Black Friday Dataset EDA And Feature Engineering

Cleaning and preparing the data for model training

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

In [2]:
```
df_train = pd.read_csv(r'C:\Users\MALU\Downloads\black friday\train.csv')
df_test = pd.read_csv(r'C:\Users\MALU\Downloads\black friday\test.csv')
```

In [3]:
```
df_train.head(5)
```

Out[3]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Produ
0	1000001	P00069042	F	0-17	10	A	2	0	3	NaN	
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	
2	1000001	P00087842	F	0-17	10	A	2	0	12	NaN	
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	
4	1000002	P00285442	M	55+	16	C	4+	0	8	NaN	

In [6]:
```
df_train.shape
```

Out[6]: (550068, 12)

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

Out[5]: (233599, 11)

In [7]:
```
df_test.head(5)
```

Out[7]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Produ
0	1000004	P00128942	M	46-50	7	B	2	1	1	11.0	
1	1000009	P00113442	M	26-35	17	C	0	0	3	5.0	
2	1000010	P00288442	F	36-45	1	B	4+	1	5	14.0	
3	1000010	P00145342	F	36-45	1	B	4+	1	4	9.0	
4	1000011	P00053842	F	26-35	1	C	1	0	4	5.0	

In [4]:
```
# Merge both train and test data
df = pd.concat([df_train,df_test],axis=0)
```

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

Out[5]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Produ
0	1000001	P00069042	F	0-17	10	A	2	0	3	NaN	
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	
2	1000001	P00087842	F	0-17	10	A	2	0	12	NaN	
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	
4	1000002	P00285442	M	55+	16	C	4+	0	8	NaN	

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

Out[6]: (783667, 12)

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 783667 entries, 0 to 233598
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   User_ID                     783667 non-null  int64
 1   Product_ID                  783667 non-null  object
 2   Gender                      783667 non-null  object
 3   Age                         783667 non-null  object
 4   Occupation                  783667 non-null  int64
 5   City_Category               783667 non-null  object
 6   Stay_In_Current_City_Years  783667 non-null  object
 7   Marital_Status              783667 non-null  int64
 8   Product_Category_1          783667 non-null  int64
 9   Product_Category_2          537685 non-null  float64
 10  Product_Category_3          237858 non-null  float64
 11  Purchase                    550068 non-null  float64
dtypes: float64(3), int64(4), object(5)
memory usage: 77.7+ MB
```

In [8]:
```
df.describe()
```

Out[8]:

	User_ID	Occupation	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
count	7.836670e+05	783667.000000	783667.000000	783667.000000	537685.000000	237858.000000	550068.000000
mean	1.003029e+06	8.079300	0.409777	5.366196	9.844506	12.668605	9263.968713
std	1.727267e+03	6.522206	0.491793	3.878160	5.089093	4.125510	5023.065394
min	1.000001e+06	0.000000	0.000000	1.000000	2.000000	3.000000	12.000000
25%	1.001519e+06	2.000000	0.000000	1.000000	5.000000	9.000000	5823.000000
50%	1.003075e+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 [9]:
```
df.drop(['User_ID'],axis=1,inplace=True)
```

In [10]:
```
df.sample(5)
```

Out[10]:

	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_C
295470	P00043742	M	26-35	2	A	3	0	5	8.0	
260271	P00217342	M	46-50	0	A	1	1	1	15.0	
1228	P00240142	F	18-25	4	C	0	0	5	NaN	
469094	P00220442	F	18-25	14	C	2	0	5	14.0	
131703	P00106942	M	46-50	6	C	2	1	8	NaN	

In [11]:
```
# Handling Categorical feature -Gender
df['Gender']=df['Gender'].map({'F':0,'M':1})
df.head(5)
```

Out[11]:

	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Categor
0	P00069042	0	0-17	10	A	2	0	3	NaN	
1	P00248942	0	0-17	10	A	2	0	1	6.0	
2	P00087842	0	0-17	10	A	2	0	12	NaN	
3	P00085442	0	0-17	10	A	2	0	12	14.0	
4	P00285442	1	55+	16	C	4+	0	8	NaN	

In [12]:
```
# Handling Categorical feature -Age
df['Age'].unique()
```

Out[12]:
```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

In [13]:
```
df['Age']=df['Age'].map({'0-17':1,'18-25':2,'26-35':3,'36-45':4,'46-50':5,'51-55':6,'55+':7})
```

In [14]:
```
df.sample(5)
```

Out[14]:

	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_C
59657	P00289842	1	2	4	C	3	0	1	16.0	
233378	P00314542	0	3	20	B	1	1	8	NaN	
189601	P00230942	1	5	16	B	0	0	5	14.0	
517875	P00249742	1	4	7	A	4+	1	1	15.0	
362446	P00109742	1	3	20	C	0	0	8	NaN	

In [15]:
```
# Handling Categorical feature -City_Category
df_city=pd.get_dummies(df['City_Category'],drop_first=True)
df_city = df_city.astype(int)
```

In [16]:
```
df_city.head(5)
```

Out[16]:

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

In [17]:
```
df=pd.concat([df,df_city],axis=1)
df.head(5)
```

Out[17]:

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

In [18]:
```
# Droping City_Category
df.drop('City_Category',axis=1,inplace=True)
```

In [19]:
```
df.head(2)
```

Out[19]:

	Product_ID	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
0	P00069042	0	1	10	2	0	3	NaN	NaN	8370.0
1	P00248942	0	1	10	2	0	1	6.0	14.0	15200.0

In [20]:
```
# Handling Missing values
df.isnull().sum()
```

Out[20]:
```
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            245982
Product_Category_3            545809
Purchase                      233599
B                                  0
C                                  0
dtype: int64
```

In [21]:
```
df['Product_Category_2'].unique()
```

Out[21]:
```
array([nan,  6., 14.,  2.,  8., 15., 16., 11.,  5.,  3.,  4., 12.,  9.,
       10., 17., 13.,  7., 18.])
```

In [22]:
```
df['Product_Category_2'].value_counts()
```

Out[22]:
```
Product_Category_2
8.0     91317
14.0    78834
2.0     70498
16.0    61687
15.0    54114
5.0     37165
4.0     36705
6.0     23575
11.0    20230
17.0    19104
13.0    15054
9.0      8177
12.0     7801
10.0     4420
3.0      4123
18.0     4027
7.0       854
Name: count, dtype: int64
```

In [43]:
```
df['Product_Category_2'].mode()[0]
```

Out[43]: 8.0

In [23]:
```
## Replace the missing values with mode
df['Product_Category_2']=df['Product_Category_2'].fillna(df['Product_Category_2'].mode()[0])
```

In [24]:
```
df['Product_Category_2'].isnull().sum()
```

Out[24]: 0

In [25]:
```
# 'Product_Category_3
df['Product_Category_3'].value_counts()
```

Out[25]:
```
Product_Category_3
16.0    46469
15.0    39968
14.0    26283
17.0    23818
5.0     23799
8.0     17861
9.0     16532
12.0    13115
13.0     7849
6.0      6888
18.0     6621
4.0      2691
11.0     2585
10.0     2501
3.0       878
Name: count, dtype: int64
```

In [47]:
```
df['Product_Category_3'].mode()[0]
```

Out[47]: 16.0

In [26]:
```
## Replace the missing values with mode
df['Product_Category_3']=df['Product_Category_3'].fillna(df['Product_Category_3'].mode()[0])
```

In [27]:
```
df['Product_Category_3'].isnull().sum()
```

Out[27]: 0

In [28]:
```
df['Stay_In_Current_City_Years'].unique()
```

Out[28]: array(['2', '4+', '3', '1', '0'], dtype=object)

In [29]:
```
df['Stay_In_Current_City_Years']=df['Stay_In_Current_City_Years'].str.replace('+','')
```

In [30]:
```
df.head(5)
```

Out[30]:

	Product_ID	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
0	P00069042	0	1	10	2	0	3	8.0	16.0	8370.0
1	P00248942	0	1	10	2	0	1	6.0	14.0	15200.0
2	P00087842	0	1	10	2	0	12	8.0	16.0	1422.0
3	P00085442	0	1	10	2	0	12	14.0	16.0	1057.0
4	P00285442	1	7	16	4	0	8	8.0	16.0	7969.0

In [31]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 783667 entries, 0 to 233598
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   Product_ID                  783667 non-null  object
 1   Gender                      783667 non-null  int64
 2   Age                         783667 non-null  int64
 3   Occupation                  783667 non-null  int64
 4   Stay_In_Current_City_Years  783667 non-null  object
 5   Marital_Status              783667 non-null  int64
 6   Product_Category_1          783667 non-null  int64
 7   Product_Category_2          783667 non-null  float64
 8   Product_Category_3          783667 non-null  float64
 9   Purchase                    550068 non-null  float64
 10  B                           783667 non-null  int32
 11  C                           783667 non-null  int32
dtypes: float64(3), int32(2), int64(5), object(2)
memory usage: 71.7+ MB
```

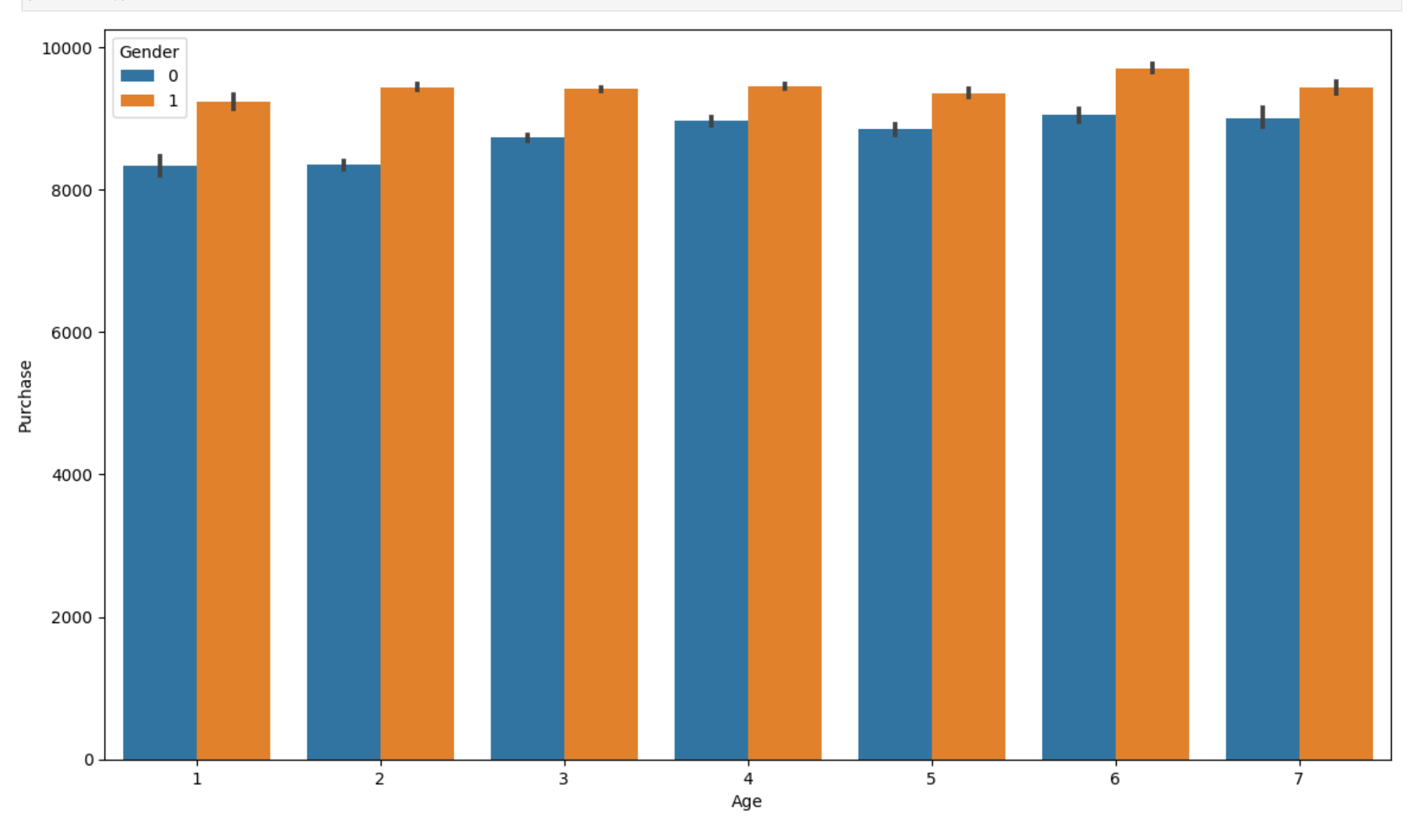
In [32]:
```
##convert object into integers
df['Stay_In_Current_City_Years']=df['Stay_In_Current_City_Years'].astype(int)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 783667 entries, 0 to 233598
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   Product_ID                  783667 non-null  object
 1   Gender                      783667 non-null  int64
 2   Age                         783667 non-null  int64
 3   Occupation                  783667 non-null  int64
 4   Stay_In_Current_City_Years  783667 non-null  int32
 5   Marital_Status              783667 non-null  int64
 6   Product_Category_1          783667 non-null  int64
 7   Product_Category_2          783667 non-null  float64
 8   Product_Category_3          783667 non-null  float64
 9   Purchase                    550068 non-null  float64
 10  B                           783667 non-null  int32
 11  C                           783667 non-null  int32
dtypes: float64(3), int32(3), int64(5), object(1)
memory usage: 68.8+ MB
```

In [33]:
```
df['B']=df['B'].astype(int)
df['C']=df['C'].astype(int)
df.info()
```

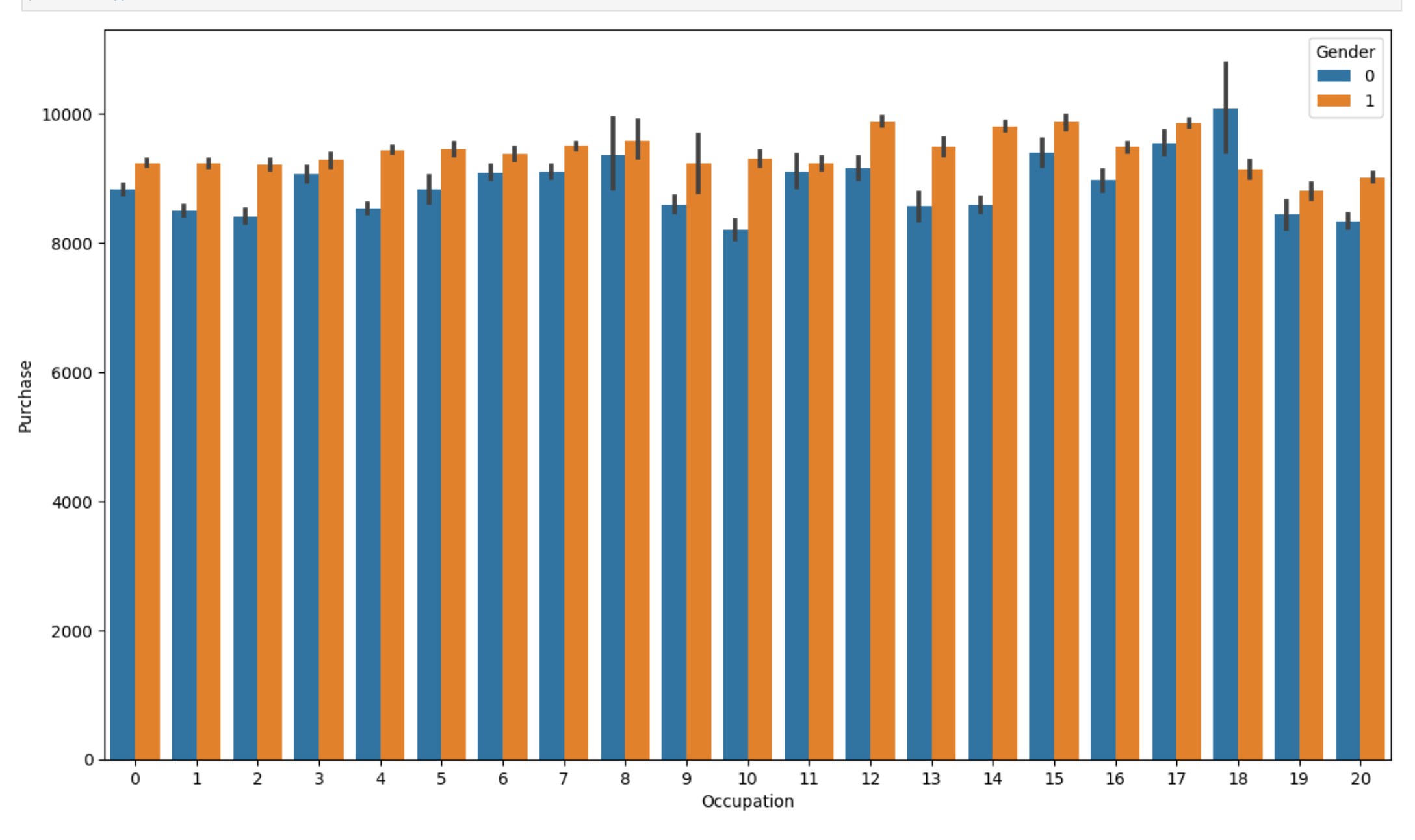
```
<class 'pandas.core.frame.DataFrame'>
Index: 783667 entries, 0 to 233598
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   Product_ID                  783667 non-null  object
 1   Gender                      783667 non-null  int64
 2   Age                         783667 non-null  int64
 3   Occupation                  783667 non-null  int64
 4   Stay_In_Current_City_Years  783667 non-null  int32
 5   Marital_Status              783667 non-null  int64
 6   Product_Category_1          783667 non-null  int64
 7   Product_Category_2          783667 non-null  float64
 8   Product_Category_3          783667 non-null  float64
 9   Purchase                    550068 non-null  float64
 10  B                           783667 non-null  int32
 11  C                           783667 non-null  int32
dtypes: float64(3), int32(3), int64(5), object(1)
memory usage: 68.8+ MB
```

In [39]:
```
plt.figure(figsize=(14,8))
sns.barplot(x='Age',y='Purchase',hue='Gender',data=df)
plt.show()
```

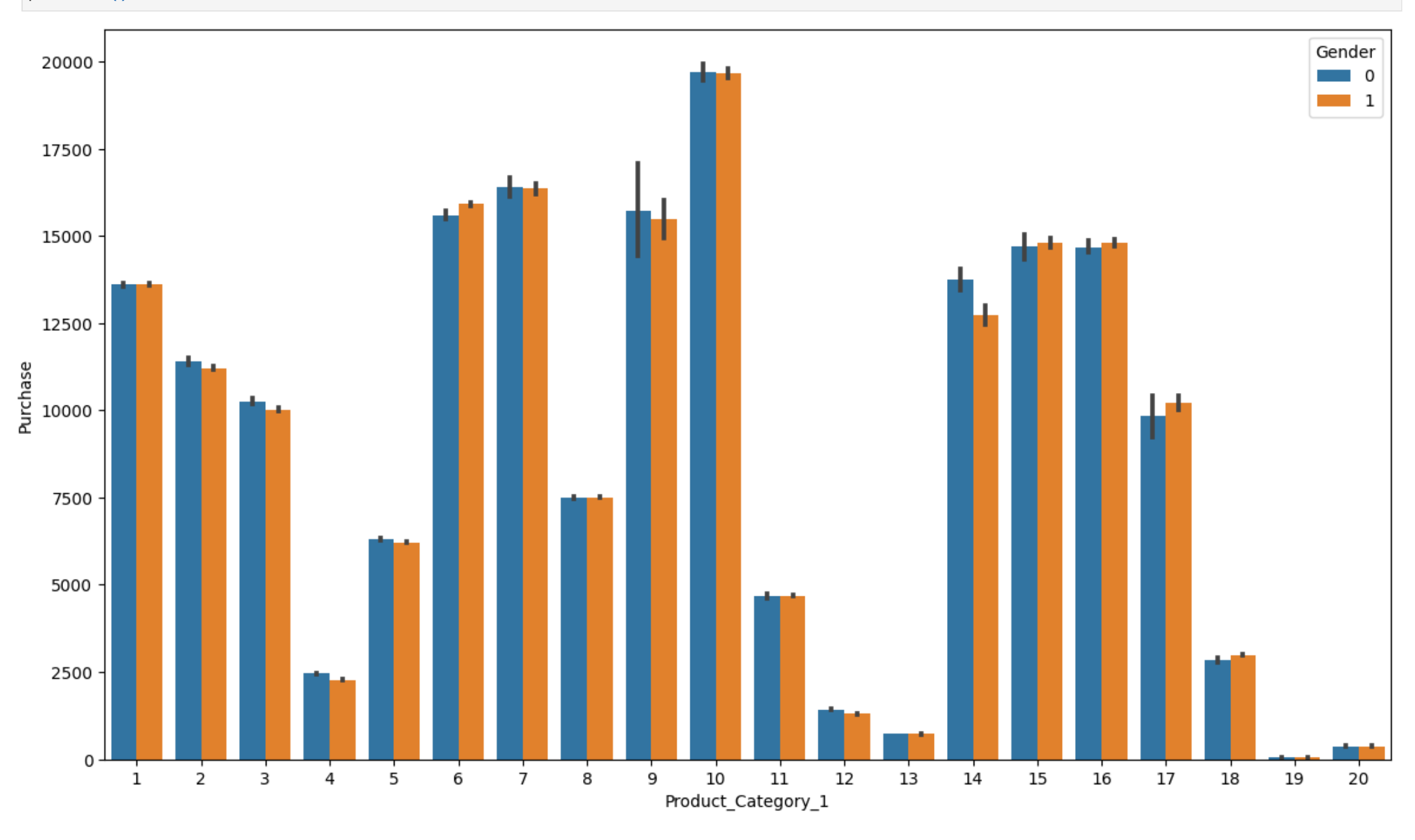


Purchasing of men is high than women

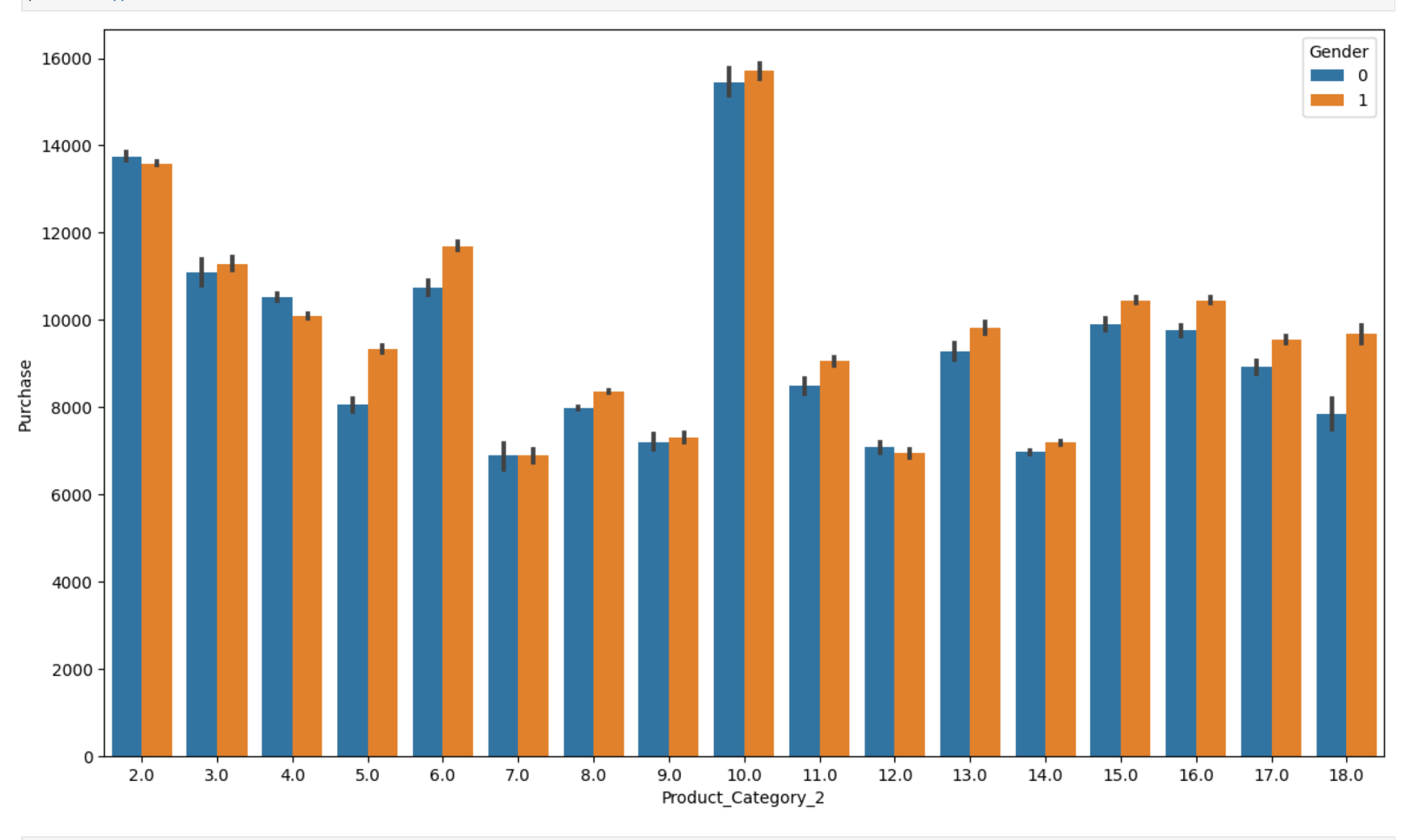
In [40]:
```
plt.figure(figsize=(14,8))
sns.barplot(x='Occupation',y='Purchase',hue='Gender',data=df)
plt.show()
```



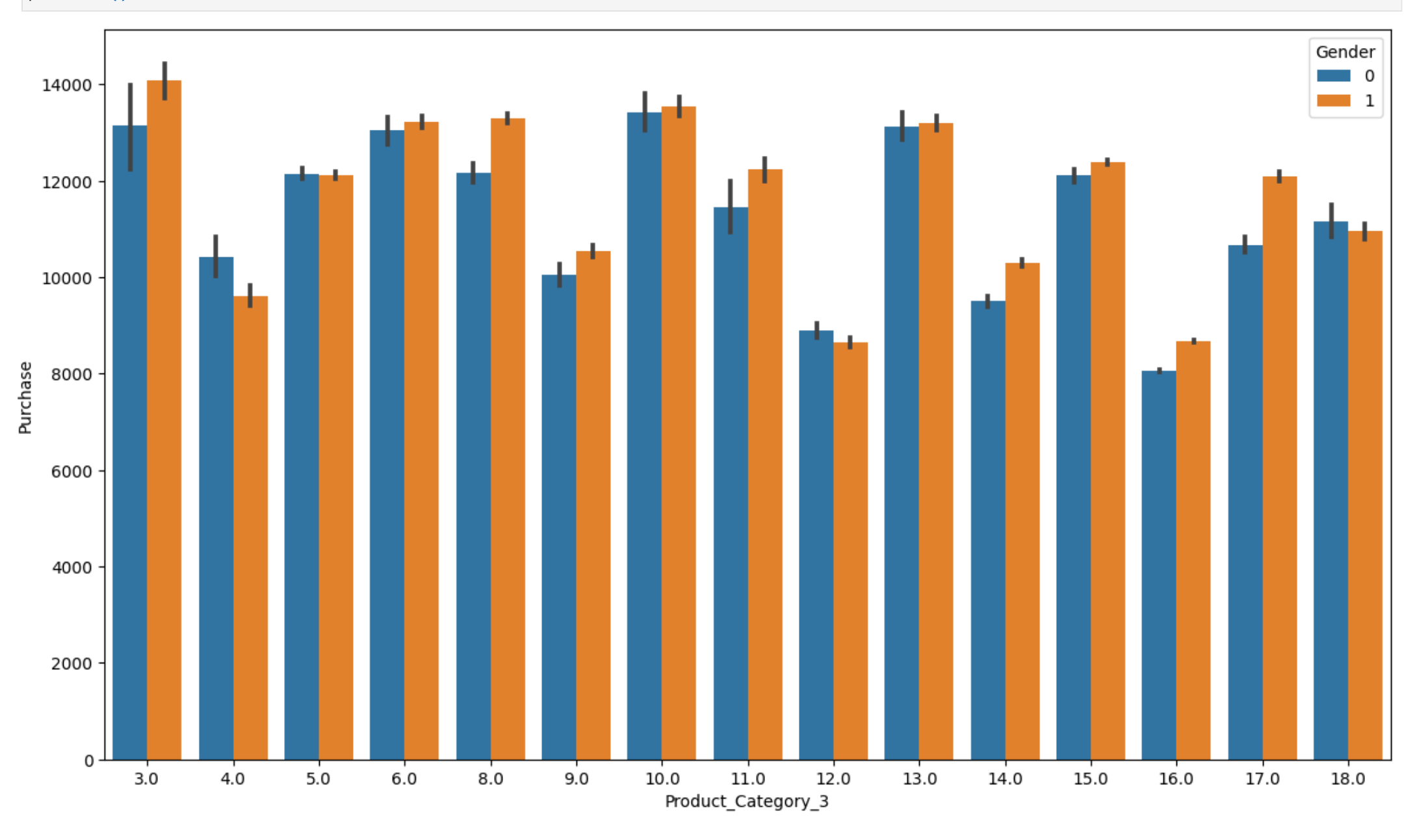
In [41]:
```
plt.figure(figsize=(14,8))
sns.barplot(x='Product_Category_1',y='Purchase',hue='Gender',data=df)
plt.show()
```



In [42]:
```
plt.figure(figsize=(14,8))
sns.barplot(x='Product_Category_2',y='Purchase',hue='Gender',data=df)
plt.show()
```



In [43]:
```
plt.figure(figsize=(14,8))
sns.barplot(x='Product_Category_3',y='Purchase',hue='Gender',data=df)
plt.show()
```



In [44]:
```
# Feature scaling
df_test=df[df['Purchase'].isnull()]
```

In [46]:
```
df_train=df[~df['Purchase'].isnull()]
```

In [48]:
```
X=df_train.drop('Purchase',axis=1)
```

In [51]:
```
y=df_train['Purchase']
```

In [53]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
     X, y, test_size=0.33, random_state=42)
```

In [54]:
```
X_train.drop('Product_ID',axis=1,inplace=True)
X_test.drop('Product_ID',axis=1,inplace=True)
```

In [55]:
```
## feature Scaling
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

In [ ]: