

Diwali Sales Analysis Using Python

Data Analysis or sometimes referred to as exploratory data analysis (EDA) is one of the core components of data science. It is also the part on the majority of the time which makes it extremely important in the field of data science. This repository demonstrates Exploratory Data Analysis methods and techniques using Python. The purpose of the used Diwali Sales dataset has been taken from Kaggle since it is one of the ideal dataset for performing EDA and taking a step towards the most amazing and interesting field of data science. Good luck with your EDA on the used Diwali Sales dataset.

Project Description :-

- Performed Data Cleaning and Data Manipulation.
- Performed Exploratory Data Analysis (EDA) using Pandas, NumPy, Matplotlib, Seaborn Libraries.
- Improved Customer experience by identifying potential customers across different states, occupation, gender and age groups.
- Improved sales by identifying most selling product categories and products, which can help to plan inventory and hence meet the demands.

Conclusion :-

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


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

# for mathematical use
# for DataFrame tables
# for visualizing data
# for Visualization

In [43]: # pandas read_csv
df = pd.read_csv(r'E:\New Folder\Data Science\YOUTUBE\PROJECTS\python\Python_Diwal1_Sales_Analysis-main\Diwal1 Sales Data.csv', encoding= 'unicode_escape')

In [43]: # load file
df.shape
Out[43]: (11251, 15)

In [43]: # showing top 5 rows
df.head(5)
```

```
df = pd.read_csv('E:\New folder\data once\YOUTUBE\PROJECTS\python\Python_Dwaili_Sales_Analysis-main\dwaili_Sales_Data.csv', encoding='unicode_escape')

In [41]: # load file
df.shape
Out[41]: (11251, 15)

In [42]: # showing top 5 Rows
df.head()
```

User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	State	Zone	Occupation	Product_Category	Orders	Amount	Status	unamed1	
0	1002003	Sanskriti	P0012942	F	26-35	28	0	Maharashtra	Western	Healthcare	Auto	1	23952.0	NaN	NaN
1	1000732	Karika	P00110942	F	26-35	35	1	Andhra Pradesh	Southern	Govt	Auto	3	23924.0	NaN	NaN
2	1001990	Brinda	P00118942	F	26-35	35	1	Uttar Pradesh	Central	Automobile	Auto	3	23924.0	NaN	NaN
3	1001425	Sudeep	P00237842	M	0-17	16	0	Karnataka	Southern	Construction	Auto	2	23912.0	NaN	NaN
4	1000588	Joan	P00057942	M	26-35	28	1	Gujarat	Western	Food Processing	Auto	2	23877.0	NaN	NaN

```
In [43]: df.head(10)
```

User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	State	Zone	Occupation	Product_Category	Orders	Amount	Status	unamed1	
0	1002003	Sanskriti	P0012942	F	26-35	28	0	Maharashtra	Western	Healthcare	Auto	1	23952.0	NaN	NaN
1	1000732	Karika	P00110942	F	26-35	35	1	Andhra Pradesh	Southern	Govt	Auto	3	23924.0	NaN	NaN
2	1001990	Brinda	P00118942	F	26-35	35	1	Uttar Pradesh	Central	Automobile	Auto	3	23924.0	NaN	NaN
3	1001425	Sudeep	P00237842	M	0-17	16	0	Karnataka	Southern	Construction	Auto	2	23912.0	NaN	NaN
4	1000588	Joan	P00057942	M	26-35	28	1	Gujarat	Western	Food Processing	Auto	2	23877.0	NaN	NaN

5	1000588	Joan	P00057942	M	18-25	28	1	Hinacal Pradesh	Northern	Food Processing	Auto	1	23877.00	NaN	NaN
6	1001132	Bakul	P00010042	F	18-25	25	1	Uttar Pradesh	Central	Lawyer	Auto	4	23811.00	NaN	NaN
7	1002002	Shivangi	P00273442	F	55+	61	0	Maharashtra	Western	IT Sector	Auto	1	NaN	NaN	NaN
8	1003024	Kushal	P0025642	M	26-35	35	0	Uttar Pradesh	Central	Govt	Auto	2	23809.00	NaN	NaN
9	1003950	Giriy	P00031142	F	26-35	28	1	Andhra Pradesh	Southern	Media	Auto	4	23799.99	NaN	NaN

```
In [44]: # dataframe information detail

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  int64  
 5   Age                   11251 non-null  object  
 6   Marital_Status        11251 non-null  object  
 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                11250 non-null  float64 
13   Status                0 non-null      float64 
14   unnamed               0 non-null      float64 
dtypes: float64(3), int64(4), object(8)
memory usage: 1.3+ MB
```

```
In [44]: # dataframe information detail
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  int64
 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                6 non-null      float64
13  Status                6 non-null      float64
14  unamed1              6 non-null      float64
dtypes: float64(13), int64(4), object(8)
memory usage: 1.3+ MB
```

```
In [44]: # drop unrelated/blank columns
df.drop(['Status', 'unamed1'], axis=1, inplace=True)
```

```
In [44]: pd.isnull(df)
```

User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	State	Zone	Occupation	Product_Category	Orders	Amount	Status	Amount
0	False	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	False
2	False	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	False
4	False	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	False
11247	False	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	False
11249	False	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	False

```
In [47]: #check for null values
pd.isnull(df).sum()
```

User_ID	0
Cust_name	0
Product_ID	0
Gender	0
Age Group	0
Age	0
Marital_Status	0
State	0
Zone	0
Occupation	0
Product_Category	0
Orders	12
Amount	6
dtype:	int64

```
In [48]: df.shape
Out[48]: (11251, 13)
```

```
In [49]: #drop null values
df.dropna(inplace=True)
```

```
In [50]: df.shape
Out[50]: (11239, 13)
```

```
In [51]: # initialize list of lists
data_test = [['madhav', 11], ['Gopi', 15], ['Keshav'], ['Lalita', 16]]
# create the pandas DataFrame using list
df_test = pd.DataFrame(data_test, columns=['Name', 'Age'])
df_test
```

	Name	Age
0	madhav	11.0
1	Gopi	15.0
2	Keshav	NaN
3	Lalita	16.0

```
In [52]: df_test.dropna(inplace=True)
```

```
In [53]: df_test
```

	Name	Age
0	madhav	11.0
1	Gopi	15.0
3	Lalita	16.0

```
In [54]: df['Amount'] = df['Amount'].astype('int')
```

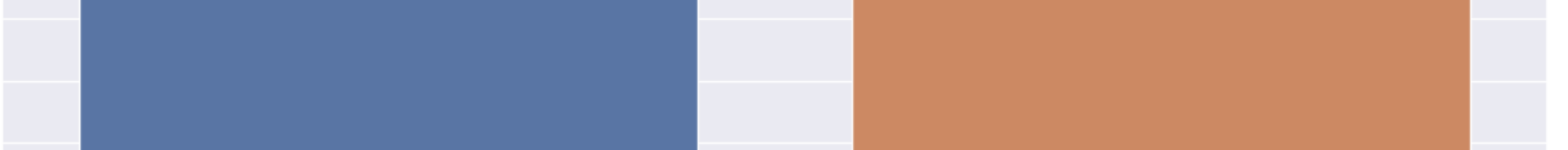
```
In [55]: df['Amount'].dtypes
Out[55]: dtype('int32')
```

```
In [56]: df.columns
```

```
Out[56]: Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age', 'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category', 'Orders', 'Amount'], dtype='object')
```

```
In [57]: # if we have to rename
df.rename(columns={'Marital_Status': 'Gshaadi'})
```

User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Shadi	State	Zone	Occupation	Product_Category	Orders	Amount	
0	1002003	Sanskriti	P0012942	F	26-35	28	0	Maharashtra	Western	Healthcare	Auto	1	23962
1	1000732	Karika	P00110942	F	26-35	35	1	Andhra Pradesh	Southern	Govt	Auto	3	23934
2	1001990	Brinda	P00118942	F	26-35	35	1	Uttar Pradesh	Central	Automobile	Auto	3	23924
3	1001425	Sudeep	P00237842	M	0-17	15	0	Karnataka	Southern	Construction	Auto	2	23912
4	1000588	Joan	P00057942	M	26-35	28	1	Gujarat	Western	Food Processing	Auto	2	23877
...
11246	1000695	Manning	P00269842	M	18-25	19	1	Maharashtra	Western	Chemical	Office	4	370
11247	1004089	Reichenbach	P00171342	M	26-35	33	0	Haryana	Northern	Healthcare	Veterinary	3	367
11248	1001209	Osheen	P00201342	F	35-45	40	0	Madhya Pradesh	Central	Textile	Office	4	213
11249	1004023	Noonan	P00059442	M	35-45	37	0	Karnataka	Southern	Agriculture	Office	3	206
11250	1002744	Brumley	P00381742	F	18-25	19	0	Maharashtra	Western	Healthcare	Office	3	188



Gender	Count
F	~3000
M	~3000

```
In [62]: ax = sns.countplot(x = 'Gender', data = df)
for bars in ax.containers:
    ax.bar_label(bars)
```

```
In [58]: # describe() method returns description of the data in the DataFrame (i.e. count, mean, std, etc)
df.describe()
```

	User_ID	Age	Marital_Status	Orders	Amount
count	1.123900e+04	11239.000000	11239.000000	11239.000000	11238.000000
mean	1.002004e+06	35.410357	0.420006	2.489624	9453.610553
std	1.71029e+03	12.753866	0.489389	1.114967	5222.355168
min	1.000000e+06	12.000000	0.000000	1.000000	188.000000
25%	1.001462e+06	27.000000	0.000000	2.000000	5443.000000
50%	1.003064e+06	33.000000	0.000000	2.000000	8109.000000
75%	1.004426e+06	43.000000	1.000000	3.000000	12675.000000
max	1.006040e+06	92.000000	1.000000	4.000000	23952.000000

```
In [59]: # use describe() for specific columns
df[['Age', 'Orders', 'Amount']].describe()
```

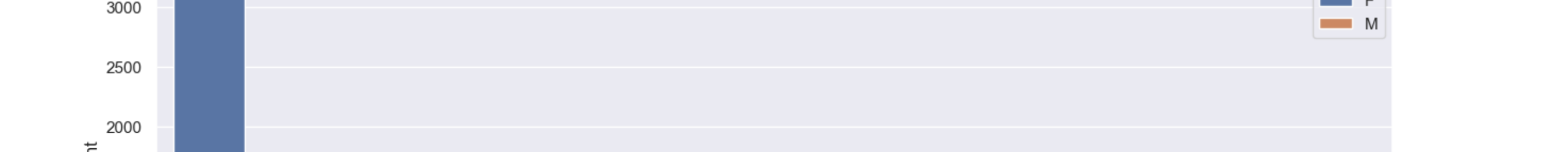
	Age	Orders	Amount
count	11239.000000	11239.000000	11239.000000
mean	35.410357	2.489624	9453.610553
std	12.753866	1.114967	5222.355168
min	12.000000	1.000000	188.000000
25%	27.000000	2.000000	5443.000000
50%	33.000000	2.000000	8109.000000
75%	43.000000	3.000000	12675.000000
max	92.000000	4.000000	23952.000000

Exploratory Data Analysis

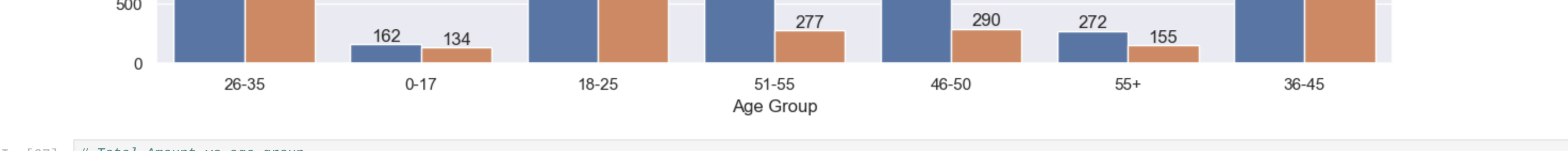
Gender

```
In [60]: df.columns
Out[60]: Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age', 'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category', 'Orders', 'Amount'], dtype='object')
```

```
In [61]: sns.countplot(x = 'Gender', data = df)
```



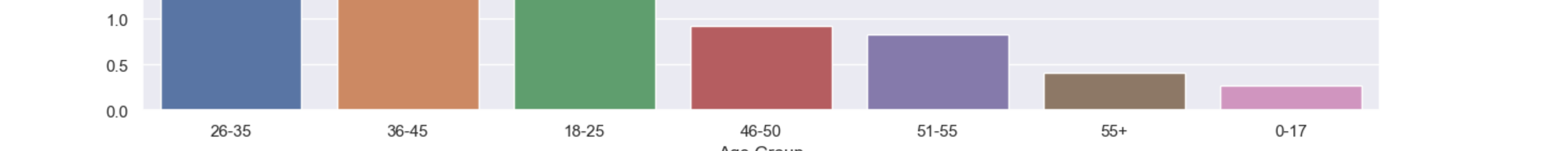
```
In [62]: ax = sns.countplot(x = 'Gender', data = df)
for bars in ax.containers:
    ax.bar_label(bars)
```



```
In [63]: df.groupby(['Gender'], as_index=False)['Amount'].sum().sort_values(by='Amount', ascending=False)
```

Gender	Amount
0	F 7433963
1	M 3391276

```
In [64]: 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)
```

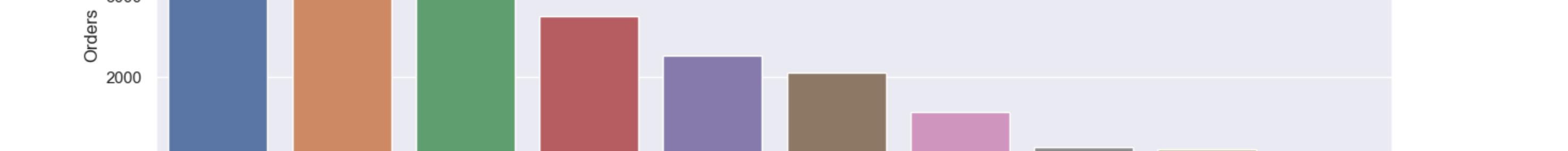


From above graphs we can see that most of the buyers are females and even the purchasing power of females are greater than men

Age

```
In [65]: df.columns
Out[65]: Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age', 'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category', 'Orders', 'Amount'], dtype='object')
```

```
In [66]: ax = sns.countplot(data = df, x = 'Age Group', hue = 'Gender')
for bars in ax.containers:
    ax.bar_label(bars)
```



```
In [67]: # 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)
```

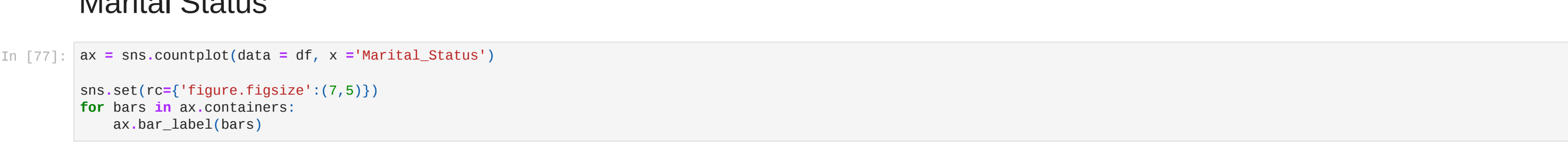


For above graphs we can see that most of the buyers are of age group between 26-35 years female

State

```
In [68]: df.columns
Out[68]: Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age', 'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category', 'Orders', 'Amount'], dtype='object')
```

```
In [69]: # 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')
```



```
In [70]: # 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')
```



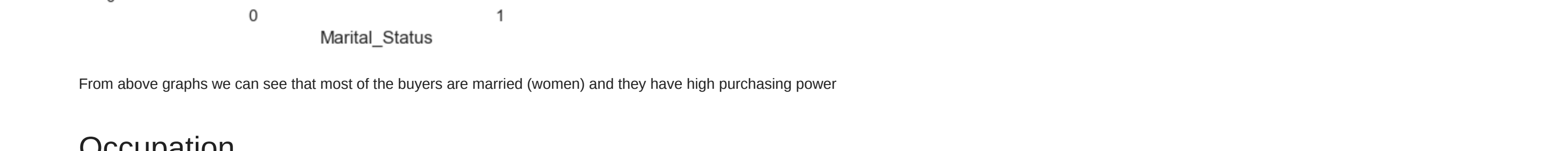
From above graphs we can see the most of the orders & total sales/amount are from Uttar Pradesh, Maharashtra and karnataka respectively

Marital status

```
In [77]: ax = sns.countplot(data = df, x = 'Marital_Status')
sns.set(rc={'figure.figsize':(28,5)})
ax = sns.countplot(data = df, x = 'Marital_Status')
for bars in ax.containers:
    ax.bar_label(bars)
```



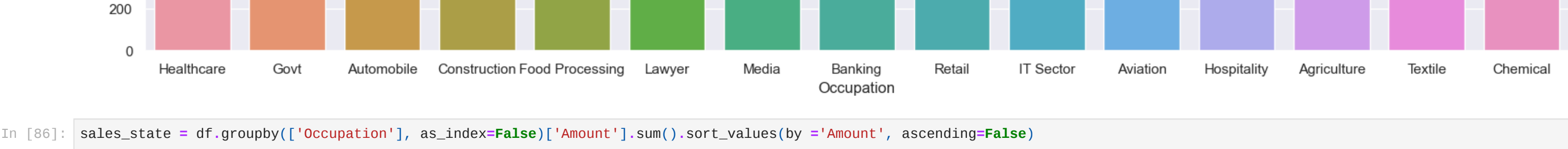
```
In [79]: sales_state = df.groupby(['Marital_Status', 'Gender'], as_index=False)['Amount'].sum().sort_values(by = 'Amount', ascending=False).head(10)
sns.set(rc={'figure.figsize':(28,5)})
sns.barplot(data = sales_state, x = 'Marital_Status', y = 'Amount', hue = 'Gender')
```



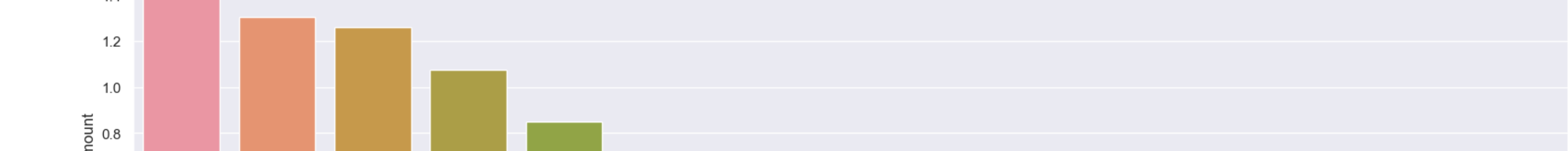
From above graphs we can see that most of the buyers are married (women) and they have high purchasing power

Occupation

```
In [85]: sns.set(rc={'figure.figsize':(28,5)})
ax = sns.countplot(data = df, x = 'Occupation')
for bars in ax.containers:
    ax.bar_label(bars)
```



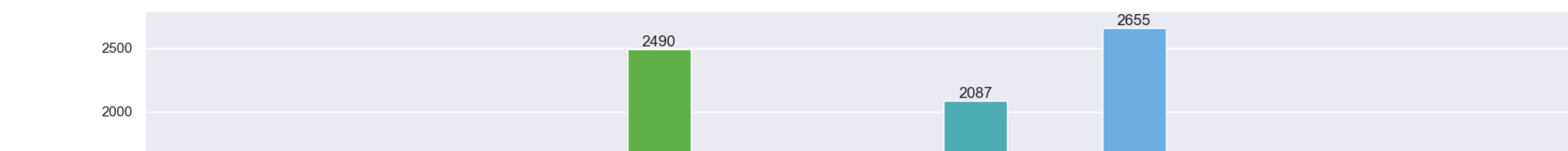
```
In [86]: sales_state = df.groupby(['Occupation'], as_index=False)['Amount'].sum().sort_values(by = 'Amount', ascending=False)
sns.set(rc={'figure.figsize':(28,5)})
sns.barplot(data = sales_state, x = 'Occupation', y = 'Amount')
```



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

Product category

```
In [81]: sns.set(rc={'figure.figsize':(28,5)})
ax = sns.countplot(data = df, x = 'Product_Category')
for bars in ax.containers:
    ax.bar_label(bars)
```



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

```
In [84]: sales_state = df.groupby(['Product_ID'], as_index=False)['Orders'].sum().sort_values(by = 'Orders', ascending=False).head(10)
sns.set(rc={'figure.figsize':(28,5)})
sns.barplot(data = sales_state, x = 'Product_ID', y = 'Orders')
```

