Sales Data Analysis Using Python

Python Data Analyst Project

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[2]:
```
# import python libraries

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

[3]:
```
# import csv file
df = pd.read_csv('/content/drive/MyDrive/Data Analysis/Python Project/Sales␣
↪Data.csv', encoding= 'unicode_escape')
```

[5]:
```
df.shape
```

[5]:
```
(11251, 15)
```

[6]:
```
df.head()
```

[6]:
```
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

            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
    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

[8]:
```
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
```

[9]:
```
#drop unrelated/blank columns
df.drop(['Status', 'unnamed1'], axis=1, inplace=True)
```

[10]:
```
#check for null values
pd.isnull(df).sum()
```

[10]:
```
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
```

[11]:
```
# drop null values
df.dropna(inplace=True)
```

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

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

[13]:
```
dtype('int64')
```

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

[15]:
```
      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 33 Karnataka Southern
Construction Auto                                                  1
4 Gujarat Western Food Processing Auto                             1
11246 Maharashtra Western Chemical Office11247 Haryana Northern
Healthcare Veterinary 3                          Error! Bookmark not defined.

…                 …        …            …            …      …
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]
```

[16]:
```
# describe() method returns description of the data in the DataFrame (i.e.
↪count, mean, std, etc)
df.describe()
```

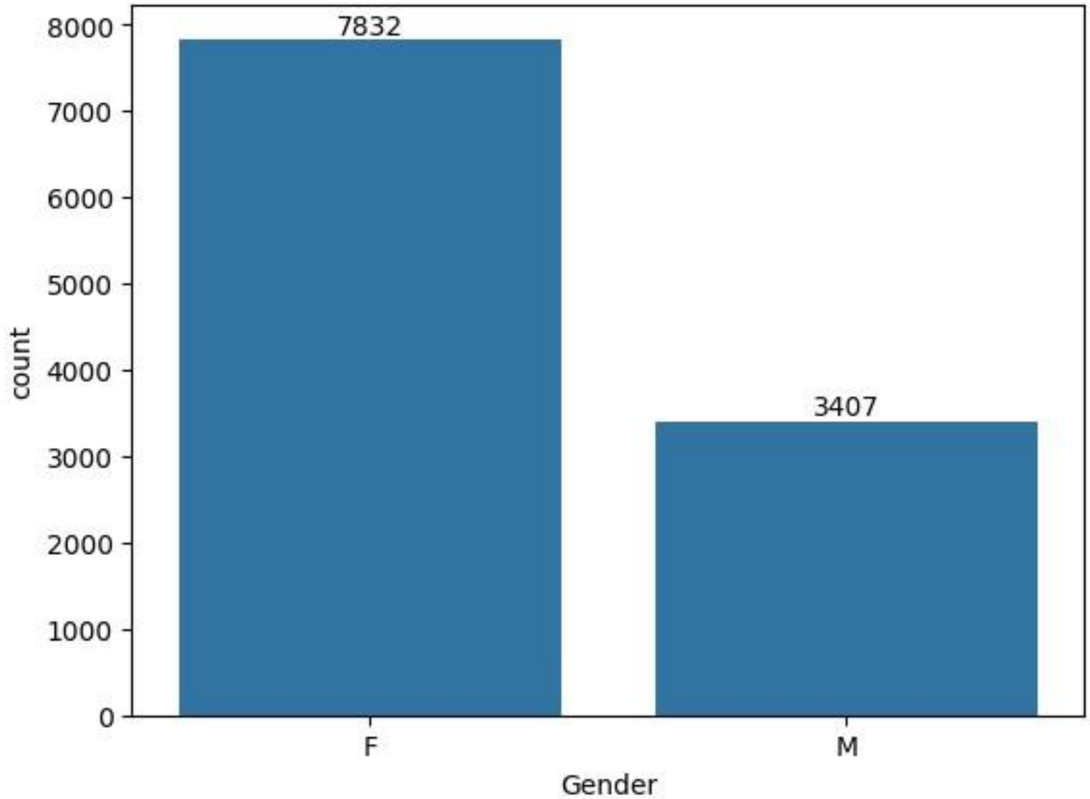
[16]:
```
            User_ID         Age Marital_Status        Orders        Amount
```

```
count 1.123900e+04 11239.000000 11239.000000          11239.000000
                                11239.000000
mean   1.003004e+06   35.410357      0.420055    2.4896349453.610553
std    1.716039e+03   12.753866      0.493589    1.1149675222.355168
min    1.000001e+06   12.000000      0.000000    1.000000 188.000000
25%    1.001492e+06   27.000000      0.000000    2.0000005443.000000
50%    1.003064e+06   33.000000      0.000000    2.0000008109.000000
75%    1.004426e+06   43.000000      1.000000    3.000000
                                                 12675.000000
max    1.006040e+06   92.000000      1.000000    4.000000
                                                 23952.000000
```

[17]:
```python
# use describe() for specific columns
df[['Age', 'Orders',
'Amount']].describe()
```

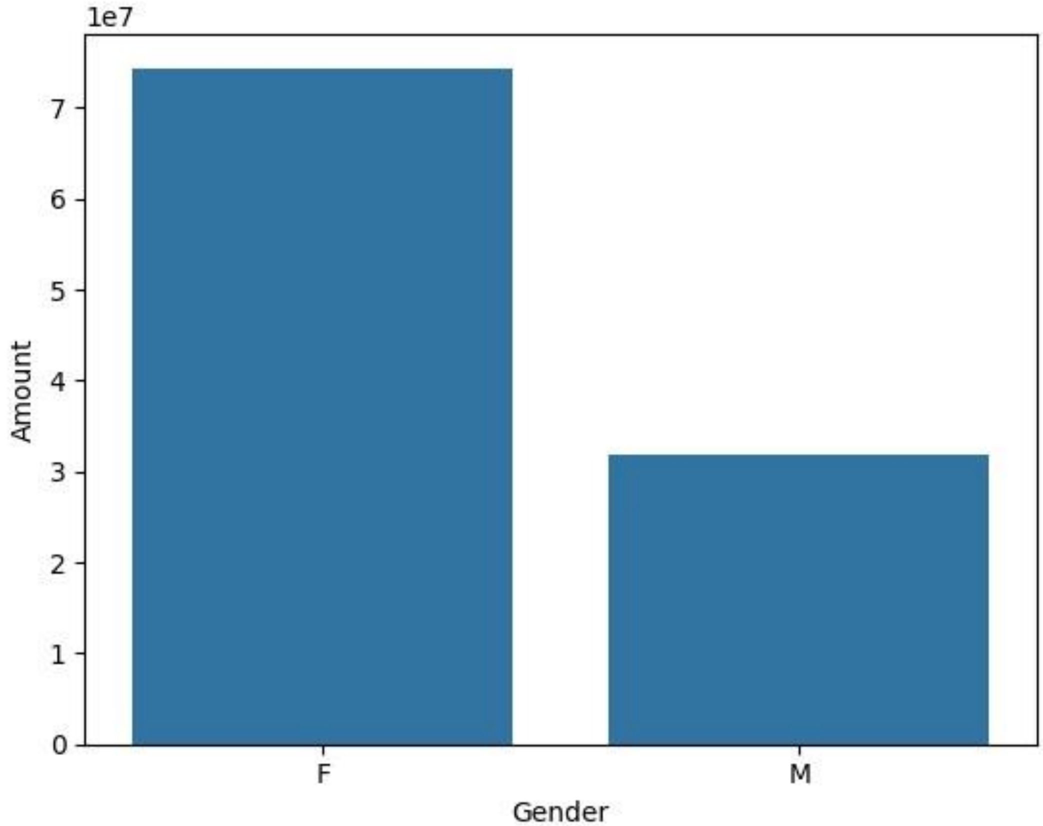
[17]:
```
Age   Orders     Amount count 11239.000000
11239.000000 11239.000000 mean   35.410357
2.489634 9453.610553 std   12.753866
1.114967 5222.355168 min   12.000000
1.000000   188.000000 25%   27.000000
2.000000 5443.000000
50%   33.000000  2.000000 8109.000000 75%
43.000000  3.000000 12675.000000 max
92.000000  4.000000 23952.000000
```
#Exploratory Data Analysis

[22]:
```python
# plotting a bar chart for Gender and it's count

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

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

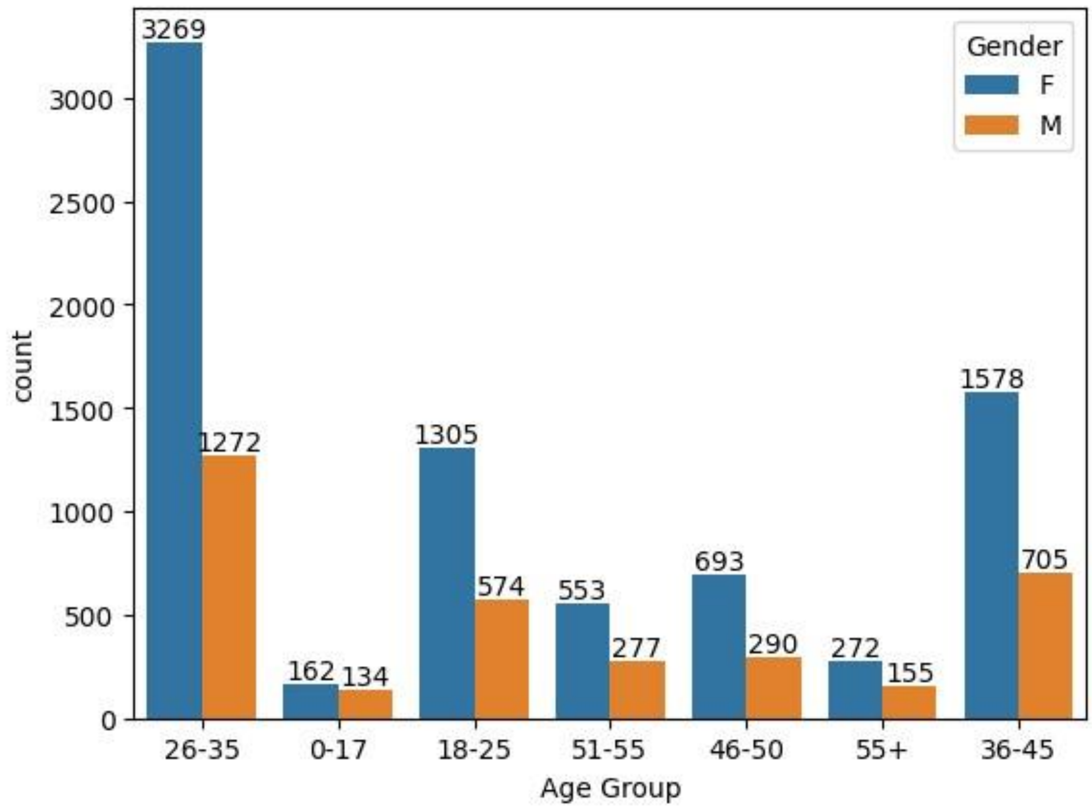


[19]:
```python
# plotting a bar chart for gender vs total amount

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)
```

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

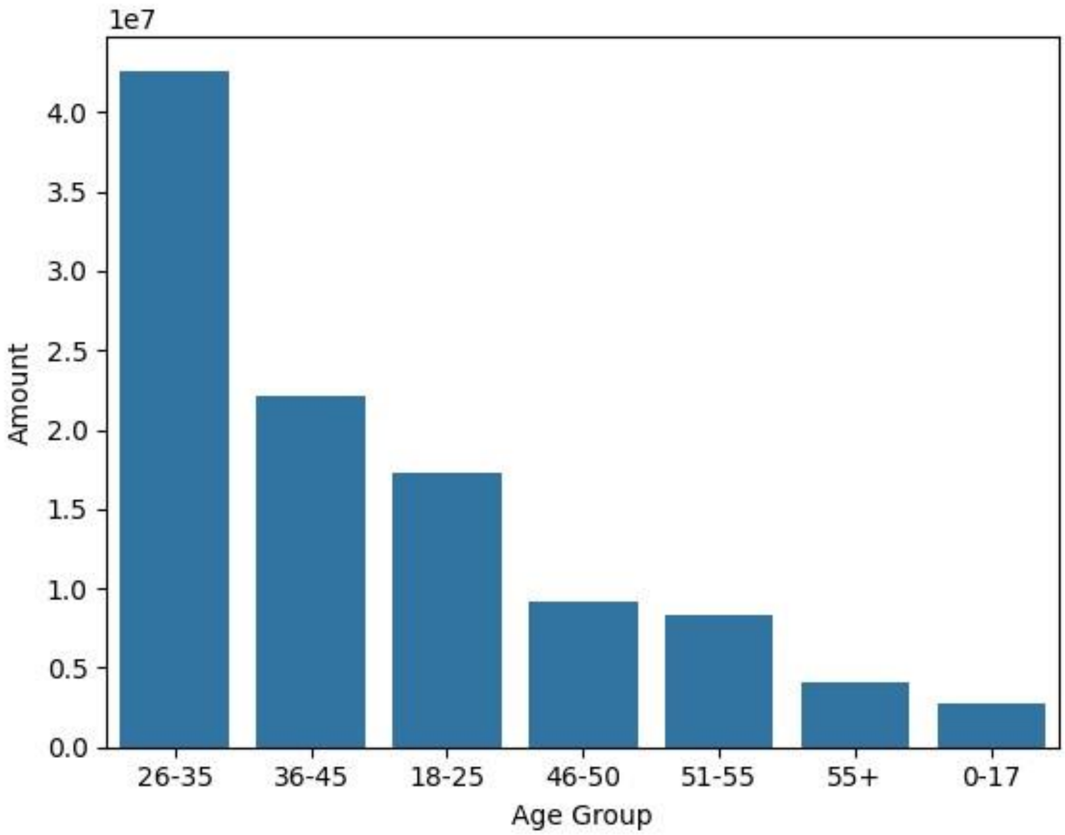


[20]:
```
ax = sns.countplot(data = df, x = 'Age Group', hue = 'Gender')

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



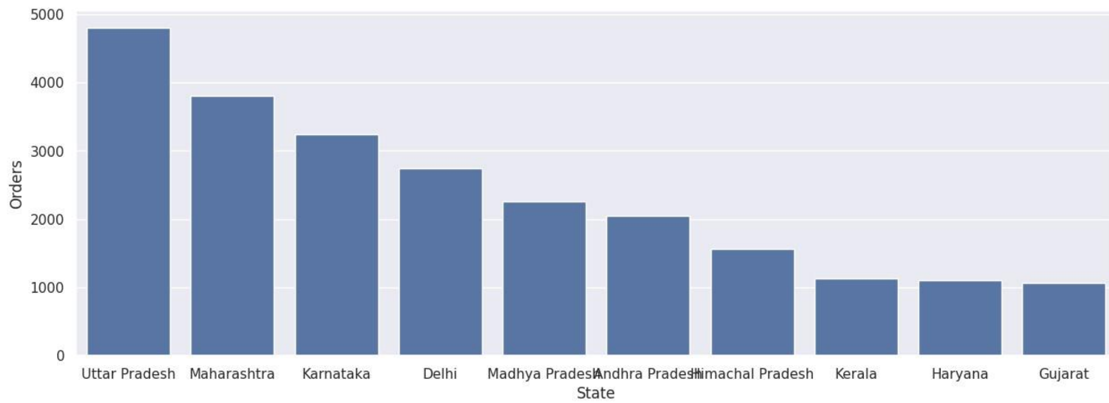

```
[21]: # Total Amount vs Age Group sales_age = df.groupby(['Age
      Group'], as_index=False)['Amount'].sum().
      ↪sort_values(by='Amount', ascending=False) sns.barplot(x =
      'Age Group',y= 'Amount' ,data = sales_age)
```

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



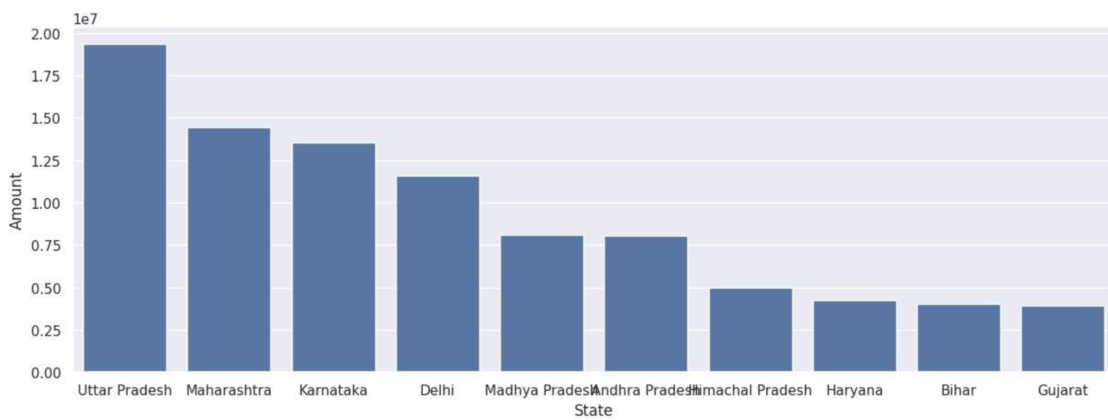
[23]:
```
# total number of orders from top 10 states

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

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

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

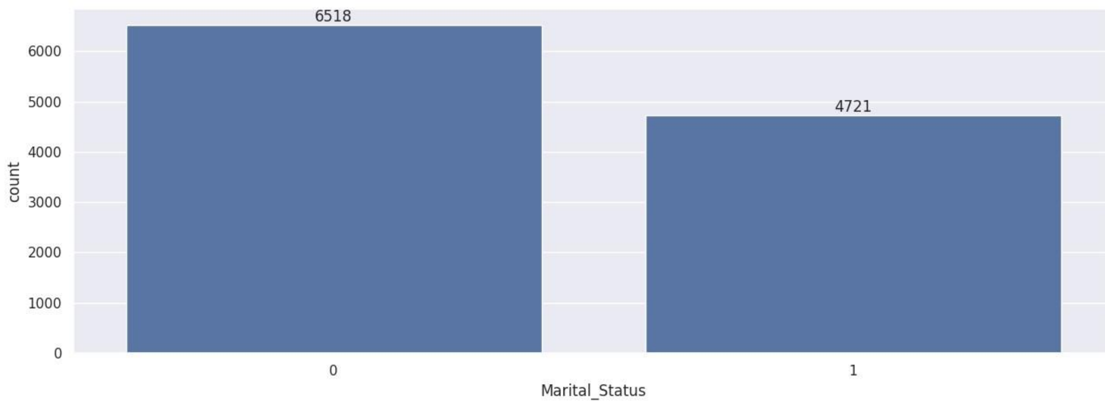


[24]:
```
# total amount/sales from top 10 states

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')
```

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

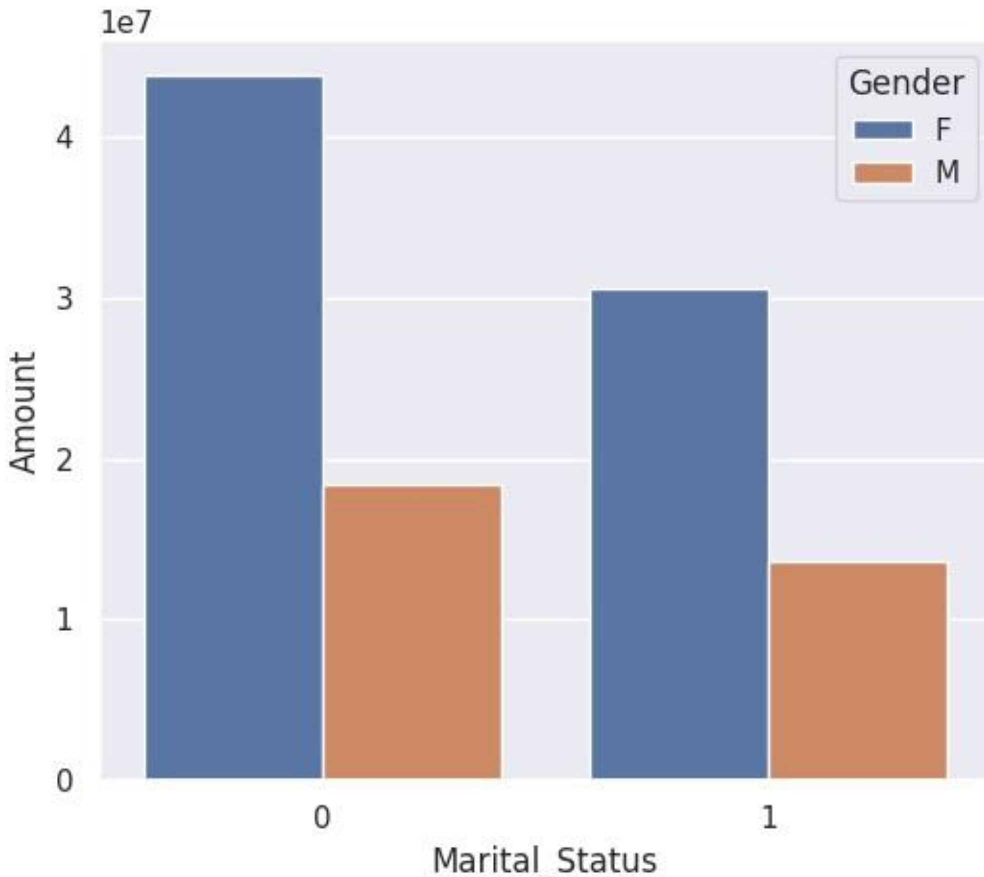


[25]:
```
ax = sns.countplot(data = df, x = 'Marital_Status')

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

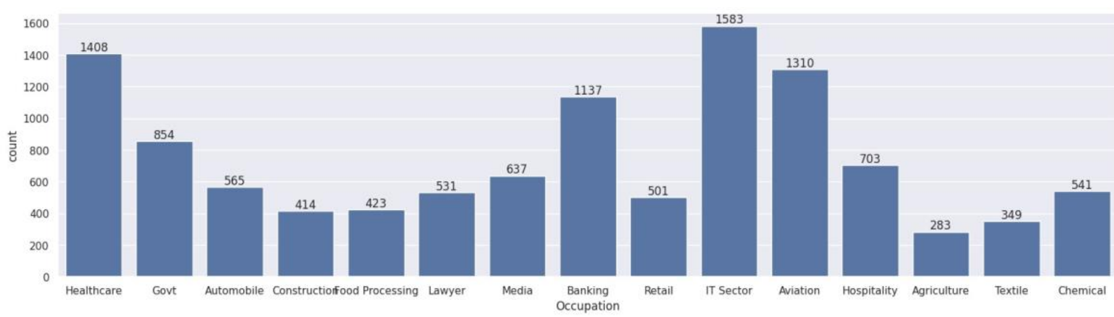



```
[26]: sales_state = df.groupby(['Marital_Status', 'Gender'],␣
      ↪as_index=False)['Amount'].sum().sort_values(by='Amount',
      ascending=False)

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

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



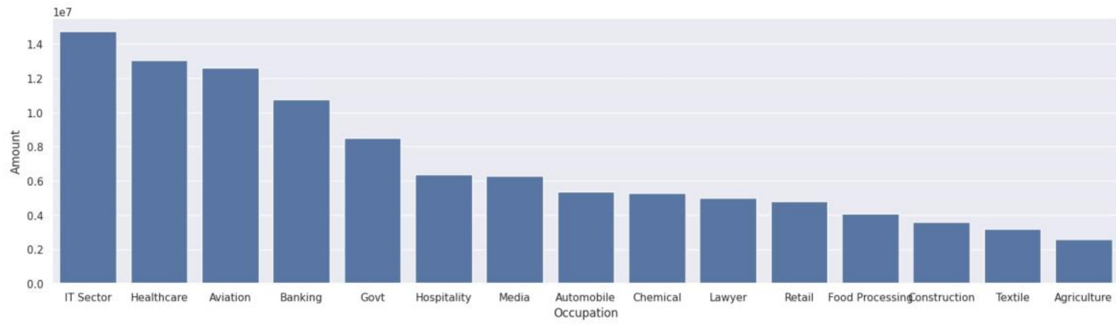
[27]:
```
sns.set(rc={'figure.figsize':(20,5)})
ax = sns.countplot(data = df, x = 'Occupation')

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



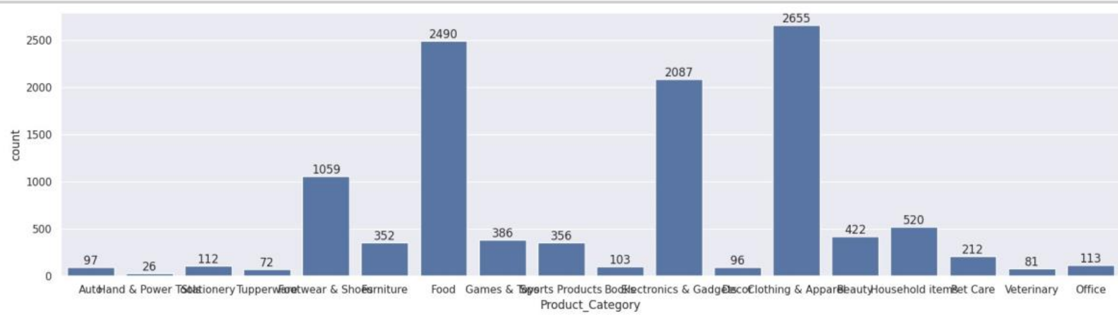
[28]:
```
sales_state = df.groupby(['Occupation'],
 as_index=False)['Amount'].sum(). ↪sort_values(by='Amount',
 ascending=False)
```

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

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

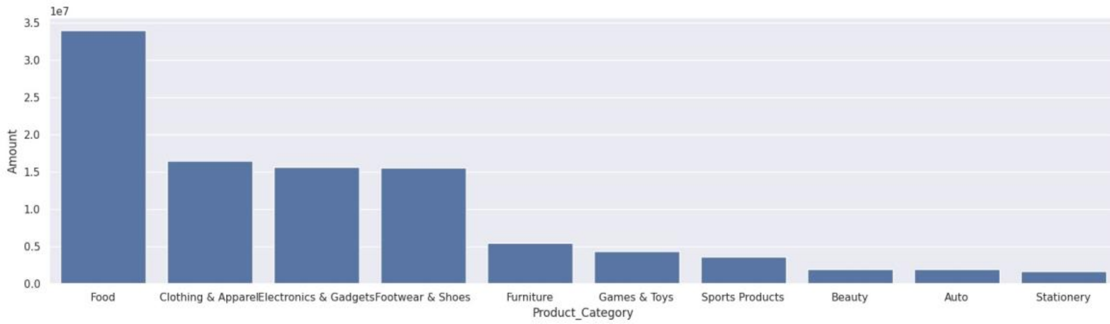


[29]:
```
sns.set(rc={'figure.figsize':(20,5)})
ax = sns.countplot(data = df, x = 'Product_Category')

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



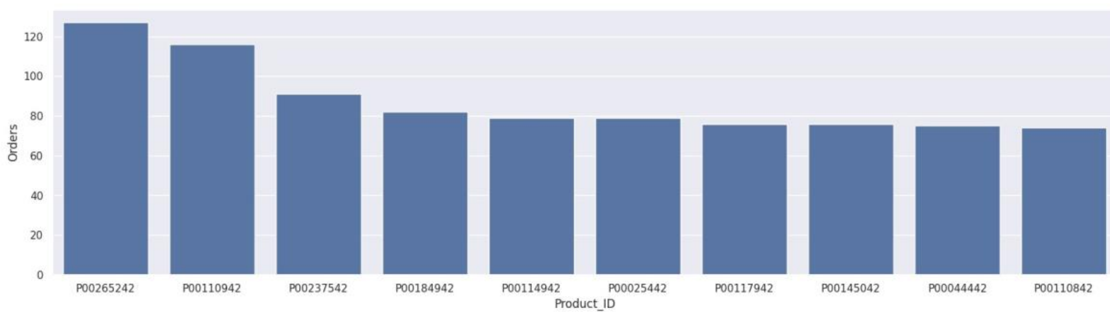
[30]:
```
sales_state = df.groupby(['Product_Category'],
 as_index=False)['Amount'].sum(). ↪sort_values(by='Amount',
 ascending=False).head(10)

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

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



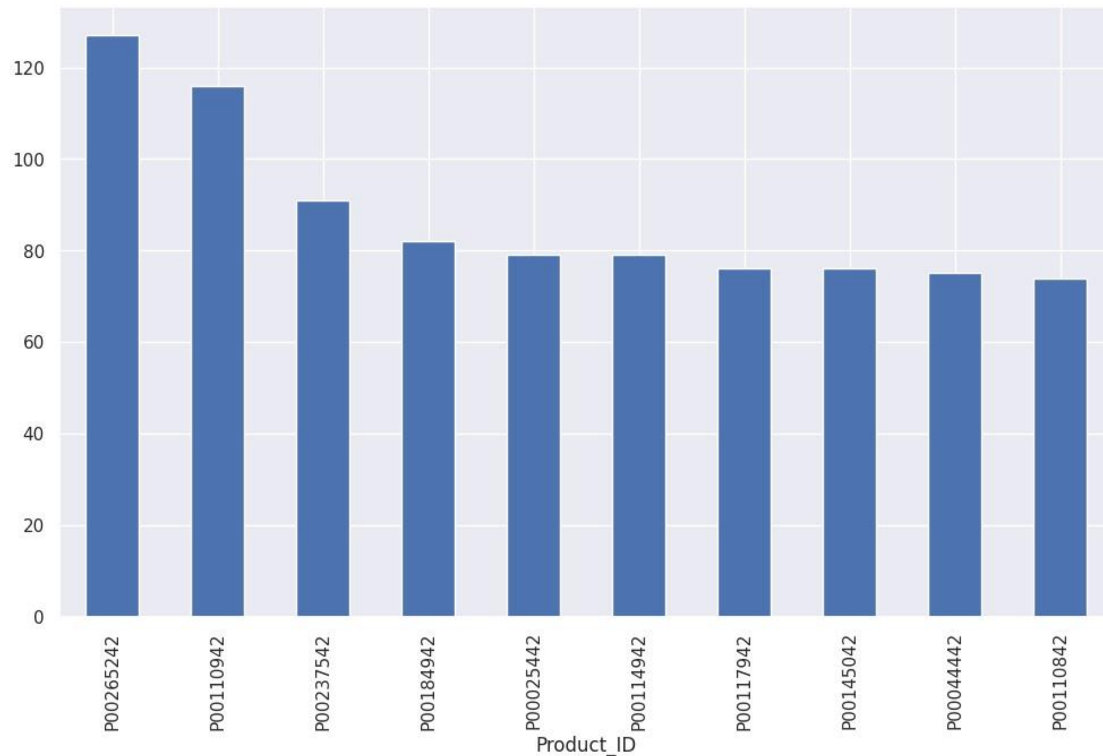
[31]:
```
sales_state = df.groupby(['Product_ID'],
 as_index=False)['Orders'].sum(). ↪sort_values(by='Orders',
 ascending=False).head(10)

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

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



[32]:
```
# 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='ba
                    r')
```

[32]: <Axes: xlabel='Product_ID'>


120
100
80
60
40
20
0
P00265242
P00110942
P00237542
P00184942
P00025442
P00114942
P00117942
P00145042
P00044442
P00110842
Product_ID

Project Summary

Project Title: Sales Data Analysis Using Python

Objective:

The project's goal is to analyze a dataset of sales transactions to extract insights regarding customer demographics, purchase behaviors, popular products, and regional sales trends, ultimately assisting in better business decision-making.

Data Overview:

- **Source**: CSV file of 11,251 entries with customer sales data.
- **Key Columns**:
  - **User Information**: User_ID, Cust_name, Gender, Age, Marital_Status, State, Zone, Occupation
  - **Product Information**: Product_ID, Product_Category
  - **Sales Metrics**: Orders, Amount
  - **Data Issues**: Null values in 'Amount' (12 entries) and columns 'Status' and 'unnamed1' with no useful data (dropped).

Data Cleaning and Preprocessing:

1. **Data Loading**: Imported using pandas.
2. **Null Value Handling**: Dropped null values from the 'Amount' column.
3. **Data Type Conversion**: Converted 'Amount' to integer type for numerical analysis.
4. **Column Renaming**: Renamed 'Marital_Status' to 'Shaadi' for clarity.

Exploratory Data Analysis (EDA):

1. **Demographics**:
   - **Gender Distribution**: Visualized the count of transactions by gender, indicating gender-specific purchase behaviors.
   - **Age Group Insights**: Analyzed the total amount spent per age group, showing spending trends across different age segments.
2. **Regional Sales**:
   - **Top States by Orders and Sales**: Identified top 10 states by the number of orders and sales revenue, helping to target key regions.
   - **Zone Analysis**: Focused on sales in specific zones to understand geographic distribution.
3. **Product Insights**:
   - **Top Product Categories**: Analyzed popular product categories by total sales to determine product demand.
   - **Most Sold Products**: Highlighted the top 10 products based on order volume, suggesting consumer preferences.
4. **Occupation and Marital Status**:
   - **Occupation Analysis**: Explored total sales by occupation, giving insight into customer profiles based on profession.
   - **Marital Status Impact**: Compared sales amounts by marital status, with further gender breakdown to identify spending patterns.

Visualizations:

Data visualizations were created using **Matplotlib** and **Seaborn**, including bar plots and count plots to represent categorical comparisons, such as gender distribution in spending, age-group-specific spending patterns, and popular product categories.

Key Insights:

- **Demographics**: Specific age groups and genders show higher spending patterns, providing a basis for targeted marketing.
- **Regional Focus**: Certain states contribute significantly to total sales, guiding regional marketing efforts.
- **Product Demand**: The analysis of product categories highlights key revenue-driving products, helping with inventory and promotion planning.

Tools Used:

- **Libraries**: pandas, numpy, matplotlib, seaborn
- **Platform**: Google Colab, with data hosted in Google Drive

Conclusion:

This project successfully uncovered valuable insights from sales data, revealing customer demographic trends, product preferences, and high-performing regions, enabling data-driven strategies for marketing and sales optimization

Shaun Mia | LinkedIn