first-file

November 5, 2024

[1]:
```
pip install numpy
```

```
Requirement already satisfied: numpy in c:\users\acer\anaconda3\lib\site-
packages (1.26.4)
Note: you may need to restart the kernel to use updated packages.
```

[3]:
```
!pip install numpy
```

```
Requirement already satisfied: numpy in c:\users\acer\anaconda3\lib\site-
packages (1.26.4)
```

[5]:
```
!pip install pandas
```

```
Requirement already satisfied: pandas in c:\users\acer\anaconda3\lib\site-
packages (2.2.2)
Requirement already satisfied: numpy>=1.26.0 in
c:\users\acer\anaconda3\lib\site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in
c:\users\acer\anaconda3\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\acer\anaconda3\lib\site-
packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in
c:\users\acer\anaconda3\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\acer\anaconda3\lib\site-
packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
```

[7]:
```
!pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\acer\anaconda3\lib\site-
packages (3.8.4)
Requirement already satisfied: contourpy>=1.0.1 in
c:\users\acer\anaconda3\lib\site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\acer\anaconda3\lib\site-
packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
c:\users\acer\anaconda3\lib\site-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
c:\users\acer\anaconda3\lib\site-packages (from matplotlib) (1.4.4)
```

```
Requirement already satisfied: numpy>=1.21 in c:\users\acer\anaconda3\lib\site-
packages (from matplotlib) (1.26.4)
Requirement already satisfied: packaging>=20.0 in
c:\users\acer\anaconda3\lib\site-packages (from matplotlib) (23.2)
Requirement already satisfied: pillow>=8 in c:\users\acer\anaconda3\lib\site-
packages (from matplotlib) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in
c:\users\acer\anaconda3\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in
c:\users\acer\anaconda3\lib\site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:\users\acer\anaconda3\lib\site-
packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

[9]:
```
!pip install seaborn
```

```
Requirement already satisfied: seaborn in c:\users\acer\anaconda3\lib\site-
packages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in
c:\users\acer\anaconda3\lib\site-packages (from seaborn) (1.26.4)
Requirement already satisfied: pandas>=1.2 in c:\users\acer\anaconda3\lib\site-
packages (from seaborn) (2.2.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in
c:\users\acer\anaconda3\lib\site-packages (from seaborn) (3.8.4)
Requirement already satisfied: contourpy>=1.0.1 in
c:\users\acer\anaconda3\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\acer\anaconda3\lib\site-
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
c:\users\acer\anaconda3\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
c:\users\acer\anaconda3\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in
c:\users\acer\anaconda3\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (23.2)
Requirement already satisfied: pillow>=8 in c:\users\acer\anaconda3\lib\site-
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in
c:\users\acer\anaconda3\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in
c:\users\acer\anaconda3\lib\site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\acer\anaconda3\lib\site-
packages (from pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in
```

```
c:\users\acer\anaconda3\lib\site-packages (from pandas>=1.2->seaborn) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\acer\anaconda3\lib\site-
packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
```

[21]:
```
# import python libraries

import numpy as np #helps to work on arrays
import pandas as pd #helps to work on data frame (data tables)
import matplotlib.pyplot as plt # visualizing data
%matplotlib inline #configures the Matplotlib backend to render plots directly␣
 ↪within the notebook output cells.
import seaborn as sns #provides a high-level interface for creating attractive␣
 ↪and informative statistical graphics.
```

```
UsageError: unrecognized arguments: #configures the Matplotlib backend to render
plots directly within the notebook output cells.
```

[19]:
```
# import csv file using panda
# unicode_escape is to avoid encoding errors
df = pd.read_csv('Diwali Sales Data.csv', encoding= 'unicode_escape')
```

[23]:
```
df.shape #will tell no. of rows and columns
```

[23]:
```
(11251, 15)
```

[25]:
```
df.head() #will show you top 5 rows of the csv file
```

[25]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	State	Zone	Occupation	Product_Category	Orders	Amount	Status	unnamed1
0	1002903	Sanskriti	P00125942	F	26-35	28	0	Maharashtra	Western	Healthcare	Auto	1	23952.0	NaN	NaN
1	1000732	Kartik	P00110942	F	26-35	35	1	Andhra Pradesh	Southern	Govt	Auto	3	23934.0	NaN	NaN
2	1001990	Bindu	P00118542	F	26-35	35	1	Uttar Pradesh	Central	Automobile	Auto	3	23924.0	NaN	NaN
3	1001425	Sudevi	P00237842	M	0-17	16	0	Karnataka	Southern	Construction	Auto	2	23912.0	NaN	NaN
4	1000588	Joni	P00057942	M	26-35	28	1	Gujarat	Western	Food Processing	Auto	2	23877.0	NaN	NaN

[ ]:
```
#DATA CLEANING.....................................
```

[27]:
```
df.info
```

[27]:
```
<bound method DataFrame.info of        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  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
…        …     …         …
11246    370.0     NaN       NaN
11247    367.0     NaN       NaN
11248    213.0     NaN       NaN
11249    206.0     NaN       NaN
11250    188.0     NaN       NaN

[11251 rows x 15 columns]>
```

[29]:
```
#drop unrelated/blank columns
#axis=1 means to select the whole column named
#inplace=True specifies that the modification should be made directly to the␣
 ↪DataFrame df itself, rather than returning a new DataFrame with the columns␣
 ↪dropped.
#Without inplace=True: If inplace is not specified or set to False (which is␣
 ↪the default), df.drop(['Status', 'unnamed1'], axis=1) would return a new␣
 ↪DataFrame with the specified columns dropped, but the original df would␣
 ↪remain unchanged.
df.drop(['Status', 'unnamed1'], axis=1 , inplace=True)
```

[31]:
```
df.info
```

[31]:
```
<bound method DataFrame.info of        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      23952.0
1      23934.0
2      23924.0
3      23912.0
4      23877.0
…        …
```

```
11246    370.0
11247    367.0
11248    213.0
11249    206.0
11250    188.0

[11251 rows x 13 columns]>
```

[33]:
```
pd.isnull(df)
#will give true(if null is there) or false value
```

[33]:
```
       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]
```

[35]:
```
#check no. nulls is every column

pd.isnull(df).sum()
```

[35]:
```
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
```

[37]:
```
# drop null values
df.dropna(inplace=True)
```

[39]:
```
df.shape
```

[39]:
```
(11239, 13)
```

[45]:
```
#example to create a dataframe from a list
# initialize List of Lists
data_test = [['madhav', 11], ['Gopi', 15], ['Keshav', ], ['Lalita', 16]]

# Create the pandas DataFrame using List
df_test = pd.DataFrame(data_test, columns=['Name', 'Age'])

df_test
```

[45]:
```
     Name   Age
0  madhav  11.0
1    Gopi  15.0
2  Keshav   NaN
3  Lalita  16.0
```

[51]:
```
df_test.dropna(inplace=True)
df_test
```

[51]:
```
     Name   Age
0  madhav  11.0
1    Gopi  15.0
3  Lalita  16.0
```

[53]:
```
print(df_test)
```

```
     Name   Age
0  madhav  11.0
1    Gopi  15.0
3  Lalita  16.0
```

[55]:
```
# change data type
df['Amount'] = df['Amount'].astype('int')
```

[57]:
```
df['Amount'].dtypes
```

[57]:
```
dtype('int32')
```

[59]:
```
df.columns
```

[59]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

[61]:
```
#rename column
df.rename(columns= {'Marital_Status':'Shaadi'})
```

[61]:
```
        User_ID    Cust_name Product_ID Gender Age Group  Age  Shaadi  \
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]
```

[63]:
```
# describe() method returns description of the data in the DataFrame (i.e.␣
 ↪count, mean, std, etc)
df.describe()
```

[63]:

	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

[65]:
```
# use describe() for specific columns
df[['Age', 'Orders', 'Amount']].describe()
```

[65]:

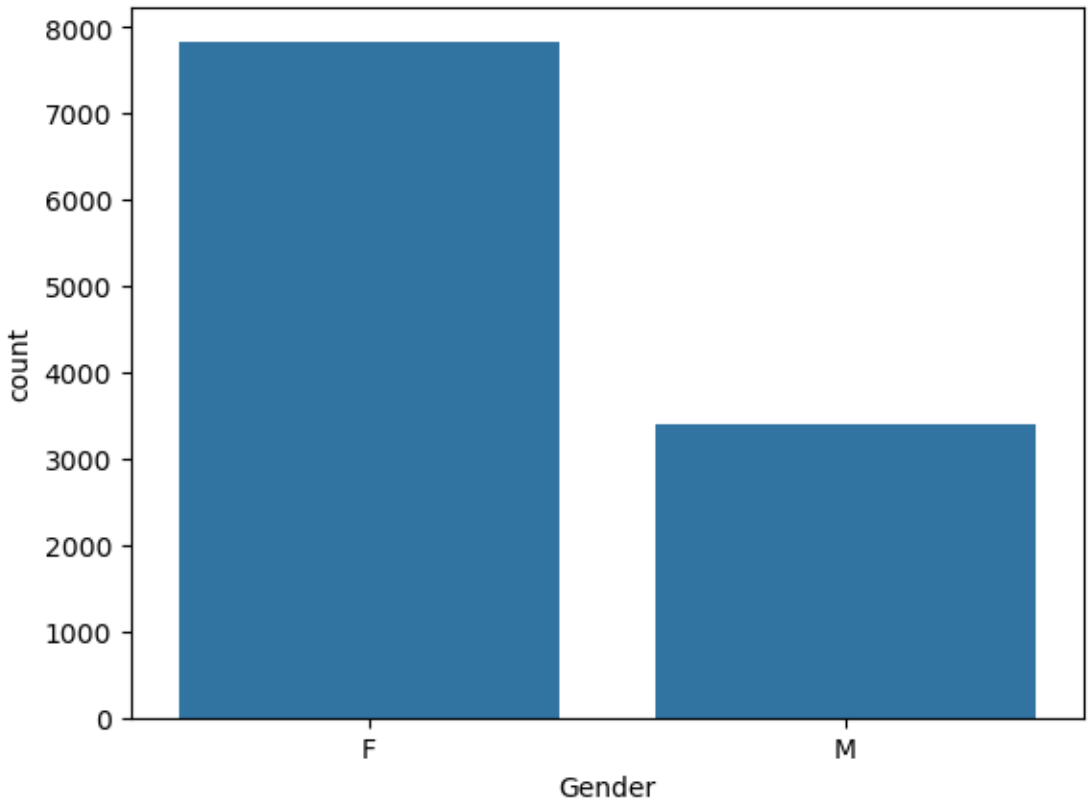
	Age	Orders	Amount
count	11239.000000	11239.000000	11239.000000
mean	35.410357	2.489634	9453.610553
std	12.753866	1.114967	5222.355168

```
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
```

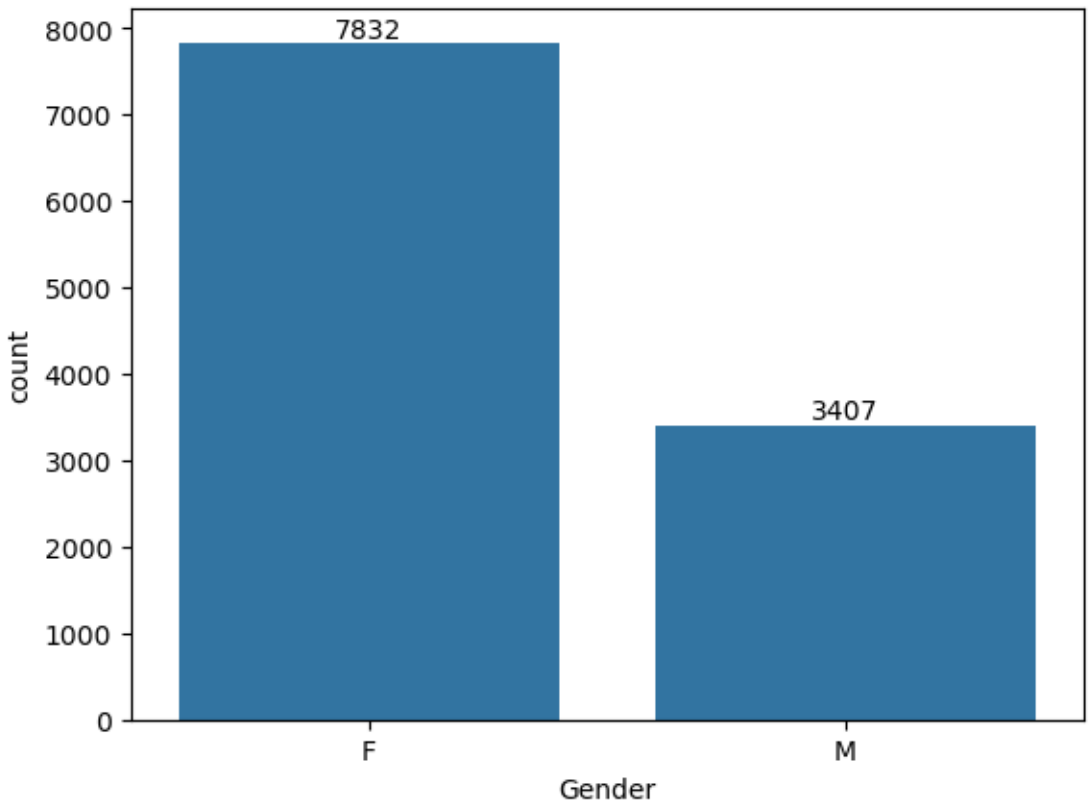
[ ]:
```
#Now we will perform Exploratory Data Analysis
```

[ ]:
```
#for GENDER
```

[67]:
```
# plotting a bar chart for Gender and it's count

ax = sns.countplot(x = 'Gender',data = df)
```



[69]:
```
ax = sns.countplot(x = 'Gender',data = df)
for bars in ax.containers:
    ax.bar_label(bars)
```



[127]:
```
sales_gen = df.groupby(['Gender'], as_index=False)['Amount'].sum().
↪sort_values(by='Amount', ascending=False)
```

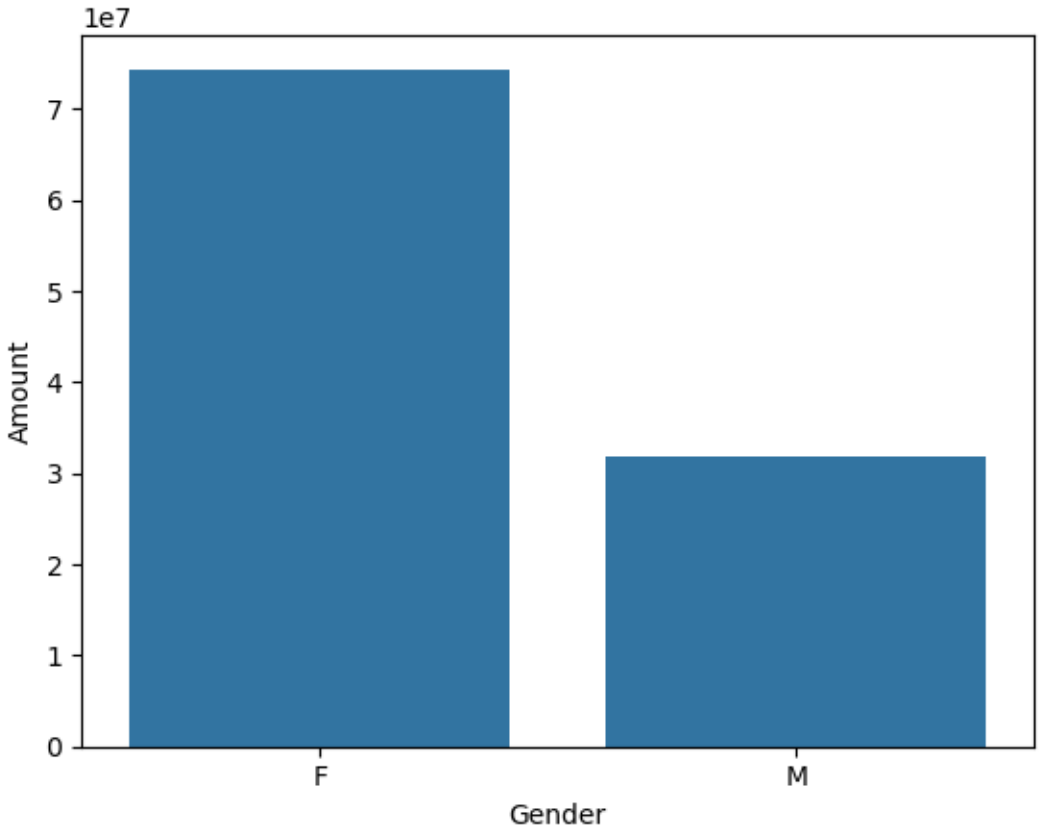
[73]:
```
sales_gen
```

[73]:

	Gender	Amount
0	F	74335853
1	M	31913276

[75]:
```
sns.barplot(x = 'Gender',y= 'Amount' ,data = sales_gen)
```

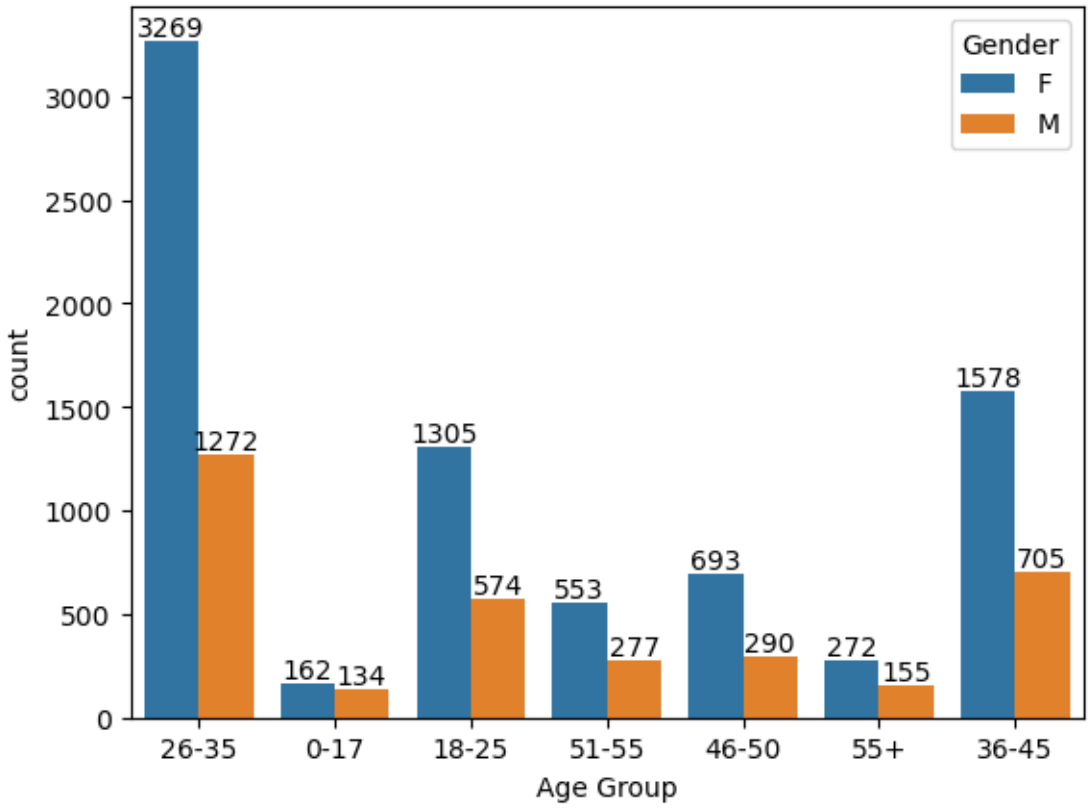
[75]:
```
<Axes: xlabel='Gender', ylabel='Amount'>
```



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

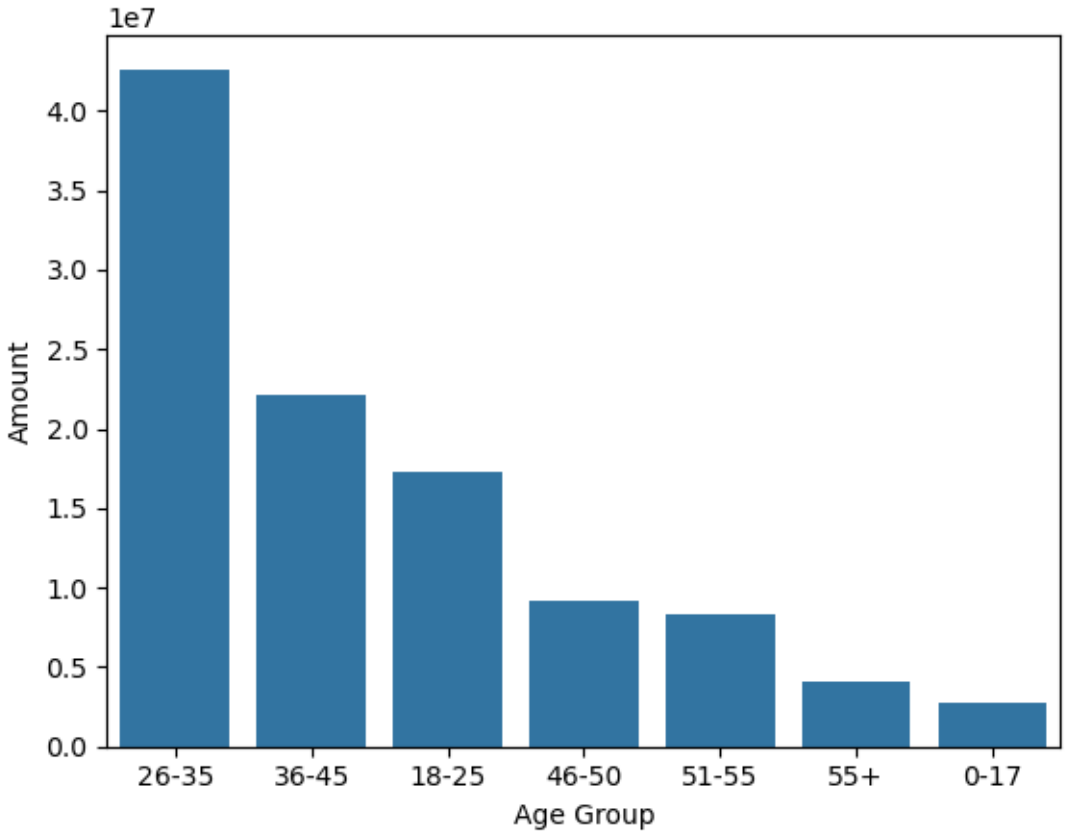
[77]:
```
#AGE
```

[79]:
```
ax = sns.countplot(data = df, x = 'Age Group', hue = 'Gender')
#hue will display diff graphs acc. to the gender
for bars in ax.containers:
    ax.bar_label(bars)
```



[81]:
```
# Total Amount vs Age Group
sales_age = df.groupby(['Age Group'], as_index=False)['Amount'].sum().
 ↪sort_values(by='Amount', ascending=False)
#as_index=False is used to control the structure of the resulting DataFrame␣
 ↪after certain operations to ensure that grouped columns or merged/joined␣
 ↪columns remain as regular columns and are not turned into indices. This␣
 ↪parameter provides flexibility in how you structure your data after␣
 ↪performing operations that may affect DataFrame indexing.



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

[81]:
```
<Axes: xlabel='Age Group', ylabel='Amount'>
```

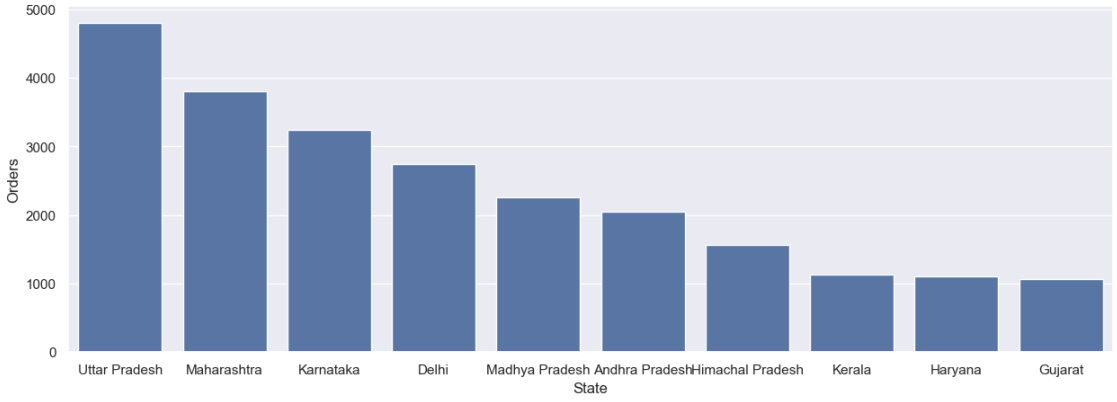


[ ]:
```
*From above graphs we can see that most of the buyers are
of age group between 26-35 yrs female*
```

[ ]:
```
#STATE
```

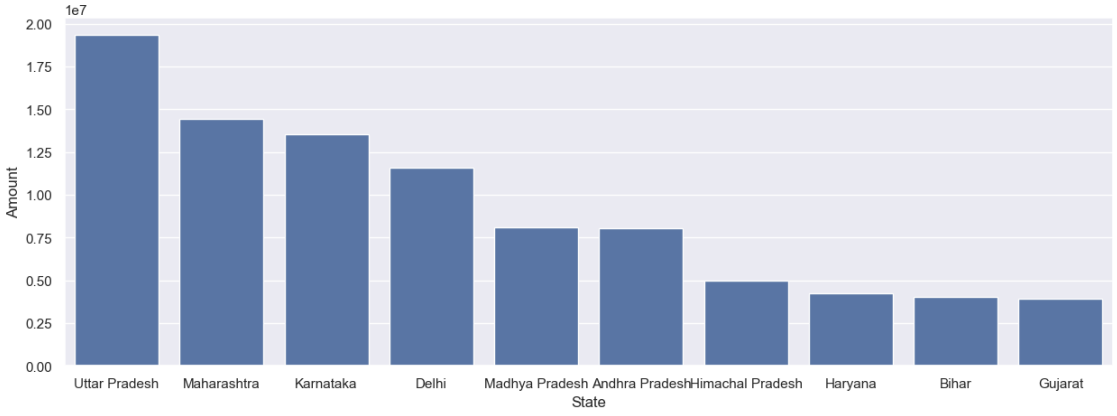
[85]:
```
# total number of orders from top 10 states

sales_state = df.groupby(['State'], as_index=False)['Orders'].sum().
 ↪sort_values(by='Orders', ascending=False).head(10)
#.head(?) mean no. of states to be mentioned in the graph
sns.set(rc={'figure.figsize':(15,5)})#to state the size of the graph
sns.barplot(data = sales_state, x = 'State',y= 'Orders')
```

[85]:
```
<Axes: xlabel='State', ylabel='Orders'>
```



[87]:
```
# 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)})
sns.barplot(data = sales_state, x = 'State',y= 'Amount')
```

[87]:
```
<Axes: xlabel='State', ylabel='Amount'>
```

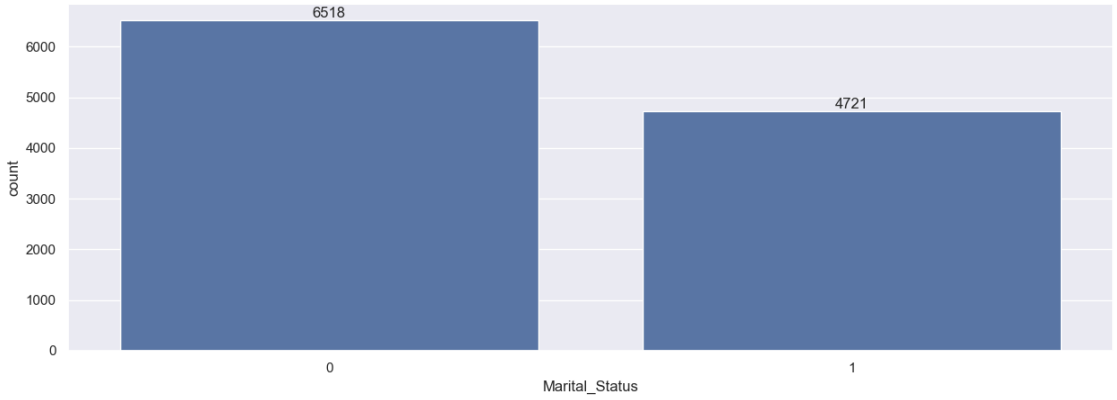


[ ]:
```
*From above graphs we can see that most of the orders & total sales/amount are
from Uttar Pradesh, Maharashtra and Karnataka respectively*
```

[ ]:
```
#MARITAL STATUS
```

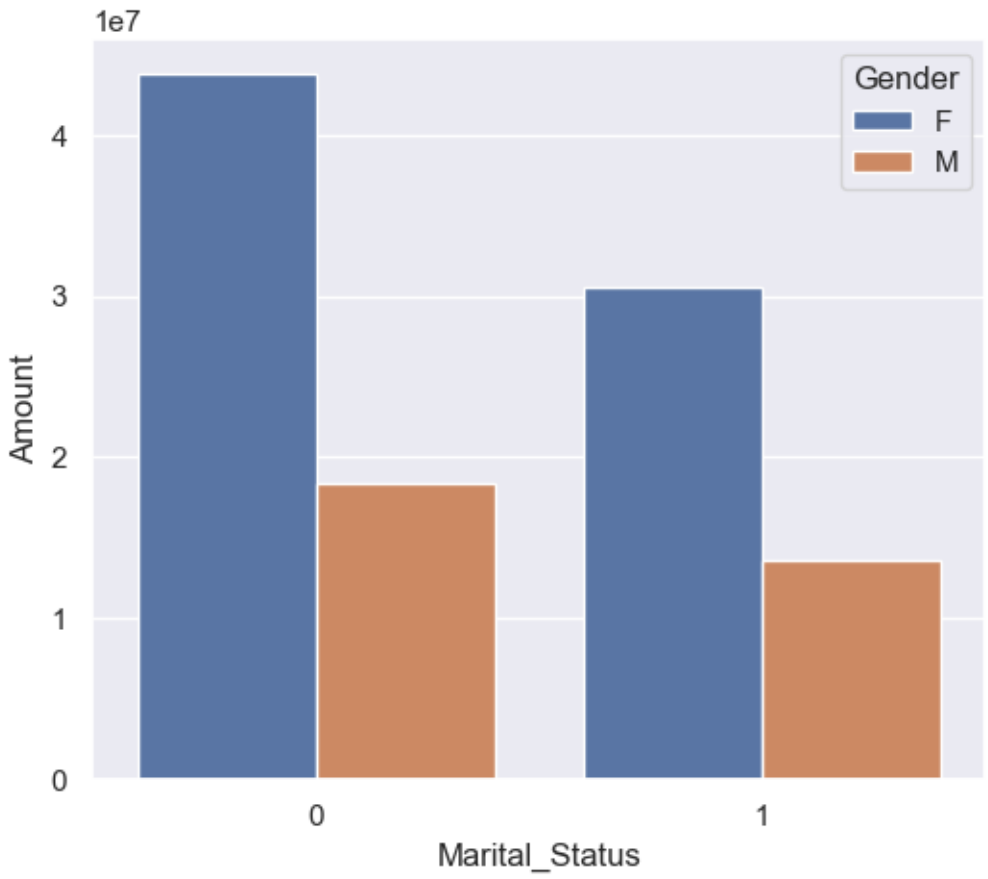
[91]:
```
ax = sns.countplot(data = df, x = 'Marital_Status')

sns.set(rc={'figure.figsize':(7,5)})
```

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



[93]:
```
sales_state = df.groupby(['Marital_Status', 'Gender'],␣
↪as_index=False)['Amount'].sum().sort_values(by='Amount', ascending=False)

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

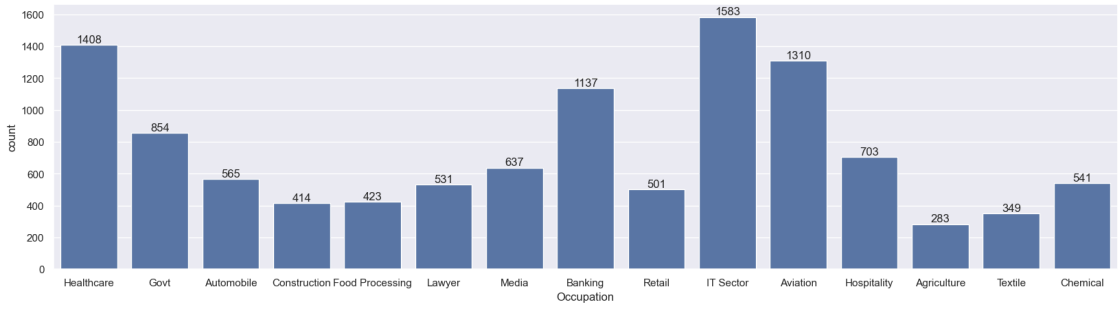
[93]: <Axes: xlabel='Marital_Status', ylabel='Amount'>



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

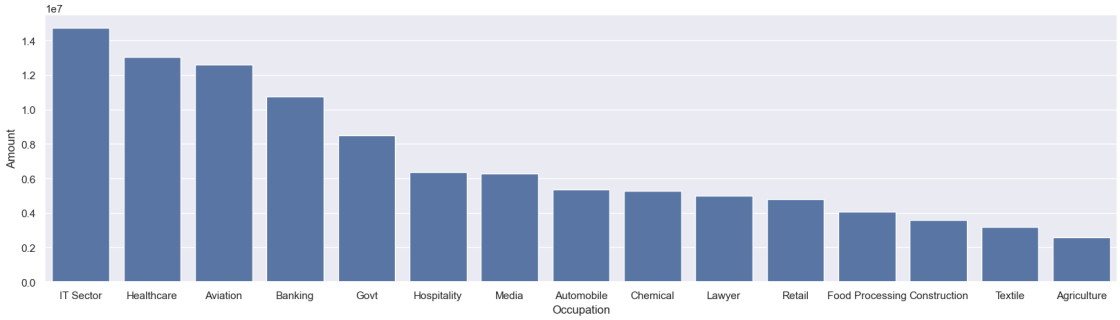
[ ]:
```
#OCCUPATION
```

[95]:
```
sns.set(rc={'figure.figsize':(20,5)})
ax = sns.countplot(data = df, x = 'Occupation')

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



[97]:
```
sales_state = df.groupby(['Occupation'], as_index=False)['Amount'].sum().
 ↪sort_values(by='Amount', ascending=False)

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

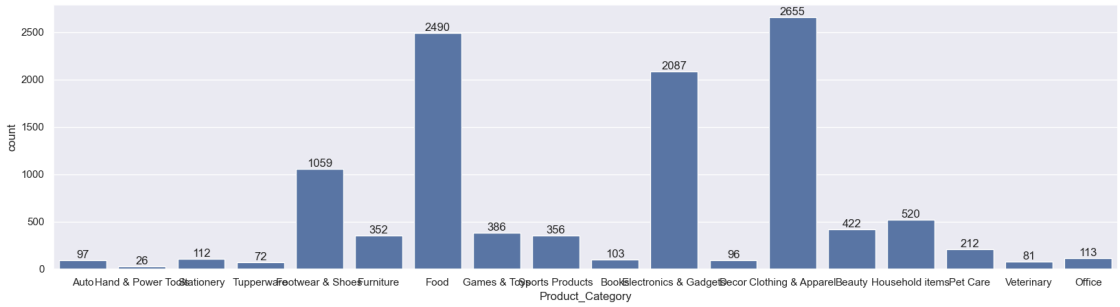
[97]:
```
<Axes: xlabel='Occupation', ylabel='Amount'>
```



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

[ ]:
```
#PRODUCT CATEGORY
```

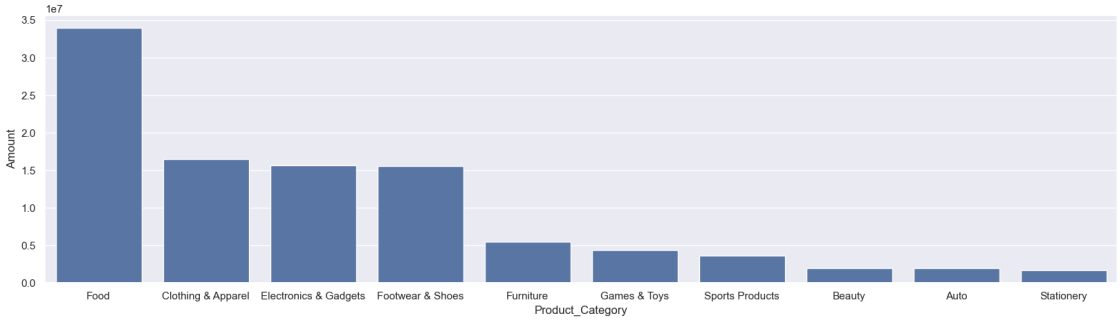
[137]:
```
sns.set(rc={'figure.figsize':(20,5)})
ax = sns.countplot(data = df, x = 'Product_Category')

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



[121]:
```
sales_state = df.groupby(['Product_Category'], as_index=False)['Amount'].sum().
 ↪sort_values(by='Amount', ascending=False).head(10)
```

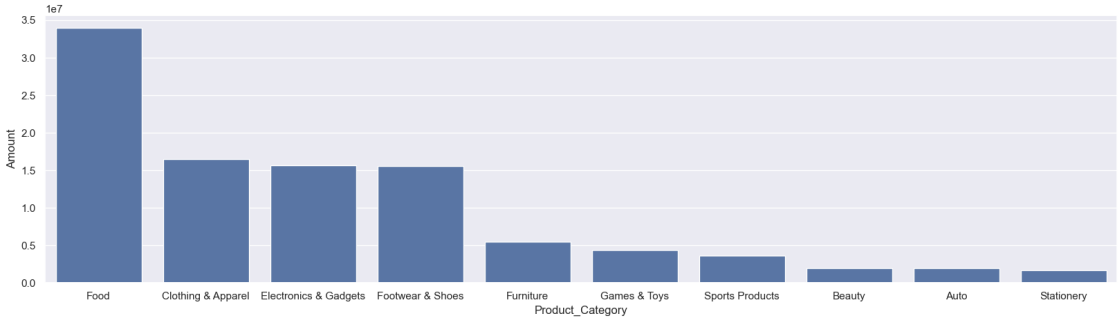
```
sns.set(rc={'figure.figsize':(20,5)})
sns.barplot(data = sales_state, x = 'Product_Category',y= 'Amount')
```

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



[123]:
```
sales_state = df.groupby(['Product_Category'], as_index=False)['Amount'].sum().
 ↪sort_values(by='Amount', ascending=False).head(10)

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

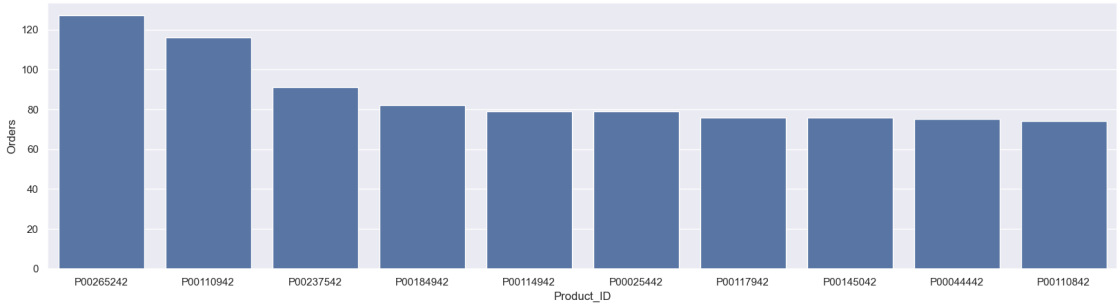
[123]: <Axes: xlabel='Product_Category', ylabel='Amount'>



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

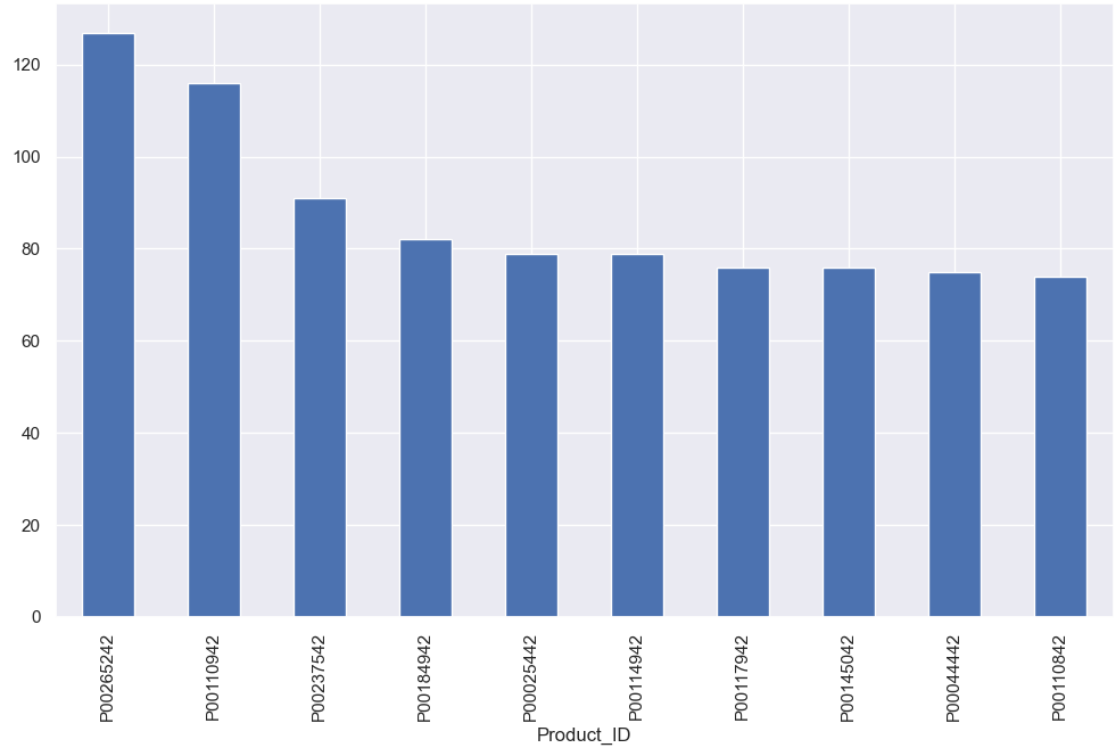
[125]:
```
sales_state = df.groupby(['Product_ID'], as_index=False)['Orders'].sum().
 ↪sort_values(by='Orders', ascending=False).head(10)

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

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



[139]:
```
# top 10 most sold products (same thing as above)
#ne way to plot grapf
#nlaegest is same as head
fig1, ax1 = plt.subplots(figsize=(12,7))
df.groupby('Product_ID')['Orders'].sum().nlargest(10).
 ↪sort_values(ascending=False).plot(kind='bar')
```

[139]: <Axes: 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*
```