In [1]:
```
# Importing Python Libraries

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

In [3]:
```
# Loading data into a Jupyter Notebook

sales = pd.read_excel("D:\Business Analytics\Python Projects\Deepavali Sales Analysis\
```

In [4]:
```
# Understanding the Data and defining the data

sales.shape # shape will provide the numbers of row and columns are there in dataset
```

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

In [5]:
```
sales.head() # head() will return the first five rows including all variables, if we m
```

Out[5]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	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	C
4	1000588	Joni	P00057942	M	26-35	28	1	Gujarat	Western	

In [6]:
```
sales.tail() # tail() will return the last 5 rows, as like head() function.
```

Out[6]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	State	Zone
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

In [7]:
```
sales.head(10)
```

Out[7]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	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
5	1000588	Joni	P00057942	M	26-35	28	1	Himachal Pradesh	Northern
6	1001132	Balk	P00018042	F	18-25	25	1	Uttar Pradesh	Central
7	1002092	Shivangi	P00273442	F	55+	61	0	Maharashtra	Western
8	1003224	Kushal	P00205642	M	26-35	35	0	Uttar Pradesh	Central
9	1003650	Ginny	P00031142	F	26-35	26	1	Andhra Pradesh	Southern

In [8]:
```
sales.info() # it display the information about the dataset
```

```
<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 [9]:
```
sales.columns # it display all the columns in the dataset
```

Out[9]:
```
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 [10]:
```
# in our dataset we have 2 unwanted columns so we can drop them

sales.drop(['Status', 'unnamed1'], axis=1, inplace=True) # axis=1 means it works on ve
```

In [11]:
```
# now we can check those columns in dataset in 2 way

sales.columns
```

Out[11]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

In [12]:
```
sales.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 [13]:
```
# now we can check is there any null values is present in dataset

pd.isnull(sales).sum()
```

Out[13]:
```
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 [14]:
```
# In our dataset has 12 null values in amount column that means,
# Customer has selected the product and kept in cart but not done payment so order prc
# We have very less null values we can avoid them by deleting

sales.dropna(inplace=True)
```

In [15]:
```
# we can confirm them wheather the null values are deleted or not

sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11239 entries, 0 to 11250
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   User_ID           11239 non-null  int64  
 1   Cust_name         11239 non-null  object 
 2   Product_ID        11239 non-null  object 
 3   Gender            11239 non-null  object 
 4   Age Group         11239 non-null  object 
 5   Age               11239 non-null  int64  
 6   Marital_Status    11239 non-null  int64  
 7   State             11239 non-null  object 
 8   Zone              11239 non-null  object 
 9   Occupation        11239 non-null  object 
 10  Product_Category  11239 non-null  object 
 11  Orders            11239 non-null  int64  
 12  Amount            11239 non-null  float64
dtypes: float64(1), int64(4), object(8)
memory usage: 1.2+ MB
```

In [16]:
```
pd.isnull(sales).sum()
```

Out[16]:
```
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 [17]:
```
# In our dataset we can see the data type of the 'Amount' variable is 'float64',
# we are not going to deal with any decimal value so can chenge into 'integer'

sales['Amount'] = sales['Amount'].astype('int32')
```

In [18]:
```
sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11239 entries, 0 to 11250
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   User_ID           11239 non-null  int64 
 1   Cust_name         11239 non-null  object
 2   Product_ID        11239 non-null  object
 3   Gender            11239 non-null  object
 4   Age Group         11239 non-null  object
 5   Age               11239 non-null  int64 
 6   Marital_Status    11239 non-null  int64 
 7   State             11239 non-null  object
 8   Zone              11239 non-null  object
 9   Occupation        11239 non-null  object
 10  Product_Category  11239 non-null  object
 11  Orders            11239 non-null  int64 
 12  Amount            11239 non-null  int32 
dtypes: int32(1), int64(4), object(8)
memory usage: 1.2+ MB
```

In [19]:
```
# We can also check each individual dtypes

sales['Amount'].dtypes
```

Out[19]:
```
dtype('int32')
```

In [20]:
```
sales.describe() # it display the description of the data in a DataFrame
```

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]:
```
# we can also check this description with numerical data by mentioning them in square

sales[['Age','Orders','Amount']].describe()
```

Out[21]:

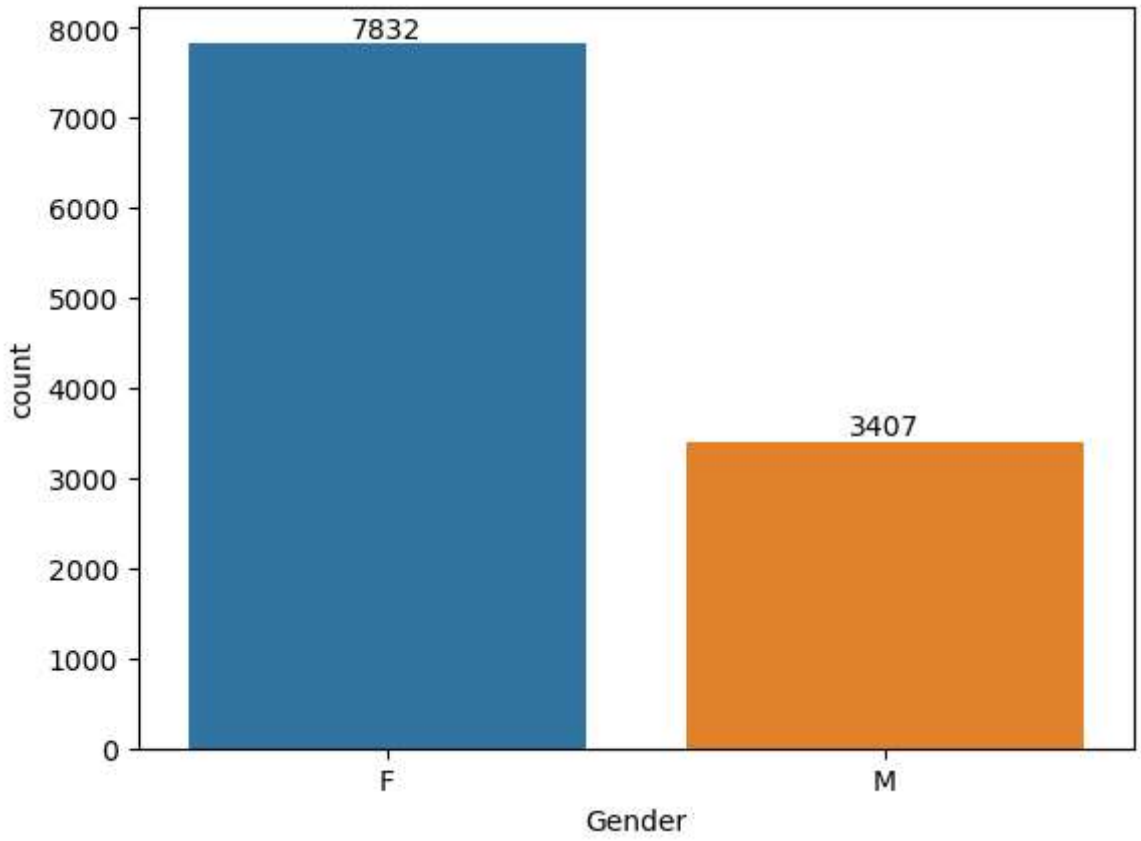
	Age	Orders	Amount
count	11239.000000	11239.000000	11239.000000
mean	35.410357	2.489634	9453.610553
std	12.753866	1.114967	5222.355168
min	12.000000	1.000000	188.000000
25%	27.000000	2.000000	5443.000000
50%	33.000000	2.000000	8109.000000
75%	43.000000	3.000000	12675.000000
max	92.000000	4.000000	23952.000000

now our data is ready to perform EDA

Exploratory Data Analysis

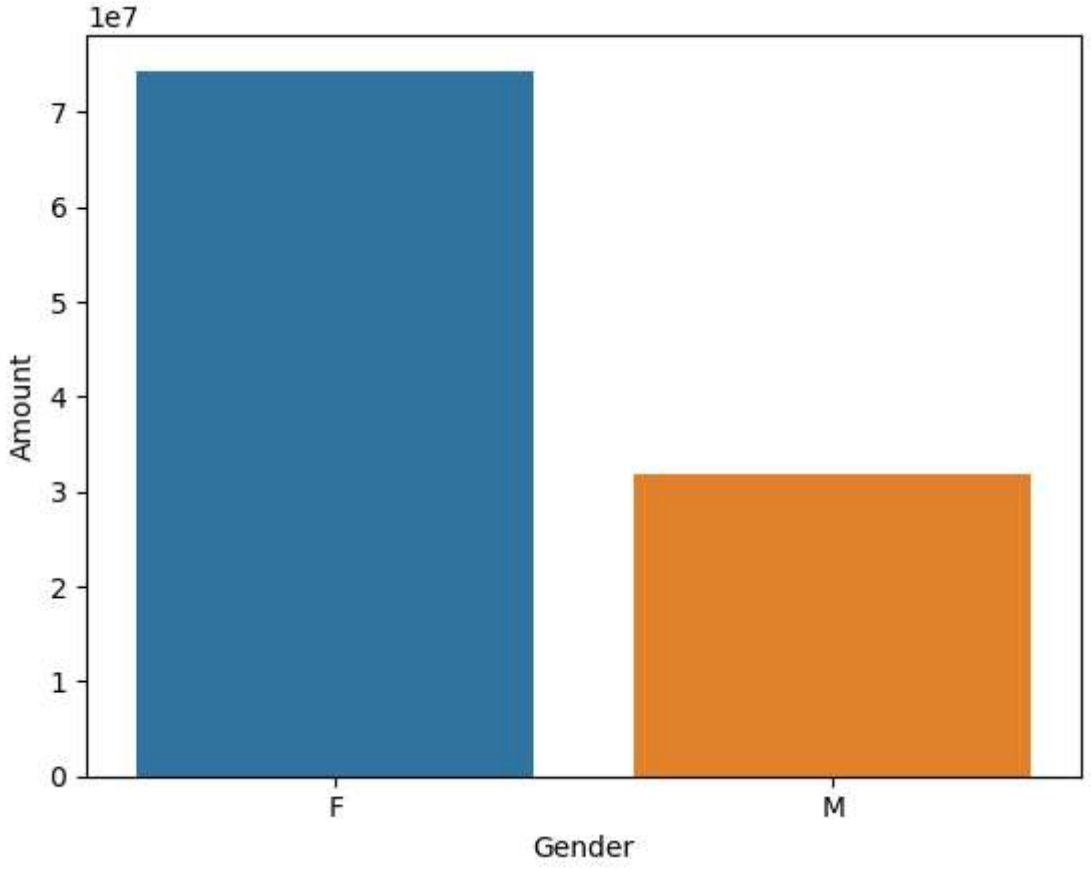
Gender Category

In [22]:
```
# plot the bar chart on gender category with total count

barchart=sns.countplot(x='Gender', data=sales)
for bars in barchart.containers:
    barchart.bar_label(bars)
```



In [24]:
```
# plotting the bar chart on Gender category and the amount they have spend in shopping

sales_gender = sales.groupby(['Gender'], as_index=False)['Amount'].sum().sort_values(b
sns.barplot(x='Gender', y='Amount', data=sales_gender)
```

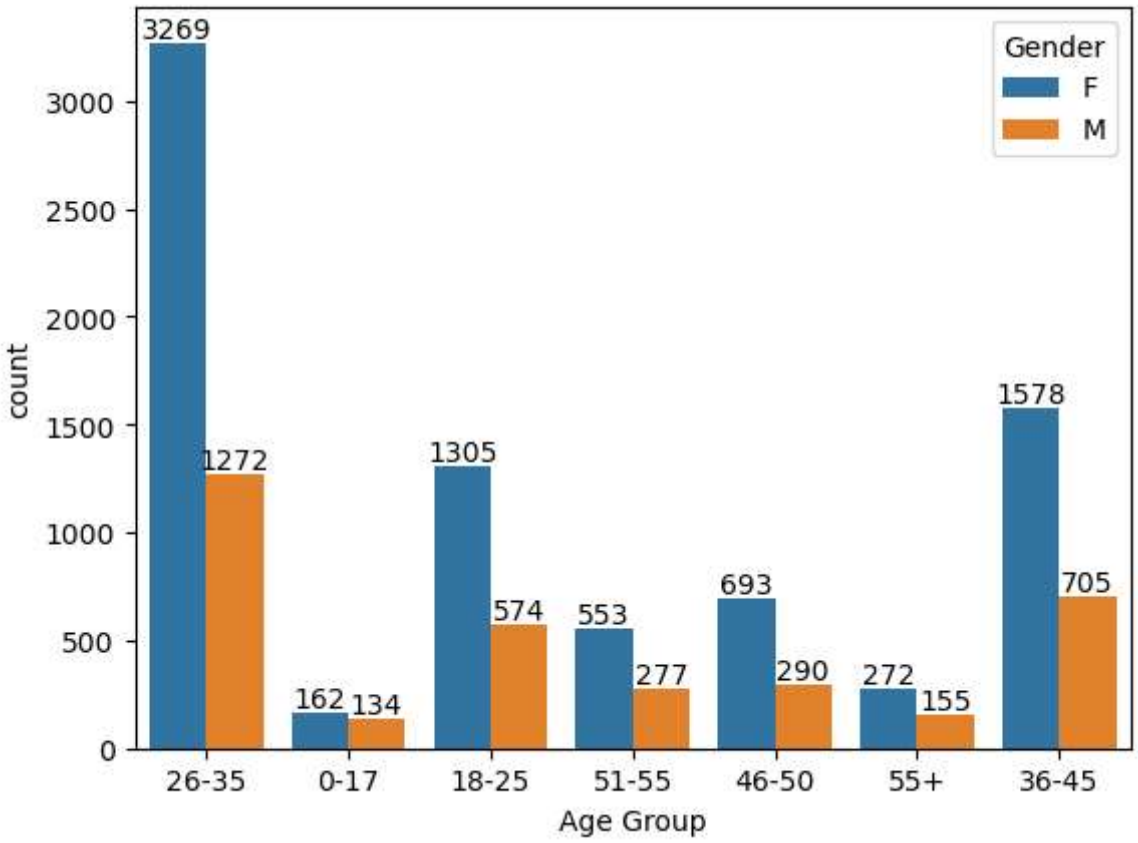
Out[24]:
```
<AxesSubplot:xlabel='Gender', ylabel='Amount'>
```



from the above graph we can see that most of the buyers are females and the purchasing power of the females are greater than men.

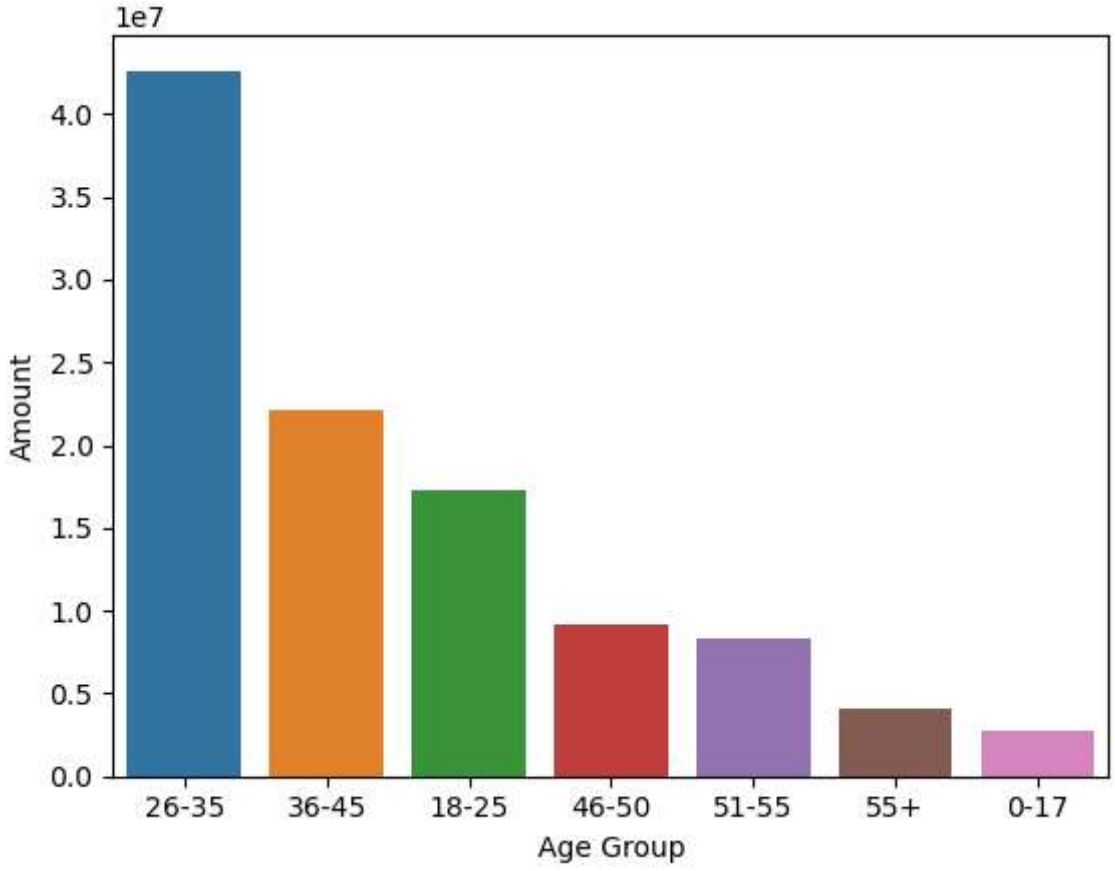
Age Group

In [25]:
```
age_group = sns.countplot(data=sales, x='Age Group', hue='Gender')
for bars in age_group.containers:
    age_group.bar_label(bars)
```



In [26]:
```
# now we can see that total amount spent by each age group

sales_age = sales.groupby(['Age Group'], as_index=False)['Amount'].sum().sort_values(b
sns.barplot(x='Age Group', y='Amount', data=sales_age)
```

Out[26]:
```
<AxesSubplot:xlabel='Age Group', ylabel='Amount'>
```

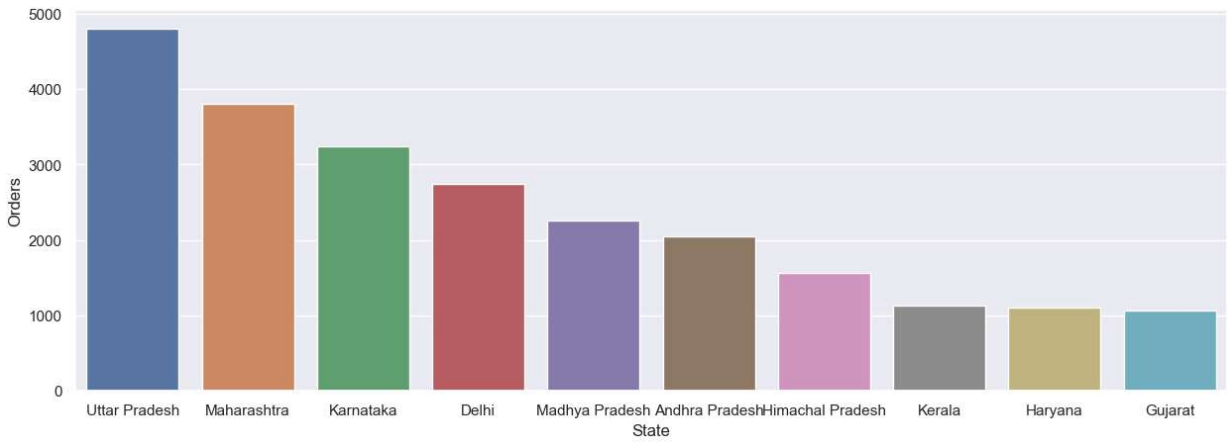


From above graphs we can see that most of the buyers are of age group between 26-35 yrs female

State wise total orders and sales

In [29]:
```
# now can calculate the top 10 states have made total number of orders

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

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

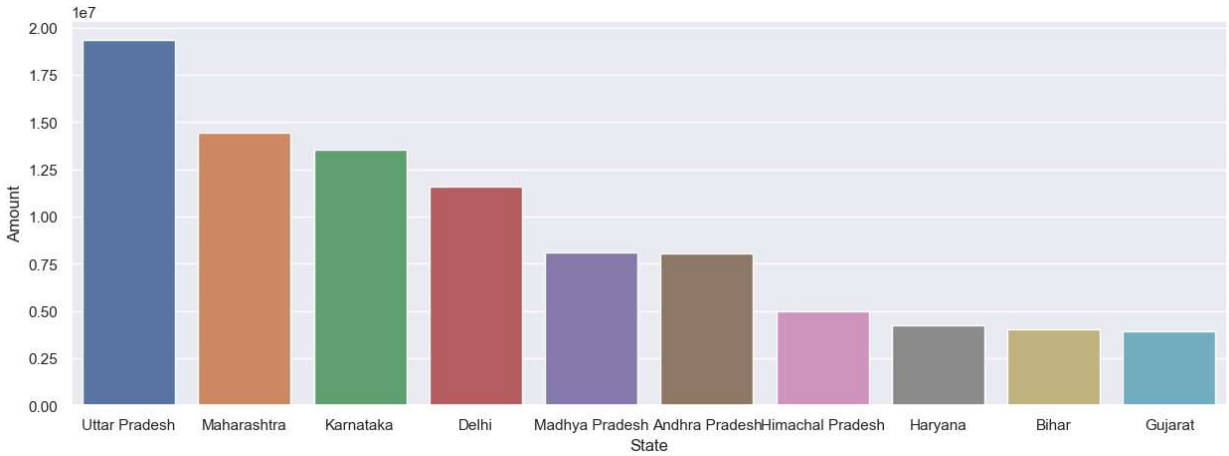
Out[29]:
```
<AxesSubplot:xlabel='State', ylabel='Orders'>
```



from the above graph we can see that highest orders are from UP followed by MH and KA

In [32]:
```
# total amount/sales from top 10 states

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

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

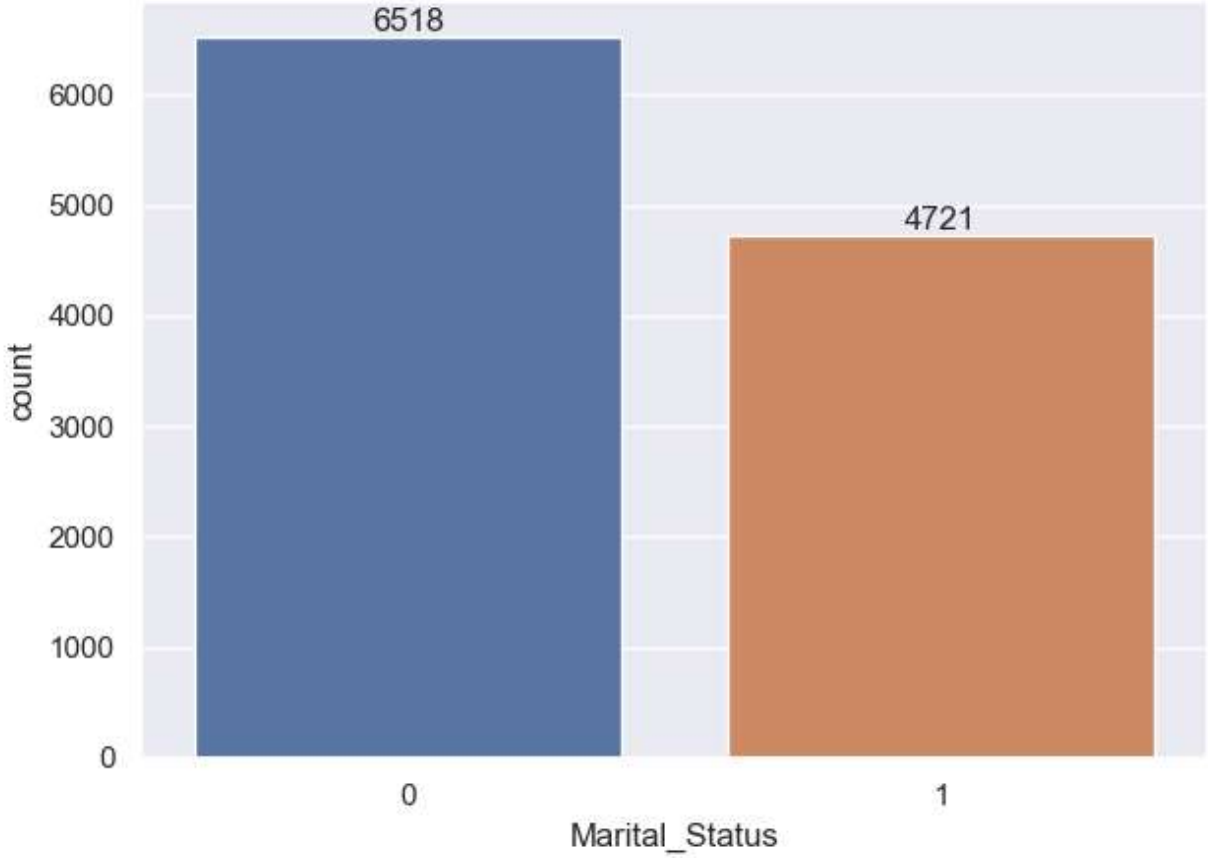
Out[32]:
```
<AxesSubplot:xlabel='State', ylabel='Amount'>
```



from the total numbers of orders have made by top 10 state graph we can see that Kerala has 8th place but, in the total amount spent by top 10 graph Kerala has not even in the 10th place.

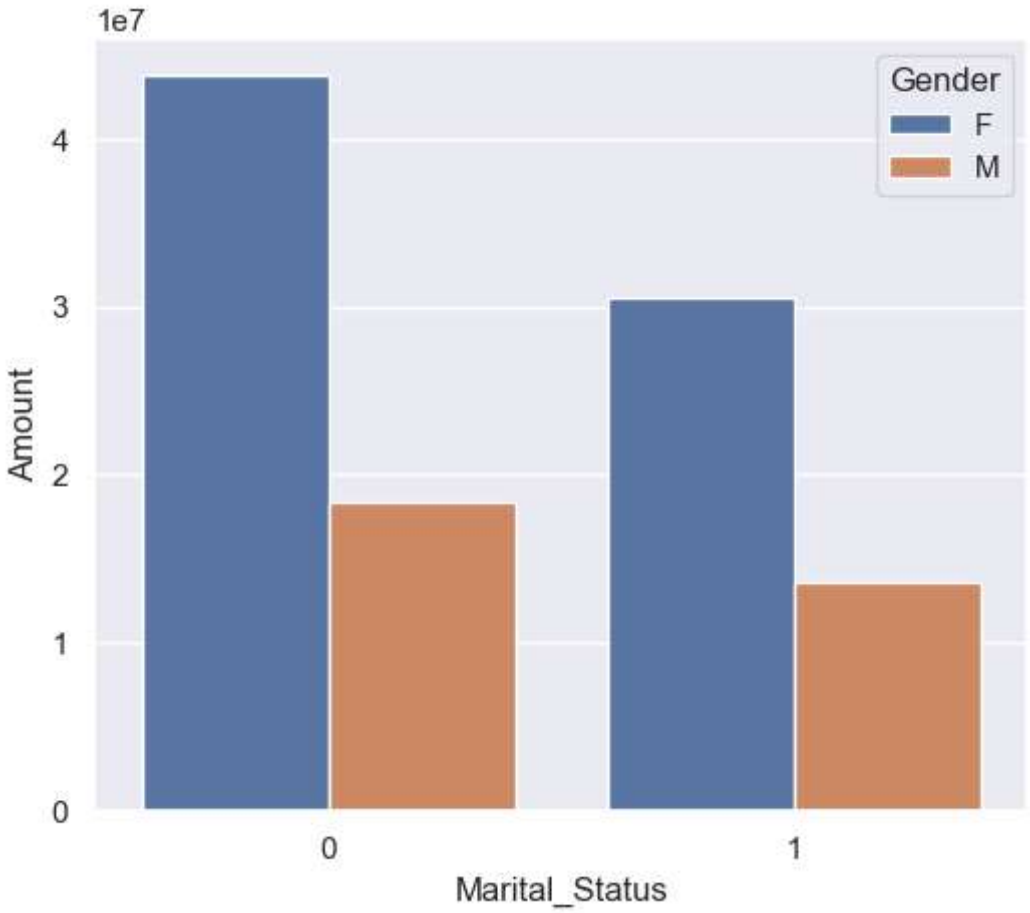
Marital Status

In [34]:
```
marital_status = sns.countplot(x="Marital_Status", data=sales)
sns.set(rc={'figure.figsize':(3,5)})
for bars in marital_status.containers:
    marital_status.bar_label(bars)
```



In [35]:
```
sales_status = sales.groupby(['Marital_Status','Gender'], as_index=False)['Amount'].su
sns.set(rc={'figure.figsize':(6,5)})
sns.barplot(data=sales_status, x='Marital_Status', y='Amount',hue='Gender')
```

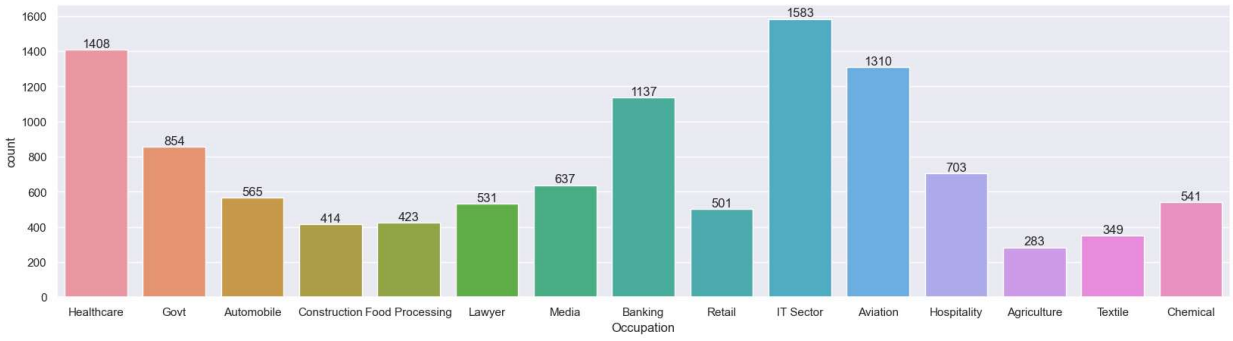
Out[35]:
```
<AxesSubplot:xlabel='Marital_Status', ylabel='Amount'>
```



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

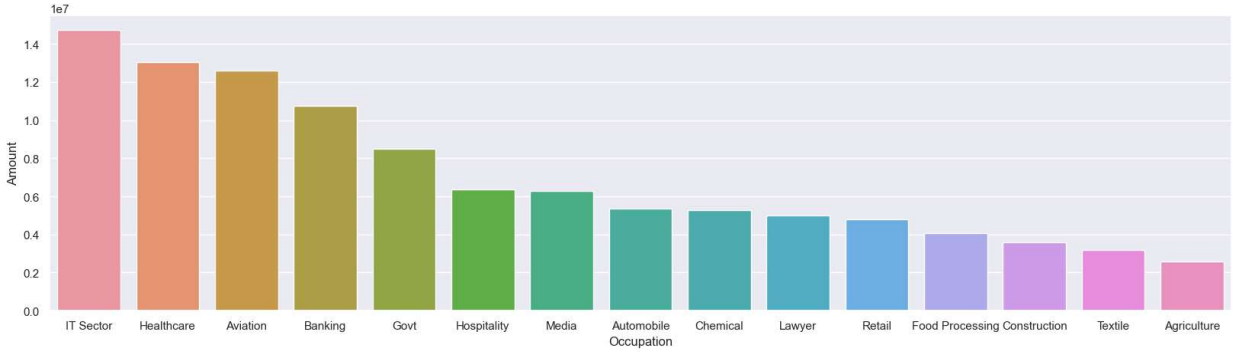
Occupation

In [37]:
```
sns.set(rc={'figure.figsize':(20,5)})
ocp = sns.countplot(data=sales, x='Occupation')
for bars in ocp.containers:
    ocp.bar_label(bars)
```



In [38]:
```
Ocp_total_sales = sales.groupby(['Occupation'], as_index=False)['Amount'].sum().sort_v
sns.barplot(data=Ocp_total_sales, x='Occupation',y='Amount')
```

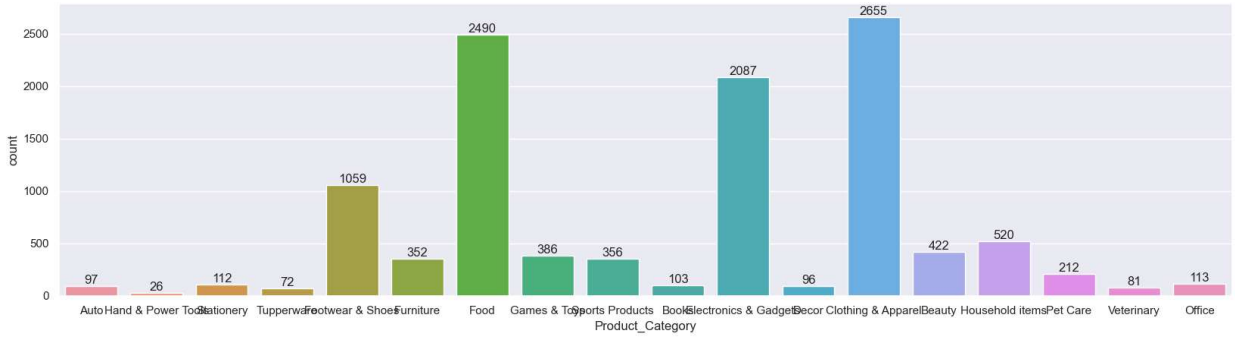
Out[38]:
```
<AxesSubplot:xlabel='Occupation', ylabel='Amount'>
```



from the above graph we can see that most of the buyers are working in IT Sector followed by Healthcare and Aviation

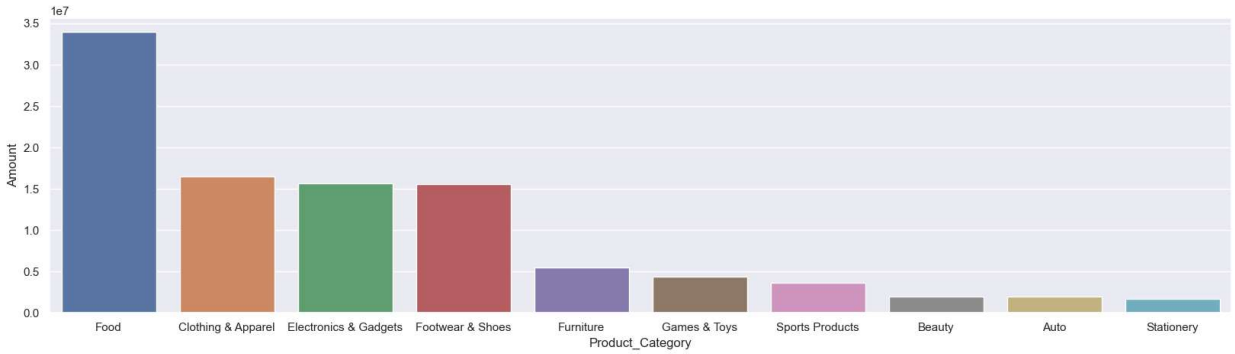
Product Category

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

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



In [44]:
```
product_total_sales = sales.groupby(['Product_Category'], as_index=False)['Amount'].su
sns.set(rc={'figure.figsize':(20,5)})
sns.barplot(data=product_total_sales, x='Product_Category', y='Amount')
```

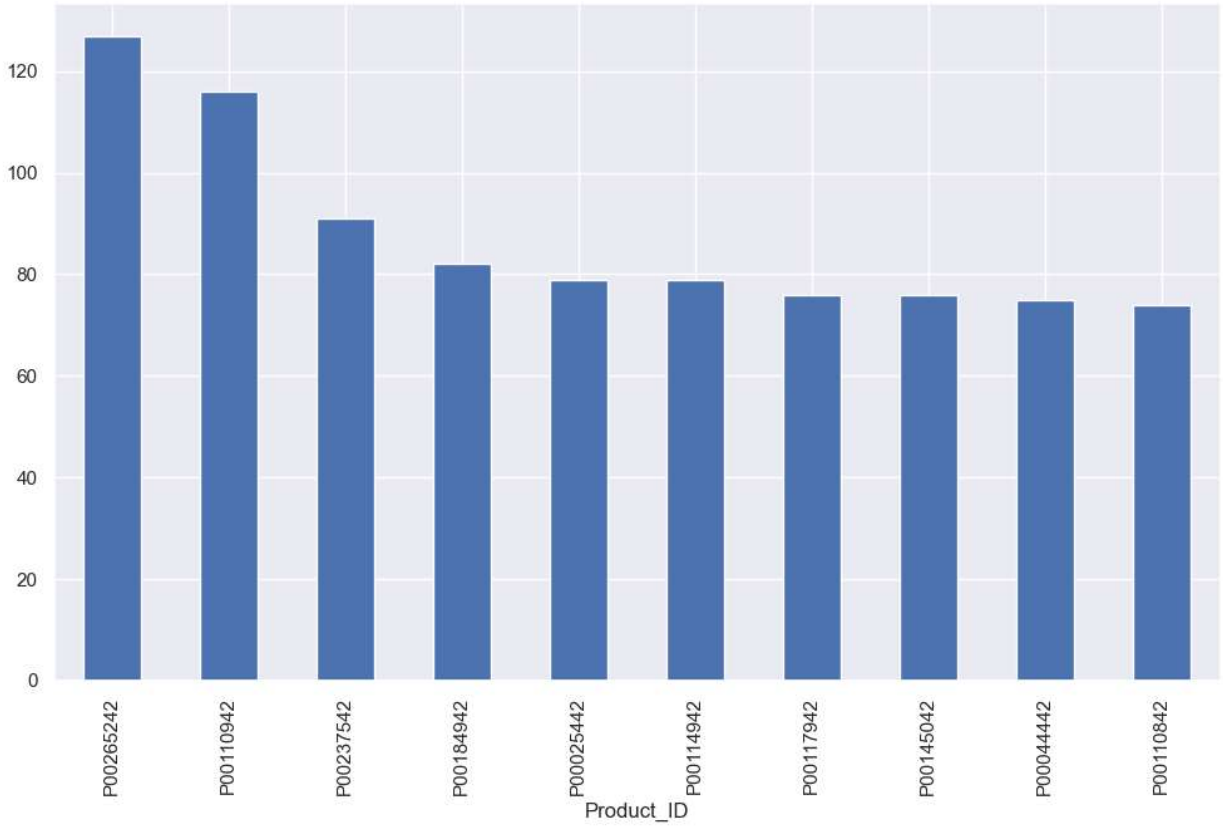
Out[44]:
```
<AxesSubplot:xlabel='Product_Category', ylabel='Amount'>
```



from the above graph can see that most of the sold products from the Food Category followed by Clothing ad Electronics

In [45]:
```
# top10 most sold products

fig1, ax1= plt.subplots(figsize=(12,7))
sales.groupby('Product_ID')['Orders'].sum().nlargest(10).sort_values(ascending=False).
```

Out[45]:
```
<AxesSubplot:xlabel='Product_ID'>
```



Conclusion:

Who are more likely to buy:

- Married women age group 26-35 yrs and they have high purchasing power
- They are from UP, Maharastra and Karnataka
- They are working in IT, Healthcare and Aviation
- Most sold products are from Food, Clothing and Electronics category