Diwaly_Sales_Analysis

April 1, 2024

1 Diwaly_Sales_Analysis project

[3]:
```
# library's for data visualizations
import numpy as pd
import pandas as pd
import matplotlib.pyplot as plt # visualizing Data
%matplotlib inline
import seaborn as sns
```

[4]:
```
df = pd.read_csv('Diwali Sales Data.csv',encoding = 'unicode_escape')
# we used encoding to avoide error.
```

[5]:
```
df.shape # provide the numbers of rows and colums in the table
```

[5]:
```
(11251, 15)
```

[6]:
```
df.head() # it use to give the top values of csv file to see weather file is␣
↪uploded correct or not
```

[6]:
```
   User_ID  Cust_name Product_ID Gender Age Group  Age  Marital_Status  \
0  1002903  Sanskriti  P00125942      F     26-35   28               0
1  1000732     Kartik  P00110942      F     26-35   35               1
2  1001990      Bindu  P00118542      F     26-35   35               1
3  1001425     Sudevi  P00237842      M      0-17   16               0
4  1000588       Joni  P00057942      M     26-35   28               1

            State      Zone       Occupation Product_Category  Orders  \
0     Maharashtra   Western       Healthcare             Auto       1
1  Andhra Pradesh  Southern             Govt             Auto       3
2   Uttar Pradesh   Central       Automobile             Auto       3
3       Karnataka  Southern     Construction             Auto       2
4         Gujarat   Western  Food Processing             Auto       2

    Amount  Status  unnamed1
0  23952.0     NaN       NaN
1  23934.0     NaN       NaN
2  23924.0     NaN       NaN
3  23912.0     NaN       NaN
```

```
 4   23877.0      NaN        NaN
```

[7]:
```
df.info() # provide the information about the csv file
```

```
<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
```

[8]:
```
#to drop the unrelated / Blank coloum
df.drop(["Status","unnamed1"], axis=1 ,inplace = True)
# axis means hole row where we are going to perform this
# inplace means whatever we did in above line it will save that
```

[9]:
```
df.info() # updated information
```

```
<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
```

[10]:
```
pd.isnull(df)# will tell that whereever ther is false that is not null
```

[10]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	\
0	False	False	False	False	False	False	
1	False	False	False	False	False	False	
2	False	False	False	False	False	False	
3	False	False	False	False	False	False	
4	False	False	False	False	False	False	
…	…	…	…	…	…	…	
11246	False	False	False	False	False	False	
11247	False	False	False	False	False	False	
11248	False	False	False	False	False	False	
11249	False	False	False	False	False	False	
11250	False	False	False	False	False	False	

	Marital_Status	State	Zone	Occupation	Product_Category	Orders	\
0	False	False	False	False	False	False	
1	False	False	False	False	False	False	
2	False	False	False	False	False	False	
3	False	False	False	False	False	False	
4	False	False	False	False	False	False	
…	…	…	…	…	…	…	
11246	False	False	False	False	False	False	
11247	False	False	False	False	False	False	
11248	False	False	False	False	False	False	
11249	False	False	False	False	False	False	
11250	False	False	False	False	False	False	

	Amount
0	False
1	False
2	False
3	False
4	False
…	…
11246	False
11247	False
11248	False
11249	False
11250	False

```
[11251 rows x 13 columns]
```

[11]:
```
pd.isnull(df).sum() # here it will tell u that sum of null values is that␣
↪perticular column
```

[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
```

[12]:
```
df.dropna(inplace=True) # to remove the null values from the table we us dropna
```

[13]:
```
df.shape # updated values
```

[13]:
```
(11239, 13)
```

1.1 Both are same

df_test.dropna(inplace=True)

df_test=df_test.dropna()

[14]:
```
# initialize list of lists
data_set =[['Sai',11],['Gopal',29],['yash',],['Harsh',19]] # here we just␣
↪assign the values

# creat the pandas DataFrame using list
df_test = pd.DataFrame(data_set,columns=['Name','Age']) # this how we use to␣
↪create the dataframe

df_test
```

[14]:

	Name	Age
0	Sai	11.0
1	Gopal	29.0
2	yash	NaN
3	Harsh	19.0

[16]:
```
df_test.dropna() # here we got the output which we want that delet null
```

[16]:

	Name	Age
0	Sai	11.0
1	Gopal	29.0
3	Harsh	19.0

[17]:
```
df_test # but aganin here we got that null again because we did use inplace ,␣
↪it worke as save the thing which we did
```

[17]:

	Name	Age
0	Sai	11.0
1	Gopal	29.0
2	yash	NaN
3	Harsh	19.0

[18]:
```
# To change data type
df['Amount'] = df['Amount'].astype('int')
```

[19]:
```
# to cheak the datatype
df['Amount'].dtypes
```

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

[20]:
```
df.info() # to cheak the updated values
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 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
```

[21]:
```
# To cheak the colums we can use
df.columns
```

[21]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

[22]:
```
# to rename  the column
df.rename(columns ={'Marital_Status':"Single/Engaged"})
```

[22]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Single/Engaged	\
0	1002903	Sanskriti	P00125942	F	26-35	28	0	
1	1000732	Kartik	P00110942	F	26-35	35	1	
2	1001990	Bindu	P00118542	F	26-35	35	1	
3	1001425	Sudevi	P00237842	M	0-17	16	0	
4	1000588	Joni	P00057942	M	26-35	28	1	
…	…	…	…	…	…	…	…	
11246	1000695	Manning	P00296942	M	18-25	19	1	
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	
11250	1002744	Brumley	P00281742	F	18-25	19	0	

	State	Zone	Occupation	Product_Category	Orders	\
0	Maharashtra	Western	Healthcare	Auto	1	
1	Andhra Pradesh	Southern	Govt	Auto	3	
2	Uttar Pradesh	Central	Automobile	Auto	3	
3	Karnataka	Southern	Construction	Auto	2	
4	Gujarat	Western	Food Processing	Auto	2	
…	…	…	…	…	…	
11246	Maharashtra	Western	Chemical	Office	4	
11247	Haryana	Northern	Healthcare	Veterinary	3	
11248	Madhya Pradesh	Central	Textile	Office	4	
11249	Karnataka	Southern	Agriculture	Office	3	
11250	Maharashtra	Western	Healthcare	Office	3	

	Amount
0	23952
1	23934
2	23924
3	23912
4	23877
…	…
11246	370
11247	367
11248	213

```
11249     206
11250     188

[11239 rows x 13 columns]
```

[23]:
```
# describe() method will provide us the description of data in the DataFrame (i.
 ↪e. Count,Mean,Std,Min,Max,etc
df.describe() # it will give descripthon for numeric values
```

[23]:

	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

[24]:
```
# If we want for perticular column we us
df[['Age','Orders','Amount']].describe()
```

[24]:

	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

2 Exploratory Data Analysis

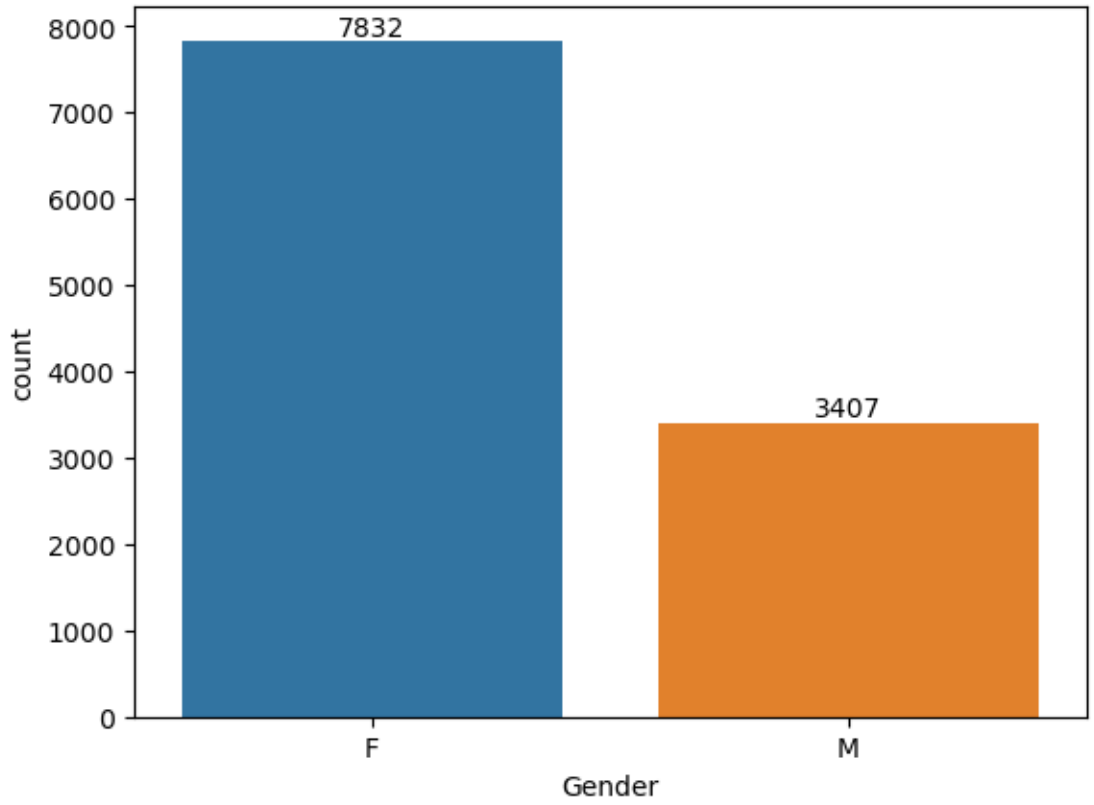
2.0.1 Gender

[25]:
```
df.columns
```

[25]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

[26]:
```
A = sns.countplot( x = 'Gender', data = df)
# here  dota is taken from csv we assign it as df (we can us only this to get␣
 ↪the data )
for bars in A.containers: # this is used to give the lable for the data
```

```
    A.bar_label(bars)
```



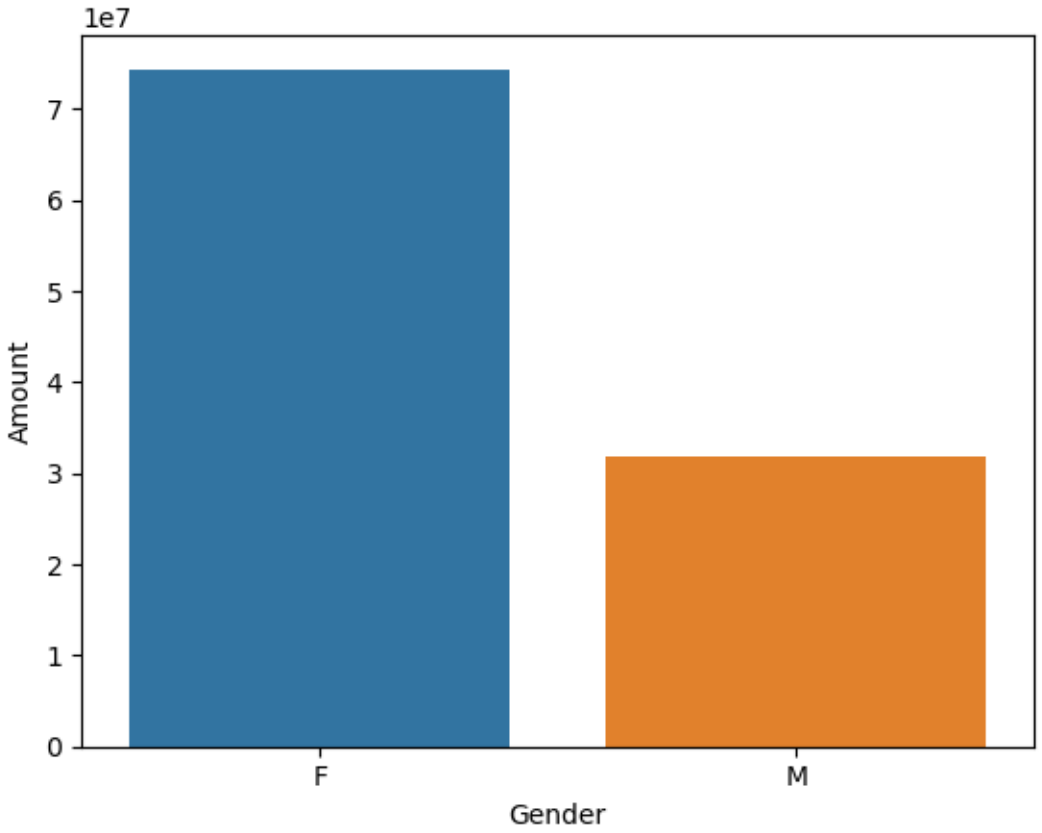
[27]:
```
#Total Amount Vs Gender in table
df.groupby(['Gender'], as_index = False) ['Amount'].sum().sort_values(by =␣
 ↪'Amount',ascending =False)
```

[27]:

	Gender	Amount
0	F	74335853
1	M	31913276

[31]:
```
# Total Amount Vs Gender in bar chart
sales_gen = df.groupby(['Gender'], as_index=False)['Amount'].sum().
 ↪sort_values(by='Amount', ascending=False)
sns.barplot(x='Gender', y='Amount', data=sales_gen)
```

[31]:
```
<Axes: xlabel='Gender', ylabel='Amount'>
```



- From above graphs we can see that most of the buyers are **Females** and the purchasing power of **Females are Greater then Man**

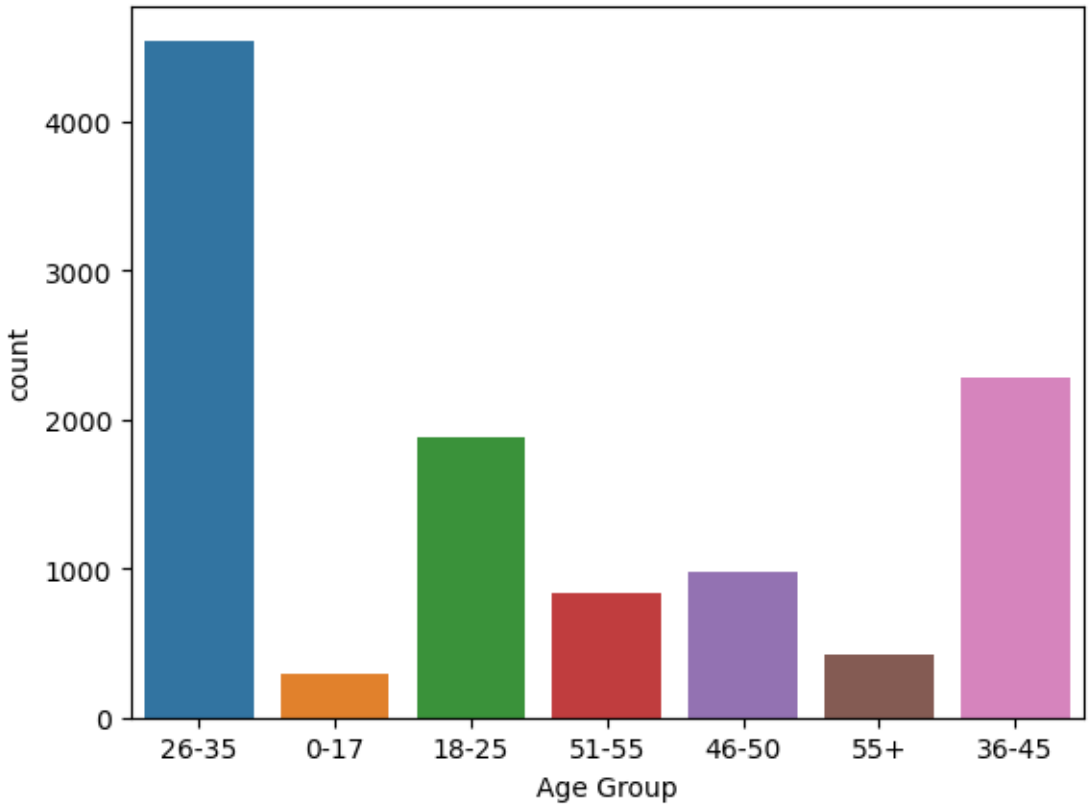
2.0.2 Age

[34]:
```
df.columns
```

[34]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

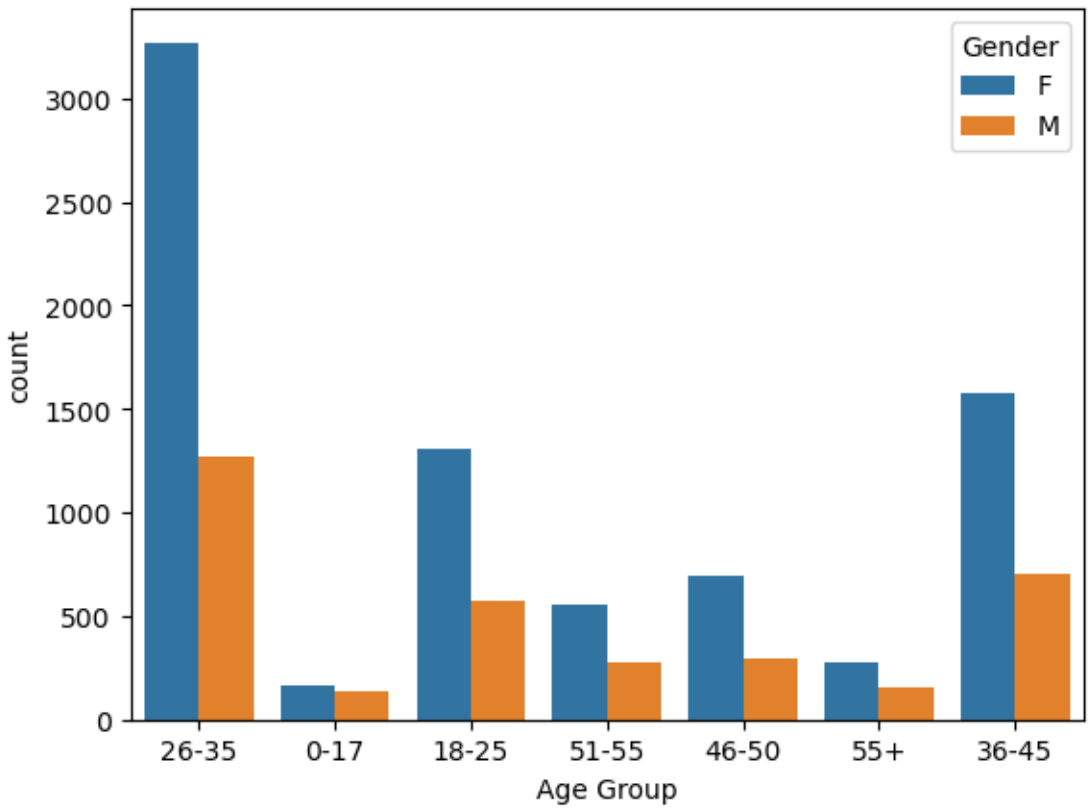
[35]:
```
sns.countplot(data = df, x = 'Age Group')
# this will give us the rough idea that which age group  had done more shoping
```

[35]:
```
<Axes: xlabel='Age Group', ylabel='count'>
```

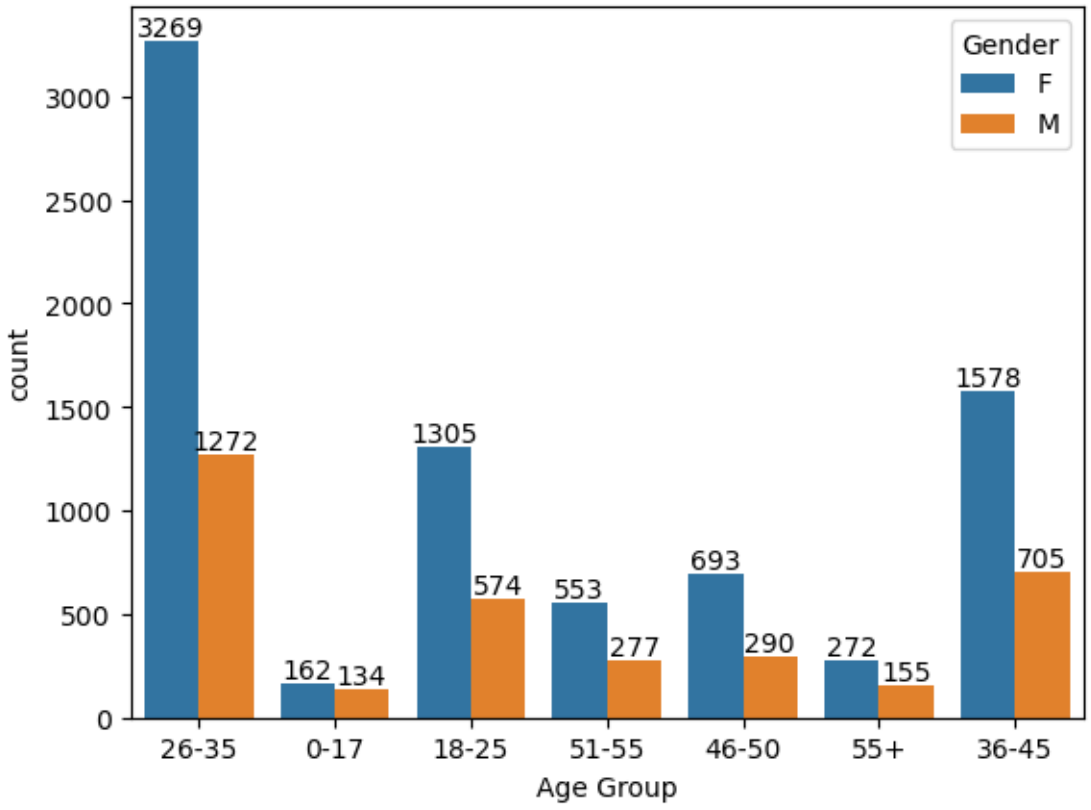


[36]:
```
sns.countplot(data = df, x = 'Age Group', hue = 'Gender')
# here we can see it all age group are split in gender for this we us hue
```

[36]:
```
<Axes: xlabel='Age Group', ylabel='count'>
```

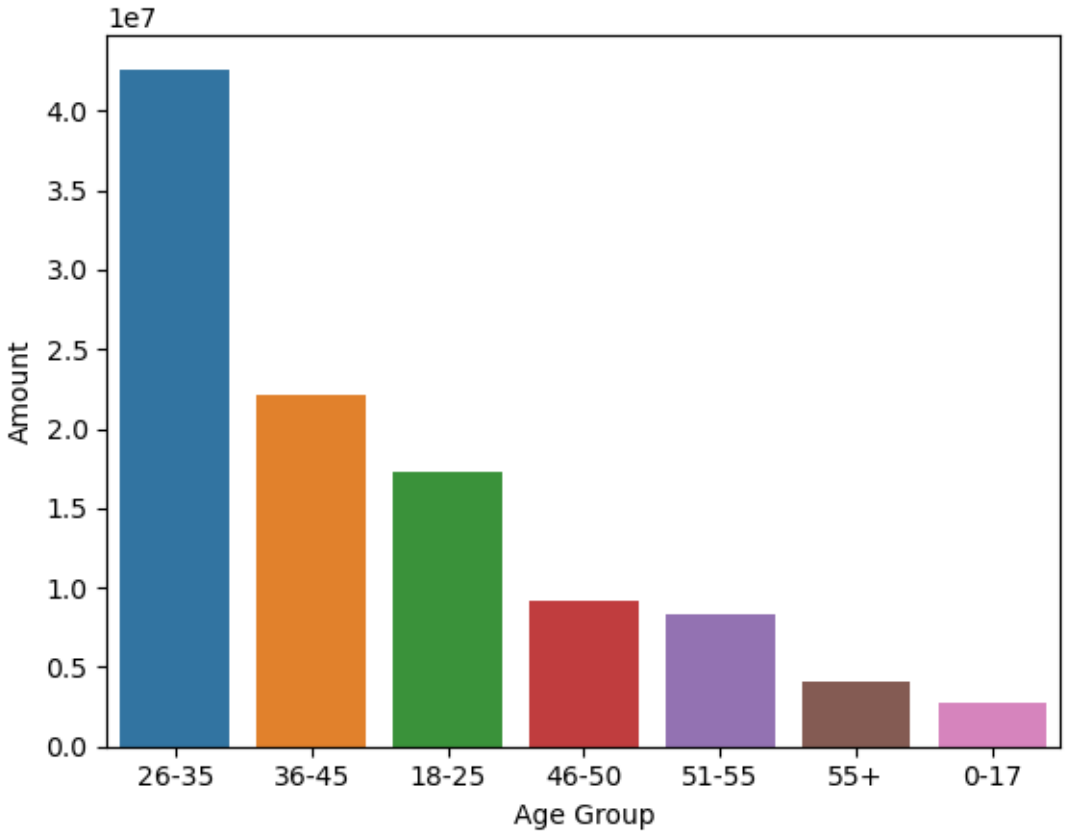


[46]:
```
# Create the count plot
A = sns.countplot(data=df, x='Age Group', hue='Gender')

# Add labels to the bars
for bars in A.containers:
    A.bar_label(bars)
```



[48]:
```
# Total Amount Vs Age Group
sales_age = df.groupby(['Age Group'], as_index=False)['Amount'].sum().
 ↪sort_values(by='Amount', ascending=False)

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

[48]: <Axes: xlabel='Age Group', ylabel='Amount'>



- From above graphs we can see that most of the buyers are of **Age Group Between 26-35 yrs Females**

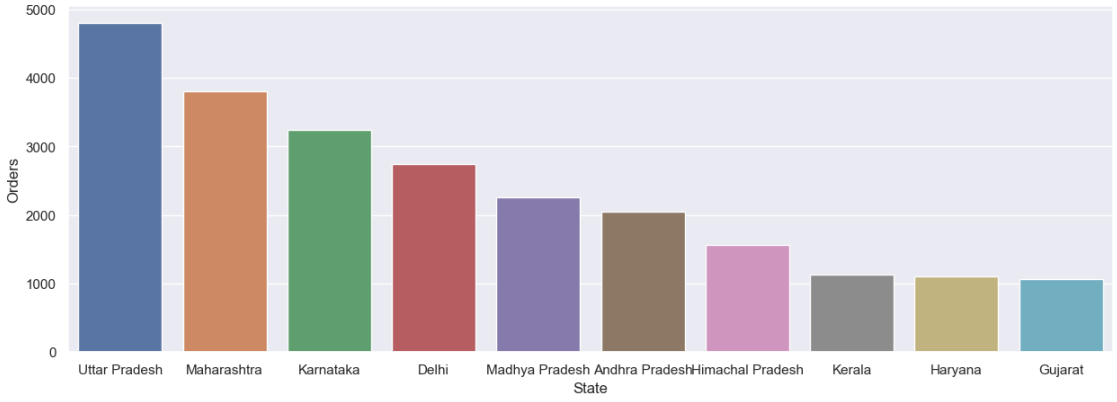
2.0.3 State

[49]:
```
df.columns
```

[49]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

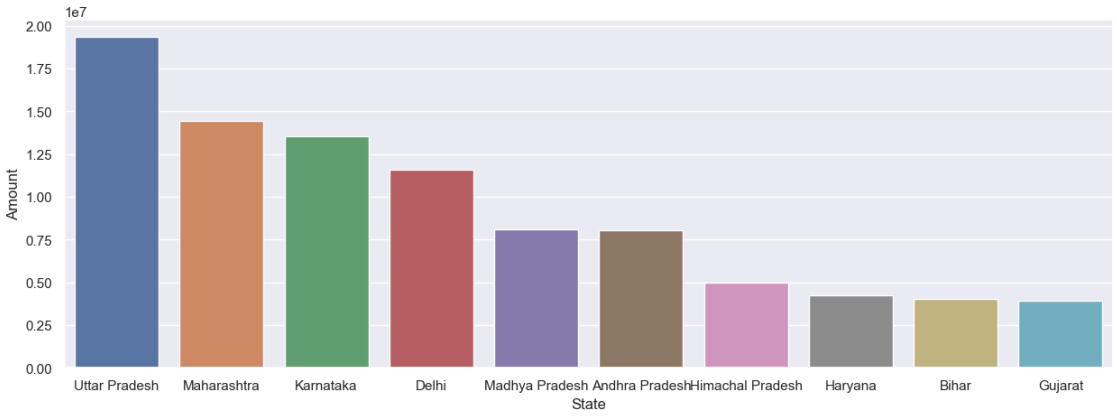
[51]:
```
# Total numbers of orders from top 10 states
sales_state = df.groupby(['State'], as_index=False)['Orders'].sum().
 ↪sort_values(by='Orders', ascending=False).head(10)
sns.set(rc={'figure.figsize': (15, 5)})  # here we gave the chart size we can␣
 ↪change according to our use
sns.barplot(x='State', y='Orders', data=sales_state)
```

[51]:
```
<Axes: xlabel='State', ylabel='Orders'>
```



[53]:
```
# Total Amount/Sales from top 10 states
sales_state = df.groupby(['State'], as_index=False)['Amount'].sum().
 ↪sort_values(by='Amount', ascending=False).head(10)
sns.set(rc={'figure.figsize': (15, 5)}) # here we gave the chart size we can␣
 ↪change according to our use
sns.barplot(x='State', y='Amount', data=sales_state)
```

[53]:
```
<Axes: xlabel='State', ylabel='Amount'>
```



- From above graph we can see that Most of the **orders & total sale/amount** are from **Uttar pradesh**, **Maharashtra**, **Karnatak** respectively*

2.0.4 Marital Status

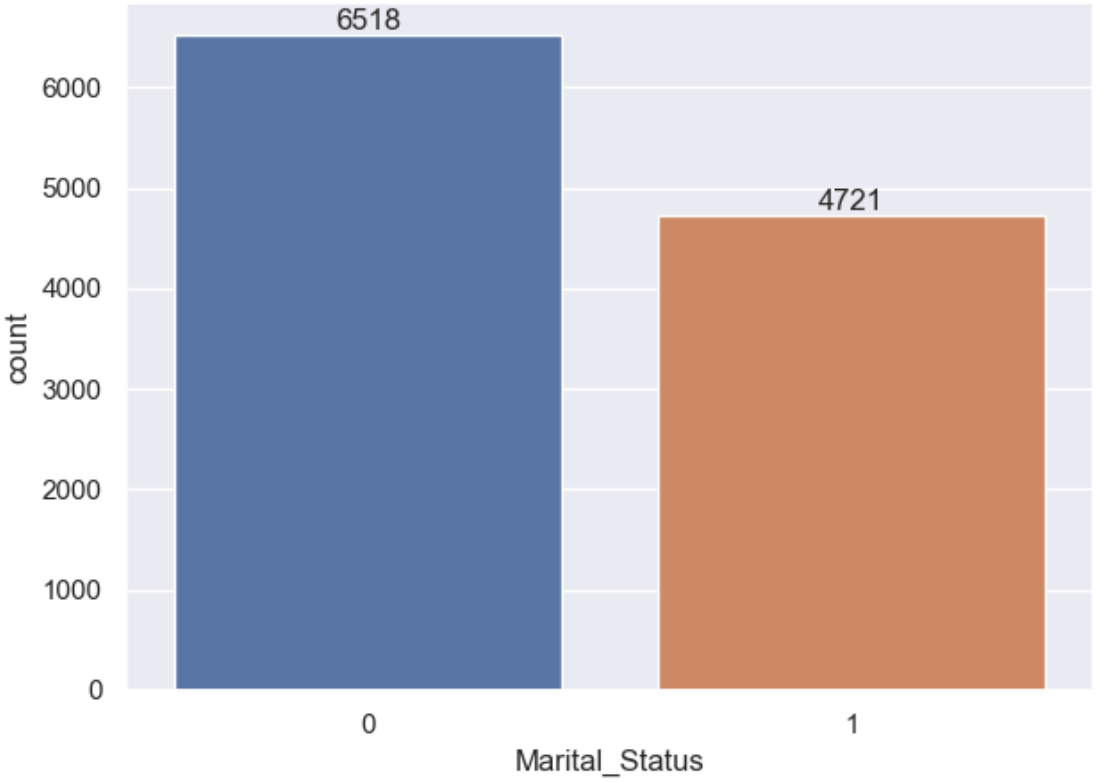
[54]:
```
# to rename  the column
df.rename(columns ={'Single/Engaged':'Marital_Status'})
```

[54]:
```
        User_ID    Cust_name Product_ID Gender Age Group  Age  Marital_Status  \
0       1002903    Sanskriti  P00125942      F     26-35   28               0
1       1000732       Kartik  P00110942      F     26-35   35               1
2       1001990        Bindu  P00118542      F     26-35   35               1
3       1001425       Sudevi  P00237842      M      0-17   16               0
4       1000588         Joni  P00057942      M     26-35   28               1
…         …          …         …    …       …  …            …
11246   1000695      Manning  P00296942      M     18-25   19               1
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
11250   1002744      Brumley  P00281742      F     18-25   19               0

                State      Zone       Occupation Product_Category  Orders  \
0         Maharashtra   Western       Healthcare             Auto       1
1      Andhra Pradesh  Southern             Govt             Auto       3
2       Uttar Pradesh   Central       Automobile             Auto       3
3           Karnataka  Southern     Construction             Auto       2
4             Gujarat   Western  Food Processing             Auto       2
…                 …       …               …              …     …
11246     Maharashtra   Western         Chemical           Office       4
11247         Haryana  Northern       Healthcare       Veterinary       3
11248  Madhya Pradesh   Central          Textile           Office       4
11249       Karnataka  Southern      Agriculture           Office       3
11250     Maharashtra   Western       Healthcare           Office       3

       Amount
0       23952
1       23934
2       23924
3       23912
4       23877
…       …
11246     370
11247     367
11248     213
11249     206
11250     188

[11239 rows x 13 columns]
```

[55]:
```
df.columns
```

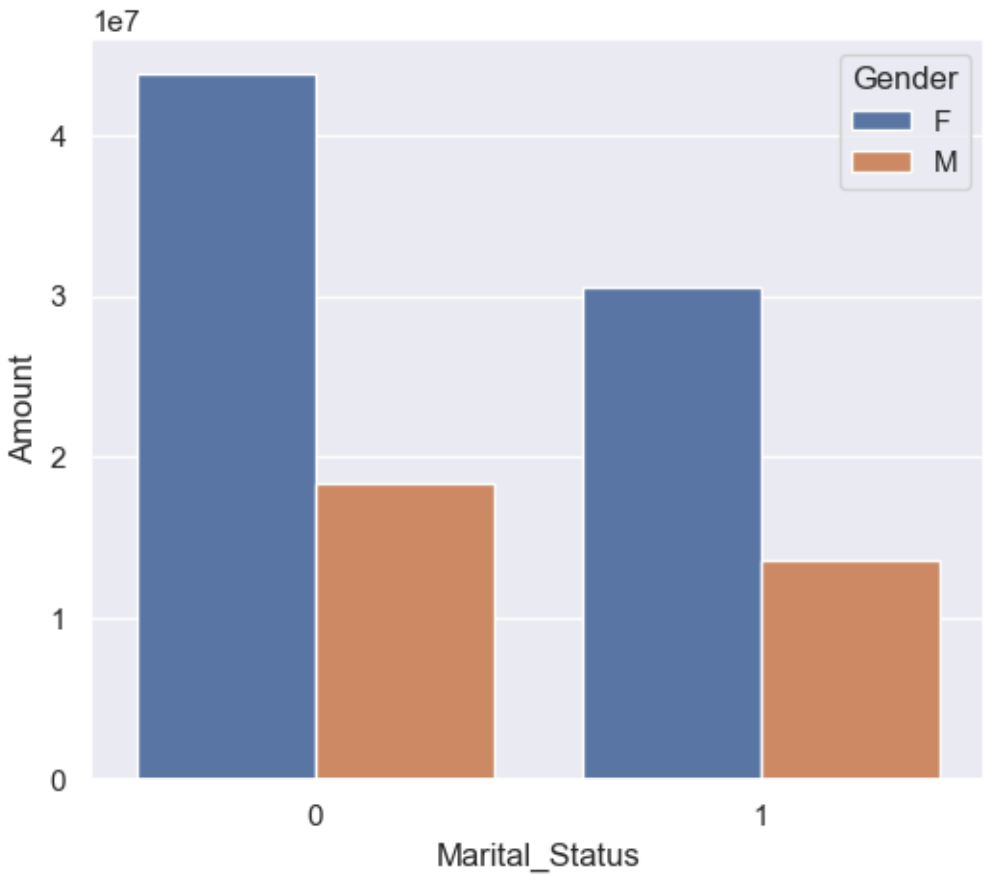
[55]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

[62]:
```
# Create the count plot
A = sns.countplot(data=df, x='Marital_Status')
sns.set(rc={'figure.figsize': (7,4)})
# here we gave the chart size we can change according to our use
# Add labels to the bars
for bars in A.containers:
    A.bar_label(bars)
```



[65]:
```
# Marital_status Vs Amount/Gender
sales_status = df.groupby(['Marital_Status','Gender'],␣
 ↪as_index=False)['Amount'].sum().sort_values(by='Amount', ascending=False)
sns.set(rc={'figure.figsize': (6, 5)})  # here we gave the chart size we can␣
 ↪change according to our use
sns.barplot(x='Marital_Status', y='Amount', hue ='Gender', data=sales_status)
```

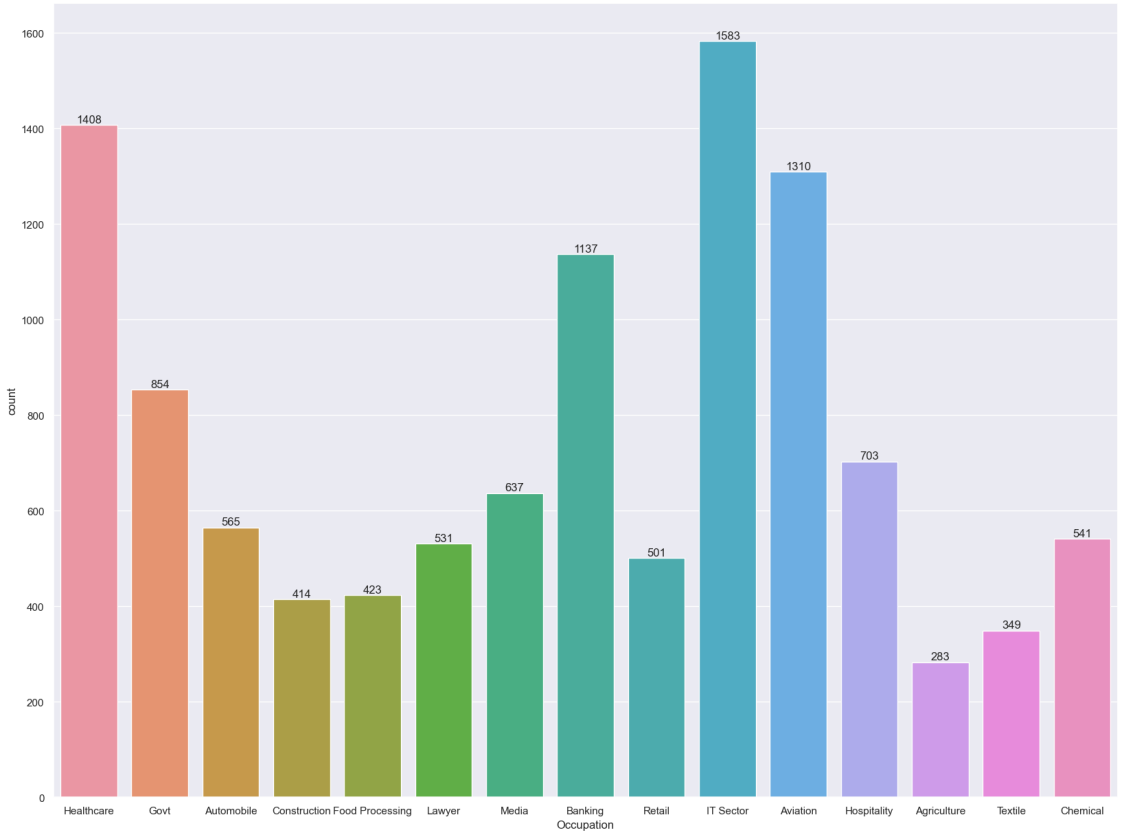
[65]: <Axes: xlabel='Marital_Status', ylabel='Amount'>



- From the above graph we can see that most of the buyers are **Married(Women)** and they have high Purchasing power

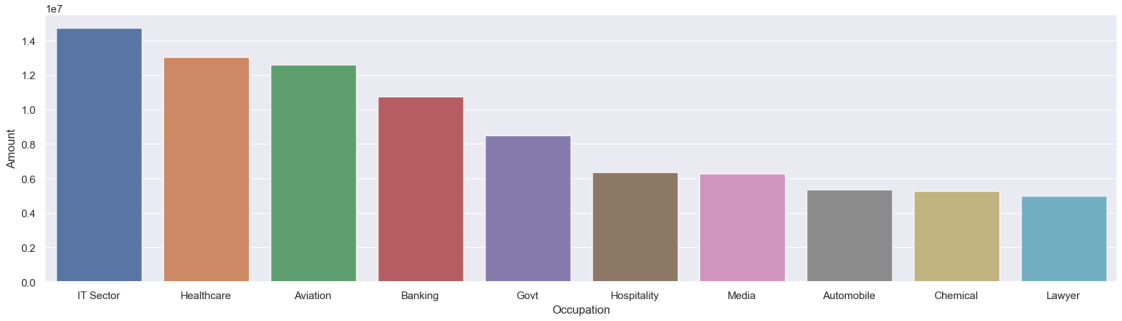
2.0.5 Occupation

[77]:
```
# Create the count plot
A = sns.countplot(data=df, x='Occupation')
sns.set(rc={'figure.figsize': (20,15)})
# here we gave the chart size we can change according to our use
# Add labels to the bars
for bars in A.containers:
    A.bar_label(bars)
```



[90]:
```
# Occupation/ amount spent
Occupation = df.groupby(['Occupation'], as_index=False)['Amount'].sum().
 ↪sort_values(by='Amount', ascending=False).head(10)
sns.set(rc={'figure.figsize': (20, 5)}) # here we gave the chart size we can␣
 ↪change according to our use
sns.barplot(x='Occupation', y='Amount', data=Occupation)
```

[90]: <Axes: xlabel='Occupation', ylabel='Amount'>



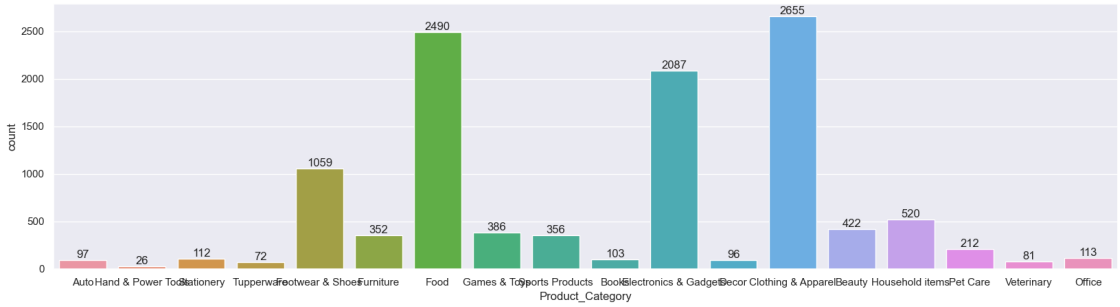
- From above graphs we can see that most of the buyers are working in **IT**, **Healthcare Sector**, and **Aviation Sector**.

[71]:
```
df.columns
```

[71]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

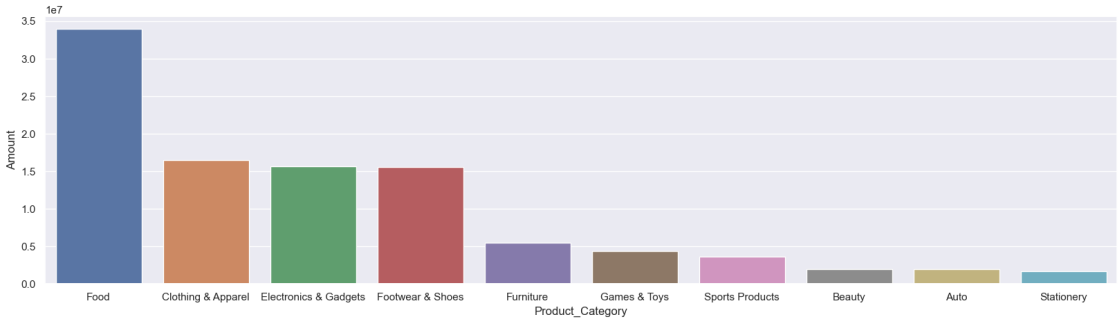
2.0.6 Product_Category

[86]:
```
# Create the count plot
A = sns.countplot(data=df, x='Product_Category')
sns.set(rc={'figure.figsize': (20,15)})
# here we gave the chart size we can change according to our use
# Add labels to the bars
for bars in A.containers:
    A.bar_label(bars)
```



[91]:
```
# Product_Category/ amount spent
Product_Category = df.groupby(['Product_Category'], as_index=False)['Amount'].
 ↪sum().sort_values(by='Amount', ascending=False).head(10)
sns.set(rc={'figure.figsize': (20, 5)}) # here we gave the chart size we can␣
 ↪change according to our use
sns.barplot(x='Product_Category', y='Amount', data=Product_Category)
```

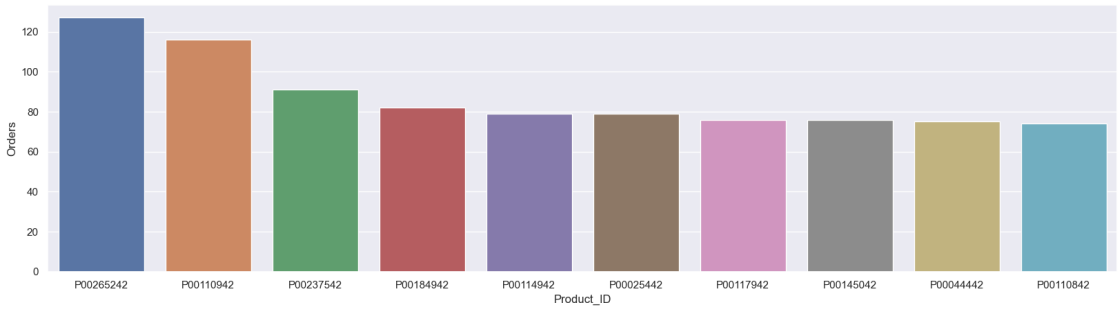
[91]:
```
<Axes: xlabel='Product_Category', ylabel='Amount'>
```



- From above graphs we can see that most of the Sold products are **Food**, **Clothing**, and **Electronic Products**.

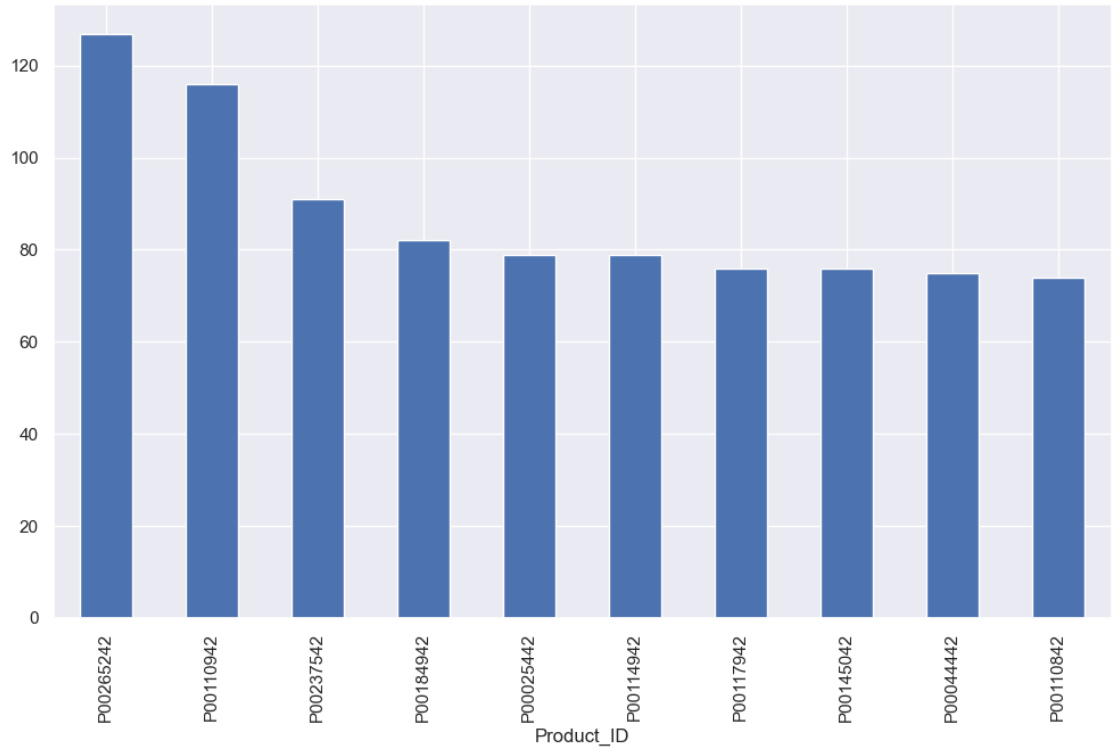
[82]:
```
# Product_ID/Orders
Product = df.groupby(['Product_ID'], as_index=False)['Orders'].sum().
 ↪sort_values(by='Orders', ascending=False).head(10)
sns.set(rc={'figure.figsize': (20, 5)}) # here we gave the chart size we can␣
 ↪change according to our use
sns.barplot(x='Product_ID', y='Orders', data=Product)
```

[82]: <Axes: xlabel='Product_ID', ylabel='Orders'>



[95]:
```
# top 10 most Sold Products (Same things as above)

fig1, A = plt.subplots(figsize = (12,7))
df.groupby('Product_ID')['Orders'].sum().nlargest(10).
 ↪sort_values(ascending=False).plot(kind = 'bar')
```

[95]: <Axes: xlabel='Product_ID'>



3 CONCLUSION

3.0.1 *Marrid Women Age group 26-35 from UP, Maharashtra, Karnatak working in IT, Healthcare and Aviation sector are more likely to buy product from Food, Clothing and Electronics Category*

[ ]: