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

In [3]:
```
df = pd.read_csv('Diwali Sales Data.csv', encoding="unicode_escape")
# encoding="unicode_escape" we use the code in order to read the unique characte
```

In [4]:
```
df
# to see the whole dataset we can use the file name in which we have stored the
```

Out[4]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	
0	1002903	Sanskriti	P00125942	F	26-35	28	0	Mah
1	1000732	Kartik	P00110942	F	26-35	35	1	Andhra
2	1001990	Bindu	P00118542	F	26-35	35	1	Uttar
3	1001425	Sudevi	P00237842	M	0-17	16	0	Ka
4	1000588	Joni	P00057942	M	26-35	28	1	
...	...	...	...	...	...	...	...	
11246	1000695	Manning	P00296942	M	18-25	19	1	Mah
11247	1004089	Reichenbach	P00171342	M	26-35	33	0	
11248	1001209	Oshin	P00201342	F	36-45	40	0	
11249	1004023	Noonan	P00059442	M	36-45	37	0	Ka
11250	1002744	Brumley	P00281742	F	18-25	19	0	Mah

11251 rows × 15 columns

In [6]:
```
df.shape
# we use the above syntax to see the number of rows and columns
```

Out[6]:
```
(11251, 15)
```

In [7]:
```
df.head()
# we use the above syntax in order to see the top 5 rows of the data
```

Out[7]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	Stat
0	1002903	Sanskriti	P00125942	F	26-35	28	0	Maharashtı
1	1000732	Kartik	P00110942	F	26-35	35	1	Andhra Prades
2	1001990	Bindu	P00118542	F	26-35	35	1	Uttar Prades
3	1001425	Sudevi	P00237842	M	0-17	16	0	Karnatak
4	1000588	Joni	P00057942	M	26-35	28	1	Gujar

In [8]:
```
df.head(2)
# we can add the number of rows we wanna see in the head()
```

Out[8]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	Stat
0	1002903	Sanskriti	P00125942	F	26-35	28	0	Maharashtı
1	1000732	Kartik	P00110942	F	26-35	35	1	Andhra Prades

In [9]:
```
df.info()
# we use this syntax to see the columns details and datatype
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11251 entries, 0 to 11250
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   User_ID           11251 non-null  int64  
 1   Cust_name         11251 non-null  object 
 2   Product_ID        11251 non-null  object 
 3   Gender            11251 non-null  object 
 4   Age Group         11251 non-null  object 
 5   Age               11251 non-null  int64  
 6   Marital_Status    11251 non-null  int64  
 7   State             11251 non-null  object 
 8   Zone              11251 non-null  object 
 9   Occupation        11251 non-null  object 
 10  Product_Category  11251 non-null  object 
 11  Orders            11251 non-null  int64  
 12  Amount            11239 non-null  float64
 13  Status            0 non-null      float64
 14  unnamed1          0 non-null      float64
dtypes: float64(3), int64(4), object(8)
memory usage: 1.3+ MB
```

In [10]:
```
# df.drop['Status' , 'unnamed1'], axis=1 , inplace = True)
# # to drop a column
```

In [11]:
```
df.isnull()
# we use ISNULL to see if there's any null value
```

Out[11]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	State	Z
0	False	False	False	False	False	False	False	False	F
1	False	False	False	False	False	False	False	False	F
2	False	False	False	False	False	False	False	False	F
3	False	False	False	False	False	False	False	False	F
4	False	False	False	False	False	False	False	False	F
...	...	...	...	...	...	...	...	...	
11246	False	False	False	False	False	False	False	False	F
11247	False	False	False	False	False	False	False	False	F
11248	False	False	False	False	False	False	False	False	F
11249	False	False	False	False	False	False	False	False	F
11250	False	False	False	False	False	False	False	False	F

11251 rows × 15 columns

In [10]:
```
df.isnull().sum()
# to get the count of missing values in each column
```

Out[10]:
```
User_ID             0
Cust_name           0
Product_ID          0
Gender              0
Age Group           0
Age                 0
Marital_Status      0
State               0
Zone                0
Occupation          0
Product_Category    0
Orders              0
Amount             12
Status          11251
unnamed1        11251
dtype: int64
```

In [12]:
```
df.dropna(inplace=True)
# to delete the null values from the rows
```

In [12]:
```
df.shape
```

Out[12]:
```
(0, 15)
```

In [13]:
```
df['Amount']=df['Amount'].astype('int')
# to change a column name we use .astype
```

In [14]:
```
df['Amount'].dtypes
```

```
# to see if the particular column datatype is changed
```

Out[14]:
```
dtype('int32')
```

In [15]:
```
df.columns
# to see the column names
```

Out[15]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount', 'Status', 'unnamed1'],
      dtype='object')
```

In [16]:
```
df.rename(columns={'State':'States'})
# to rename a column we use this syntax
```

Out[16]:

User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	States	Zone

In [17]:
```
df.describe()
# to see the numeric value columns count,mean,std etc
```

Out[17]:

	User_ID	Age	Marital_Status	Orders	Amount	Status	unnamed1
count	0.0	0.0	0.0	0.0	0.0	0.0	0.0
mean	NaN	NaN	NaN	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [18]:
```
df[['User_ID','Amount']].describe()
# to see a particular columns count,mean,std etc
```

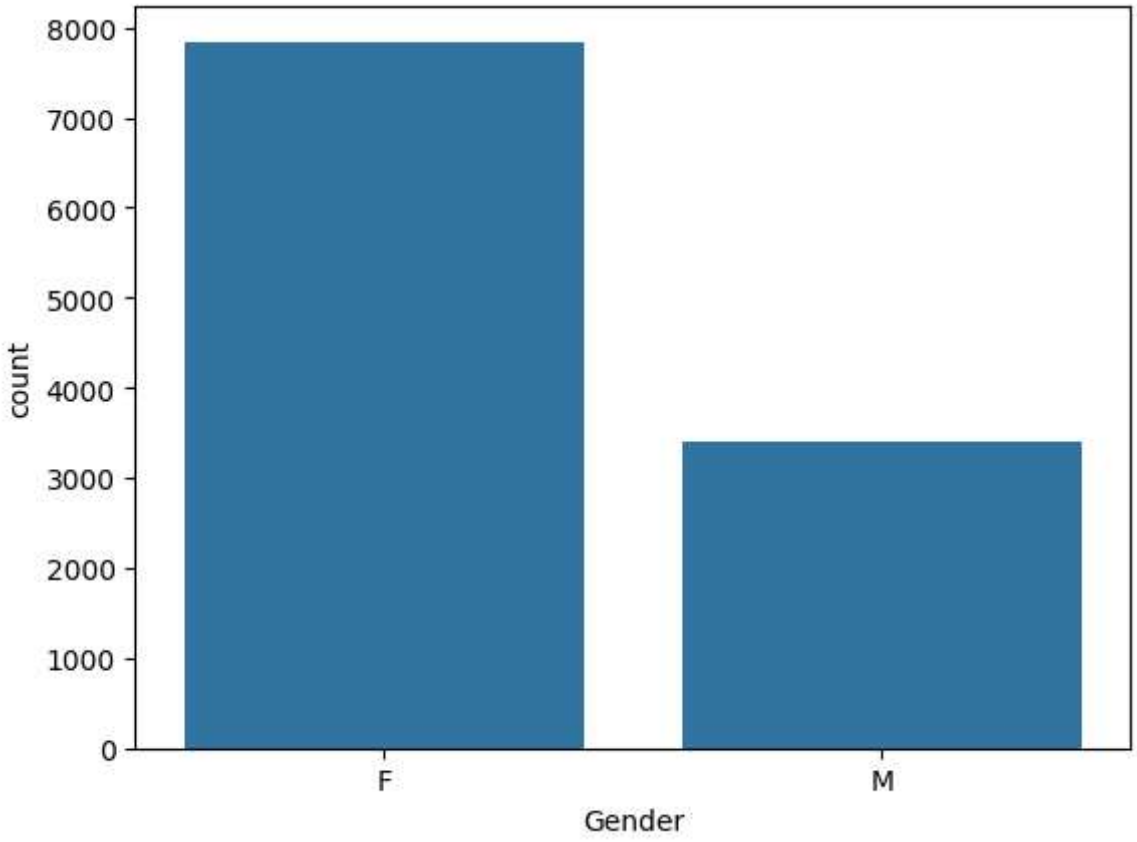
Out[18]:

	User_ID	Amount
count	0.0	0.0
mean	NaN	NaN
std	NaN	NaN
min	NaN	NaN
25%	NaN	NaN
50%	NaN	NaN
75%	NaN	NaN
max	NaN	NaN

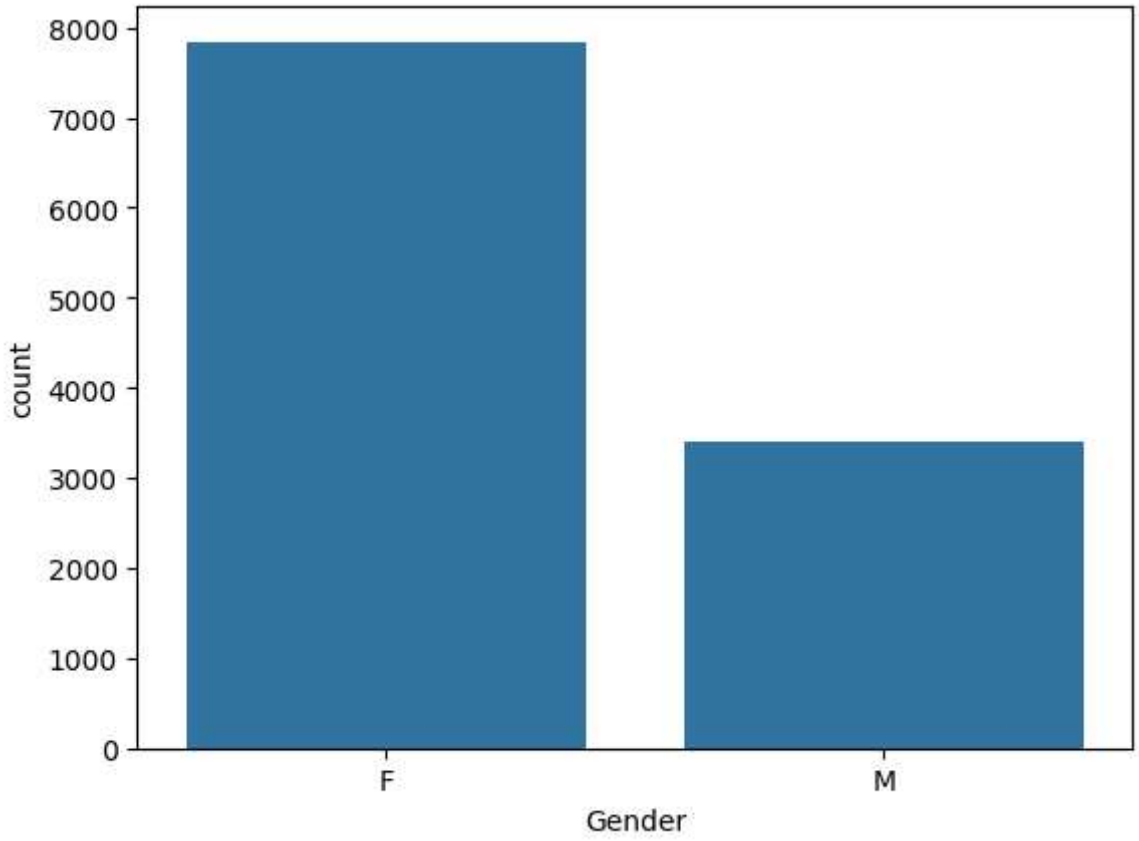
Exploratory Data Analysis

Gender

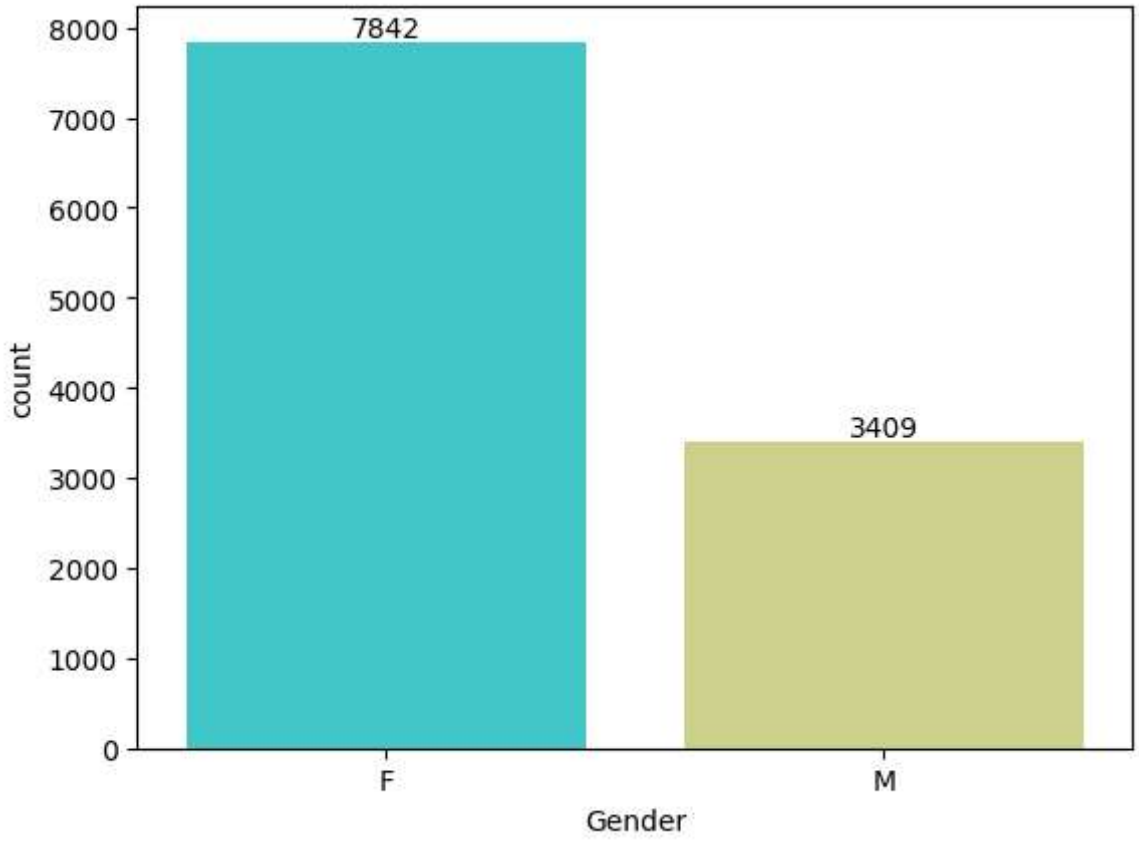
In [5]:
```
sns.countplot(x='Gender',data=df)
plt.show()
```



In [6]:
```
ax = sns.countplot(x='Gender',data=df)
# creating a barchart without the count
```



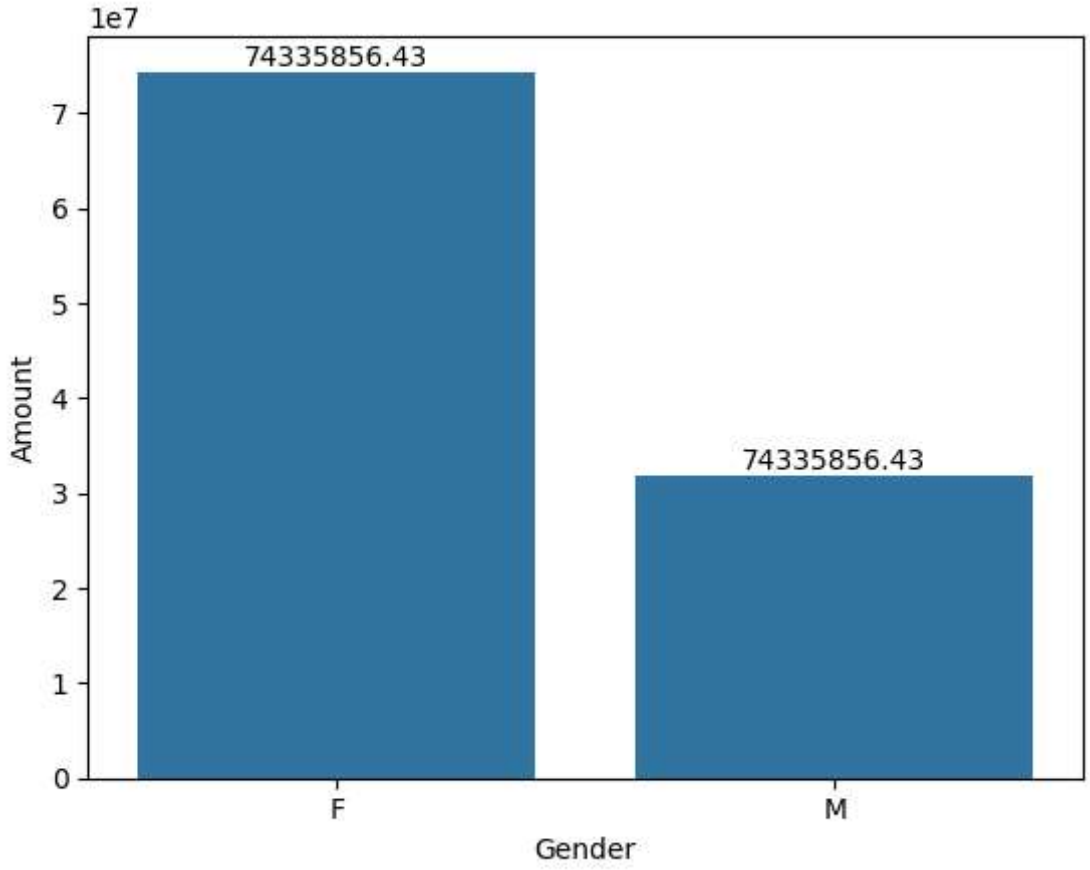
In [7]:
```
# plotting a bar chart for Gender and it's count

ax = sns.countplot(x = 'Gender',
                   data = df,
                   palette="rainbow",
                   hue="Gender"
)
for bars in ax.containers:
    ax.bar_label(bars)
```



In [20]:
```
sales_gender = df.groupby('Gender')['Amount'].sum().reset_index()
print(sales_gender)
# to find the sales amount by grouping method with gender
```

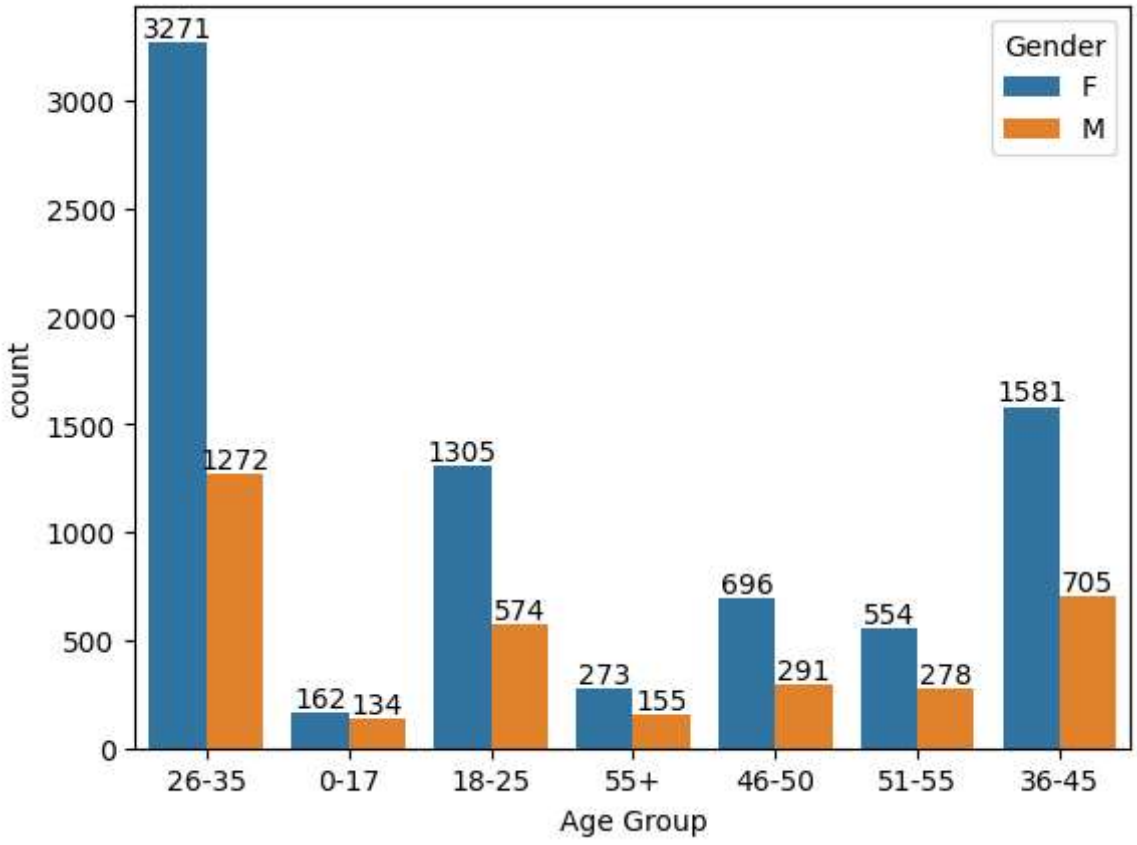
```
  Gender       Amount
0      F  74335856.43
1      M  31913276.00
```

In [21]:
```
# Assuming 'sales_gender' is your DataFrame
ax = sns.barplot(x='Gender', y='Amount', data=sales_gender)

# Adding labels to the bars with formatted values
for bar, label in zip(ax.containers, sales_gender['Amount']):
    ax.bar_label(bar, fmt='%.2f' % label)
plt.show()
```

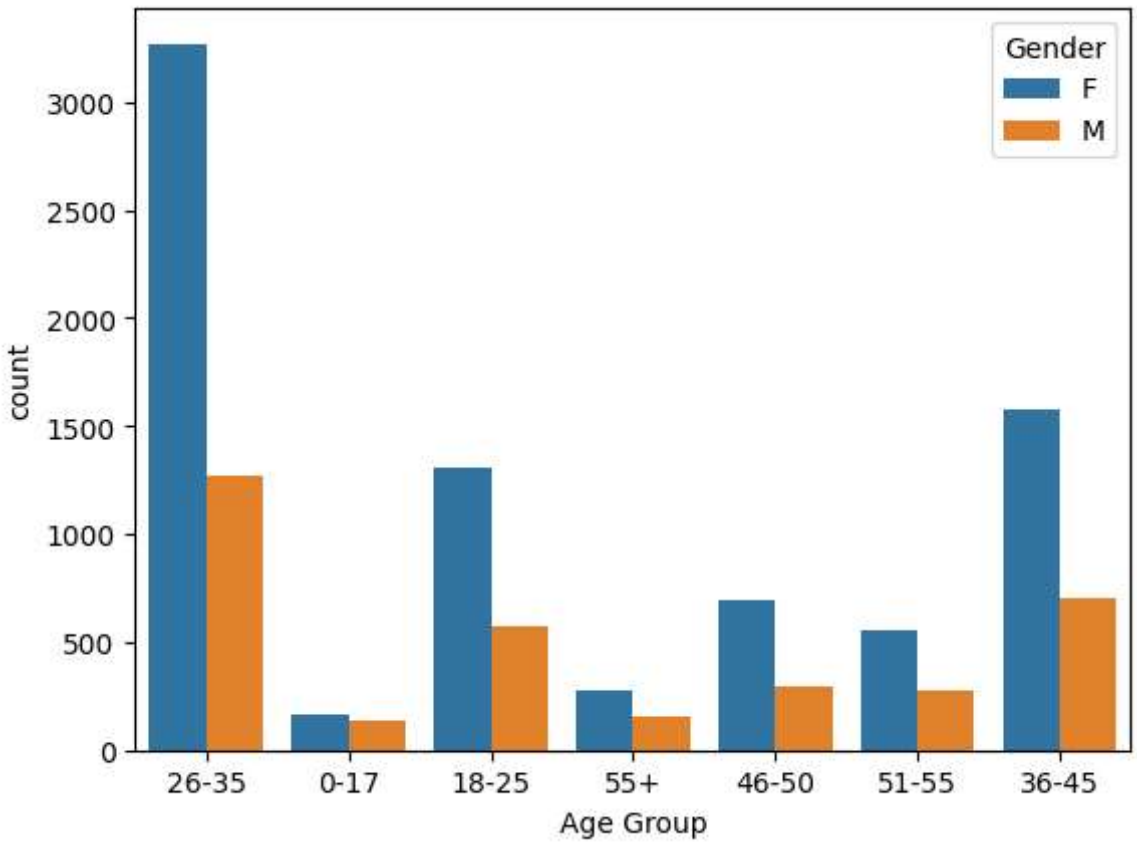


AGE

In [22]:
```
ax=sns.countplot(x='Age Group',
                 hue='Gender',
                 data=df
)
for bars in ax.containers:
    ax.bar_label(bars)
```



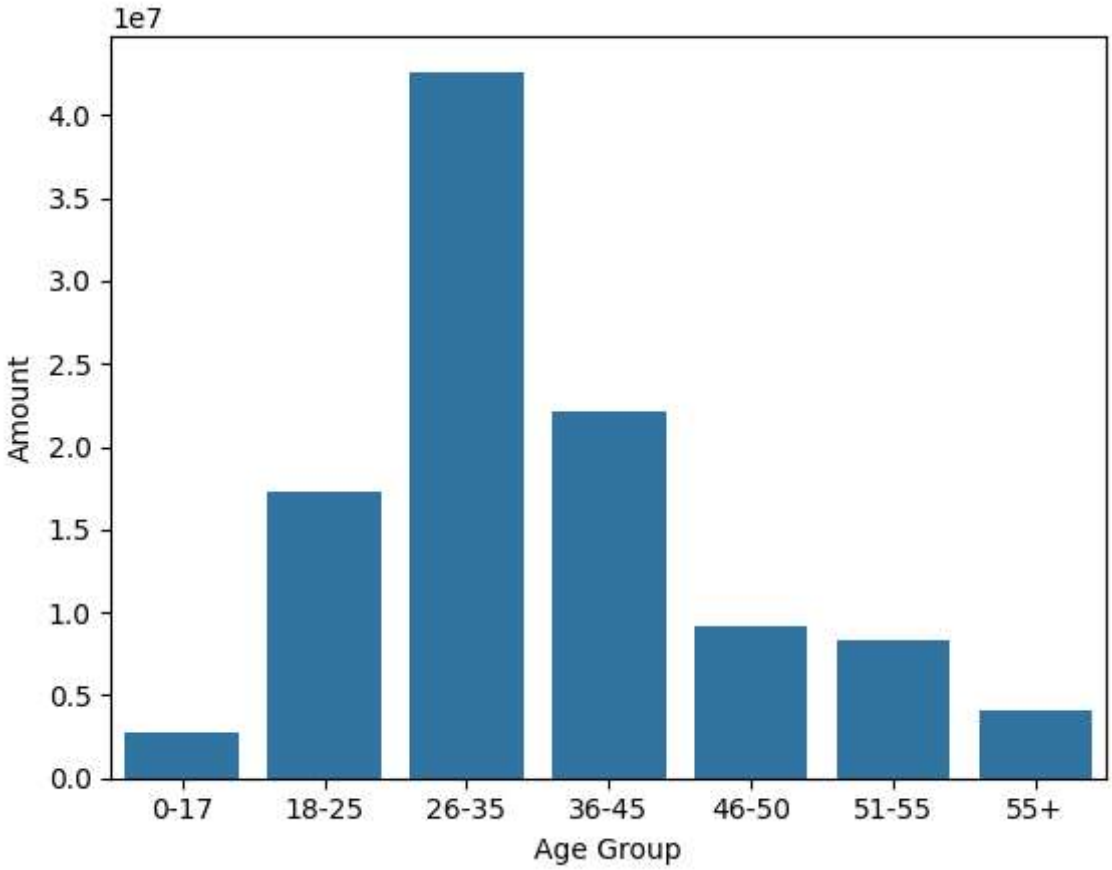
In [23]:
```
sns.countplot(x='Age Group',
              hue='Gender',
              data=df
)
plt.show()
```



In [25]:
```
sales_age=df.groupby('Age Group')['Amount'].sum().reset_index()
print(sales_age)
```

```
  Age Group       Amount
0      0-17   2699653.00
1     18-25  17240732.00
2     26-35  42613443.94
3     36-45  22144995.49
4     46-50   9207844.00
5     51-55   8261477.00
6       55+   4080987.00
```

In [26]:
```
sns.barplot(x='Age Group',
            y='Amount',
            data=sales_age
)
plt.show()
```



State

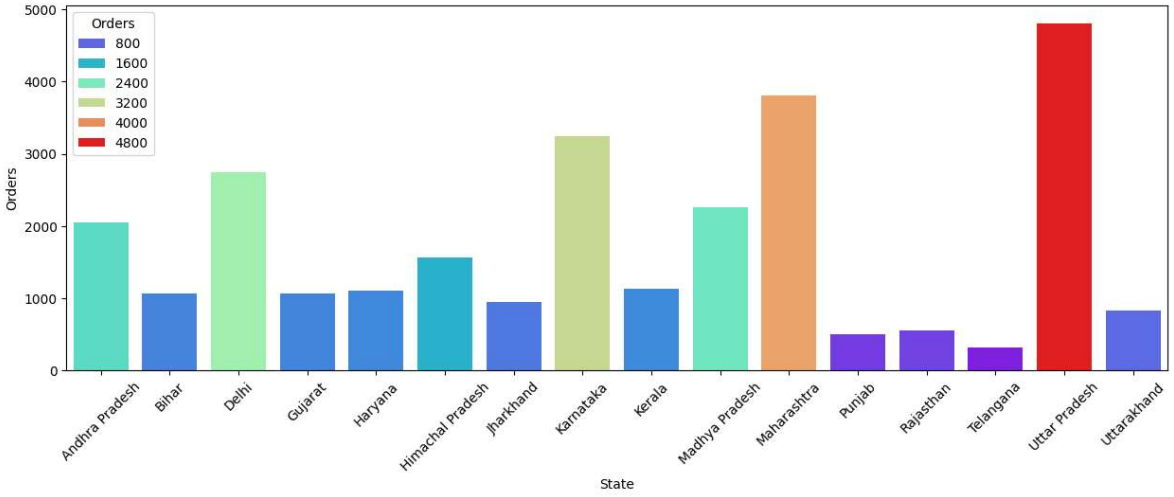
In [28]:
```
df
```

Out[28]:

User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	State	Zone	C

In [27]:
```
sales_state=df.groupby('State')['Orders'].sum().reset_index()
plt.figure(figsize=(15,5))
sns.barplot(x='State',
```

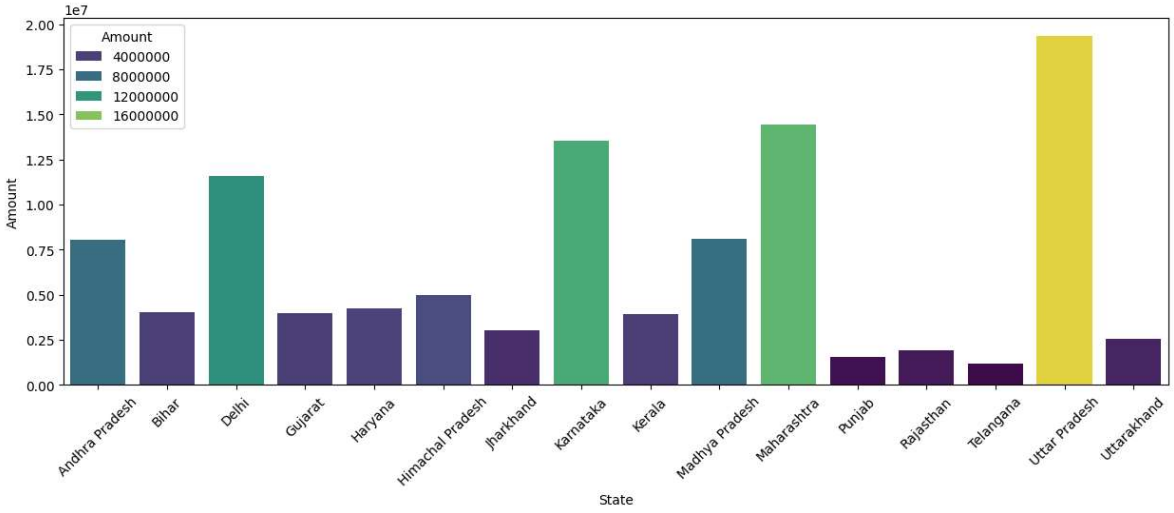
```
            y='Orders',
            data=sales_state,
            palette='rainbow',
            hue='Orders'
)
plt.xticks(rotation=45)
plt.show()
```



In [59]:
```
sales_state=df.groupby('State')['Amount'].sum().reset_index
print(sales_state)
```

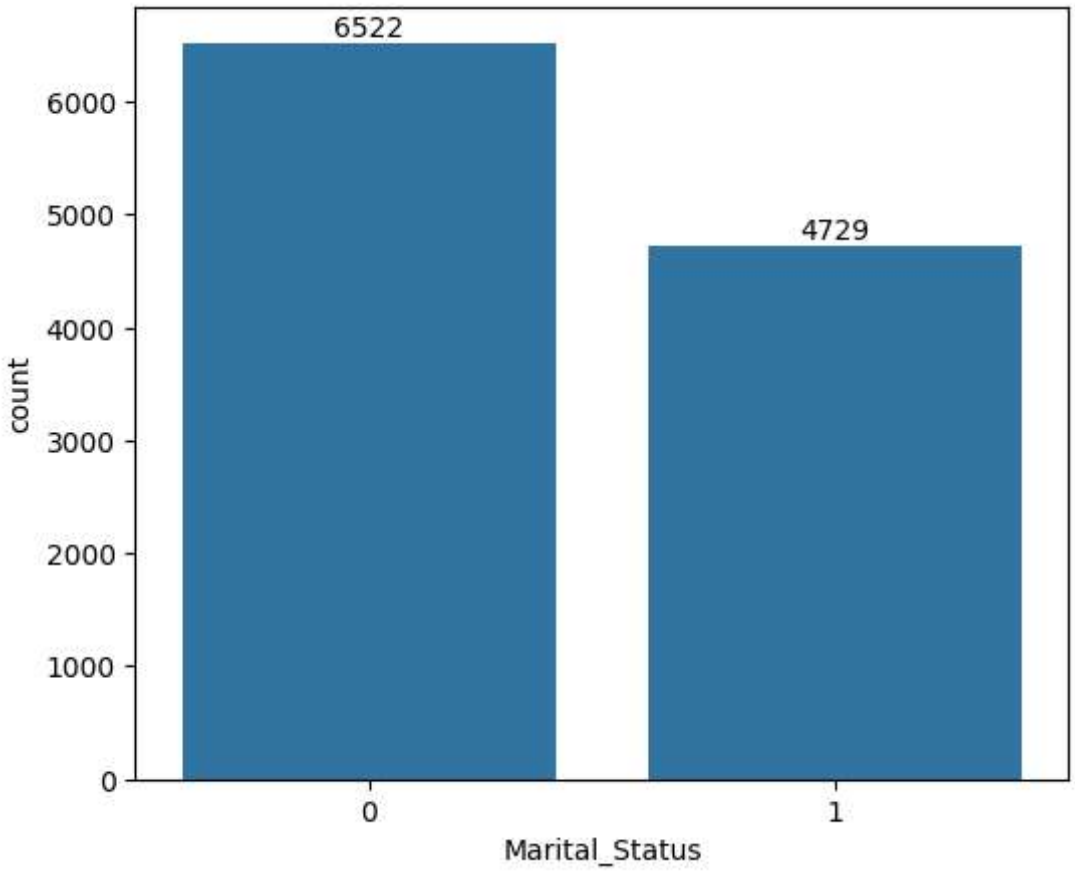
```
<bound method Series.reset_index of State
Andhra Pradesh       8037146.99
Bihar                4022757.00
Delhi               11603819.45
Gujarat              3946082.00
Haryana              4220175.00
Himachal Pradesh     4963368.00
Jharkhand            3026456.00
Karnataka           13523540.00
Kerala               3894491.99
Madhya Pradesh       8101142.00
Maharashtra         14427543.00
Punjab               1525800.00
Rajasthan            1909409.00
Telangana            1151490.00
Uttar Pradesh       19374968.00
Uttarakhand          2520944.00
Name: Amount, dtype: float64>
```

In [28]:
```
# States getting highest amount of top 10

sales_state=df.groupby('State')['Amount'].sum().reset_index()
plt.figure(figsize=(15,5))
sns.barplot(x='State',y='Amount',
            data=sales_state,
            palette='viridis',
            hue='Amount'
)
plt.xticks(rotation=45)
plt.show()
```



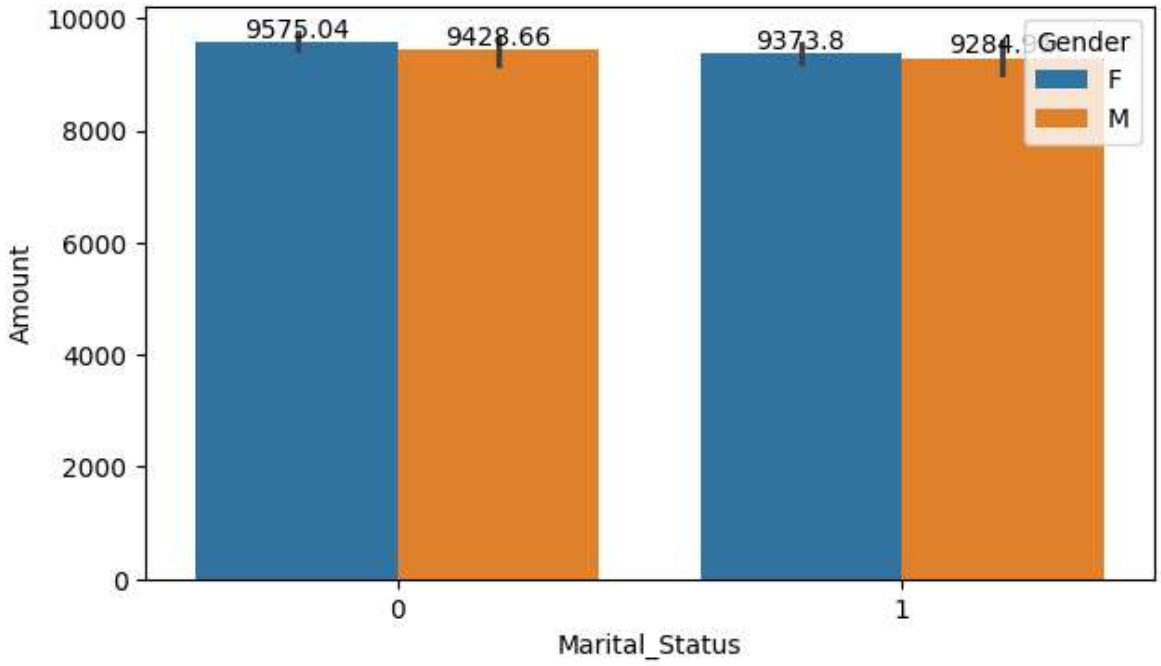
Martial Status

In [30]:
```
plt.figure(figsize=(6,5))
ax=sns.countplot(data=df,
                 x='Marital_Status'
)
for bars in ax.containers:
    ax.bar_label(bars)
plt.show()
```



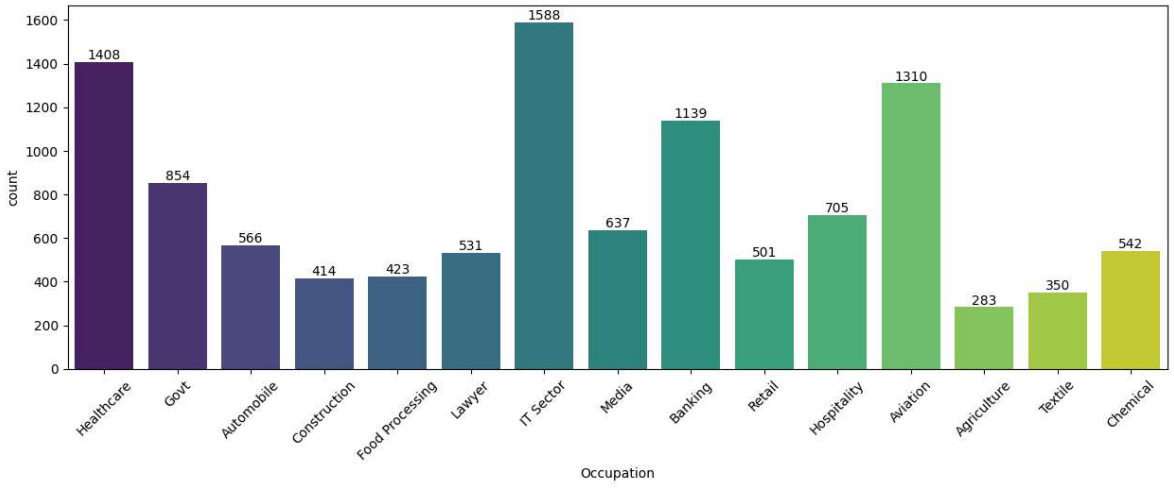
In [31]:
```
plt.figure(figsize=(7,4))
ax=sns.barplot(data=df,
               x='Marital_Status',
               y='Amount',
```

```
                hue='Gender'
)
for bars in ax.containers:
    ax.bar_label(bars)
plt.show()
```



Occupation

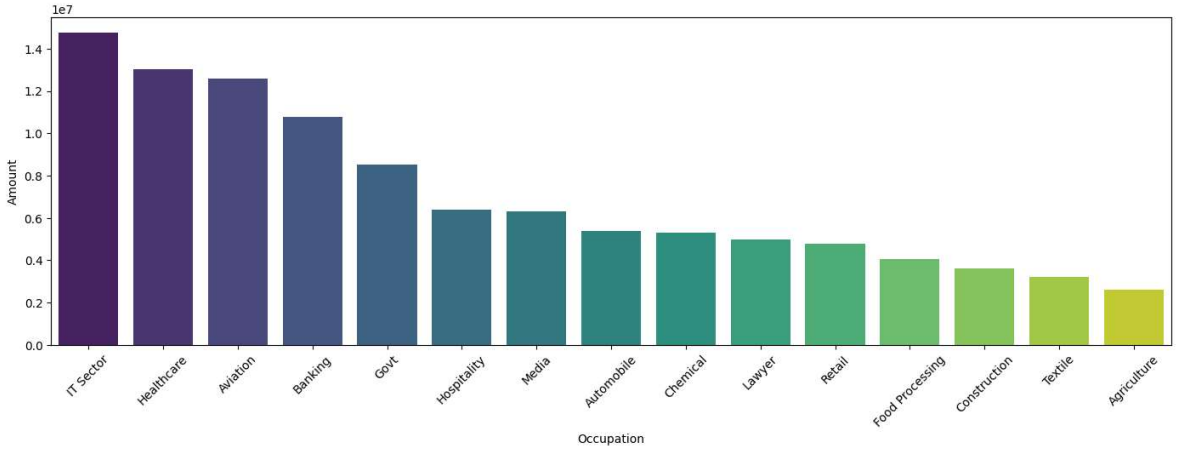
In [32]:
```
plt.figure(figsize=(15,5))
ax=sns.countplot(data=df,
                 x='Occupation',
                 palette='viridis',
                 hue='Occupation'
)
plt.xticks(rotation=45)
for bars in ax.containers:
    ax.bar_label(bars)
plt.show()
```



In [11]:
```
sales_state=df.groupby('Occupation')['Amount'].sum().reset_index().sort_values(b
print(sales_state)
```

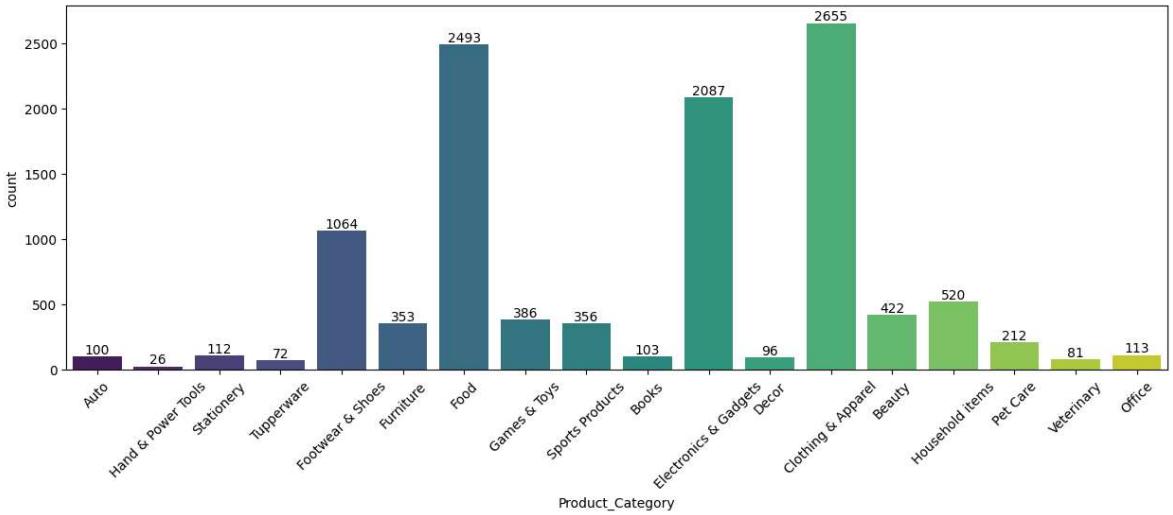
	Occupation	Amount
10	IT Sector	14755079.00
8	Healthcare	13034587.49
2	Aviation	12602298.00
3	Banking	10770610.95
7	Govt	8517212.00
9	Hospitality	6376405.00
12	Media	6295832.99
1	Automobile	5368596.00
4	Chemical	5297436.00
11	Lawyer	4981665.00
13	Retail	4783170.00
6	Food Processing	4070670.00
5	Construction	3597511.00
14	Textile	3204972.00
0	Agriculture	2593087.00

In [12]:
```
plt.figure(figsize=(17,5))
ax=sns.barplot(data=sales_state,
               x='Occupation',
               y='Amount',
               palette='viridis',
               hue='Occupation'
)
plt.xticks(rotation=45)
plt.show()
```



Product Category

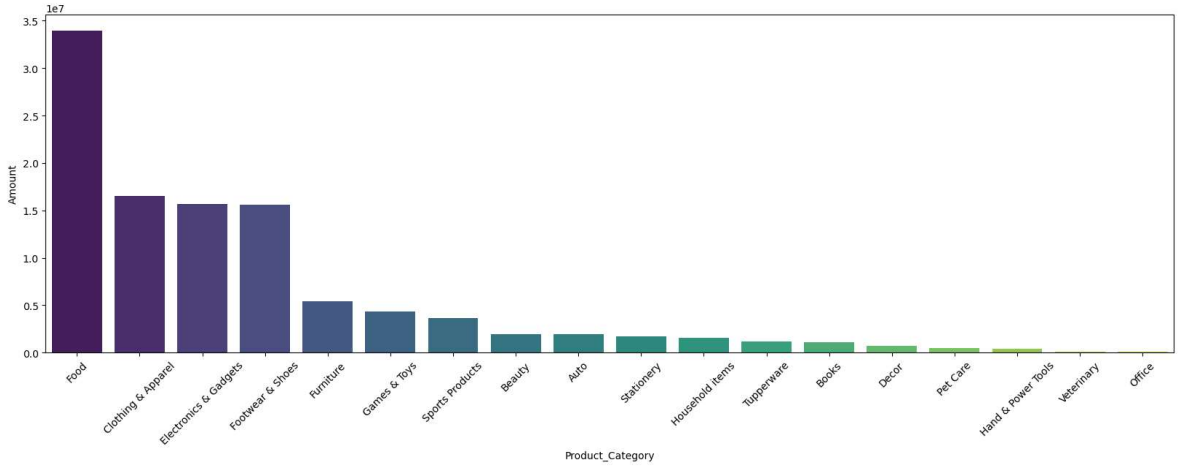
In [66]:
```
plt.figure(figsize=(15,5))
ax=sns.countplot(data=df,x='Product_Category',palette='viridis',hue='Product_Cat
for bars in ax.containers:
    ax.bar_label(bars)
plt.xticks(rotation=45)
plt.show()
```



In [9]:
```
Product_Category_Amount=df.groupby('Product_Category')['Amount'].sum().reset_ind
print(Product_Category_Amount)
```

	Product_Category	Amount
6	Food	33933883.50
3	Clothing & Apparel	16495019.00
5	Electronics & Gadgets	15643846.00
7	Footwear & Shoes	15575209.45
8	Furniture	5440051.99
9	Games & Toys	4331694.00
14	Sports Products	3635933.00
1	Beauty	1959484.00
0	Auto	1958609.99
15	Stationery	1676051.50
11	Household items	1569337.00
16	Tupperware	1155642.00
2	Books	1061478.00
4	Decor	730360.00
13	Pet Care	482277.00
10	Hand & Power Tools	405618.00
17	Veterinary	112702.00
12	Office	81936.00

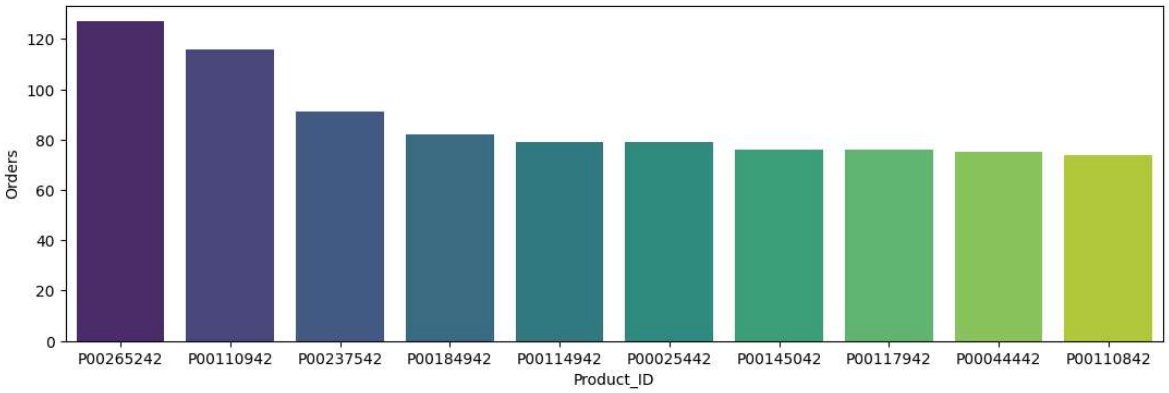
In [10]:
```
plt.figure(figsize=(20,6))


ax=sns.barplot(x='Product_Category',
               y='Amount',
               data=Product_Category_Amount,
               palette='viridis',
               hue='Product_Category')
plt.xticks(rotation=45)
plt.show()
```



In [5]:
```
sales_state=df.groupby('Product_ID')['Orders'].sum().reset_index().sort_values(b
print(sales_state)
```

```
     Product_ID  Orders
1680  P00265242     127
645   P00110942     116
1505  P00237542      91
1147  P00184942      82
680   P00114942      79
172   P00025442      79
889   P00145042      76
709   P00117942      76
299   P00044442      75
644   P00110842      74
```

In [7]:
```
plt.figure(figsize=(13,4))
sns.barplot(x='Product_ID',
            y='Orders',
            data=sales_state,
            palette='viridis',
            hue='Product_ID'
)
plt.show()
```



In [ ]: