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

In [2]:
```
bf = pd.read_csv(r"D:\Python\blackfriday.csv")
bf
```

Out[2]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Yea
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	
...	...	...	...	...	...	...	
550063	1006033	P00372445	M	51-55	13	B	
550064	1006035	P00375436	F	26-35	1	C	
550065	1006036	P00375436	F	26-35	15	B	
550066	1006038	P00375436	F	55+	1	C	
550067	1006039	P00371644	F	46-50	0	B	

550068 rows × 12 columns

In [3]:
```
bf.describe()
```

Out[3]:

	User_ID	Occupation	Marital_Status	Product_Category_1	Product_Category_2	Pr
count	5.500680e+05	550068.000000	550068.000000	550068.000000	376430.000000	
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	

In [4]:
```
bf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   User_ID                     550068 non-null  int64  
 1   Product_ID                  550068 non-null  object 
 2   Gender                      550068 non-null  object 
 3   Age                         550068 non-null  object 
 4   Occupation                  550068 non-null  int64  
 5   City_Category               550068 non-null  object 
 6   Stay_In_Current_City_Years  550068 non-null  object 
 7   Marital_Status              550068 non-null  int64  
 8   Product_Category_1          550068 non-null  int64  
 9   Product_Category_2          376430 non-null  float64
 10  Product_Category_3          166821 non-null  float64
 11  Purchase                    550068 non-null  int64  
dtypes: float64(2), int64(5), object(5)
memory usage: 50.4+ MB
```

In [ ]:
```
```

In [5]:
```
bf.isnull().sum()
```

Out[5]:
```
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 [6]:
```
bf["Product_Category_2"].isnull().sum()
```

Out[6]:
```
173638
```

In [7]:
```
bf["Product_Category_2"].mode()
```

Out[7]:
```
0    8.0
Name: Product_Category_2, dtype: float64
```

In [8]:
```
bf["Product_Category_2"].unique()
```

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

In [9]:
```
bf["Product_Category_2"].info()
```

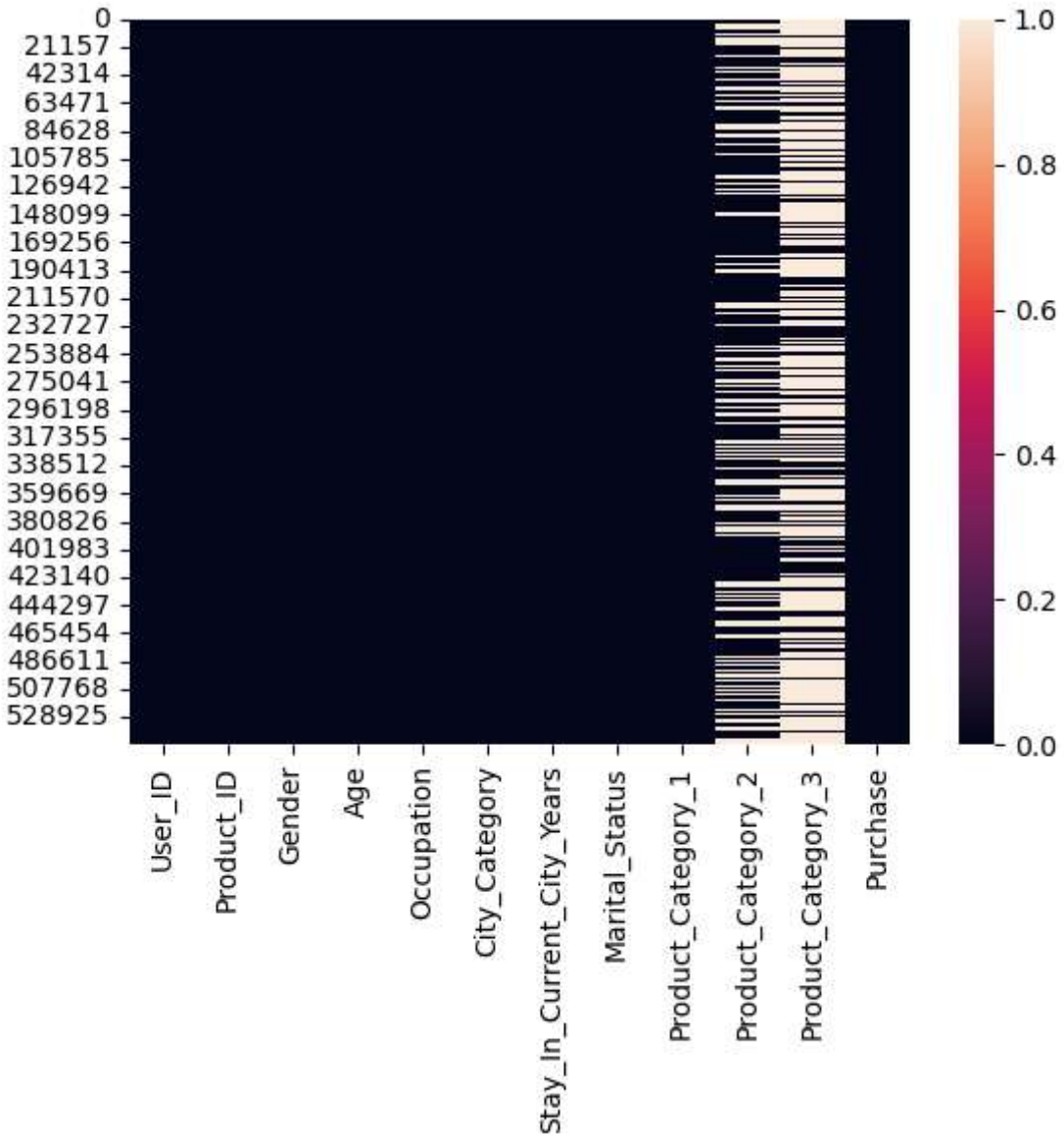
```
<class 'pandas.core.series.Series'>
RangeIndex: 550068 entries, 0 to 550067
Series name: Product_Category_2
Non-Null Count   Dtype
--------------   -----
376430 non-null  float64
dtypes: float64(1)
memory usage: 4.2 MB
```

In [10]:
```
bf.groupby('Product_Category_2').size()
```

Out[10]:
```
Product_Category_2
2.0     49217
3.0      2884
4.0     25677
5.0     26235
6.0     16466
7.0       626
8.0     64088
9.0      5693
10.0     3043
11.0    14134
12.0     5528
13.0    10531
14.0    55108
15.0    37855
16.0    43255
17.0    13320
18.0     2770
dtype: int64
```

In [11]:
```
sns.heatmap(bf.isnull())
```

Out[11]: <Axes: >



Handling Missing in Product_Category_2

In [12]:
```
bf["Product_Category_2"].unique()
```

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

In [13]:
```
bf["Product_Category_2"] = bf["Product_Category_2"].fillna(bf["Product_Categor
```

In [14]:
```
bf["Product_Category_2"].isnull().sum()
```

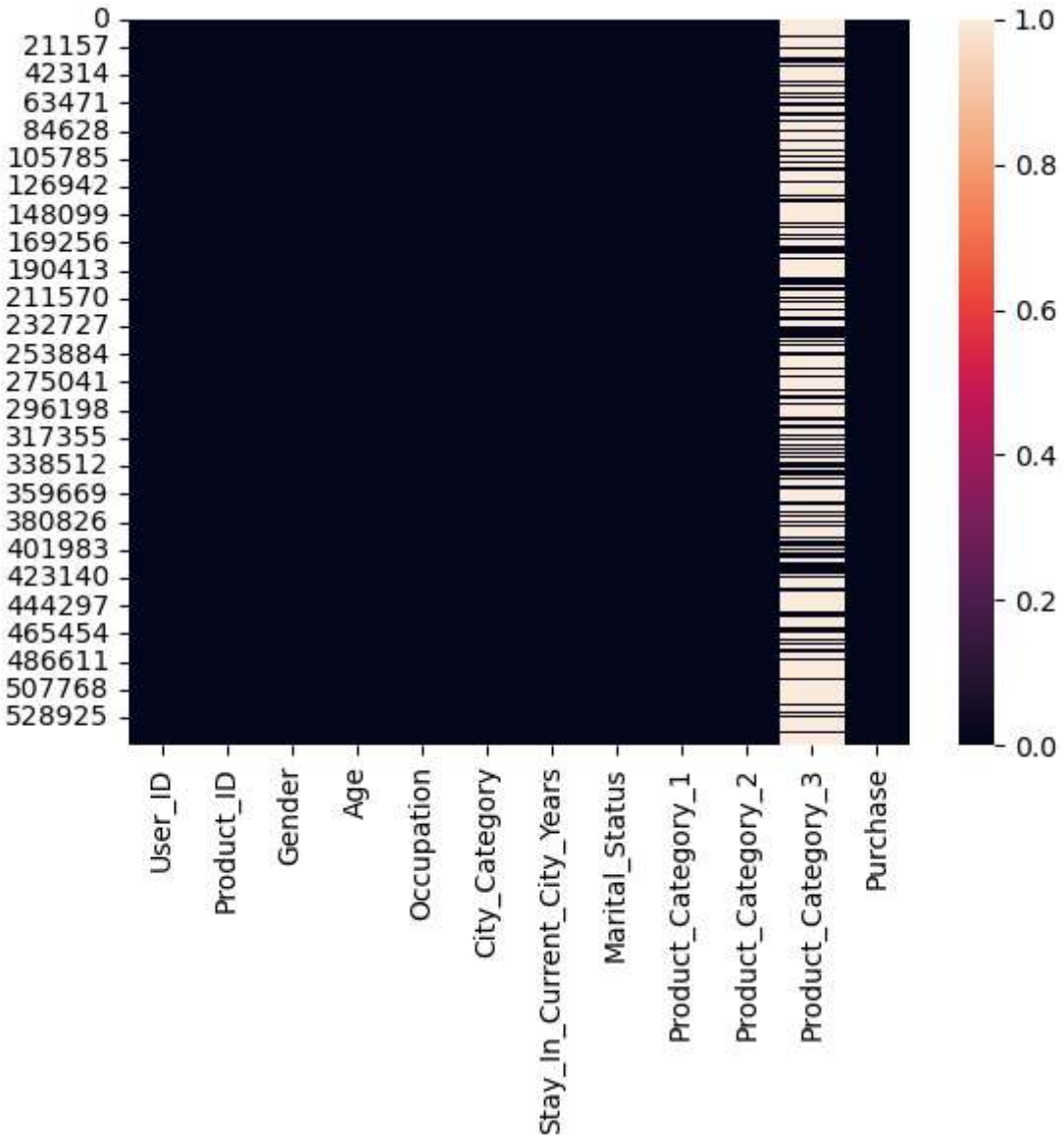
Out[14]: 0

In [15]:
```
bf["Product_Category_2"].unique()
```

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

In [16]:
```
sns.heatmap(bf.isnull())
```

Out[16]:
```
<Axes: >
```



Handling Missing in Product_Category_3

In [17]:
```
bf["Product_Category_3_fill"] = bf["Product_Category_3"].ffill
```

In [18]:
```
bf["Product_Category_3_fill"].isnull().sum()
```

Out[18]:
```
0
```

null value = 0

but NaN value is still there inside this column so we skip this step to fill null value because it will be very long step we have to replace nan with mode to make data clean and relevent so in next step i directly do this with mode in next step

we use mode to fill the categorical column i make special column to check this

In [19]:
```
bf["Product_Category_3_fill"].unique()
```

Out[19]:
```
array([<bound method Series.ffill of 0          NaN
       1         14.0
       2          NaN
       3          NaN
       4          NaN
                 ...
       550063     NaN
       550064     NaN
       550065     NaN
       550066     NaN
       550067     NaN
       Name: Product_Category_3, Length: 550068, dtype: float64>],
      dtype=object)
```

we use mode to fill null value

In [20]:
```
bf["Product_Category_3"].unique()
```

Out[20]:
```
array([nan, 14., 17.,  5.,  4., 16., 15.,  8.,  9., 13.,  6., 12.,  3.,
       18., 11., 10.])
```

In [21]:
```
bf["Product_Category_3"] = bf["Product_Category_3"].fillna(bf["Product_Categor
```

In [22]:
```
bf["Product_Category_3"].unique()
```

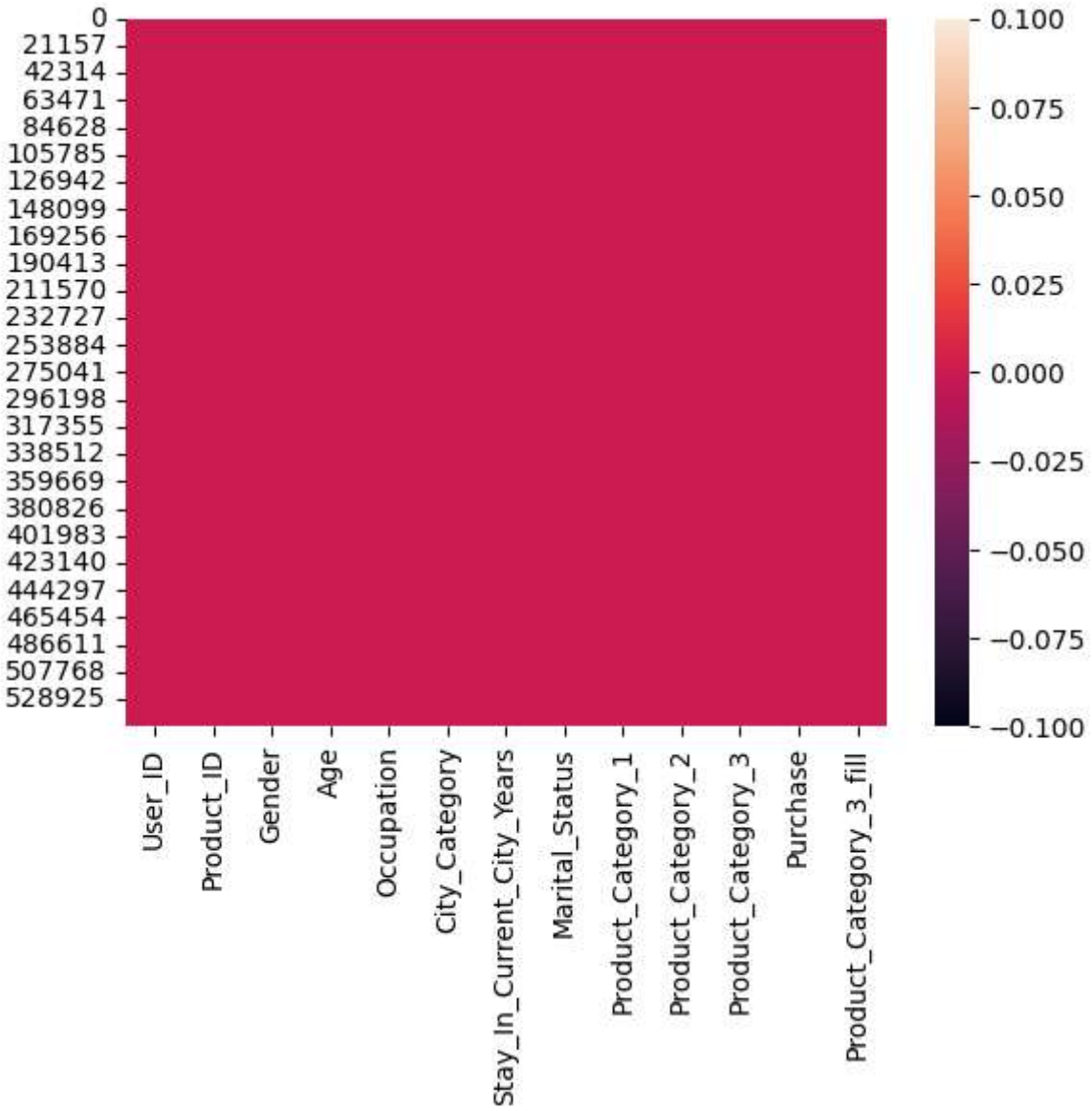
Out[22]:
```
array([16., 14., 17.,  5.,  4., 15.,  8.,  9., 13.,  6., 12.,  3., 18.,
       11., 10.])
```

In [23]:
```
bf["Product_Category_3"].isnull().sum()
```

Out[23]:
```
0
```

In [24]:
```
sns.heatmap(bf.isnull())
```

Out[24]:
```
<Axes: >
```



Now after dealing with null value we are going to drop unusual column

In [25]:
```
bf.head()
```

Out[25]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	M
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 this data we will work according product_id

product_id will use for which product has more no sale

there is no use user id we are going to drop user_id

In [26]:
```
bf.drop("User_ID",axis = 1,inplace = True)
```

now convert gender in numerical data

In [27]:
```
bf["Gender"]=bf["Gender"].map({"F":0,"M":1})
```

In [28]:
```
bf["Gender"].unique()
```

Out[28]:
```
array([0, 1], dtype=int64)
```

now convert age column in normal interval and distribution

according to analysis of buying behaviour

In [29]:
```
bf["Age"].unique()
```

Out[29]:
```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

In [30]:
```
bf["Age"] = bf["Age"].map({"0-17":1,"18-25":2,"26-35":3,"36-45":4,"46-50":5,"5
```

In [31]:
```
bf["Age"].unique()
```

Out[31]:
```
array([1, 7, 3, 5, 6, 4, 2], dtype=int64)
```

In [32]:
```
bf.head()
```

Out[32]:

	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Stat
0	P00069042	0	1	10	A	2	
1	P00248942	0	1	10	A	2	
2	P00087842	0	1	10	A	2	
3	P00085442	0	1	10	A	2	
4	P00285442	1	7	16	C	4+	

now city_category change into numerical data

In [33]:
```
bf["City_Category"].unique()
```

Out[33]:
```
array(['A', 'C', 'B'], dtype=object)
```

In [34]:
```
bf_city=pd.get_dummies(bf['City_Category'],drop_first=True)
```

now we are going to merge bf_city into bf dataframe

In [35]:
```
df=pd.concat([bf,bf_city],axis=1)
df.head()
```

Out[35]:

	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Stat
0	P00069042	0	1	10	A	2	
1	P00248942	0	1	10	A	2	
2	P00087842	0	1	10	A	2	
3	P00085442	0	1	10	A	2	
4	P00285442	1	7	16	C	4+	

now we are going to clean Stay_In_Current_City_Years column

In [36]:
```
df["Stay_In_Current_City_Years"].unique()
```

Out[36]:
```
array(['2', '4+', '3', '1', '0'], dtype=object)
```

no we remove all the category inside the string from Stay_In_Current_City_Years

In [37]:
```
df['Stay_In_Current_City_Years']=df['Stay_In_Current_City_Years'].str.replace(

```

```
C:\Users\Subham Ranjan\AppData\Local\Temp\ipykernel_16252\3061240698.py:1: Fu
tureWarning: The default value of regex will change from True to False in a f
uture version. In addition, single character regular expressions will *not* b
e treated as literal strings when regex=True.
  df['Stay_In_Current_City_Years']=df['Stay_In_Current_City_Years'].str.repla
ce('+','').astype(int)
```

In [38]:
```
df["Stay_In_Current_City_Years"].unique()
```

Out[38]:
```
array([2, 4, 3, 1, 0])
```

In [39]:
```
df["Stay_In_Current_City_Years"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 550068 entries, 0 to 550067
Series name: Stay_In_Current_City_Years
Non-Null Count   Dtype
--------------   -----
550068 non-null  int32
dtypes: int32(1)
memory usage: 2.1 MB
```

In [40]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 14 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   City_Category               550068 non-null  object
 5   Stay_In_Current_City_Years  550068 non-null  int32
 6   Marital_Status              550068 non-null  int64
 7   Product_Category_1          550068 non-null  int64
 8   Product_Category_2          550068 non-null  float64
 9   Product_Category_3          550068 non-null  float64
 10  Purchase                    550068 non-null  int64
 11  Product_Category_3_fill     550068 non-null  object
 12  B                           550068 non-null  uint8
 13  C                           550068 non-null  uint8
dtypes: float64(2), int32(1), int64(6), object(3), uint8(2)
memory usage: 49.3+ MB
```

convert B and C into int

In [41]:
```
df["B"] = df["B"].astype(int)
df["C"] = df["C"].astype(int)
```

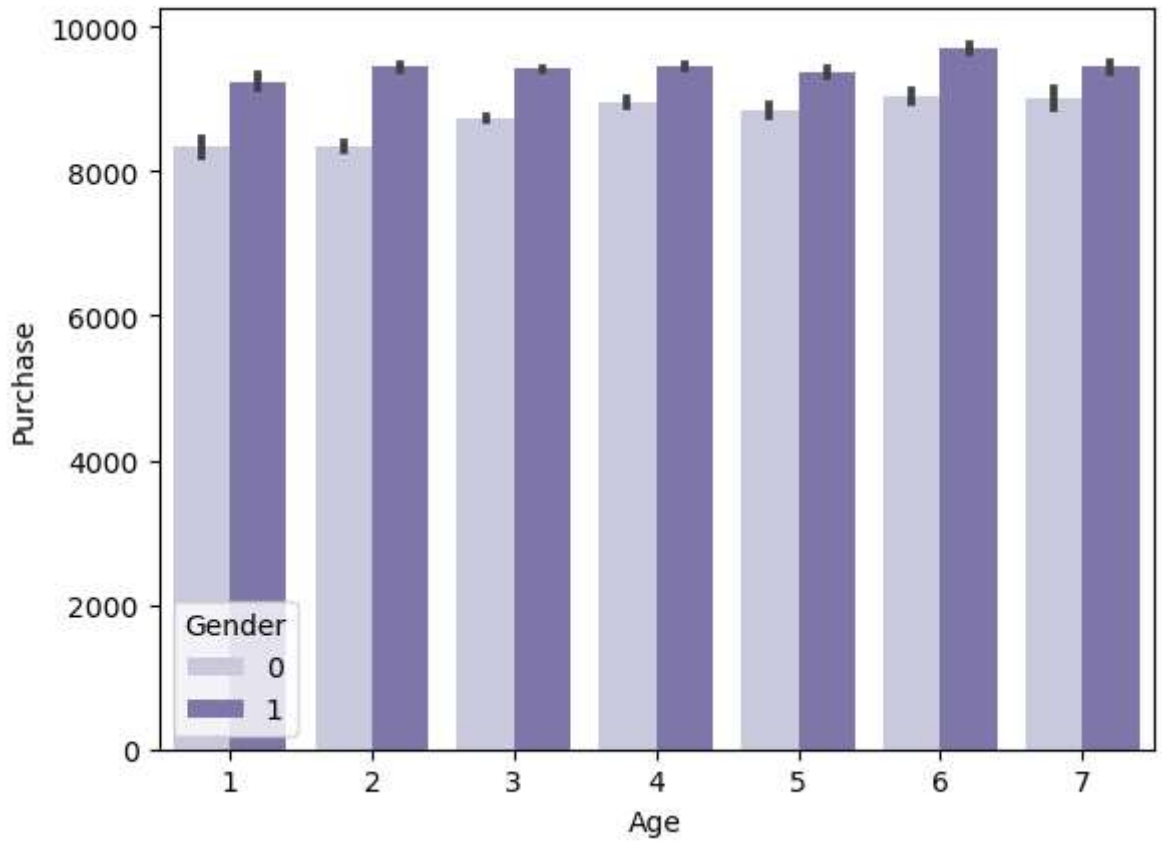
In [42]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 14 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   City_Category               550068 non-null  object 
 5   Stay_In_Current_City_Years  550068 non-null  int32  
 6   Marital_Status              550068 non-null  int64  
 7   Product_Category_1          550068 non-null  int64  
 8   Product_Category_2          550068 non-null  float64
 9   Product_Category_3          550068 non-null  float64
 10  Purchase                    550068 non-null  int64  
 11  Product_Category_3_fill     550068 non-null  object 
 12  B                           550068 non-null  int32  
 13  C                           550068 non-null  int32  
dtypes: float64(2), int32(3), int64(6), object(3)
memory usage: 52.5+ MB
```

1 Purchases are done more by men than women with respect to all age criteria.

In [43]:
```
sns.barplot(x = "Age",y = "Purchase",hue = "Gender",data = df,palette = "Purpl
```

Out[43]:
```
<Axes: xlabel='Age', ylabel='Purchase'>
```



In [44]:
```
df.head()
```

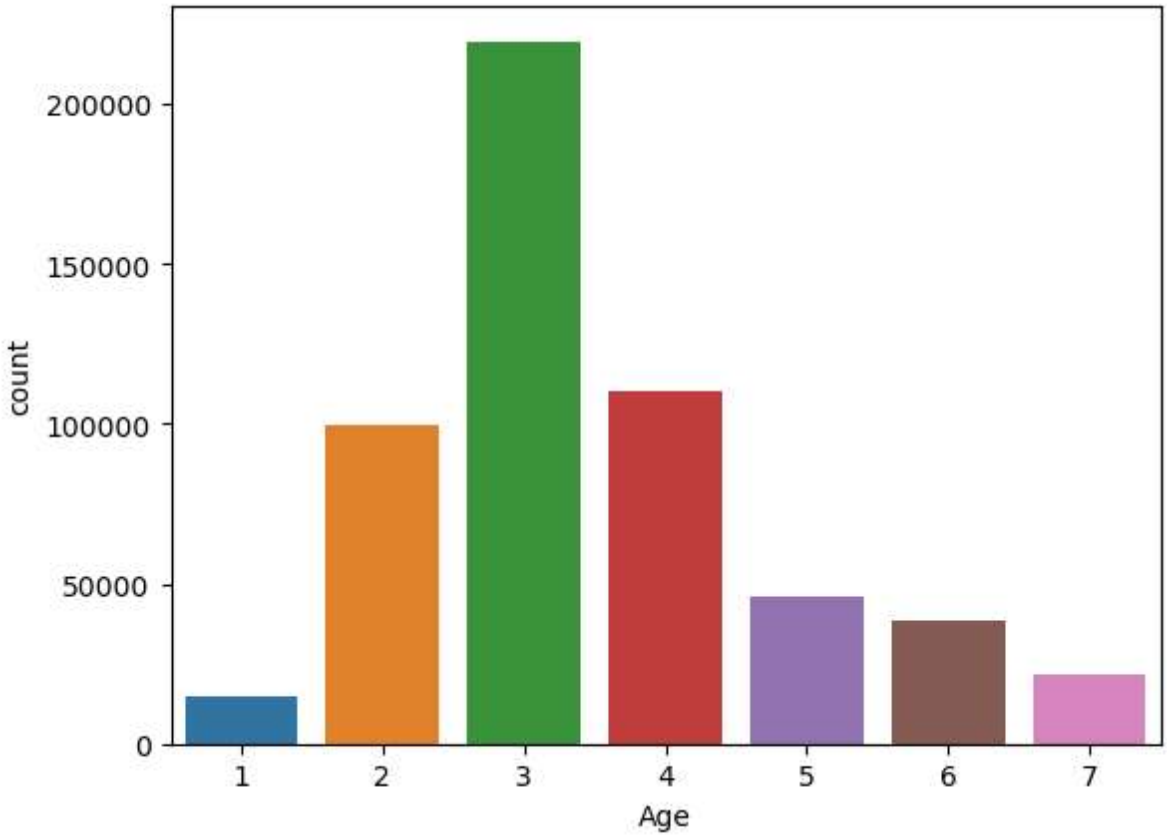
Out[44]:

	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Stat
0	P00069042	0	1	10	A	2	
1	P00248942	0	1	10	A	2	
2	P00087842	0	1	10	A	2	
3	P00085442	0	1	10	A	2	
4	P00285442	1	7	16	C	4	

2 Which age category has highest no of buyers?

In [45]:
```
sns.countplot(x = "Age",data = bf)#26-35 age category people has highest no bu
```

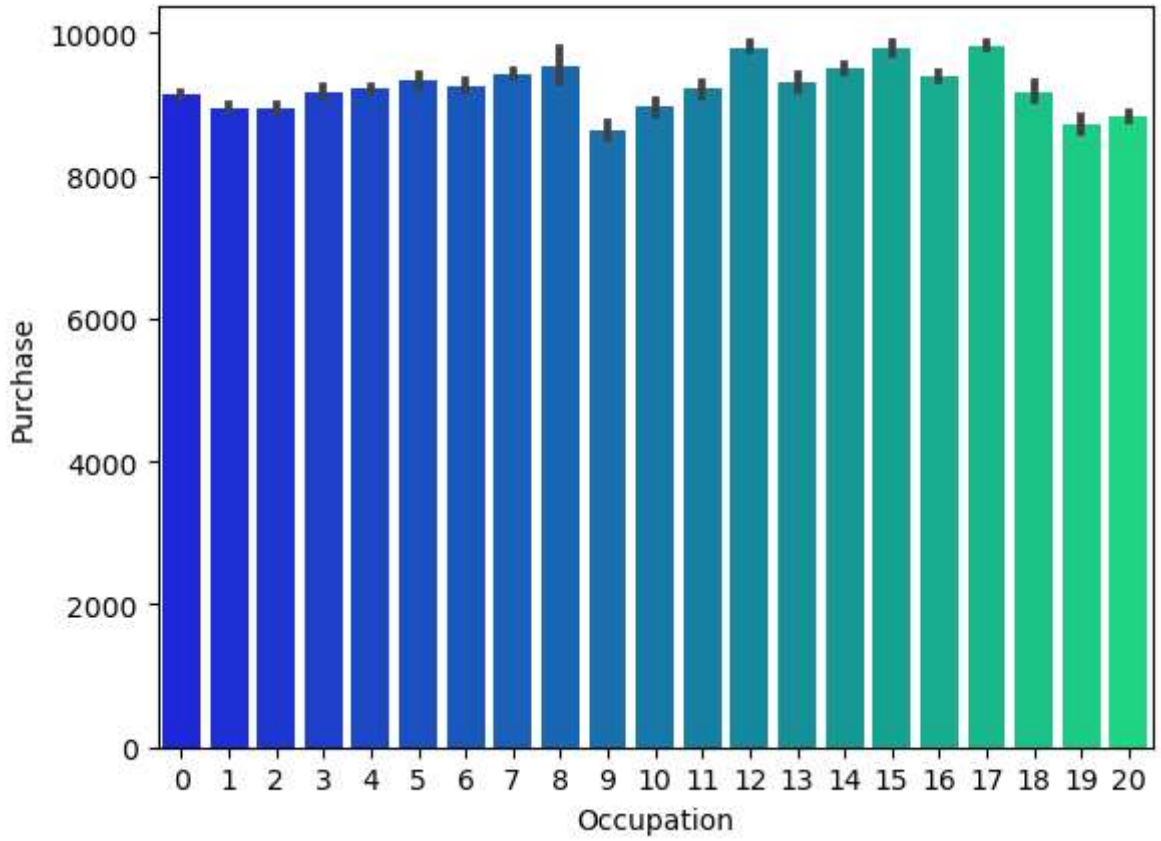
Out[45]:
```
<Axes: xlabel='Age', ylabel='count'>
```



3 we see the purchases had made according to the occuption?

In [47]:
```
sns.barplot(x = "Occupation",y ="Purchase",data = df ,palette = "winter")
```

Out[47]:
```
<Axes: xlabel='Occupation', ylabel='Purchase'>
```

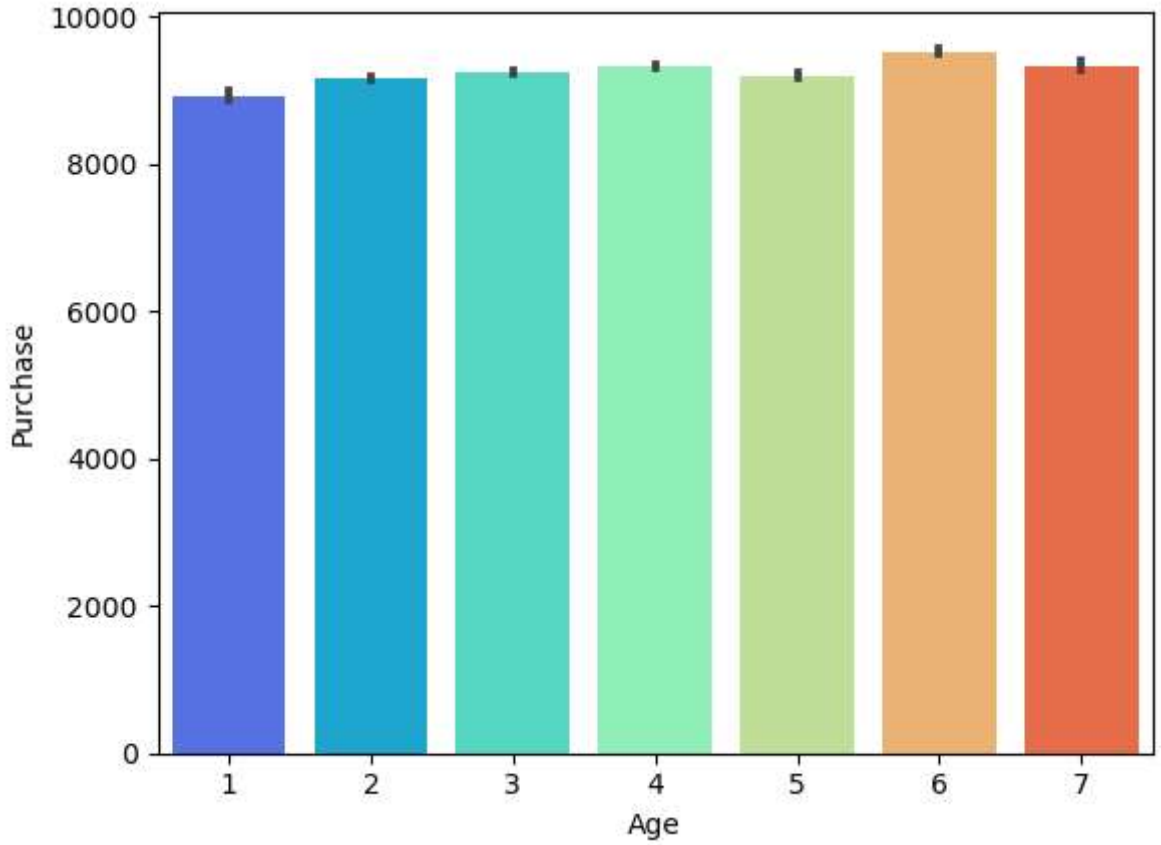


4 Which Age category has done highest no purchases?

According to this data 51 - 55 age people has done highest no of purchases after the overall analysis older people has done more purchases

In [48]:
```
#"0-17":1,"18-25":2,"26-35":3,"36-45":4,"46-50":5,"51-55":6,"55+":7
sns.barplot(x  = "Age",y ="Purchase" ,data = df,palette = "rainbow")
```

Out[48]: <Axes: xlabel='Age', ylabel='Purchase'>

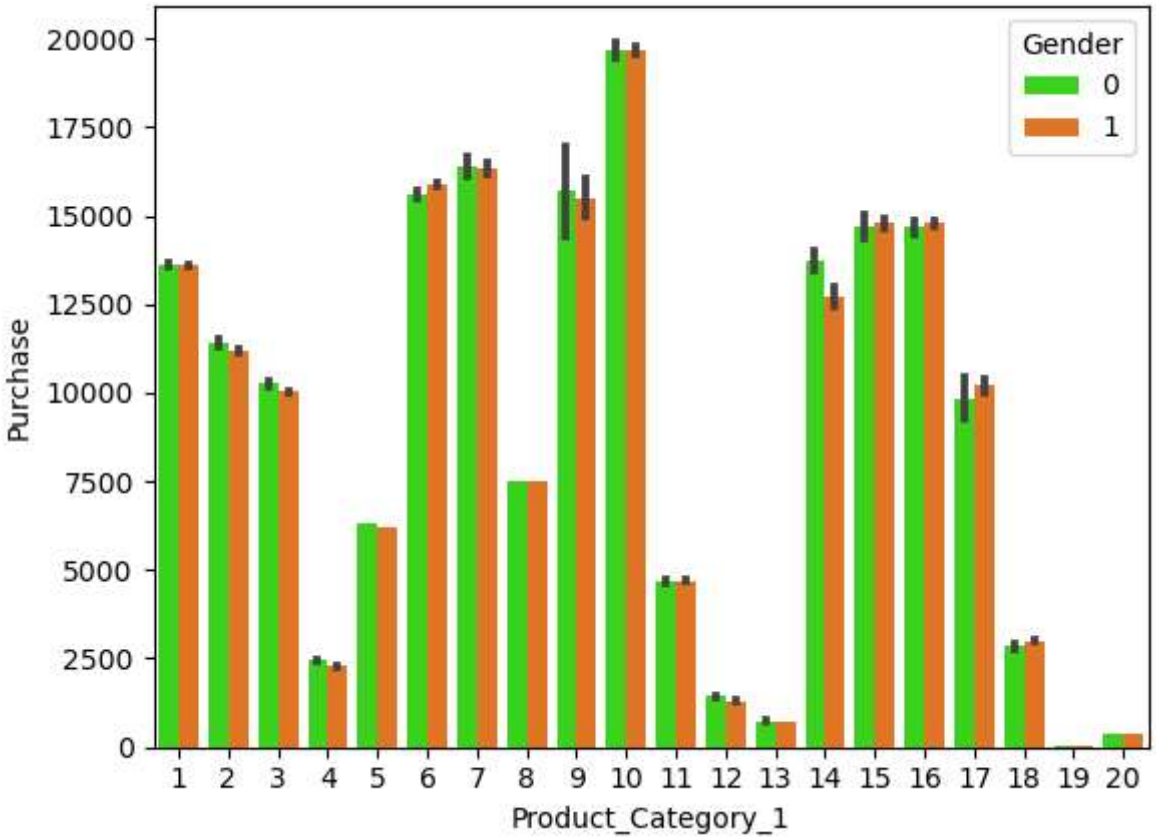


In [ ]:

5 how many purchases are made by people in Product_Category_1?

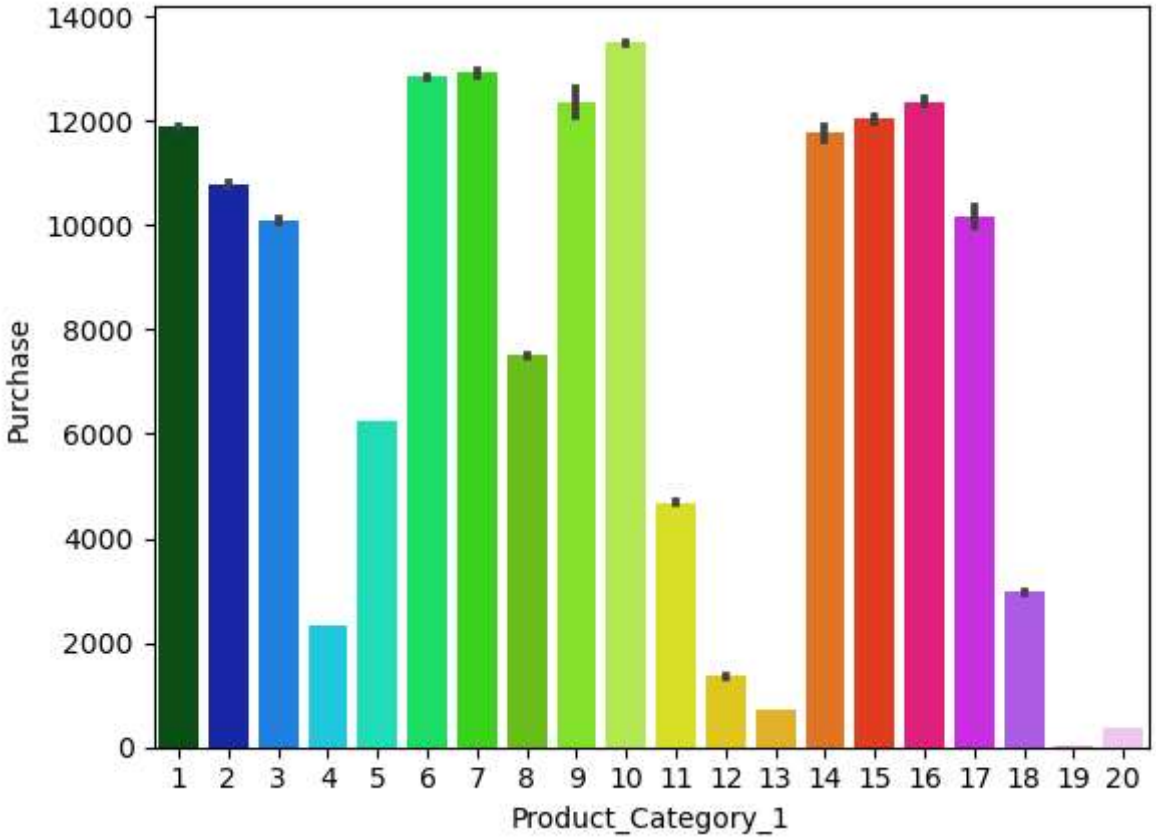
In [49]:
```
sns.barplot(x = "Product_Category_1",y  = "Purchase",hue = "Gender",data = df,
```

Out[49]:
```
<Axes: xlabel='Product_Category_1', ylabel='Purchase'>
```



In [77]:
```
sns.barplot(x = "Product_Category_1",y  = "Purchase",data = df,palette = "gist
```

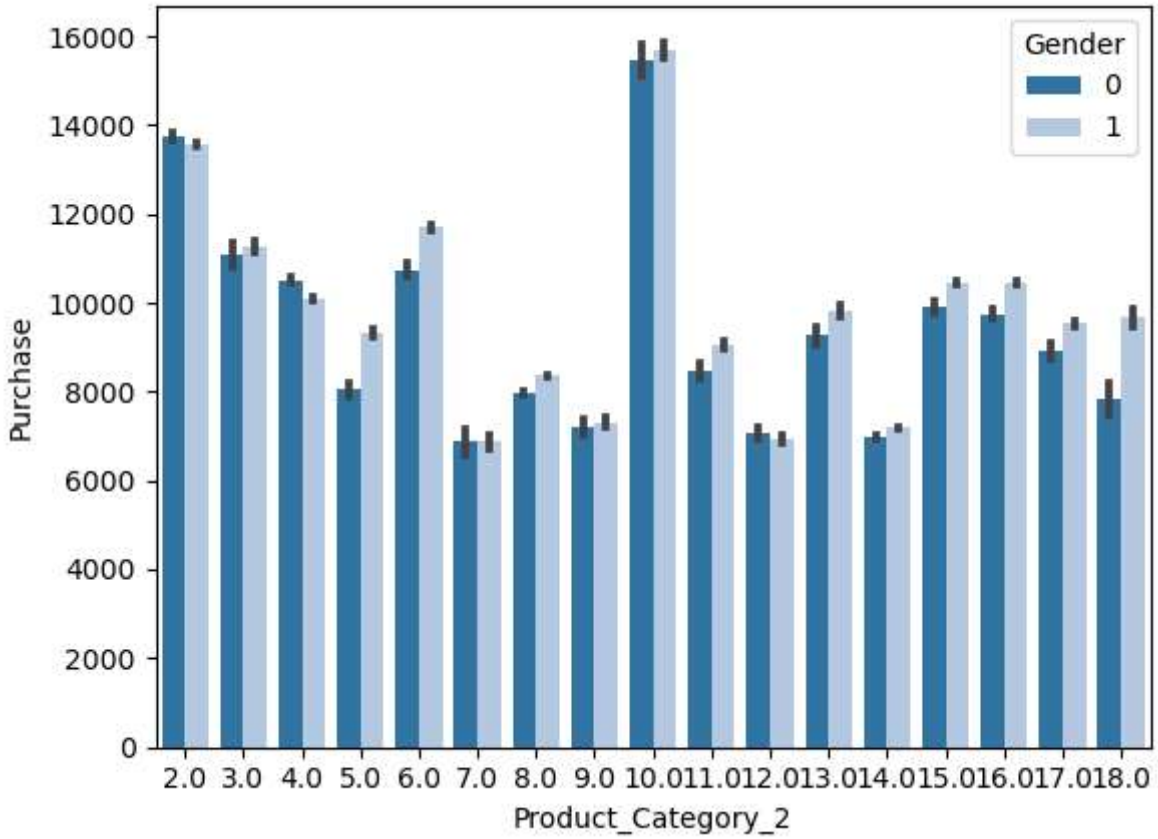
Out[77]:
```
<Axes: xlabel='Product_Category_1', ylabel='Purchase'>
```



6 how many purchases are made by people in Product_Category_2?

In [50]:
```
sns.barplot(x = "Product_Category_2",y  = "Purchase",hue = "Gender",data = df,
```

Out[50]:
```
<Axes: xlabel='Product_Category_2', ylabel='Purchase'>
```



In [51]:
```
df.groupby(['Product_Category_2'], as_index=False)['Purchase'].max()
```

Out[51]:

	Product_Category_2	Purchase
0	2.0	19708
1	3.0	19573
2	4.0	19612
3	5.0	19708
4	6.0	19708
5	7.0	8906
6	8.0	23959
7	9.0	16504
8	10.0	20690
9	11.0	20688
10	12.0	21034
11	13.0	23960
12	14.0	23939
13	15.0	23961
14	16.0	23960
15	17.0	21569
16	18.0	19695

In [52]:
```
top_nreviews = df['Purchase'].nlargest(n=5).index
top_nreviews
```

Out[52]:
```
Int64Index([87440, 93016, 370891, 292083, 321782], dtype='int64')
```

In [53]:
```
top_rating_df = df.iloc[top_nreviews]
top_rating_df
```

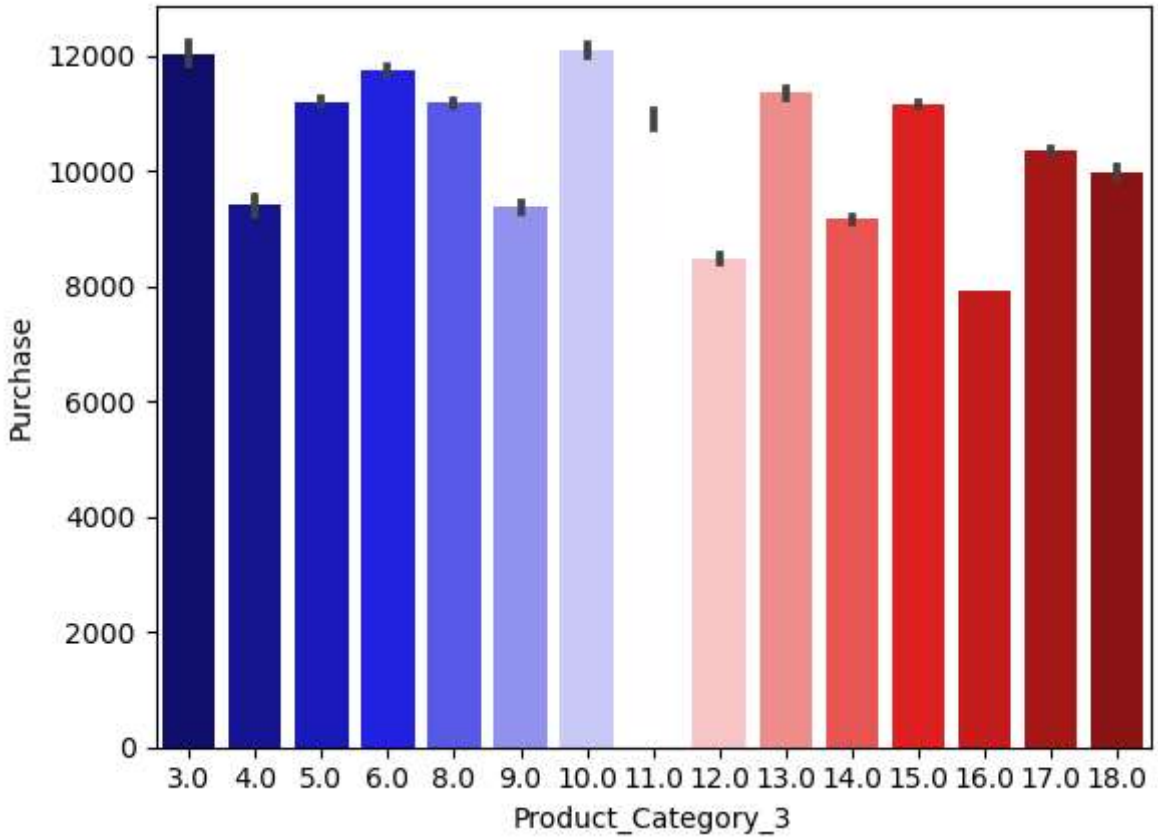
Out[53]:

	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marita
87440	P00052842	1	3	4	A	2	
93016	P00052842	1	3	0	C	1	
370891	P00052842	1	3	17	C	3	
292083	P00052842	1	5	1	B	2	
321782	P00052842	1	7	0	C	1	

7 how many purchases are made by people in Product_Category_3?

In [78]:
```
sns.barplot(x = "Product_Category_3",y  = "Purchase",data = df,palette = "seis
```

Out[78]:
```
<Axes: xlabel='Product_Category_3', ylabel='Purchase'>
```



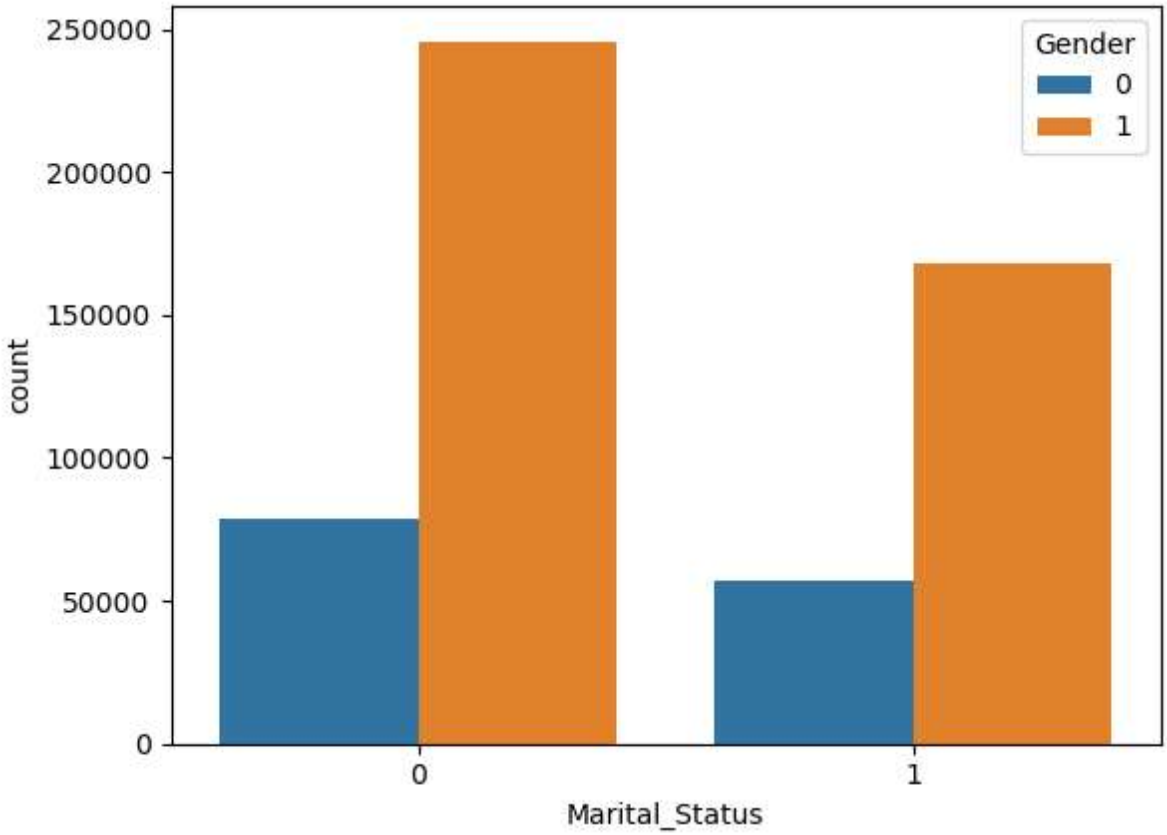
How many married and unmarried buyers in this dataset

In [55]:
```
df.groupby('Marital_Status').size()
```

Out[55]:
```
Marital_Status
0    324731
1    225337
dtype: int64
```

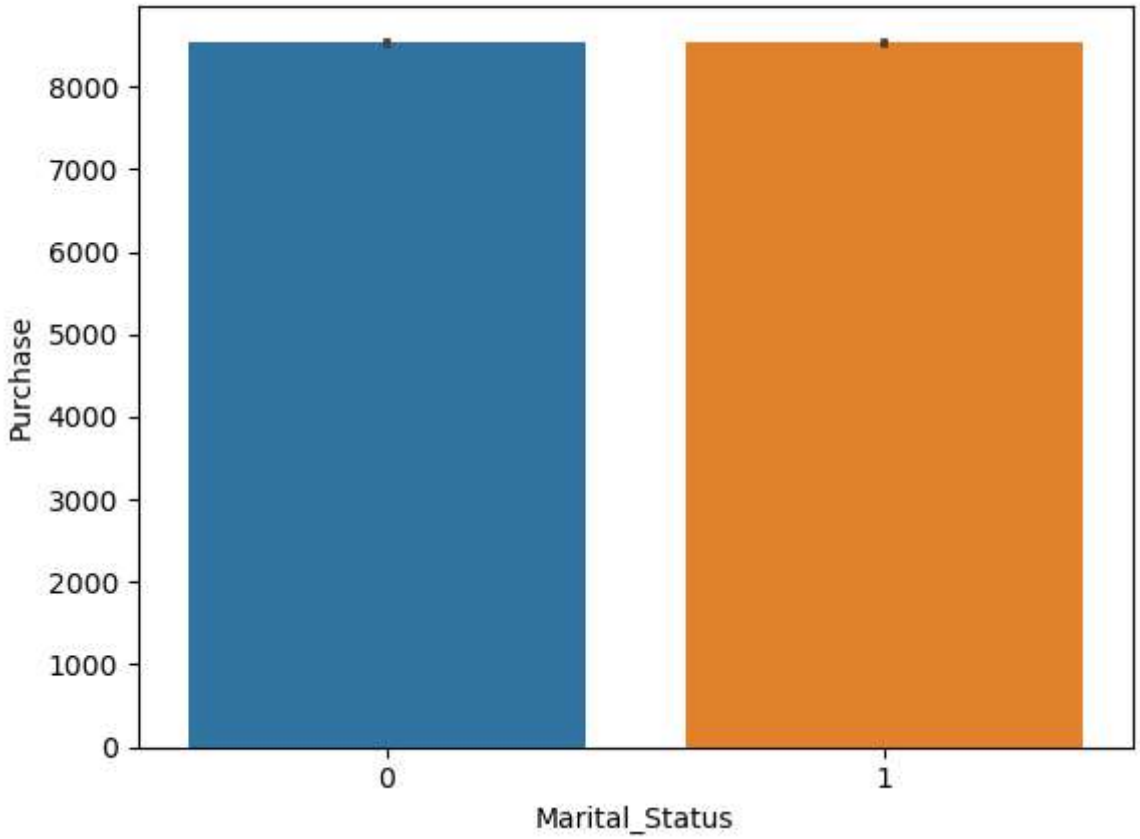
In [56]:
```
sns.countplot(x = "Marital_Status",hue = "Gender",data = df)#No of Unmarried a
```

Out[56]:
```
<Axes: xlabel='Marital_Status', ylabel='count'>
```



In [76]:
```
sns.barplot(x = "Marital_Status",y = "Purchase",data = df)
```

Out[76]:
```
<Axes: xlabel='Marital_Status', ylabel='Purchase'>
```



No of Purchases done by cities?

which city had done highest no purchases?

In [57]:
```
df.groupby('City_Category').size()
```

Out[57]:
```
City_Category
A    147720
B    231173
C    171175
dtype: int64
```

In [58]:
```
df["City_Category_binary"]=df["City_Category"].map({"A":0,"B":1,"C":2})
```

In [59]:
```
df
```

Out[59]:

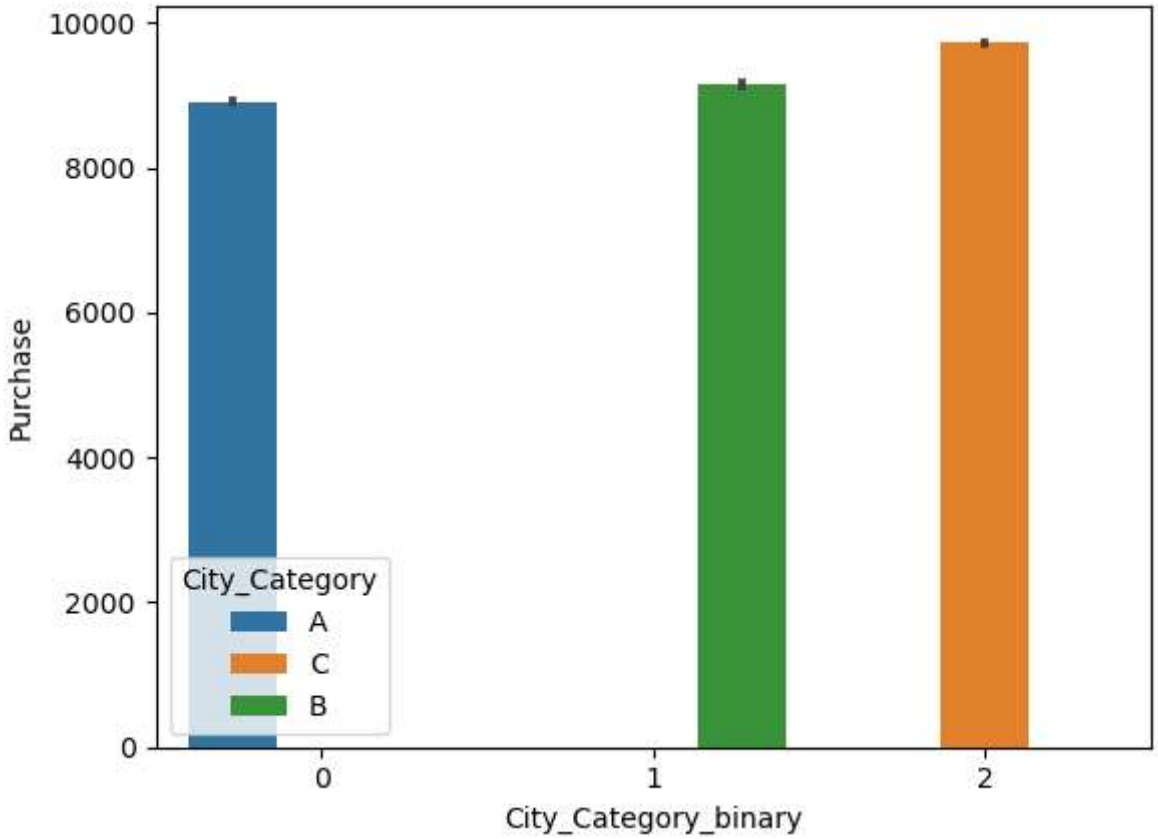
	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marita
0	P00069042	0	1	10	A	2	
1	P00248942	0	1	10	A	2	
2	P00087842	0	1	10	A	2	
3	P00085442	0	1	10	A	2	
4	P00285442	1	7	16	C	4	
...	...	...	...	...	...	...	
550063	P00372445	1	6	13	B	1	
550064	P00375436	0	3	1	C	3	
550065	P00375436	0	3	15	B	4	
550066	P00375436	0	7	1	C	2	
550067	P00371644	0	5	0	B	4	

550068 rows × 15 columns

Highest no of purchases done by city C ?

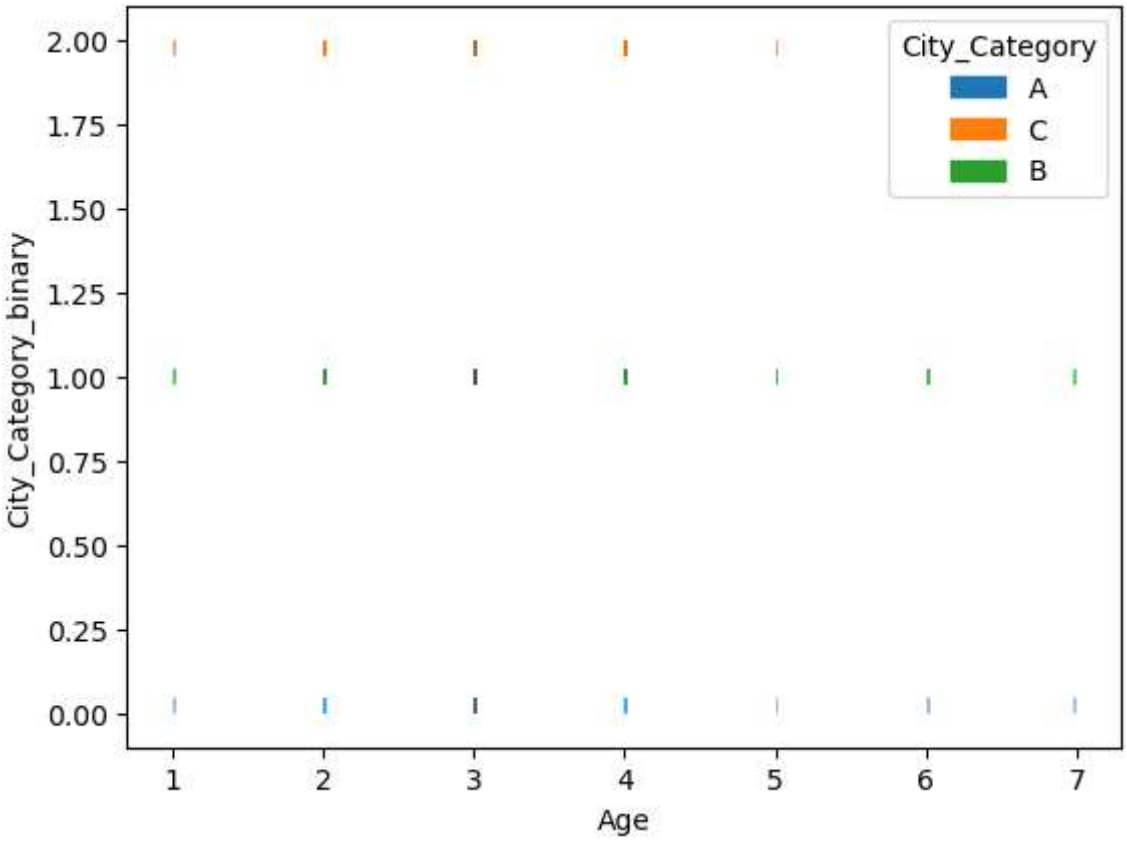
In [60]:
```
sns.barplot(x = 'City_Category_binary',y = 'Purchase',hue ="City_Category" ,da
```

Out[60]:
```
<Axes: xlabel='City_Category_binary', ylabel='Purchase'>
```



In [61]:
```
sns.histplot(x="Age" ,y = "City_Category_binary",hue = "City_Category",data =
```

Out[61]:
```
<Axes: xlabel='Age', ylabel='City_Category_binary'>
```

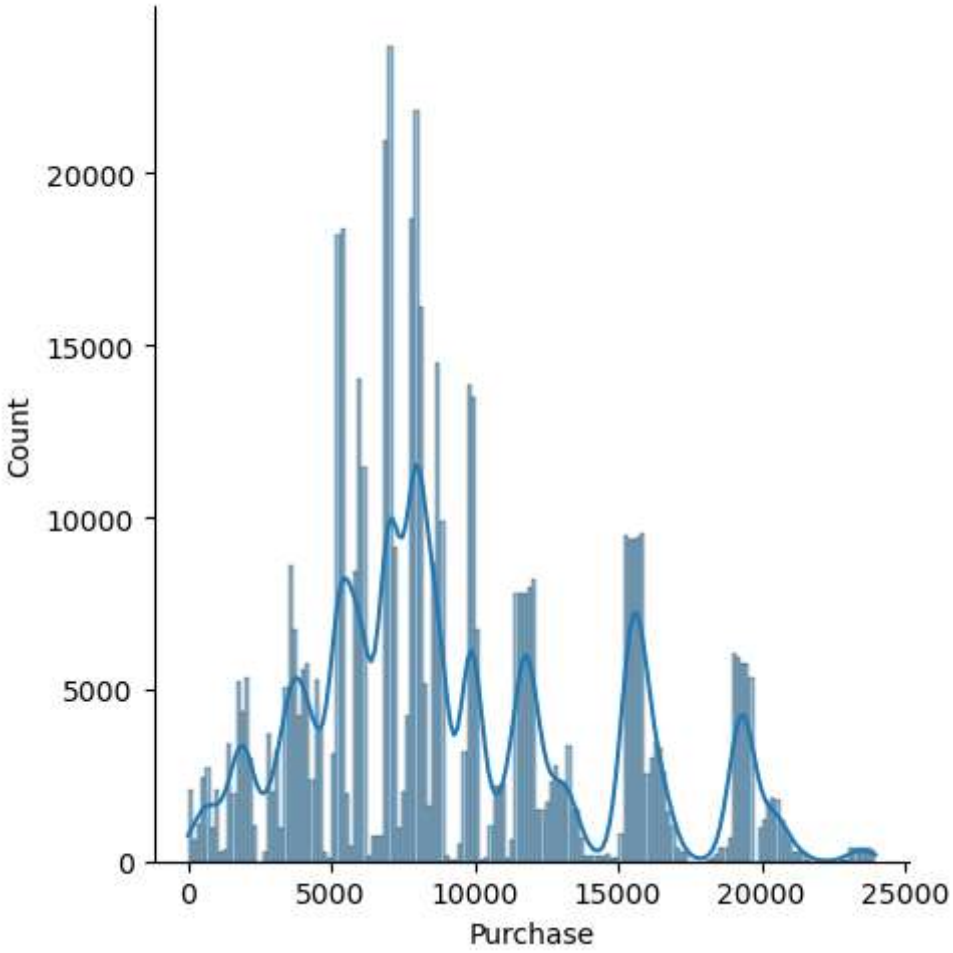


In [62]:
```
df["Age"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 550068 entries, 0 to 550067
Series name: Age
Non-Null Count   Dtype
--------------   -----
550068 non-null  int64
dtypes: int64(1)
memory usage: 4.2 MB
```

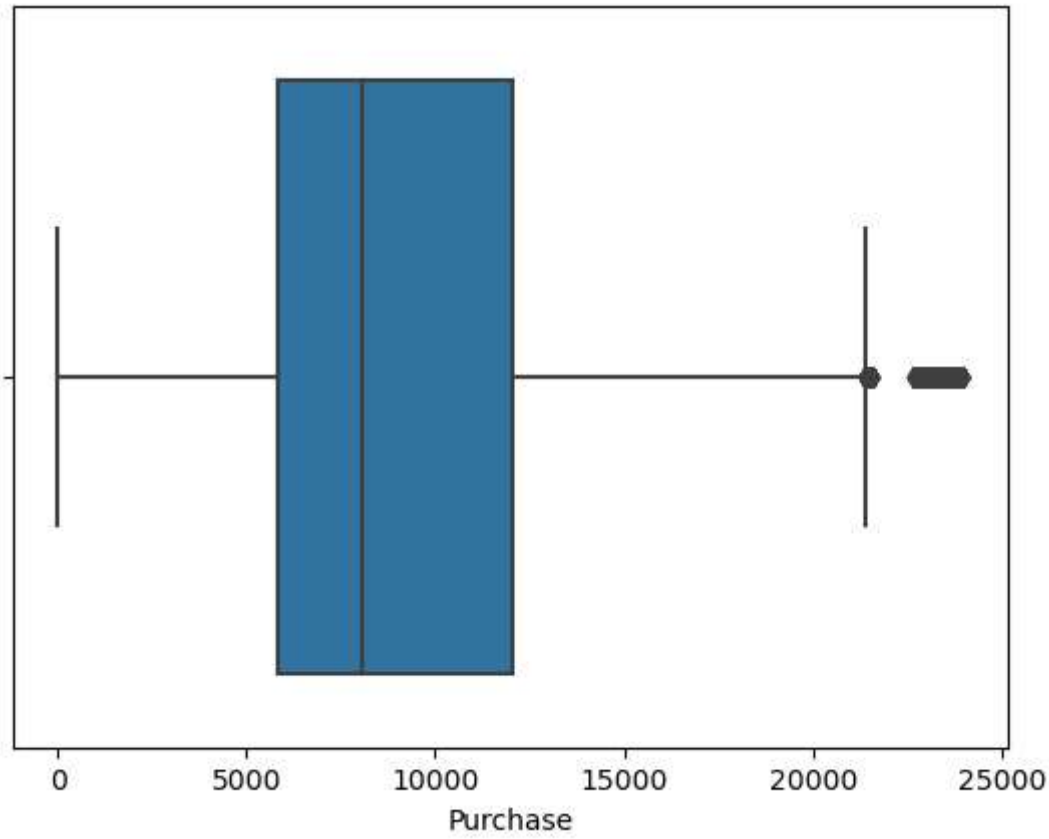
In [63]:
```
sns.displot(x = "Purchase",data = df,kde = True)
```

Out[63]:
```
<seaborn.axisgrid.FacetGrid at 0x2811d45cb50>
```



In [64]:
```
sns.boxplot(x = "Purchase",data = df)
```

Out[64]:
```
<Axes: xlabel='Purchase'>
```



In [65]:
```
df["Purchase"].unique()
```

Out[65]:
```
array([ 8370, 15200,  1422, ...,   135,   123,   613], dtype=int64)
```

In [106]:
```
std = df["Purchase"].std()
std
```

Out[106]:
```
3816.555961008771
```

In [107]:
```
df["Purchase"].quantile(0.75)
```

Out[107]:
```
12054.0
```

In [108]:
```
IQR = df["Purchase"].quantile(0.75)-df["Purchase"].quantile(0.25)
IQR
```

Out[108]:
```
6231.0
```

In [109]:
```
upperlimit = IQR + 3*std
upperlimit
```

Out[109]: 17680.66788302631

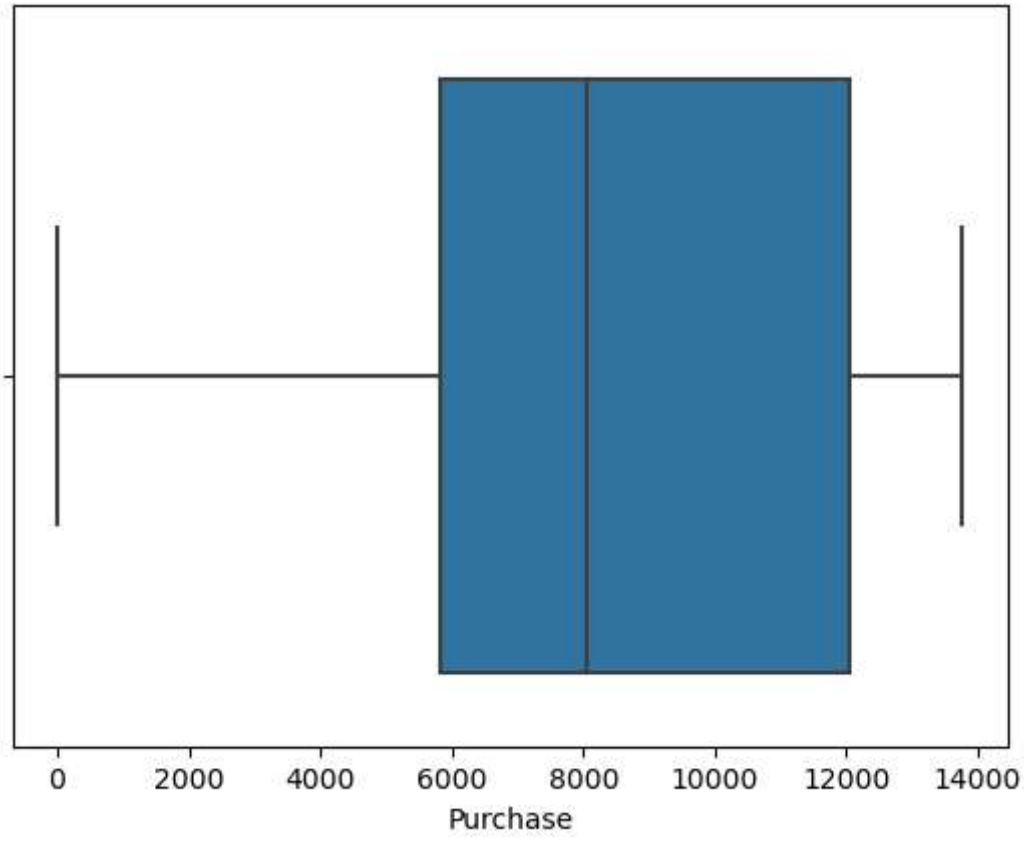
In [110]:
```
lowerlimit = IQR - 3*std
lowerlimit
```

Out[110]: -5218.667883026314

In [111]:
```
df.loc[df["Purchase"]>21300,"Purchase"] = 21300
```

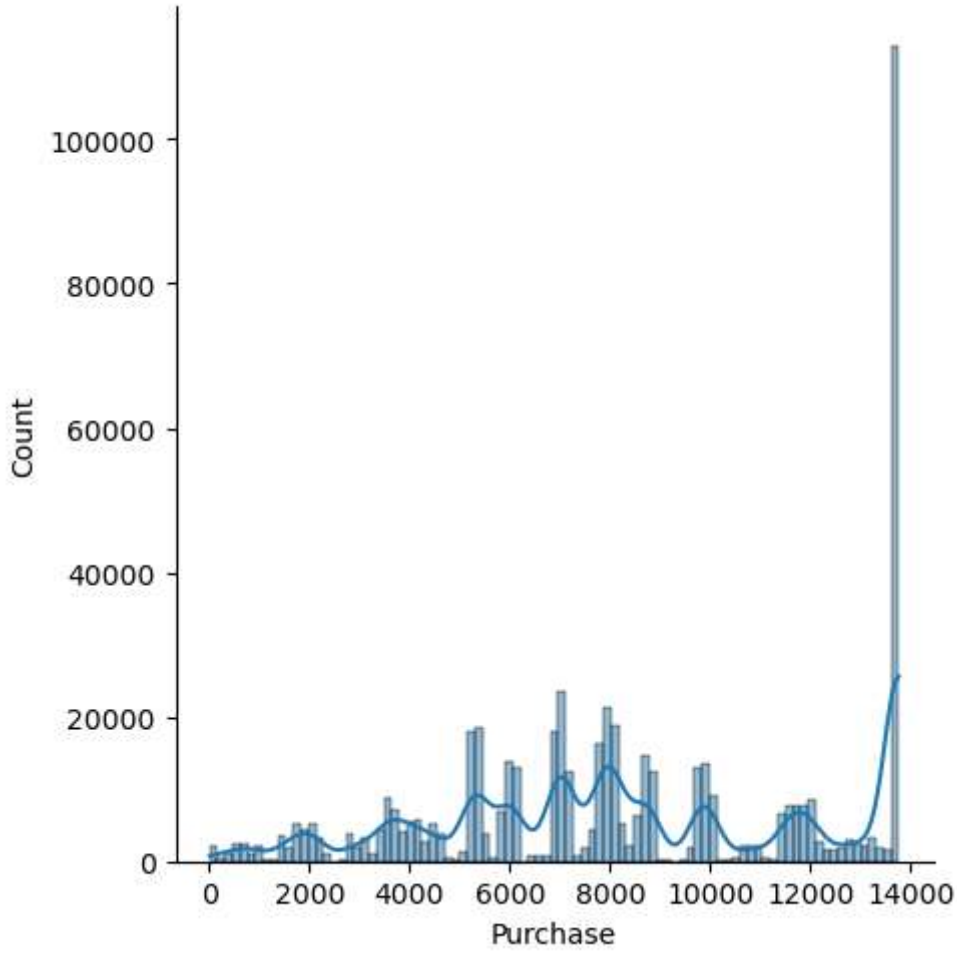
In [112]:
```
sns.boxplot(x = "Purchase",data = df)
```

Out[112]: <Axes: xlabel='Purchase'>



In [113]:
```
sns.displot(x = "Purchase",data = df,kde = True)
```

Out[113]:
```
<seaborn.axisgrid.FacetGrid at 0x2811de99050>
```



In [73]:
```
df.groupby(['Product_Category_2'], as_index=False)['Purchase'].max()
```

Out[73]:

	Product_Category_2	Purchase
0	2.0	13765.598091
1	3.0	13765.598091
2	4.0	13765.598091
3	5.0	13765.598091
4	6.0	13765.598091
5	7.0	8906.000000
6	8.0	13765.598091
7	9.0	13765.598091
8	10.0	13765.598091
9	11.0	13765.598091
10	12.0	13765.598091
11	13.0	13765.598091
12	14.0	13765.598091
13	15.0	13765.598091
14	16.0	13765.598091
15	17.0	13765.598091
16	18.0	13765.598091

In [74]:
```
df.head()
```

Out[74]:

rent_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	
2	0	3	8.0	16.0	83
2	0	1	6.0	14.0	137
2	0	12	8.0	16.0	14
2	0	12	14.0	16.0	10
4	0	8	8.0	16.0	79

In [ ]:
```

```