Waiter's Tip Prediction

The food server of a restaurant recorded data about the tips given to the waiters for serving the food. The data recorded by the food server is as follows:

total bill: Total bill in dollars including taxes\ tip : Tip given to waiters in dollars\ sex: gender of the person paying the bill\ smoker: whether the person smoked or not\ day: day of the week\ time: lunch or dinner\ size: number of people in a table

So this is the data recorded by the restaurant. Based on this data, our task is to find the factors affecting waiter tips and train a machine learning model to predict the waiter's tipping.

Importing Libraries


```
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.graph_objects as go
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

EDA

Exploratory Data Analysis (EDA) is an approach to analyzing datasets with the objective of summarizing their main characteristics, often employing statistical graphics and other data visualization methods. The primary goal of EDA is to gain insights, detect patterns, and understand the structure of the data in order to inform subsequent steps in the data analysis process.

Load the dataset

```
df = pd.read_csv("tips.csv")
```

```
df.head()
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4


```
df.shape
```

```
(244, 7)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   total_bill  244 non-null    float64
 1   tip         244 non-null    float64
```

```
 2   sex         244 non-null    object
 3   smoker      244 non-null    object
 4   day         244 non-null    object
 5   time        244 non-null    object
 6   size        244 non-null    int64
dtypes: float64(2), int64(1), object(4)
memory usage: 13.5+ KB
```

Missing Values

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

	0
total_bill	0
tip	0
sex	0
smoker	0
day	0
time	0
size	0

dtype: int64

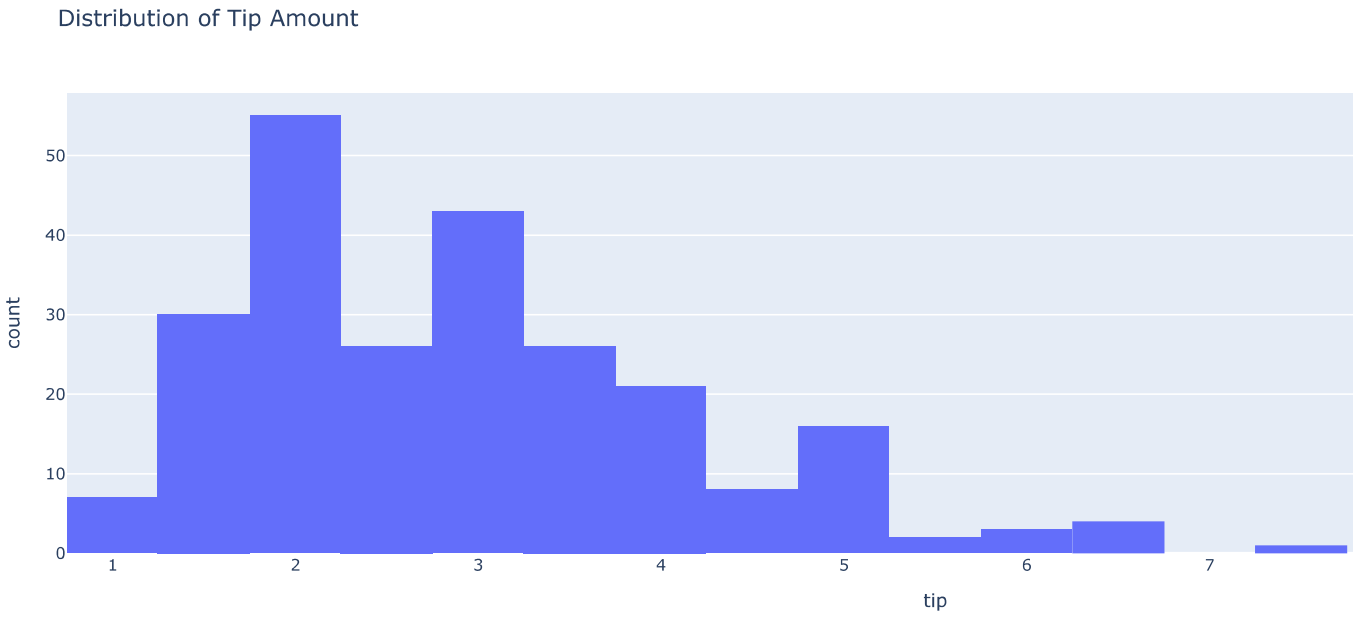
Descriptive Statistics

```
df.describe()
```

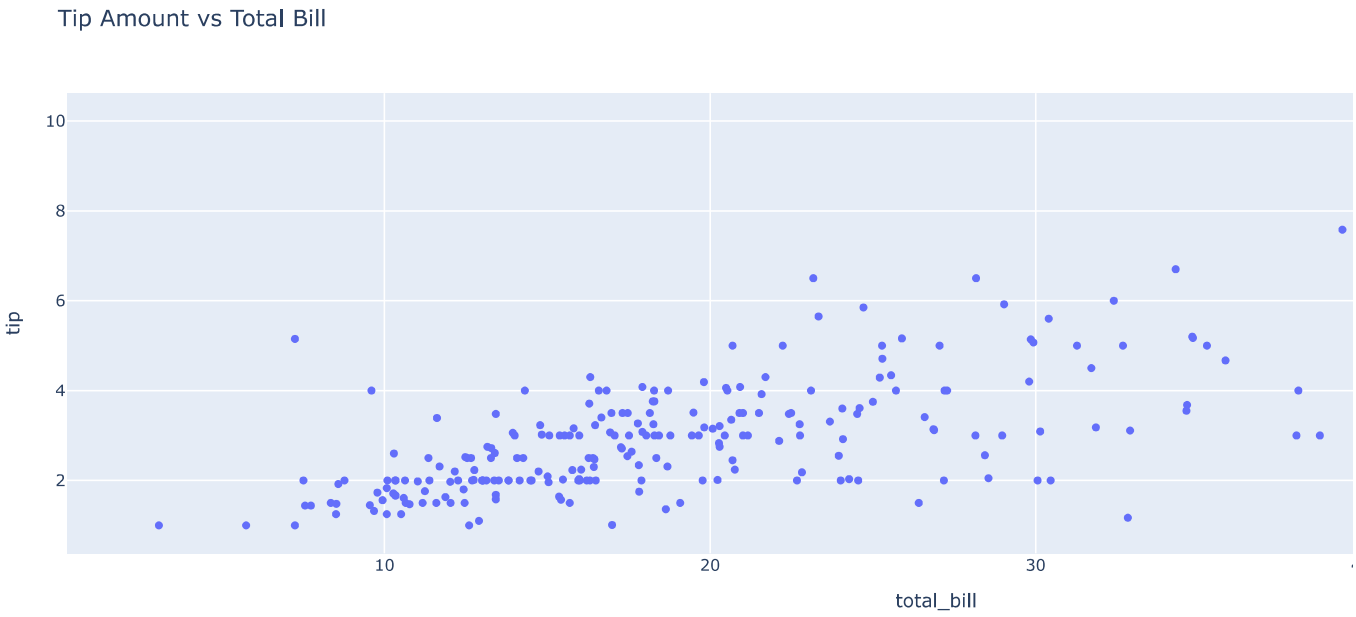
	total_bill	tip	size
count	244.000000	244.000000	244.000000
mean	19.785943	2.998279	2.569672
std	8.902412	1.383638	0.951100
min	3.070000	1.000000	1.000000
25%	13.347500	2.000000	2.000000
50%	17.795000	2.900000	2.000000
75%	24.127500	3.562500	3.000000
max	50.810000	10.000000	6.000000

Total Bill and Tip

```
fig = px.histogram(df, x='tip', title='Distribution of Tip Amount')
fig.show()
```




```
fig = px.scatter(df, x='total_bill', y='tip', title='Tip Amount vs Total Bill')
fig.show()
```




```
correlation = df['total_bill'].corr(df['tip'])
print("Correlation coefficient between total bill and tip:", correlation)
```

```
Correlation coefficient between total bill and tip: 0.6757341092113641
```

The Pearson correlation coefficient between the 'total_bill' and 'tip' variables is approximately 0.676.

This positive correlation indicates a moderately strong linear relationship between the total bill amount and the tip amount. In other words, as the total bill amount increases, the tip amount tends to increase as well.

The Highest Total Bill is 50.810000 and the Lowest is 3.070000

The Hightest Tip is 10.00 and the Lowest Tip is 1.0. Whereas the Average Tip is 2.998279

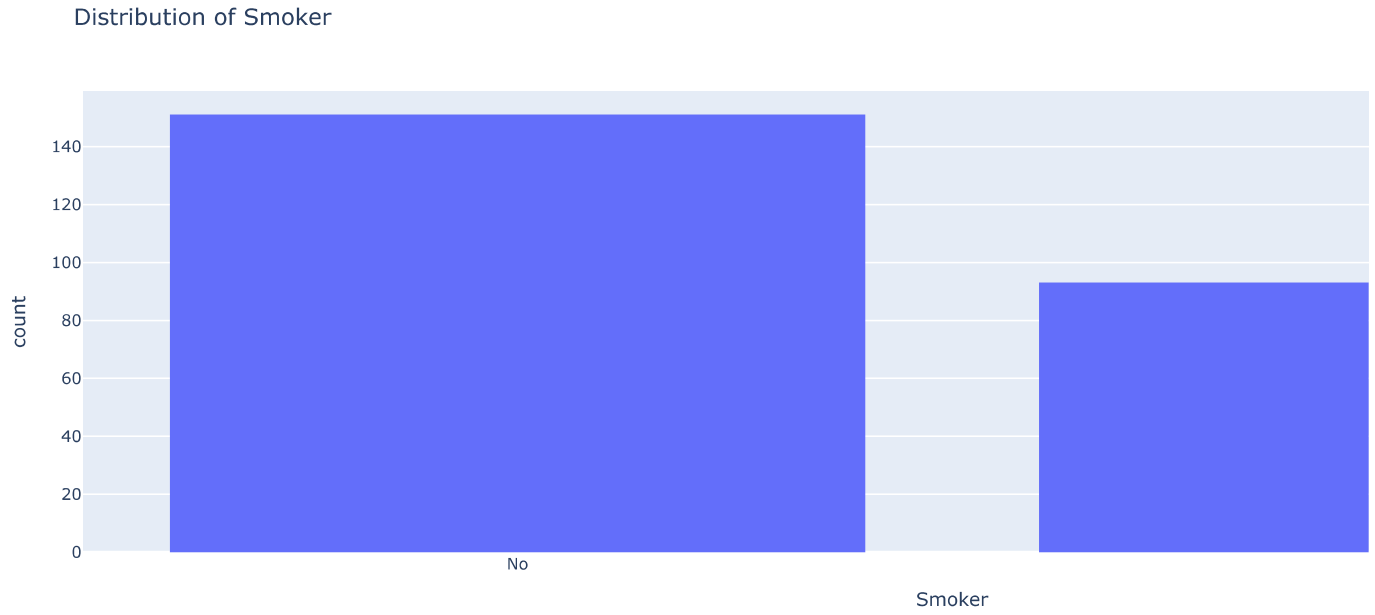
Smoker VS Non-Smoker

```
df['smoker'].value_counts()
```

smoker	count
No	151
Yes	93

dtype: int64

```
fig = px.histogram(df, x='smoker', title='Distribution of Smoker', labels={'smoker': 'Smoker'})
fig.show()
```



151 individuals are Not-Smoker and 93 individuals are Smokers

Time

```
df['time'].value_counts()
```

time	count
Dinner	176
Lunch	68

dtype: int64

```
fig = px.histogram(df, x='time', title='Distribution of Time', labels={'time': 'Time'})
fig.show()
```

There are 176 instances recorded as 'Dinner' and 68 instances recorded as 'Lunch' in the dataset.

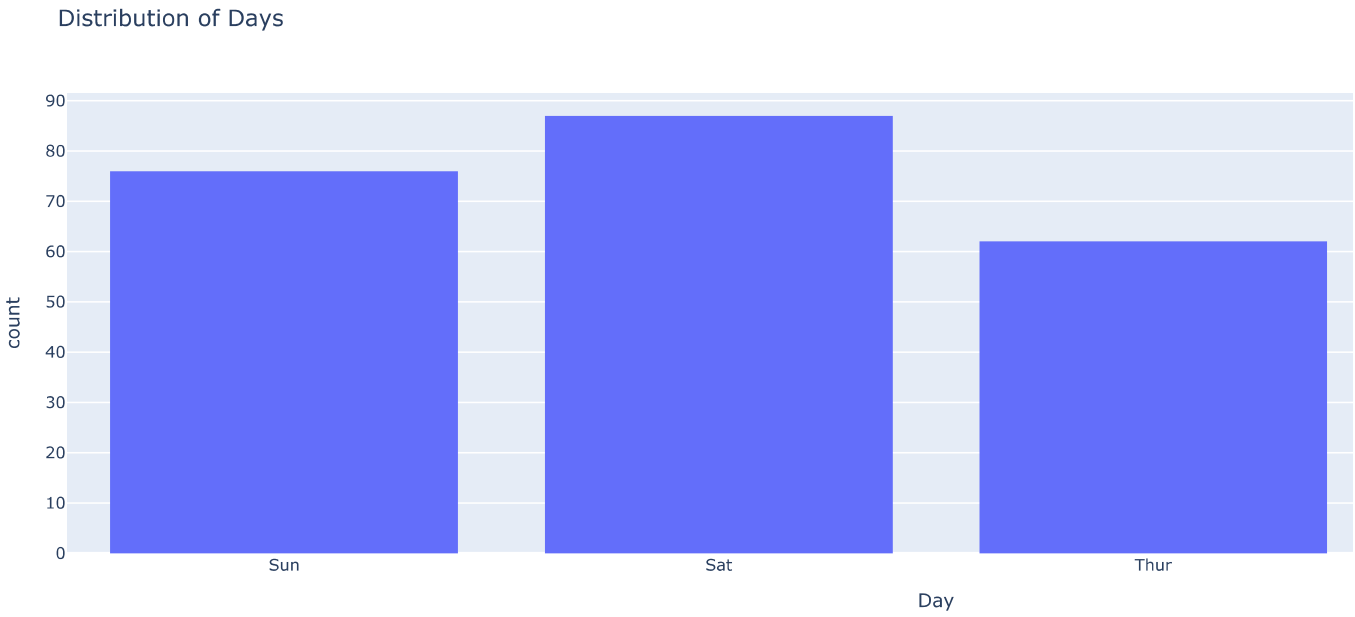
Day

```
df['day'].value_counts()
```

day	count
Sat	87
Sun	76
Thur	62
Fri	19

dtype: int64

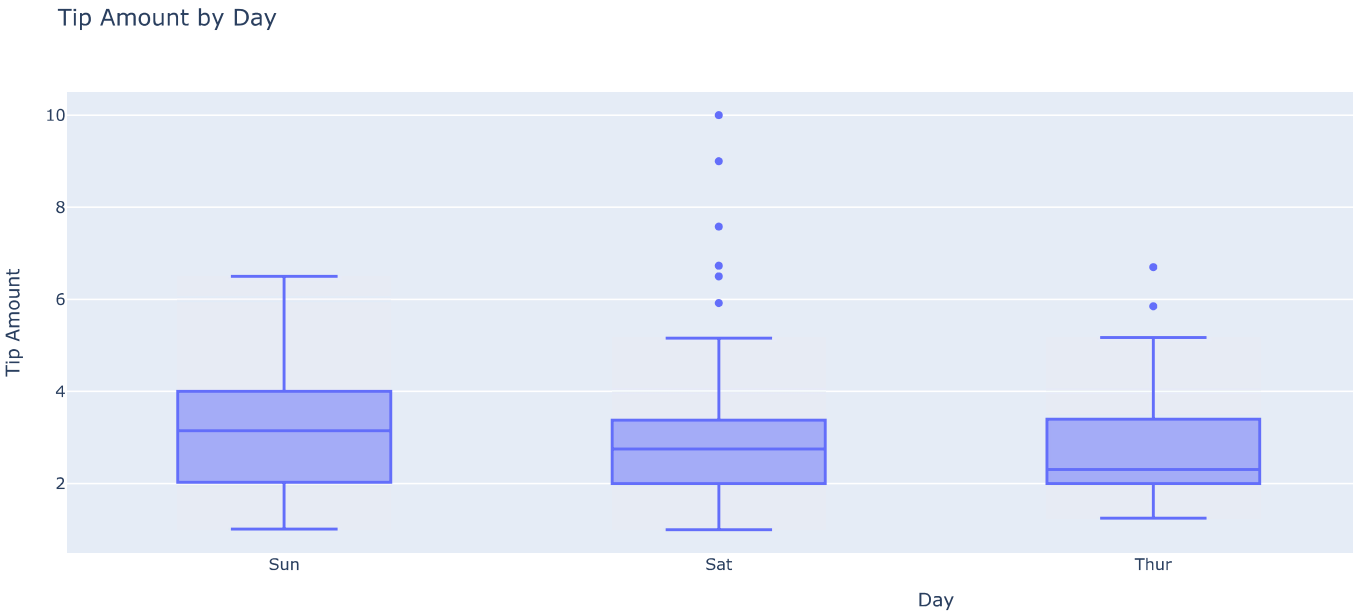
```
fig = px.histogram(df, x='day', title='Distribution of Days', labels={'day': 'Day'})
fig.show()
```



87instances recorded on Saturday, 76 instances recorded on Sunday, 62 instances recorded onThursday, and 19 instances recorded on Friday.

Tip Amount by Day

```
fig = px.box(df, x='day', y='tip', title='Tip Amount by Day', labels={'day': 'Day', 'tip': 'Tip Amount'})
fig.show()
```

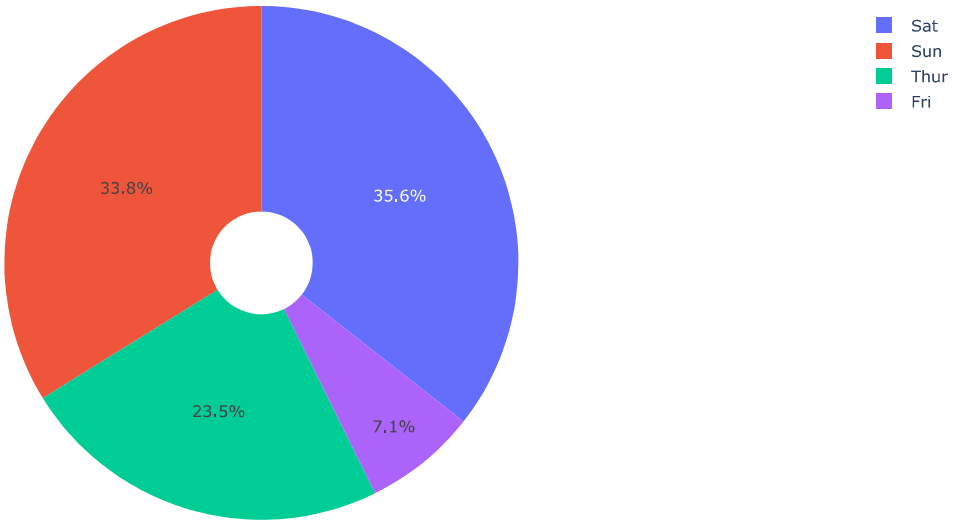



```
total_tips_by_day = df.groupby('day')['tip'].sum()
print(total_tips_by_day)
```

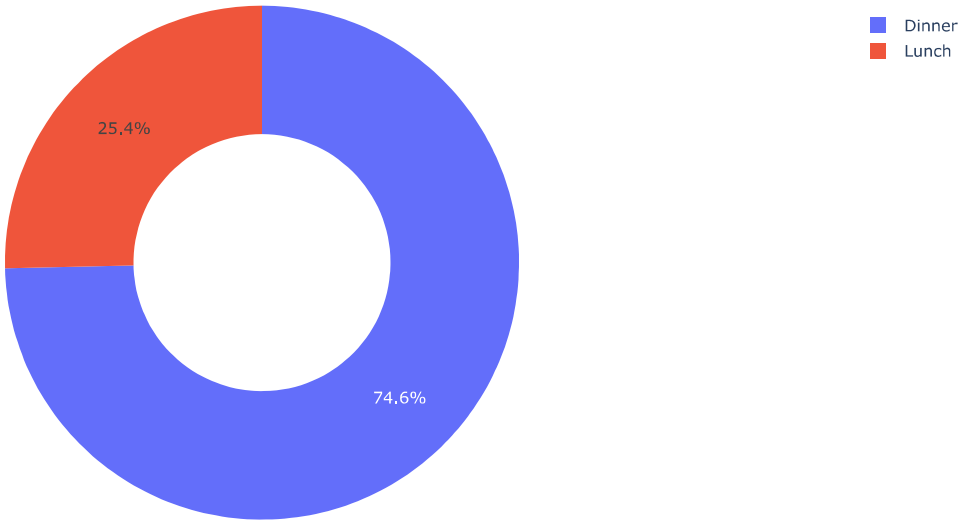
```
day
Fri      51.96
Sat     260.40
```

```
Sun     247.39
Thur    171.83
Name: tip, dtype: float64
```

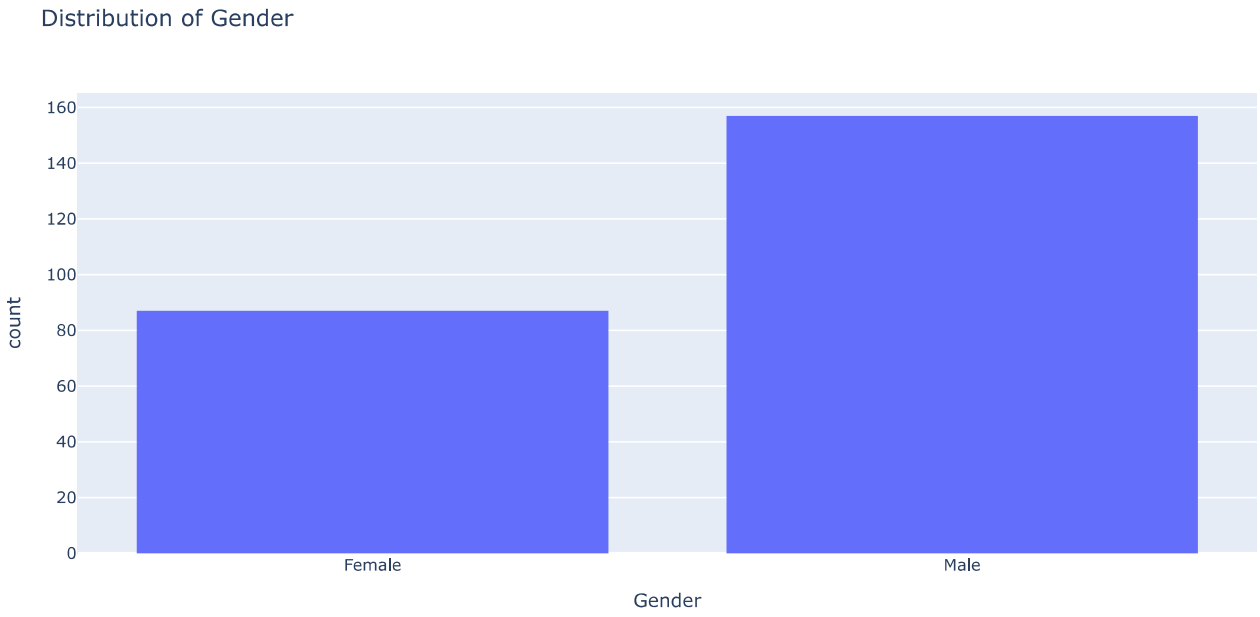
```
figure = px.pie(df, values='tip', names='day', hole = 0.2)
figure.show()
```



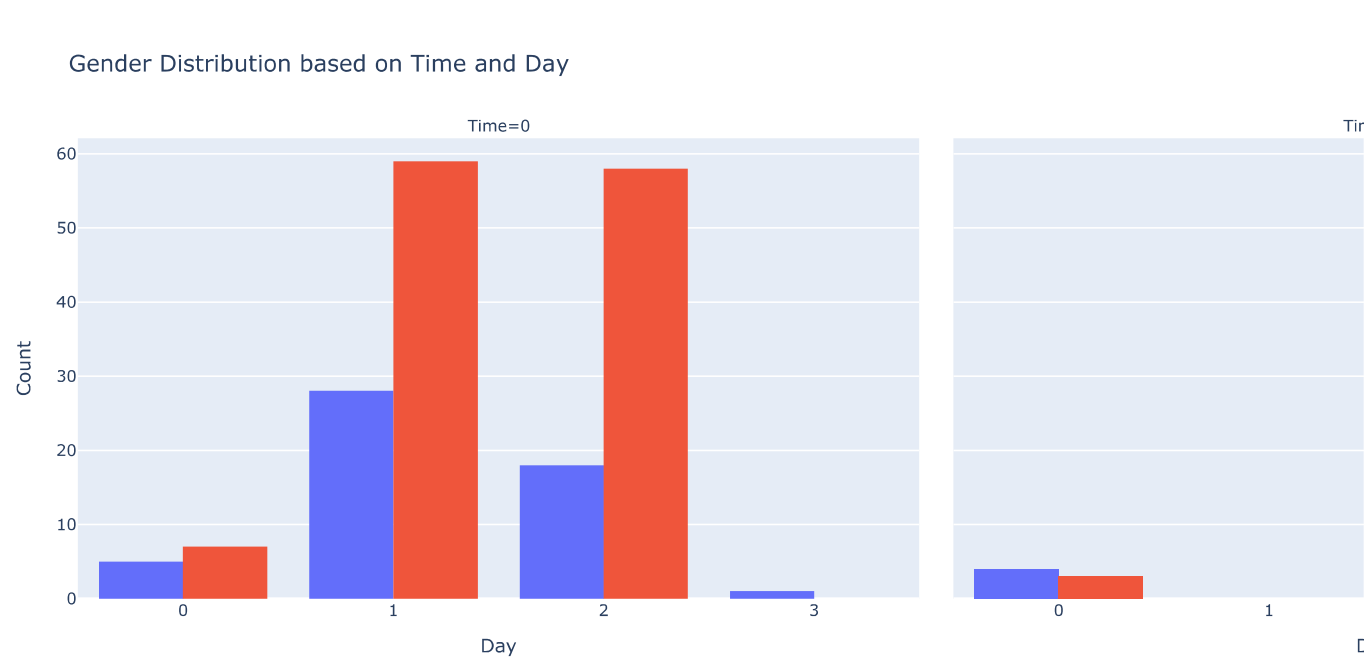

```
figure = px.pie(df, values='tip', names='time', hole = 0.5)
figure.show()
```




```
fig = px.histogram(df, x='sex', title='Distribution of Gender', labels={'sex': 'Gender'})
fig.show()
```

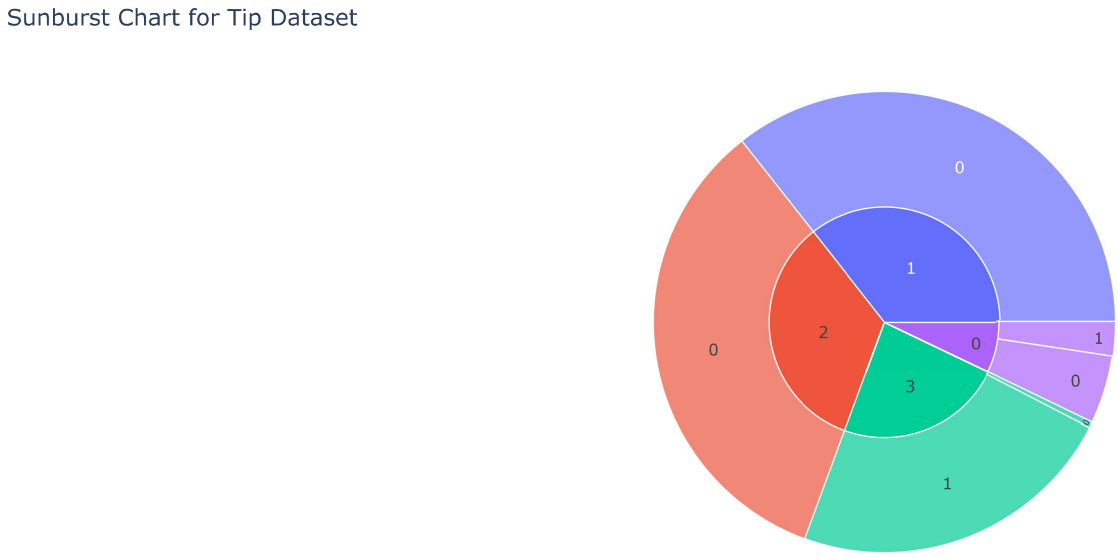



```
fig = px.histogram(df, x='day', color='sex', facet_col='time',
                   title='Gender Distribution based on Time and Day',
                   labels={'day': 'Day', 'sex': 'Gender', 'time': 'Time'},
                   barmode='group')

fig.update_layout(xaxis_title='Day', yaxis_title='Count')

fig.show()
```




```
agg_data = df.groupby(['day', 'time'])['tip'].sum().reset_index()

fig = px.sunburst(agg_data, path=['day', 'time'], values='tip', title='Sunburst Chart for Tip Dataset')
```

```
fig.show()
```



Preprocess the Data

Convert categorical variables into numerical ones using Label Encoding.

```
df.head()
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

Next steps: Generate code with df | View recommended plots | New interactive sheet

Encoding the Data

```
label_encoder = LabelEncoder()
df['sex'] = label_encoder.fit_transform(df['sex'])
df['smoker'] = label_encoder.fit_transform(df['smoker'])
df['day'] = label_encoder.fit_transform(df['day'])
df['time'] = label_encoder.fit_transform(df['time'])

df.head()
```

```
     total_bill  tip  sex  smoker  day  time  size
0         16.99 1.01    0       0    2     0     2
1         10.34 1.66    1       0    2     0     3
```

Generate code with df | View recommended plots | New interactive sheet

Split the data into training and testing sets

```
X = df.drop('tip', axis=1)
y = df['tip']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Machine Learning

Double-click (or enter) to edit

A Linear Regression Model in machine learning is like drawing a straight line through data points to predict a continuous outcome based on input features. It's used to understand how changes in the input features relate to changes in the target variable.

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
features = np.array([[24.50, 1, 0, 0, 1, 4]])
model.predict(features)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493: UserWarning:

X does not have valid feature names, but LinearRegression was fitted with feature names

array([3.97416925])
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score


y_pred = model.predict(X_test)


mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("Mean Absolute Error:", mae)
```