```
!pip install numpy
!pip install pandas
!pip install matplotlib
!pip install seaborn
```

```
Requirement already satisfied: numpy in /Library/Frameworks/Python.framework/V
Requirement already satisfied: pandas in /Library/Frameworks/Python.framework/
Requirement already satisfied: numpy>=1.23.2 in /Library/Frameworks/Python.fra
Requirement already satisfied: python-dateutil>=2.8.2 in /Library/Frameworks/P
Requirement already satisfied: pytz>=2020.1 in /Library/Frameworks/Python.fram
Requirement already satisfied: tzdata>=2022.7 in /Library/Frameworks/Python.fr
Requirement already satisfied: six>=1.5 in /Library/Frameworks/Python.framewor
Requirement already satisfied: matplotlib in /Library/Frameworks/Python.framew
Requirement already satisfied: contourpy>=1.0.1 in /Library/Frameworks/Python.
Requirement already satisfied: cycler>=0.10 in /Library/Frameworks/Python.fram
Requirement already satisfied: fonttools>=4.22.0 in /Library/Frameworks/Python
Requirement already satisfied: kiwisolver>=1.3.1 in /Library/Frameworks/Python
Requirement already satisfied: numpy>=1.23 in /Library/Frameworks/Python.frame
Requirement already satisfied: packaging>=20.0 in /Library/Frameworks/Python.f
Requirement already satisfied: pillow>=8 in /Library/Frameworks/Python.framewo
Requirement already satisfied: pyparsing>=2.3.1 in /Library/Frameworks/Python.
Requirement already satisfied: python-dateutil>=2.7 in /Library/Frameworks/Pyt
Requirement already satisfied: six>=1.5 in /Library/Frameworks/Python.framewor
Requirement already satisfied: seaborn in /Library/Frameworks/Python.framework
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /Library/Frameworks/Pyt
Requirement already satisfied: pandas>=1.2 in /Library/Frameworks/Python.frame
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /Library/Frameworks/
Requirement already satisfied: contourpy>=1.0.1 in /Library/Frameworks/Python.
Requirement already satisfied: cycler>=0.10 in /Library/Frameworks/Python.fram
Requirement already satisfied: fonttools>=4.22.0 in /Library/Frameworks/Python
Requirement already satisfied: kiwisolver>=1.3.1 in /Library/Frameworks/Python
Requirement already satisfied: packaging>=20.0 in /Library/Frameworks/Python.f
Requirement already satisfied: pillow>=8 in /Library/Frameworks/Python.framewo
Requirement already satisfied: pyparsing>=2.3.1 in /Library/Frameworks/Python.
Requirement already satisfied: python-dateutil>=2.7 in /Library/Frameworks/Pyt
Requirement already satisfied: pytz>=2020.1 in /Library/Frameworks/Python.fram
Requirement already satisfied: tzdata>=2022.7 in /Library/Frameworks/Python.fr
Requirement already satisfied: six>=1.5 in /Library/Frameworks/Python.framewor
```

```
# importing libraries

import numpy as np #parsing arrays
import pandas as pd #parsing tables
import matplotlib.pyplot as plt # visualizing data
%matplotlib inline
import seaborn as sns #visualization
```

```
# import csv f
df = pd.read_csv('/Users/kaustubhchati/Downloads/Python_Diwali_Sales_Analysis-mai
```

```
df.shape
```

```
(11251, 15)
```

```
df.head(15)
```

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	S
0	1002903	Sanskriti	P00125942	F	26-35	28	0	Mahar
1	1000732	Kartik	P00110942	F	26-35	35	1	Andhra Pr
2	1001990	Bindu	P00118542	F	26-35	35	1	Uttar Pr
3	1001425	Sudevi	P00237842	M	0-17	16	0	Karr
4	1000588	Joni	P00057942	M	26-35	28	1	G
5	1000588	Joni	P00057942	M	26-35	28	1	Him Pr
6	1001132	Balk	P00018042	F	18-25	25	1	Uttar Pr
7	1002092	Shivangi	P00273442	F	55+	61	0	Mahar
8	1003224	Kushal	P00205642	M	26-35	35	0	Uttar Pr
9	1003650	Ginny	P00031142	F	26-35	26	1	Andhra Pr
10	1003829	Harshita	P00200842	M	26-35	34	0	
11	1000214	Kargatis	P00119142	F	18-25	20	0	Andhra Pr
12	1004035	Elijah	P00080342	F	18-25	20	1	Andhra Pr
13	1001680	Vasudev	P00324942	M	26-35	26	1	Andhra Pr
14	1003858	Cano	P00293742	M	46-50	46	1	M Pr

```
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
```

```
#drop unrelated/blank columns
df.drop(["Status", "unnamed1"], axis=1, inplace=True)
# axis refers to deletion of entire vertical column
```

```
#check for null values
pd.isnull(df)
```

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	Stat
0	False	False	False	False	False	False	False	Fals
1	False	False	False	False	False	False	False	Fals
2	False	False	False	False	False	False	False	Fals
3	False	False	False	False	False	False	False	Fals
4	False	False	False	False	False	False	False	Fals
...	...	...	...	...	...	...	...	.
11246	False	False	False	False	False	False	False	Fals
11247	False	False	False	False	False	False	False	Fals
11248	False	False	False	False	False	False	False	Fals
11249	False	False	False	False	False	False	False	Fals
11250	False	False	False	False	False	False	False	Fals

11251 rows × 13 columns

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

```
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
```

```
# drop null values
df.dropna(inplace=True)
#dropna is function to drop null values
#inplace is used to save the command for future
```

```
pd.isnull(df).sum()
# rechecking to see if the null values are deleted
```

```
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
```

```
# change data type
df['Amount'] = df['Amount'].astype('int')
#rounding amount value to the nearest rupee
```

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

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

```
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 column
df.rename(columns= {'Marital_Status':'Shaadi'})
# {} entail dictionary in Key: Value format
```

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Shaadi	State
0	1002903	Sanskriti	P00125942	F	26-35	28	0	Maharashtra
1	1000732	Kartik	P00110942	F	26-35	35	1	Andhra Pradesh
2	1001990	Bindu	P00118542	F	26-35	35	1	Uttar Pradesh
3	1001425	Sudevi	P00237842	M	0-17	16	0	Karnataka
4	1000588	Joni	P00057942	M	26-35	28	1	Gujarat
...	...	...	...	...	...	...	...	...
11246	1000695	Manning	P00296942	M	18-25	19	1	Maharashtra
11247	1004089	Reichenbach	P00171342	M	26-35	33	0	Haryana
11248	1001209	Oshin	P00201342	F	36-45	40	0	Madhya Pradesh
11249	1004023	Noonan	P00059442	M	36-45	37	0	Karnataka
11250	1002744	Brumley	P00281742	F	18-25	19	0	Maharashtra

```
# describe() method returns description of the data in the DataFrame (i.e. count,
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

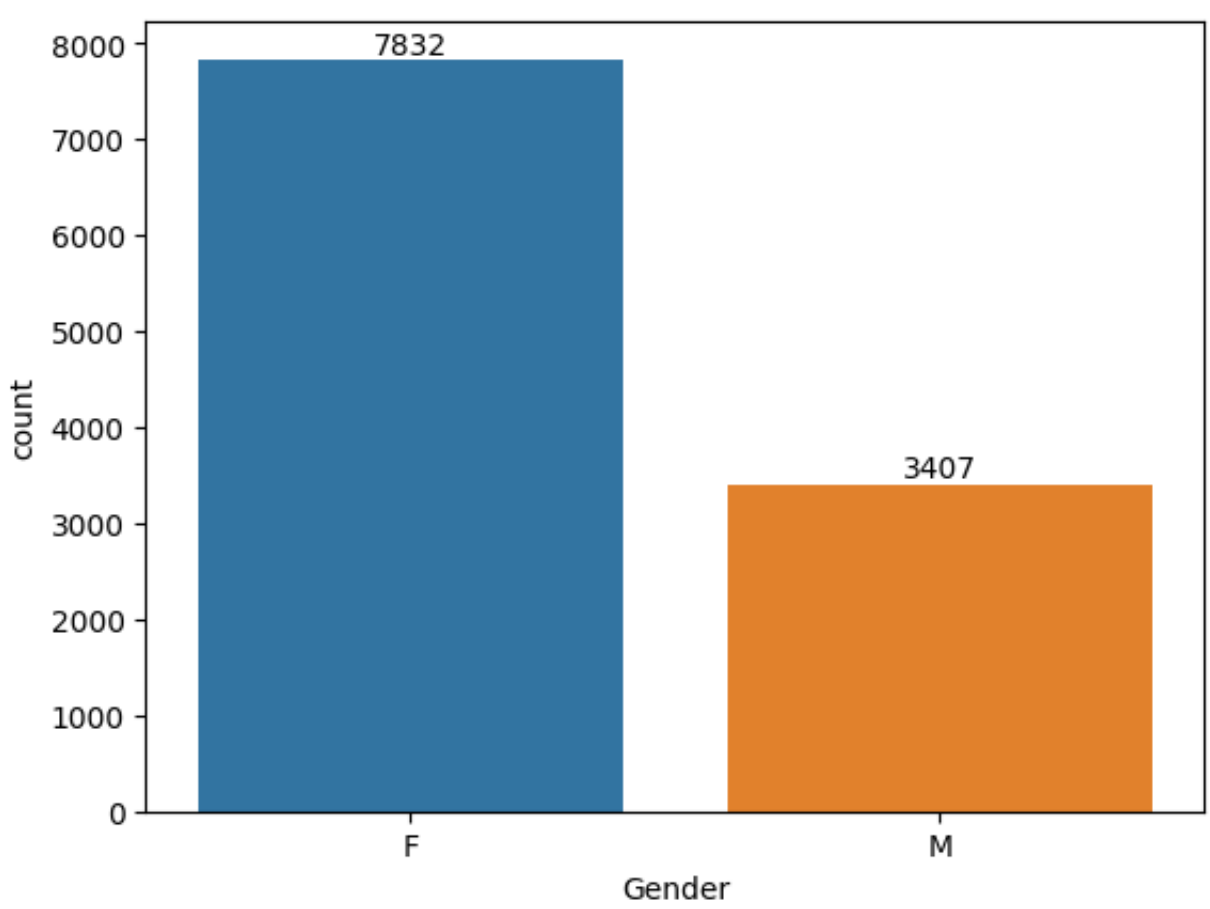
```
# describe() for specific columns
df[['Age', 'Orders', 'Amount']].describe()
```

	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

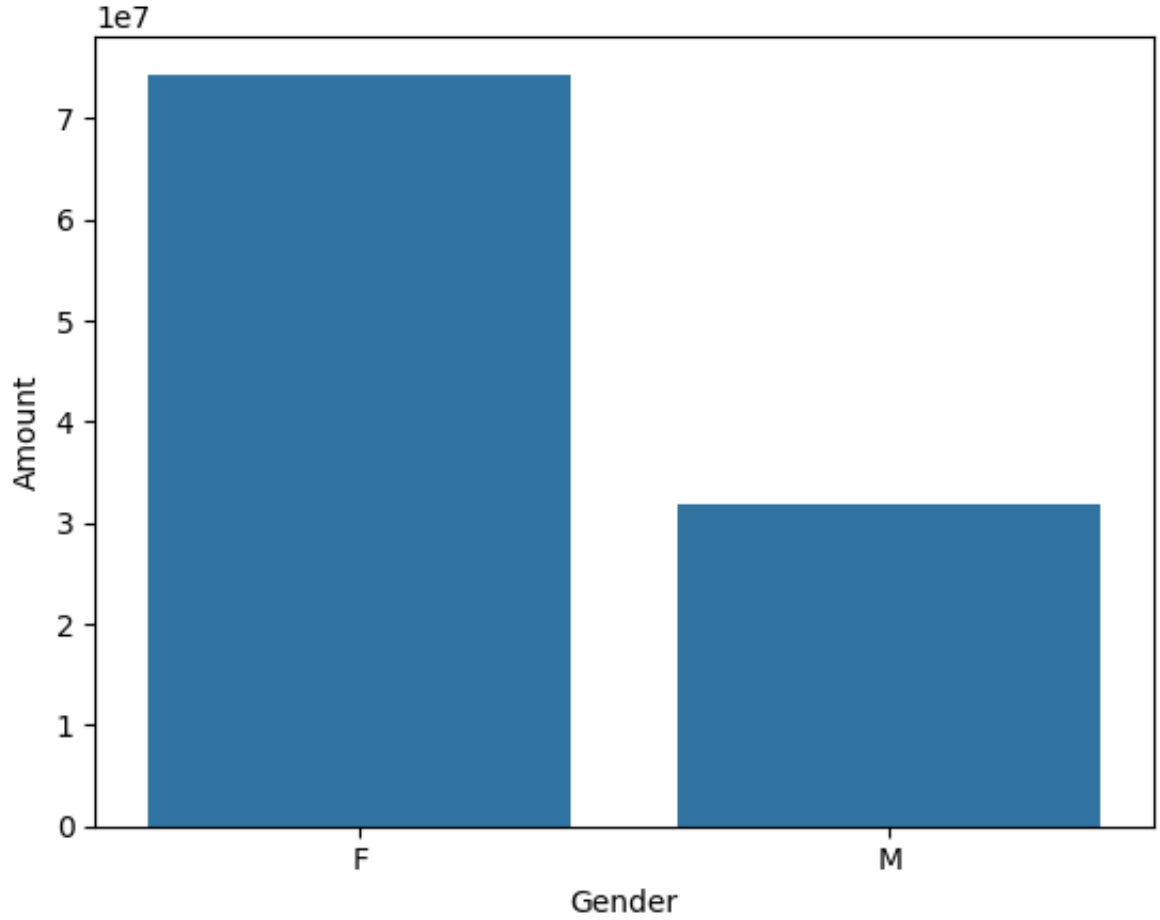
Gender

```
#analysing no. of males and demales in the data


# plotting a bar chart for Gender and it's count

ax = sns.countplot(x = 'Gender',data = df)
#labelling the bars in the data
for bars in ax.containers:
    ax.bar_label(bars)
```



```
# plotting a bar chart for gender vs total amount

sales_gen = df.groupby(['Gender'], as_index=False)['Amount'].sum().sort_values(by=

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

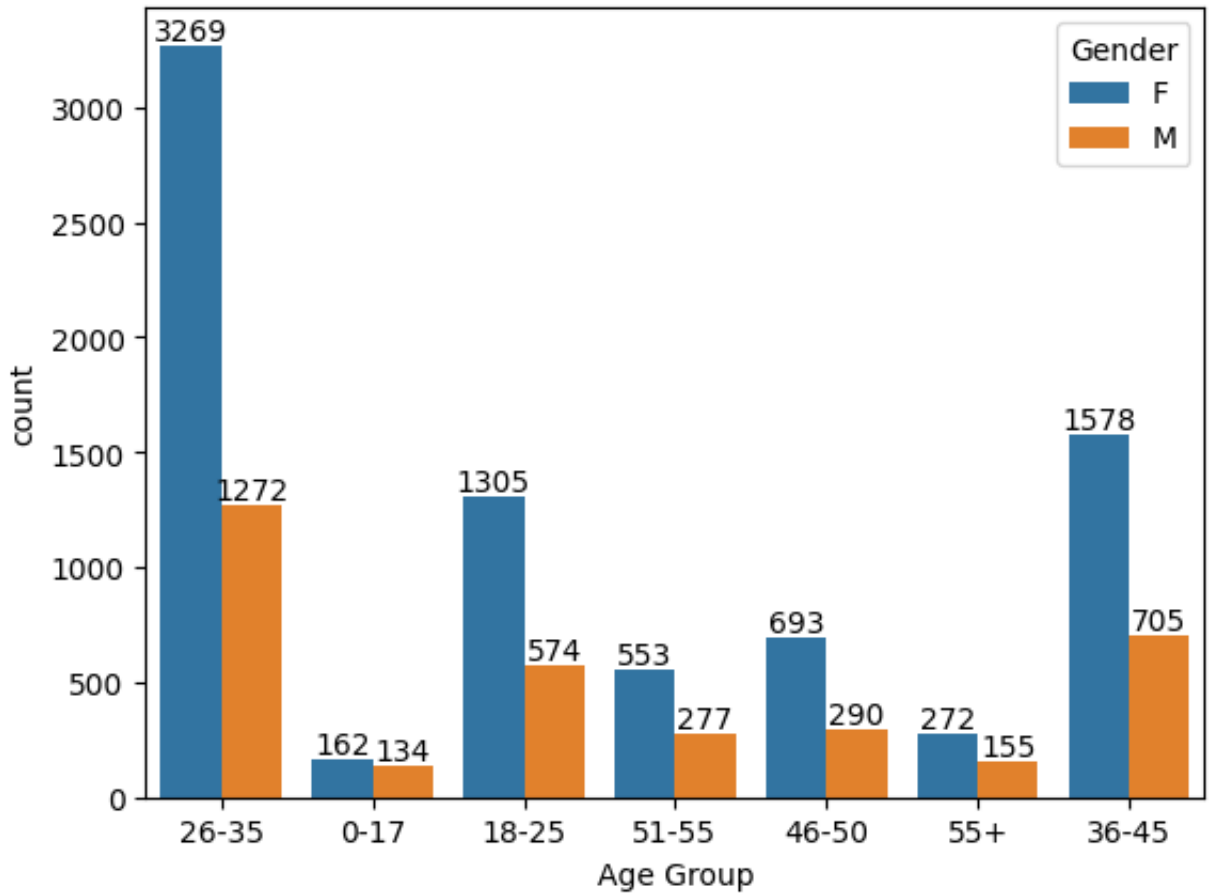
```
<Axes: xlabel='Gender', ylabel='Amount'>
```



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

Age

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

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

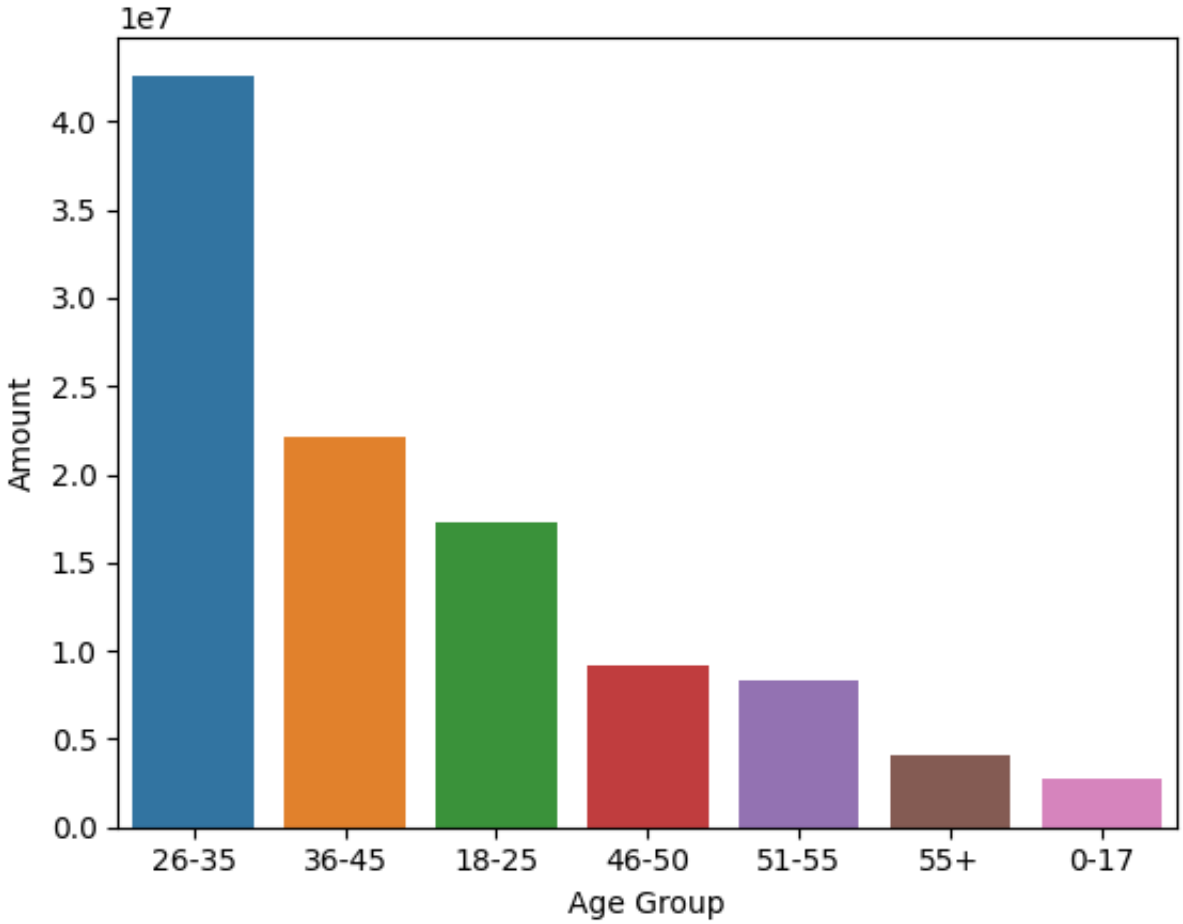


```
df.groupby(['Age Group'], as_index=False)['Amount'].sum().sort_values(by='Amount'
```

	Age Group	Amount
2	26-35	42613442
3	36-45	22144994
1	18-25	17240732
4	46-50	9207844
5	51-55	8261477
6	55+	4080987
0	0-17	2699653

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

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

```
<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

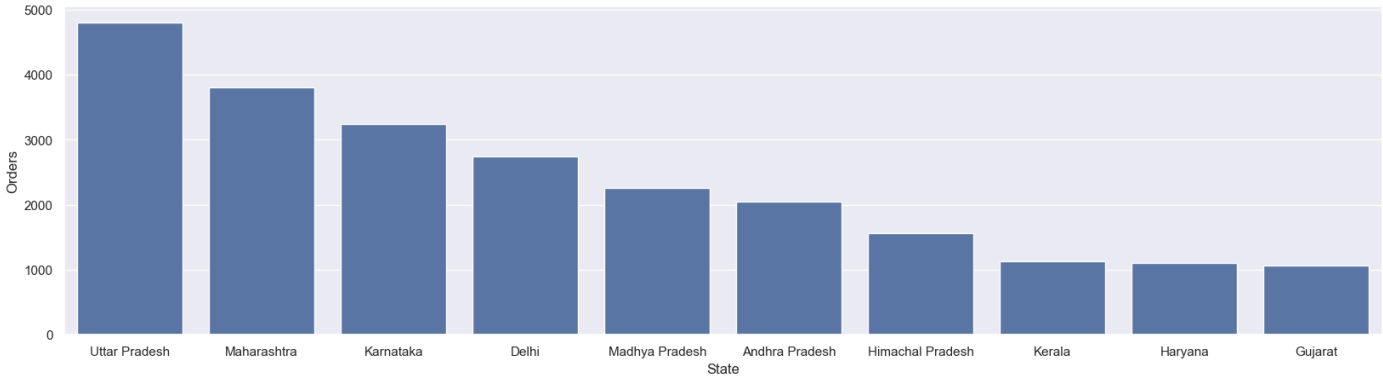
State

```
df.groupby(['State'], as_index=False)['Orders'].sum().sort_values(by='Orders', as
#evaluarting sales order from all states
```

	State	Orders
14	Uttar Pradesh	4807
10	Maharashtra	3810
7	Karnataka	3240
2	Delhi	2740
9	Madhya Pradesh	2252
0	Andhra Pradesh	2051
5	Himachal Pradesh	1568
8	Kerala	1137
4	Haryana	1109
3	Gujarat	1066
1	Bihar	1062
6	Jharkhand	953
15	Uttarakhand	824
12	Rajasthan	555
11	Punjab	495
13	Telangana	312

```
# total number of orders from top 10 states

sales_state = df.groupby(['State'], as_index=False)['Orders'].sum().sort_values(b

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

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

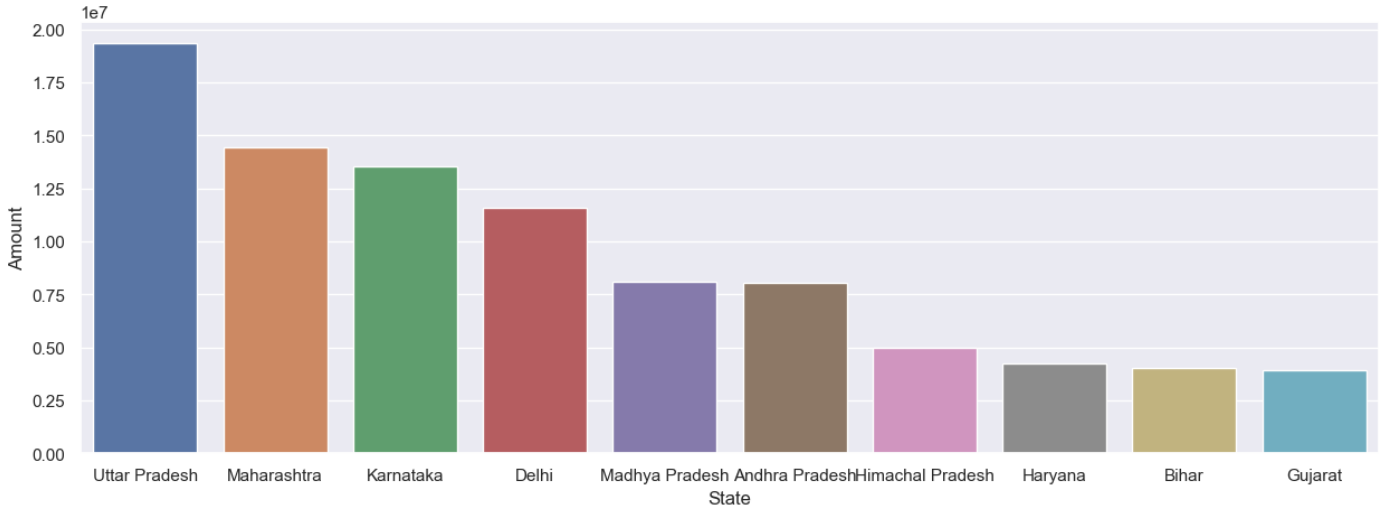


```
#total sales from all states
df.groupby(['State'], as_index=False)['Amount'].sum().sort_values(by='Amount', as
```

	State	Amount
14	Uttar Pradesh	19374968
10	Maharashtra	14427543
7	Karnataka	13523540
2	Delhi	11603818
9	Madhya Pradesh	8101142
0	Andhra Pradesh	8037146
5	Himachal Pradesh	4963368
4	Haryana	4220175
1	Bihar	4022757
3	Gujarat	3946082
8	Kerala	3894491
6	Jharkhand	3026456
15	Uttarakhand	2520944
12	Rajasthan	1909409
11	Punjab	1525800
13	Telangana	1151490

```
# total amount/sales from top 10 states

sales_state = df.groupby(['State'], as_index=False)['Amount'].sum().sort_values(b

sns.set(rc={'figure.figsize':(15,5)})
sns.barplot(data = sales_state, x = 'State',y= 'Amount')
```

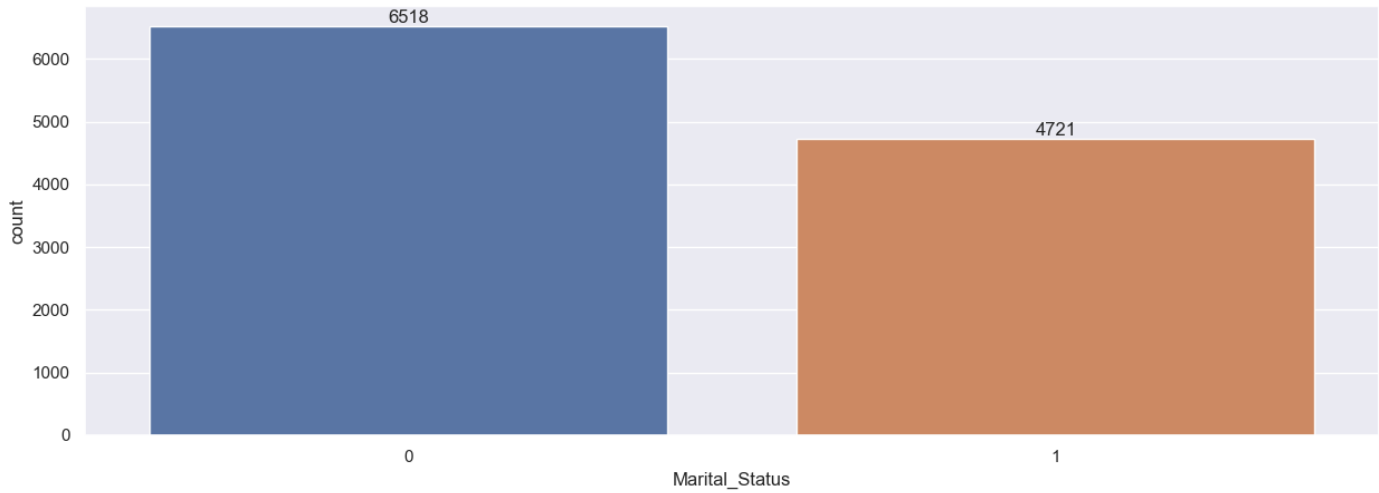
```
<Axes: xlabel='State', ylabel='Amount'>
```



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

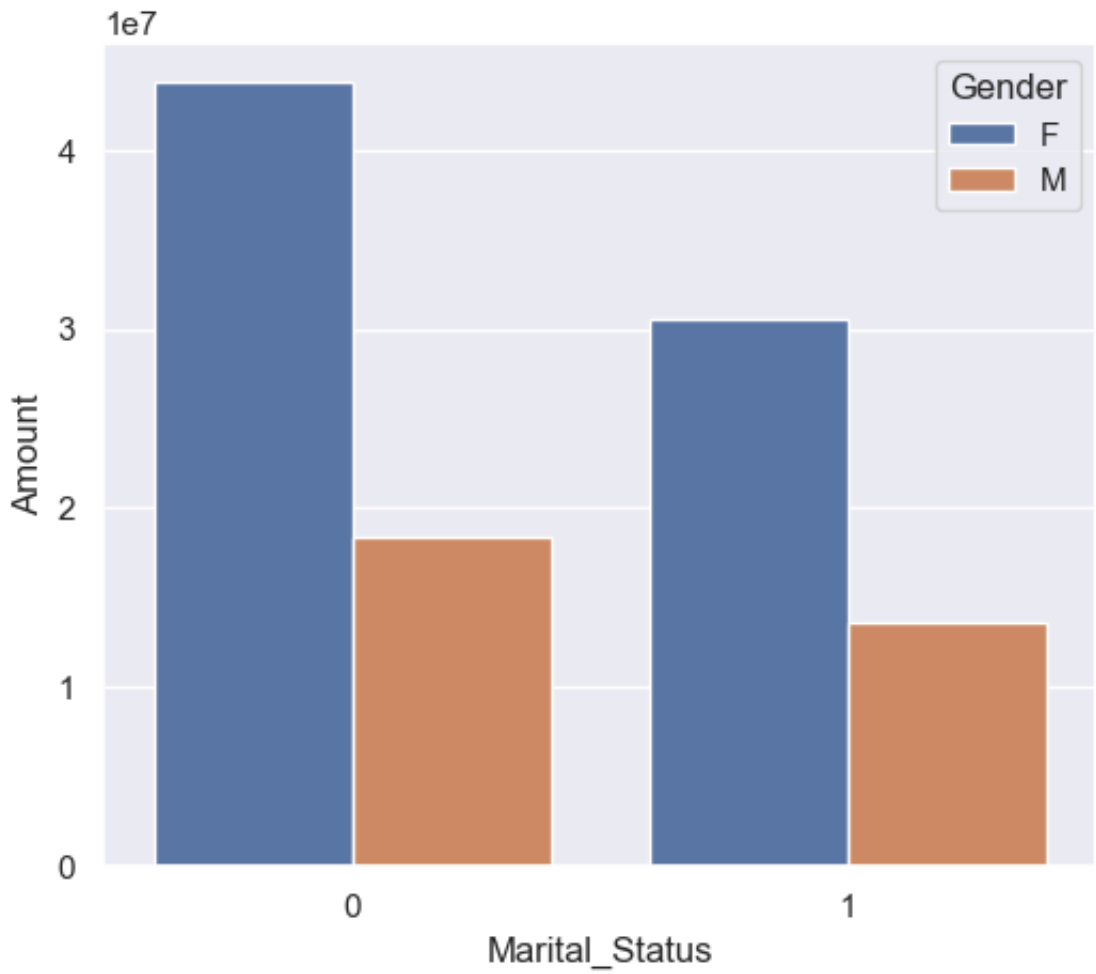
Marital Status

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

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



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

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

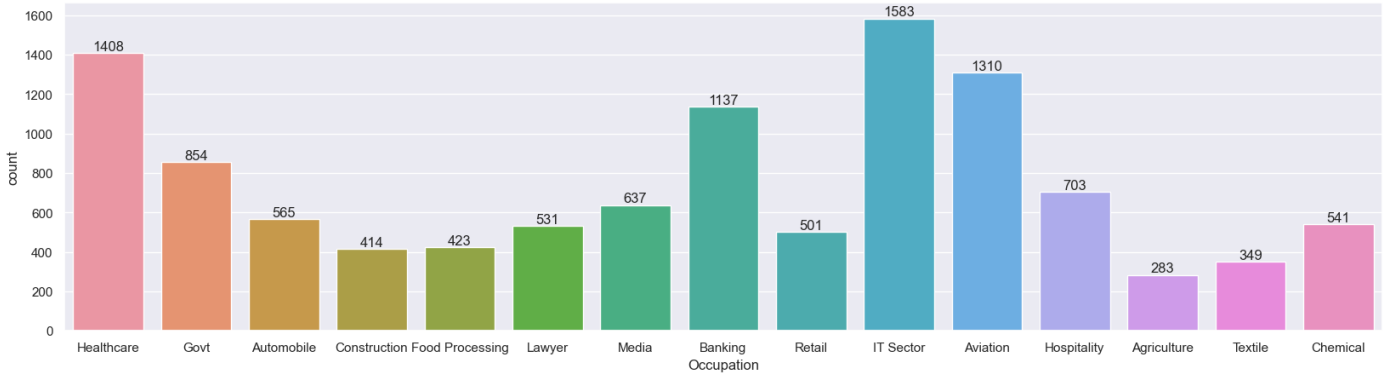
```
<Axes: xlabel='Marital_Status', ylabel='Amount'>
```



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

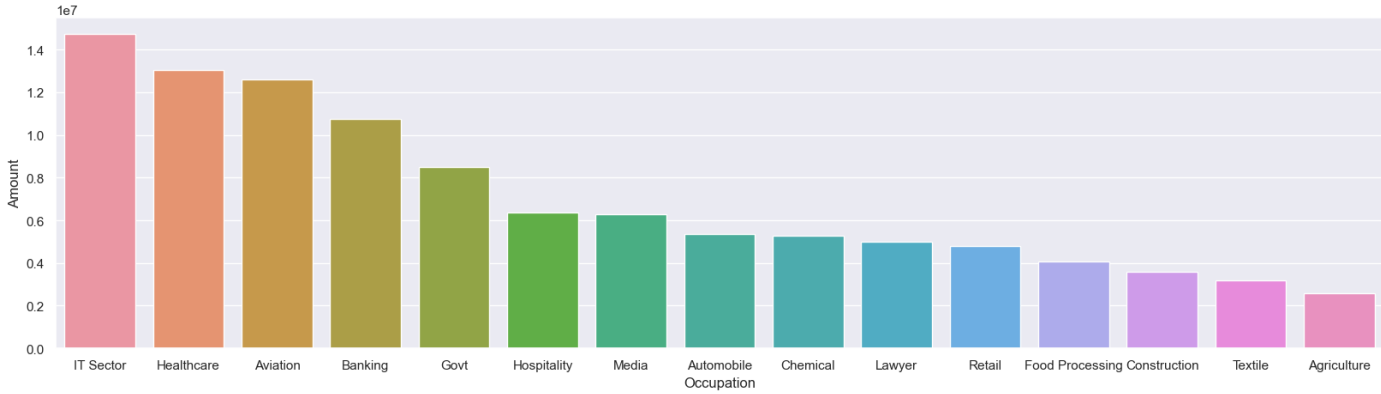
Occupation

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

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



```
sales_state = df.groupby(['Occupation'], as_index=False)['Amount'].sum().sort_val

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

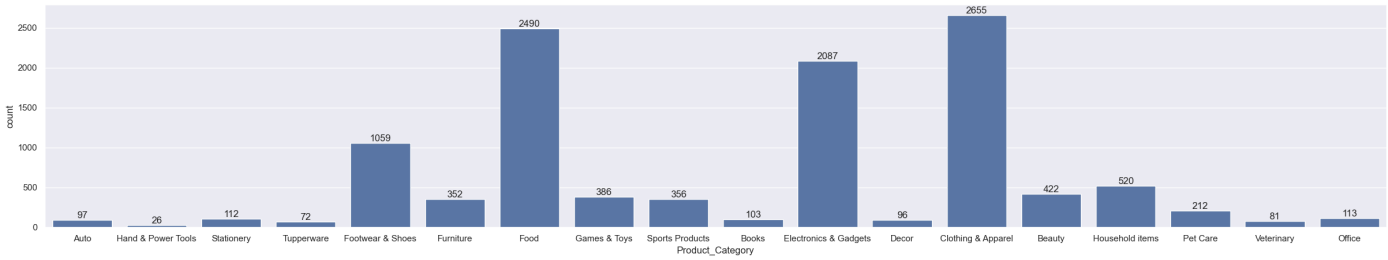
```
<Axes: xlabel='Occupation', ylabel='Amount'>
```



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

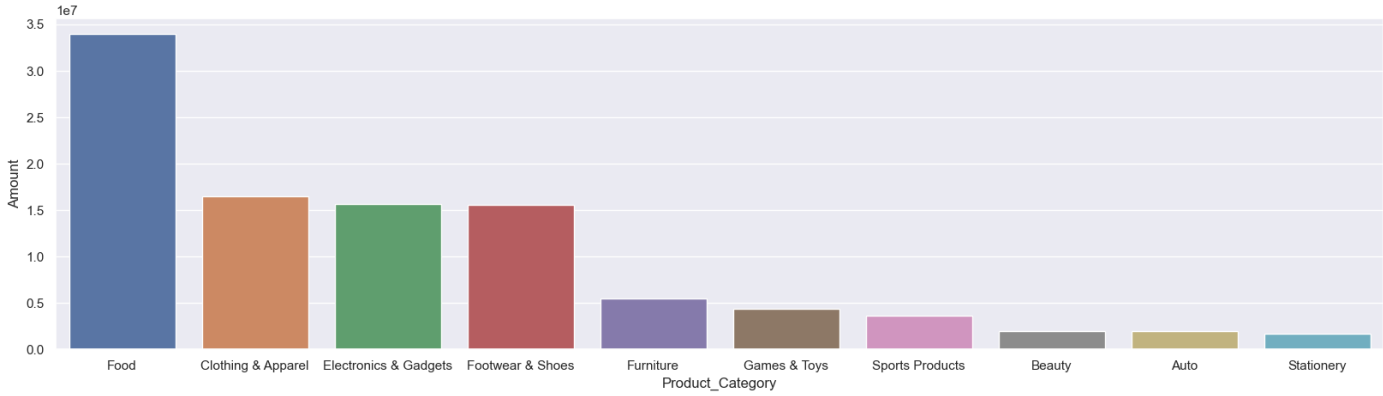
Product Category

```
sns.set(rc={'figure.figsize':(30,5)})
ax = sns.countplot(data = df, x = 'Product_Category')

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



```
sales_state = df.groupby(['Product_Category'], as_index=False)['Amount'].sum().so

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

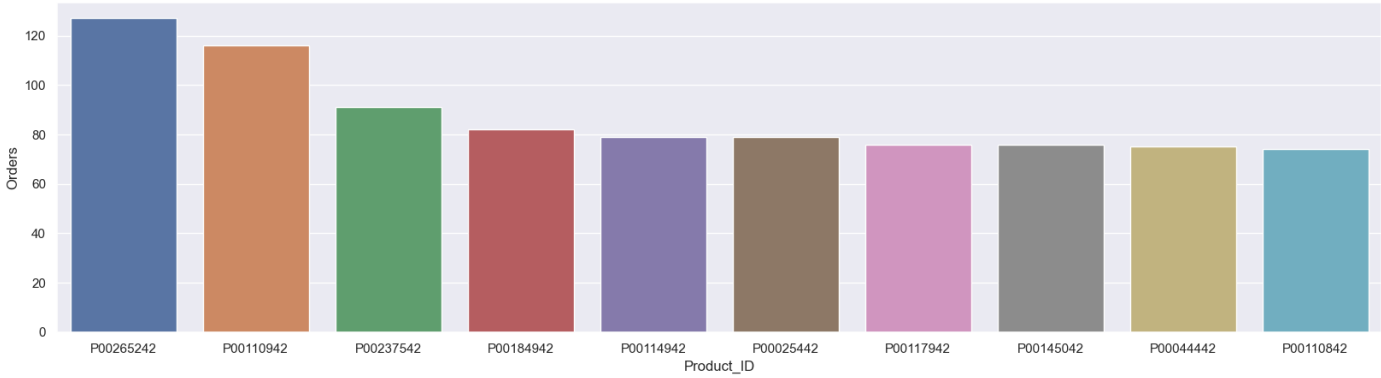
```
<Axes: xlabel='Product_Category', ylabel='Amount'>
```



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

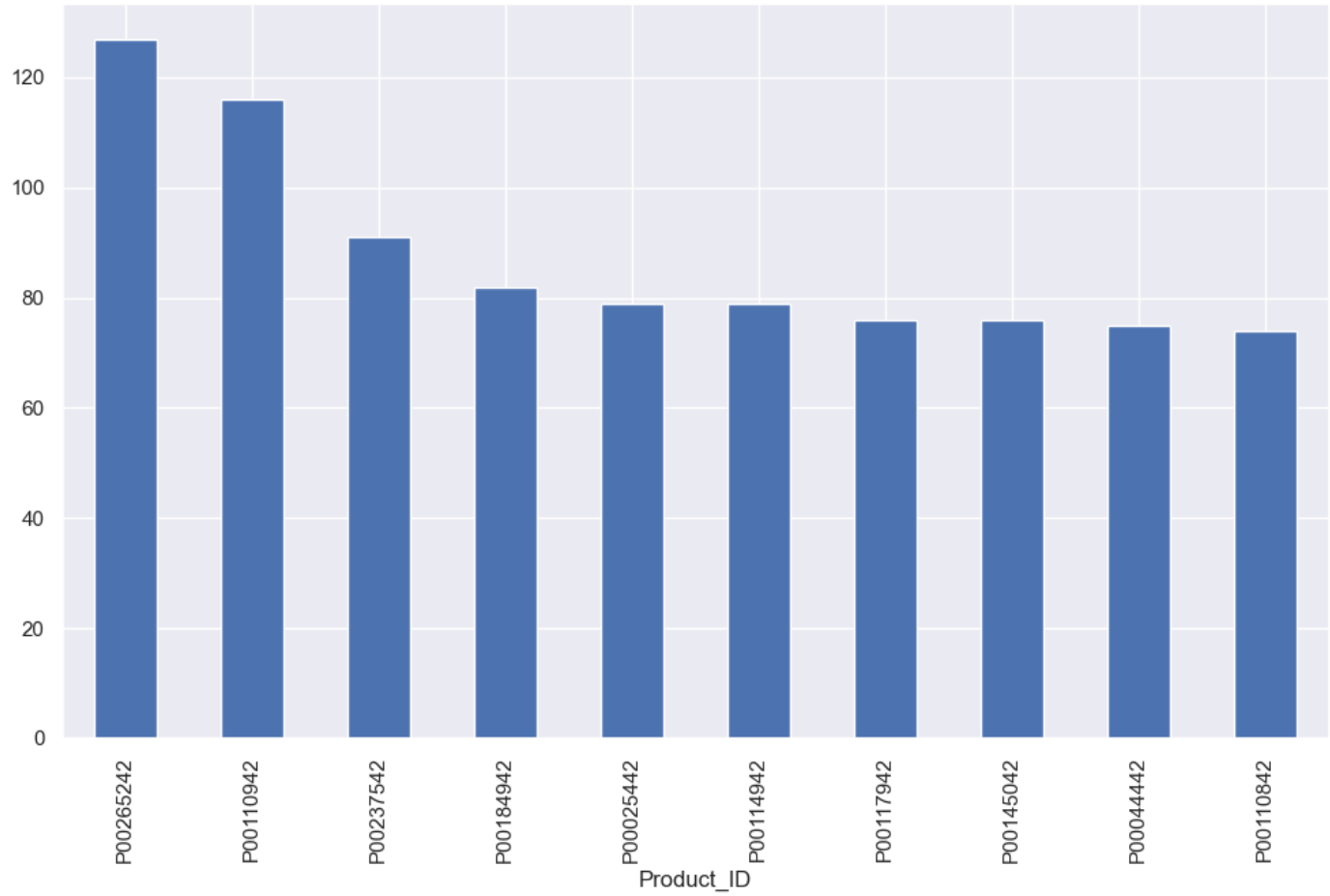
```
sales_state = df.groupby(['Product_ID'], as_index=False)['Orders'].sum().sort_val

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

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



```
# 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
```

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



Conclusion:

###

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

Thank you!