diwali-sales-analysis

October 24, 2025

[33]:
```
# import python libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt # visualizing data
%matplotlib inline
import seaborn as sns
from warnings import filterwarnings
filterwarnings('ignore')
```

[34]:
```
# import csv file
df = pd.read_csv('Diwali Sales Data.csv', encoding= 'unicode_escape')
```

[35]:
```
df.shape
```

[35]:
```
(11251, 15)
```

[36]:
```
df.head()
```

[36]:

	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			
4	1000588	Joni	P00057942	M	26-35	28	1	Gujarat	Western	Food Processing	Auto	2			

```
3  23912.0     NaN       NaN
4  23877.0     NaN       NaN
```

[37]:
```
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
```

[38]:
```
#drop unrelated/blank columns
df.drop(['Status', 'unnamed1'], axis=1, inplace=True)
```

[39]:
```
#check for null values
pd.isnull(df).sum()
```

[39]:
```
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
```

[40]:
```
# drop null values
df.dropna(inplace=True)
```

[41]:
```
# change data type
df['Amount'] = df['Amount'].astype('int')
```

[42]:
```
df['Amount'].dtypes
```

[42]:
```
dtype('int64')
```

[43]:
```
df.columns
```

[43]:
```
Index(['User_ID', 'Cust_name', 'Product_ID', 'Gender', 'Age Group', 'Age',
       'Marital_Status', 'State', 'Zone', 'Occupation', 'Product_Category',
       'Orders', 'Amount'],
      dtype='object')
```

[44]:
```
#rename column
df.rename(columns= {'Marital_Status':'Shaadi'})
```

[44]:

	User_ID	Cust_name	Product_ID	Gender	Age Group	Age	Shaadi	\
0	1002903	Sanskriti	P00125942	F	26-35	28	0	
1	1000732	Kartik	P00110942	F	26-35	35	1	
2	1001990	Bindu	P00118542	F	26-35	35	1	
3	1001425	Sudevi	P00237842	M	0-17	16	0	
4	1000588	Joni	P00057942	M	26-35	28	1	
…	…	…	…	…	…	…	…	
11246	1000695	Manning	P00296942	M	18-25	19	1	
11247	1004089	Reichenbach	P00171342	M	26-35	33	0	
11248	1001209	Oshin	P00201342	F	36-45	40	0	
11249	1004023	Noonan	P00059442	M	36-45	37	0	
11250	1002744	Brumley	P00281742	F	18-25	19	0	

	State	Zone	Occupation	Product_Category	Orders	\
0	Maharashtra	Western	Healthcare	Auto	1	
1	Andhra Pradesh	Southern	Govt	Auto	3	
2	Uttar Pradesh	Central	Automobile	Auto	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
0        23952
1        23934
2        23924
3        23912
4        23877
…        …
11246      370
11247      367
11248      213
11249      206
11250      188

[11239 rows x 13 columns]
```

[45]:
```
# describe() method returns description of the data in the DataFrame (i.e.␣
↪count, mean, std, etc)
df.describe()
```

[45]:

	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

[46]:
```
# use describe() for specific columns
df[['Age', 'Orders', 'Amount']].describe()
```

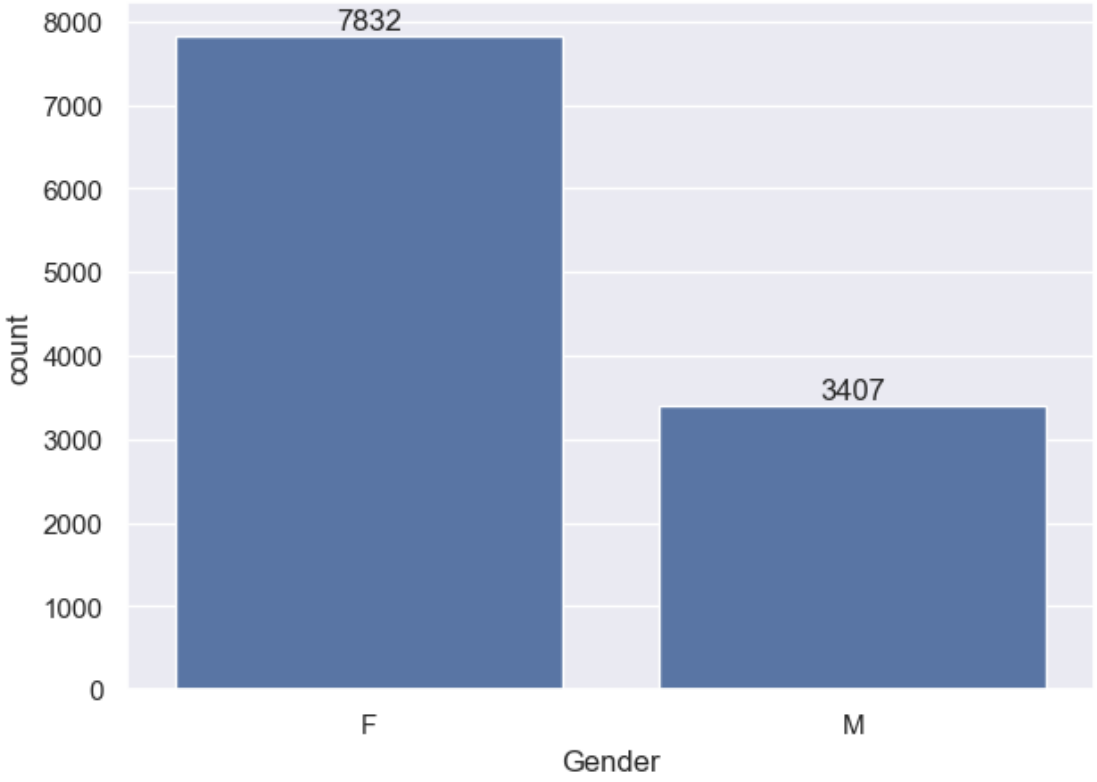
[46]:

	Age	Orders	Amount
count	11239.000000	11239.000000	11239.000000
mean	35.410357	2.489634	9453.610553
std	12.753866	1.114967	5222.355168
min	12.000000	1.000000	188.000000
25%	27.000000	2.000000	5443.000000
50%	33.000000	2.000000	8109.000000
75%	43.000000	3.000000	12675.000000
max	92.000000	4.000000	23952.000000

1 Exploratory Data Analysis

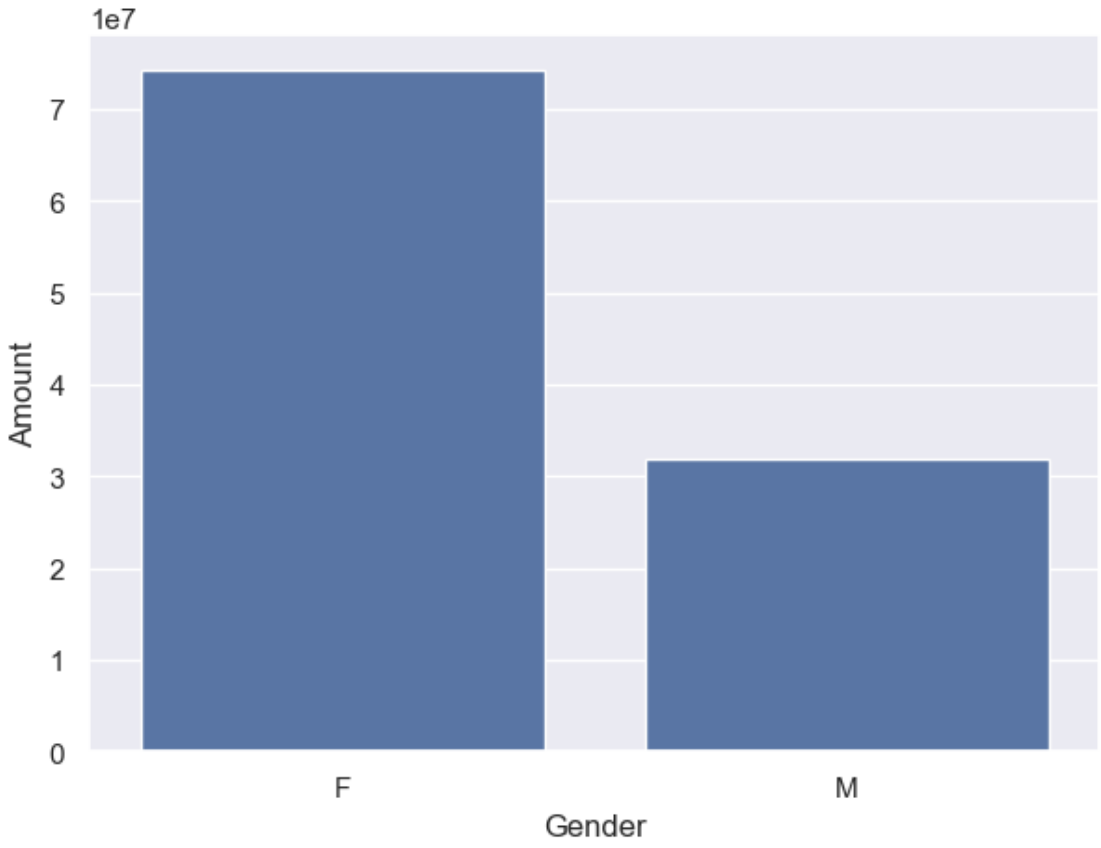
1.0.1 Gender

[47]:
```
# plotting a bar chart for Gender and it's count

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

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



[48]:
```
# plotting a bar chart for gender vs total amount

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

sns.barplot(x = 'Gender',y= 'Amount' ,data = sales_gen)
```

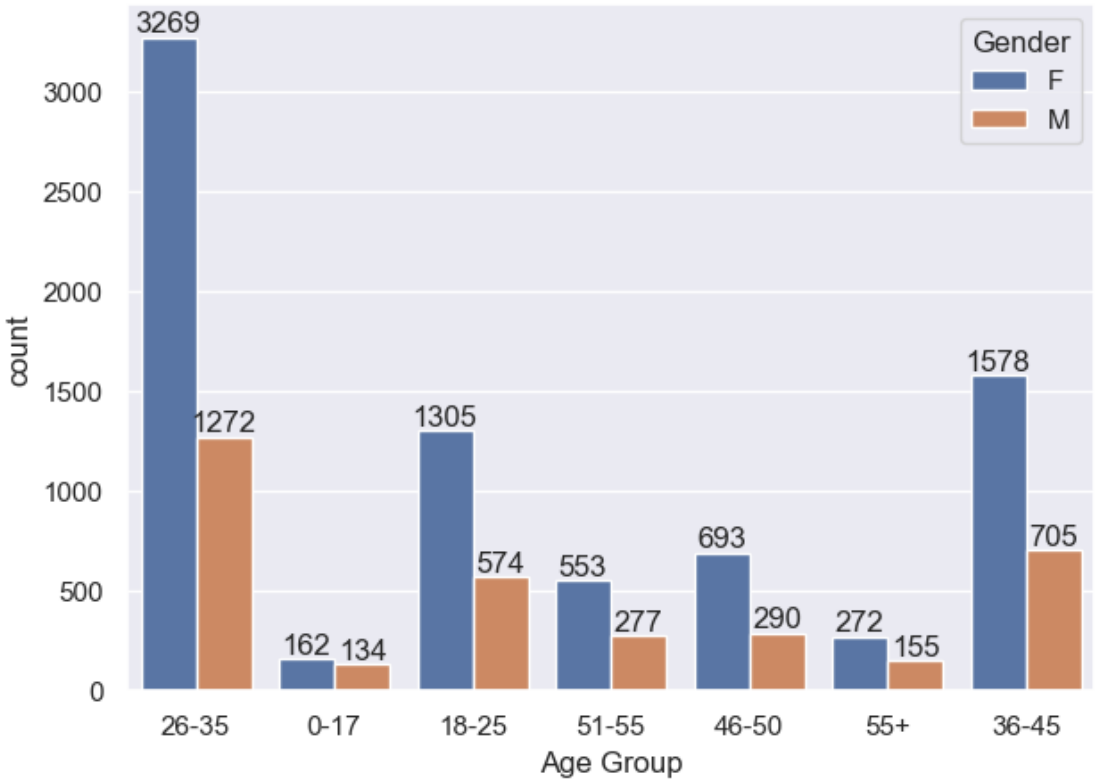
[48]:
```
<Axes: xlabel='Gender', ylabel='Amount'>
```



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

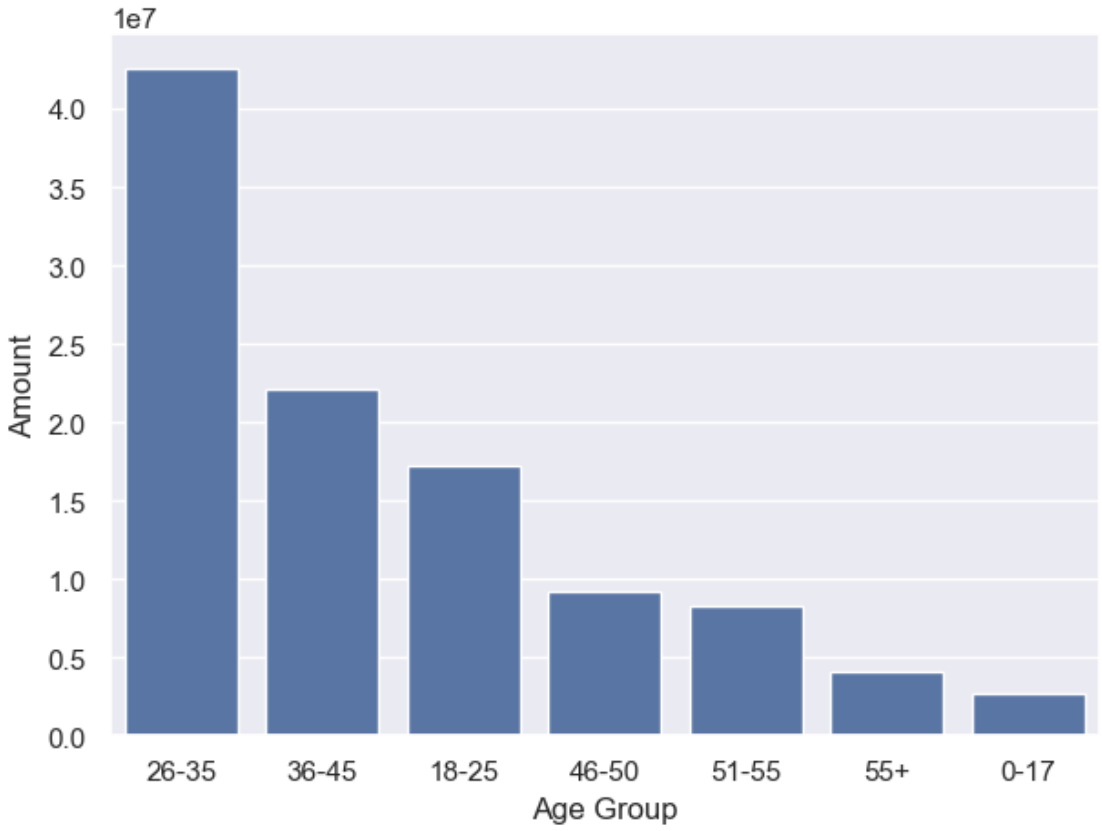
1.0.2 Age

[49]:
```
ax = sns.countplot(data = df, x = 'Age Group', hue = 'Gender')

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



[50]:
```
# 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)
```

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

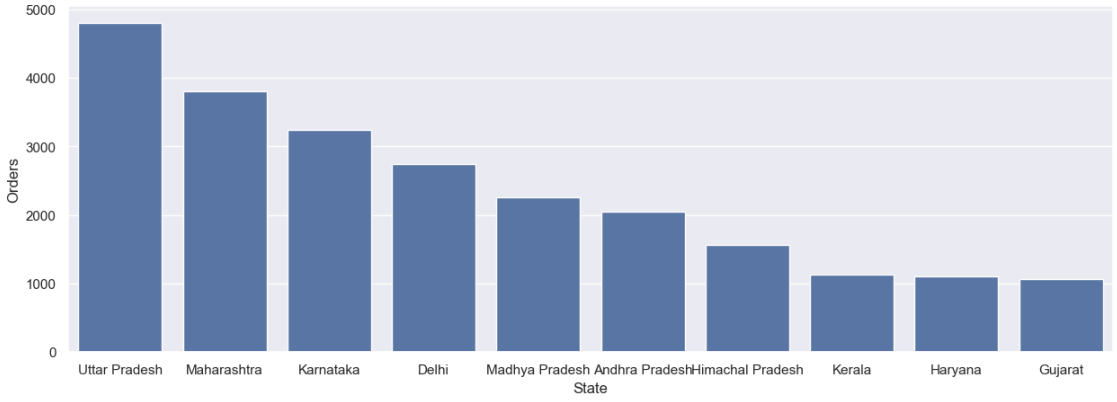


From above graphs we can see that most of the buyers are of age group between 26-35 yrs female

1.0.3 State

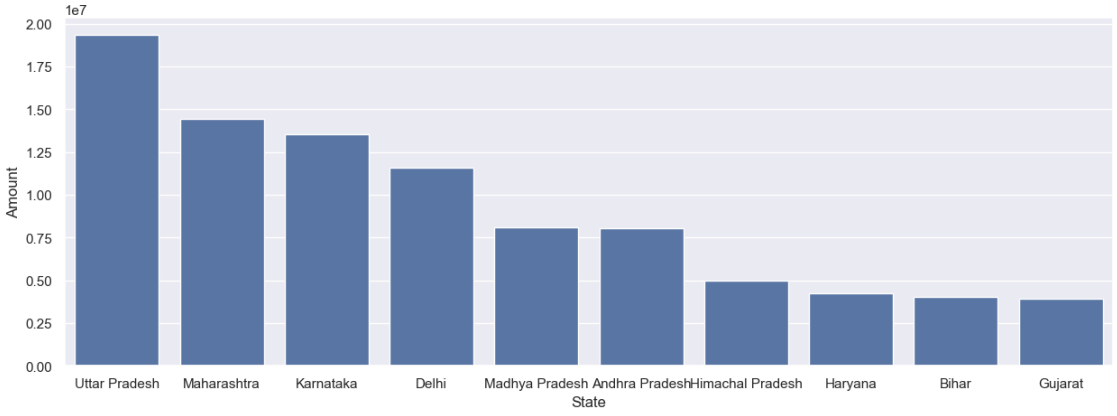
[51]:
```
# total number of orders from top 10 states

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')
```

[51]:
```
<Axes: xlabel='State', ylabel='Orders'>
```



[52]:
```
# 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) # type: ignore

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

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

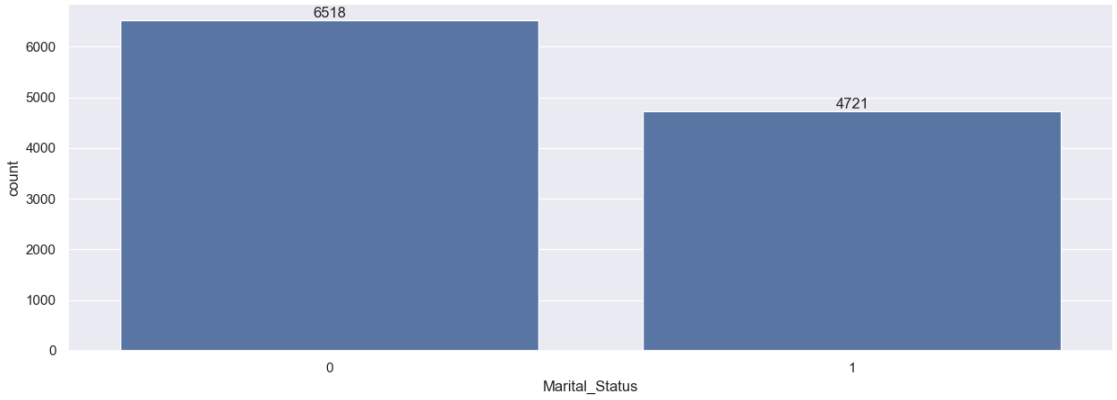


From above graphs we can see that most of the orders & total sales/amount are from Uttar Pradesh, Maharashtra and Karnataka respectively

1.0.4 Marital Status

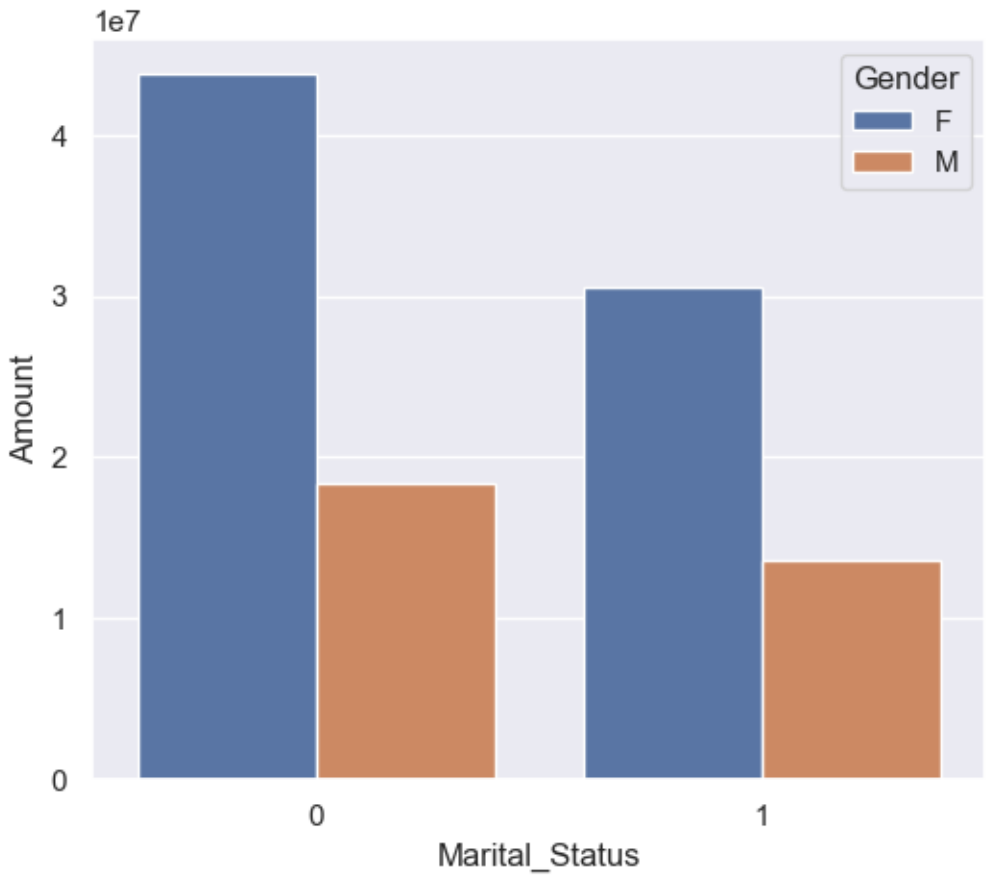
[53]:
```
ax = sns.countplot(data = df, x = 'Marital_Status')

sns.set(rc={'figure.figsize':(7,5)})
```

```
for bars in ax.containers:
    ax.bar_label(bars) # type: ignore
```



[54]:
```
sales_state = df.groupby(['Marital_Status', 'Gender'],␣
↪as_index=False)['Amount'].sum().sort_values(by='Amount', ascending=False) #␣
↪type: ignore

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

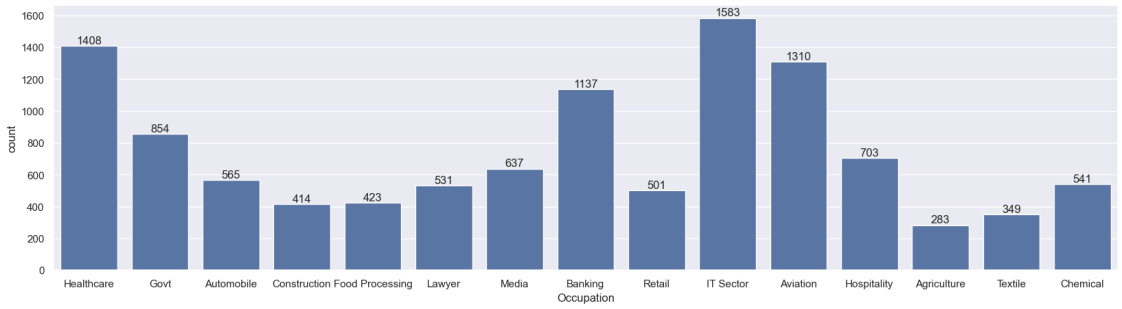
[54]: <Axes: xlabel='Marital_Status', ylabel='Amount'>



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

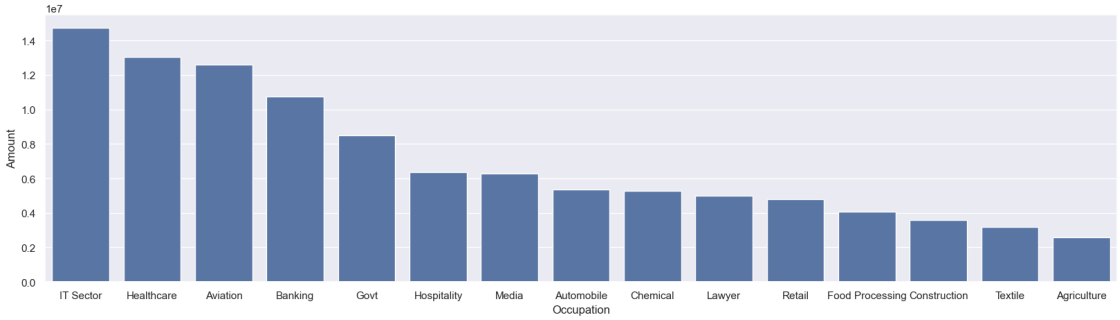
1.0.5 Occupation

[55]:
```
sns.set(rc={'figure.figsize':(20,5)})
ax = sns.countplot(data = df, x = 'Occupation')

for bars in ax.containers:
    ax.bar_label(bars) # type: ignore
```



[56]:
```
sales_state = df.groupby(['Occupation'], as_index=False)['Amount'].sum().
↪sort_values(by='Amount', ascending=False) # type: ignore

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

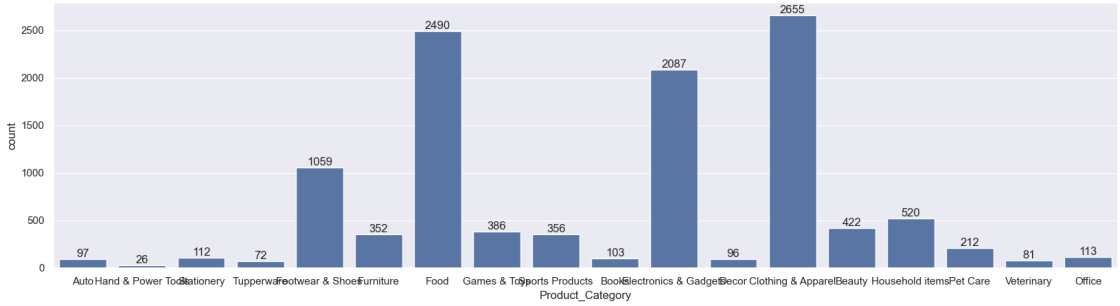
[56]:
```
<Axes: xlabel='Occupation', ylabel='Amount'>
```



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

1.0.6 Product Category

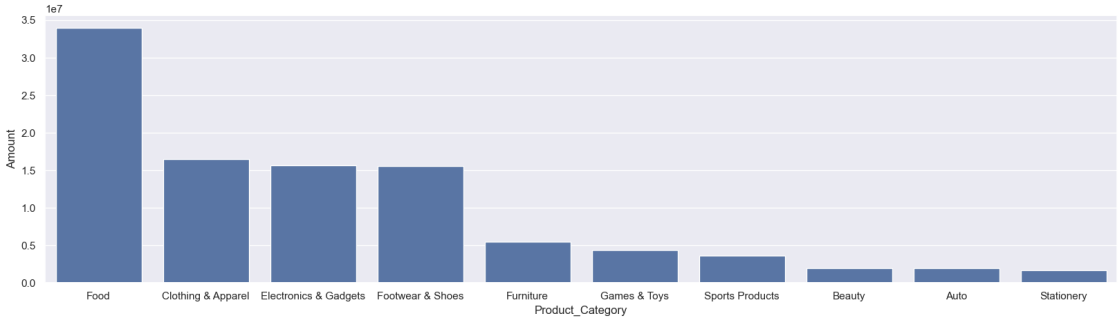
[57]:
```
sns.set(rc={'figure.figsize':(20,5)})
ax = sns.countplot(data = df, x = 'Product_Category')

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



[58]:
```
sales_state = df.groupby(['Product_Category'], as_index=False)['Amount'].sum().
↪sort_values(by='Amount', ascending=False).head(10) # type: ignore
```

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

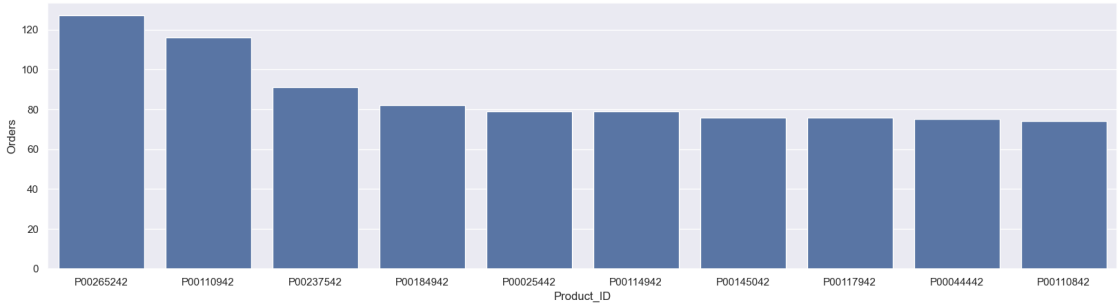
[58]: <Axes: xlabel='Product_Category', ylabel='Amount'>



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

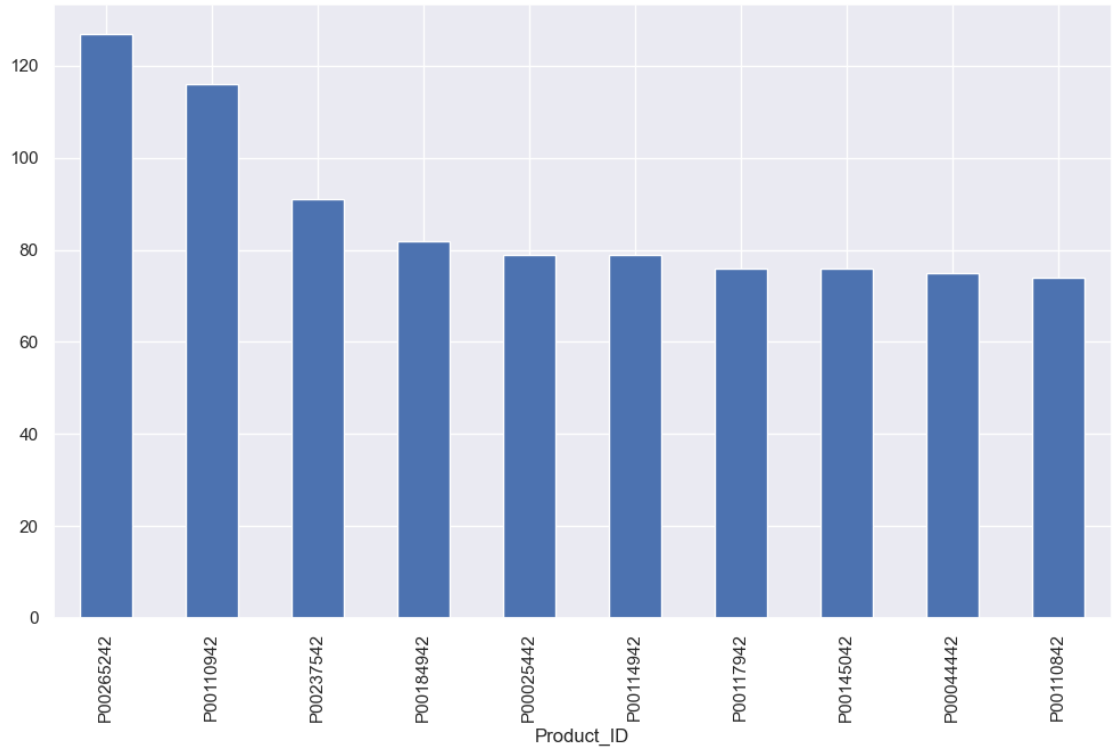
[59]:
```
sales_state = df.groupby(['Product_ID'], as_index=False)['Orders'].sum().
 ↪sort_values(by='Orders', ascending=False).head(10) # type: ignore

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

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



[60]:
```
# top 10 most sold products (same thing as above)

fig1, ax1 = plt.subplots(figsize=(12,7))
df.groupby('Product_ID')['Orders'].sum().nlargest(10).
 ↪sort_values(ascending=False).plot(kind='bar')
```

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



1.1 Conclusion:

- The majority of buyers during Diwali sales are females, especially in the 26-35 age group.
- Most purchases are made by married women, indicating high purchasing power in this segment.
- Top contributing states for orders and sales are Uttar Pradesh, Maharashtra, and Karnataka.
- IT, Healthcare, and Aviation professionals are the most active buyers.
- Food, Clothing, and Electronics are the most popular product categories.
- The top 10 products by sales are dominated by essential and frequently used items.

1.1.1 Business Recommendations:

- Focus marketing efforts on female customers aged 26-35, especially in top-performing states.
- Stock popular categories (Food, Clothing, Electronics) and top-selling products in higher quantities during festive seasons.
- Target married women and professionals in IT, Healthcare, and Aviation for personalized offers.
- Use insights from state and occupation analysis to optimize inventory and regional promotions.

For more details and code, visit the complete project on GitHub.

Thank you!

2 Tips Data Analysis Report

This section demonstrates a full data analysis workflow for the classic tips dataset using pandas, numpy, seaborn, matplotlib, and plotly. Each plot is followed by an interpretation suitable for inclusion in a report.

[61]:
```
# Install required libraries if needed
# !pip install statsmodels scipy plotly
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
from scipy import stats
import statsmodels.api as sm

# Load the tips dataset
df = sns.load_dataset('tips')
```

2.1 1. Data Composition Report

- **Shape:**
- **Columns:**
- **Data Types:**
- **Missing Values:**
- **Basic Statistics:**

[62]:
```
print('Shape:', df.shape)
print('Columns:', df.columns.tolist())
print('Data Types:', df.dtypes)
print('Missing Values:', df.isnull().sum())
display(df.describe())
```

```
Shape: (244, 7)
Columns: ['total_bill', 'tip', 'sex', 'smoker', 'day', 'time', 'size']
Data Types: total_bill     float64
tip            float64
sex           category
smoker        category
day           category
time          category
size             int64
dtype: object
Missing Values: total_bill    0
tip           0
sex           0
smoker        0
day           0
```

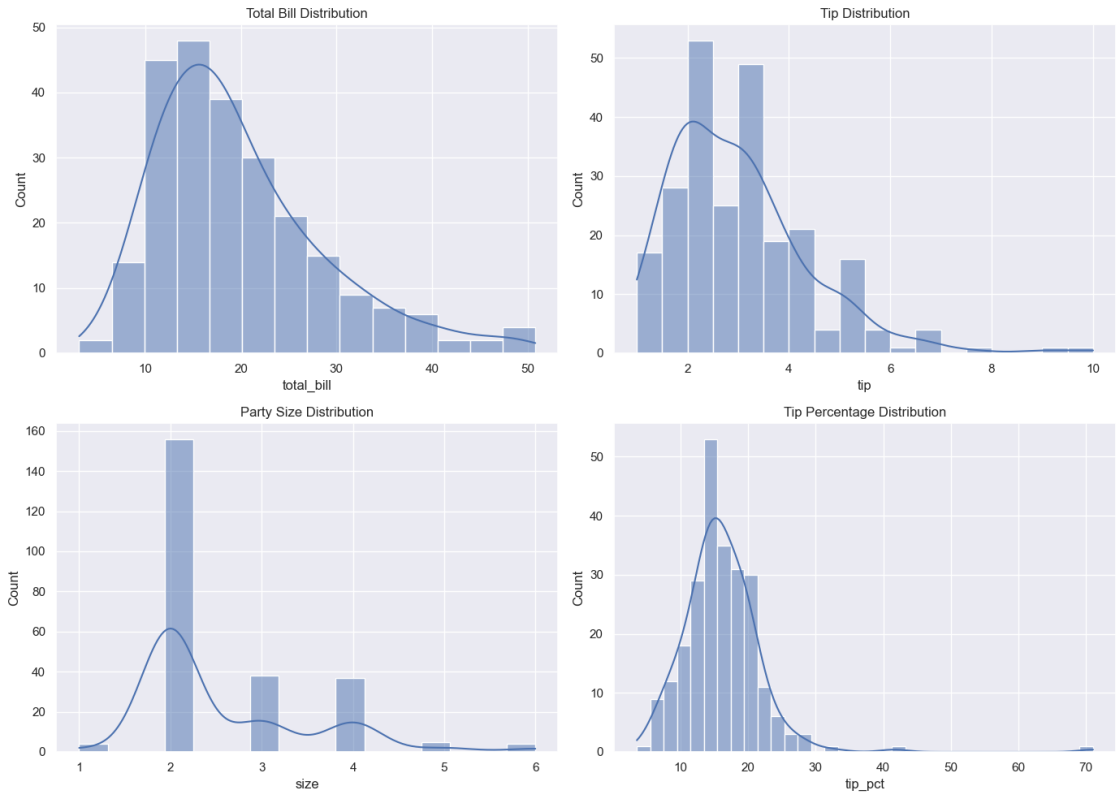
```
time          0
size          0
dtype: int64
```

```
       total_bill         tip        size
count  244.000000  244.000000  244.000000
mean    19.785943    2.998279    2.569672
std      8.902412    1.383638    0.951100
min      3.070000    1.000000    1.000000
25%     13.347500    2.000000    2.000000
50%     17.795000    2.900000    2.000000
75%     24.127500    3.562500    3.000000
max     50.810000   10.000000    6.000000
```

Interpretation: - The dataset contains information about restaurant bills, tips, and customer demographics. - No missing values are present. Numerical columns include `total_bill`, `tip`, and `size`.

2.2 2. Data Distribution Report

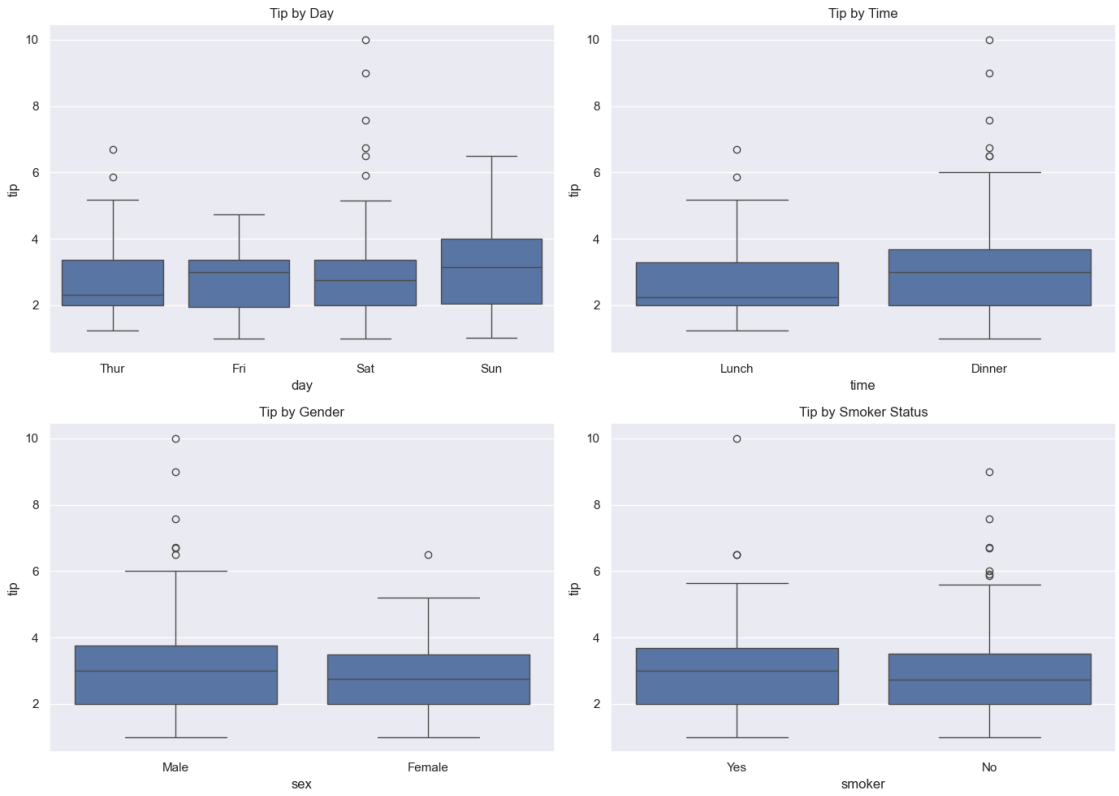
[63]:
```
df['tip_pct'] = df['tip'] / df['total_bill'] * 100
fig, axes = plt.subplots(2, 2, figsize=(14, 10))
sns.histplot(df['total_bill'], kde=True, ax=axes[0,0])
axes[0,0].set_title('Total Bill Distribution')
sns.histplot(df['tip'], kde=True, ax=axes[0,1])
axes[0,1].set_title('Tip Distribution')
sns.histplot(df['size'], kde=True, ax=axes[1,0])
axes[1,0].set_title('Party Size Distribution')
sns.histplot(df['tip_pct'], kde=True, ax=axes[1,1])
axes[1,1].set_title('Tip Percentage Distribution')
plt.tight_layout()
plt.show()
```



Interpretation: - Total bill and tip are right-skewed; most bills are between $10-20 and tips between $2-4. - Party size is most commonly 2. - Tip percentage is centered around 15-20%.

2.3 3. Data Comparison Report

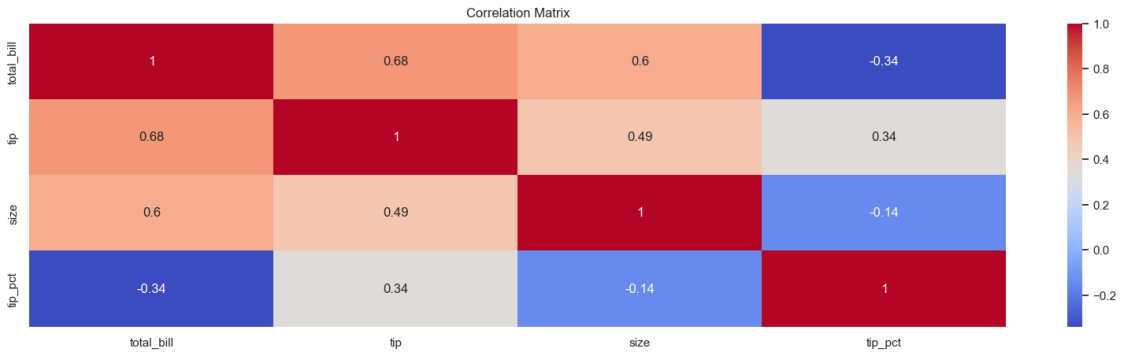
[64]:
```
fig, axes = plt.subplots(2, 2, figsize=(14, 10))
sns.boxplot(x='day', y='tip', data=df, ax=axes[0,0])
axes[0,0].set_title('Tip by Day')
sns.boxplot(x='time', y='tip', data=df, ax=axes[0,1])
axes[0,1].set_title('Tip by Time')
sns.boxplot(x='sex', y='tip', data=df, ax=axes[1,0])
axes[1,0].set_title('Tip by Gender')
sns.boxplot(x='smoker', y='tip', data=df, ax=axes[1,1])
axes[1,1].set_title('Tip by Smoker Status')
plt.tight_layout()
plt.show()
```



Interpretation: - Dinner tips are higher than lunch tips. - Weekend days (Sat, Sun) show higher tips. - No significant difference in tips by gender or smoker status.

2.4 4. Data Relationship Report

[65]:
```
corr = df[['total_bill', 'tip', 'size', 'tip_pct']].corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()

fig = px.scatter(df, x='total_bill', y='tip', color='time', trendline='ols',␣
↪title='Total Bill vs Tip')
fig.show()
```



Interpretation: - Strong positive correlation between total bill and tip. - Weak correlation between party size and tip percentage. - Scatter plot shows a linear relationship between bill and tip.

2.5 5. Statistical Testing (Optional)

[66]:
```
stat, p = stats.mannwhitneyu(df[df['smoker']=='Yes']['tip'],␣
↪df[df['smoker']=='No']['tip'])
print(f"Mann-Whitney U test p-value (smoker vs non-smoker): {p:.4f}")
```

```
Mann-Whitney U test p-value (smoker vs non-smoker): 0.7919
```

Interpretation: - If $p < 0.05$, there is a statistically significant difference in tips between smokers and non-smokers. Otherwise, there is no significant difference.