In [32]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('diwali.csv', encoding= 'unicode_escape')
```

In [3]:
```
df.head(10)
```

Out[3]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	State	Zone	C
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	
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 [4]:
```
df.info()
#from indfo we can see that 2 columns are empty
```

```
<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 [5]:
```
#to drop the empty columns from table
df.drop(['Status', 'unnamed1'], axis=1, inplace = True)
df.head(10)
```

Out[5]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	State	Zone	C
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	
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 [6]:
```
pd.isnull(df).sum()
#it shows there are 12 null values in amout column
```

Out[6]:
```
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 [7]:
```
#drop the null values
df.dropna(inplace=True)
df.head(10)
#all the columns are removed which had null values
```

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
8	1003224	Kushal	P00205642	M	26-35	35	0	Uttar Pradesh	Central
9	1003650	Ginny	P00031142	F	26-35	26	1	Andhra Pradesh	Southern
10	1003829	Harshita	P00200842	M	26-35	34	0	Delhi	Central

In [8]:
```
#convert 'Amount' column from floating to whole number
df['Amount'] = df['Amount'].astype('int')
df.head(10)
```

Out[8]:

	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
8	1003224	Kushal	P00205642	M	26-35	35	0	Uttar Pradesh	Central
9	1003650	Ginny	P00031142	F	26-35	26	1	Andhra Pradesh	Southern
10	1003829	Harshita	P00200842	M	26-35	34	0	Delhi	Central

In [9]:
```
df.columns
```

Out[9]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

In [10]:
```
#to change the column name
df.rename(columns={'Age':'Age_category'}, inplace=True)
df.columns
```

Out[10]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group',
       'Age_category', 'Marital_Status', 'State', 'Zone', 'Occupation',
       'Product_Category', 'Orders', 'Amount'],
      dtype='object')
```

In [11]:
```
# describe() method returns description of the data in the DataFrame (i.e. count, mean
df.describe()
```

Out[11]:

	User_ID	Age_category	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

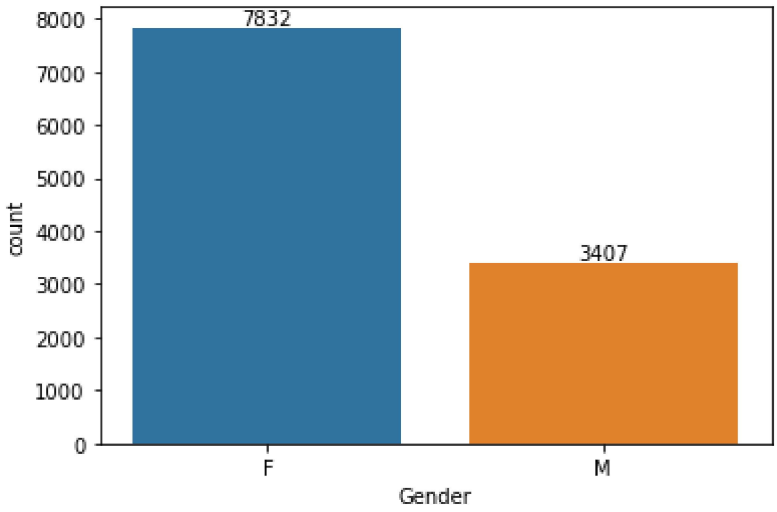
In [12]:
```
# use describe() for specific columns
df[['Age_category', 'Orders', 'Amount']].describe()
```

Out[12]:

	Age_category	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

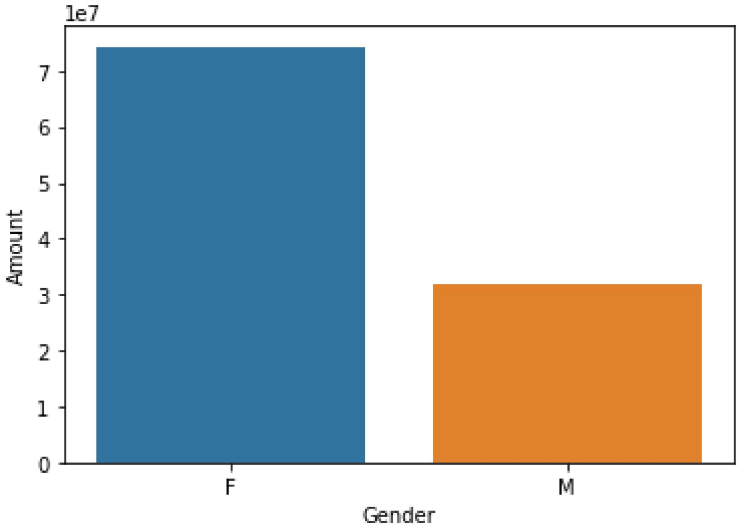
EDA

In [13]:
```
#GENDER
# plotting a bar chart for Gender and it's count
ax = sns.countplot(x='Gender', data=df)

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



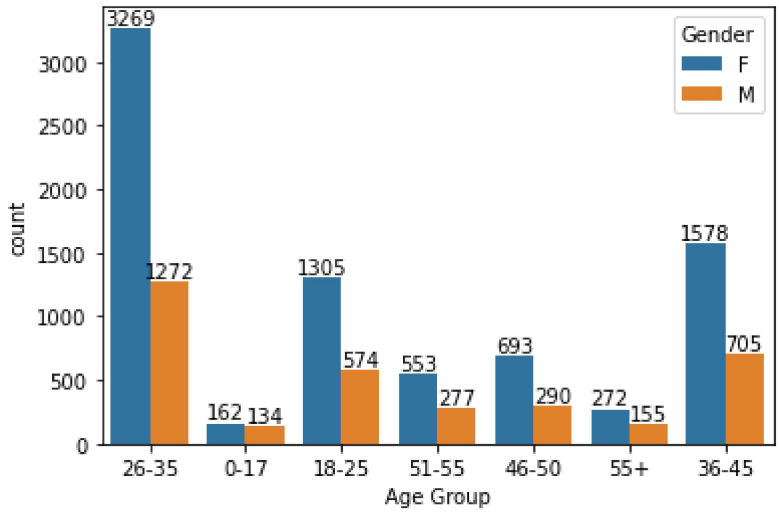
In [15]:
```
# plotting a bar chart for gender vs total amount
sales_gen = df.groupby(['Gender'], as_index=False)['Amount'].sum().sort_values(by='Amo
sns.barplot(x='Gender', y='Amount', data=sales_gen)
```

Out[15]:
```
<AxesSubplot:xlabel='Gender', ylabel='Amount'>
```



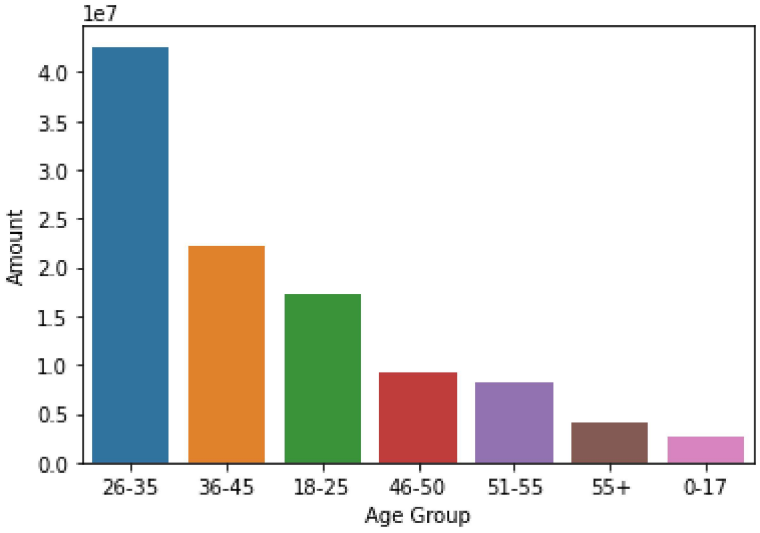
From above graphs we can see that most of the buyers are females and even the purchasing power of females are greater than men

In [16]:
```
#AGE
ax = sns.countplot(data=df, x='Age Group', hue='Gender')

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



In [18]:
```
# Total Amount vs Age Group
sales_age = df.groupby(['Age Group'], as_index=False)['Amount'].sum().sort_values(by='

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

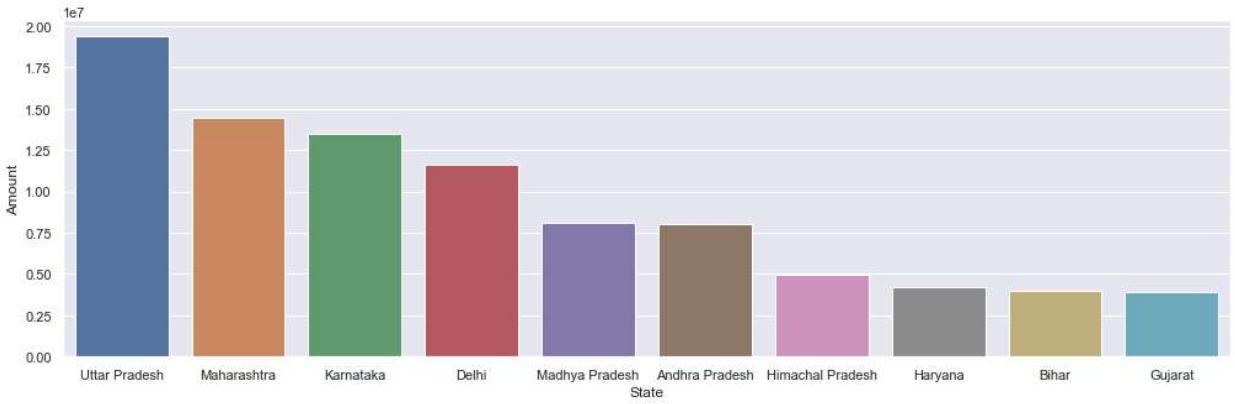
Out[18]:
```
<AxesSubplot:xlabel='Age Group', ylabel='Amount'>
```



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

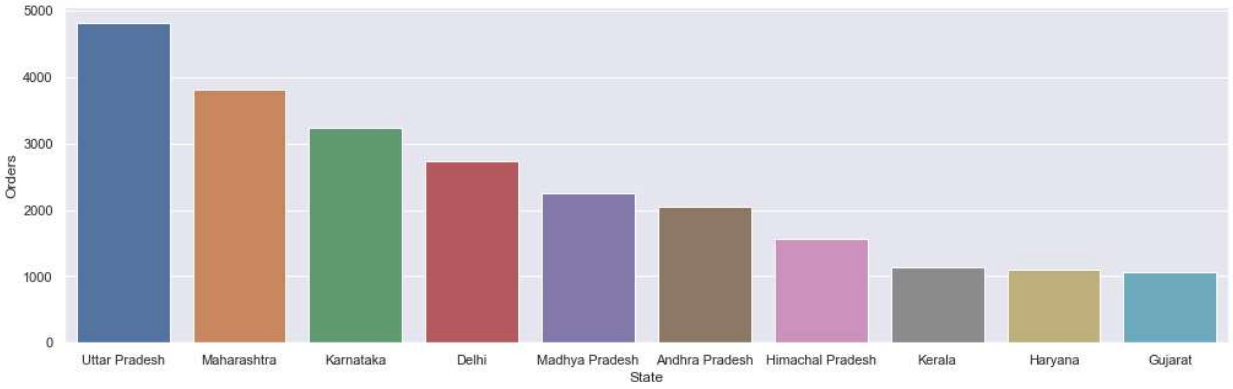
In [21]:
```
#STATE
# total amount/sales from top 10 states
sales_state = df.groupby(['State'], as_index=False)['Amount'].sum().sort_values(by='Am
sns.set(rc={'figure.figsize':(17,5)})
sns.barplot(data=sales_state, x='State', y='Amount')
```

Out[21]:
```
<AxesSubplot:xlabel='State', ylabel='Amount'>
```



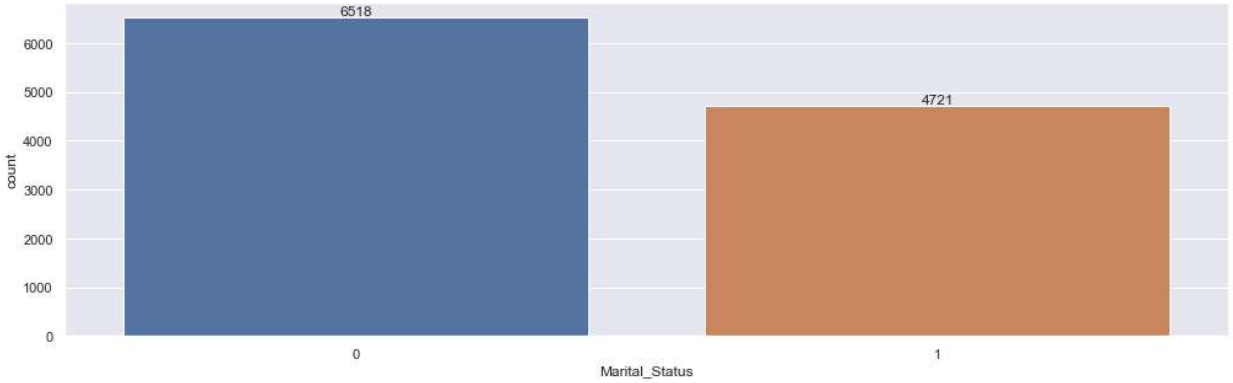
In [23]:
```
# total number of orders from top 10 states
sales_state = df.groupby(['State'], as_index=False)['Orders'].sum().sort_values(by='Or
sns.set(rc={'figure.figsize':(17,5)})
sns.barplot(data=sales_state, x='State', y='Orders')
```

Out[23]:
```
<AxesSubplot:xlabel='State', ylabel='Orders'>
```



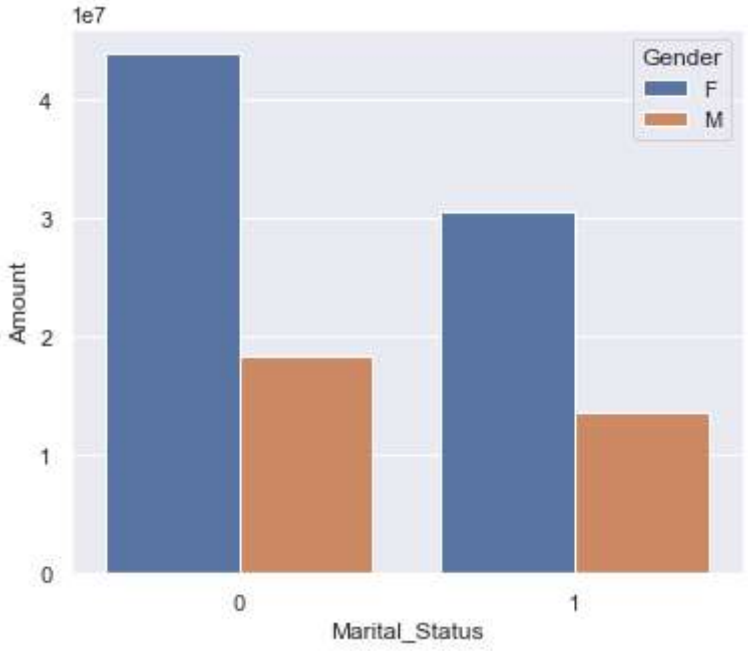
From above graphs we can see that most of the orders & total sales/amount are from Uttar Pradesh, Maharashtra and Karnataka respectively

In [24]:
```
#MARTIAL STATUS
ax = sns.countplot(data = df, x = 'Marital_Status')

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



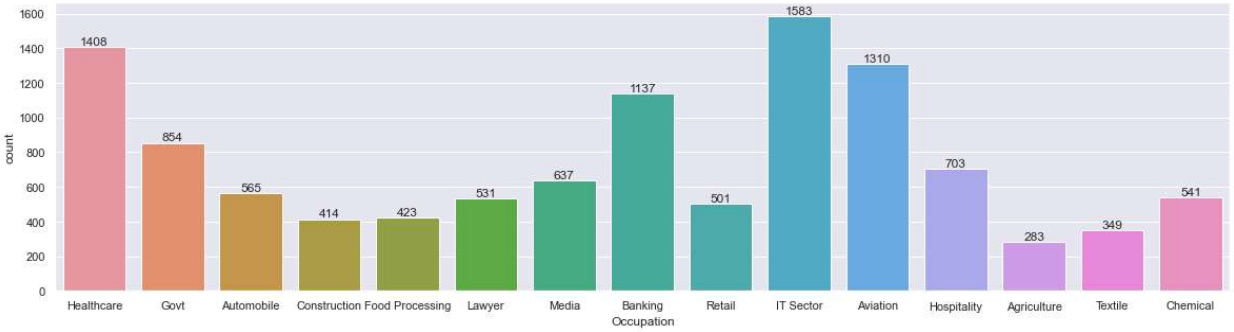
In [25]:
```
sales_state = df.groupby(['Marital_Status', 'Gender'], as_index=False)['Amount'].sum()

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

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



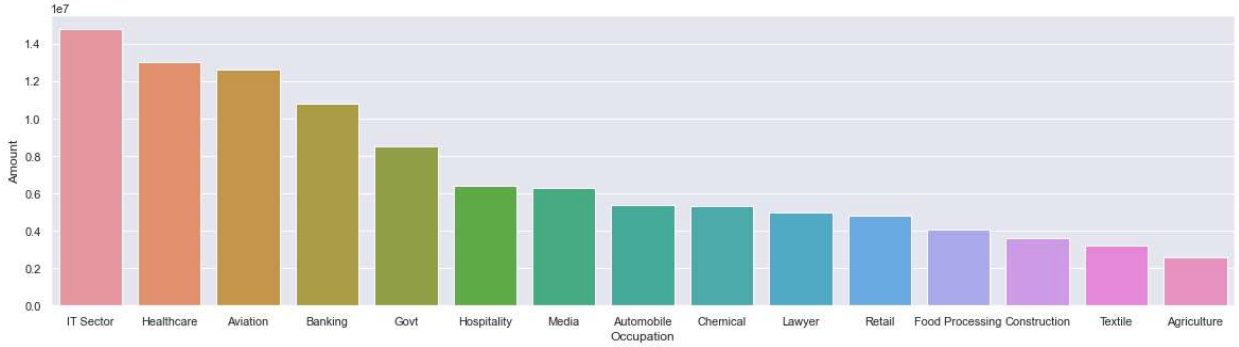
From above graphs we can see that most of the buyers are married (women) and they have high purchasing power

In [26]:
```
#OCCUPATION
sns.set(rc={'figure.figsize':(20,5)})
ax = sns.countplot(data = df, x = 'Occupation')

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



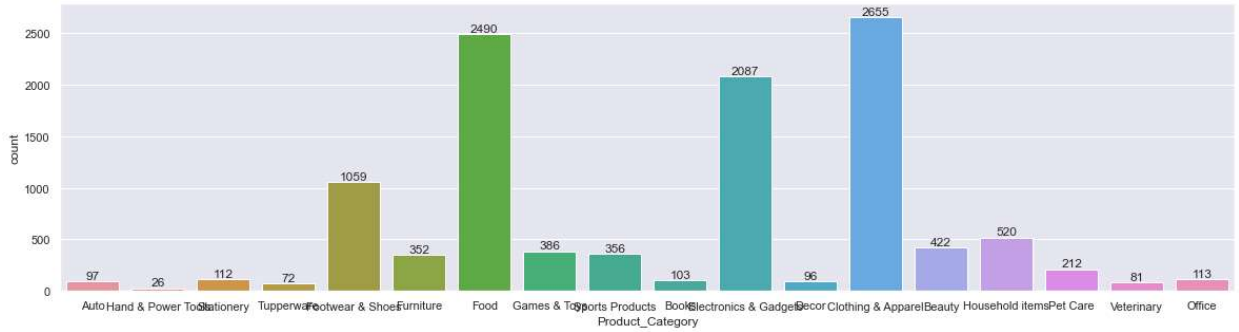
In [27]:
```
sales_state = df.groupby(['Occupation'], as_index=False)['Amount'].sum().sort_values(b

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

Out[27]:
```
<AxesSubplot:xlabel='Occupation', ylabel='Amount'>
```



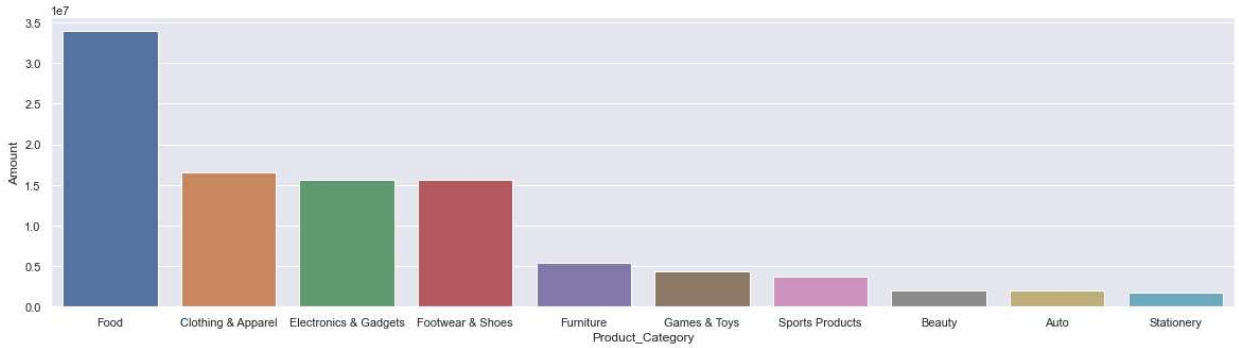
From above graphs we can see that most of the buyers are working in IT, Healthcare and Aviation sector

In [28]:
```
#PRODUCT CATEGORY
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 [29]:
```
sales_state = df.groupby(['Product_Category'], as_index=False)['Amount'].sum().sort_va

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

Out[29]:
```
<AxesSubplot:xlabel='Product_Category', ylabel='Amount'>
```



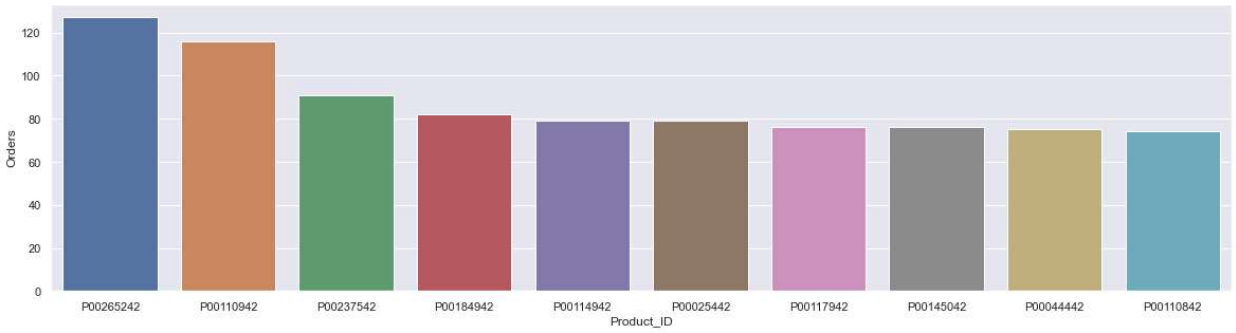
From above graphs we can see that most of the sold products are from Food, Clothing and Electronics category

In [30]:
```
sales_state = df.groupby(['Product_ID'], as_index=False)['Orders'].sum().sort_values(b
```

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

Out[30]:

```
<AxesSubplot:xlabel='Product_ID', ylabel='Orders'>
```

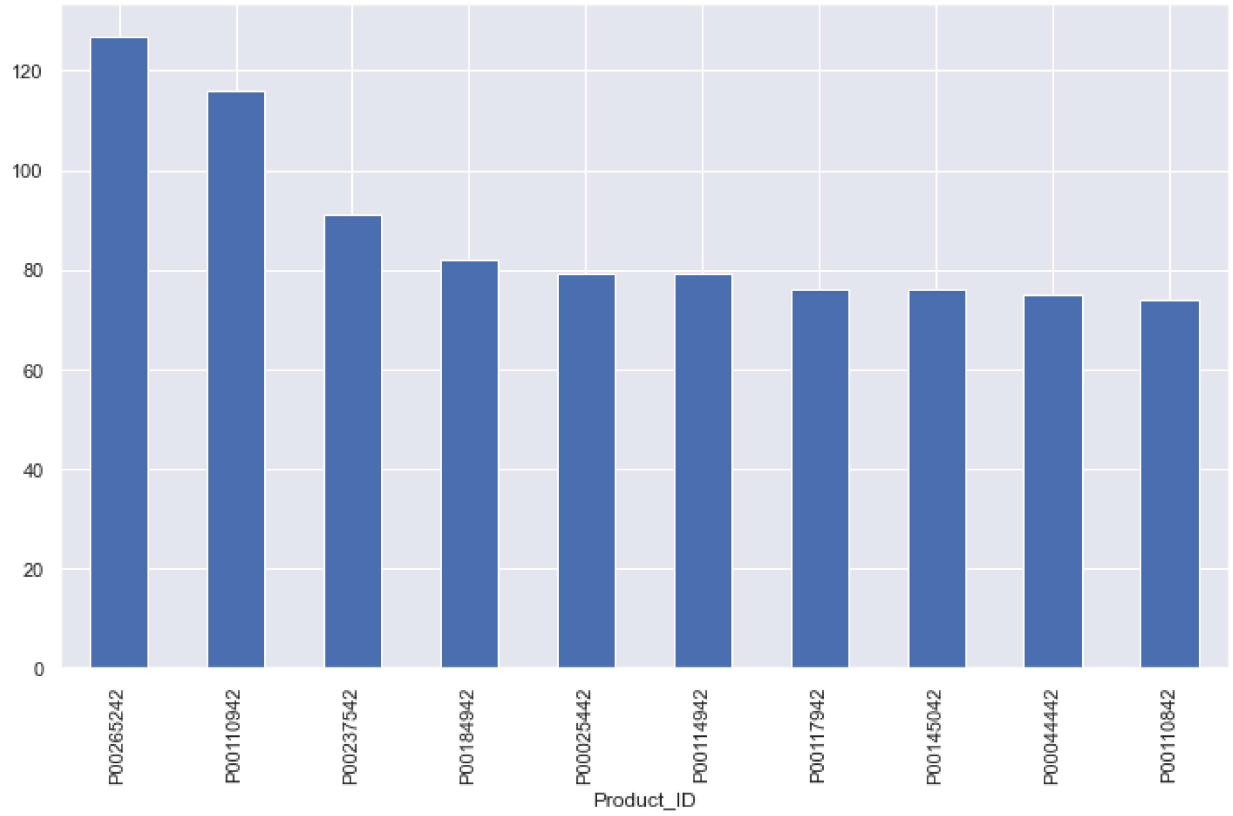


In [33]:

```
# top 10 most sold products (same thing as above)

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

Out[33]:

```
<AxesSubplot:xlabel='Product_ID'>
```



Conclusion:

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