Diwali_Sales_Analysis

April 21, 2025

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt                    # For Visualizing Data.
%matplotlib inline
import seaborn as sns
```

[2]:
```
df = pd.read_csv("C:\\Users\\DELL\\OneDrive\\Documents\\Diwali Sales Data.csv",␣
 ↪encoding="cp1252")
# to avoid encoding error, use'ISO-8859-1' or 'cp1252'.
```

[3]:
```
df.shape
```

[3]:
```
(11251, 15)
```

[4]:
```
df.head()
```

[4]:

	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

[5]:
```
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
```

[6]:
```
# drop unrelated/blank columns.
df.drop(["Status","unnamed1"], axis=1, inplace=True)
```

[7]:
```
# to check the deletion of blank columns.
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
```

[8]:
```
# Check for null values.
pd.isnull(df).sum()
```

[8]:
```
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
```

[9]:
```
df.shape
```

[9]:
```
(11251, 13)
```

[10]:
```
# drop null values.
df.dropna(inplace=True)
```

[11]:
```
# to check whether null values dropped or not.
pd.isnull(df).sum()
```

[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              0
dtype: int64
```

[12]:
```
# change data type.
df["Amount"] = df["Amount"].astype("int")
```

[13]:
```
df["Amount"].dtypes
```

[13]:
```
dtype('int32')
```

[14]:
```
df.columns
```

[14]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

[15]:
```
# describe() method returns description of the data in the DataFrame (i.e.␣
 ↪count, mean, std, etc).
df.describe()
```

[15]:

	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

[16]:
```
# using describe() for specific columns.
df[["Age","Orders","Amount"]].describe()
```

[16]:

	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

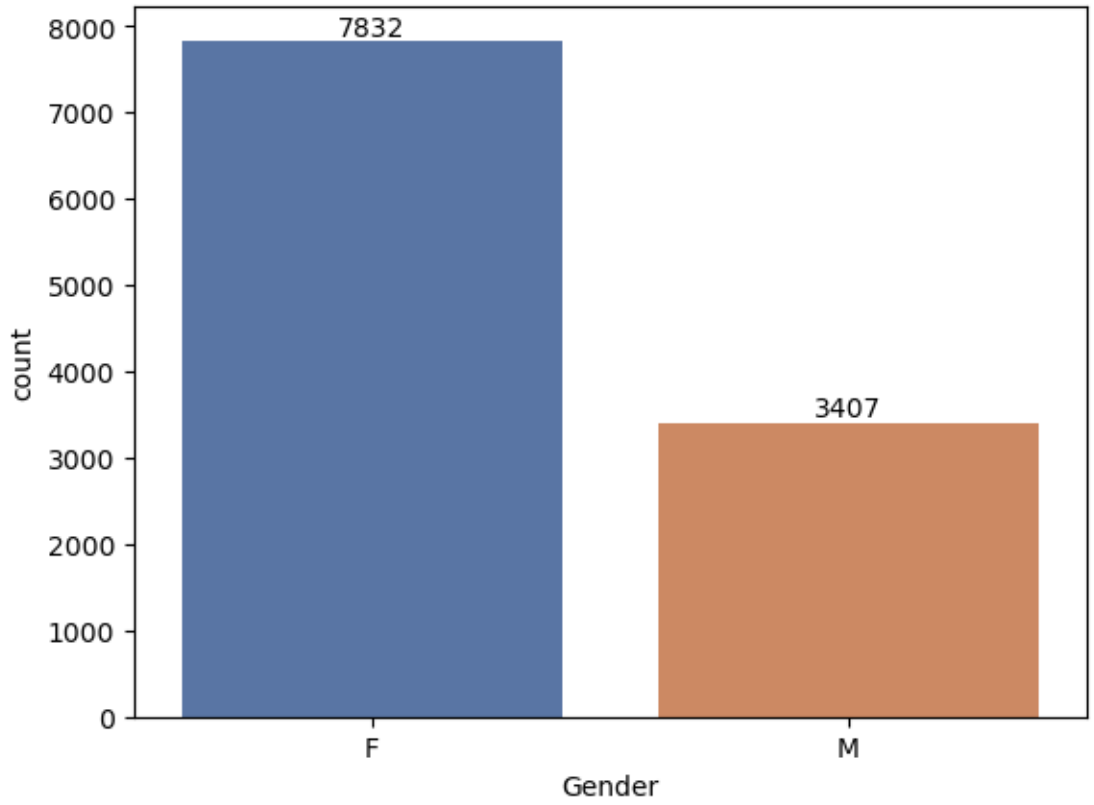
1 Exploratory Data Analysis

1.1 Gender

[19]:
```
# plotting a bar chart for Gender and it's count.

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

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

```
plt.show()
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_19048\2983538898.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

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

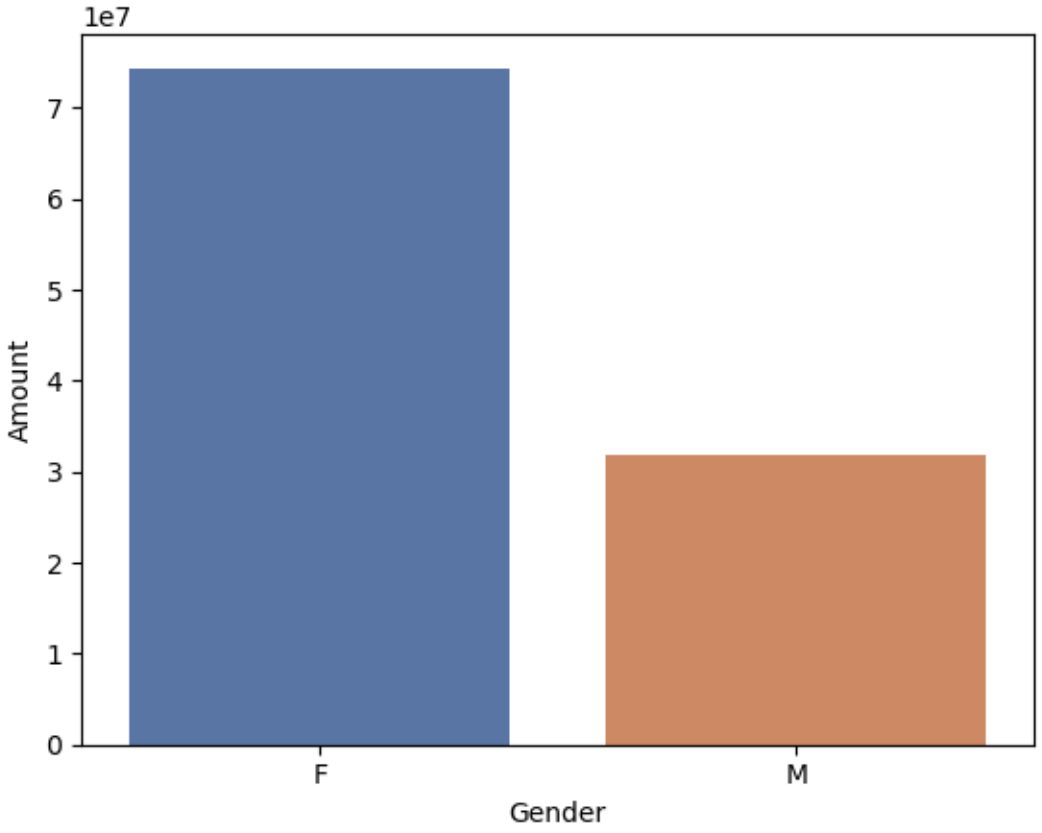


[20]:
```
# plotting a bar chart for gender vs total amount.

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, palette='deep')

plt.show()
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_19048\1266775660.py:5: FutureWarning:
```

```
Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.
```

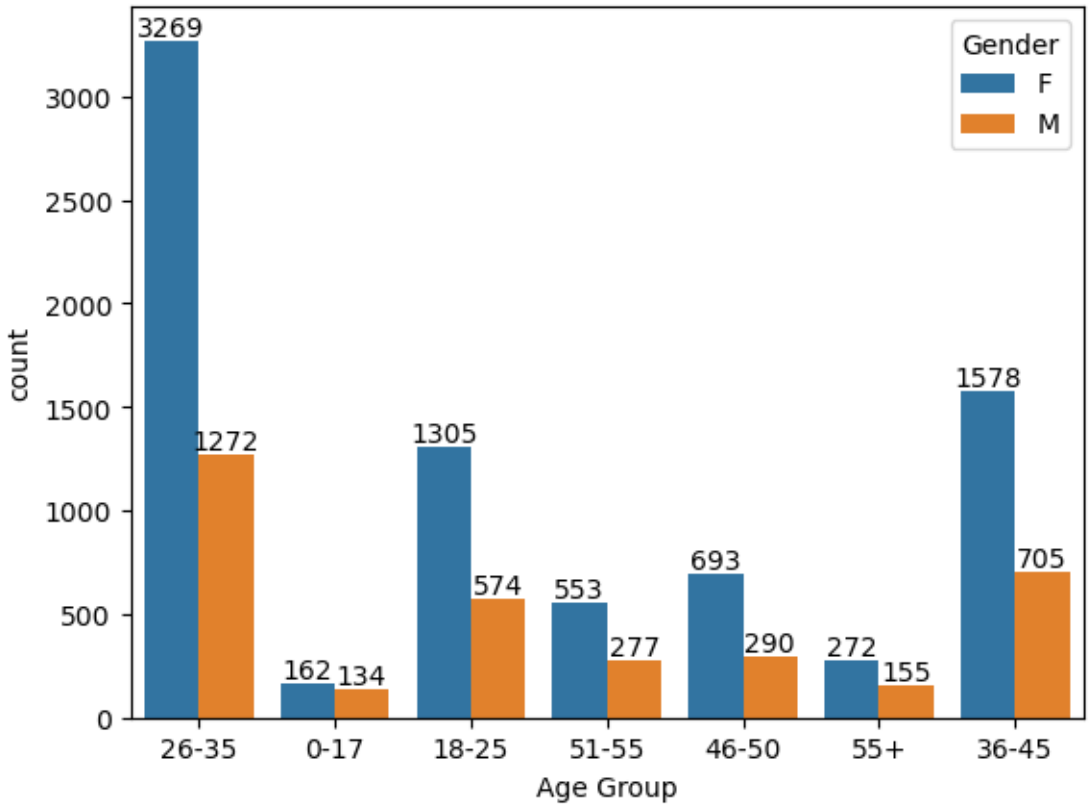
```
  sns.barplot(x='Gender', y='Amount', data=sales_gen, palette='deep')
```



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

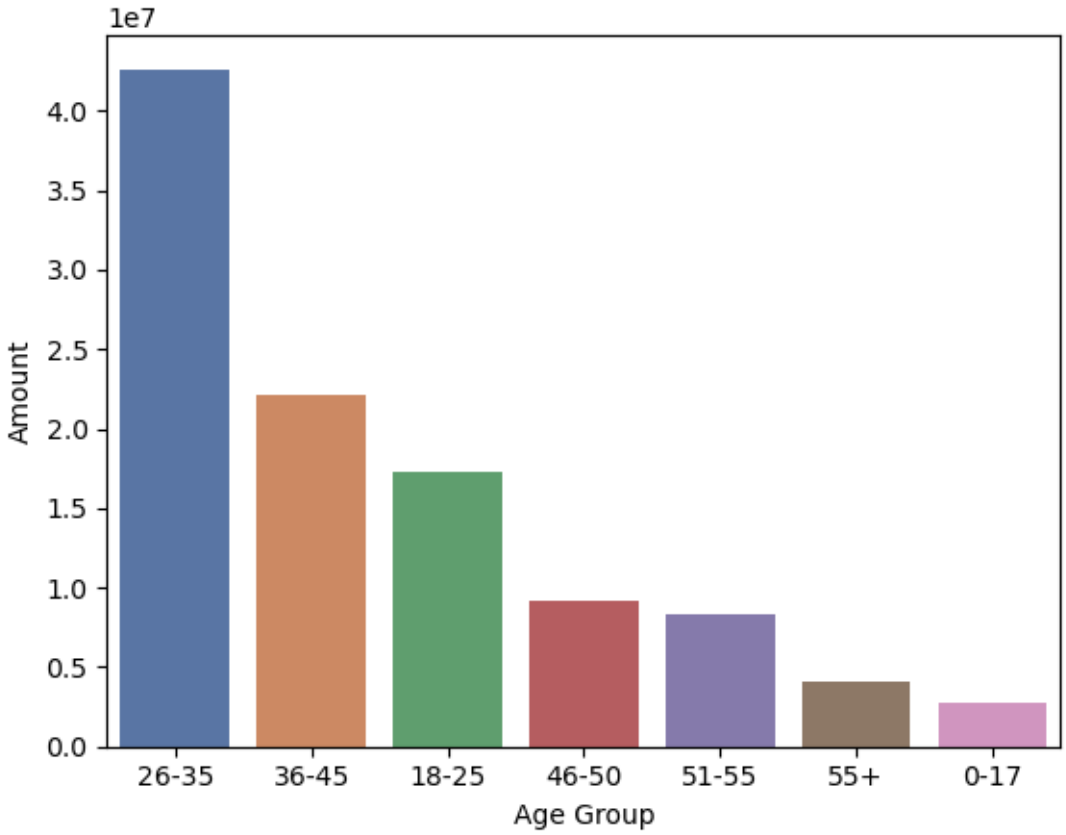
1.2 Age

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

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

plt.show()
```



[24]:
```
# 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, palette='deep')

plt.show()
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_19048\584450945.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

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

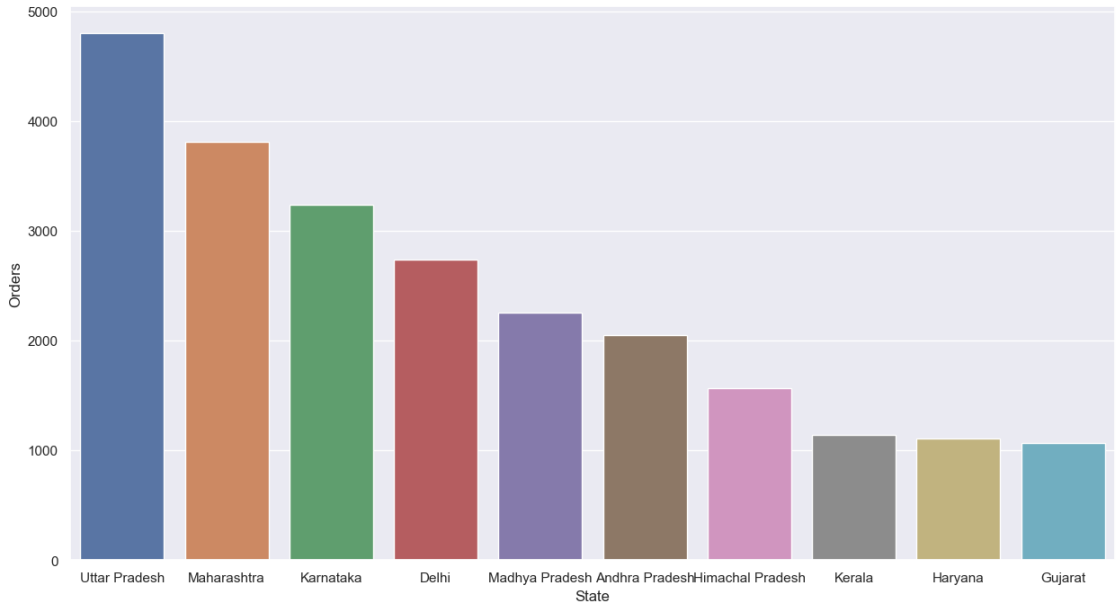


From above graphs we can see that most of the buyers are of age group between 26-35 yrs female.

1.3 State

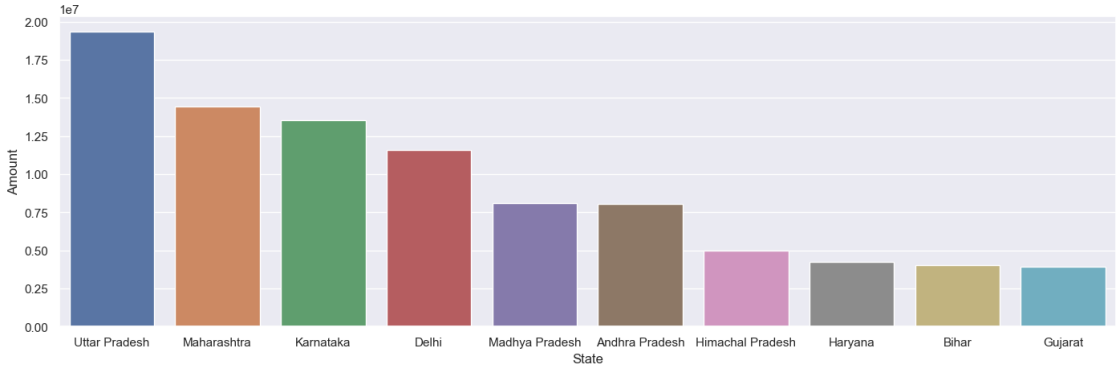
[66]:
```
# total number of orders from top 10 states

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

sns.set(rc={'figure.figsize':(15,8)})
sns.barplot(x = 'State',y= 'Orders', data = sales_state, palette='deep')

plt.show()
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_19048\1627813283.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

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



[76]:
```
# 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':(17,5)})
sns.barplot(x = 'State',y= 'Amount', data = sales_state, palette='deep')

plt.show()
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_19048\2123863288.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x = 'State',y= 'Amount', data = sales_state, palette='deep')
```

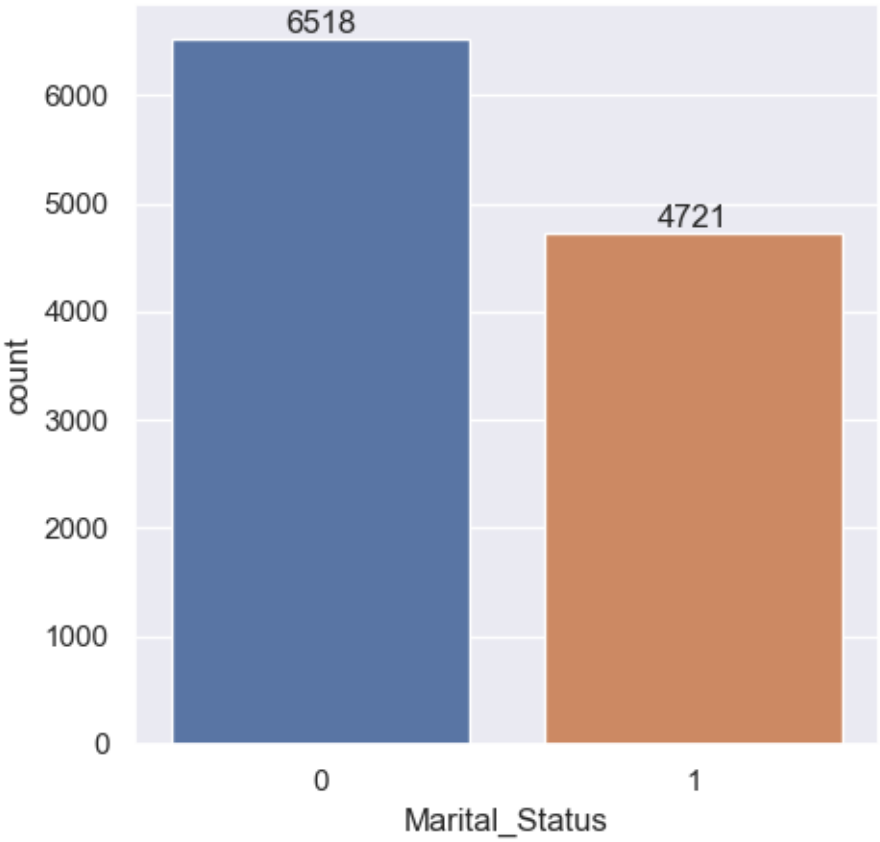


From above graphs we can see that most of the orders & total sales/amount are from Uttar Pradesh, Maharashtra and Karnataka respectively.

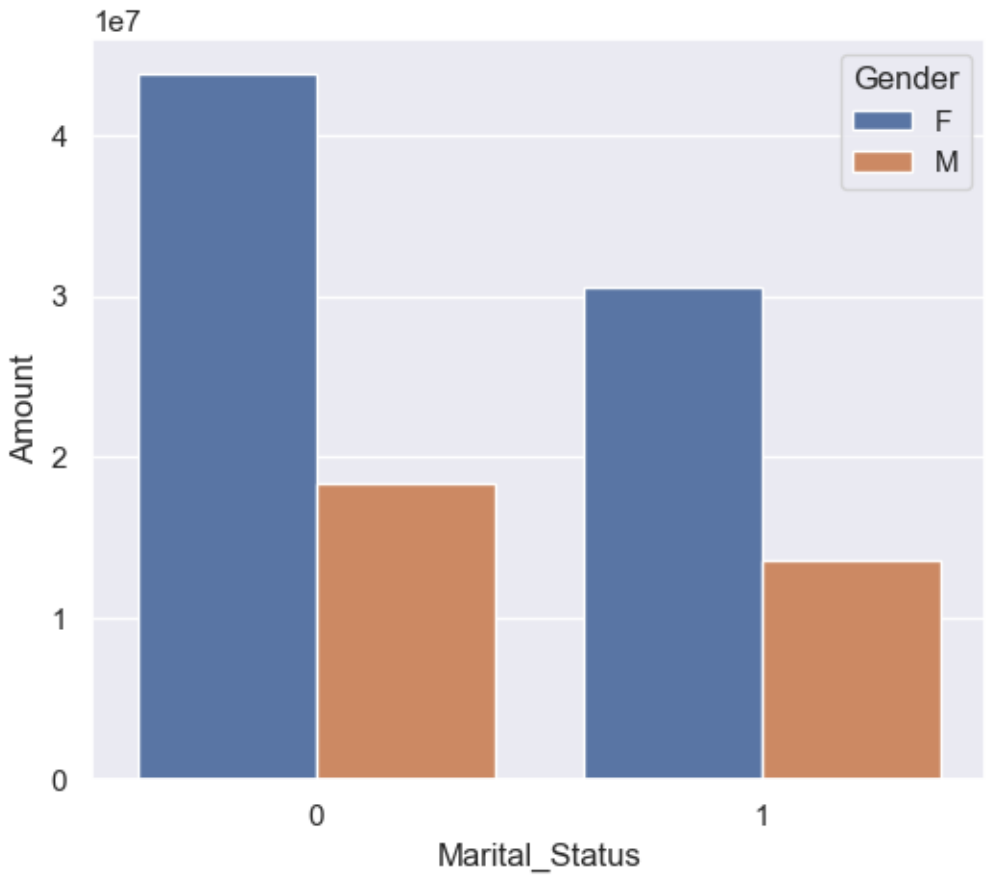
1.4 Marital Status

[82]:
```
ax = sns.countplot(x = 'Marital_Status', data = df, palette='deep')

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

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

plt.show()
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_19048\117303514.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  ax = sns.countplot(x = 'Marital_Status', data = df, palette='deep')
```



[32]:
```
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')

plt.show()
```

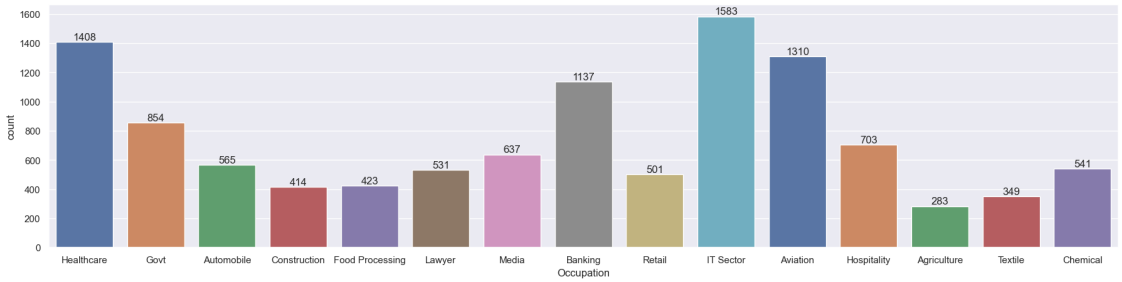


From above graphs we can see that most of the buyers are married (women) and they have high purchasing power.

1.5 Occupation

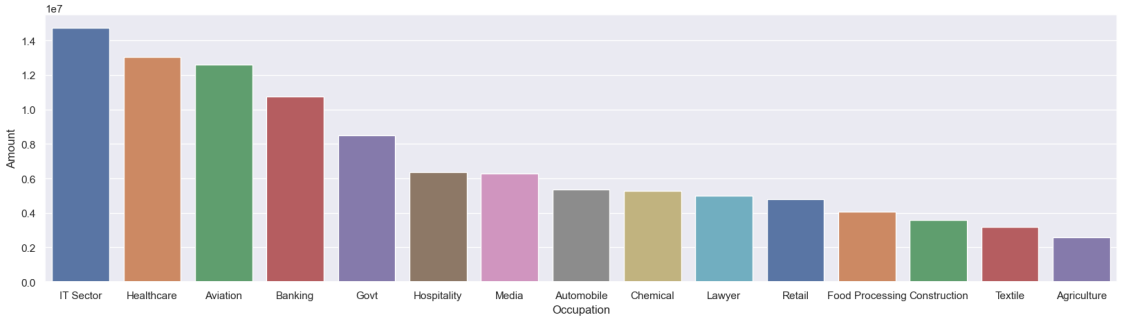
[78]:
```
sns.set(rc={'figure.figsize':(22,5)})
ax = sns.countplot(x = 'Occupation', data = df, palette='deep')

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

plt.show()
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_19048\155188593.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  ax = sns.countplot(x = 'Occupation', data = df, palette='deep')
```



[36]:
```
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(x = 'Occupation', y= 'Amount', data = sales_state, palette='deep')

plt.show()
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_19048\3203661734.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x = 'Occupation', y= 'Amount', data = sales_state, palette='deep')
```



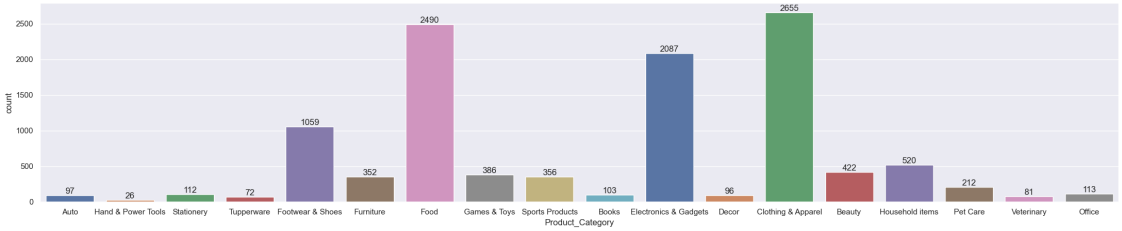
From above graphs we can see that most of the buyers are working in IT, Healthcare and Aviation sector.

1.6 Product Category

[102]:
```
sns.set(rc={'figure.figsize':(27,5)})
ax = sns.countplot(x = 'Product_Category', data = df, palette='deep')
```

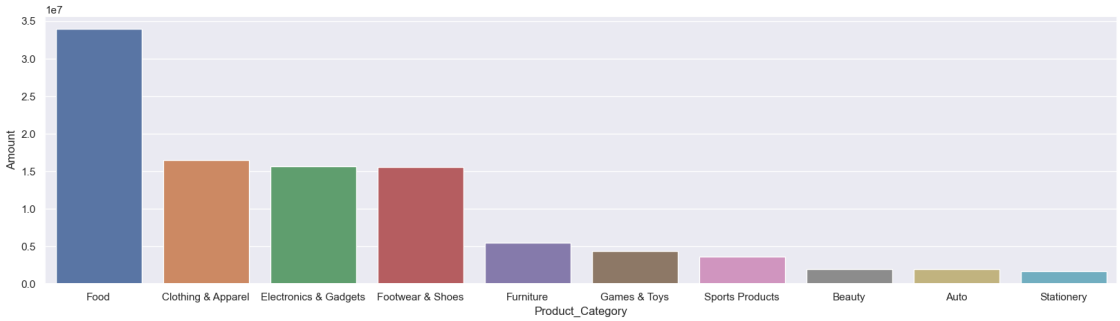
```
for bars in ax.containers:
    ax.bar_label(bars)

plt.show()
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_19048\822112133.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  ax = sns.countplot(x = 'Product_Category', data = df, palette='deep')
```



[110]:
```
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(x='Product_Category', y='Amount', data = sales_state,␣
 ↪palette='deep')

plt.show()
```

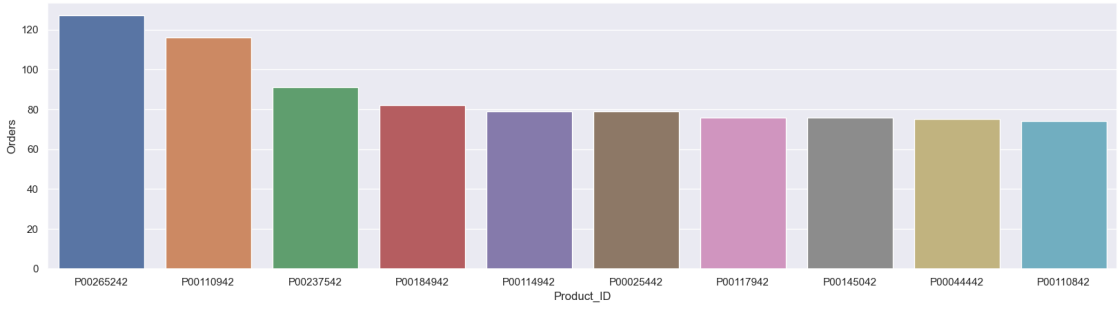
```
C:\Users\DELL\AppData\Local\Temp\ipykernel_19048\2715292879.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x='Product_Category', y='Amount', data = sales_state,
palette='deep')
```



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

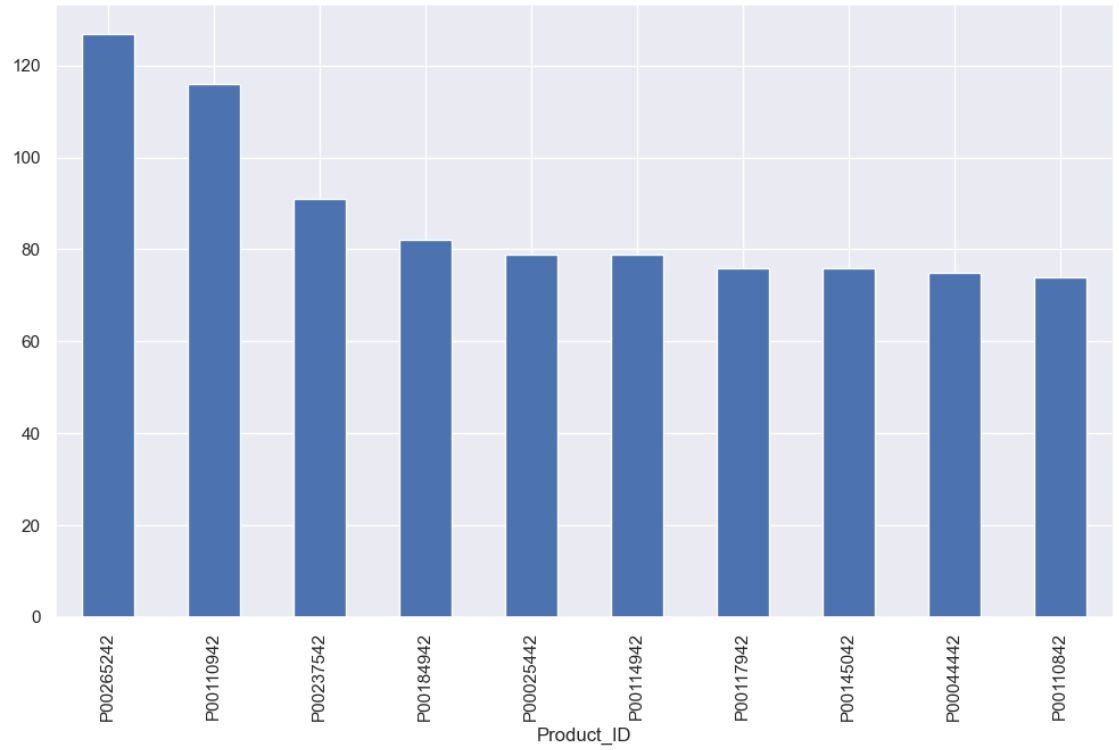
[112]:
```
sales_state = df.groupby(['Product_ID'], as_index=False)['Orders'].sum().
 ↪sort_values(by='Orders', ascending=False).head(10)

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

plt.show()
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_19048\1410169906.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x='Product_ID', y='Orders', data = sales_state, palette='deep')
```



[114]:
```
# 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).plot(kind='bar')
```

```
plt.show()
```



1.7 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.

Thank you!