products puchased dataset

December 17, 2023

[2]:
```
import pandas as pd
import numpy as npn
import matplotlib.pyplot as plt
import seaborn as sns
```

[3]:
```
#3 import the dataset
data_train = pd.read_csv('C:/Users/Vikas/Desktop/train.csv')
```

[4]:
```
data_train.head()
```

[4]:

	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

[5]:
```
data_test = pd.read_csv('C:/Users/Vikas/Desktop/test.csv')
data_test.head()
```

[5]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category
0	1000004	P00128942	M	46-50	7	B
1	1000009	P00113442	M	26-35	17	C
2	1000010	P00288442	F	36-45	1	B

```
3  1000010  P00145342      F  36-45           1             B
4  1000011  P00053842      F  26-35           1             C

  Stay_In_Current_City_Years  Marital_Status  Product_Category_1  \
0                          2               1                   1
1                          0               0                   3
2                         4+               1                   5
3                         4+               1                   4
4                          1               0                   4

   Product_Category_2  Product_Category_3
0                11.0                 NaN
1                 5.0                 NaN
2                14.0                 NaN
3                 9.0                 NaN
4                 5.0                12.0
```

[6]:
```
## i want to combine the train and test data
data = data_train.append(data_test)
```

```
C:\Users\Vikas\AppData\Local\Temp\ipykernel_20108\3179184531.py:2:
FutureWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  data = data_train.append(data_test)
```

[7]:
```
data.head()
```

[7]:
```
   User_ID Product_ID Gender   Age  Occupation City_Category  \
0  1000001  P00069042      F  0-17          10             A
1  1000001  P00248942      F  0-17          10             A
2  1000001  P00087842      F  0-17          10             A
3  1000001  P00085442      F  0-17          10             A
4  1000002  P00285442      M   55+          16             C

  Stay_In_Current_City_Years  Marital_Status  Product_Category_1  \
0                          2               0                   3
1                          2               0                   1
2                          2               0                  12
3                          2               0                  12
4                         4+               0                   8

   Product_Category_2  Product_Category_3  Purchase
0                 NaN                 NaN    8370.0
1                 6.0                14.0   15200.0
2                 NaN                 NaN    1422.0
3                14.0                 NaN    1057.0
4                 NaN                 NaN    7969.0
```

[8]:
```
## basic code see
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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
```

[9]:
```
## i want to delet the userid
data.drop(['User_ID'],axis=1,inplace=True)
```

[10]:
```
data.head()
```

[10]:

	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	P00069042	F	0-17	10	A	2	0	3	NaN	NaN	8370.0
1	P00248942	F	0-17	10	A	2	0	1	6.0	14.0	15200.0
2	P00087842	F	0-17	10	A	2	0	12	NaN	NaN	1422.0
3	P00085442	F	0-17	10	A	2	0	12	14.0	NaN	1057.0
4	P00285442	M	55+	16	C	4+	0	8	NaN	NaN	7969.0

[12]:
```
## i want to convert gender col the numerical femal is0 male is 1
data['Gender']=data['Gender'].map({'F':0,'M':1})
data.head()
```

[12]:

	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	P00069042	0	0-17	10	A	2	0	3	NaN	NaN	8370.0
1	P00248942	0	0-17	10	A	2	0	1	6.0	14.0	15200.0
2	P00087842	0	0-17	10	A	2	0	12	NaN	NaN	1422.0
3	P00085442	0	0-17	10	A	2	0	12	14.0	NaN	1057.0
4	P00285442	1	55+	16	C	4+	0	8	NaN	NaN	7969.0

[13]:
```
## HANDLE CATEGORICAL FEATURE AGE
data['Age'].unique()
```

[13]:
```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

[22]:
```
data['Age'] = data['Age'].map({'0-17':1,'18-25':2,'26-35':3,'36-45':4,'46-50':
↪5,'51-55:6,'55+':7})
data['Age']
```

```
  Cell In[22], line 1
    data['Age'] = data['Age'].map({'0-17':1,'18-25':2,'26-35':3,'36-45':
↪4,'46-50':5,'51-55:6,'55+':7})

↪                                ^
SyntaxError: unterminated string literal (detected at line 1)
```

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

[28]:
```
df_city = pd.get_dummies(data['City_Category'],drop_first=True)
df_city
```

[28]:
```
         B  C
0        0  0
1        0  0
2        0  0
3        0  0
4        0  1
…      .. ..
233594   1  0
233595   1  0
233596   1  0
233597   0  1
233598   1  0

[783667 rows x 2 columns]
```

[30]:
```
## i want to cancatenate the data
data = pd.concat([data,df_city],axis=1)
data.head()
```

[30]:
```
   Product_ID  Gender  Age  Occupation City_Category  \
0  P00069042       0  NaN          10             A
1  P00248942       0  NaN          10             A
2  P00087842       0  NaN          10             A
3  P00085442       0  NaN          10             A
4  P00285442       1  NaN          16             C

  Stay_In_Current_City_Years  Marital_Status  Product_Category_1  \
0                          2               0                   3
1                          2               0                   1
2                          2               0                  12
3                          2               0                  12
4                         4+               0                   8

   Product_Category_2  Product_Category_3  Purchase  B  C  B  C
0                 NaN                 NaN    8370.0  0  0  0  0
1                 6.0                14.0   15200.0  0  0  0  0
2                 NaN                 NaN    1422.0  0  0  0  0
3                14.0                 NaN    1057.0  0  0  0  0
4                 NaN                 NaN    7969.0  0  1  0  1
```

[32]:
```
data.drop('City_Category',axis=1,inplace=True)
```

[33]:
```
data.head()
```

[33]:
```
  Product_ID  Gender  Age  Occupation Stay_In_Current_City_Years  \
0  P00069042       0  NaN          10                          2
1  P00248942       0  NaN          10                          2
2  P00087842       0  NaN          10                          2
3  P00085442       0  NaN          10                          2
4  P00285442       1  NaN          16                         4+

   Marital_Status  Product_Category_1  Product_Category_2  Product_Category_3  \
0               0                   3                 NaN                 NaN
1               0                   1                 6.0                14.0
2               0                  12                 NaN                 NaN
3               0                  12                14.0                 NaN
4               0                   8                 NaN                 NaN

   Purchase  B  C  B  C
0    8370.0  0  0  0  0
1   15200.0  0  0  0  0
2    1422.0  0  0  0  0
3    1057.0  0  0  0  0
4    7969.0  0  1  0  1
```

[35]:
```
data.isnull().sum()
```

[35]:
```
Product_ID                        0
Gender                            0
Age                          783667
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
B                                 0
C                                 0
dtype: int64
```

[37]:
```
data['Product_Category_2'].mode()[0]
```

[37]:
```
8.0
```

[38]:
```
## focus on replceing missing values
## replace the missing values with mode
data['Product_Category_2'] = data['Product_Category_2'].
 ↪fillna(data['Product_Category_2'].mode()[0])
```

[39]:
```
data['Product_Category_2'].isnull().sum()
```

[39]:
```
0
```

[40]:
```

```

[41]:
```
data['Product_Category_3'].unique()
```

[41]:
```
array([16., 14., 17.,  5.,  4., 15.,  8.,  9., 13.,  6., 12.,  3., 18.,
       11., 10.])
```

[43]:
```
data['Product_Category_3'].value_counts()
```

[43]:
```
16.0    592278
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: Product_Category_3, dtype: int64
```

[44]:
```
## product category 3 replace missing values
data['Product_Category_3'] = data['Product_Category_3'].
 ↪fillna(data['Product_Category_3'].mode()[0])
```

[45]:
```
data.head()
```

[45]:

	Product_ID	Gender	Age	Occupation	Stay_In_Current_City_Years	\
0	P00069042	0	NaN	10	2	
1	P00248942	0	NaN	10	2	
2	P00087842	0	NaN	10	2	
3	P00085442	0	NaN	10	2	
4	P00285442	1	NaN	16	4+	

	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	\
0	0	3	8.0	16.0	
1	0	1	6.0	14.0	
2	0	12	8.0	16.0	
3	0	12	14.0	16.0	

```
4                0                   8                 8.0                16.0

   Purchase  B  C  B  C
0    8370.0  0  0  0  0
1   15200.0  0  0  0  0
2    1422.0  0  0  0  0
3    1057.0  0  0  0  0
4    7969.0  0  1  0  1
```

[49]:
```
## convert the object inro integer
data['Stay_In_Current_City_Years'] = data['Stay_In_Current_City_Years'].
 ↪replace('4+', 5).astype(int)
data.head()
```

[49]:
```
   Product_ID  Gender  Age  Occupation  Stay_In_Current_City_Years  \
0  P00069042       0  NaN          10                           2
1  P00248942       0  NaN          10                           2
2  P00087842       0  NaN          10                           2
3  P00085442       0  NaN          10                           2
4  P00285442       1  NaN          16                           5

   Marital_Status  Product_Category_1  Product_Category_2  Product_Category_3  \
0               0                   3                 8.0                16.0
1               0                   1                 6.0                14.0
2               0                  12                 8.0                16.0
3               0                  12                14.0                16.0
4               0                   8                 8.0                16.0

   Purchase  B  C  B  C
0    8370.0  0  0  0  0
1   15200.0  0  0  0  0
2    1422.0  0  0  0  0
3    1057.0  0  0  0  0
4    7969.0  0  1  0  1
```

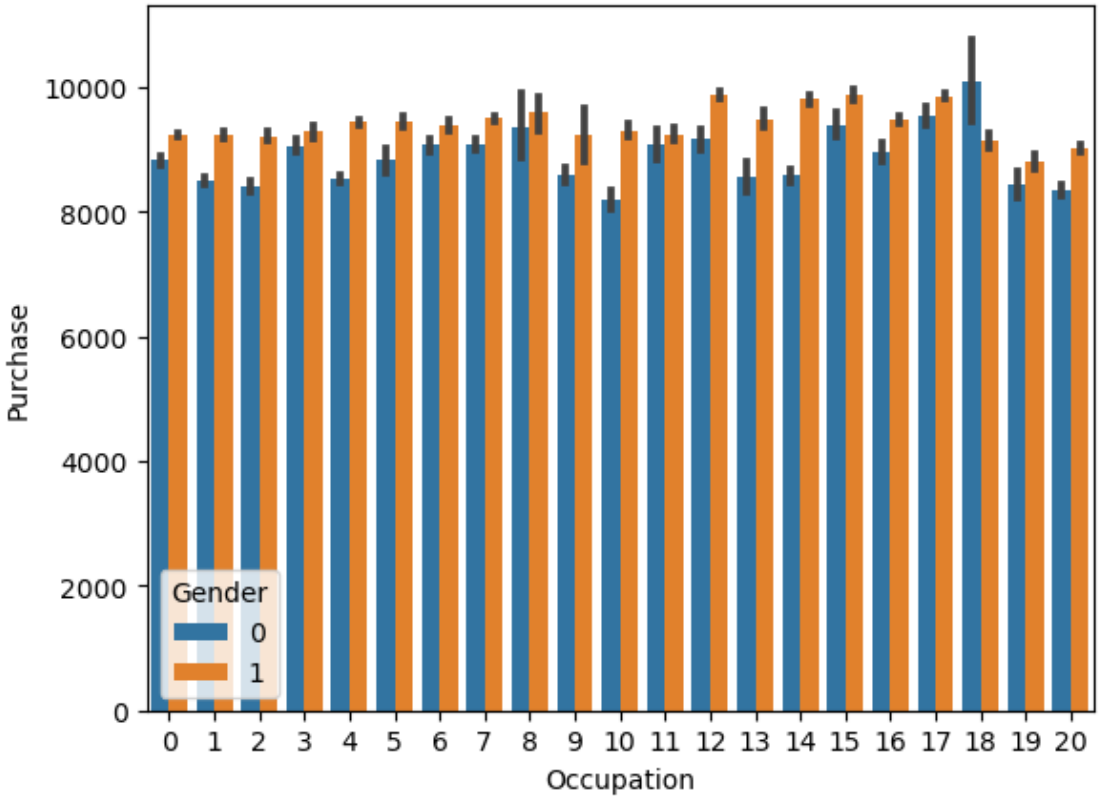
[50]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 783667 entries, 0 to 233598
Data columns (total 14 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   Product_ID                  783667 non-null  object
 1   Gender                      783667 non-null  int64
 2   Age                         0 non-null       float64
 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  uint8
 11  C                           783667 non-null  uint8
 12  B                           783667 non-null  uint8
 13  C                           783667 non-null  uint8
dtypes: float64(4), int32(1), int64(4), object(1), uint8(4)
memory usage: 65.8+ MB
```

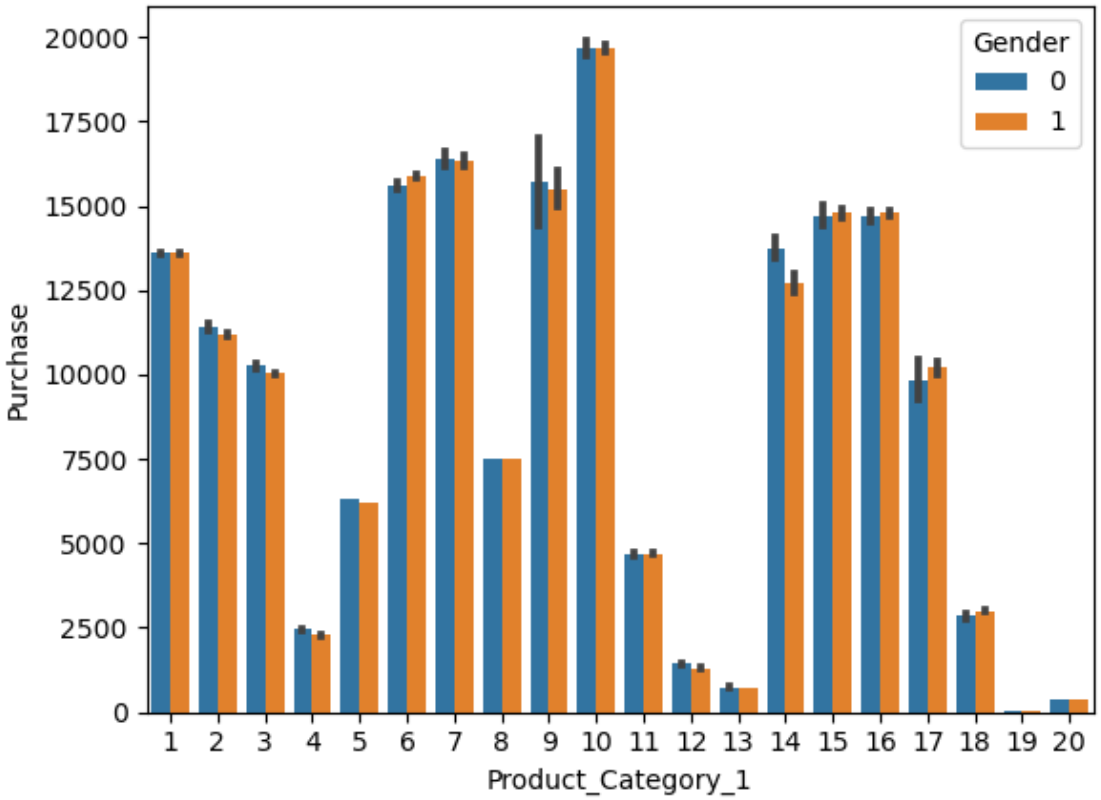
[63]:
```
sns.barplot(x='Occupation',y='Purchase', hue='Gender', data=data)
```

[63]: <Axes: xlabel='Occupation', ylabel='Purchase'>



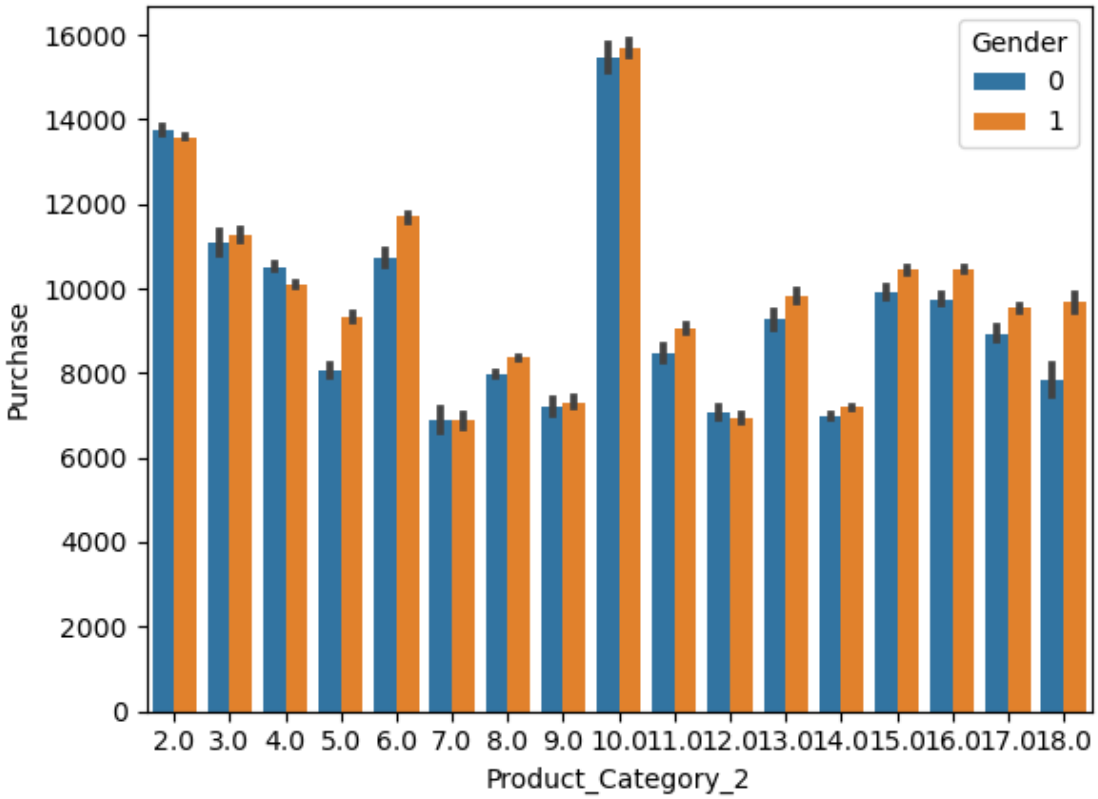
[65]:
```
## product category purchased by gender
sns.barplot(x='Product_Category_1',y='Purchase',hue='Gender',data=data)
```

[65]: <Axes: xlabel='Product_Category_1', ylabel='Purchase'>



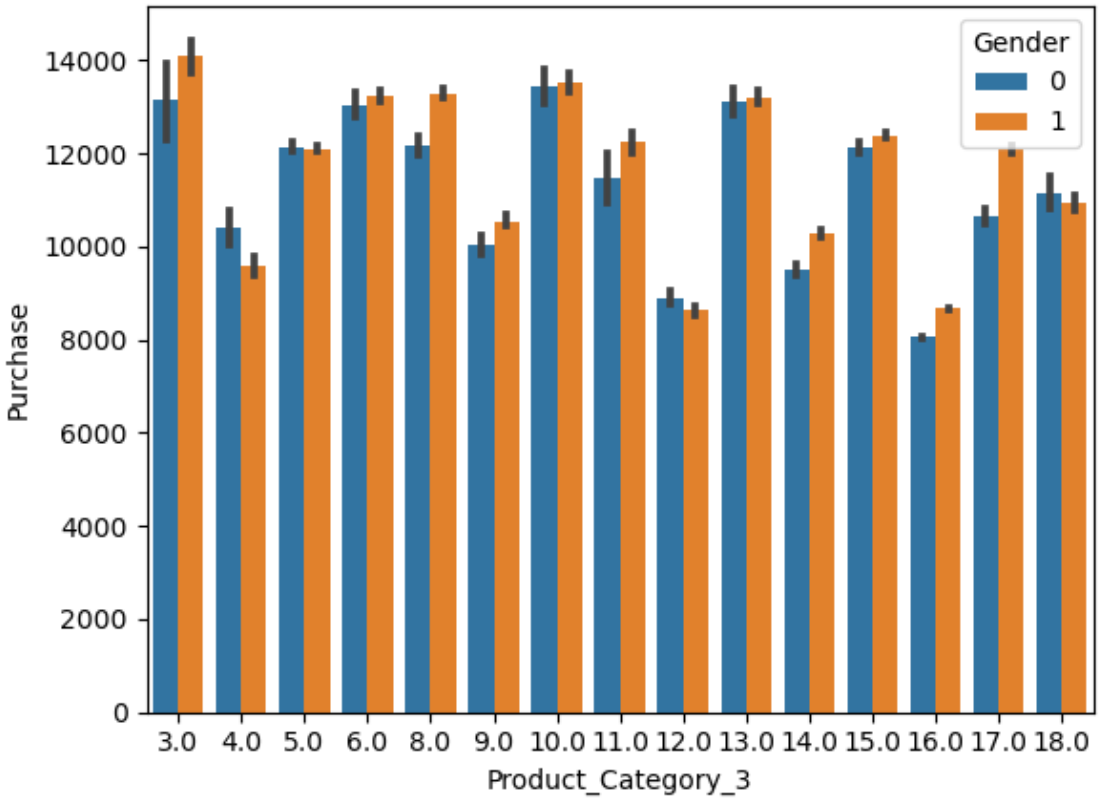
[66]:
```
sns.barplot(x='Product_Category_2',y='Purchase',hue='Gender',data=data)
```

[66]: <Axes: xlabel='Product_Category_2', ylabel='Purchase'>



[68]:
```
sns.barplot(x='Product_Category_3',y='Purchase',hue='Gender',data=data)
```

[68]: <Axes: xlabel='Product_Category_3', ylabel='Purchase'>



[69]:
```
data.head()
```

[69]:

	Product_ID	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase	B	C	B	C
0	P00069042	0	NaN	10	2	0	3	8.0	16.0	8370.0	0	0	0	0
1	P00248942	0	NaN	10	2	0	1	6.0	14.0	15200.0	0	0	0	0
2	P00087842	0	NaN	10	2	0	12	8.0	16.0	1422.0	0	0	0	0
3	P00085442	0	NaN	10	2	0	12	14.0	16.0	1057.0	0	0	0	0
4	P00285442	1	NaN	16	5	0	8	8.0	16.0					

```
4    7969.0  0  1  0  1
```

[74]:
```
data.drop(['Product_ID'],axis=1,inplace=True)
```

[75]:
```
data.head()
```

[75]:
```
   Gender  Age  Occupation  Stay_In_Current_City_Years  Marital_Status  \
0       0  NaN          10                           2               0
1       0  NaN          10                           2               0
2       0  NaN          10                           2               0
3       0  NaN          10                           2               0
4       1  NaN          16                           5               0

   Product_Category_1  Product_Category_2  Product_Category_3  Purchase  B  C  \
0                   3                 8.0                16.0    8370.0  0  0
1                   1                 6.0                14.0   15200.0  0  0
2                  12                 8.0                16.0    1422.0  0  0
3                  12                14.0                16.0    1057.0  0  0
4                   8                 8.0                16.0    7969.0  0  1

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

[103]:
```
## feature scalling
data_test = data[data['Purchase'].isnull()]
```

[104]:
```
data_train = data[~data['Purchase'].isnull()]
```

[105]:
```
x = data_train.drop('Purchase',axis=1)
```

[109]:
```
X_train.drop('Product_ID',axis=1,inplace=True)
X_test.drop('Product_ID',axis=1,inplace=True)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[109], line 1
----> 1 X_train.drop('Product_ID',axis=1,inplace=True)
      2 X_test.drop('Product_ID',axis=1,inplace=True)

NameError: name 'X_train' is not defined
```

[95]:
```
x.head()
```

[95]:
```
   Gender  Age  Occupation  Stay_In_Current_City_Years  Marital_Status  \
0       0  NaN          10                           2               0
1       0  NaN          10                           2               0
2       0  NaN          10                           2               0
3       0  NaN          10                           2               0
4       1  NaN          16                           5               0

   Product_Category_1  Product_Category_2  Product_Category_3  B  C  B  C
0                   3                 8.0                16.0  0  0  0  0
1                   1                 6.0                14.0  0  0  0  0
2                  12                 8.0                16.0  0  0  0  0
3                  12                14.0                16.0  0  0  0  0
4                   8                 8.0                16.0  0  1  0  1
```

[96]:
```
y = data['Purchase']
```

[97]:
```
y
```

[97]:
```
0          8370.0
1         15200.0
2          1422.0
3          1057.0
4          7969.0
           ...
233594        NaN
233595        NaN
233596        NaN
233597        NaN
233598        NaN
Name: Purchase, Length: 783667, dtype: float64
```

[101]:
```
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)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[101], line 2
      1 from sklearn.model_selection import train_test_split
----> 2 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3 ,
      3 random_state=42)

File ~\anaconda3\lib\site-packages\sklearn\model_selection\_split.py:2559, in␣
 ↪train_test_split(test_size, train_size, random_state, shuffle, stratify,␣
 ↪*arrays)
   2556 if n_arrays == 0:
   2557     raise ValueError("At least one array required as input")
-> 2559 arrays = indexable(*arrays)
```

```
   2561 n_samples = _num_samples(arrays[0])
   2562 n_train, n_test = _validate_shuffle_split(
   2563     n_samples, test_size, train_size, default_test_size=0.25
   2564 )

File ~\anaconda3\lib\site-packages\sklearn\utils\validation.py:443, in␣
 ↪indexable(*iterables)
    424 """Make arrays indexable for cross-validation.
    425
    426 Checks consistent length, passes through None, and ensures that␣
 ↪everything
   (…)
    439     sparse matrix, or dataframe) or `None`.
    440 """
    442 result = [_make_indexable(X) for X in iterables]
--> 443 check_consistent_length(*result)
    444 return result

File ~\anaconda3\lib\site-packages\sklearn\utils\validation.py:397, in␣
 ↪check_consistent_length(*arrays)
    395 uniques = np.unique(lengths)
    396 if len(uniques) > 1:
--> 397     raise ValueError(
    398         "Found input variables with inconsistent numbers of samples: %r
    399         % [int(l) for l in lengths]
    400     )

ValueError: Found input variables with inconsistent numbers of samples: [550068 ␣
 ↪783667]
```

[79]:
```
## feature scalling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
Cell In[79], line 2
      1 ## feature scalling
----> 2 from sklearn.preprocessing import standardScaler

ImportError: cannot import name 'standardScaler' from 'sklearn.preprocessing' (C :
 ↪\Users\Vikas\anaconda3\lib\site-packages\sklearn\preprocessing\__init__.py)
```

[ ]:
```

```