Diwali Sales Data Analysis Project

Exploratory Data Analysis (EDA) and Insights

By : Parshwa Jain

❖ Introduction

❑ Objective :

The objective of this project is to conduct an in-depth analysis of Diwali sales data to uncover key insights into consumer behavior, regional trends, and product preferences during this festive period.

❑ Importance :

Diwali, the festival of lights, marks a significant period of increased consumer spending across various product categories. Understanding the dynamics of consumer behavior during this festive season is crucial for businesses to optimize their marketing strategies and offerings.

❑ Dataset Overview :

- **Dataset:** Diwali_Sales_Data.csv
- **Source:** GitHub Link

Dataset Overview

[4]:
```
df.head()
```

[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

[3]:
```
df.shape
```

[3]: (11251, 15)

- This is the Dataset before the Data cleaning.
- It contains total 11251 rows and 15 columns.

```
[5]: 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
```

- This is the information about dataset before the data cleaning.
- It displays column name, Non-Null values in our data and Data-Type for each column.

```
[189]: #it will give the mathematical description for the specific columns that are mentioned

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

[189]:

	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

```
[190]: #to genrate all the column name in the dataset to use it in the function

df.columns
```

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

- It will give the mathematical description for the specific columns that are mentioned.

Data Cleaning

Prepare the dataset for analysis by ensuring data quality.

```
[183]: #this will remove the unwanted columns from the dataset. and  because of inplace it will reflect in our original dataset.

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

- This will drop unrelated/blank/unwanted columns from the dataset.

```
[7]: #checking if there is any null value means is there any empty cell in the dataset for any column

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

```
[7]: 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
```

[186]:
```python
#this will drope all rows which has null value means it has any empty cell from the dataset

df.dropna(inplace = True)
```

[187]:
```python
#confirming by shape function

df.shape
```

[187]:
```
(11239, 14)
```

- This will drop all rows which has null value means it has any empty cell from the dataset.

Exploratory Data Analysis (EDA) Overview

Explore relationships, patterns, and distributions in the dataset.

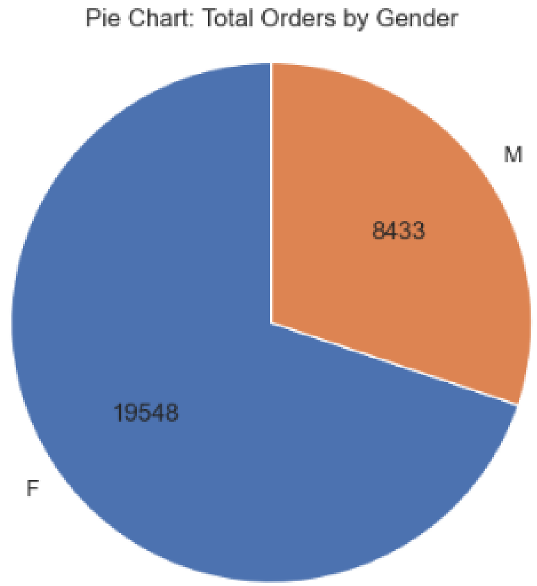
[239]:

```python
SalesByGen = df.groupby(['Gender'], as_index = False)['Orders'].sum().sort_values(by = 'Orders', ascending = False)

#sns.set(rc={'figure.figsize':(7,5)})
#OrdByGen = sns.barplot(x = 'Gender', y = 'Orders', data = SalesByGen, hue = 'Gender')
#plt.title('Total Orders by Gender')

#for bars in OrdByGen.containers:
 #   OrdByGen.bar_label(bars)

plt.figure(figsize=(5, 5))
plt.pie(SalesByGen['Orders'], labels=SalesByGen['Gender'], autopct=lambda p: f'{p * sum(SalesByGen['Orders']) / 100:.0f}', startangle=90)
plt.title('Pie Chart: Total Orders by Gender')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```



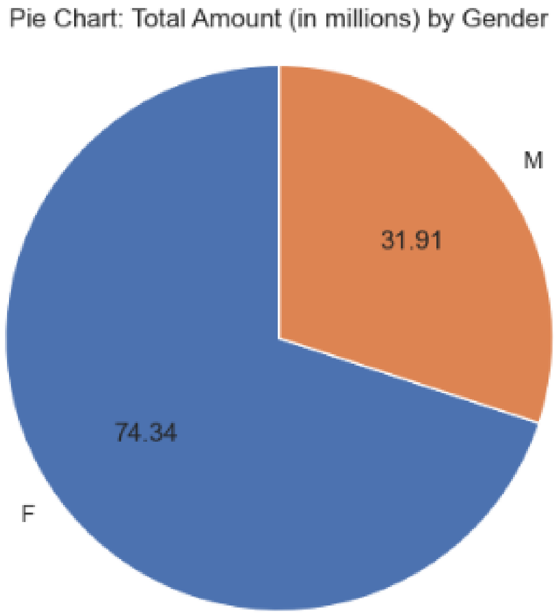
Total Orders by Gender

[240]:

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

SalesByGen['Amount'] = SalesByGen['Amount'] / 1e6

plt.figure(figsize=(5, 5))
plt.pie(SalesByGen['Amount'], labels=SalesByGen['Gender'], autopct=lambda p: f'{p * sum(SalesByGen['Amount']) / 100:.2f}', startangle=90)
plt.title('Pie Chart: Total Amount (in millions) by Gender')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```



Total Amount (in millions) by Gender

➢ Total Orders By Gender

INSIGHTS :

From the above graphs, it is clear that females constitute the majority of buyers. Moreover, both the number of orders and the purchasing power of females are higher compared to males.

[194]:
```
#this will sum all the Orders("['Orders'].sum()") based on gender(".groupby(['Gender'], as_index = False)")

df.groupby(['Gender'], as_index = False)['Orders'].sum().sort_values(by = 'Orders', ascending = False)
```

[194]:

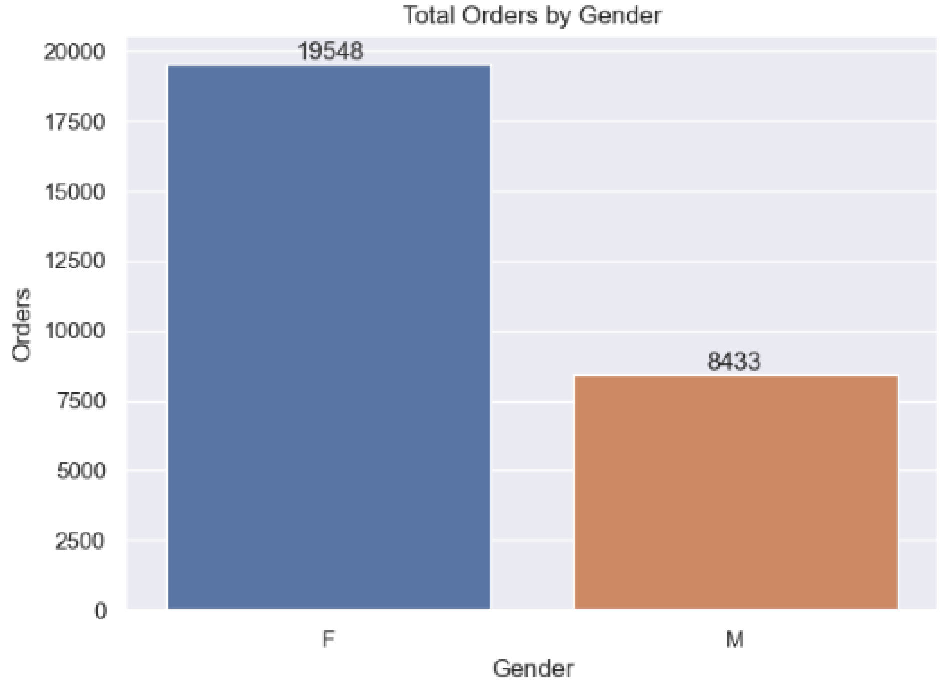
	Gender	Orders
0	F	19548
1	M	8433

[195]:
```
#genrating barplot to show the total number of order by gender
#genrating barplot based on data that we have from above.
#("hue = 'Gender',legend=False, palette='viridis'") this is just to use different color in bars.(optional)
#other process are just for displaying data label(optional).

SalesByGen = df.groupby(['Gender'], as_index = False)['Orders'].sum().sort_values(by = 'Orders', ascending = False)

sns.set(rc={'figure.figsize':(7,5)})
OrdByGen = sns.barplot(x = 'Gender', y = 'Orders', data = SalesByGen, hue = 'Gender')
plt.title('Total Orders by Gender')

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



➢ Total Orders By Age - Group

[281]:
```
#this will sum all the Orders("['Orders'].sum()") based on Age-Group(".groupby(['Age Group'], as_index = False)")

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

[281]:

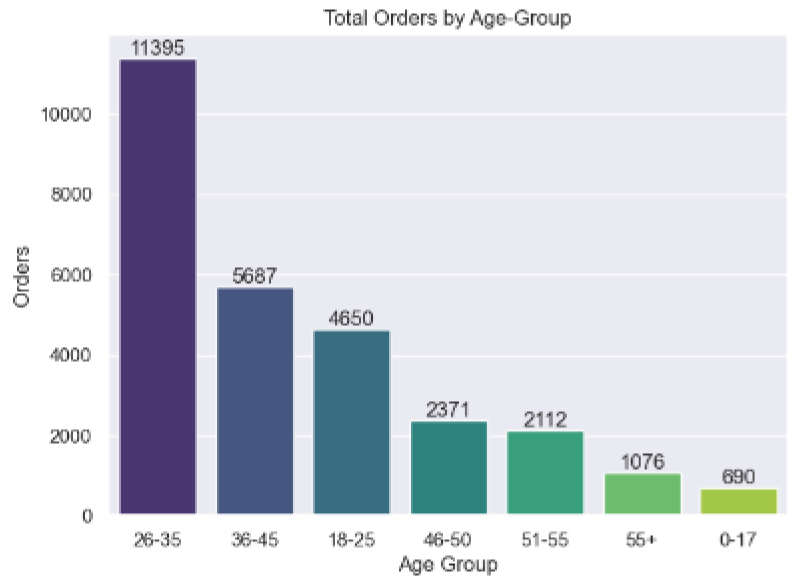
	Age Group	Orders
2	26-35	11395
3	36-45	5687
1	18-25	4650
4	46-50	2371
5	51-55	2112
6	55+	1076
0	0-17	690

[282]:
```
#genrating barplot to show the total number of order by Age-Group
#genrating barplot based on data that we have from above.
#("hue = 'Gender',legend=False, palette='viridis'") this is just to use different color in bars.(optional)
#other process are just for displaying data label(optional).

SalesByAge = df.groupby(['Age Group'], as_index = False)['Orders'].sum().sort_values(by = 'Orders', ascending = False)

sns.set(rc={'figure.figsize':(7,5)})
OrdByAge = sns.barplot(x = 'Age Group', y = 'Orders', data = SalesByAge, hue = 'Age Group',legend=False, palette='viridis')
plt.title('Total Orders by Age-Group')


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



INSIGHTS :

From the above graphs, it is evident that most buyers are females belonging to the age group of 26-35 years. This age group not only dominates in terms of the number of orders but also exhibits higher purchasing power compared to other age groups.

➢ Total Orders By Top 10 States

INSIGHTS :

From the above graphs, it is evident that most buyers come from the top three states:

- *Uttar Pradesh*
- *Maharashtra*
- *Karnataka.*

These states also exhibit the highest number of orders and total purchase amounts in that order.

However, there are notable exceptions when examining specific states:

1. *Haryana has fewer orders compared to Kerala, yet its total purchase amount surpasses that of Kerala.*
2. *Kerala's total purchase amount is significantly lower than Bihar and Gujarat.*

[288]:
```
#this will sum all the orders("['Orders'].sum()") based on State(".groupby(['State'], as_index = False)")
#it will display only top 10 values beacause of (".head(10)")

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

[288]:

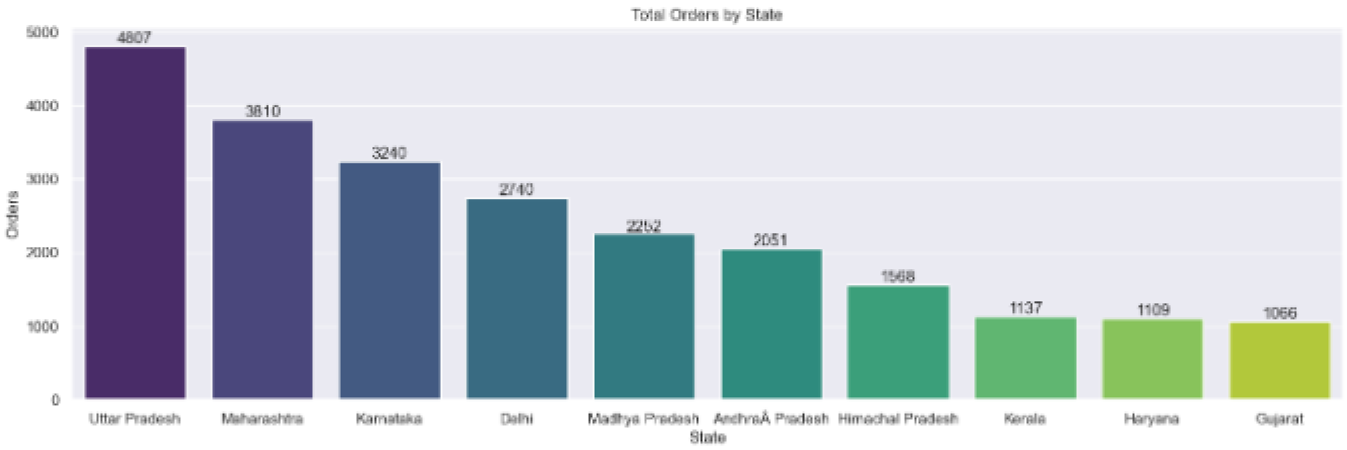
	State	Orders
14	Uttar Pradesh	4807
10	Maharashtra	3810
7	Karnataka	3240
2	Delhi	2740
9	Madhya Pradesh	2252
0	AndhraÂ Pradesh	2051
5	Himachal Pradesh	1568
8	Kerala	1137
4	Haryana	1109
3	Gujarat	1066

[289]:
```
#genrating barplot to show the total number of order by State
#genrating barplot based on data that we have from above.
#("hue = 'State',legend=False, palette='viridis'") this is just to use different color in bars.(optional)
#other process are just displaying data label(optional).

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

sns.set(rc={'figure.figsize':(17,5)})
OrdByState = sns.barplot(x = 'State', y = 'Orders', data = SalesByState, hue = 'State',legend=False, palette='viridis')
plt.title('Total Orders by State')

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



➢ Total Orders By Marital Status

INSIGHTS :

From the above graphs, we can see that most of the buyers are unmarried females. Additionally, both the number of orders and the purchasing power are higher for unmarried individuals compared to married individuals.

[215]:
```
#this will sum all the orders("['Orders'].sum()") based on Marital-Status(".groupby(['Marital_Status'], as_index = False)")

df.groupby(['Marital_Status'], as_index = False)['Orders'].sum().sort_values(by = 'Orders', ascending = False)
```

[215]:

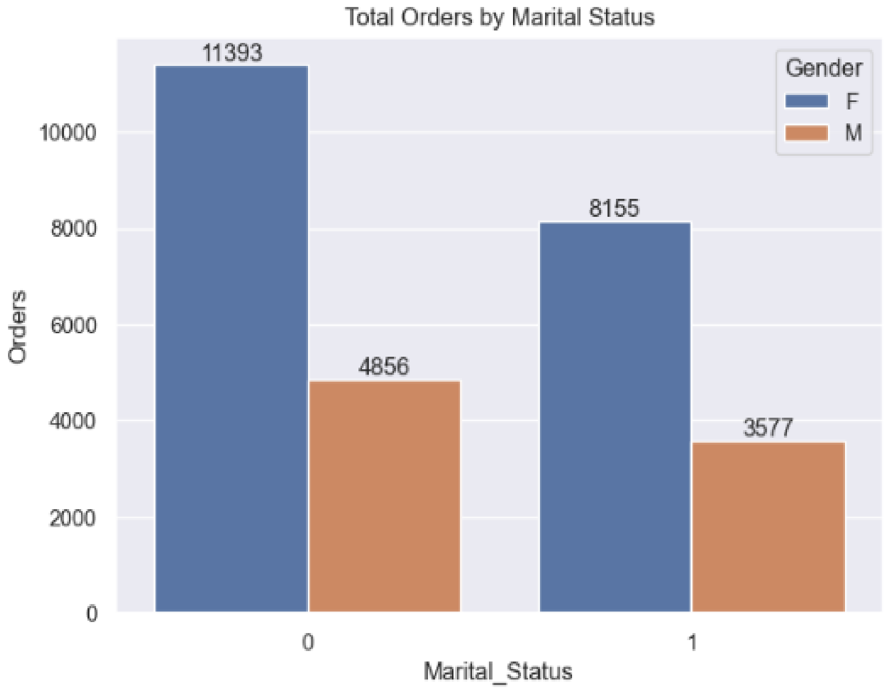
	Marital_Status	Orders
0	0	16249
1	1	11732

[216]:
```
#genrating barplot to show the total number of order by Marital Status
#genrating barplot based on data that we have from above.
#("hue = 'Marital_Status',legend=False, palette='viridis'") this is just to use different color in bars.(optional)
#other process are just for displaying data label(optional).

sns.set(rc={'figure.figsize':(7,5)})
SalesByMarital = df.groupby(['Marital_Status','Gender'], as_index = False)['Orders'].sum().sort_values(by = 'Orders', ascending = False)

OrdByMarital = sns.barplot(x = 'Marital_Status', y = 'Orders', data = SalesByMarital, hue = 'Gender')
plt.title('Total Orders by Marital Status')

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



➢ Total Orders By Top 10 Occupation

INSIGHTS :

From the above graphs, we can see that most of the buyers are from the following top 3 occupation fields:

- *IT Sector*
- *Healthcare*
- *Aviation*

These occupation fields also have the highest number of orders and purchasing power compared to other occupation fields. Additionally, there are certain differences within particular occupation fields where:

- *The number of orders is lower, but the total amount spent is higher. Conversely, the number of orders is higher, but the total amount spent is lower.*

- *This indicates varying purchasing behaviors and spending capacities across different occupation fields.*

[222]:
```
#this will sum all the orders("['Orders'].sum()") based on Occupation(".groupby(['Occupation'], as_index = False)")

df.groupby(['Occupation'], as_index = False)['Orders'].sum().sort_values(by = 'Orders', ascending = False)
```

[222]:

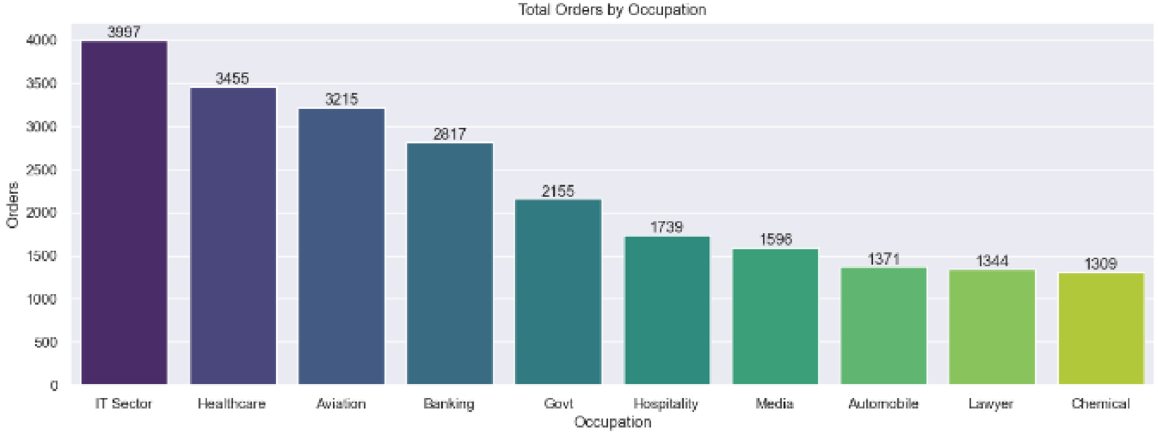
	Occupation	Orders
10	IT Sector	3997
8	Healthcare	3455
2	Aviation	3215
3	Banking	2817
7	Govt	2155
9	Hospitality	1739
12	Media	1596
1	Automobile	1371
11	Lawyer	1344
4	Chemical	1309
13	Retail	1270
6	Food Processing	1073
5	Construction	1025
14	Textile	893
0	Agriculture	722

[223]:
```
#genrating barplot to show the total number of order by Occupation
#genrating barplot based on data that we have from above.
#("hue = 'Occupation',legend=False, palette='viridis'") this is just to use different color in bars.(optional)
#other process are just for displaying data label(optional).

sns.set(rc={'figure.figsize':(15,5)})
SalesByOcp = df.groupby(['Occupation'], as_index = False)['Orders'].sum().sort_values(by = 'Orders', ascending = False).head(10)

OrdByOcp = sns.barplot(x = 'Occupation', y = 'Orders', data = SalesByOcp, hue = 'Occupation',legend=False, palette='viridis')
plt.title('Total Orders by Occupation')

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



➢ Total Orders By Top 10 Category

INSIGHTS :

From the graphs, we can see that most of the buyers are females, and they primarily order from the following top 3 product categories:

- *Clothing & Apparel*
- *Electronics & Gadgets*
- *Food*

However, the number of orders is highest in the following product categories:

- *Clothing & Apparel*
- *Food*
- *Electronics & Gadgets*

And the total amount spent is highest in the following product categories:

- *Food*
- *Clothing & Apparel*
- *Electronics & Gadgets*

[230]:

```python
#this will sum all the orders("['Orders'].sum()") based on Product Category(".groupby(['Product_Category'], as_index = False)")

df.groupby(['Product_Category'], as_index = False)['Orders'].sum().sort_values(by = 'Orders', ascending = False).head(10)
```

[230]:

	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

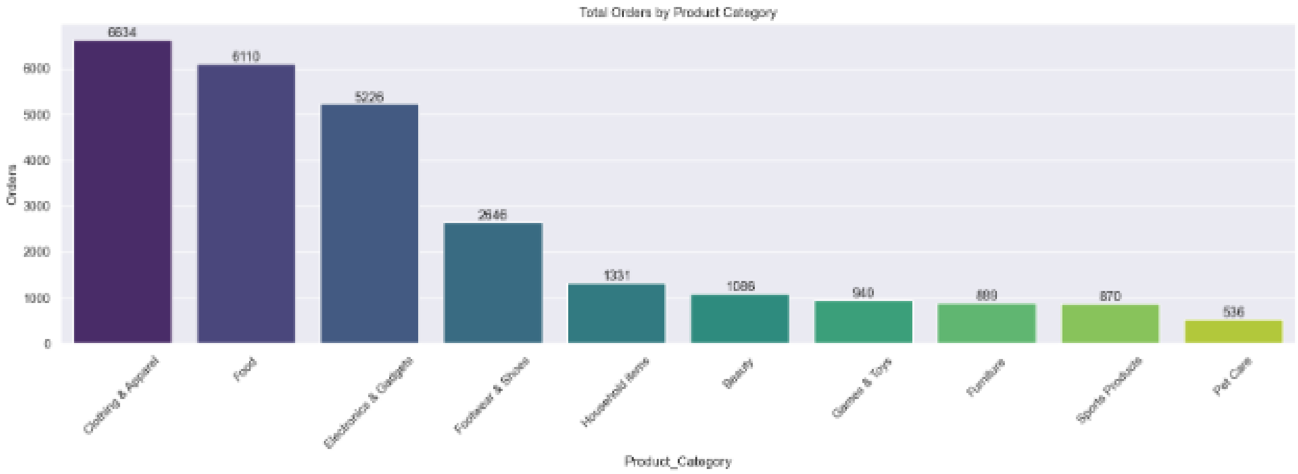
[231]:

```python
#genrating barplot to show the total number of order by Product_Category
#genrating barplot based on data that we have from above.
#("hue = 'Product_Category',legend=False, palette='viridis'") this is just to use different color in bars.(optional)
#other process are just for displaying data label(optional).

sns.set(rc={'figure.figsize':(20,5)})
SalesByPdc = df.groupby(['Product_Category'], as_index = False)['Orders'].sum().sort_values(by = 'Orders', ascending = False).head(10)

OrdByPdc = sns.barplot(x = 'Product_Category', y = 'Orders', data = SalesByPdc, hue = 'Product_Category',legend=False, palette='viridis')
plt.title('Total Orders by Product Category')
plt.xticks(rotation=45)

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



```
[236]: # Calculate AOV (Average Order Value) and add it as a new column
df['AOV'] = df['Amount'] / df['Orders']

# Save the updated data back to the same CSV file, overwriting the existing data
df.to_csv('Diwali_Sales_Data.csv', index=False)
```

Creating new Column for "Average Order Value (AOV)" using python in our dataset

[237]:

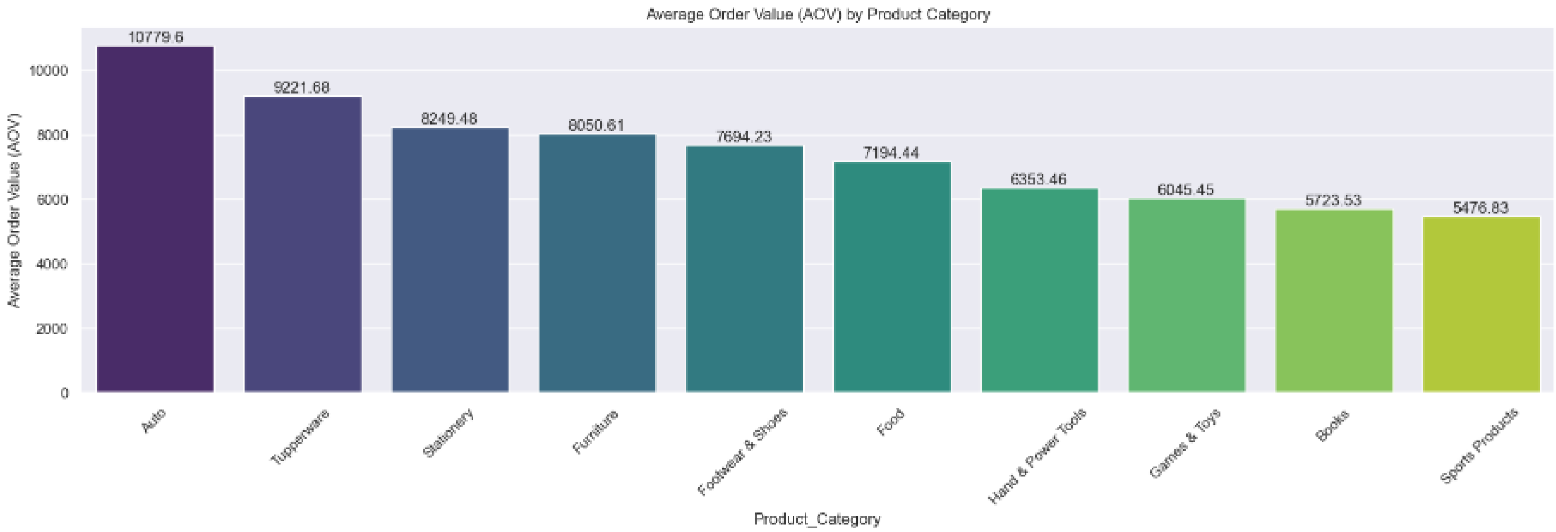
```
#Compare the AOV across different product categories.

CatByAOV = df.groupby(['Product_Category'], as_index = False)['AOV'].mean().sort_values(by = 'AOV', ascending = False).head(10)

#CatByAOV['AOV'] = CatByAOV['AOV'] / 1e6

sns.set(rc={'figure.figsize':(20,5)})
AOVByPdc = sns.barplot(x = 'Product_Category', y = 'AOV', data = CatByAOV, hue = 'Product_Category',legend=False, palette='viridis')
plt.title('Average Order Value (AOV) by Product Category')
plt.ylabel('Average Order Value (AOV)')
plt.xticks(rotation=45)

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



Compare the Average Order Value (AOV) across different product categories.

❖ Conclusion

1) **Gender Analysis:**
    - Females dominate as buyers, with higher numbers of orders and greater purchasing power compared to males.

2) **Age Group Analysis:**
    - The age group of 26-35 years shows the highest number of orders and significant purchasing power, indicating strong consumer activity during Diwali.

3) **State-wise Analysis:**
    - Uttar Pradesh, Maharashtra, and Karnataka are the top states in terms of both number of orders and total purchase amounts.
    - Notable exceptions include Haryana, which despite fewer orders, exhibits higher total purchase amounts compared to Kerala.

❖ Conclusion

4) **Marital Status Analysis:**
   - Unmarried individuals, especially females, account for the majority of buyers, with higher orders and purchasing power than married individuals.

5) **Occupation Analysis:**
   - The IT sector, Healthcare, and Aviation are the primary occupation fields with the highest number of orders and significant purchasing power.
   - Variances within occupation fields highlight differing spending behaviors and capacities.

6) **Product Category Analysis:**
   - Clothing & Apparel, Electronics & Gadgets, and Food are the top product categories in terms of buyer preference.
   - While Clothing & Apparel and Food lead in number of orders, Food surpasses in total purchase amounts.

THANK YOU !