Exploratory Data Analysis of Diwali Sales | Python Project-2

Objective: Analysing Diwali sales data to improve customer experience and sales by making use of NumPy, Pandas, Matplotlib and Seaborn libraries

Step-1: Importing all the necessary libraries, uploading the csv file and checking the file contents

In [1]:
```
#Importing the libraries

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

In [2]:
```
#Reading the Diwali_Sales CSV file and assigning it to a DataFrame

df = pd.read_csv(r"D:\Project Portfolio\Python\Diwali Sales Analysis_Project-2\Diwali Sales Data.csv", encoding = 'unicode_escape')
```

In [3]:
```
#Checking the total number of rows and columns

df.shape
```

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

In [4]:
```
#Checking all the columns and first 5 rows

df.head()
```

Out[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

In [5]:
```
#Digging deeper to know more about the columns, data types etc

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
```

In [6]:
```
#Fetching all the column names

df.columns
```

Out[6]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount', 'Status', 'unnamed1'],
      dtype='object')
```

In [7]:
```
#Checking the rows*columns count

df.size
```

Out[7]:
```
168765
```

Step-2: Data cleaning & Exploring the data using Pandas and NumPy

In [8]:
```
#Dropping the empty columns

df.drop(['Status','unnamed1'], axis=1, inplace=True)
```

In [9]:
```
#Cross verifying the contents of the file

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
```

In [10]:
```
#Checking for the null values in the file

df.isnull()
```

Out[10]:

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

11251 rows × 13 columns

In [11]:
```
#Checking the total count of null values in all the columns

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

Out[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
```

In [12]:
```
df.shape
```

Out[12]:
```
(11251, 13)
```

In [13]:
```
#Dropping the null values

df.dropna(inplace=True)
```

In [14]:
```
#Cross verfying the total rows and columns

df.shape
```

Out[14]:
```
(11239, 13)
```

In [15]:
```
#Converting the datatype for our convenience

df['Amount'] = df['Amount'].astype(int)
```

In [16]:
```
#Cross verifying the conversion of datatype
df['Amount'].dtype
```

Out[16]:
```
dtype('int32')
```

In [18]:
```
#Checking the basic statistics of required columns

df[['Age','Orders','Amount']].describe()
```

Out[18]:

	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

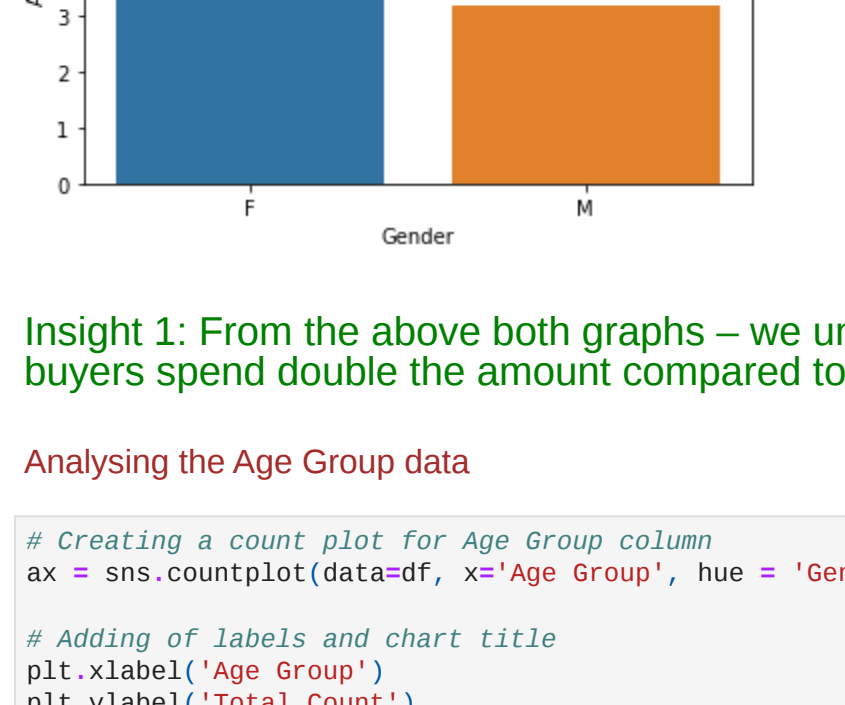
Step-3: Exploratory Data Analysis using Matplotlib and Seaborn

In [29]:
```
#Checking the column names for creating various charts
df.columns
```

Out[29]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

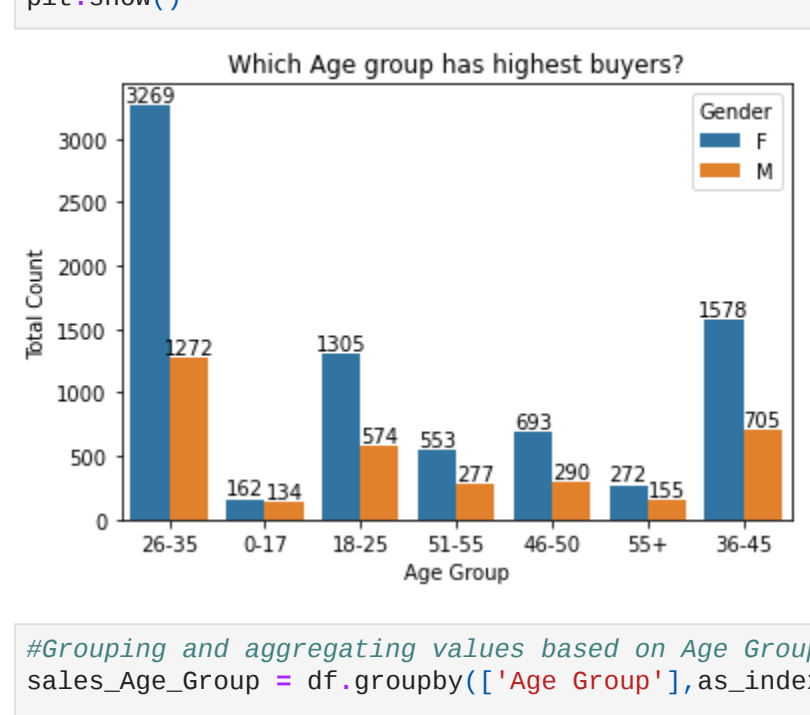
Analysing the Gender data

In [57]:
```
# Creating a count plot for Gender column
ax = sns.countplot(data=df, x='Gender')

# Adding of labels and chart title
plt.xlabel('Gender')
plt.ylabel('Total Count')
plt.title('Total buyers in each gender')

#Inserting total value above the bars
for bars in ax.containers:
    ax.bar_label(bars)

#Displaying the plot
plt.show()
```



In [21]:
```
#Grouping and aggregating values based on Gender
sales_gender_wise = df.groupby(['Gender'],as_index = False) ['Amount'].sum().sort_values(by='Amount',ascending=False)

print(sales_gender_wise)
sns.barplot(x='Gender', y='Amount', data = sales_gender_wise)

#Displaying the plot
plt.show()
```

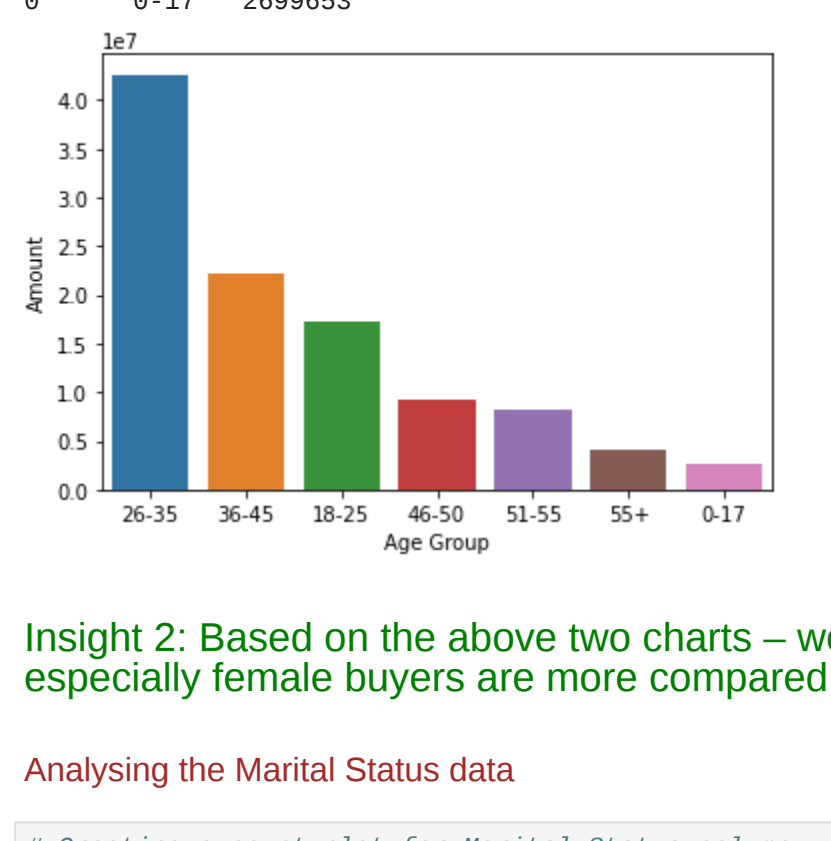
```
  Gender    Amount
0      F  74335853
1      M  31913276
```



Insight 1: From the above both graphs – we understand that females buyers are double the male buyers. And also females buyers spend double the amount compared to male buyers

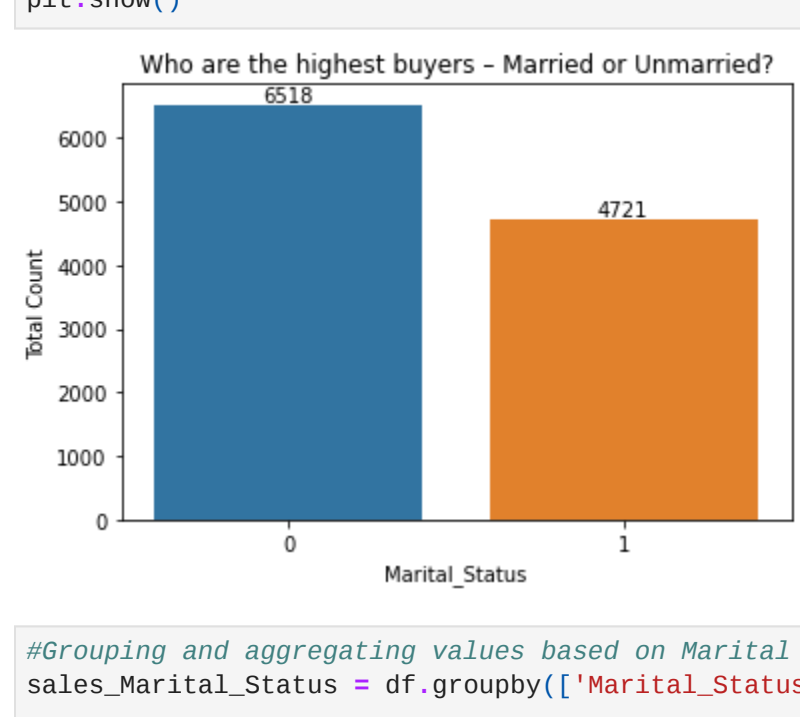
Analysing the Age Group data

In [26]:
```
# Creating a count plot for Age Group column
ax = sns.countplot(data=df, x='Age Group', hue = 'Gender')

# Adding of labels and chart title
plt.xlabel('Age Group')
plt.ylabel('Total Count')
plt.title('Which Age group has highest buyers?')

#Inserting total value above the bars
for bars in ax.containers:
    ax.bar_label(bars)

#Displaying the plot
plt.show()
```



In [23]:
```
#Grouping and aggregating values based on Age Group
sales_Age_Group = df.groupby(['Age Group'],as_index = False) ['Amount'].sum().sort_values(by='Amount',ascending=False)

print(sales_Age_Group)
sns.barplot(x='Age Group', y='Amount', data = sales_Age_Group)

#Displaying the plot
plt.show()
```

```
  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
```



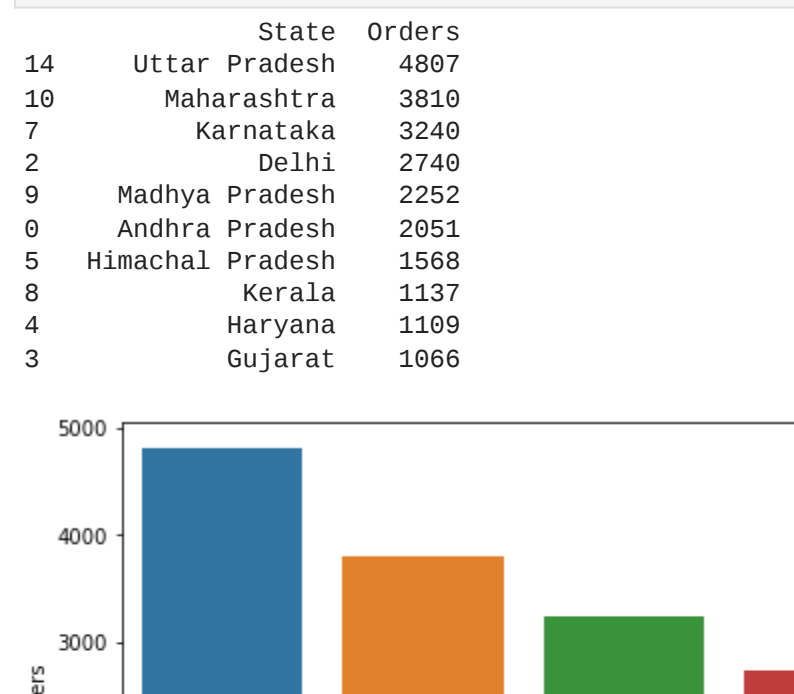
Insight 2: Based on the above two charts – we understand that the 'Age group 26-35' contains highest number of buyers and especially female buyers are more compared to male buyers

Analysing the Marital Status data

In [56]:
```
# Creating a count plot for Marital_Status column
ax = sns.countplot(data=df, x='Marital_Status')

# Adding of labels and chart title
plt.xlabel('Marital_Status')
plt.ylabel('Total Count')
plt.title('Who are the highest buyers - Married or Unmarried?')

#Inserting total value above the bars
for bars in ax.containers:
    ax.bar_label(bars)

#Displaying the plot
plt.show()
```

In [55]:
```
#Grouping and aggregating values based on Marital Status data
sales_Marital_Status = df.groupby(['Marital_Status','Gender'],as_index = False) ['Amount'].sum().sort_values(by='Amount',ascending=False)

print(sales_Marital_Status)
sns.barplot(x='Marital_Status', y='Amount', data = sales_Marital_Status, hue = 'Gender')
plt.title('Who are the highest buyers - Married or Unmarried?')

#Displaying the plot
plt.show()
```

```
   Marital_Status Gender    Amount
0               0      F  43786646
2               1      F  30549207
1               0      M  18338738
3               1      M  13574538
```

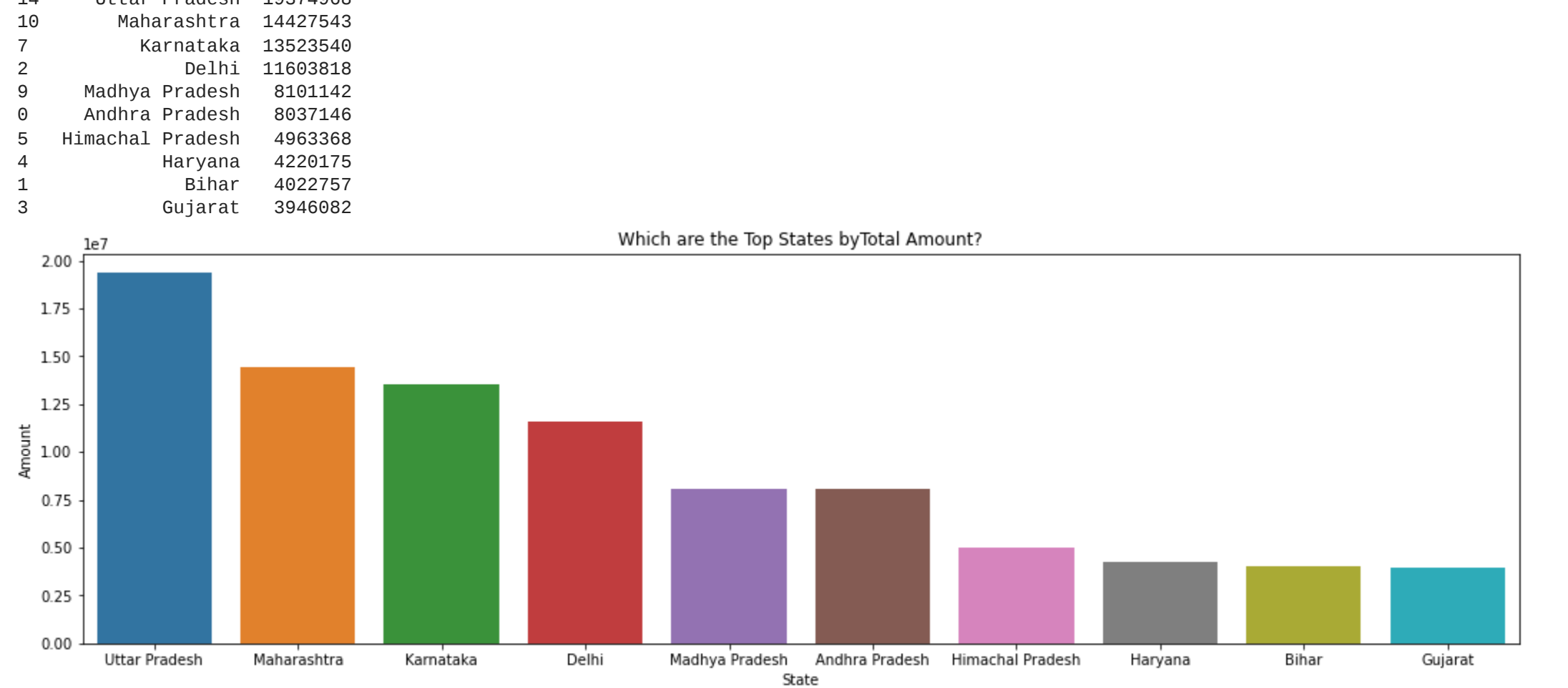


Insight 3: From the above graphs – we understand that the highest buyers are married people(especially women)

Analysing the State data

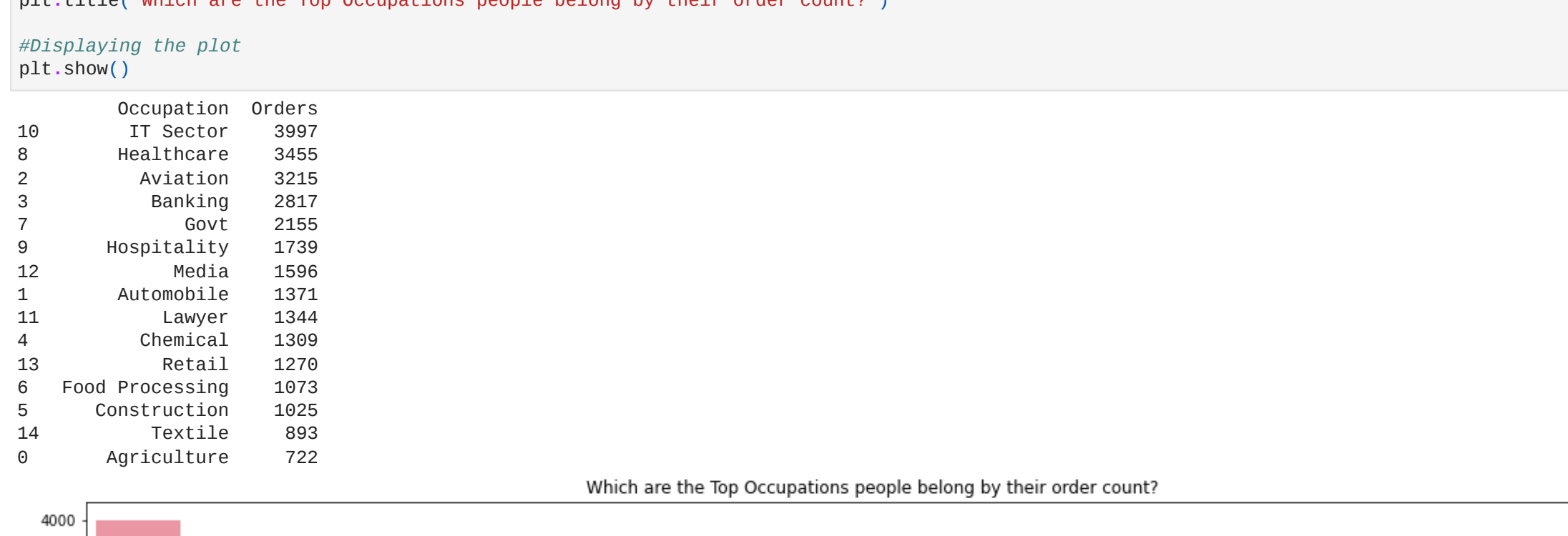
In [54]:
```
#Checking top 10 states based on order count

#Grouping and aggregating values
sales_State = df.groupby(['State'],as_index = False) ['Orders'].sum().sort_values(by='Orders',ascending=False).head(10)

#Printing of order values by state-wise for reference
print(sales_State)

#Plotting of graph
plt.figure(figsize=(18, 5))
sns.barplot(x='State', y='Orders', data = sales_State)
plt.title('Which are the Top States by order count?')

#Displaying the plot
plt.show()
```

```
               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
```



In [53]:
```
#Checking top 10 states based on amount

#Grouping and aggregating values
sales_State = df.groupby(['State'],as_index = False) ['Amount'].sum().sort_values(by='Amount',ascending=False).head(10)

#Printing of order values by state-wise for reference
print(sales_State)

#Plotting of graph
plt.figure(figsize=(18, 5))
sns.barplot(x='State', y='Amount', data = sales_State)
plt.title('Which are the Top States byTotal Amount?')

#Displaying the plot
plt.show()
```

```
               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
```

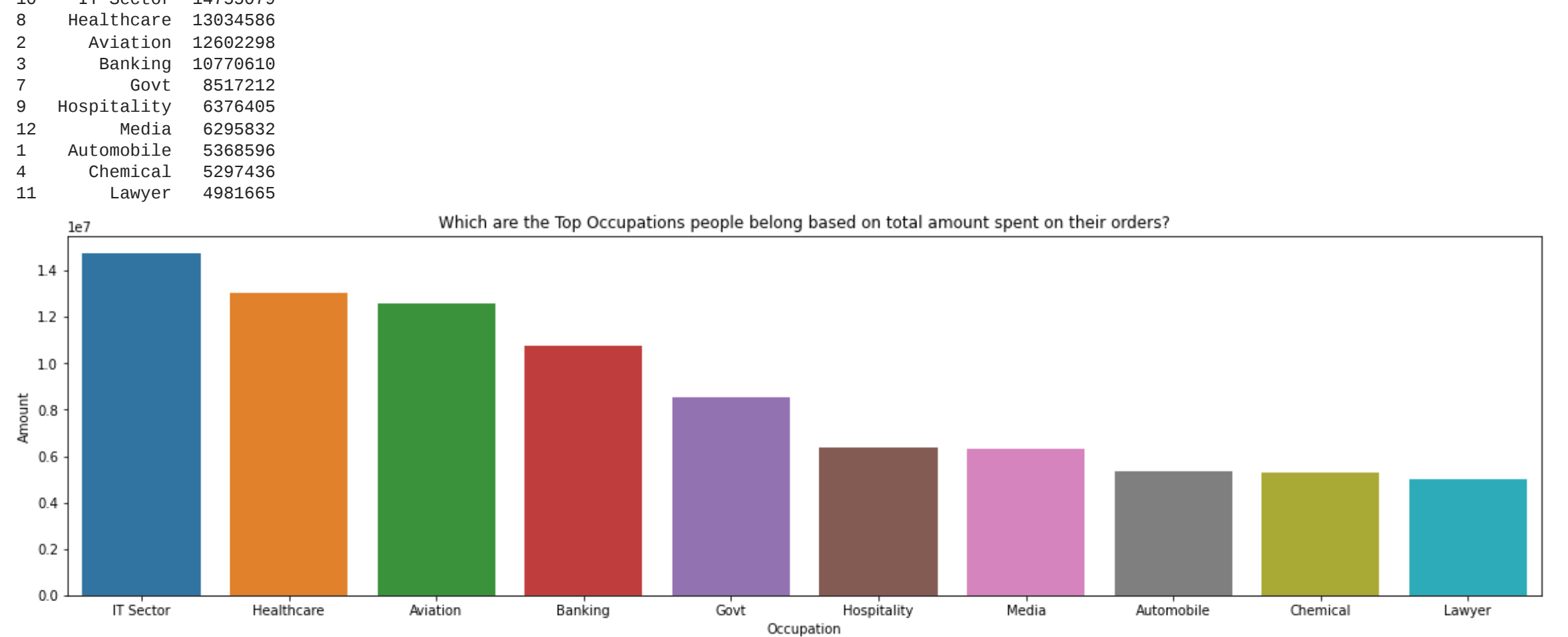


Insight 3: From the above graphs – we understand that Uttar Pradesh, Maharastra and Karnataka states have highest order count and amount

Analysing the Occupation data

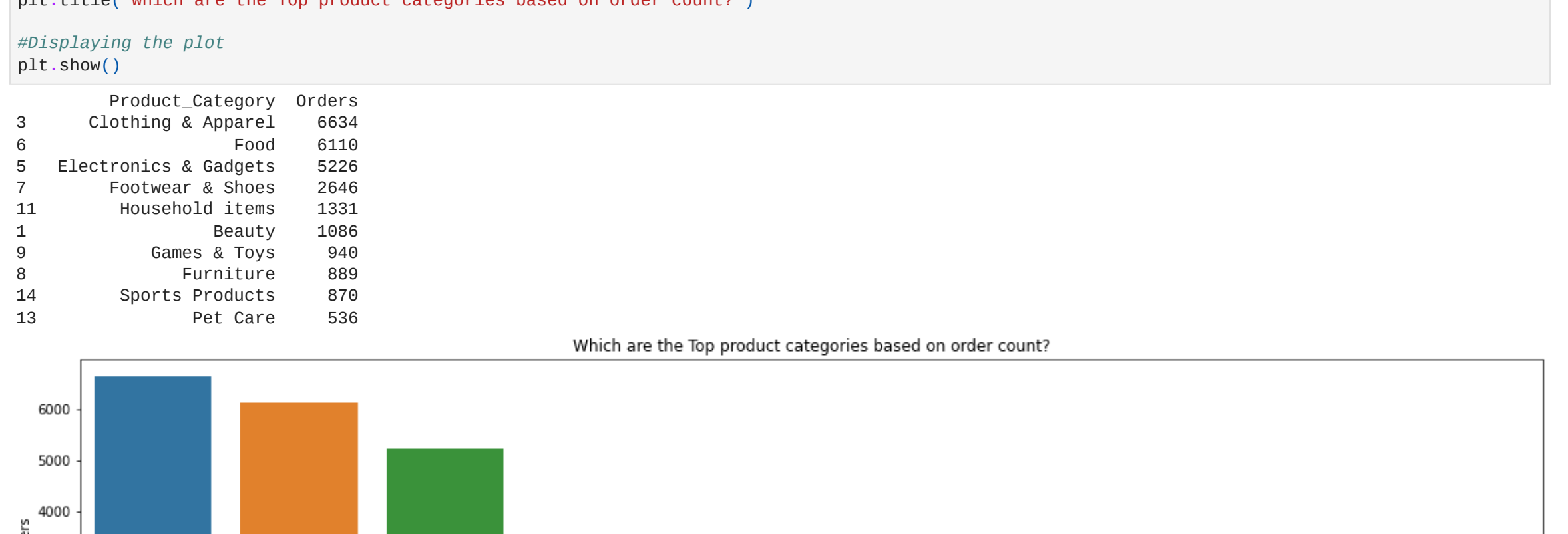
In [65]:
```
#Checking top occupations people belong based on their order count

#Grouping and aggregating values
sales_Occupation = df.groupby(['Occupation'],as_index = False) ['Orders'].sum().sort_values(by='Orders',ascending=False)

#Printing of order values by state-wise for reference
print(sales_Occupation)

#Plotting of graph
plt.figure(figsize=(20, 5))
sns.barplot(x='Occupation', y='Orders', data = sales_Occupation)
plt.title('Which are the Top Occupations people belong by their order count?')

#Displaying the plot
plt.show()
```

```
         Occupation  Orders
10        IT Sector    3997
8        Healthcare    3455
2          Aviation    3215
3           Banking    2817
7              Govt    2155
9       Hospitality    1739
12            Media    1596
1        Automobile    1371
11           Lawyer    1344
4          Chemical    1309
13           Retail    1270
6   Food Processing    1073
5      Construction    1025
14          Textile     893
0       Agriculture     722
```



In [66]:
```
#Checking top 10 occupations people belong based on total amount spent on their orders

#Grouping and aggregating values
sales_Occupation = df.groupby(['Occupation'],as_index = False) ['Amount'].sum().sort_values(by='Amount',ascending=False).head(10)

#Printing of order values by state-wise for reference
print(sales_Occupation)

#Plotting of graph
plt.figure(figsize=(20, 5))
sns.barplot(x='Occupation', y='Amount', data = sales_Occupation)
plt.title('Which are the Top Occupations people belong based on total amount spent on their orders?')

#Displaying the plot
plt.show()
```

```
     Occupation    Amount
10    IT Sector  14755079
8    Healthcare  13034586
2      Aviation  12602298
3       Banking  10770610
7          Govt   8517212
9   Hospitality   6376405
12        Media   6295832
1    Automobile   5368596
4      Chemical   5297436
11       Lawyer   4981665
```

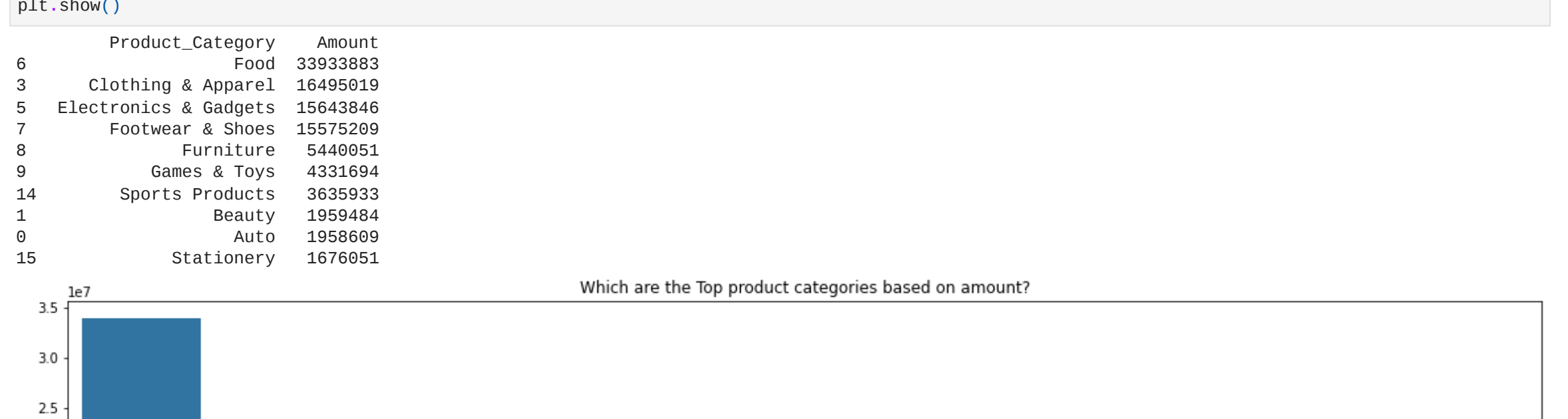


Insight 3: From the above graphs – we understand that most of the buyers belong to IT sector, Healthcare and Aviation sectors

Analysing the Product Category data

In [70]:
```
#Checking top 10 product categories based on order count

#Grouping and aggregating values
sales_Product_Category = df.groupby(['Product_Category'],as_index = False) ['Orders'].sum().sort_values(by='Orders',ascending=False).head(10)

#Printing of order values by state-wise for reference
print(sales_Product_Category)

#Plotting of graph
plt.figure(figsize=(20, 5))
sns.barplot(x='Product_Category', y='Orders', data = sales_Product_Category)
plt.title('Which are the Top product categories based on order count?')

#Displaying the plot
plt.show()
```

```
         Product_Category  Orders
3       Clothing & Apparel    6634
6                     Food    6110
5    Electronics & Gadgets    5226
7         Footwear & Shoes    2646
11         Household items    1331
1                   Beauty    1086
9             Games & Toys     940
8                Furniture     889
14         Sports Products     870
13                Pet Care     536
```

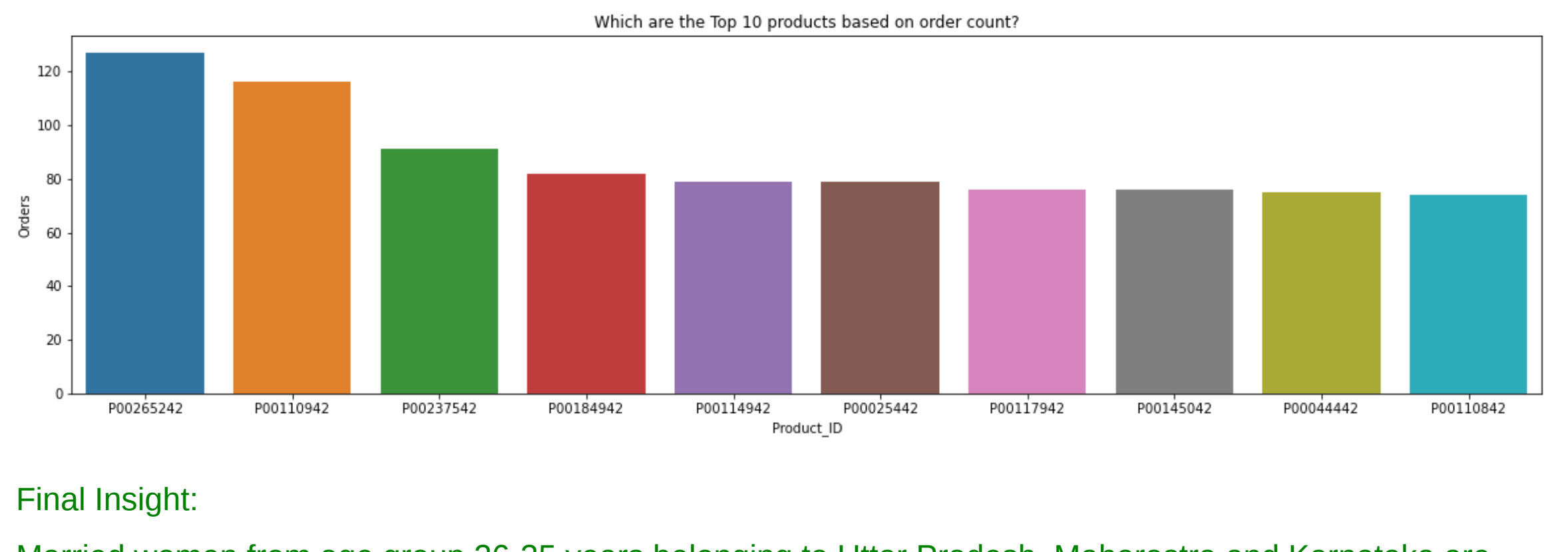


In [71]:
```
#Checking top 10 product categories based on amount

#Grouping and aggregating values
sales_Product_Category = df.groupby(['Product_Category'],as_index = False) ['Amount'].sum().sort_values(by='Amount',ascending=False).head(10)

#Printing of order values by state-wise for reference
print(sales_Product_Category)

#Plotting of graph
plt.figure(figsize=(20, 5))
sns.barplot(x='Product_Category', y='Amount', data = sales_Product_Category)
plt.title('Which are the Top product categories based on amount?')

#Displaying the plot
plt.show()
```

```
         Product_Category    Amount
6                    Food  33933883
3      Clothing & Apparel  16495019
5   Electronics & Gadgets  15643846
7        Footwear & Shoes  15575209
8               Furniture   5440051
9            Games & Toys   4331694
14        Sports Products   3635933
1                  Beauty   1959484
0                    Auto   1958609
15             Stationery   1676051
```

In [72]:
```
#Checking top 10 products based on order count

#Grouping and aggregating values
sales_Product = df.groupby(['Product_ID'],as_index = False) ['Orders'].sum().sort_values(by='Orders',ascending=False).head(10)

#Printing of order values by state-wise for reference
print(sales_Product)

#Plotting of graph
plt.figure(figsize=(20, 5))
sns.barplot(x='Product_ID', y='Orders', data = sales_Product)
plt.title('Which are the Top 10 products based on order count?')

#Displaying the plot
plt.show()
```

```
     Product_ID  Orders
1679  P00265242     127
644   P00110942     116
1504  P00237542      91
1146  P00184942      82
679   P00114942      79
171   P00025442      79
708   P00117942      76
888   P00145042      76
298   P00044442      75
643   P00110842      74
```



Final Insight:

Married women from age group 26-35 years belonging to Uttar Pradesh, Maharastra and Karnataka are the highest buyers. Also these buyers belong to IT sector, Healthcare and Aviation sectors

In [ ]: