In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv("Train.csv")
df.head()
```

Out[2]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Cate
0	1000001	P00069042	F	0-17	10	A	2	0	3	
1	1000001	P00248942	F	0-17	10	A	2	0	1	
2	1000001	P00087842	F	0-17	10	A	2	0	12	
3	1000001	P00085442	F	0-17	10	A	2	0	12	
4	1000002	P00285442	M	55+	16	C	4+	0	8	

In [3]:
```
## Appending test data to train data
df_test=pd.read_csv("blackFriday_test.csv")
df.append(df_test).head()
```

Out[3]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Cate
0	1000001	P00069042	F	0-17	10	A	2	0	3	
1	1000001	P00248942	F	0-17	10	A	2	0	1	
2	1000001	P00087842	F	0-17	10	A	2	0	12	
3	1000001	P00085442	F	0-17	10	A	2	0	12	
4	1000002	P00285442	M	55+	16	C	4+	0	8	

In [4]:
```
## Handling categorial value Gender
df.replace({"Gender":{"F":0,"M":1}},inplace=True)
```

In [5]:
```
df
```

Out[5]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Produc
0	1000001	P00069042	0	0-17	10	A	2	0	3	
1	1000001	P00248942	0	0-17	10	A	2	0	1	
2	1000001	P00087842	0	0-17	10	A	2	0	12	
3	1000001	P00085442	0	0-17	10	A	2	0	12	
4	1000002	P00285442	1	55+	16	C	4+	0	8	
...	...	...	...	...	...	...	...	...	...	...
550063	1006033	P00372445	1	51-55	13	B	1	1	20	
550064	1006035	P00375436	0	26-35	1	C	3	0	20	
550065	1006036	P00375436	0	26-35	15	B	4+	1	20	
550066	1006038	P00375436	0	55+	1	C	2	0	20	
550067	1006039	P00371644	0	46-50	0	B	4+	1	20	

550068 rows × 12 columns

In [6]:
```
df.drop(["User_ID"],axis=1,inplace=True)
```

In [7]:
```
## Handling categorial value AGE
df.Age.unique()
```

Out[7]:
```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

In [8]:
```
df.replace({"Age":{"0-17":1,"18-25":2,"26-35":3,"36-45":4,"46-50":5,"51-55":6,"55+":7}},inplace=True)
```

In [9]:
```
df_c=pd.get_dummies(df["City_Category"],drop_first=True)
df=pd.concat([df,df_c],axis=1)
df.drop("City_Category",axis=1,inplace=True)
```

In [10]:
```
#Missing values
df.isnull().sum()
df.head()
```

Out[10]:

	Product_ID	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Categor
0	P00069042	0	1	10	2	0	3	NaN	N
1	P00248942	0	1	10	2	0	1	6.0	1
2	P00087842	0	1	10	2	0	12	NaN	N
3	P00085442	0	1	10	2	0	12	14.0	N
4	P00285442	1	7	16	4+	0	8	NaN	N

In [11]:
```
## Replacing missing values
df["Product_Category_1"].value_counts()
```

Out[11]:
```
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 [12]:
```
df["Product_Category_1"].mode()[0]
```

Out[12]:
```
5
```

In [13]:
```
# Replace NaN values to mode values
df["Product_Category_1"]=df["Product_Category_1"].fillna(df["Product_Category_1"].mode()[0])
df["Product_Category_2"]=df["Product_Category_2"].fillna(df["Product_Category_2"].mode()[0])
df["Product_Category_3"]=df["Product_Category_3"].fillna(df["Product_Category_3"].mode()[0])
```

In [14]:
```
df.isnull().sum()
```

Out[14]:
```
Product_ID                  0
Gender                      0
Age                         0
Occupation                  0
Stay_In_Current_City_Years  0
```

```
Marital_Status                0
Product_Category_1            0
Product_Category_2            0
Product_Category_3            0
Purchase                      0
B                             0
C                             0
dtype: int64
```

In [15]:
```
df.head()
```

Out[15]:

	Product_ID	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Categor
0	P00069042	0	1	10	2	0	3	8.0	1
1	P00248942	0	1	10	2	0	1	6.0	1
2	P00087842	0	1	10	2	0	12	8.0	1
3	P00085442	0	1	10	2	0	12	14.0	1
4	P00285442	1	7	16	4+	0	8	8.0	1

In [16]:
```
df.Stay_In_Current_City_Years.unique()
```

Out[16]:
```
array(['2', '4+', '3', '1', '0'], dtype=object)
```

In [17]:
```
df.replace({"Stay_In_Current_City_Years":{"4+":4}},inplace=True)
```

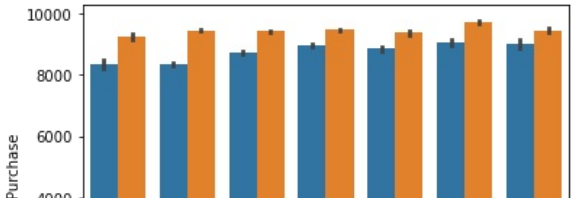
In [18]:
```
#Converting Stay_In_Current_City_Years to interger value
df["Stay_In_Current_City_Years"]=df["Stay_In_Current_City_Years"].astype(int)

df["B"]=df["B"].astype(int)
df["C"]=df["C"].astype(int)
df.info()

df.drop(["Product_ID"],axis=1,inplace=True)
```

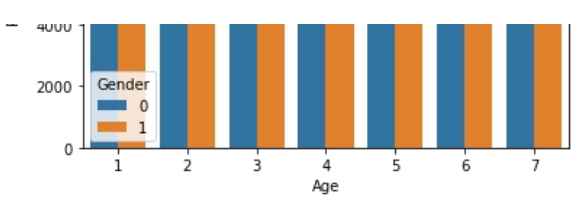
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   Product_ID                  550068 non-null  object 
 1   Gender                      550068 non-null  int64  
 2   Age                         550068 non-null  int64  
 3   Occupation                  550068 non-null  int64  
 4   Stay_In_Current_City_Years  550068 non-null  int32  
 5   Marital_Status              550068 non-null  int64  
 6   Product_Category_1          550068 non-null  int64  
 7   Product_Category_2          550068 non-null  float64
 8   Product_Category_3          550068 non-null  float64
 9   Purchase                    550068 non-null  int64  
 10  B                           550068 non-null  int32  
 11  C                           550068 non-null  int32  
dtypes: float64(2), int32(3), int64(6), object(1)
memory usage: 44.1+ MB
```

In [19]:
```
sns.barplot("Age","Purchase",hue="Gender",data=df)
```

```
C:\Users\mohit\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[19]:
```
<AxesSubplot:xlabel='Age', ylabel='Purchase'>
```





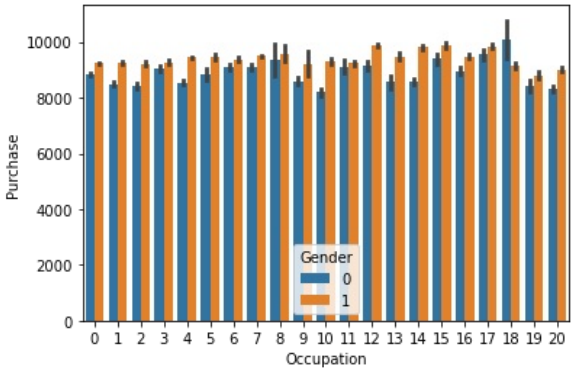
Purchase of Men is higher than Women

(0 = woman and 1 = Man)

In [20]:
```
sns.barplot("Occupation","Purchase",hue="Gender",data=df)
```

```
C:\Users\mohit\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables
as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other ar
guments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

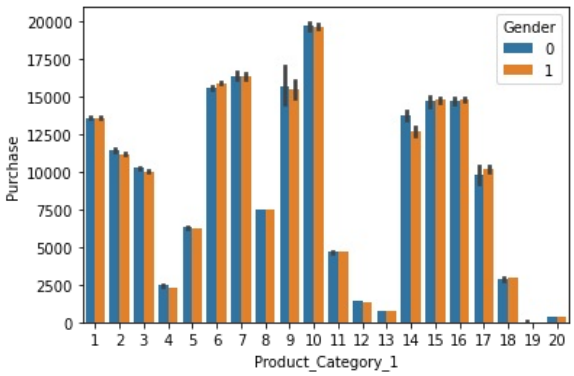
Out[20]:
```
<AxesSubplot:xlabel='Occupation', ylabel='Purchase'>
```



In [21]:
```
sns.barplot("Product_Category_1","Purchase",hue="Gender",data=df)
```

```
C:\Users\mohit\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables
as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other ar
guments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

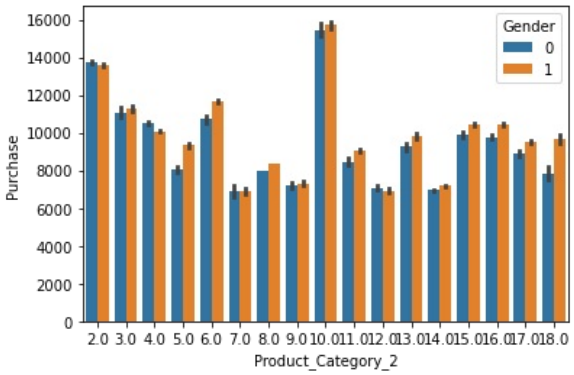
Out[21]:
```
<AxesSubplot:xlabel='Product_Category_1', ylabel='Purchase'>
```



In [22]:
```
sns.barplot("Product_Category_2","Purchase",hue="Gender",data=df)
```

```
C:\Users\mohit\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables
as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other ar
guments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

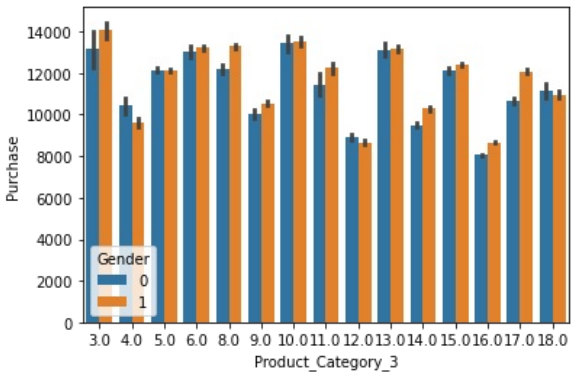
Out[22]: <AxesSubplot:xlabel='Product_Category_2', ylabel='Purchase'>



In [23]:
```
sns.barplot("Product_Category_3","Purchase",hue="Gender",data=df)
```

```
C:\Users\mohit\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables
as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other ar
guments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[23]: <AxesSubplot:xlabel='Product_Category_3', ylabel='Purchase'>



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

Out[24]:

	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchas
0	0	1	10	2	0	3	8.0	16.0	837
1	0	1	10	2	0	1	6.0	14.0	1520
2	0	1	10	2	0	12	8.0	16.0	142
3	0	1	10	2	0	12	14.0	16.0	105
4	1	7	16	4	0	8	8.0	16.0	796

In [25]:
```
df[df["Purchase"].isnull()]
```

Out[25]:

	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase

In [26]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

X = df[['Gender', 'Age', 'Occupation', 'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category_1', 'Pro
y = df['Purchase']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LinearRegression()
```

```
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

In [27]:

```
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:",mse)
```

```
Mean Squared Error: 21849890.001479708
```

Observation : The MSE of 21849890.001479708 suggests that model's predictions have a high level of error, and the model may not be accurately capturing the variation in purchase amounts.

In [28]:

```
coefficients = model.coef_
intercept = model.intercept_

print("Coefficients:")
for i, coef in enumerate(coefficients):
    print(X.columns[i],":",coef)
    print("")
print("Intercept:",intercept)
```

```
Coefficients:
Gender : 521.8378155162695

Age : 119.78653615344957

Occupation : 6.279789701724108

Stay_In_Current_City_Years : 9.266420789969565

Marital_Status : -51.735667010569536

Product_Category_1 : -403.3422550002329

Product_Category_2 : -1.8324551399585656

Product_Category_3 : -154.99633682049944

B : 154.86873164820193

C : 662.8101381185604

Intercept: 12656.439496204008
```

As we got all coefficients of independent variables we can apply multiple linear regression.

like y=b1(x1)+b2(x2)+...+bn(xn)+c

In [29]:

```
## Taking an example
gen=1
age=1
occ=4
city=2
m_status=0
p_cat1=12
p_cat2=6.0
p_cat3=16.0
b=1
c=0


predict_purchase=gen*(coefficients[0])+age*(coefficients[1])+occ*(coefficients[2])+city*(coefficients[3])+(m_stat
print("Gender : {}\n• Age : {}\n• Cccupation : {} (1,2.. represents diffrent occupation)\n• Marital status :{} (s
print("• Predicted purchase value :",predict_purchase)
```

```
Gender : 1
• Age : 1
• Cccupation : 4 (1,2.. represents diffrent occupation)
• Marital status :2 (single=0 , married=1)
• Product category 1 : 0
• Product category 2 : 12
• Product category 3 : 6.0
• B : 16.0 (here b and c shows city category)
• C : 1 (if B==0 and C==0 then the city is A)
```

- Predicted purchase value : 6165.541399938227

In [ ]: