Diwali Sales Exploratory Data Analysis

This data is taken from kaggle.com for the purpose of project work, and it belongs to diwali sales in a country on a ecommerce site. The dataset contains 11k+ rows and 15 columns.

Importing Libraries

In [1]:
```
import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
```

Data Cleaning and Manipulating

In [3]:
```
df = pd.read_csv("C:\\Users\\Shivam\\Downloads\\Python_Diwali_Sales_Analysis\\Python_Diwali_Sales_Analysis\\Diwa
df
```

Out[3]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	State	Zone	Occupation	Product_Category	Ord
0	1002903	Sanskriti	P00125942	F	26-35	28	0	Maharashtra	Western	Healthcare	Auto	
1	1000732	Kartik	P00110942	F	26-35	35	1	Andhra Pradesh	Southern	Govt	Auto	
2	1001990	Bindu	P00118542	F	26-35	35	1	Uttar Pradesh	Central	Automobile	Auto	
3	1001425	Sudevi	P00237842	M	0-17	16	0	Karnataka	Southern	Construction	Auto	
4	1000588	Joni	P00057942	M	26-35	28	1	Gujarat	Western	Food Processing	Auto	
...	...	...	...	...	...	...	...	...	...	...	...	
11246	1000695	Manning	P00296942	M	18-25	19	1	Maharashtra	Western	Chemical	Office	
11247	1004089	Reichenbach	P00171342	M	26-35	33	0	Haryana	Northern	Healthcare	Veterinary	
11248	1001209	Oshin	P00201342	F	36-45	40	0	Madhya Pradesh	Central	Textile	Office	
11249	1004023	Noonan	P00059442	M	36-45	37	0	Karnataka	Southern	Agriculture	Office	
11250	1002744	Brumley	P00281742	F	18-25	19	0	Maharashtra	Western	Healthcare	Office	

11251 rows × 15 columns

In [4]:
```
df.shape
```

Out[4]:
```
(11251, 15)
```

In [5]:
```
df.head()
```

Out[5]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Marital_Status	State	Zone	Occupation	Product_Category	Orders	A
0	1002903	Sanskriti	P00125942	F	26-35	28	0	Maharashtra	Western	Healthcare	Auto	1	
1	1000732	Kartik	P00110942	F	26-35	35	1	Andhra Pradesh	Southern	Govt	Auto	3	
2	1001990	Bindu	P00118542	F	26-35	35	1	Uttar Pradesh	Central	Automobile	Auto	3	
3	1001425	Sudevi	P00237842	M	0-17	16	0	Karnataka	Southern	Construction	Auto	2	
4	1000588	Joni	P00057942	M	26-35	28	1	Gujarat	Western	Food Processing	Auto	2	

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11251 entries, 0 to 11250
Data columns (total 15 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   User_ID           11251 non-null  int64  
 1   Cust_name         11251 non-null  object 
 2   Product_ID        11251 non-null  object 
 3   Gender            11251 non-null  object 
 4   Age Group         11251 non-null  object 
 5   Age               11251 non-null  int64  
 6   Marital_Status    11251 non-null  int64  
 7   State             11251 non-null  object 
 8   Zone              11251 non-null  object 
 9   Occupation        11251 non-null  object 
 10  Product_Category  11251 non-null  object 
 11  Orders            11251 non-null  int64  
 12  Amount            11239 non-null  float64
 13  Status            0 non-null      float64
 14  unnamed1          0 non-null      float64
dtypes: float64(3), int64(4), object(8)
memory usage: 1.3+ MB
```

In [7]:
```
# dropping unused columns
df.drop(["Status", "unnamed1"], inplace = True, axis =1)
```

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11251 entries, 0 to 11250
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   User_ID           11251 non-null  int64  
 1   Cust_name         11251 non-null  object 
 2   Product_ID        11251 non-null  object 
 3   Gender            11251 non-null  object 
 4   Age Group         11251 non-null  object 
 5   Age               11251 non-null  int64  
 6   Marital_Status    11251 non-null  int64  
 7   State             11251 non-null  object 
 8   Zone              11251 non-null  object 
 9   Occupation        11251 non-null  object 
 10  Product_Category  11251 non-null  object 
 11  Orders            11251 non-null  int64  
 12  Amount            11239 non-null  float64
dtypes: float64(1), int64(4), object(8)
memory usage: 1.1+ MB
```

In [9]:
```
# gettings True for null values
pd.isnull(df)
```

Out[9]:

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

11251 rows × 13 columns

In [10]:
```
# getting sum of null value
pd.isnull(df).sum()
```

Out[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
```

In [11]:
```
df.shape
```

Out[11]:
```
(11251, 13)
```

In [12]:
```
# saving changes in original dataset
df.dropna(inplace = True)
```

In [13]:
```
df.shape
```

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

In [14]:
```
# changing data type of Amount Column
df["Amount"] = df["Amount"].astype(int)
```

In [15]:
```
df["Amount"].dtype
```

Out[15]:
```
dtype('int32')
```

In [16]:
```
df.columns
```

Out[16]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

In [17]:
```
# renaming a col but not save it as we not used inplace = True
df.rename(columns = {"Marital_Status": "Shaadi"})
```

Out[17]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Shaadi	State	Zone	Occupation	Product_Category	Orders	An
0	1002903	Sanskriti	P00125942	F	26-35	28	0	Maharashtra	Western	Healthcare	Auto	1	
1	1000732	Kartik	P00110942	F	26-35	35	1	Andhra Pradesh	Southern	Govt	Auto	3	
2	1001990	Bindu	P00118542	F	26-35	35	1	Uttar Pradesh	Central	Automobile	Auto	3	
3	1001425	Sudevi	P00237842	M	0-17	16	0	Karnataka	Southern	Construction	Auto	2	
4	1000588	Joni	P00057942	M	26-35	28	1	Gujarat	Western	Food Processing	Auto	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	
11246	1000695	Manning	P00296942	M	18-25	19	1	Maharashtra	Western	Chemical	Office	4	
11247	1004089	Reichenbach	P00171342	M	26-35	33	0	Haryana	Northern	Healthcare	Veterinary	3	
11248	1001209	Oshin	P00201342	F	36-45	40	0	Madhya Pradesh	Central	Textile	Office	4	
11249	1004023	Noonan	P00059442	M	36-45	37	0	Karnataka	Southern	Agriculture	Office	3	
11250	1002744	Brumley	P00281742	F	18-25	19	0	Maharashtra	Western	Healthcare	Office	3	

11239 rows × 13 columns

In [18]:
```
# describing about the dataframe like count, min, ,ax, mean, etc.
df.describe()
```

Out[18]:

	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	2.489634	9453.610553
std	1.716039e+03	12.753866	0.493589	1.114967	5222.355168
min	1.000001e+06	12.000000	0.000000	1.000000	188.000000
25%	1.001492e+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 [19]:
```
df[["Amount", "Age"]].describe()
```

Out[19]:

	Amount	Age
count	11239.000000	11239.000000
mean	9453.610553	35.410357
std	5222.355168	12.753866
min	188.000000	12.000000
25%	5443.000000	27.000000
50%	8109.000000	33.000000
75%	12675.000000	43.000000
max	23952.000000	92.000000

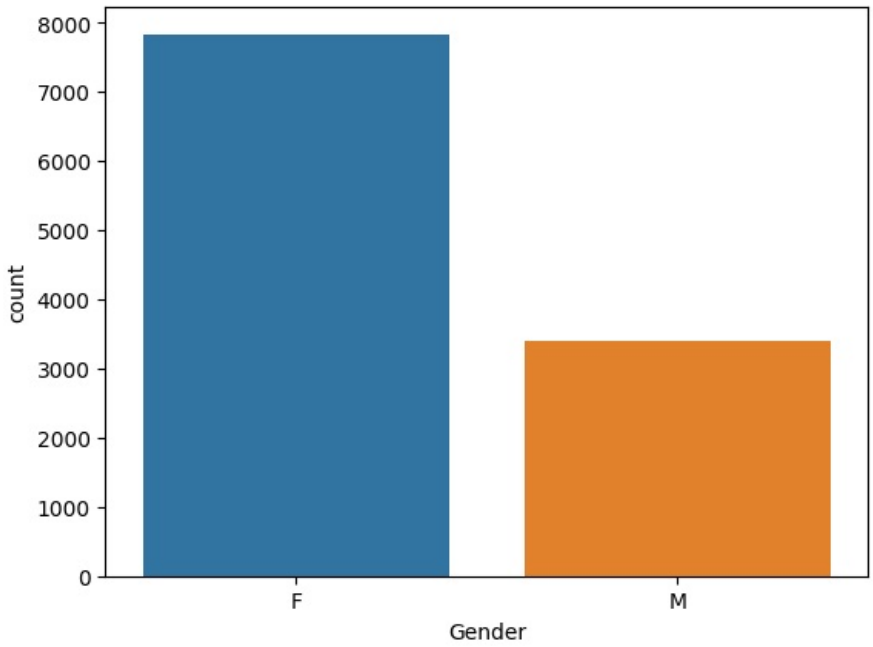
Exploratory Data Analysis

1. Gender

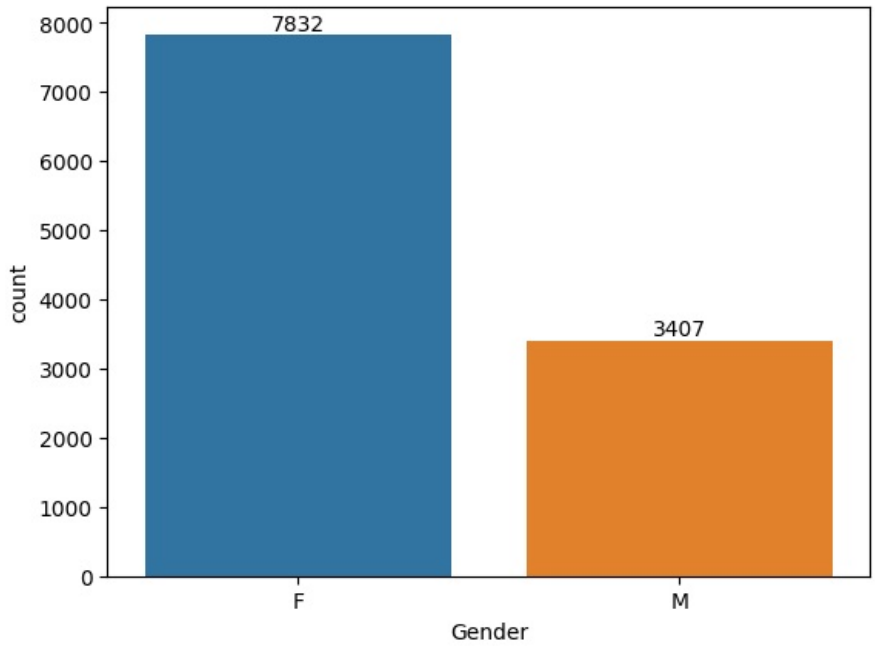
In [20]:
```
df.columns
```

Out[20]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

In [21]:
```
ax = sns.countplot(x = "Gender", data = df)
```



In [22]:
```
# ax contains whole graph including label, title, containers, etc. loop in conatiners where each conatiner will
ax = sns.countplot(x = "Gender", data = df)
for bars in ax.containers:
    ax.bar_label(bars)
```



In [23]:
```
df.groupby(['Gender'], as_index = False)['Amount'].sum().sort_values(by="Amount", ascending = False)
```

Out[23]:

	Gender	Amount
0	F	74335853
1	M	31913276

In [24]:
```
sales_gen = df.groupby(['Gender'], as_index = False)['Amount'].sum().sort_values(by="Amount", ascending = False
sns.barplot(x = 'Gender', y = 'Amount', data = data_set)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In [24], line 2
      1 sales_gen = df.groupby(['Gender'], as_index = False)['Amount'].sum().sort_values(by="Amount", ascending
= False)
----> 2 sns.barplot(x = 'Gender', y = 'Amount', data = data_set)

NameError: name 'data_set' is not defined
```

Most of the Buyers are Females and even Females have higher purchasing power than Males

In [ ]:
```
```

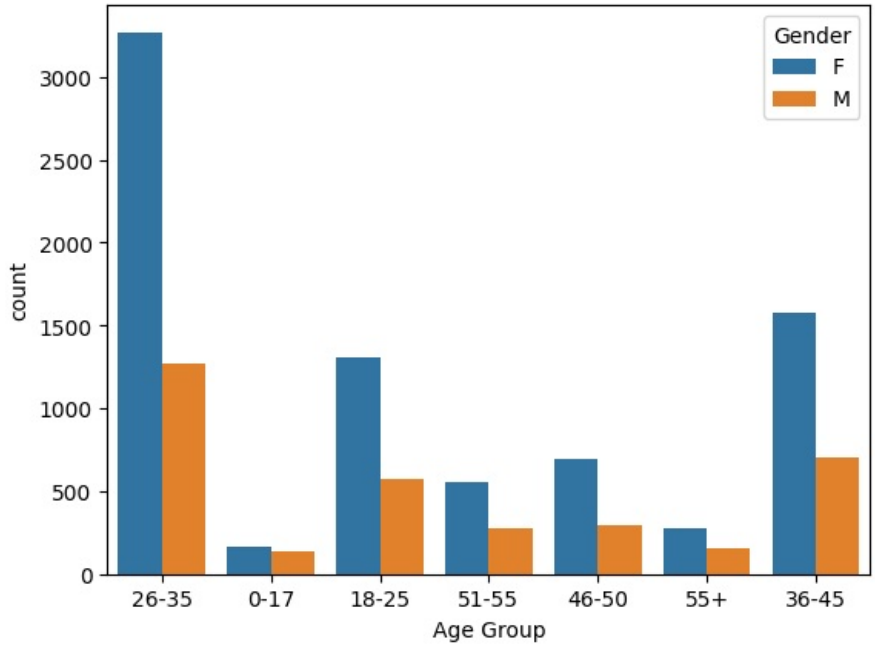
2. Age

In [25]:
```
df.columns
```

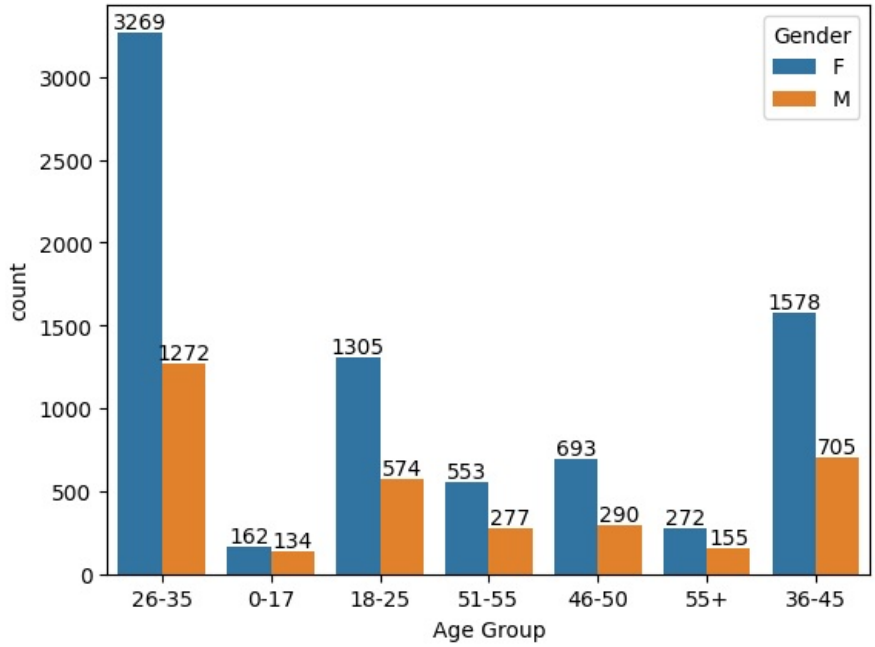
Out[25]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

In [26]:
```
# countplot can take x and y but once at a time, if x -> horizontally, if y -> vertically. One axis will take d
sns.countplot(x = "Age Group", data = df, hue = "Gender")
```

Out[26]:
```
<AxesSubplot: xlabel='Age Group', ylabel='count'>
```

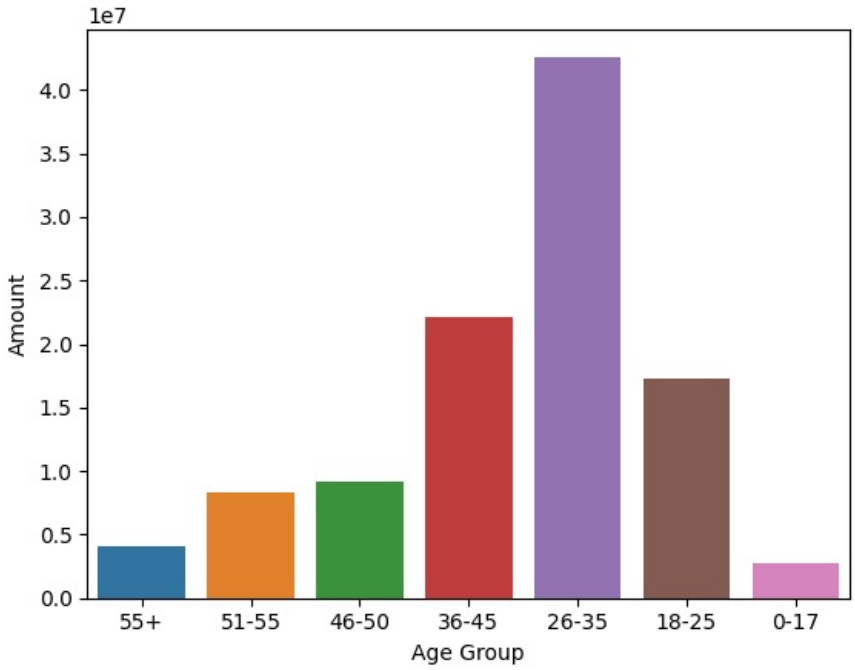


In [27]:
```
ax = sns.countplot(x = 'Age Group', hue = 'Gender', data = df)
for bars in ax.containers:
    ax.bar_label(bars)
```



In [28]:
```
# Total Amount vs Age Group
sales_age = df.groupby(['Age Group'], as_index = False)['Amount'].sum().sort_values(by = 'Age Group', ascending
sns.barplot(x = 'Age Group', y = "Amount", data = sales_age)
```

Out[28]:
```
<AxesSubplot: xlabel='Age Group', ylabel='Amount'>
```



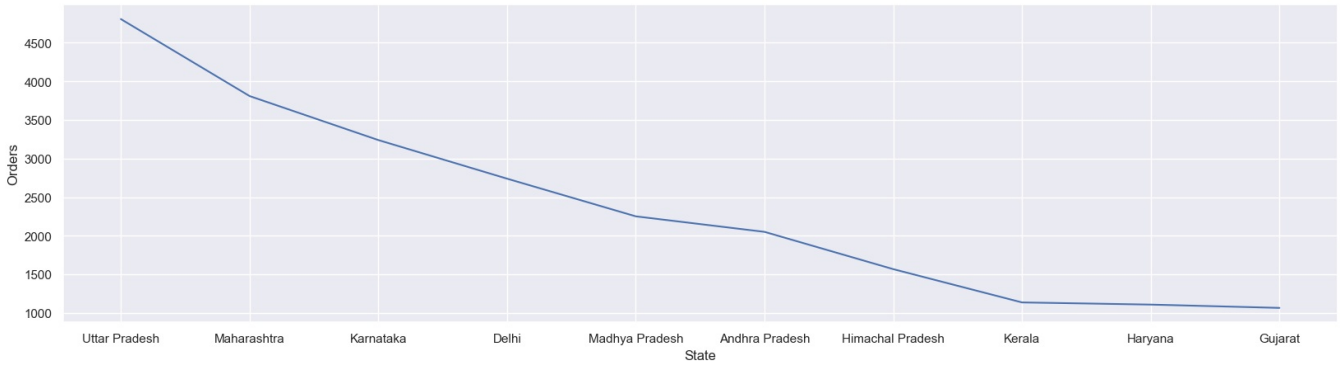
From above graph, most of the buyers are from age group 26-35.

3. State

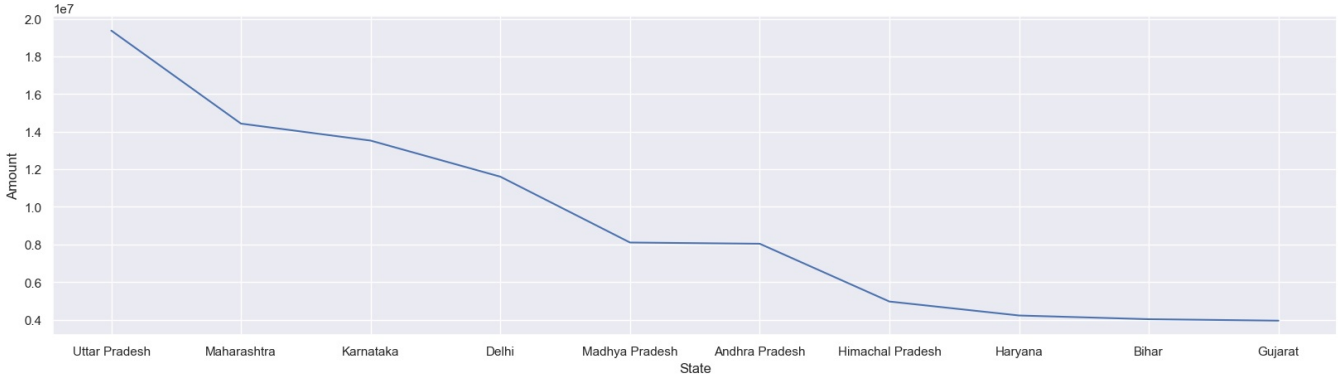
In [29]:
```
df.columns
```

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

In [30]:
```
sales_state = df.groupby(['State'], as_index = False)['Orders'].sum().sort_values(by = 'Orders', ascending = Fa
sns.set(rc={'figure.figsize':(20,5)})
sx = sns.lineplot(x = 'State', y = "Orders", data = sales_state)
```



In [31]:
```
sales_state = df.groupby(['State'], as_index = False)['Amount'].sum().sort_values(by = 'Amount', ascending = Fa
sns.set(rc={'figure.figsize':(20,5)})
sx = sns.lineplot(x = 'State', y = "Amount", data = sales_state)
```



From above graph, we can see that most of the orders and sales are from Uttar Pradesh, Maharashtra, and Karnataka respectively.

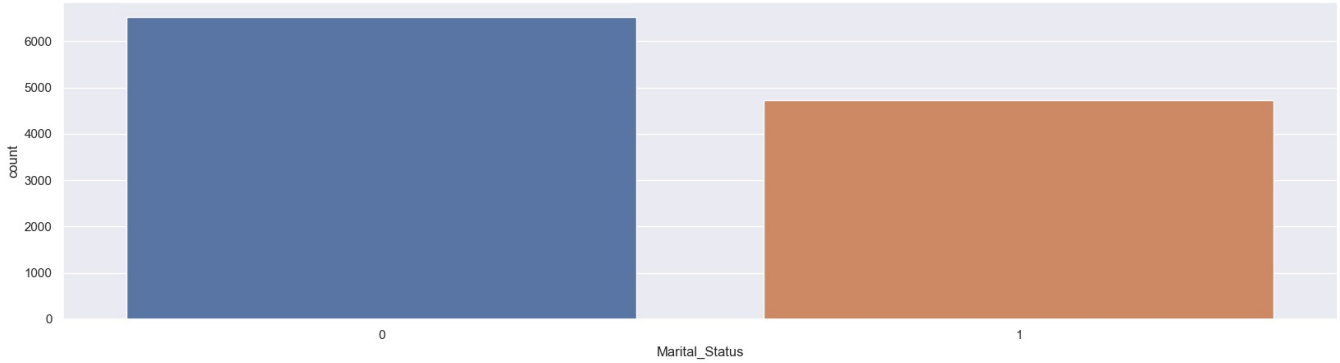
4. Marital Status

4. Marital Status

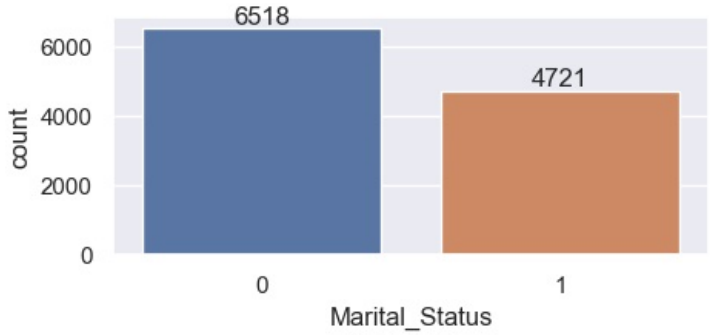
In [32]:
```
df.columns
```

Out[32]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

In [33]:
```
mx = sns.countplot(x = 'Marital_Status', data = df)
sns.set(rc={'figure.figsize':(5,2)})
```



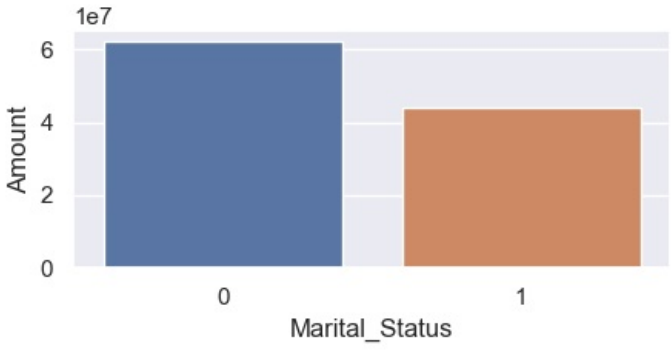
In [34]:
```
mx = sns.countplot(x = 'Marital_Status', data = df)
for bars in mx.containers:
    mx.bar_label(bars)
```



In [35]:
```
# Unlike countplot, barplot can take x and y axis at same time.

sales_marital = df.groupby(['Marital_Status'], as_index = False)['Amount'].sum().sort_values(by = "Marital_Stat
sns.barplot(x = 'Marital_Status', y = 'Amount' , data = sales_marital)
```

Out[35]:
```
<AxesSubplot: xlabel='Marital_Status', ylabel='Amount'>
```



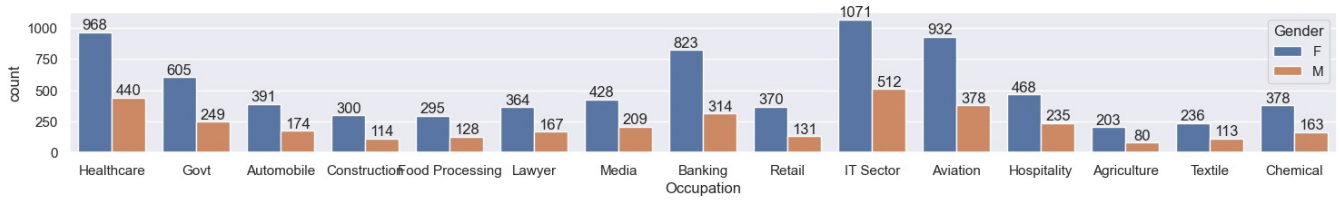
The above graph shows that married people have high purchasing power.

5. Occupation

In [36]:
```
df.columns
```

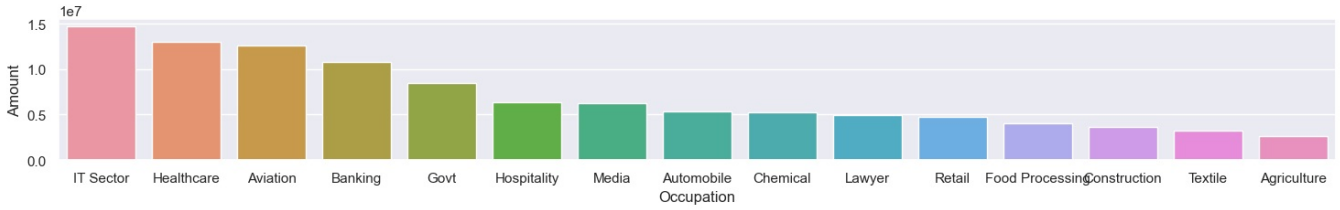
Out[36]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

In [43]:
```
ox = sns.countplot(x = 'Occupation', hue = 'Gender', data = df)
sns.set(rc={'figure.figsize':(18,2)})
for bars in ox.containers:
    ox.bar_label(bars)
```



In [38]:
```
sales_occp = df.groupby(['Occupation'], as_index = False)['Amount'].sum().sort_values(by = 'Amount', ascending =
sns.barplot(x = 'Occupation',y = 'Amount', data = sales_occp)
```

Out[38]: <AxesSubplot: xlabel='Occupation', ylabel='Amount'>



This shows that people working in IT Sector spend more in Diwali.

6. Product Category

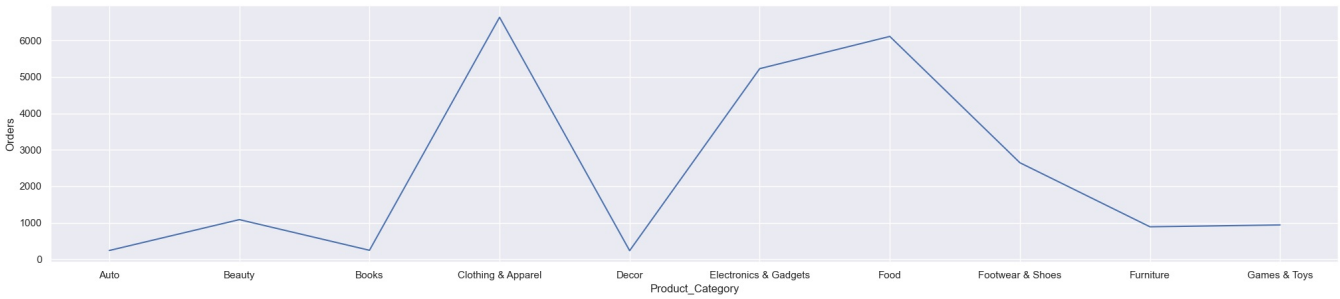
In [39]:
```
px = sns.countplot(x = 'Product_Description', hue)
```

```
  Cell In [39], line 1
    px = sns.countplot(x = 'Product_Description', hue)
                                                  ^
SyntaxError: positional argument follows keyword argument
```

In [48]:
```
sns.lineplot(x = 'Product_Category', y = 'Amount', data = df)
sns.set(rc={'figure.figsize':(25,5)})
```

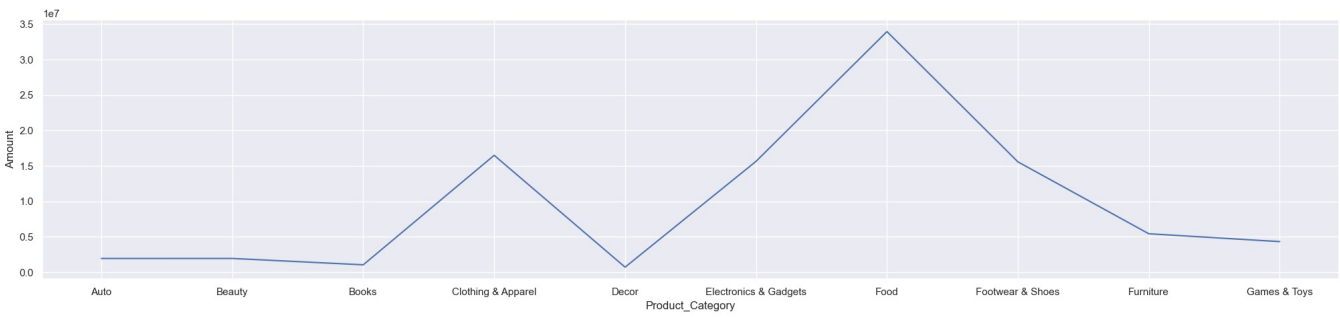
In [49]:
```
sales_prod = df.groupby(['Product_Category'], as_index = False)['Orders'].sum().head(10)
sns.lineplot(x = 'Product_Category', y = 'Orders', data = sales_prod)
```

Out[49]: <AxesSubplot: xlabel='Product_Category', ylabel='Orders'>



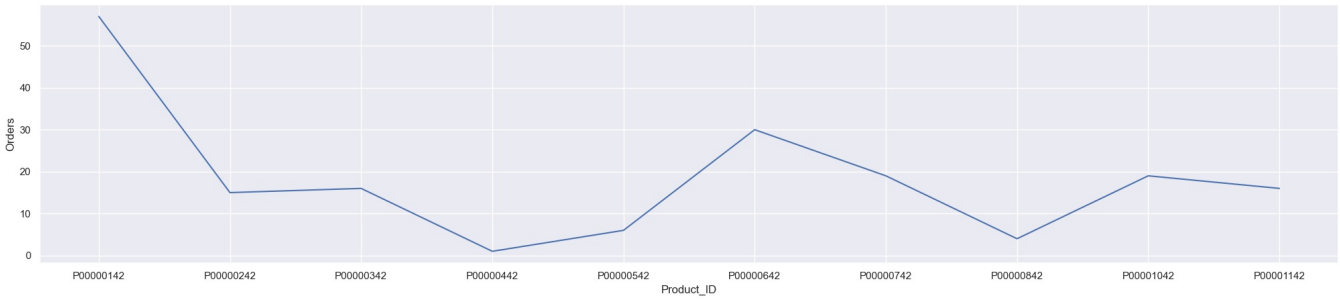
In [50]:
```
sales_prod = df.groupby(['Product_Category'], as_index = False)['Amount'].sum().head(10)
sns.lineplot(x = 'Product_Category', y = 'Amount', data = sales_prod)
```

Out[50]: <AxesSubplot: xlabel='Product_Category', ylabel='Amount'>



In [52]:
```
# showing product-id wise orders
sales_prod = df.groupby(['Product_ID'], as_index = False)['Orders'].sum().head(10)
sns.lineplot(x = 'Product_ID', y = 'Orders', data = sales_prod)
```

Out[52]: <AxesSubplot: xlabel='Product_ID', ylabel='Orders'>



This shows that most sold items are Food, Electronics & Gadgets and Footwear & Shoes.

Conclusion: Married women aged 26-35 years from U.P., Maharashtra and Karnataka working in IT Sector are more likely to buy product from food, footwear & shoes, and electronics category.

In [ ]: