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

In [3]:
```
df=pd.read_csv('Diwali Sales Data.csv', encoding='unicode_escape')
#to avoid encoding error, use 'unicode_escape'
```

In [4]:
```
df.shape
```

Out[4]:
```
(11251, 15)
```

In [5]:
```
df.head()
##First 5 values.
```

Out[5]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	State	Zon
0	1002903	Sanskriti	P00125942	F	26-35	28	0	Maharashtra	Wester
1	1000732	Kartik	P00110942	F	26-35	35	1	Andhra Pradesh	Souther
2	1001990	Bindu	P00118542	F	26-35	35	1	Uttar Pradesh	Centra
3	1001425	Sudevi	P00237842	M	0-17	16	0	Karnataka	Souther
4	1000588	Joni	P00057942	M	26-35	28	1	Gujarat	Wester

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

```
<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 [7]:
```
##Drop unrelated/blank columns
df.drop(['Status', 'unnamed1'], axis=1, inplace=True)
##axis 1 means deleting the entire row at once.
##inplace = True means whatever the changes we do in this line, it should be preser
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11251 entries, 0 to 11250
Data columns (total 13 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
dtypes: float64(1), int64(4), object(8)
memory usage: 1.1+ MB
```

In [9]:
```
##Last 2 columns not visible
```

In [10]:
```
pd.isnull(df)
```

Out[10]:

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

11251 rows × 13 columns

In [11]:
```
pd.isnull(df).sum()
##Check for null values
```

Out[11]:
```
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
dtype: int64
```

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

Out[12]:
```
(11251, 13)
```

In [13]:
```
##Drop null values
df.dropna(inplace=True)
```

In [14]:
```
pd.isnull(df).sum()
```

Out[14]:
```
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               0
dtype: int64
```

In [15]:
```
##The Amount turned to be 0.
```

In [16]:
```
##change data type
##Function used to change the Data type
df['Amount'] = df['Amount'].astype('int')
```

In [17]:
```
df['Amount'].dtypes
```

Out[17]:
```
dtype('int32')
```

In [18]:
```
df.columns
```

Out[18]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

In [19]:
```
##rename column
df.rename(columns= {'Marital_Status':'Shaadi'})
```

Out[19]:

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

11239 rows × 13 columns

In [20]:
```
##description of the data
df.describe()
```

Out[20]:

	User_ID	Age	Marital_Status	Orders	Amount
count	1.123900e+04	11239.000000	11239.000000	11239.000000	11239.000000
mean	1.003004e+06	35.410357	0.420055	2.489634	9453.610553
std	1.716039e+03	12.753866	0.493589	1.114967	5222.355168
min	1.000001e+06	12.000000	0.000000	1.000000	188.000000
25%	1.001492e+06	27.000000	0.000000	2.000000	5443.000000
50%	1.003064e+06	33.000000	0.000000	2.000000	8109.000000
75%	1.004426e+06	43.000000	1.000000	3.000000	12675.000000
max	1.006040e+06	92.000000	1.000000	4.000000	23952.000000

In [21]:
```
#use describe() for specific columns
df[['Age', 'Orders', 'Amount']].describe()
```

Out[21]:

	Age	Orders	Amount
count	11239.000000	11239.000000	11239.000000
mean	35.410357	2.489634	9453.610553
std	12.753866	1.114967	5222.355168
min	12.000000	1.000000	188.000000
25%	27.000000	2.000000	5443.000000
50%	33.000000	2.000000	8109.000000
75%	43.000000	3.000000	12675.000000
max	92.000000	4.000000	23952.000000

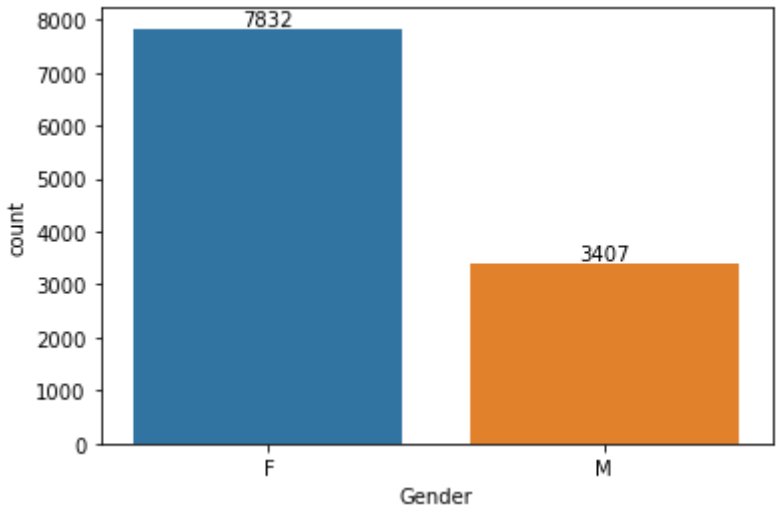
Exploratory Data Analysis

Gender

In [22]:
```
df.columns
```

Out[22]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

In [23]:
```
ax = sns.countplot(x = 'Gender' ,data = df)

for bars in ax.containers:
    ax.bar_label(bars)

## Created Containers to find out the total number of females and males
```



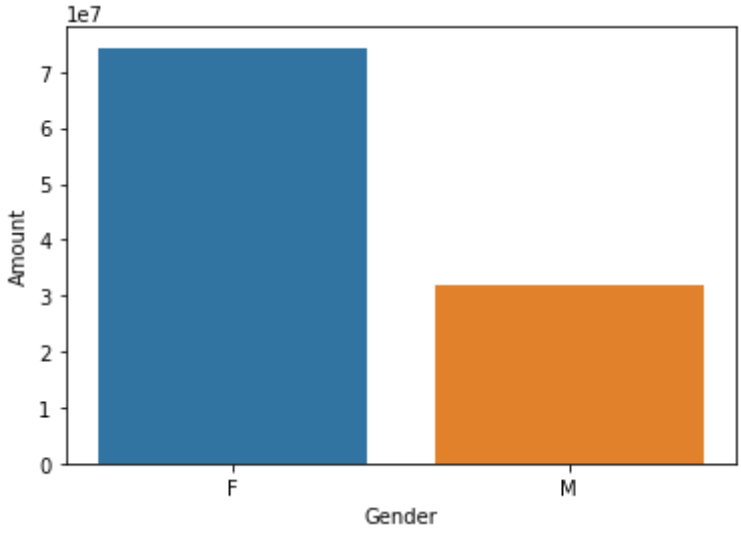
In [24]:
```
df.groupby(['Gender'], as_index=False)['Amount'].sum().sort_values(by='Amount', as
```

Out[24]:

	Gender	Amount
0	F	74335853
1	M	31913276

In [25]:
```
sales_gen = df.groupby(['Gender'], as_index=False)['Amount'].sum().sort_values(by=

sns.barplot(x = 'Gender', y ='Amount', data = sales_gen)
```

Out[25]:
```
<AxesSubplot:xlabel='Gender', ylabel='Amount'>
```



In [26]:
```
##Females are buying more than men.
```

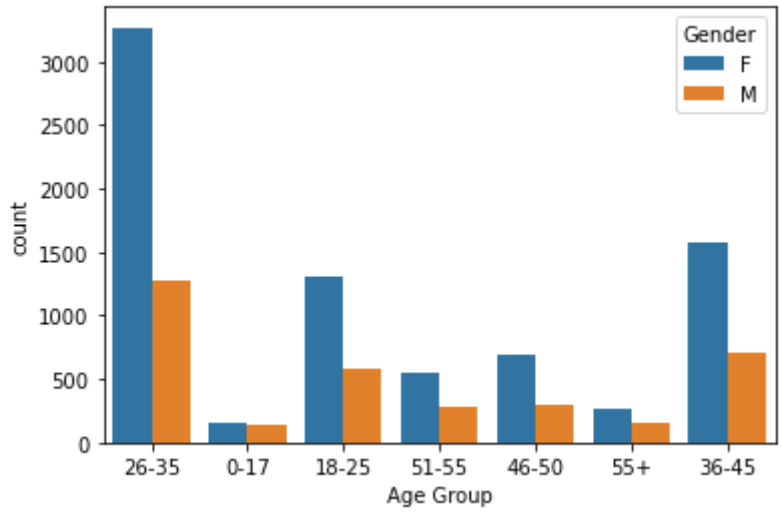
Age

In [27]:
```
df.columns
```

Out[27]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

In [28]:
```
sns.countplot(data = df, x = 'Age Group' , hue = 'Gender')
##If removed hue the difference of Men and Women will be lost and it won't show.
```

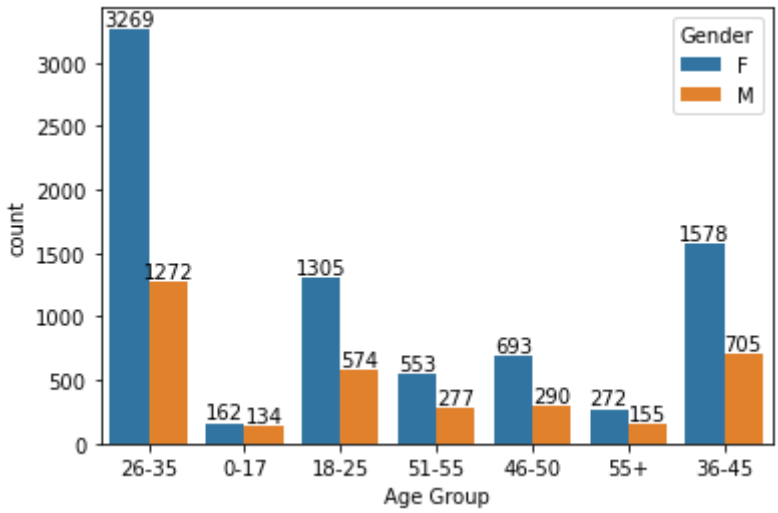
Out[28]:
```
<AxesSubplot:xlabel='Age Group', ylabel='count'>
```



Age

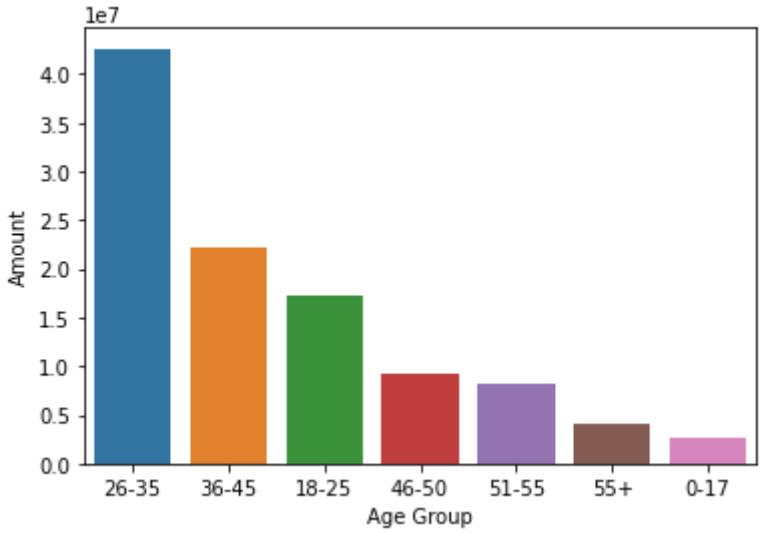
In [29]:
```
ax = sns.countplot(data = df, x = 'Age Group' , hue = 'Gender')
```

```
for bars in ax.containers:
    ax.bar_label(bars)
```



In [30]:
```
## Total Amount vs Age Group
sales_age = df.groupby(['Age Group'], as_index=False)['Amount'].sum().sort_values(

sns.barplot(x = 'Age Group', y= 'Amount', data = sales_age)
```

Out[30]:
```
<AxesSubplot:xlabel='Age Group', ylabel='Amount'>
```



From above graphs we can see that most of the buyers are of age group between 26-35 years female.

State

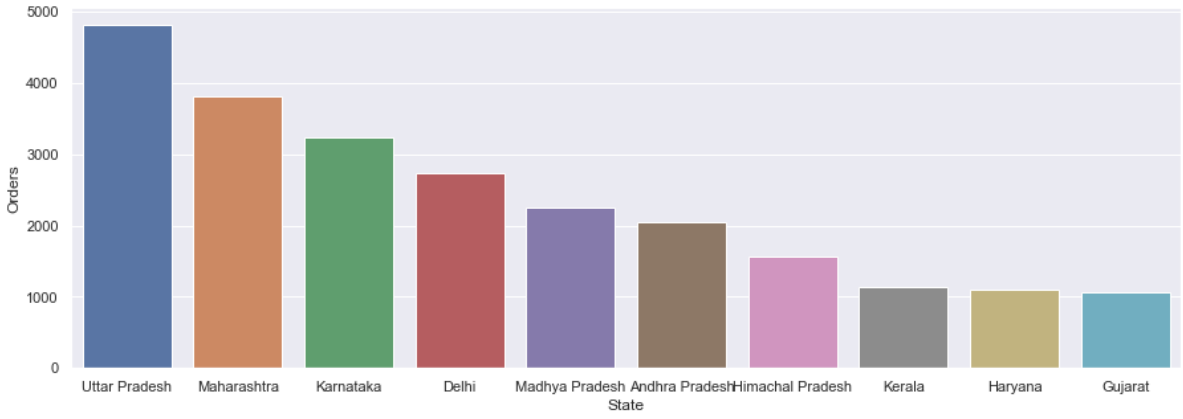
In [31]:
```
df.columns
```

Out[31]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

In [33]:
```
sales_state = df.groupby(['State'], as_index=False)['Orders'].sum().sort_values(by=

sns.set(rc={'figure.figsize':(15,5)})
##Making the Chart size properly so that it doesn't collates.
```

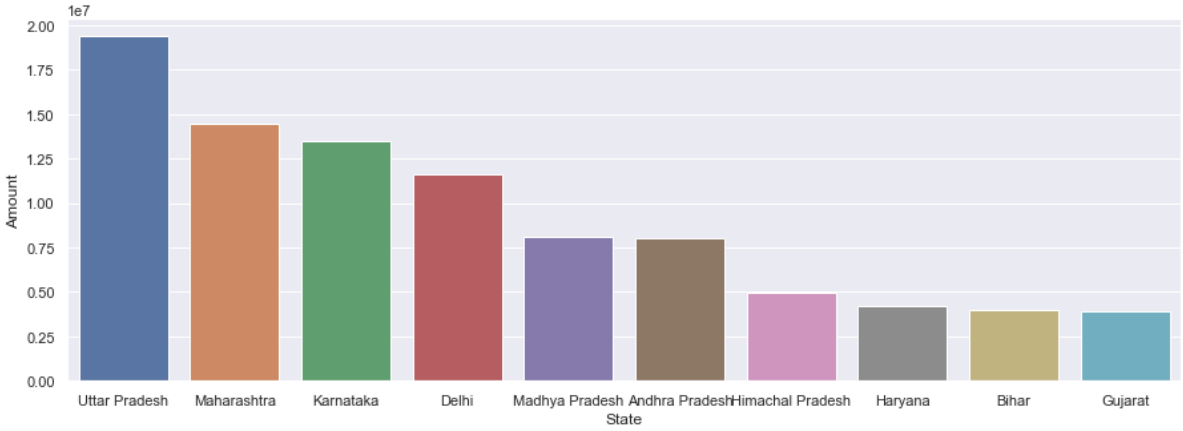
```
sns.barplot(data = sales_state, x = 'State', y = 'Orders')
```

Out[33]:
```
<AxesSubplot:xlabel='State', ylabel='Orders'>
```



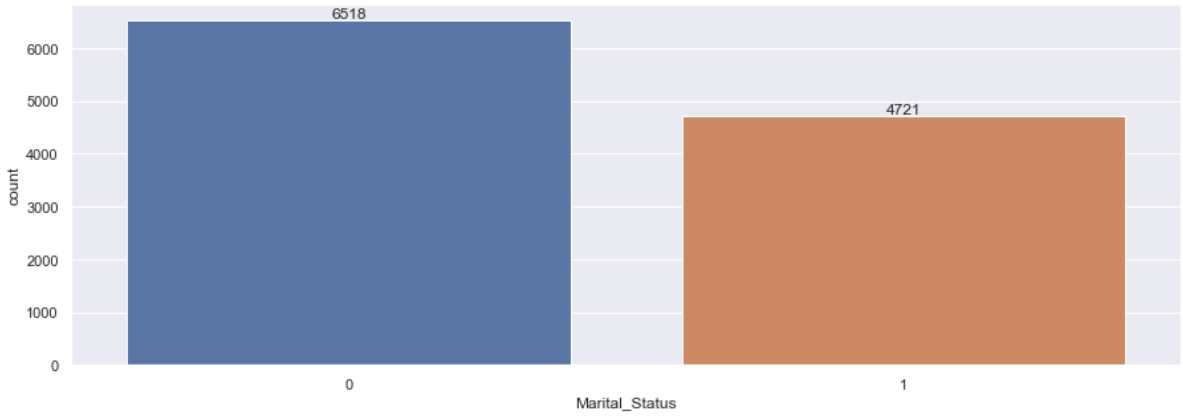
In [34]:
```
## Total Amount/Sales from Top 10 States

sales_state = df.groupby(['State'], as_index=False)['Amount'].sum().sort_values(by

sns.set(rc={'figure.figsize':(15,5)})
sns.barplot(data = sales_state, x = 'State', y = 'Amount')
```

Out[34]:
```
<AxesSubplot:xlabel='State', ylabel='Amount'>
```



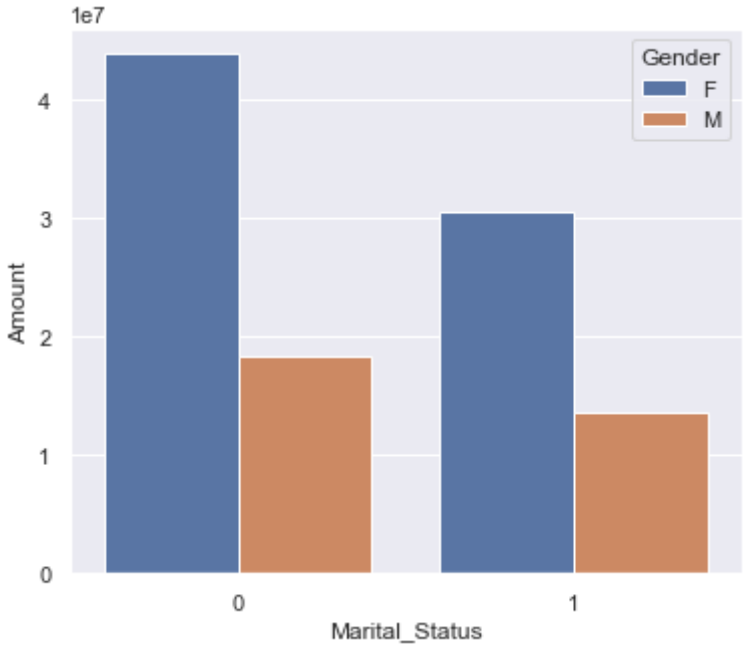
Marital Status

In [35]:
```
ax = sns.countplot(data = df, x = 'Marital_Status')

sns.set(rc={'figure.figsize':(7,5)})
for bars in ax.containers:
    ax.bar_label(bars)
```



In [36]:
```
sales_state = df.groupby(['Marital_Status', 'Gender'], as_index=False)['Amount'].su

sns.set(rc={'figure.figsize':(6,5)})
sns.barplot(data = sales_state, x = 'Marital_Status', y = 'Amount', hue = 'Gender'
```

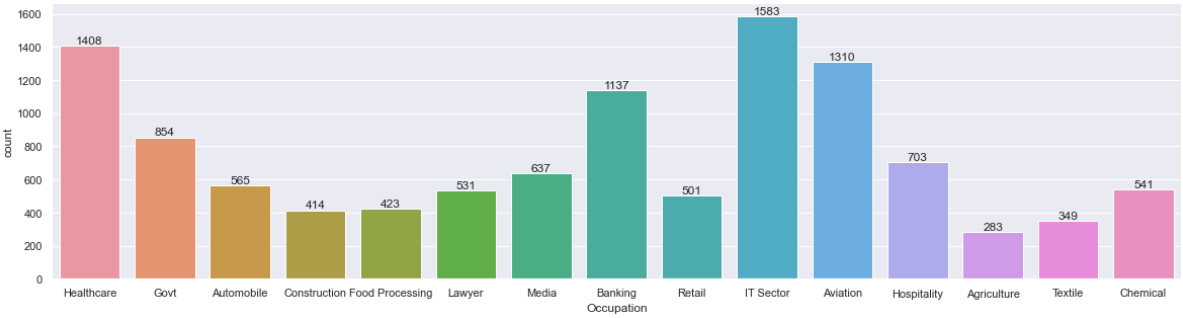
Out[36]:
```
<AxesSubplot:xlabel='Marital_Status', ylabel='Amount'>
```



From the above graphs we can see that most of the buyers are married (women) and they have high purchasing power.

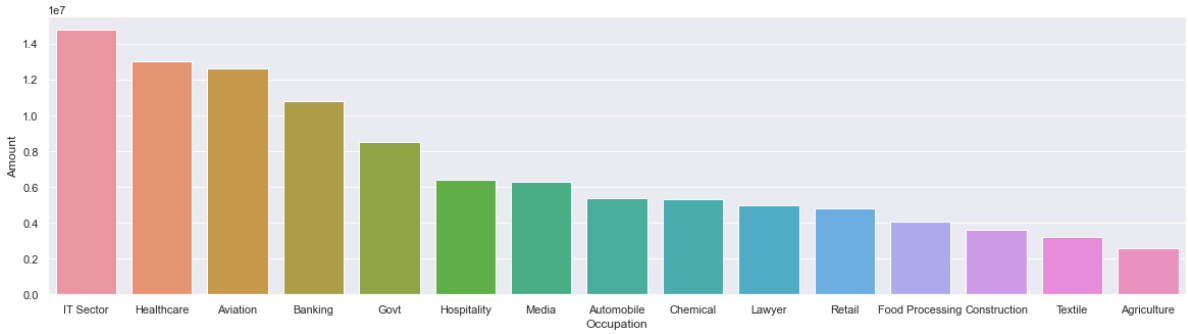
Occupation

In [38]:
```
sns.set(rc={'figure.figsize':(20,5)})
ax = sns.countplot(data = df, x = 'Occupation')

for bars in ax.containers:
    ax.bar_label(bars)
```



In [40]:
```
sales_state = df.groupby(['Occupation'], as_index=False)['Amount'].sum().sort_valu

sns.set(rc={'figure.figsize':(20,5)})
sns.barplot(data = sales_state, x = 'Occupation', y = 'Amount')
```

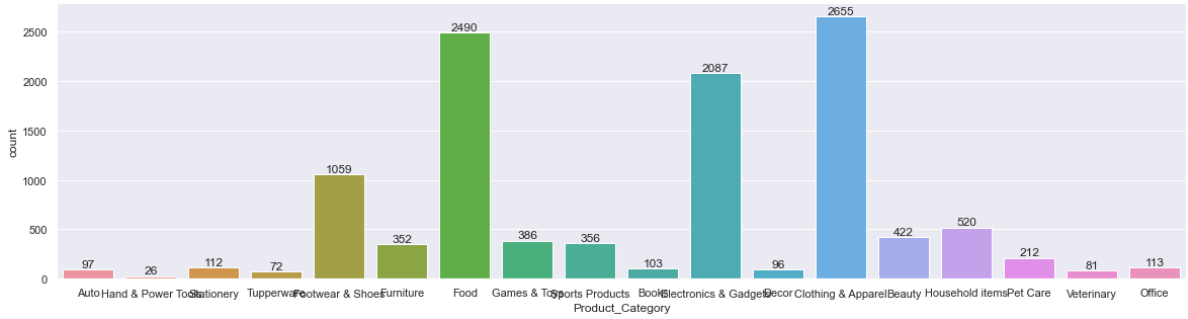
Out[40]:
```
<AxesSubplot:xlabel='Occupation', ylabel='Amount'>
```



Most of the buyers are working in IT Sector, Healthcare, Aviation.

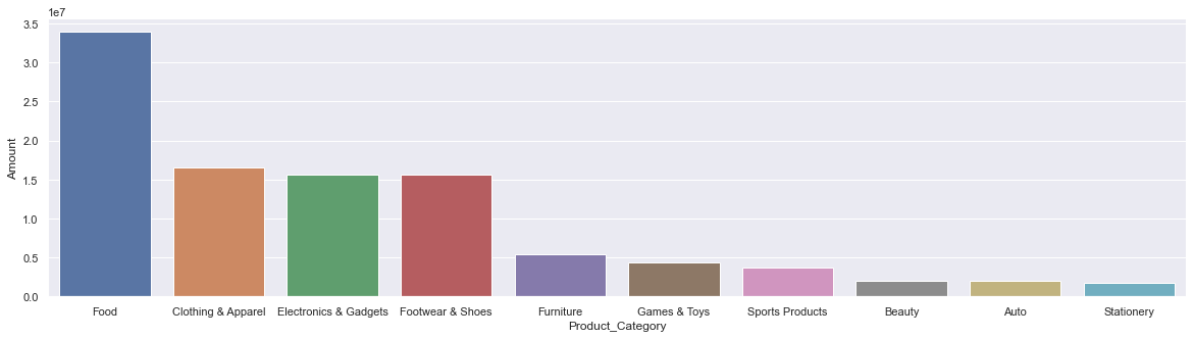
Product Category

In [41]:
```
sns.set(rc={'figure.figsize':(20,5)})
ax = sns.countplot(data = df, x = 'Product_Category')


for bars in ax.containers:
    ax.bar_label(bars)
```



In [45]:
```
sales_state = df.groupby(['Product_Category'], as_index=False)['Amount'].sum().sor

sns.set(rc={'figure.figsize':(20,5)})
sns.barplot(data = sales_state, x = 'Product_Category', y = 'Amount')
```

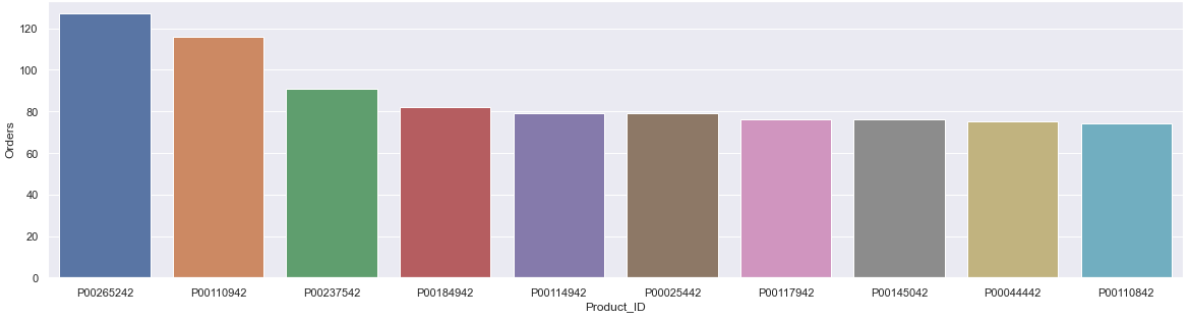
Out[45]:
```
<AxesSubplot:xlabel='Product_Category', ylabel='Amount'>
```



Most of the sold Products are from Food, Clothing and Electronics Category

In [46]:
```
sales_state = df.groupby(['Product_ID'], as_index=False)['Orders'].sum().sort_valu

sns.set(rc={'figure.figsize':(20,5)})
sns.barplot(data = sales_state, x = 'Product_ID', y = 'Orders')
```

Out[46]:
```
<AxesSubplot:xlabel='Product_ID', ylabel='Orders'>
```



Conclusion

Married women age group 26-35 yrs from UP, Maharashtra and Karnataka working in IT, Healthcare and Aviation are more likely to buy products from Food, Clothing and Electronics Category.

In [ ]: