LOADING LIBRARIES

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

LOADING DATASET FROM GITHUB

```
!git clone https://github.com/rishabhnmishra/Python_Diwali_Sales_Analysis
```

```
Cloning into 'Python_Diwali_Sales_Analysis'...
remote: Enumerating objects: 19, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 19 (delta 2), reused 0 (delta 0), pack-reused 13 (from 1)
Receiving objects: 100% (19/19), 666.76 KiB | 9.01 MiB/s, done.
Resolving deltas: 100% (4/4), done.
```

READING DATASET

```
df=pd.read_csv('/content/Python_Diwali_Sales_Analysis/Diwali Sales Data.csv',encoding='unicode_escape')
```

```
df
```

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	State	Zone	Occupation	Product_Category	O
0	1002903	Sanskriti	P00125942	F	26-35	28	0	Maharashtra	Western	Healthcare	Auto	
1	1000732	Kartik	P00110942	F	26-35	35	1	Andhra Pradesh	Southern	Govt	Auto	
2	1001990	Bindu	P00118542	F	26-35	35	1	Uttar Pradesh	Central	Automobile	Auto	
3	1001425	Sudevi	P00237842	M	0-17	16	0	Karnataka	Southern	Construction	Auto	
4	1000588	Joni	P00057942	M	26-35	28	1	Gujarat	Western	Food Processing	Auto	
...	...	...	...	...	...	...	...	...	...	...	...	
11246	1000695	Manning	P00296942	M	18-25	19	1	Maharashtra	Western	Chemical	Office	
11247	1004089	Reichenbach	P00171342	M	26-35	33	0	Haryana	Northern	Healthcare	Veterinary	
11248	1001209	Oshin	P00201342	F	36-45	40	0	Madhya Pradesh	Central	Textile	Office	
11249	1004023	Noonan	P00059442	M	36-45	37	0	Karnataka	Southern	Agriculture	Office	

Next steps: Generate code with df | View recommended plots | New interactive sheet

tells about no. of rows and columns

```
df.shape
```

```
(11251, 15)
```

give top 5 rows

```
df.head()
```

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	State	Zone	Occupation	Product_Category	Orders
0	1002903	Sanskriti	P00125942	F	26-35	28	0	Maharashtra	Western	Healthcare	Auto	1
1	1000732	Kartik	P00110942	F	26-35	35	1	Andhra Pradesh	Southern	Govt	Auto	3
2	1001990	Bindu	P00118542	F	26-35	35	1	Uttar Pradesh	Central	Automobile	Auto	3
3	1001425	Sudevi	P00237842	M	0-17	16	0	Karnataka	Southern	Construction	Auto	2

Next steps: Generate code with df View recommended plots New interactive sheet

DATA CLEANING

```
# taking all the information about the column
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
```

inplace = true /// saves the changes

```
# droping the columns which have 0 values
df.drop(['Status','unnamed1'],axis=1,inplace=True)
```

```
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
```

```
#checks the null value
pd.isnull(df)
```

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	State	Zone	Occupation	Product_Category	Orders	Amoun
0	False	False	False	False	False	False	False	False	False	False	False	False	Fals
1	False	False	False	False	False	False	False	False	False	False	False	False	Fals
2	False	False	False	False	False	False	False	False	False	False	False	False	Fals
3	False	False	False	False	False	False	False	False	False	False	False	False	Fals
4	False	False	False	False	False	False	False	False	False	False	False	False	Fals
...	...	...	...	...	...	...	...	...	...	...	...	...	.
11246	False	False	False	False	False	False	False	False	False	False	False	False	Fals
11247	False	False	False	False	False	False	False	False	False	False	False	False	Fals
11248	False	False	False	False	False	False	False	False	False	False	False	False	Fals
11249	False	False	False	False	False	False	False	False	False	False	False	False	Fals
11250	False	False	False	False	False	False	False	False	False	False	False	False	Fals

```
#checks the null value acc to the column and gives value in integer
pd.isnull(df).sum()
```

	0
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

```
#give the counting of no. of columns and rows
df.shape
```

```
(11251, 13)
```

```
#drop null values
df.dropna(inplace=True)
```

```
df.shape
```

```
(11239, 13)
```

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

```
# Checking datatype
df['Amount'].dtype
```

```
dtype('int64')
```

```
# showing column
df.columns
```

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

```
#rename columns but it will not save because we are not using inplace or initialization
df.rename(columns={'Marital_Status':'saddi'})
```

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	saddi	State	Zone	Occupation	Product_Category	Orders	Am
0	1002903	Sanskriti	P00125942	F	26-35	28	0	Maharashtra	Western	Healthcare	Auto	1	2
1	1000732	Kartik	P00110942	F	26-35	35	1	Andhra Pradesh	Southern	Govt	Auto	3	2
2	1001990	Bindu	P00118542	F	26-35	35	1	Uttar Pradesh	Central	Automobile	Auto	3	2
3	1001425	Sudevi	P00237842	M	0-17	16	0	Karnataka	Southern	Construction	Auto	2	2
4	1000588	Joni	P00057942	M	26-35	28	1	Gujarat	Western	Food Processing	Auto	2	2
...	...	...	...	...	...	...	...	...	...	...	...	...	
11246	1000695	Manning	P00296942	M	18-25	19	1	Maharashtra	Western	Chemical	Office	4	
11247	1004089	Reichenbach	P00171342	M	26-35	33	0	Haryana	Northern	Healthcare	Veterinary	3	
11248	1001209	Oshin	P00201342	F	36-45	40	0	Madhya Pradesh	Central	Textile	Office	4	
11249	1004023	Noonan	P00059442	M	36-45	37	0	Karnataka	Southern	Agriculture	Office	3	

```
#decribe
df.describe()
```

	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

```
# describing specefic columns as we want
df[['Age','Orders','Amount']].describe()
```

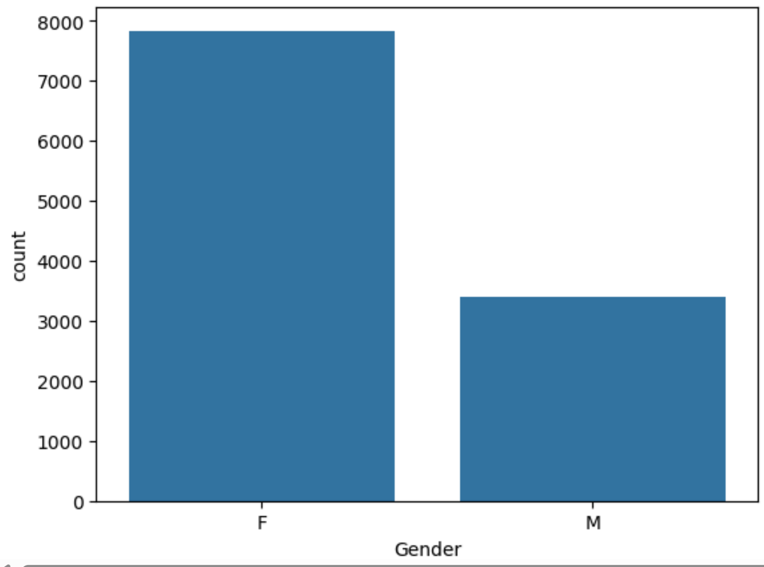
	Age	Orders	Amount
count	11239.000000	11239.000000	11239.000000
mean	35.410357	2.489634	9453.610553
std	12.753866	1.114967	5222.355168
min	12.000000	1.000000	188.000000
25%	27.000000	2.000000	5443.000000
50%	33.000000	2.000000	8109.000000
75%	43.000000	3.000000	12675.000000
max	92.000000	4.000000	23952.000000

EXPLORATORY DATA ANALYSIS

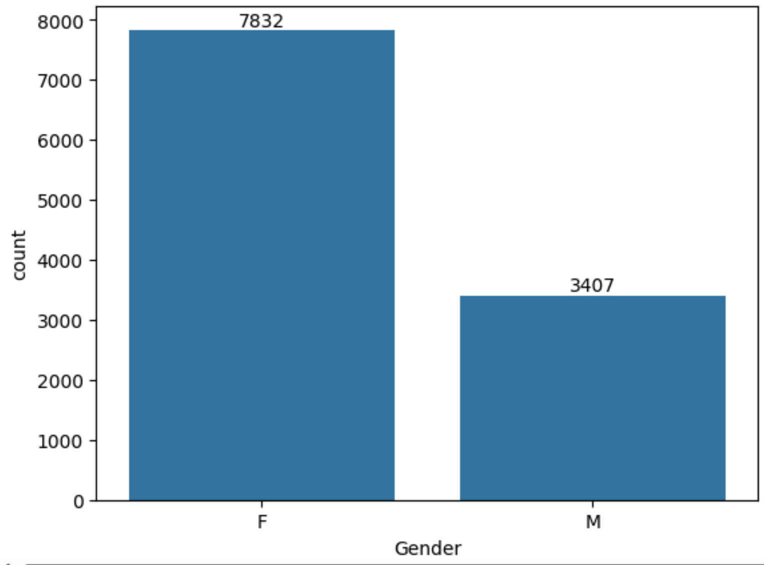
Gender

```
#male vs female
sns.countplot(x='Gender',data=df)
```

```
<Axes: xlabel='Gender', ylabel='count'>
```

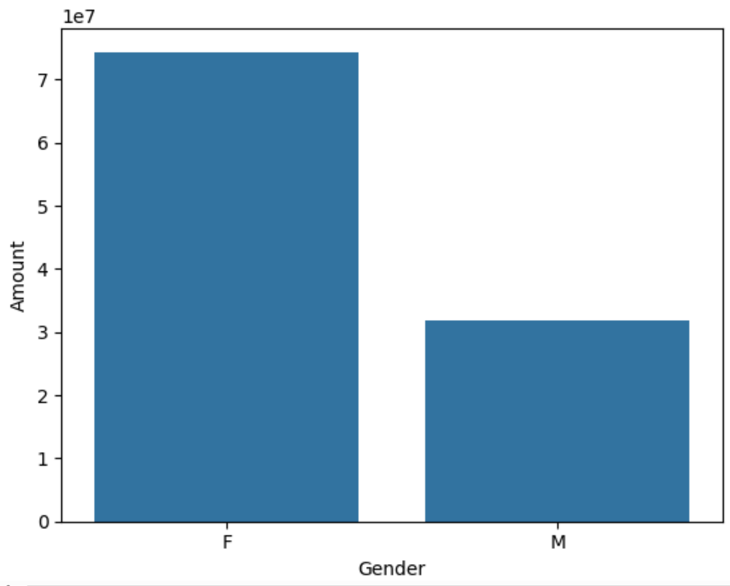


```
#bar graph with numbers
ax = sns.countplot(x='Gender',data=df)
for bars in ax.containers:
  ax.bar_label(bars)
```



```
# creating bar graph by joining total purchase of male and female and checking who has more purchasing power
df_gender=df.groupby(['Gender'],as_index=False)['Amount'].sum().sort_values(by='Amount',ascending=False)
sns.barplot(x='Gender',y='Amount',data=df_gender)
```

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



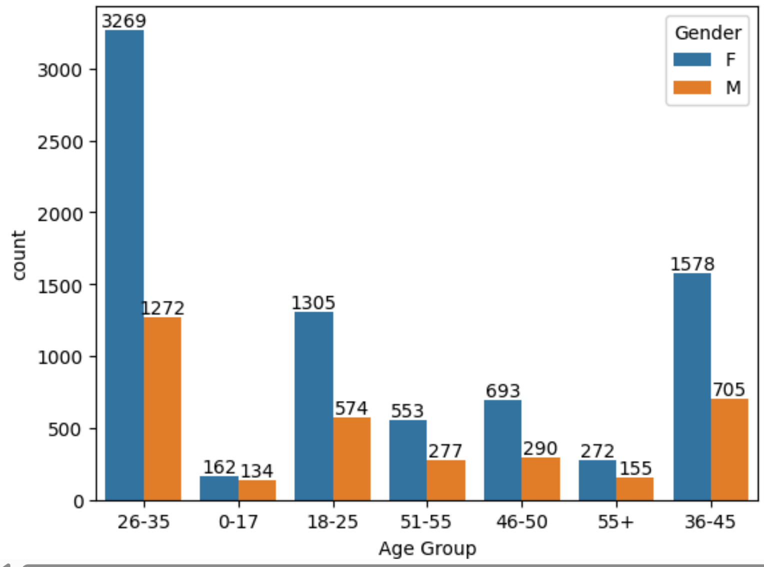
// This shows female has more purchasing power

Age

```
df.columns
```

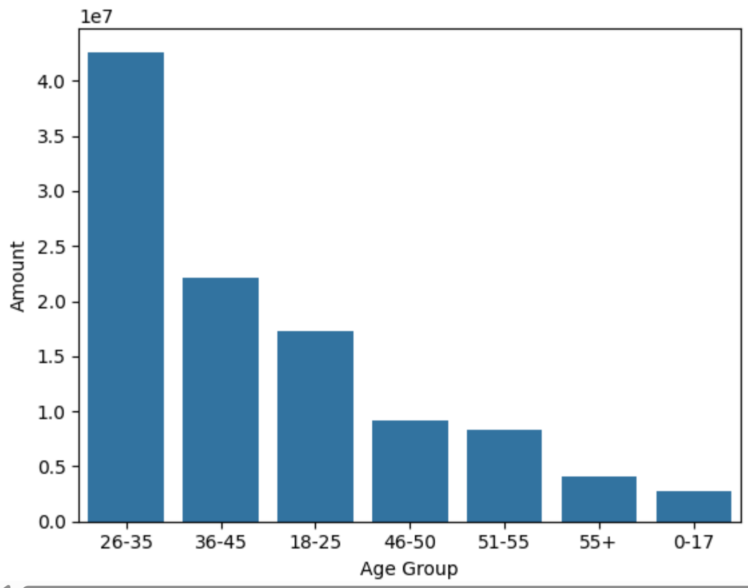
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

```
# graph on the basis of age group on hue level division of gender
ax=sns.countplot (x='Age Group',data=df , hue='Gender')
#this next line adds marking over bars
for bars in ax.containers:
  ax.bar_label(bars)
```



```
#total amount vs age group
df_age=df.groupby(['Age Group'],as_index=False)['Amount'].sum().sort_values(by='Amount',ascending=False)
sns.barplot(x='Age Group',y='Amount',data=df_age)
```

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



From above graph we can say that women of 26-35 year old has maximum purchasing power

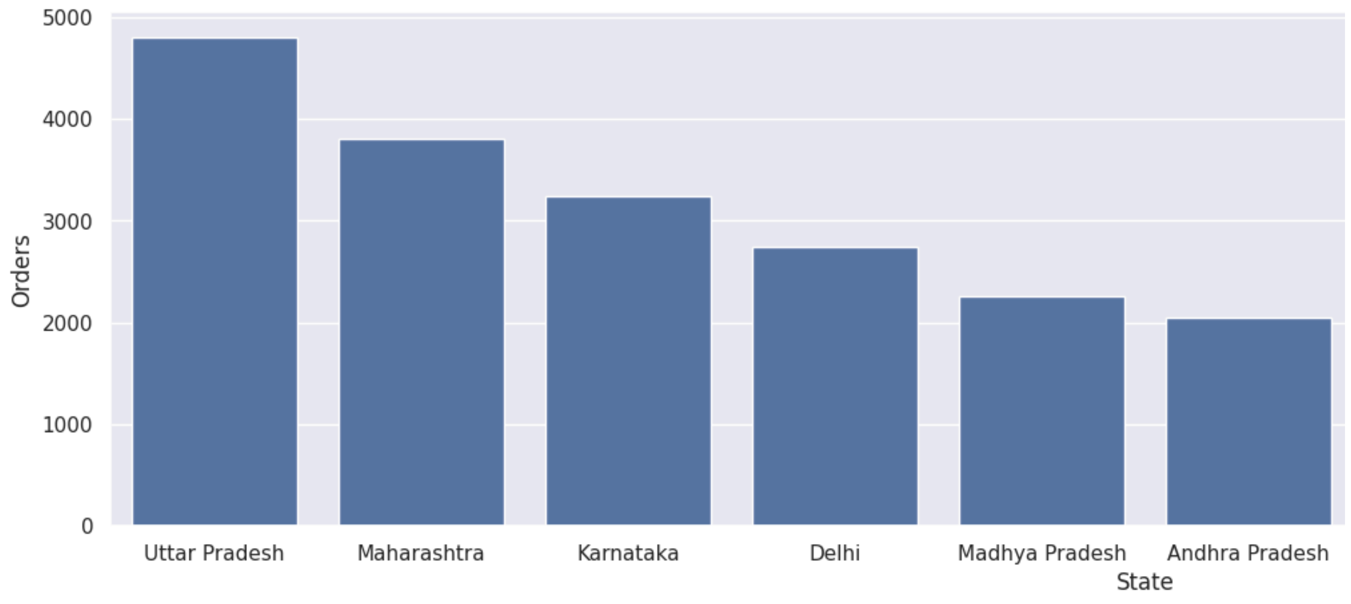
State

```
df.columns
```

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

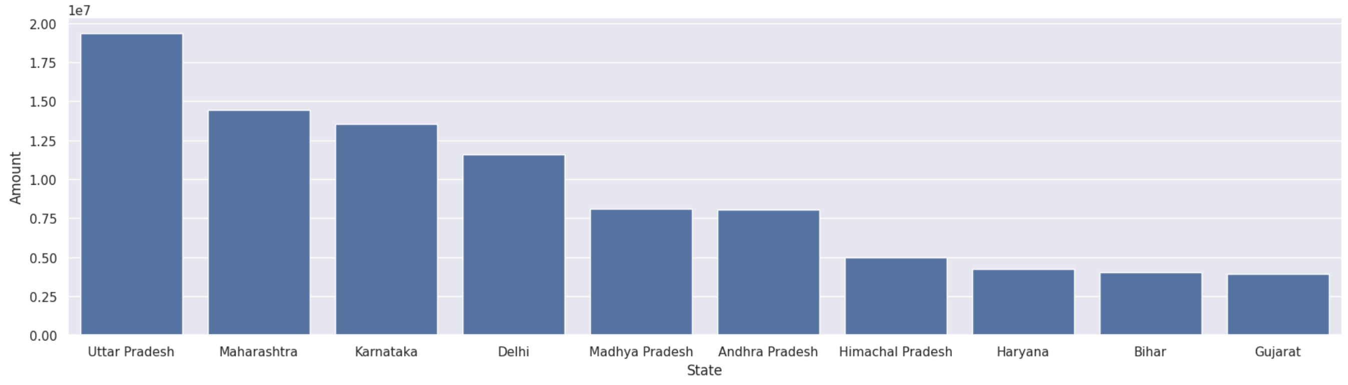
```
# top 10 states acc to the total number of orders
sales_state=df.groupby(['State'],as_index=False)['Orders'].sum().sort_values(by='Orders',ascending=False).head(10)
sns.set(rc={'figure.figsize':(20,5)})
sns.barplot(x='State',y='Orders',data=sales_state)
```

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



```
# top 10 states acc to the total amount of sales
sales_state=df.groupby(['State'],as_index=False)['Amount'].sum().sort_values(by='Amount',ascending=False).head(10)
sns.set(rc={'figure.figsize':(20,5)})
sns.barplot(x='State',y='Amount',data=sales_state)
```

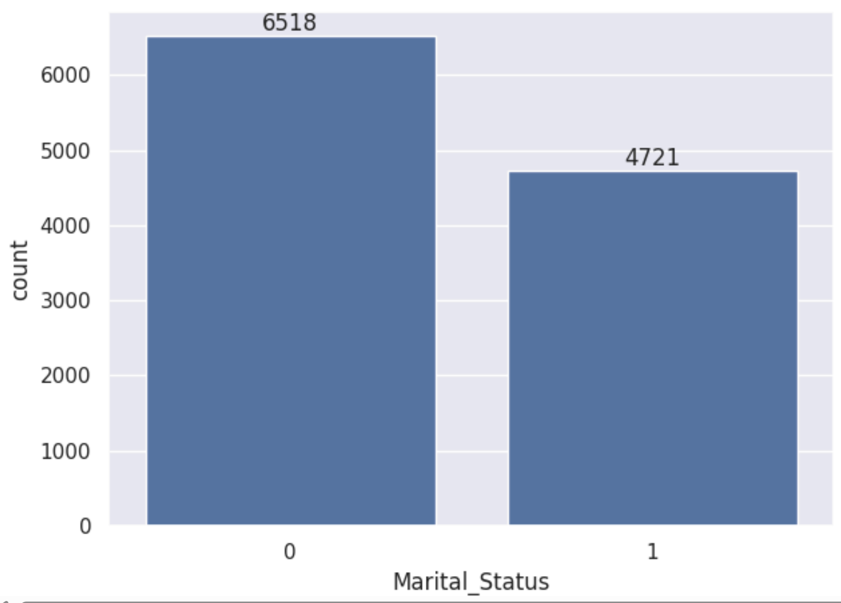
```
<Axes: xlabel='State', ylabel='Amount'>
```



from both these graph we understand on basis of no. of orders kerala is at 8 position but when value of orders comes in its not even in list

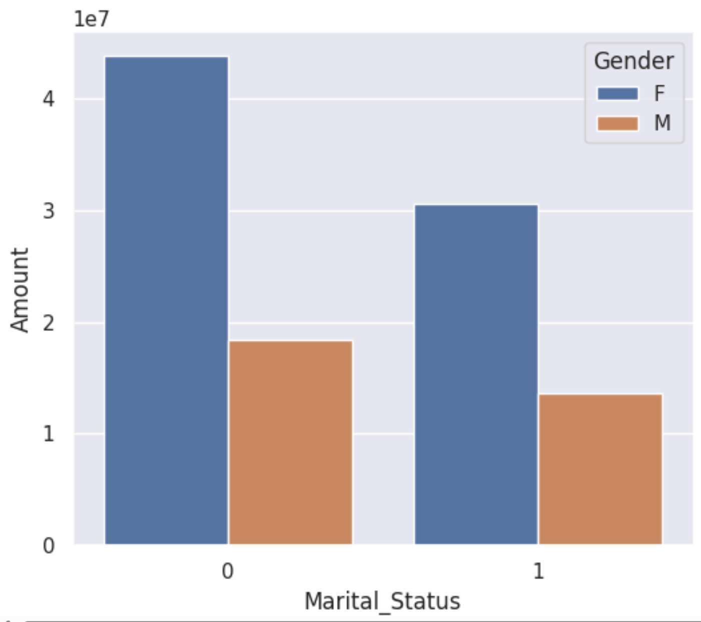
Marital status

```
# who buy more single vs couple
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 martital status who is buying more
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')
```

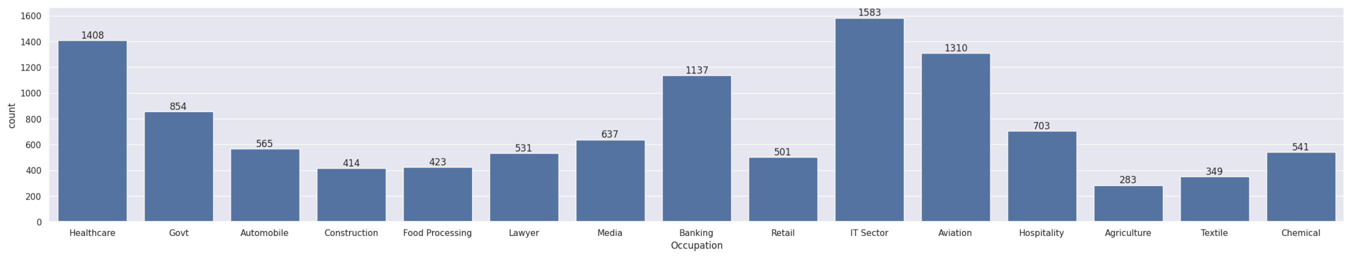
```
<Axes: xlabel='Marital_Status', ylabel='Amount'>
```



// Most of the purchases is done by married women

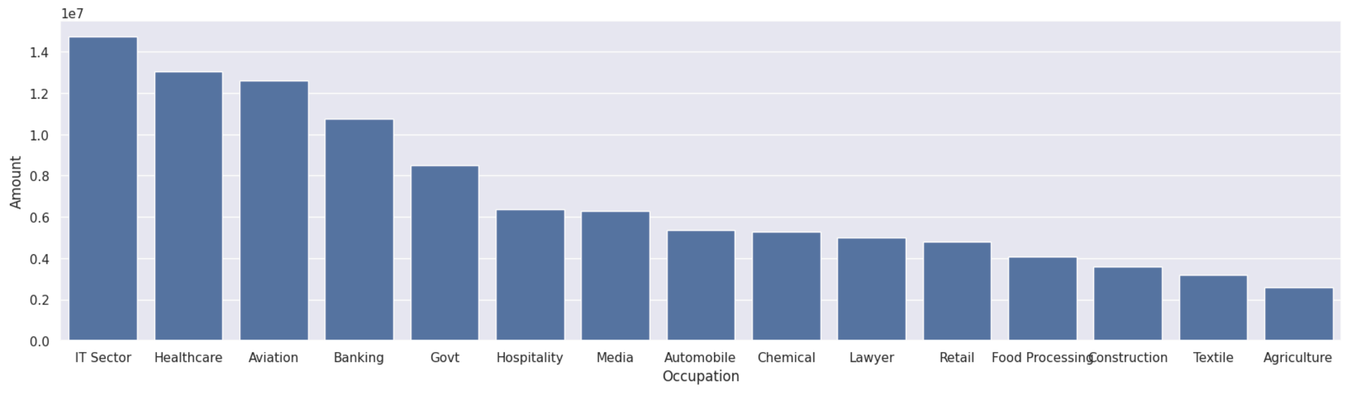
Occupation

```
ax=sns.countplot(data=df,x='Occupation')
sns.set(rc={'figure.figsize':(30,4)})
for bars in ax.containers:
  ax.bar_label(bars)
```



```
# diff occupation on graph who has maximum purchasing power
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)
```

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

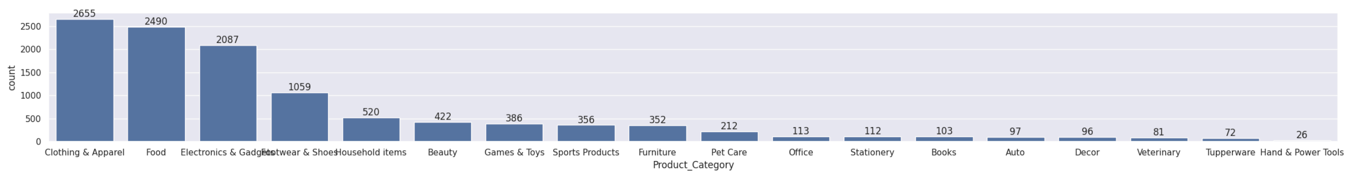


// From the above two graph we conclude that IT sector person orders maximum and have highest purchasing power

Product category

```
# which product category has maximum sales (on behalf of no. of sales)

ax=sns.countplot(data=df,x='Product_Category',order=order)      # just by adding order = order this graph becomes decending in order
sns.set(rc={'figure.figsize':(30,3)})
for bars in ax.containers:
  ax.bar_label(bars)
```


count
Product_Category
Clothing & Apparel
Food
Electronics & Gadgets
Footwear & Shoes
Household items
Beauty
Games & Toys
Sports Products
Furniture
Pet Care
Office
Stationery
Books
Auto
Decor
Veterinary
Tupperware
Hand & Power Tools