	Purchase
0	11492.071104
1	10785.269623
2	9091.441497
3	9528.730516
4	9739.615697

In [94]:
```
final = pd.concat([train_join_test,pred],axis=1)
final.head()
```

Out[94]:

	User_ID	Product_ID	Purchase
0	1000001	P00069042	11492.071104
1	1000001	P00248942	10785.269623
2	1000001	P00087842	9091.441497
3	1000001	P00085442	9528.730516
4	1000002	P00285442	9739.615697

In [95]:
```
final.to_csv('finalpurchase-MLR.csv', index = False) # RMSE: 4713.9227352998 , Rank: 1344
```

4.2 Decision Tree Regression

In [96]:
```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(X_train, y_train)
```

Out[96]:
```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=0, splitter='best')
```

In [97]:
```
prediction3 = regressor.predict(X_test)
pred = pd.DataFrame(prediction3, columns = ['Purchase'])
pred.head()
```

Out[97]:

	Purchase
0	16599.531250
1	10205.857143
2	6984.247191
3	2372.000000
4	2184.375000

In [99]:
```
final = pd.concat([train_join_test,pred],axis=1)
final.head()
```

Out[99]:

	User_ID	Product_ID	Purchase
0	1000001	P00069042	16599.531250
1	1000001	P00248942	10205.857143
2	1000001	P00087842	6984.247191
3	1000001	P00085442	2372.000000
4	1000002	P00285442	2184.375000

In [100]:
```
final.to_csv('finalpurchase-DTR.csv', index = False) # RMSE:  , Rank:
```

4.3 Random Forest Regression

In [101]:
```
from sklearn.ensemble import RandomForestRegressor
reg = RandomForestRegressor(n_estimators = 10, random_state = 0)
reg.fit(X_train, y_train)
```

Out[101]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
           oob_score=False, random_state=0, verbose=0, warm_start=False)
```

In [102]:
```
prediction4 = regressor.predict(X_test)
pred = pd.DataFrame(prediction4, columns = ['Purchase'])
pred.head()
```

Out[102]:

	Purchase
0	16599.531250
1	10205.857143
2	6984.247191
3	2372.000000
4	2184.375000

In [104]:
```
final = pd.concat([train_join_test,pred],axis=1)
final.head()
```

Out[104]:

	User_ID	Product_ID	Purchase
0	1000001	P00069042	16599.531250
1	1000001	P00248942	10205.857143
2	1000001	P00087842	6984.247191
3	1000001	P00085442	2372.000000
4	1000002	P00285442	2184.375000

In [105]:
```
final.to_csv('finalpurchase-RFR.csv', index = False) # RMSE: 3119.9813685216
 , Rank: 1143
```

ANN

In [ ]:
```
#import keras
from keras.models import Sequential
from keras.layers import Dense
```

In [ ]:
```
ann = Sequential()
```

In [ ]:
```
ann.add(Dense(6, init = 'uniform', activation = 'relu', input_dim = 7))
```

In [ ]:
```
ann.add(Dense(6, init = 'uniform', activation = 'relu'))
```

In [ ]:
```
ann.add(Dense(1, activation='linear'))
```

In [ ]:
```
ann.compile(loss='mse', optimizer='adam', metrics=['mse','mae'])
```

In [ ]:
```
ann.fit(X_train, y_train, batch_size = 10, nb_epoch = 10)
```

In [ ]:
```
prediction5 = ann.predict(X_test)
pred = pd.DataFrame(prediction5, columns = ['Purchase'])
pred.head()
```

In [ ]:
```
final = pd.concat([df_join_test,pred],axis=1)
final.head()
```

In [ ]:
```
final.to_csv('finalpurchase-ANN.csv', index = False) # RMSE:   , Rank:
```

Evaluation Results

Multiple Linear Regression- RMSE: 4713.9227352998 , Rank: 1344

Random Forest Regression- RMSE: 3119.9813685216 , Rank: 1143