**Title: DIWALI SALES**

In this repository, I've delved into a diverse dataset, extracting valuable insights and visualizations that shed light on various aspects. The dataset encompasses crucial columns such as 'State', 'Occupation', 'Product_Category', 'Age Group', and more, allowing for comprehensive analysis. From demographic trends to product preferences, this exploration aims to showcase the power of data-driven decision-making.

1.Importing all the libraries that are going to be used.
2.Data Pre-processing and cleaning.
3.Exploratory Data Analysis.
4.Data Visualization using various Graphs.

[114]:
```
#Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

[115]:
```
df = pd.read_csv("Diwali Sales Data.csv")
df
```

[115]:

| | 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 | | | | | |
| 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 | | | | | |

```
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
…             …      …             …             …     …
11246       Maharashtra Western Chemical   Office     4
11247       Haryana Northern    Healthcare Veterinary 3
11248       Madhya Pradesh Central    Textile    Office     4
11249       Karnataka Southern  Agriculture     Office     3
11250       Maharashtra Western Healthcare Office     3
       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
…        …     …       …
      11246   370.0   NaN   NaN
      11247   367.0   NaN   NaN
      11248   213.0   NaN   NaN
      11249   206.0   NaN   NaN
      11250   188.0   NaN   NaN
      11251   rows x 15 columns]
```

# 1 Pre-processing And Cleaning

[116]:
```
#First 5 Values
df.head()
```

[116]:
```
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
```

[117]:
```
#Last 5 values
df.tail(5)
```

[117]:
```
      User_ID  Cust_name Product_ID Gender Age Group Age Marital_Status
\
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     ZoneOccupation Product_Category Orders Amount \
11246   Maharashtra Western    Chemical         Office      4   370.0
11247       Haryana Northern Healthcare     Veterinary     3   367.0
11248 Madhya Pradesh Central    Textile         Office      4   213.0
11249    Karnataka Southern Agriculture        Office      3   206.0
11250   Maharashtra Western Healthcare         Office      3   188.0

       Status unnamed1
11246    NaN     NaN
11247    NaN     NaN
11248    NaN     NaN
11249    NaN     NaN
11250    NaN     NaN
```

[118]:
```
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
```

[119]:
```
df.describe
```

[119]:
```
<bound method NDFrame.describe of 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 04 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                          1

11248 1001209 Oshin P00201342 F 36-45 40 011249 1004023 Noonan P00059442 M
36-45 37 011250 1002744 Brumley P00281742 F 18-25 19 0State Zone Occupation
Product_Category Orders \0 Maharashtra Western Healthcare Auto            1

1 Andhra Pradesh Southern Govt Auto                                       2

2 Uttar Pradesh Central Automobile Auto                                   2


      …        …          …        …    …        … …              …
```

```
3           Karnataka Southern   Construction    Auto  2
4           Gujarat Western Food Processing Auto  2
      …              …       …             …             …     …
11246       Maharashtra Western  Chemical   Office     4
11247       Haryana Northern     Healthcare Veterinary 3
11248       Madhya Pradesh Central    Textile    Office     4
11249       Karnataka Southern   Agriculture     Office     3
11250       Maharashtra Western  Healthcare Office     3
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
      …       …     …       …
11246   370.0    NaN   NaN
11247   367.0    NaN   NaN
11248   213.0    NaN   NaN
11249   206.0    NaN   NaN
11250   188.0    NaN   NaN
      [11251 rows x 15 columns]>
```

[120]:
```
df.shape
```

[120]:
```
(11251, 15)
```

[121]:
```
df.drop(['Status','unnamed1'],axis=1,inplace = True)
```

[122]:
```
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
```

[123]:
```
df.isnull().sum()
```

[123]:
```
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
```

[124]:
```
df.dropna(inplace = True)
```

[125]:
```
df.info()
```

```
<class
'pandas.core.frame.DataFrame'>
Int64Index: 11239 entries, 0 to
11250 Data columns (total 13
columns):
 # Column          Non-Null Count Dtype
--- ------          -------------- ----
                    -
 0 User_ID          11239 non-null
                    int64
 1 Cust_name        11239      non-null
                    object
 2 Product_ID       11239      non-null
                    object
 3 Gender           11239      non-null
                    object
 4 Age Group        11239      non-null
                    object
 5 Age              11239 non-null
                    int64
 6 Marital_Status   11239      non-null
                    int64
 7 State            11239      non-null
                    object
 8 Zone             11239      non-null
                    object
 9 Occupation       11239      non-null
                    object
 10  Product_Category 11239 non-null object
 11  Orders 11239 non-null int64
 12  Amount 11239 non-null float64
dtypes: float64(1), int64(4), object(8)
memory usage: 1.2+ MB
```

[126]:
```
df[['Age','Orders','Amount']].describe()
```

[126]:
```
Age  Orders     Amount count 11239.000000
11239.000000 11239.000000
mean     35.410357    2.489634
                      9453.610858
std      12.753866    1.114967
                      5222.355869
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
```

[127]:
```
age_mean,age_std,*_ = df.Age.describe()
```

[128]:
```
print(age_mean)
print(age_std)
```

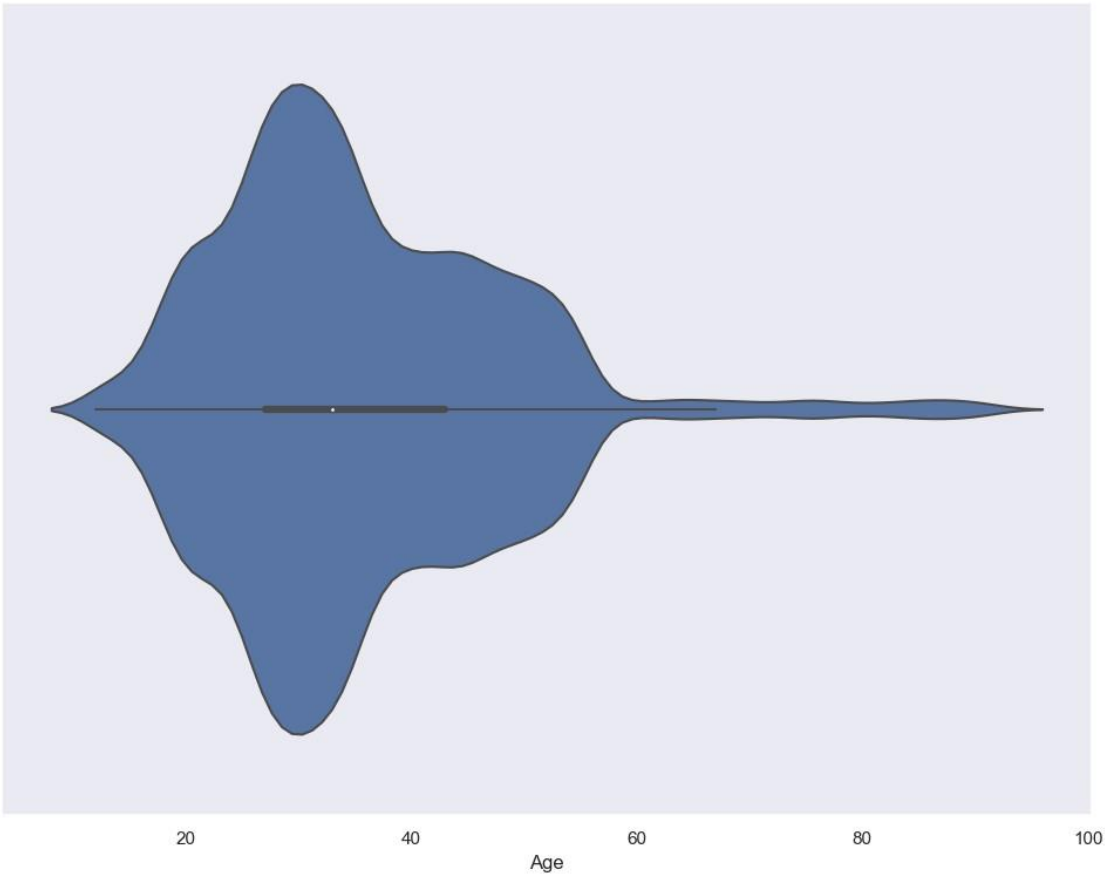
```
11239.0
35.41035679330901
```

[129]:
```
df[df.Age >(age_mean + 3.5*age_std)]
```

[129]:
```
Empty DataFrame
Columns: [User_ID, Cust_name, Product_ID, Gender, Age Group, Age,
Marital_Status, State, Zone, Occupation, Product_Category, Orders,
Amount] Index: []
```

[130]:
```
import warnings
warnings.filterwarnings("ignore")
```

[131]:
```
plt.figure(figsize = (12,9))
sns.violinplot(df.Age)
plt.show()
```




[132]:
```
amount_mean,amount_std, *_ = df.Amount.describe()
print(amount_mean)
print(amount_std)
```

```
11239.0
9453.610857727557
```

[133]:
```
df[df.Amount > amount_mean +(3.5*amount_std)]
```

[133]:
```
Empty DataFrame
  Columns: [User_ID, Cust_name, Product_ID, Gender, Age
                        Group, Age,
Marital_Status, State, Zone, Occupation, Product_Category, Orders,
Amount] Index: []
```

[134]:
```
df.describe(include = 'object')
```

[134]:
```
       Cust_name Product_ID Gender Age Group      State     Zone \
count      11239     11239  11239     11239      11239    11239
unique      1250      2350      2         7         16        5
top    Vishakha P00265242      F     26-35    Uttar     Pradesh
                                               Central
freq         42        53   7832      4541       1944     4289

       Occupation  Product_Category
count       11239             11239
unique         15                18
top     IT Sector Clothing &
        Apparel
freq         1583              2655
```

[135]:
```
df.shape
```

[135]:
```
(11239, 13)
```

[136]:
```
df['Cust_name'].value_counts()
```

[136]:
```
Vishakha     42
Shreyshi     32
Sudevi       30
Akshat       29
Alejandro    28
             ..
Overcash      2
Madan Mohan   2
Madhav        2
Laal          1
Bindu         1
Name: Cust_name, Length: 1250, dtype: int64
```

[137]:
```
df['Cust_name'].value_counts().nlargest(3)
```

[137]:
```
Vishakha  42
Shreyshi  32
Sudevi    30
Name: Cust_name, dtype: int64
```

[138]:
```
df['Cust_name'].value_counts().nsmallest(3)
```

[138]:
```
Laal 1 Bindu
1
Sarita   2
Name: Cust_name, dtype: int64
```

[139]:
```
df['Product_ID'].value_counts().nlargest(10)
```

[139]:
```
P00265242 53
```

```
P00110942  44
P00184942  37
P00237542  35
P00112142  34
P00114942  33
P00110742  32
P00112542  30
P00110842  30
P00145042  30
Name: Product_ID, dtype: int64
```

[140]:
```
cat = df.describe(include = 'object').columns
cat
```

[140]:
```
Index(['Cust_name', 'Product_ID', 'Gender', 'Age Group', 'State',
'Zone',
       'Occupation', 'Product_Category'],
      dtype='object')
```

[141]:
```
num = df.describe().columns
num
```

[141]:
```
Index(['User_ID', 'Age', 'Marital_Status', 'Orders', 'Amount'],
dtype='object')
```

[142]:
```
df['Gender'].describe()
```

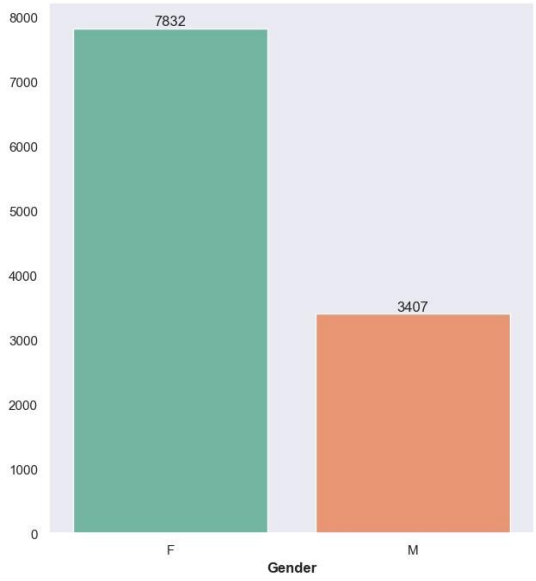
[142]:
```
count   11239
unique      2
top         F
freq     7832
Name: Gender, dtype: object
```

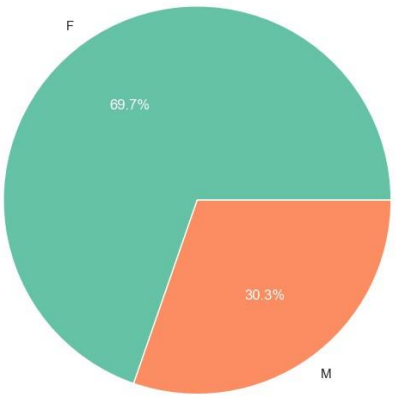
[143]:
```
df['Gender'].value_counts()
```

[143]:
```
F
     783
2 M
     340
7
```

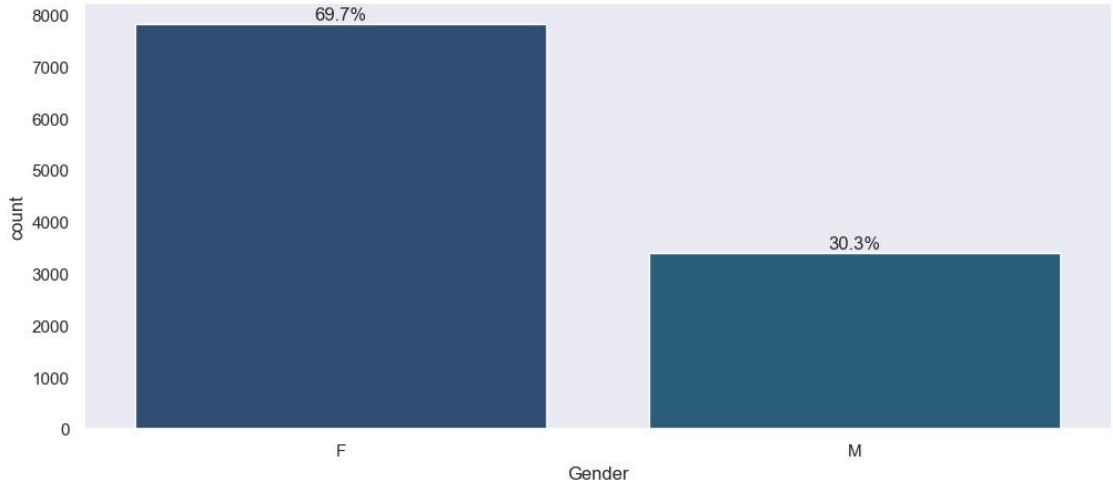
```
Name: Gender, dtype: int64
```

[144]:
```
# Create subplots
fig, ax = plt.subplots(1, 2, figsize=(16, 8))
# Get the value counts for the Gender column
data = df.Gender.value_counts()
# Set the Seaborn style
sns.set(style="dark", color_codes=True)
# Define the color palette
pal = sns.color_palette("Set2", len(data))
# Create the bar plot
sns.barplot(x=data.index, y=data.values, palette=pal, ax=ax[0])
# Add annotations to the bar plot
for bar in ax[0].patches:
    ax[0].annotate(
        '{:.0f}'.format(bar.get_height()),
        (bar.get_x() + bar.get_width() / 2, bar.get_height()),
        ha='center', va='bottom'
    )
# Set the label for the x-axis
ax[0].set_xlabel("Gender", weight="semibold")
# Create the pie chart
_, _, autotexts = ax[1].pie(data.values, labels=data.index, autopct="%.1f%%",
  colors=pal)
# Set the color of the text in the pie chart to white
for autotext in autotexts:
    autotext.set_color("white")
# Show the plot
plt.show()
```



[145]:
```
# Set the figure size
plt.figure(figsize=(12, 5))

# Calculate the total number of records
total = len(df)

# Create the count plot with the "magma" color palette
ax = sns.countplot(x='Gender', data=df, palette=sns.color_palette("crest_r"))

# Add annotations to the bars
for bar in ax.patches:
    height = bar.get_height()
    ax.annotate(
        '{:.1f}%'.format(height / total * 100),
        (bar.get_x() + bar.get_width() / 2, height),
        ha='center', va='bottom'
    )

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



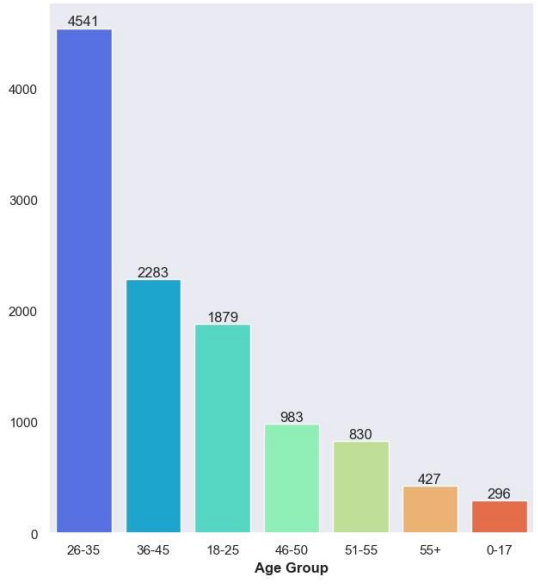

[146]:
```
df['Age Group'].describe()
```

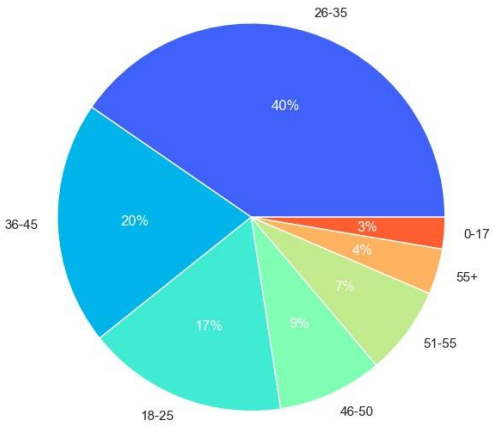
[146]:
```
count   11239
unique      7
top     26-35
freq     4541
```

```
Name: Age Group, dtype: object
```

[147]:
```
fig, ax = plt.subplots(1, 2, figsize=(16, 8))

# Get the value counts for the 'Age Group' column
data = df['Age Group'].value_counts()

# Set the Seaborn style and color palette
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("rainbow", len(data))
```

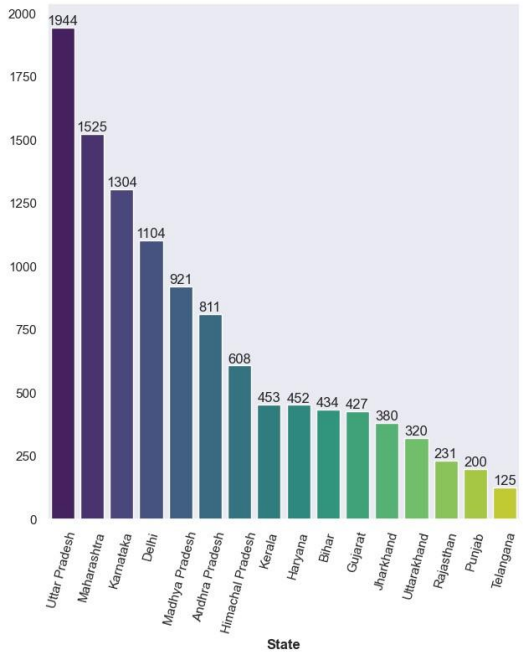
```
# Create the bar plot
sns.barplot(x=data.index, y=data.values, palette=pal, ax=ax[0])

# Add annotations to the bar plot
for bar in ax[0].patches:
    ax[0].annotate(
        '{:.0f}'.format(bar.get_height()),
        (bar.get_x() + bar.get_width() / 2, bar.get_height()),
        ha='center', va='bottom'
    )

# Set the label for the x-axis
ax[0].set_xlabel("Age Group", weight="semibold")

# Create the pie chart
_, _, autotexts = ax[1].pie(data.values, labels=data.index, autopct='%.0f%%',
  colors=pal)

# Set the color of the text in the pie chart to white
for autotext in autotexts:
    autotext.set_color("white")

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

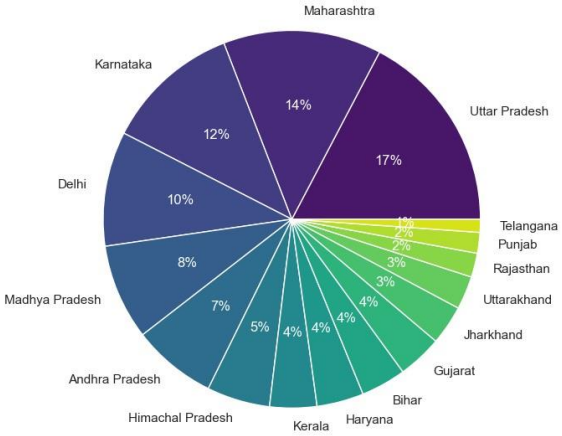



[148]:
```
# Create subplots
fig, ax = plt.subplots(1, 2, figsize=(16, 8))
```

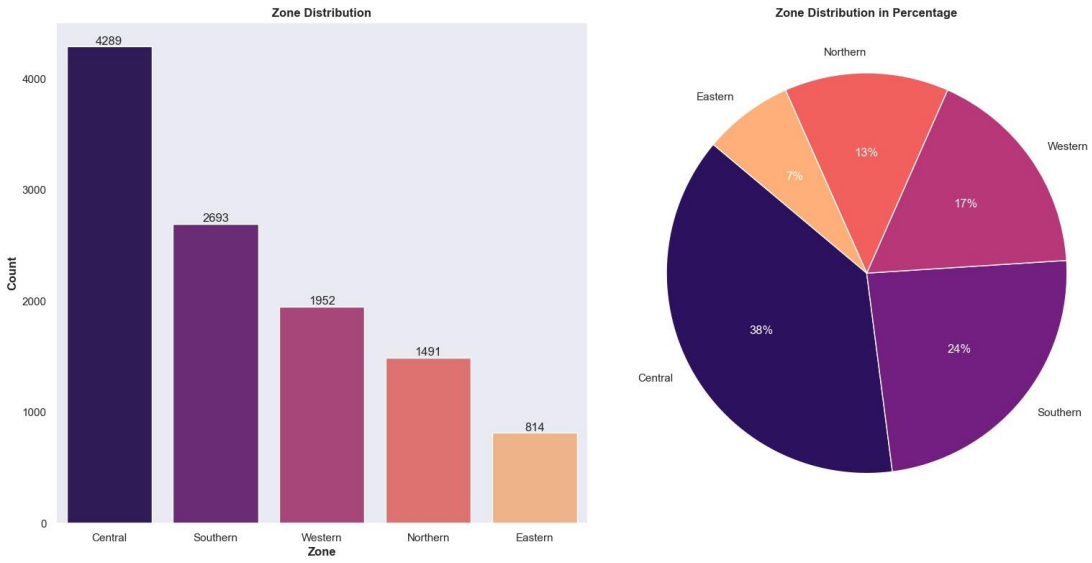
```
# Get the value counts for the 'State' column
data = df['State'].value_counts()

# Set the Seaborn style and color palette
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("viridis", len(data))

# Create the bar plot
my_plot = sns.barplot(x=data.index, y=data.values, palette=pal, ax=ax[0])

# Add annotations to the bar plot
for bar in ax[0].patches:
    ax[0].annotate(
        '{:.0f}'.format(bar.get_height()),
        (bar.get_x() + bar.get_width() / 2, bar.get_height()),
        ha='center', va='bottom'
    )

# Set the label for the x-axis
ax[0].set_xlabel("State", weight="semibold")

# Rotate x-axis labels for better readability
my_plot.set_xticklabels(my_plot.get_xticklabels(), rotation=75)

# Create the pie chart
_, _, autotexts = ax[1].pie(data.values, labels=data.index, autopct='%.0f%%',
  colors=pal)

# Set the color of the text in the pie chart to white
for autotext in autotexts:
    autotext.set_color("white")

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




[149]:
```
# Create subplots
fig, ax = plt.subplots(1, 2, figsize=(16, 8))

# Get the value counts for the 'Zone' column
data = df['Zone'].value_counts()

# Set the Seaborn style and use the "husl" palette for an attractive color
 scheme
sns.set(style="dark", color_codes=True)
pal = sns.color_palette("magma", len(data))

# Create the bar plot
sns.barplot(x=data.index, y=data.values, palette=pal, ax=ax[0])

# Add annotations to the bar plot
for bar in ax[0].patches:
    ax[0].annotate(
        '{:.0f}'.format(bar.get_height()),
        (bar.get_x() + bar.get_width() / 2, bar.get_height()),
        ha='center', va='bottom'
    )

# Set the labels and title for the bar plot
ax[0].set_xlabel("Zone", weight="semibold")
ax[0].set_ylabel("Count", weight="semibold")
```

```
ax[0].set_title("Zone Distribution", weight="bold")

# Create the pie chart
_, _, autotexts = ax[1].pie(data.values, labels=data.index, autopct='%.0f%%',
  colors=pal, startangle=140)

# Set the color of the text in the pie chart to white
for autotext in autotexts:
    autotext.set_color("white")

# Set the title for the pie chart
ax[1].set_title("Zone Distribution in Percentage ", weight="bold")

# Adjust layout to prevent overlap
plt.tight_layout()

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




[150]:
```
df['Occupation'].describe()
```

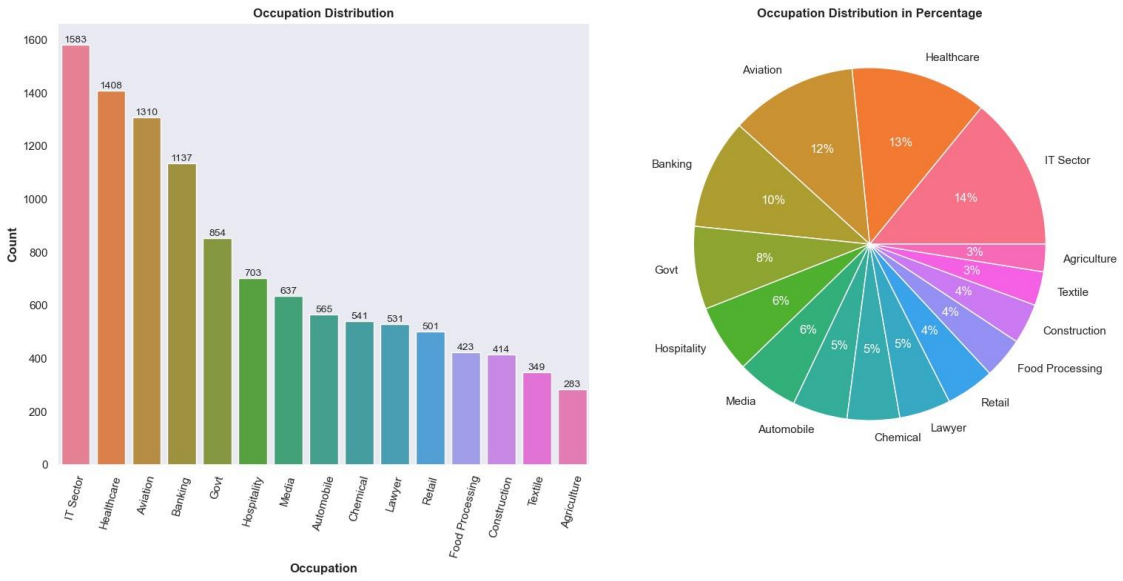
[150]:
```
count     11239
unique    15 top
     IT Sector freq
     1583
Name: Occupation, dtype: object
```

[151]:
```
df['Occupation'].value_counts()
```

[151]:
```
IT Sector       1583
Healthcare      1408
 Aviation       1310
 Banking        1137
 Govt            854
Hospitality      703
 Media           637
Automobile       565
 Chemical        541
 Lawyer          531
 Retail          501
 Food Processing 423
Construction     414
 Textile         349
Agriculture      283
 Name: Occupation, dtype: int64
```

[152]:
```
# Create subplots
fig, ax = plt.subplots(1, 2, figsize=(16, 8))

# Get the value counts for the 'Occupation' column
data = df['Occupation'].value_counts()

# Set a custom palette with distinct colors for each category
unique_colors = sns.color_palette("husl", len(data))
pal = {occupation: color for occupation, color in zip(data.index,
 unique_colors)}

# Set the Seaborn style
sns.set(style="dark", color_codes=True)

# Create the bar plot
my_plot = sns.barplot(x=data.index, y=data.values, palette=pal, ax=ax[0])

# Add annotations to the bar plot
for bar in ax[0].patches:
    ax[0].annotate(
        '{:.0f}'.format(bar.get_height()),
        (bar.get_x() + bar.get_width() / 2, bar.get_height()),
        ha='center', va='bottom', fontsize=10  # Increase font size for
 visibility
    )

# Set the labels and title for the bar plot
ax[0].set_xlabel("Occupation", weight="semibold")
ax[0].set_ylabel("Count", weight="semibold")
ax[0].set_title("Occupation Distribution", weight="bold")
```

```
# Rotate x-axis labels for better readability
my_plot.set_xticklabels(my_plot.get_xticklabels(), rotation=75)

# Create the pie chart
_, _, autotexts = ax[1].pie(data.values, labels=data.index, autopct='%.0f%%',
    colors=unique_colors)

# Set the color of the text in the pie chart to white and increase font size
for autotext in autotexts:
    autotext.set_color("white")
    autotext.set_fontsize(12)  # Increase font size for visibility

# Set the title for the pie chart
ax[1].set_title("Occupation Distribution in Percentage", weight="bold")

# Adjust layout to prevent overlap
plt.tight_layout()

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




[153]:
```
df['Product_Category'].describe()
```

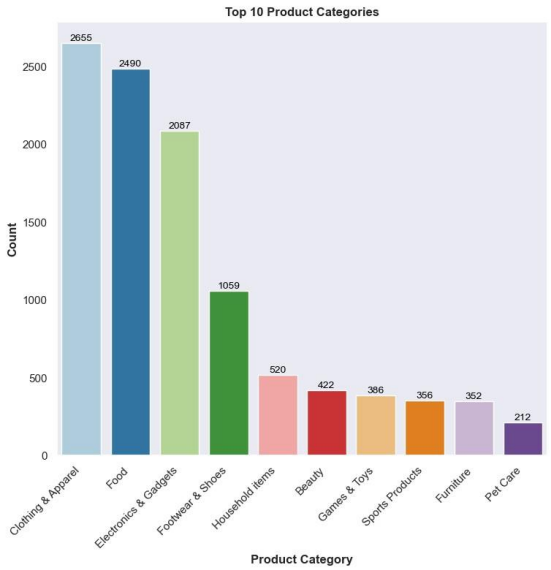
```
[153]: count     11239 unique
        18 top      Clothing &
          Apparel freq   2655
Name: Product_Category, dtype: object
```

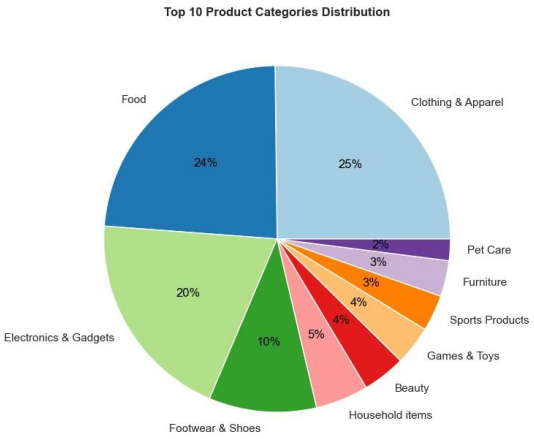
[154]:
```
df['Product_Category'].value_counts()
```

[154]:
```
Clothing & Apparel   2655
Food                 2490
Electronics &        2087
Gadgets
Footwear & Shoes     1059
Household items       520
Beauty                422
Games & Toys          386
Sports Products       356
Furniture             352
Pet Care              212
Office                113
Stationery            112
Books                 103
Auto                   97
Decor                  96
Veterinary             81
Tupperware             72
Hand & Power Tools     26
Name: Product_Category, dtype: int64
```

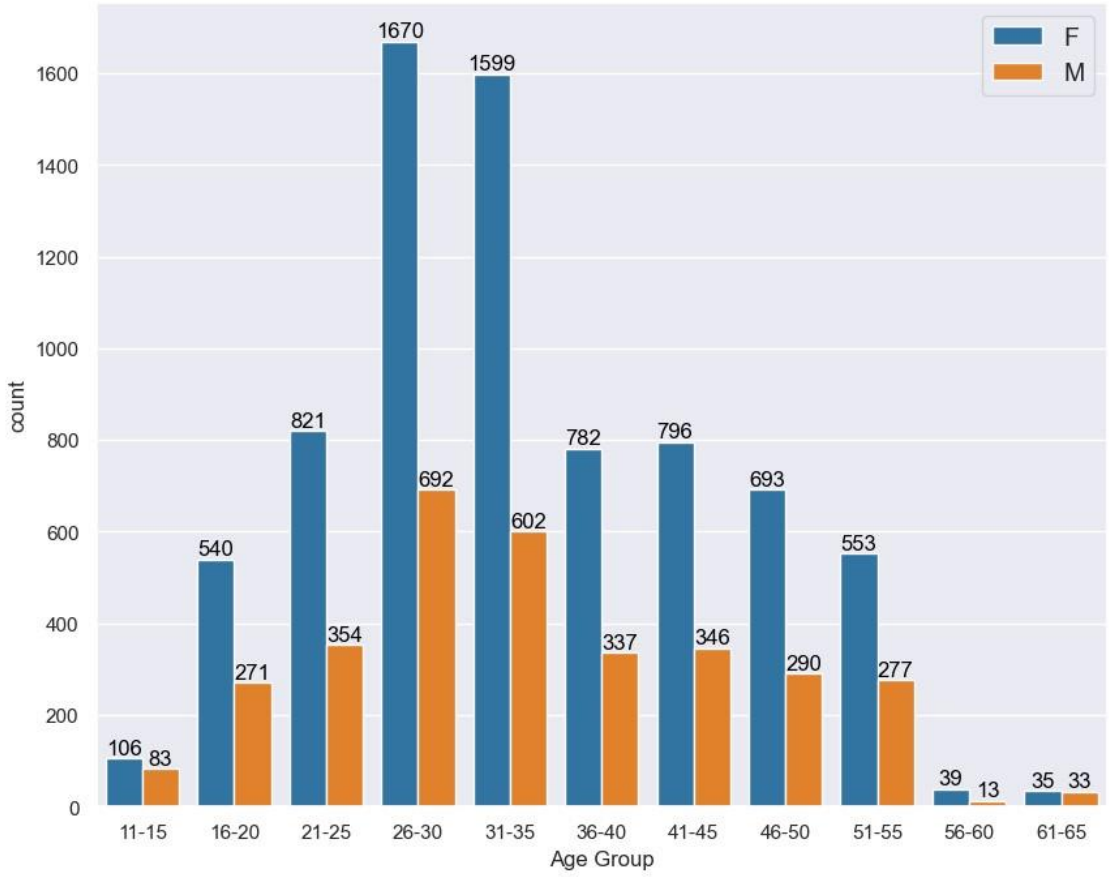
[155]:
```
# Filter the top 10 observations for 'Product_Category'
top_categories = df['Product_Category'].value_counts().head(10)

# Create subplots
fig, ax = plt.subplots(1, 2, figsize=(16, 8))

# Set a visually appealing palette
pal = sns.color_palette("Paired")

# Set the Seaborn style
sns.set(style="darkgrid", color_codes=True)

# Create the bar plot with the top 10 categories
my_plot = sns.barplot(x=top_categories.index, y=top_categories.values,
  palette=pal, ax=ax[0])

# Add annotations to the bar plot
for bar in ax[0].patches:
    ax[0].annotate(
        '{:.0f}'.format(bar.get_height()),
        (bar.get_x() + bar.get_width() / 2, bar.get_height()),
        ha='center', va='bottom', fontsize=10, color='black'  # Font properties
  for visibility
```

```
    )

# Set the labels and title for the bar plot
ax[0].set_xlabel("Product Category", weight="semibold")
ax[0].set_ylabel("Count", weight="semibold")
ax[0].set_title("Top 10 Product Categories ", weight="bold")

# Rotate x-axis labels for better readability
my_plot.set_xticklabels(my_plot.get_xticklabels(), rotation=45, ha='right')

# Create the pie chart with the top 10 categories
_, _, autotexts = ax[1].pie(top_categories.values, labels=top_categories.index,
  autopct='%.0f%%', colors=pal)

# Set the color of the text in the pie chart to black for visibility
for autotext in autotexts:
    autotext.set_color("black")

# Set the title for the pie chart
ax[1].set_title("Top 10 Product Categories Distribution ", weight="bold")

# Adjust layout to prevent overlap
plt.tight_layout()

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



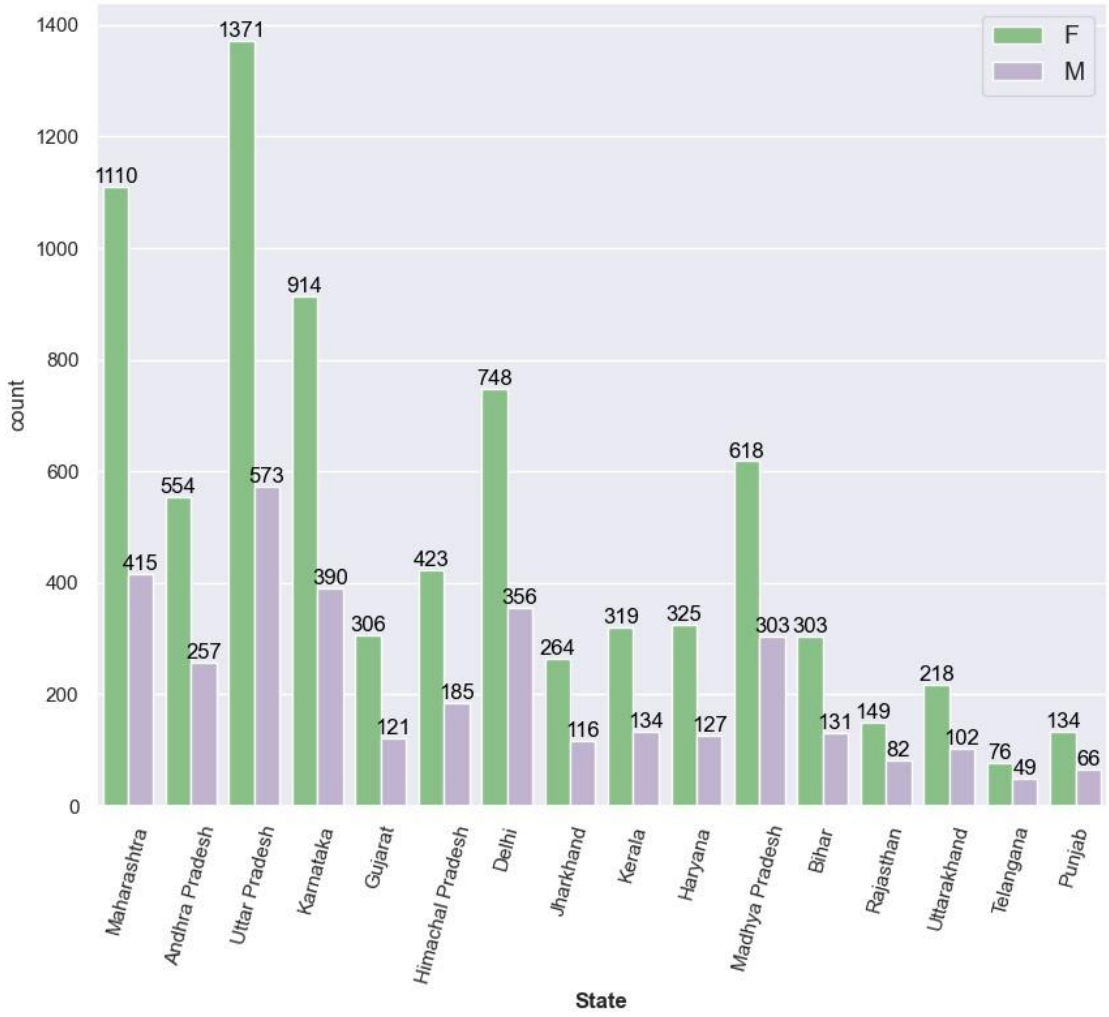



[156]:
```
# Set the figure size
plt.figure(figsize=(10, 8))

# Define the age group categories in increasing order
age_groups = ['11-15', '16-20', '21-25', '26-30', '31-35', '36-40', '41-45',␣
 ↪'46-50', '51-55', '56-60', '61-65']

# Re-categorize the 'Age Group' column based on the defined categories
df['Age Group'] = pd.cut(df['Age'], bins=[10, 15, 20, 25, 30, 35, 40, 45, 50,␣
 ↪55, 60, 65], labels=age_groups)

# Get the value counts for 'Age Group' and choose a palette
data = df['Age Group'].value_counts()
pal = sns.color_palette("tab10", len(data))

# Set the Seaborn style
sns.set(style="darkgrid")

# Create the countplot with hue based on 'Gender'
ax = sns.countplot(x='Age Group', data=df, hue='Gender', palette=pal,␣
 ↪order=age_groups)

# Add annotations to the bars
for bar in ax.patches:
    ax.annotate('{:.0f}'.format(bar.get_height()),
                (bar.get_x() + bar.get_width() / 2, bar.get_height()),
                ha='center', va='bottom', fontsize=12, color='black')

# Set legend with increased font size
plt.legend(loc='best', fontsize=14)

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




[157]:
```
# Set the figure size
plt.figure(figsize=(10, 8))

# Choose a visually appealing palette
pal = sns.color_palette("Accent")

# Set the Seaborn style
sns.set(style="darkgrid")

# Create the countplot with hue based on 'Gender'
ax = sns.countplot(x='State', data=df, hue='Gender', palette=pal)

# Add annotations to the bars
for bar in ax.patches:
    ax.annotate('{:.0f}'.format(bar.get_height()),
                (bar.get_x() + bar.get_width() / 2, bar.get_height()),
                ha='center', va='bottom', fontsize=12, color='black')

# Set legend with increased font size
```

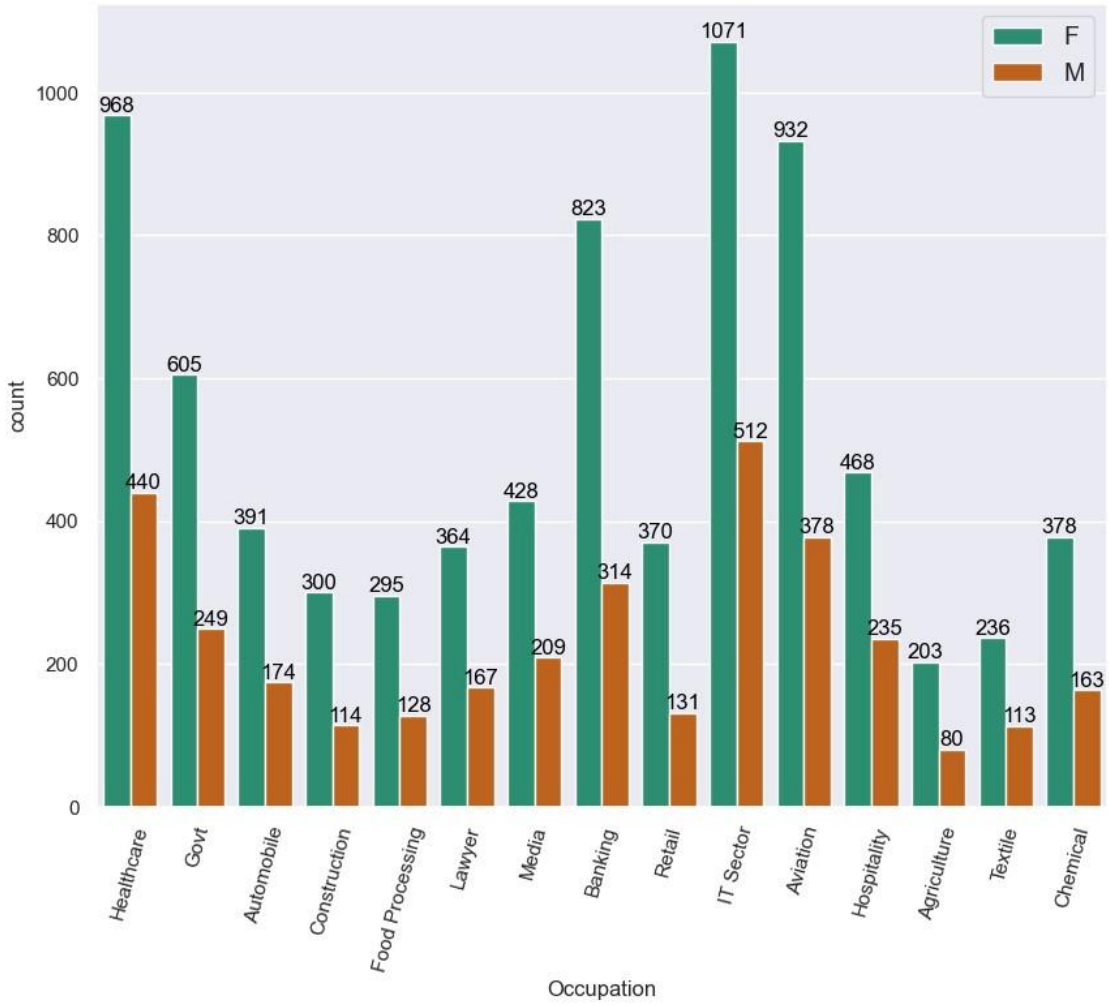
```
plt.legend(loc='best', fontsize=14)

# Rotate x-axis labels for better readability
plt.xticks(rotation=75)

# Set bold x-axis title
plt.xlabel("State", weight="bold")

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



[158]:
```
cat
```

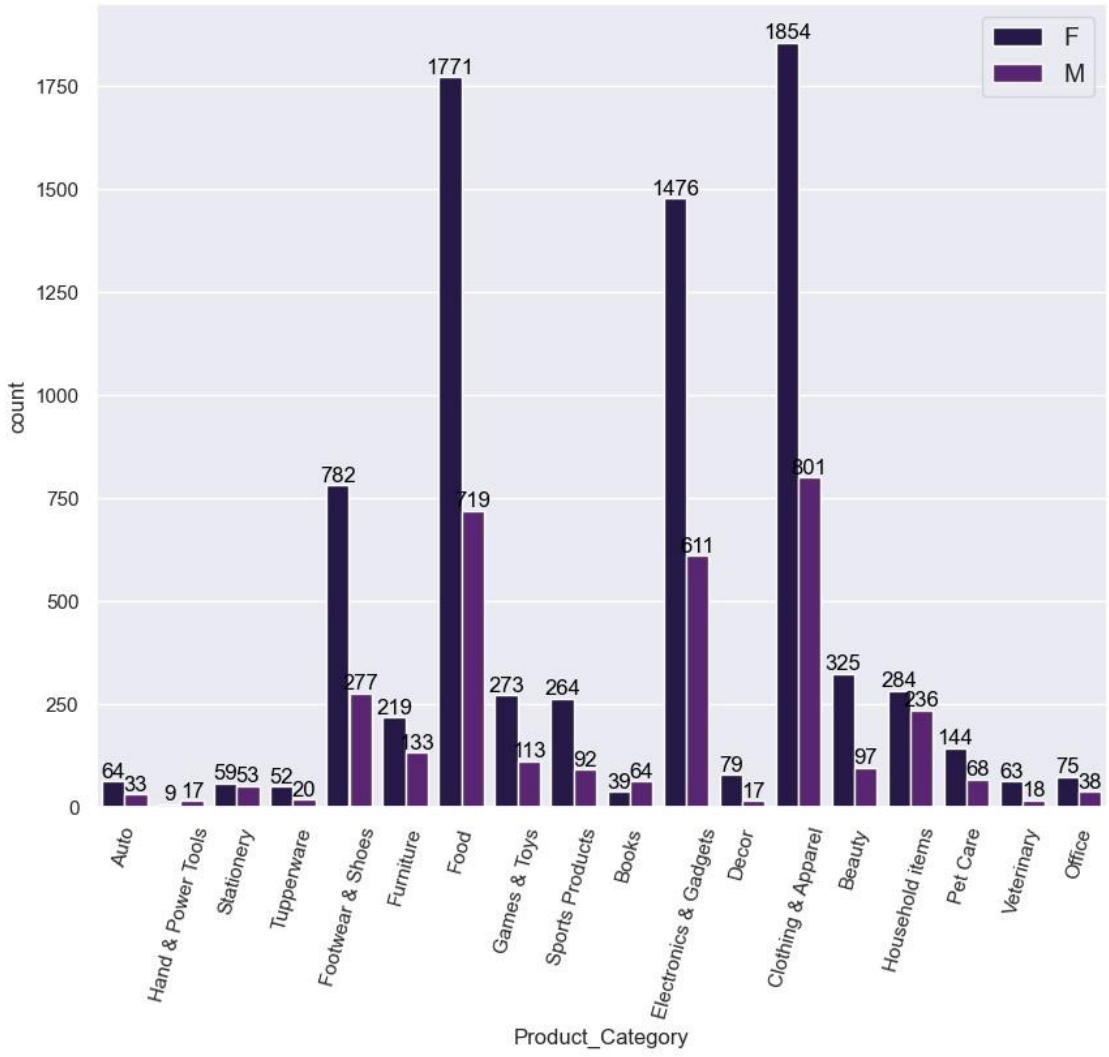
[158]: Index(['Cust_name', 'Product_ID', 'Gender', 'Age Group', 'State',
'Zone',
'Occupation', 'Product_Category'],
dtype='object')

[159]:
```
plt.figure(figsize=(10, 8))

# Choose a visually appealing palette
pal = sns.color_palette("Dark2")

# Set the Seaborn style
sns.set(style="darkgrid")

# Create the countplot with hue based on 'Gender'
ax = sns.countplot(x='Occupation', data=df, hue='Gender', palette=pal)

# Add annotations to the bars
for bar in ax.patches:
    ax.annotate('{:.0f}'.format(bar.get_height()),
                (bar.get_x() + bar.get_width() / 2, bar.get_height()),
                ha='center', va='bottom', fontsize=12, color='black')

# Set legend with increased font size
plt.legend(loc='best', fontsize=14)

# Rotate x-axis labels for better readability
plt.xticks(rotation=75)

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




[160]:
```
# Set the figure size
plt.figure(figsize=(10, 8))

# Choose a visually appealing palette
pal = sns.color_palette("magma")

# Set the Seaborn style
sns.set(style="darkgrid")

# Create the countplot with hue based on 'Gender'
ax = sns.countplot(x='Product_Category', data=df, hue='Gender', palette=pal)

# Add annotations to the bars
for bar in ax.patches:
    ax.annotate('{:.0f}'.format(bar.get_height()),
                (bar.get_x() + bar.get_width() / 2, bar.get_height()),
```

```
                ha='center', va='bottom', fontsize=12, color='black')

# Set legend with increased font size
plt.legend(loc='best', fontsize=14)

# Rotate x-axis labels for better readability
plt.xticks(rotation=75)

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

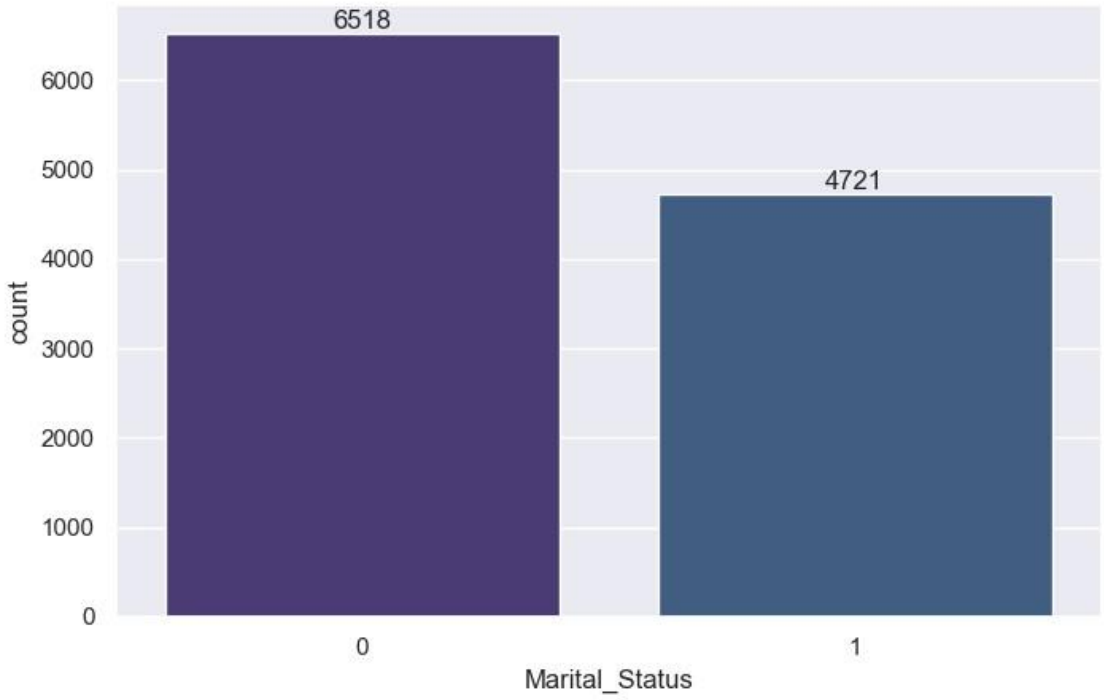



[161]:
```
df['Marital_Status'] = df['Marital_Status'].astype('str')
```

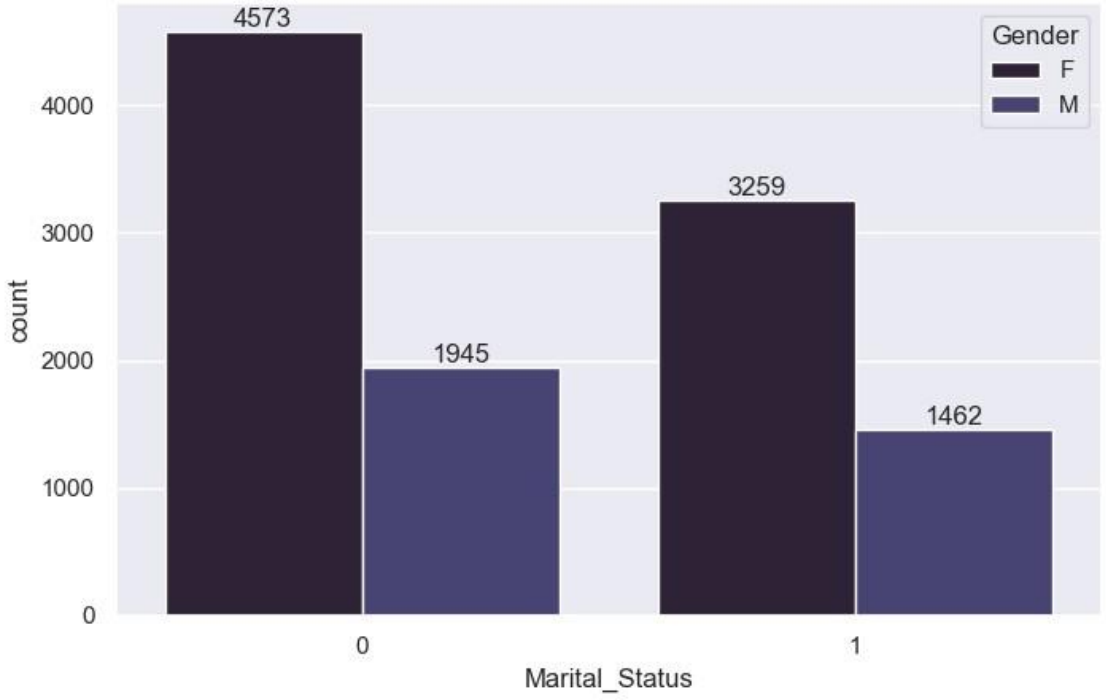
[162]:
```
df.Marital_Status.value_counts()
```

```
[162]: 0
  6518 1
    4721
Name: Marital_Status, dtype: int64
```

[163]:
```
# Set the figure size
plt.figure(figsize=(8, 5))

# Choose a different visually appealing palette
pal = sns.color_palette('viridis')

# Create the countplot with the chosen palette
ax = sns.countplot(x='Marital_Status', data=df, palette=pal)

# Add annotations to the bars
for bar in ax.containers:
    ax.bar_label(bar)

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



[164]:
```
# Set the figure size
plt.figure(figsize=(8, 5))

# Choose a visually appealing palette
pal = sns.color_palette('mako')

# Create the countplot with hue based on 'Gender' and the chosen palette
ax = sns.countplot(x='Marital_Status', data=df, hue='Gender', palette=pal)

# Add annotations to the bars
for bar in ax.containers:
    ax.bar_label(bar)

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




[165]:
```
df.describe(include = 'object')
```

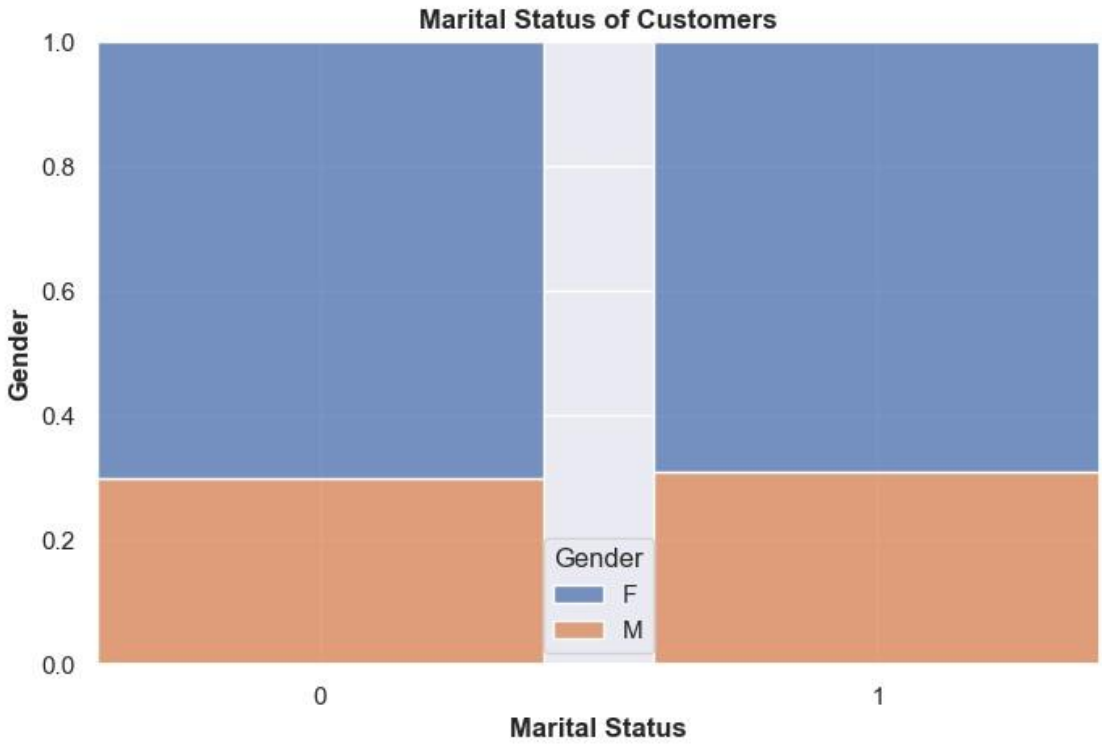
[165]:

| | Cust_name | Product_ID | Gender | Marital_Status | State | Zone |
|---|---|---|---|---|---|---|
| count | 11239 | 11239 | 11239 | 11239 | 11239 | 11239 |
| unique | 1250 | 2350 | 2 | 2 | 16 | 5 |
| top | Vishakha | P00265242 | F | 0 | Uttar Pradesh | Central |

```
freq         42         53 7832          6518          1944    4289
```

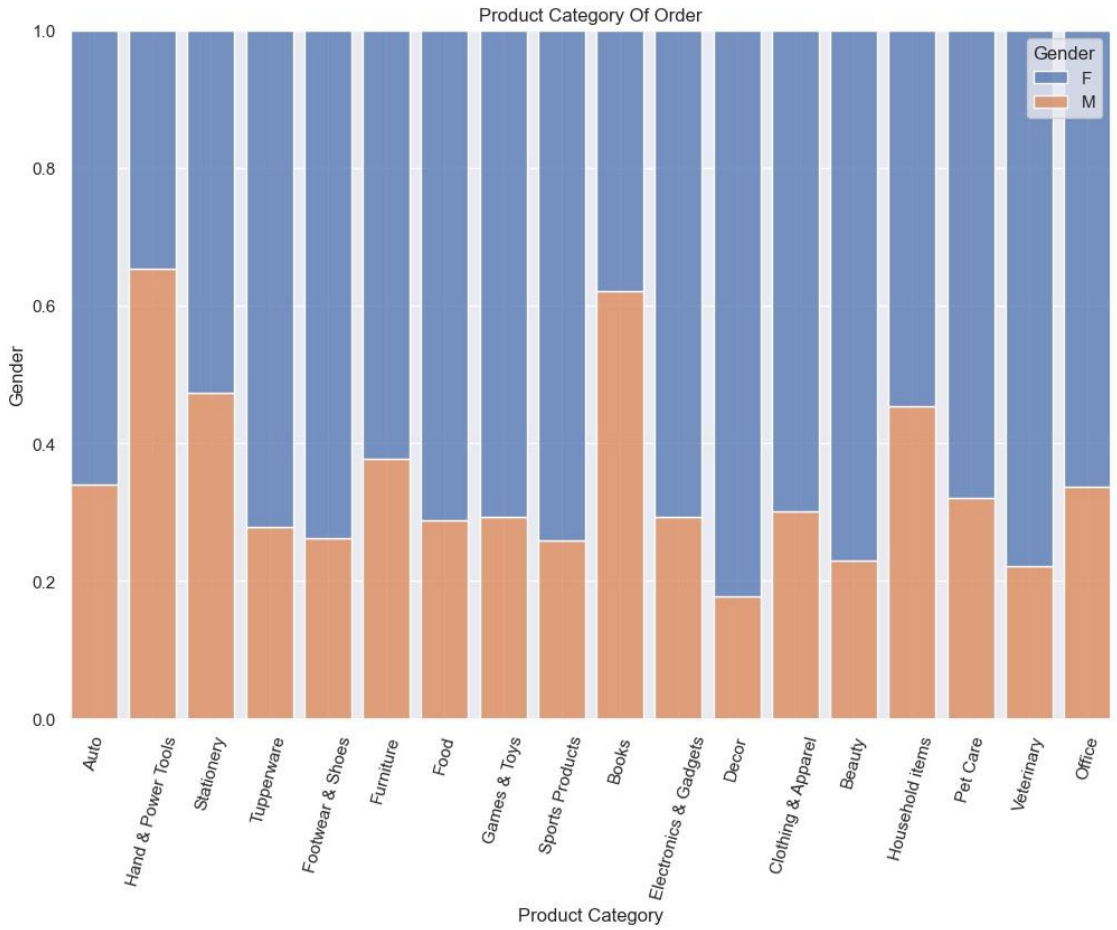
```
       Occupation Product_Category
count      11239             11239
unique        15                18
top     IT Sector Clothing & Apparel
freq         1583              2655
```

[166]:
```
# Set the figure size
plt.figure(figsize=(8, 5))

# Create the histogram with hue based on 'Gender'
ax = sns.histplot(
    data=df,
    x="Marital_Status",
    hue="Gender",
    multiple="fill",
    stat="proportion",
    discrete=True,
    shrink=.8
)

# Set title with bold font
plt.title('Marital Status of Customers ', fontweight='bold')

# Set x-axis label with bold font
plt.xlabel('Marital Status', fontweight='bold')

# Set y-axis label with bold font
plt.ylabel('Gender', fontweight='bold')

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




[167]:
```
plt.figure(figsize = (12,8))
ax = sns.histplot(
data=df,
x="Product_Category", hue="Gender",
multiple="fill", stat="proportion",
discrete=True, shrink=.8
)
plt.xticks(rotation = 75)
plt.title('Product Category Of Order ')
plt.xlabel('Product Category')
plt.ylabel('Gender')
```

[167]:
```
Text(0, 0.5, 'Gender')
```

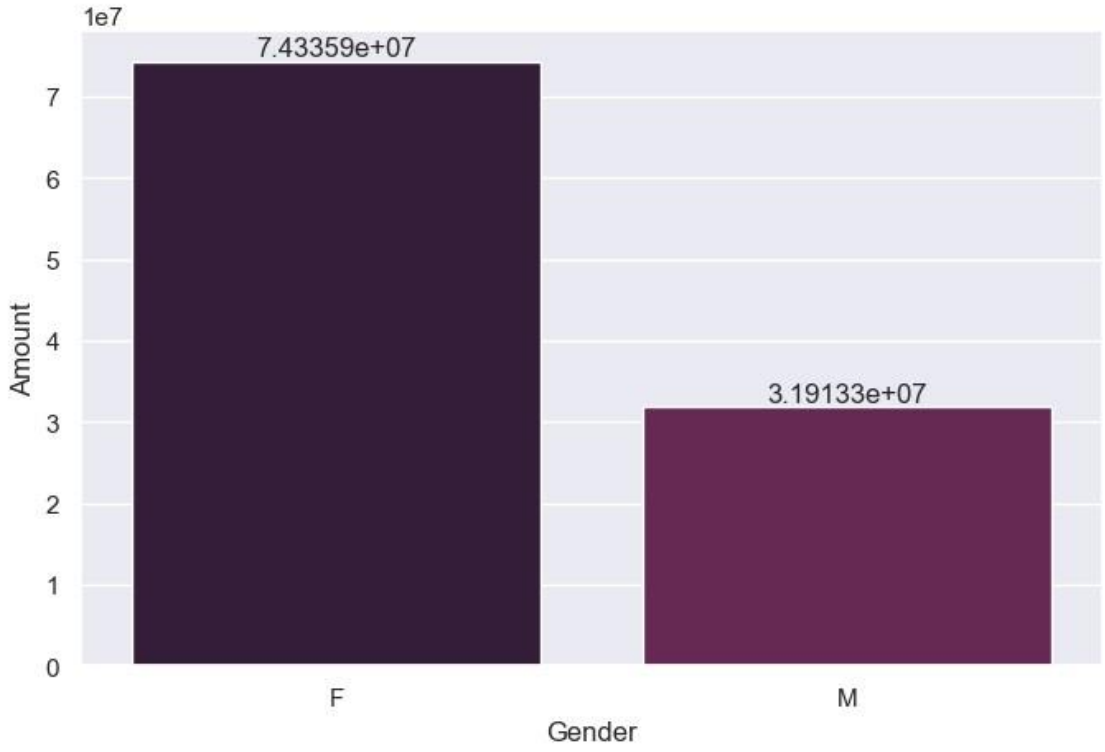



[168]:

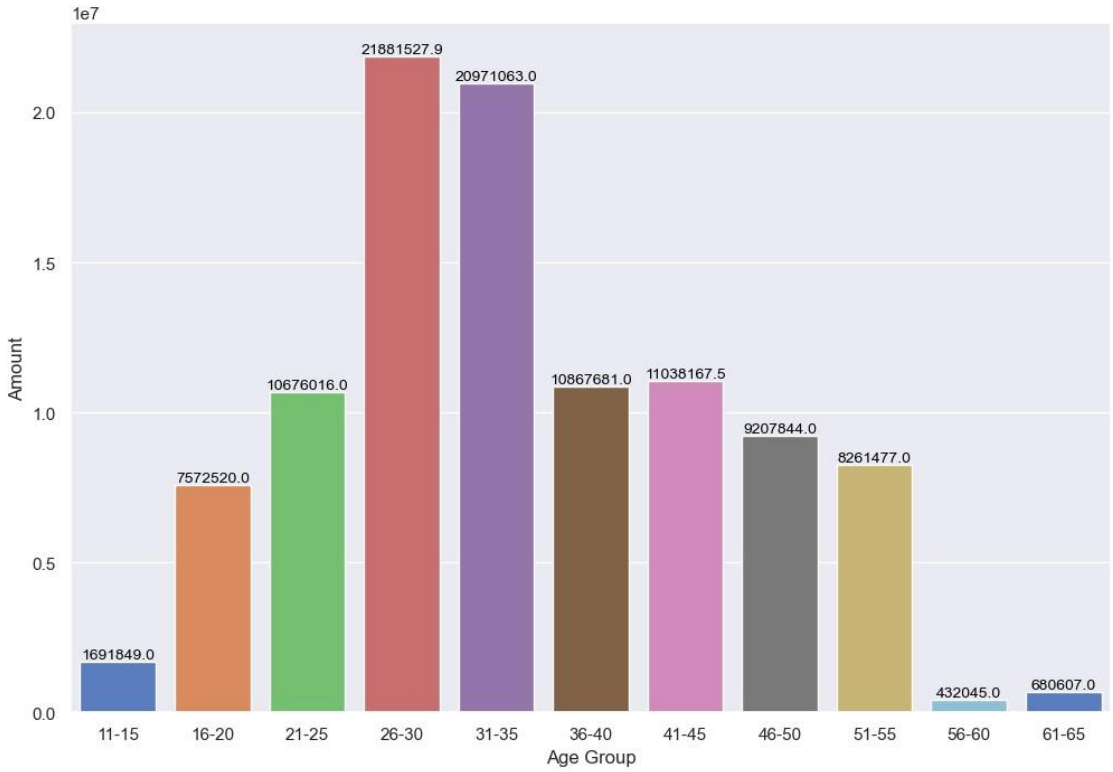
num

[168]: Index(['User_ID', 'Age', 'Marital_Status', 'Orders', 'Amount'], dtype='object')

[169]:
```
# Set the figure size
plt.figure(figsize=(8, 5))

# Choose a different palette
pal = sns.color_palette('rocket')

# Create the grouped data
data = df.groupby('Gender', as_index=False)['Amount'].sum().
  sort_values(by='Amount', ascending=False)

# Create the bar plot with the chosen palette
ax = sns.barplot(x='Gender',  ='Amount', data=data, palette=pal)

# Add annotations to the bars
for bar in ax.containers:
    ax.bar_label(bar)

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

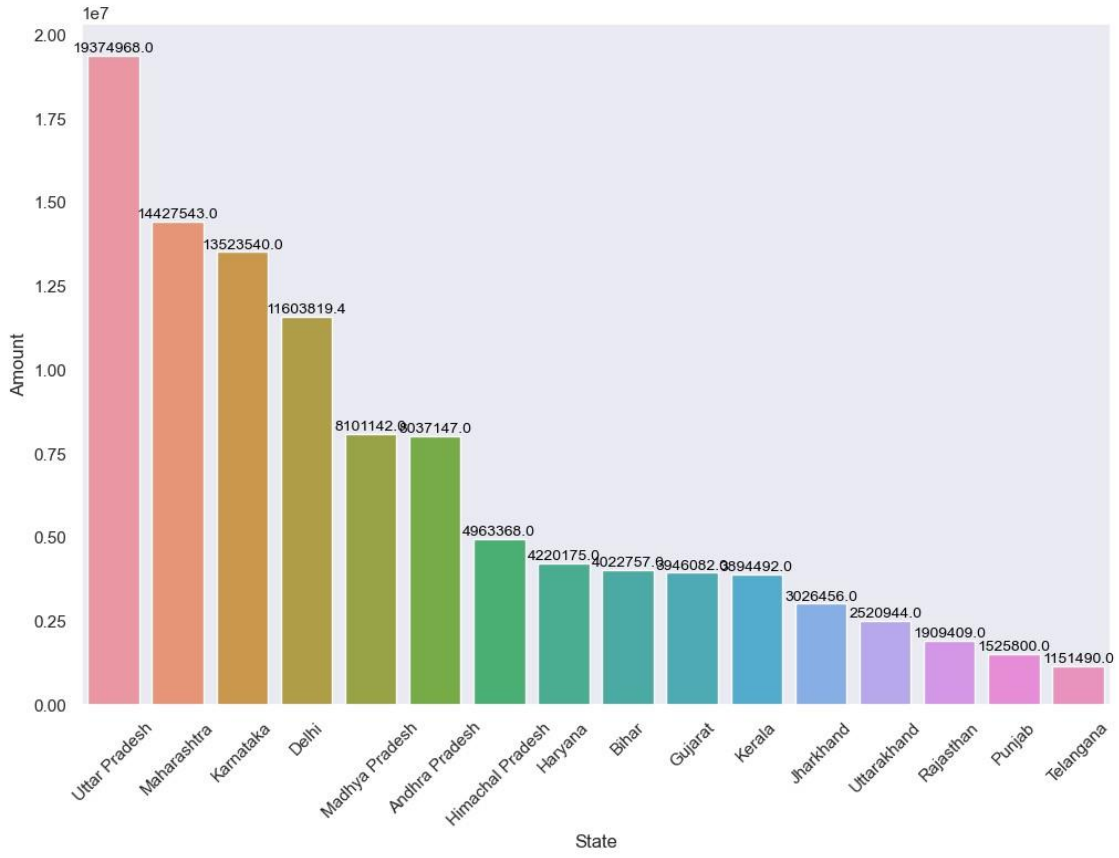


[170]:
```
# Set the figure size
plt.figure(figsize=(12, 8))

# Choose a different palette
pal = sns.color_palette('muted')

# Create the grouped data
data = df.groupby('Age Group', as_index=False)['Amount'].sum().
  sort_values(by='Amount', ascending=False)

# Create the bar plot with the chosen palette
ax = sns.barplot(x='Age Group',  ='Amount', data=data, palette=pal)

# Add annotations to the bars
for bar in ax.patches:
    ax.annotate('{:.1f}'.format(bar.get_height()),
                (bar.get_x() + bar.get_width() / 2, bar.get_height()),
                ha='center', va='bottom', fontsize=10, color='black')

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



[171]:
```
# Set the figure size
plt.figure(figsize=(12, 8))

# Set the Seaborn style and color codes
sns.set(style='dark', color_codes=True)

# Create the grouped data
data = df.groupby('State', as_index=False)['Amount'].sum().
  sort_values(by='Amount', ascending=False)

# Create the bar plot
ax = sns.barplot(x='State',  ='Amount', data=data)

# Add annotations to the bars
for bar in ax.patches:
    ax.annotate('{:.1f}'.format(bar.get_height()),
```

```
                (bar.get_x() + bar.get_width()/2, bar.get_height()),
                ha='center', va='bottom', fontsize=10, color='black')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

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




## 2 Interactive Graphs

[172]:
```
data = df.groupby('State', as_index=False)['Amount'].sum().
    sort_values(by='Amount', ascending=False)

# Define a custom color palette
colors =
px.colors.qualitative.Dark24
```

```
fig = px.bar(data, x='State', y='Amount', title='Total Amount by
 State',  color_discrete_sequence=colors)
```

```
fig.update_traces(marker_line_color='rgb(8,48,107)', marker_line_width=1.5,␣
 ↪opacity=0.6)
fig.update_layout(xaxis_tickangle=-45, xaxis_title='State',␣
 ↪yaxis_title='Amount')
fig.show()
```

[173]:
```
data = df.groupby('Product_Category', as_index=False)['Amount'].sum().
 ↪sort_values(by='Amount', ascending=False)

# Create the interactive bar plot using Plotly Express
fig = px.bar(data, x='Product_Category', y='Amount', title='Total Amount by␣
 ↪Product Category',
             color='Product_Category', color_discrete_sequence=px.colors.
 ↪qualitative.Set3)

# Update the layout for better readability
fig.update_layout(xaxis_tickangle=-45, xaxis_title='Product Category',␣
 ↪yaxis_title='Amount')

# Show the plot
fig.show()
```

[174]:
```
import plotly.express as px
data = df.groupby('Occupation', as_index=False)['Amount'].sum().
 ↪sort_values(by='Amount', ascending=False)

# Create the interactive bar plot using Plotly Express
fig = px.bar(data, x='Occupation', y='Amount', title='Total Amount by␣
 ↪Occupation',
             color='Occupation', color_continuous_scale='magma')

# Update the layout for better readability
fig.update_layout(xaxis_tickangle=-45, xaxis_title='Occupation',␣
 ↪yaxis_title='Amount')

# Show the plot
fig.show()
```

[176]:
```
data = df.groupby('Zone', as_index=False)['Amount'].sum().
 ↪sort_values(by='Amount', ascending=False)

# Define a custom color palette
colors = ['blue', 'green', 'orange', 'red', 'purple', 'pink', 'cyan']

# Create the interactive bar plot using Plotly Express with the custom color␣
 ↪palette
```

```
fig = px.bar(data, x='Zone', y='Amount', title='Total Amount by Zone',
             color='Zone', color_discrete_sequence=colors)

# Update the layout for better readability
fig.update_layout(xaxis_tickangle=-45, xaxis_title='Zone', yaxis_title='Amount')

# Show the plot
fig.show()
```

[179]:
```
# Create the interactive scatter plot using Plotly Express
fig = px.scatter(df, x='Age', y='Amount', color='Gender', title='Scatter Plot␣
 ↪by Age and Amount',
                 labels={'Age': 'Age', 'Amount': 'Amount', 'Gender': 'Gender'})

# Update the layout for better readability
fig.update_layout(xaxis_title='Age', yaxis_title='Amount')

# Show the plot
fig.show()
```

[180]:
```
# Create the interactive scatter plot using Plotly Express
fig = px.scatter(df, x='Orders', y='Age', color='Gender', title='Scatter Plot␣
 ↪by Orders and Age',
                 labels={'Orders': 'Orders', 'Age': 'Age', 'Gender': 'Gender'})

# Update the layout for better readability
fig.update_layout(xaxis_title='Orders', yaxis_title='Age')

# Show the plot
fig.show()
```

[ ]:
```

```

[ ]: