In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

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 selected high volume products from last month. The data set also contains customer demographics (age, gender, marital status, city_type, stay_in_current_city), product details (product_id and product category) and Total purchase_amount from last month.

Now, 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.

In [2]:

```
#importing the dataset
df_train=pd.read_csv(r"C:\Users\Admin\Downloads\train.csv")
df_train.head()
```

Out[2]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_C
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]:

```
##  import the test data
df_test=pd.read_csv(r"C:\Users\Admin\Downloads\test.csv")
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_C
0	1000004	P00128942	M	46-50	7	B	2	1	1	
1	1000009	P00113442	M	26-35	17	C	0	0	3	
2	1000010	P00288442	F	36-45	1	B	4+	1	5	
3	1000010	P00145342	F	36-45	1	B	4+	1	4	
4	1000011	P00053842	F	26-35	1	C	1	0	4	

In [4]:

```
##MErge both train and test data
df=df_train.append(df_test)
df.head()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_13064\665716105.py:2: FutureWarning: The frame.append method is dep
recated and will be removed from pandas in a future version. Use pandas.concat instead.
  df=df_train.append(df_test)
```

Out[4]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_C
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 [5]:

```
##Basic
df.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
```

In [6]:
```
df.head()
```

Out[6]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_C
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 [7]:
```
df.describe()
```

Out[7]:

	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 [8]:
```
df.drop(['User_ID'],axis=1,inplace=True)
```

In [9]:
```
df.head()
```

Out[9]:

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

In [11]:
```
pd.get_dummies(df['Gender'],drop_first=1)
```

Out[11]:

	M
0	0
1	0
2	0
3	0
4	1
...	...
233594	0
233595	0
233596	0
233597	0
233598	0

783667 rows × 1 columns

In [10]:

```
pd.get_dummies(df["Gender"])
```

Out[10]:

	F	M
0	1	0
1	1	0
2	1	0
3	1	0
4	0	1
...	...	...
233594	1	0
233595	1	0
233596	1	0
233597	1	0
233598	1	0

783667 rows × 2 columns

In [12]:

```
##HAndling categorical feature Gender
df['Gender']=df['Gender'].map({'F':0,'M':1})
df.head()
```

Out[12]:

	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2
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 [13]:

```
## Handle categorical feature Age
df['Age'].unique()
```

Out[13]:

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

In [14]:

```
pd.get_dummies(df['Age'])
```

Out[14]:

	0-17	18-25	26-35	36-45	46-50	51-55	55+
0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0
2	1	0	0	0	0	0	0
3	1	0	0	0	0	0	0
4	0	0	0	0	0	0	1
...	...	...	...	...	...	...	...
233594	0	0	1	0	0	0	0
233595	0	0	1	0	0	0	0
233596	0	0	1	0	0	0	0
233597	0	0	0	0	1	0	0
233598	0	0	0	0	1	0	0

783667 rows × 7 columns

In [15]:

```
#.get_dummies
pd.get_dummies(df['Age'],drop_first=True)
```

Out[15]:

	18-25	26-35	36-45	46-50	51-55	55+
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	1
...	...	...	...	...	...	...
233594	0	1	0	0	0	0
233595	0	1	0	0	0	0
233596	0	1	0	0	0	0
233597	0	0	0	1	0	0
233598	0	0	0	1	0	0

783667 rows × 6 columns

In [16]:

```
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 [18]:

```
##second technqiue
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'species'.
df['Age']= label_encoder.fit_transform(df['Age'])

df['Age'].unique()
```

Out[18]:

```
array([0, 6, 2, 4, 5, 3, 1], dtype=int64)
```

In [19]:

```
df.head()
```

Out[19]:

	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2
0	P00069042	0	0	10	A	2	0	3	NaN
1	P00248942	0	0	10	A	2	0	1	6.0
2	P00087842	0	0	10	A	2	0	12	NaN
3	P00085442	0	0	10	A	2	0	12	14.0
4	P00285442	1	6	16	C	4+	0	8	NaN

In [24]:

```
df_city.head()
```

Out[24]:

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

In [25]:
```
df=pd.concat([df,df_city],axis=1)
df.head()
```

Out[25]:

	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2
0	P00069042	0	0	10	A	2	0	3	NaN
1	P00248942	0	0	10	A	2	0	1	6.0
2	P00087842	0	0	10	A	2	0	12	NaN
3	P00085442	0	0	10	A	2	0	12	14.0
4	P00285442	1	6	16	C	4+	0	8	NaN

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

In [48]:
```
df.head()
```

Out[48]:

	Product_ID	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Categ
0	P00069042	0	0	10	2	0	3	NaN	
1	P00248942	0	0	10	2	0	1	6.0	
2	P00087842	0	0	10	2	0	12	NaN	
3	P00085442	0	0	10	2	0	12	14.0	
4	P00285442	1	6	16	4+	0	8	NaN	

In [50]:
```
## Missing Values
df.isnull().sum()
```

Out[50]:
```
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 [51]:
```
## Focus on replacing missing values
df['Product_Category_2'].unique()
```

Out[51]:
```
array([nan,  6., 14.,  2.,  8., 15., 16., 11.,  5.,  3.,  4., 12.,  9.,
       10., 17., 13.,  7., 18.])
```

In [52]:
```
df['Product_Category_2'].value_counts()
```

Out[52]:
```
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: Product_Category_2, dtype: int64
```

In [53]:
```
df['Product_Category_2'].mode()[0]
```

Out[53]:
```
8.0
```

In [54]:
```
## Replace the missing values with mode
df['Product_Category_2']=df['Product_Category_2'].fillna(df['Product_Category_2'].mode()[0])
```

In [55]:
```
## Product_category 3 replace missing values
df['Product_Category_3'].unique()
```

Out[55]:
```
array([nan, 14., 17.,  5.,  4., 16., 15.,  8.,  9., 13.,  6., 12.,  3.,
       18., 11., 10.])
```

In [56]:
```
df['Product_Category_3'].value_counts()
```

Out[56]:
```
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: Product_Category_3, dtype: int64
```

In [57]:
```
## Replace the missing values with mode
df['Product_Category_3']=df['Product_Category_3'].fillna(df['Product_Category_3'].mode()[0])
```

In [58]:
```
df.head()
```

Out[58]:

	Product_ID	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Categ
0	P00069042	0	0	10	2	0	3	8.0	
1	P00248942	0	0	10	2	0	1	6.0	
2	P00087842	0	0	10	2	0	12	8.0	
3	P00085442	0	0	10	2	0	12	14.0	
4	P00285442	1	6	16	4+	0	8	8.0	

In [59]:
```
df.shape
```

Out[59]:
```
(783667, 12)
```

In [60]:
```
df['Stay_In_Current_City_Years'].unique()
```

Out[60]:
```
array(['2', '4+', '3', '1', '0'], dtype=object)
```

In [61]:
```
df['Stay_In_Current_City_Years']=df['Stay_In_Current_City_Years'].str.replace('+','')
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_13064\2063355665.py:1: FutureWarning: The default value of regex wi
ll change from True to False in a future version. In addition, single character regular expressions will *not*
be treated as literal strings when regex=True.
  df['Stay_In_Current_City_Years']=df['Stay_In_Current_City_Years'].str.replace('+','')
```

In [62]:
```
df.head()
```

Out[62]:

	Product_ID	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Categ
0	P00069042	0	0	10	2	0	3	8.0	
1	P00248942	0	0	10	2	0	1	6.0	
2	P00087842	0	0	10	2	0	12	8.0	
3	P00085442	0	0	10	2	0	12	14.0	
4	P00285442	1	6	16	4	0	8	8.0	

In [63]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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  uint8  
 11  C                           783667 non-null  uint8  
dtypes: float64(3), int64(5), object(2), uint8(2)
memory usage: 67.3+ MB
```

In [64]:

```
##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'>
Int64Index: 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  uint8  
 11  C                           783667 non-null  uint8  
dtypes: float64(3), int32(1), int64(5), object(1), uint8(2)
memory usage: 64.3+ MB
```

In [65]:

```
df['B']=df['B'].astype(int)
df['C']=df['C'].astype(int)
```

In [66]:

```
df.info()
```

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

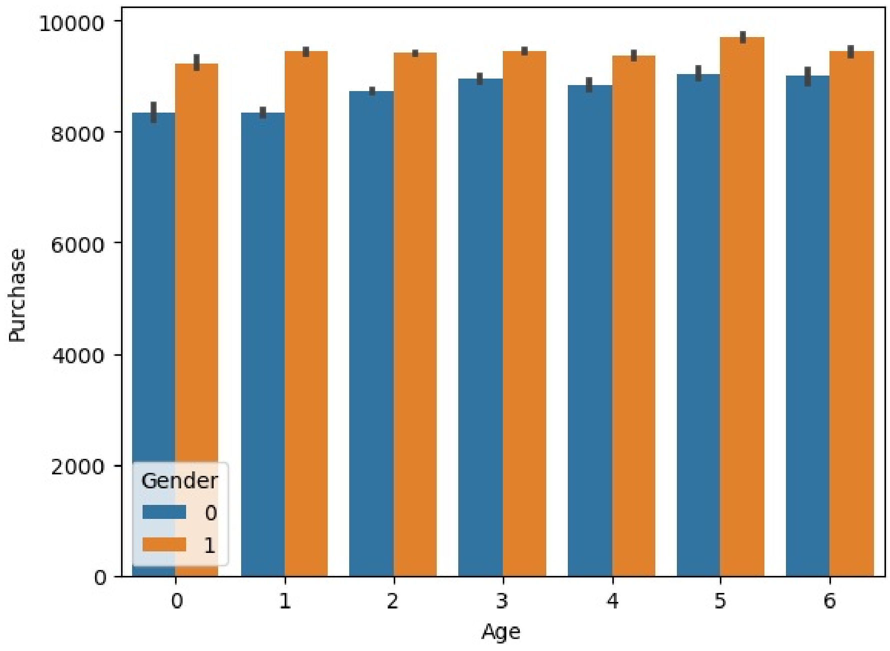
In [67]:

```
##Visualisation Age vs Purchased
sns.barplot('Age','Purchase',hue='Gender',data=df)
```

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

Out[67]:

```
<AxesSubplot:xlabel='Age', ylabel='Purchase'>
```



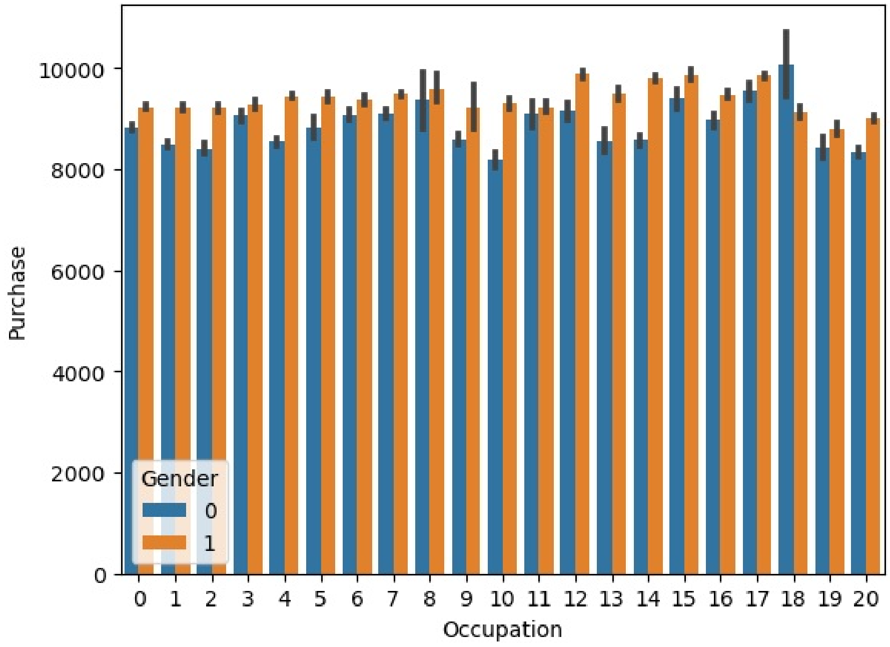
In [68]:

```
## Visualization of Purchase with occupation
sns.barplot('Occupation','Purchase',hue='Gender',data=df)
```

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

Out[68]:

```
<AxesSubplot:xlabel='Occupation', ylabel='Purchase'>
```



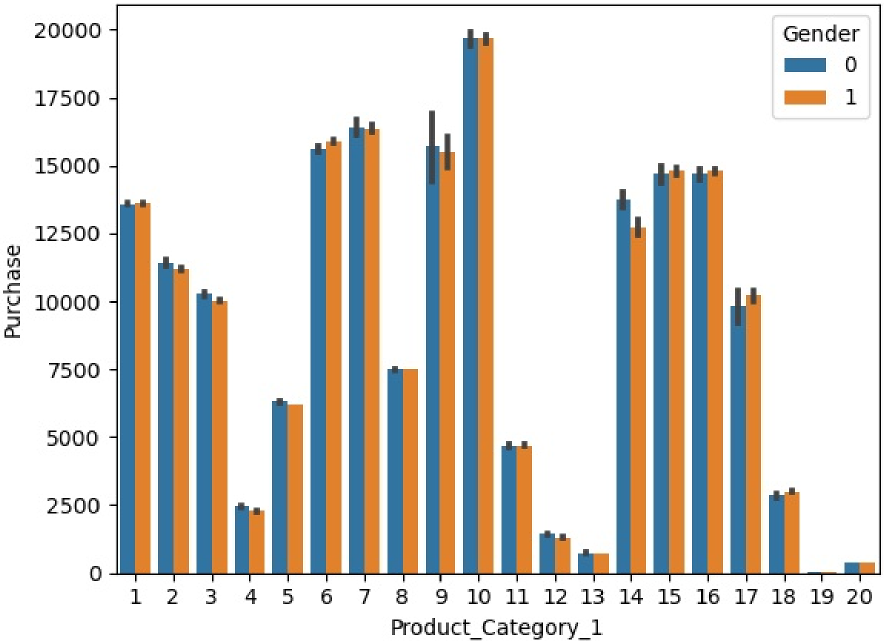
In [69]:

```
sns.barplot('Product_Category_1','Purchase',hue='Gender',data=df)
```

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

Out[69]:

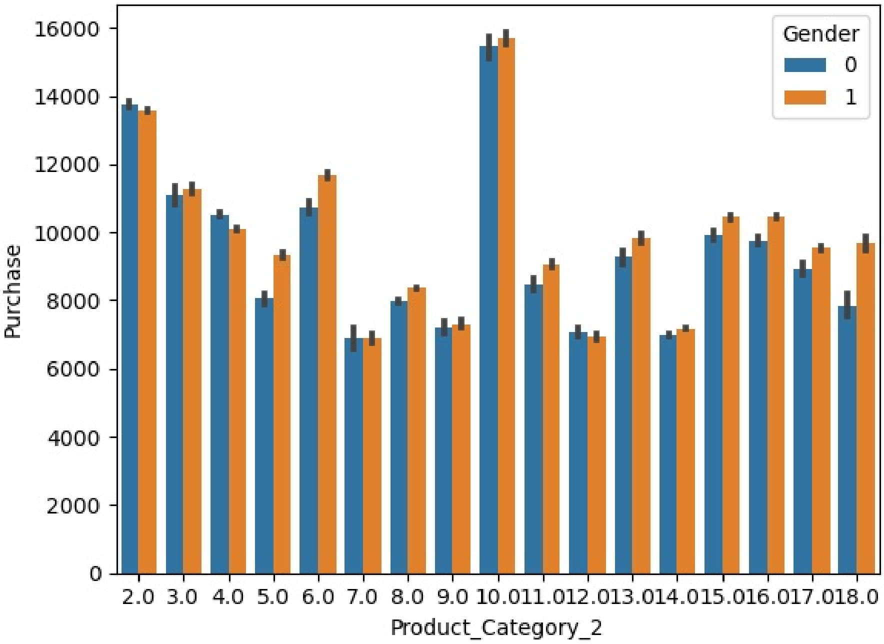
```
<AxesSubplot:xlabel='Product_Category_1', ylabel='Purchase'>
```



In [70]:
```
sns.barplot('Product_Category_2','Purchase',hue='Gender',data=df)
```

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

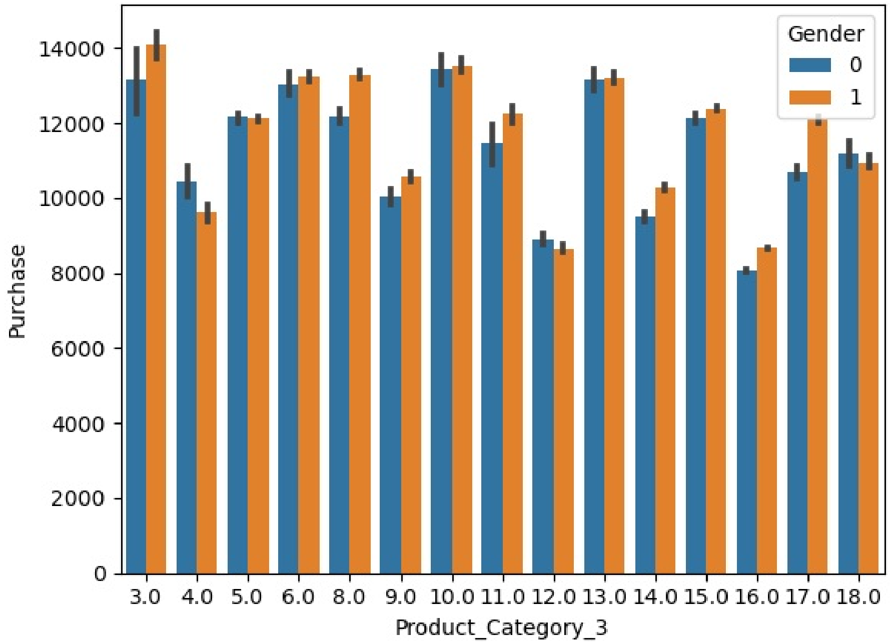
Out[70]:
```
<AxesSubplot:xlabel='Product_Category_2', ylabel='Purchase'>
```



In [71]:
```
sns.barplot('Product_Category_3','Purchase',hue='Gender',data=df)
```

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

Out[71]:
```
<AxesSubplot:xlabel='Product_Category_3', ylabel='Purchase'>
```



In [72]:
```
df.head()
```

Out[72]:

	Product_ID	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Categ
0	P00069042	0	0	10	2	0	3	8.0	
1	P00248942	0	0	10	2	0	1	6.0	
2	P00087842	0	0	10	2	0	12	8.0	
3	P00085442	0	0	10	2	0	12	14.0	
4	P00285442	1	6	16	4	0	8	8.0	

In [73]:
```
##Feature Scaling
df_test=df[df['Purchase'].isnull()]
```

In [74]:
```
df_train=df[~df['Purchase'].isnull()]
```

In [88]:
```
X=df_train.drop('Purchase',axis=1)
```

In [89]:
```
X.head()
```

Out[89]:

	Product_ID	Gender	Age	Occupation	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Categ
0	P00069042	0	0	10	2	0	3	8.0	
1	P00248942	0	0	10	2	0	1	6.0	
2	P00087842	0	0	10	2	0	12	8.0	
3	P00085442	0	0	10	2	0	12	14.0	
4	P00285442	1	6	16	4	0	8	8.0	

In [90]:
```
y = df_train['Purchase']
```

In [91]:
```
y
```

```
Out[91]: 0          8370.0
         1         15200.0
         2          1422.0
         3          1057.0
         4          7969.0
                     ...
         550063      368.0
         550064      371.0
         550065      137.0
         550066      365.0
         550067      490.0
         Name: Purchase, Length: 550068, dtype: float64
```

```
In [77]: X.shape
```

```
Out[77]: (550068, 11)
```

```
In [92]: y.shape
```

```
Out[92]: (550068,)
```

```
In [ ]: y
```

```
In [93]: 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 [94]: X_train.drop('Product_ID',axis=1,inplace=True)
         X_test.drop('Product_ID',axis=1,inplace=True)
```

```
In [95]: ## feature Scaling
         from sklearn.preprocessing import StandardScaler
         sc=StandardScaler()
         X_train=sc.fit_transform(X_train)
         X_test=sc.transform(X_test)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js