Table of Content

Tips Dataset

The tips dataset is a great starting point for practicing data visualization, statistical analysis, and machine learning techniques. This dataset contains information about the total bill and tip amount for different meals served in a restaurant. The dataset has 244 observations and 7 variables, including total bill, tip, gender, smoker status, day of the week, time of day, and size of the party.

Data Dictionary

Variable	Description
total_bill	Total bill (cost of the meal), in US dollars
tip	Tip (gratuity) in US dollars
sex	Gender of the person who paid the bill (Male or Female)
smoker	Whether the person who paid the bill is a smoker (Yes or No)
day	Day of the week the bill was paid (Thur, Fri, Sat, Sun)
time	Time of dining (Lunch or Dinner)
size	Size of the party (number of people)

Goal

The goal is to gain insights into the data, identify patterns, and draw conclusions based on the visualizations.

Visualization libraries in Python

1. **Plots using Seaborn**
2. **Plots using Matplotlib**

There are different visualization libraries in python, that provides an interface for drawing various graphics. Some most widely used libraries are Seaborn and Matplotlib

1. Seaborn

Seaborn is a Python visualization library based on matplotlib. The library provides a high-

level interface for plotting statistical graphics. As the library uses matplotlib in the backend, we can use the functions in matplotlib along with functions in seaborn.

How to install Seaborn?

1. You can use-
   `!pip install seaborn`

In [1]:
```
!pip install seaborn
```

```
Requirement already satisfied: seaborn in /Users/isalah/opt/anaconda3/lib/pyth
on3.9/site-packages (0.12.1)
Requirement already satisfied: numpy>=1.17 in /Users/isalah/opt/anaconda3/lib/
python3.9/site-packages (from seaborn) (1.21.5)
Requirement already satisfied: pandas>=0.25 in /Users/isalah/opt/anaconda3/li
b/python3.9/site-packages (from seaborn) (1.4.4)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in /Users/isalah/opt/an
aconda3/lib/python3.9/site-packages (from seaborn) (3.5.3)
Requirement already satisfied: fonttools>=4.22.0 in /Users/isalah/opt/anaconda
3/lib/python3.9/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (4.25.0)
Requirement already satisfied: packaging>=20.0 in /Users/isalah/opt/anaconda3/
lib/python3.9/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (21.3)
Requirement already satisfied: python-dateutil>=2.7 in /Users/isalah/opt/anaco
nda3/lib/python3.9/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (2.8.
2)
Requirement already satisfied: cycler>=0.10 in /Users/isalah/opt/anaconda3/li
b/python3.9/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /Users/isalah/opt/anaconda
3/lib/python3.9/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.4.2)
Requirement already satisfied: pyparsing>=2.2.1 in /Users/isalah/opt/anaconda
3/lib/python3.9/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (3.0.9)
Requirement already satisfied: pillow>=6.2.0 in /Users/isalah/opt/anaconda3/li
b/python3.9/site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (9.2.0)
Requirement already satisfied: pytz>=2020.1 in /Users/isalah/opt/anaconda3/li
b/python3.9/site-packages (from pandas>=0.25->seaborn) (2022.1)
Requirement already satisfied: six>=1.5 in /Users/isalah/opt/anaconda3/lib/pyt
hon3.9/site-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1->seab
orn) (1.16.0)
```

In [2]:
```
# Import Seaborn and alias it as sns
import seaborn as sns
```

In [20]:
```
# Get the list of sample datasets from the seaborn library
sns.get_dataset_names()
```

Out[20]:
```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'dowjones',
 'exercise',
 'flights',
 'fmri',
 'geyser',
 'glue',
 'healthexp',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'seaice',
 'taxis',
 'tips',
 'titanic']
```

In [4]:
```
# Now load the tips dataset from the Seaborn library and store it in df
df = sns.load_dataset('tips')
```

In [5]:
```
# check the variable type
type(df)
```

Out[5]:
```
pandas.core.frame.DataFrame
```

The `sns.load_dataset()` function returns a Pandas DataFrame object, which is then assigned to the variable `df`. The `load_dataset` function loads a sample dataset from the seaborn library, in this case the "tips" dataset, and returns the data in a Pandas DataFrame. The resulting object is then assigned to the variable df, which is now a DataFrame.

In [6]:
```
# check top 5 rows
df.head()
```

Out[6]:

	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

In [7]:
```
df.info()
```

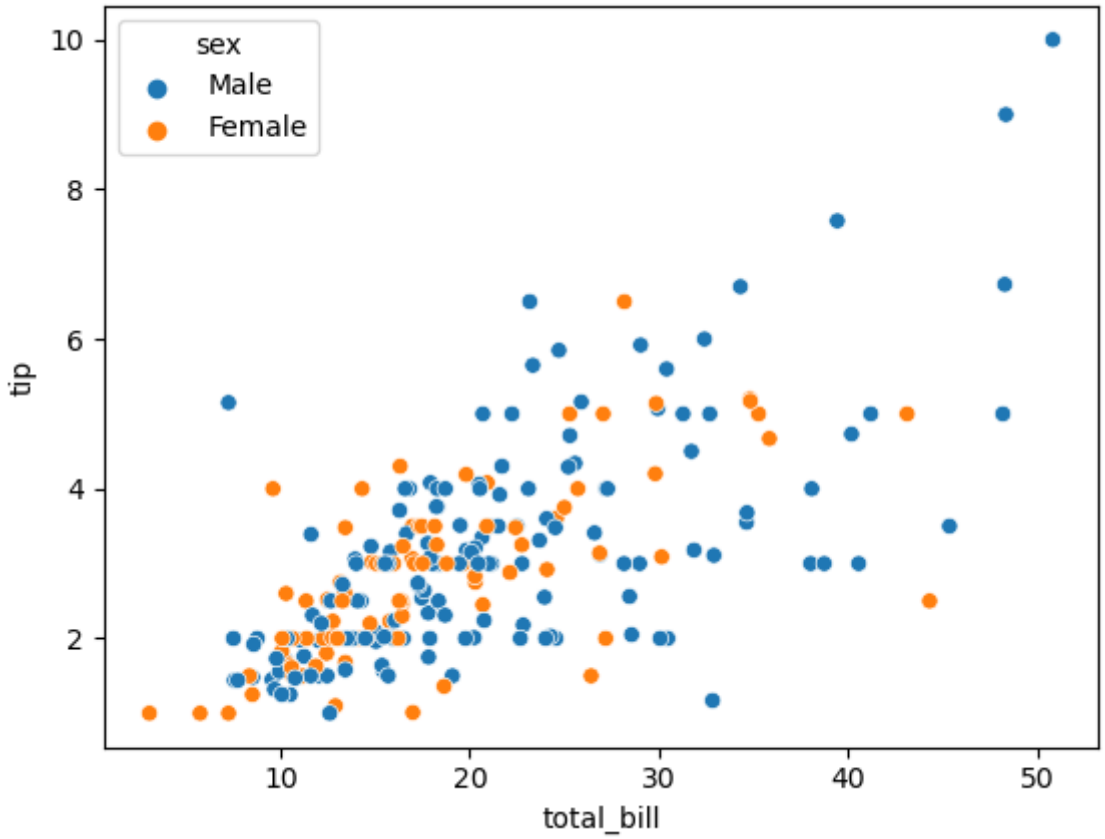
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   total_bill  244 non-null    float64
 1   tip         244 non-null    float64
 2   sex         244 non-null    category
 3   smoker      244 non-null    category
 4   day         244 non-null    category
 5   time        244 non-null    category
 6   size        244 non-null    int64
dtypes: category(4), float64(2), int64(1)
memory usage: 7.4 KB
```

1. Relationship between the total bill and the tip amount:

You can use a scatter plot to visualize the relationship between the total bill and the tip amount. This will help you understand the correlation between these two variables and the extent to which people tip based on the cost of their meal.

In [10]:
```
# plot a scatterplot to check the relationship between the variables 'tip' and
sns.scatterplot(x='total_bill', y='tip', data=df, hue='sex')
```

Out[10]:
```
<AxesSubplot:xlabel='total_bill', ylabel='tip'>
```



1.1 Strip Plot

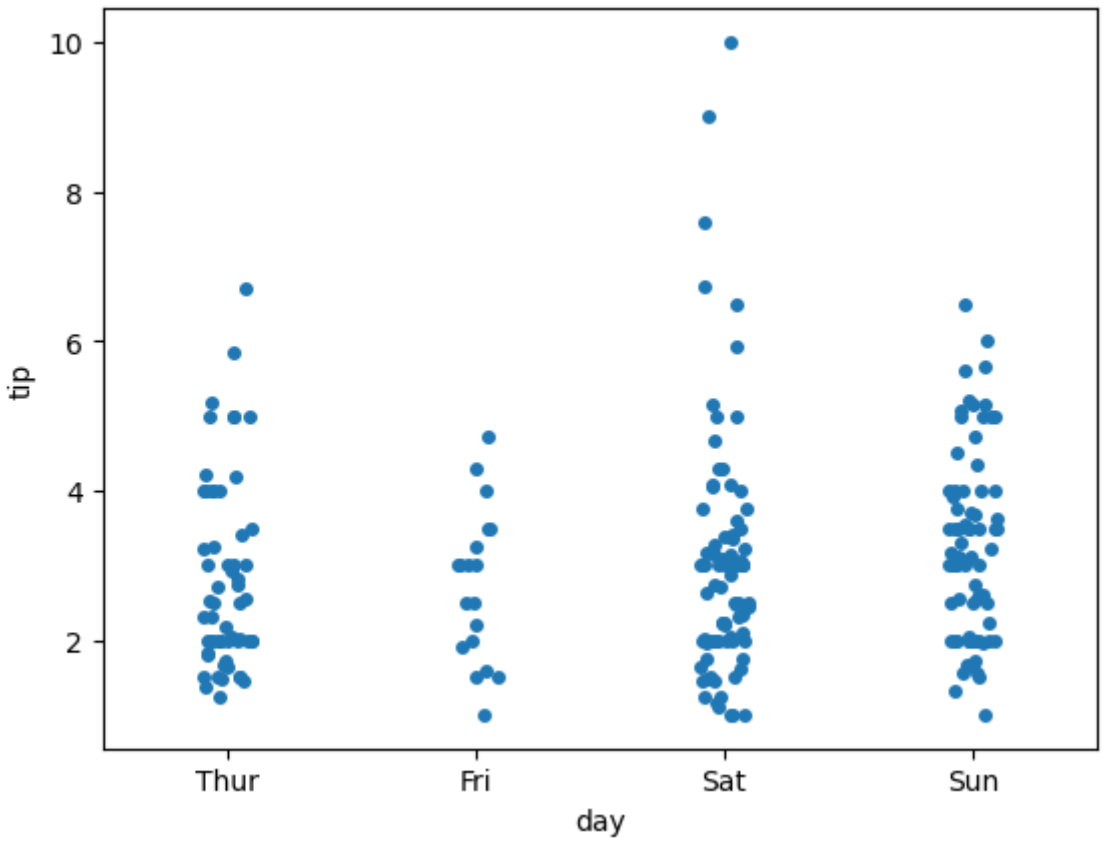
The strip plot resembles a scatterplot when one variable is categorical. This plot can help study the underlying distribution.

2. Average tip amount by time of day:

You can use a bar plot/strip plot to visualize the average tip amount by time of day (lunch or dinner). This will help you understand if people tip differently based on the time of day.

In [13]:
```
# Plot a strip plot to check the relationship between the variables 'tip' and
sns.stripplot(x='day', y='tip', data=df)
```

Out[13]:
```
<AxesSubplot:xlabel='day', ylabel='tip'>
```



Observations

- It can be seen that the tip amount is more at dinner time than at lunchtime.
- The plot shows that for most of the observations the tip amount is in the range 1 to 3 irrespective of the time.

Also plot:

3. Average tip amount by day of the week:

Average tip amount by day of the week: You can use a bar plot/Strip plot to visualize the average tip amount by day of the week. This will help you understand if people tip differently based on the day of the week.

4. Average tip amount by size of the party:

You can use a bar plot to visualize the average tip amount by the size of the party. This will help you understand if people tip differently based on the size of the group.

5. Average tip amount by smoker status:

You can use a bar plot to visualize the average tip amount by smoker status (smoker or non-smoker). This will help you understand if people tip differently based on their smoking status.

6. Average tip amount by gender:

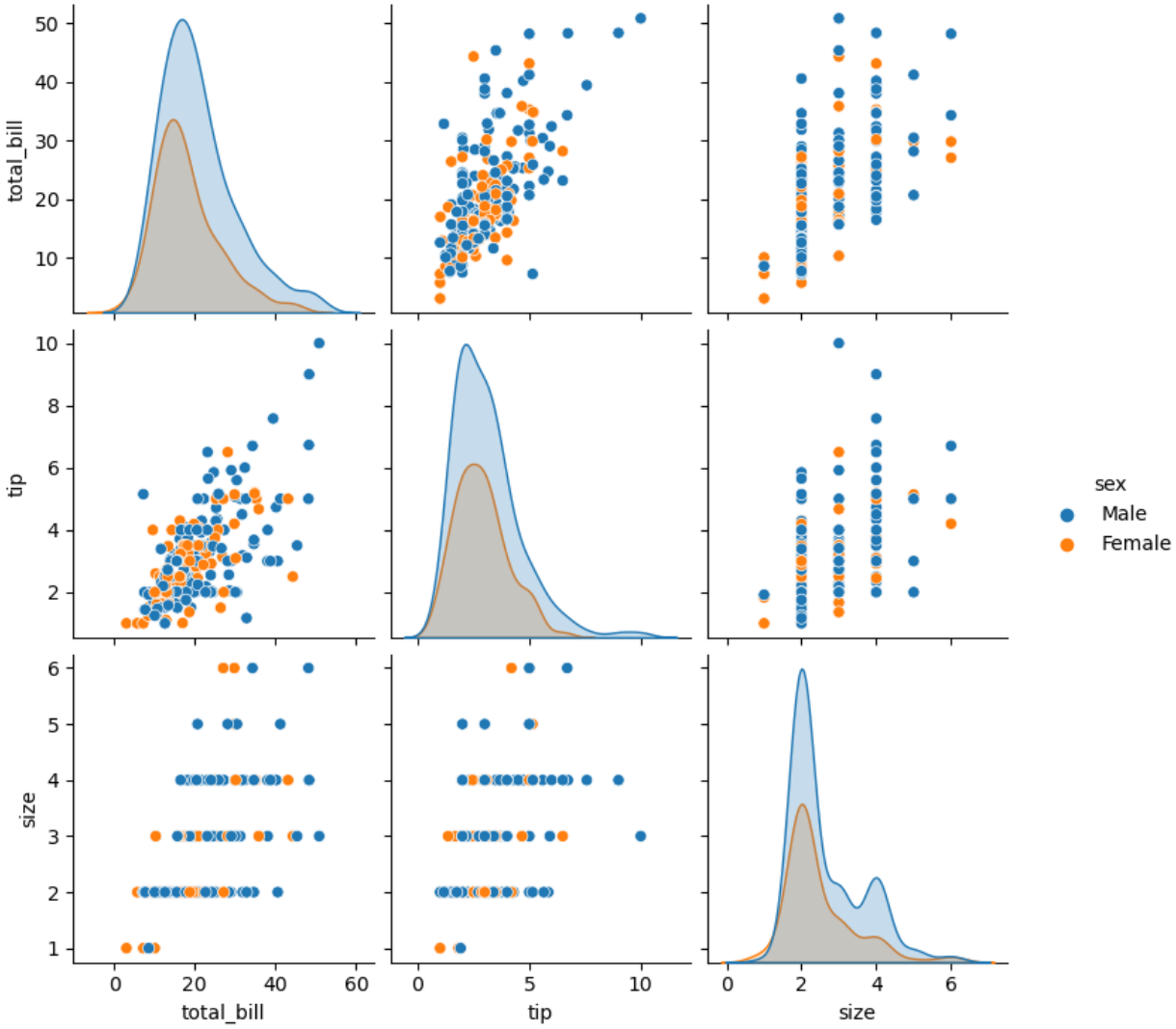
You can use a bar plot to visualize the average tip amount by gender. This will help you understand if people tip differently based on their gender.

1.2 Pair Plot

The pair plot gives a pairwise distribution of variables in the dataset. `pairplot()` function creates a matrix such that each grid shows the relationship between a pair of variables. On the diagonal axes, a plot shows the univariate distribution of each variable.

In [15]:
```
# Plot a pair plot for the tips dataset. Test the hue (by sex) argument
sns.pairplot(df, hue='sex')
```

Out[15]:
```
<seaborn.axisgrid.PairGrid at 0x7fd001614340>
```



Observations

The above plot shows the relationship between all the numerical variables. 'total_bill' and 'tip' has a positive linear relationship with each other. Also, 'total_bill' and 'tip' are positively skewed. 'size' has a significant impact on the 'total_bill', as the minimum bill amount is increasing with an increasing number of customers (size).

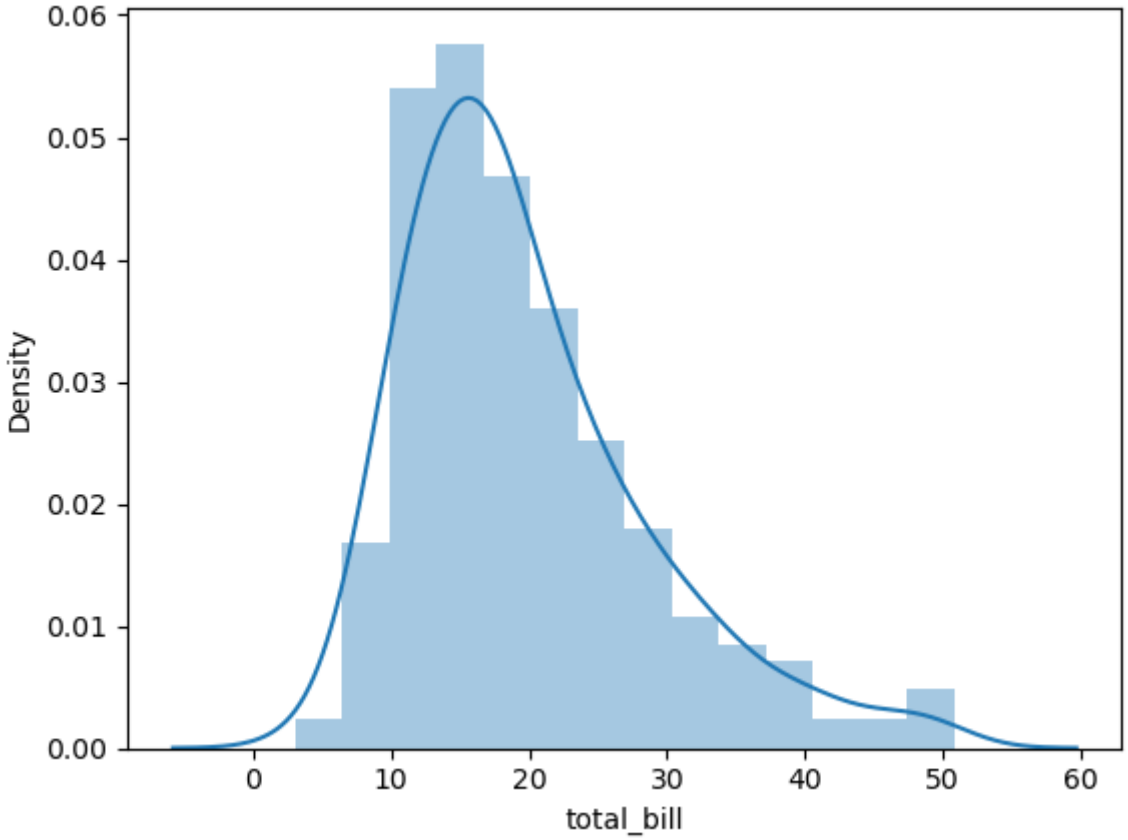
1.3 Distribution Plot

A seaborn provides a `displot()` function which is used to visualize a distribution of the univariate variable. This function uses matplotlib to plot a histogram and fit a kernel density estimate (KDE)

In [17]:
```
# plot a distribution plot of 'total_bill'
sns.distplot(df['total_bill'])
```

```
/var/folders/f3/h2kxxzmn2_14ktt8kxzyk2dr0000gn/T/ipykernel_24153/1675165828.p
y:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['total_bill'])
```

Out[17]:
```
<AxesSubplot:xlabel='total_bill', ylabel='Density'>
```



Observations

- We can interpret from the above plot that the total bill amount is between the range 10 to 20 for a large number of observations.

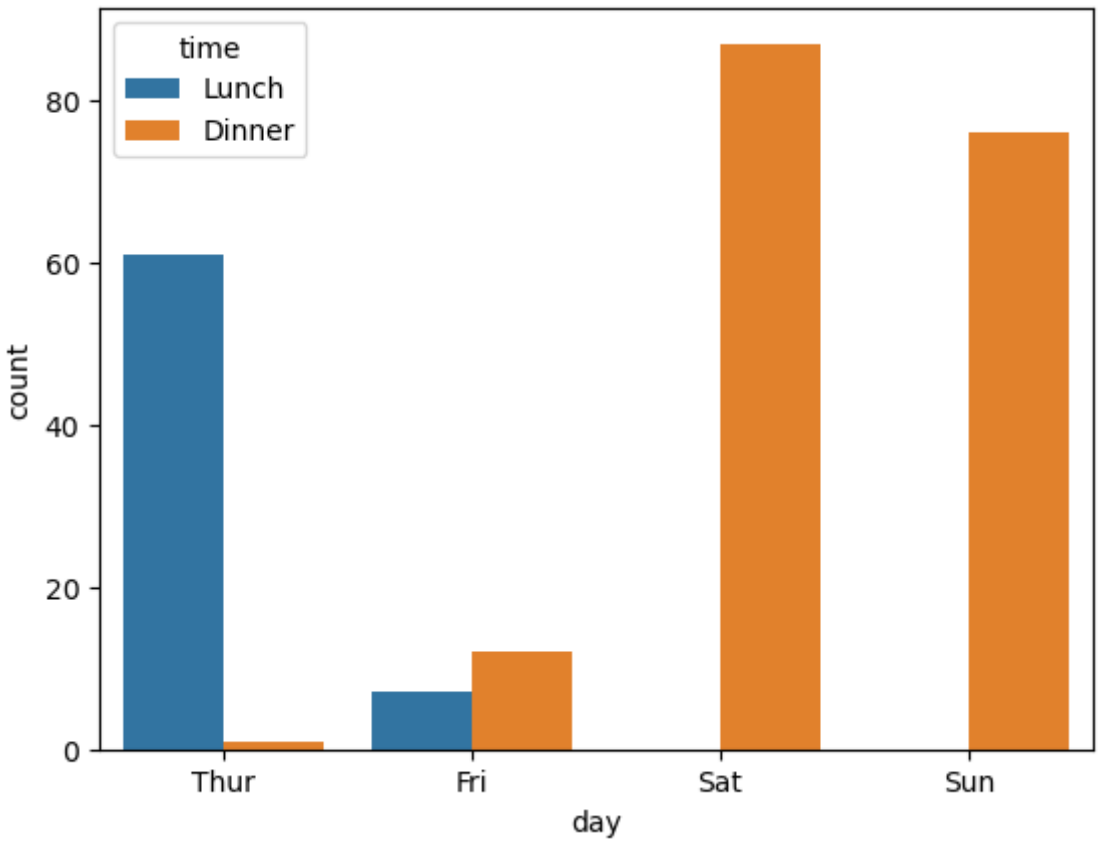
1.4 Count Plot

Count plot shows the count of observations in each category of a categorical variable. We can add another variable using a parameter 'hue'.

In [19]:
```
# count of observations for each day based on time
# set 'time' as hue parameter
```

```
sns.countplot(x='day', hue='time', data=df)
```

Out[19]:
```
<AxesSubplot:xlabel='day', ylabel='count'>
```



Observations

- All the observations recorded on Saturday and Sunday are for dinner. Observations for lunch on Thursday is highest among lunchtime.

1.5 Heatmap

Heatmap is a two-dimensional graphical representation of data where the individual values that are contained in a matrix are represented as colors. Each square in the heatmap shows the correlation between variables on each axis.

Compute correlation between the variables using `.corr()` function.

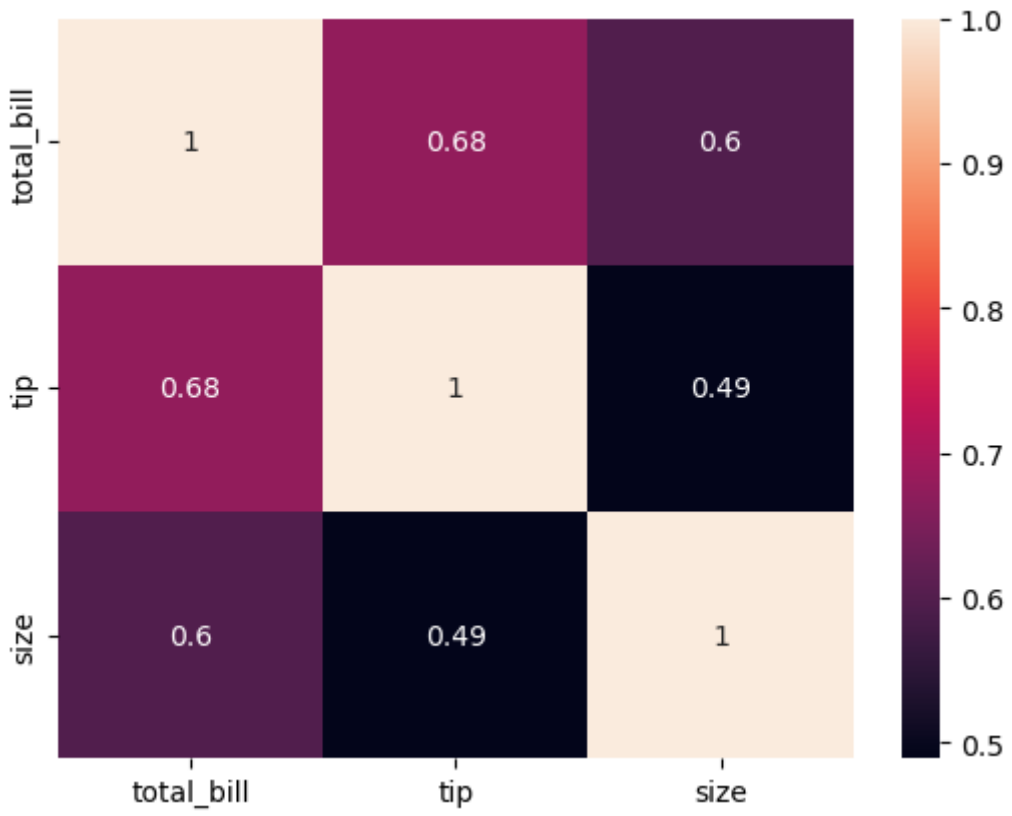
In [22]:
```
# Plot a heatmap of the correlation matrix.
df.corr()
```

Out[22]:

	total_bill	tip	size
total_bill	1.000000	0.675734	0.598315
tip	0.675734	1.000000	0.489299
size	0.598315	0.489299	1.000000

In [26]:
```
# plot heatmap
# 'annot=True' returns the correlation values, set cmap = 'YlGnBu'
sns.heatmap(df.corr(), annot=True)
```

Out[26]:
```
<AxesSubplot:>
```



Observations

- The above plot shows that there is a moderate correlation between 'total_bill' and 'tip' (0.68). The diagonal values are '1' as it is the correlation of the variable with itself.

Take home exercise

2. Matplotlib

- `matplotlib.pyplot` is a mostly used package because it is very simple to use and it generates plots in less time.

- `%matplotlib inline` helps plot the chart without having to use `plt.show( )` everytime. This is useful in Jupyter, but in IDE' like Spyder, we have to do `plt.show( )` to see the chart.

How to install Matplotlib?

1. You can use-
   `!pip install matplotlib`

In [22]:
```
# Import matplotlib
import matplotlib.pyplot as plt
```

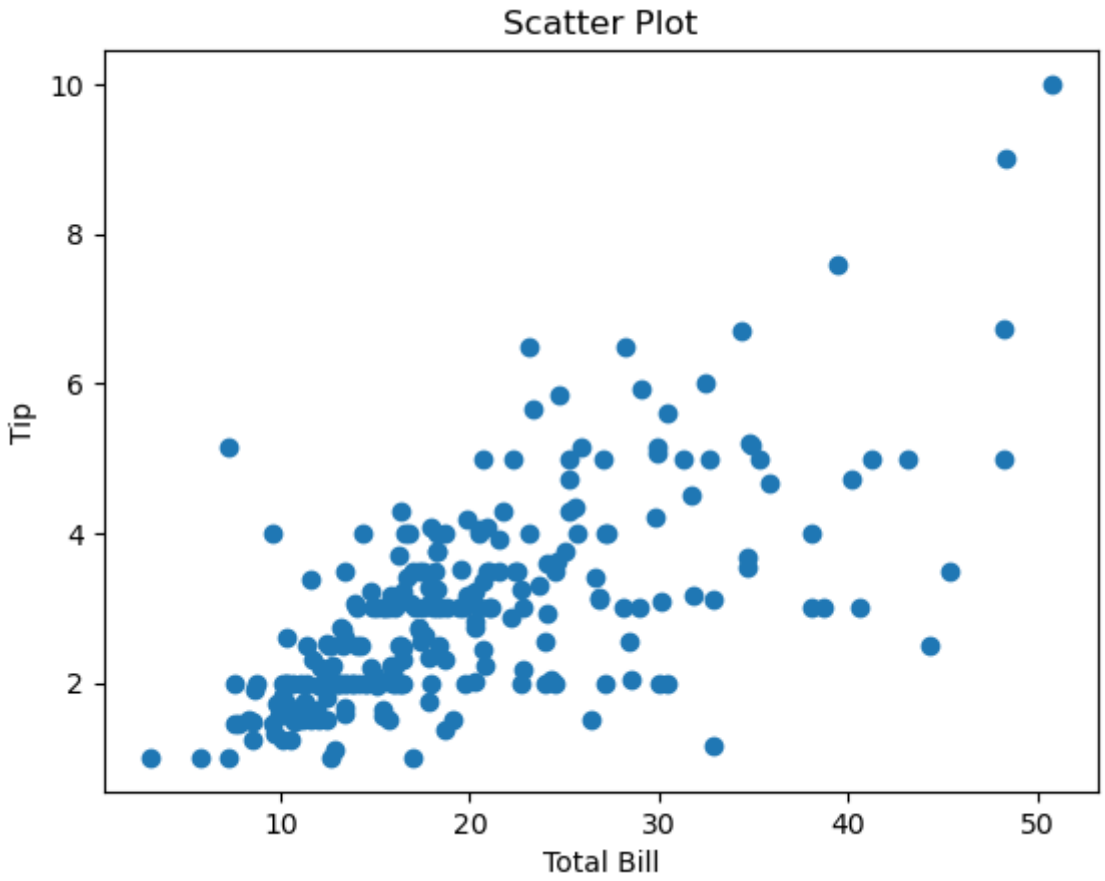
2.1 Scatter Plot

A scatter plot is a set of points plotted on horizontal and vertical axes. The scatter plot can be used to study the correlation between the two variables. One can also detect the extreme data points using a scatter plot.

Plot the scatter plot for the variables 'total_bill' and 'tip'

In [23]:
```
# plot a scatter plot
plt.scatter(df['total_bill'],df['tip'])

# add the axes labels to the plot
plt.title('Scatter Plot')
plt.xlabel('Total Bill')
plt.ylabel ('Tip')
```

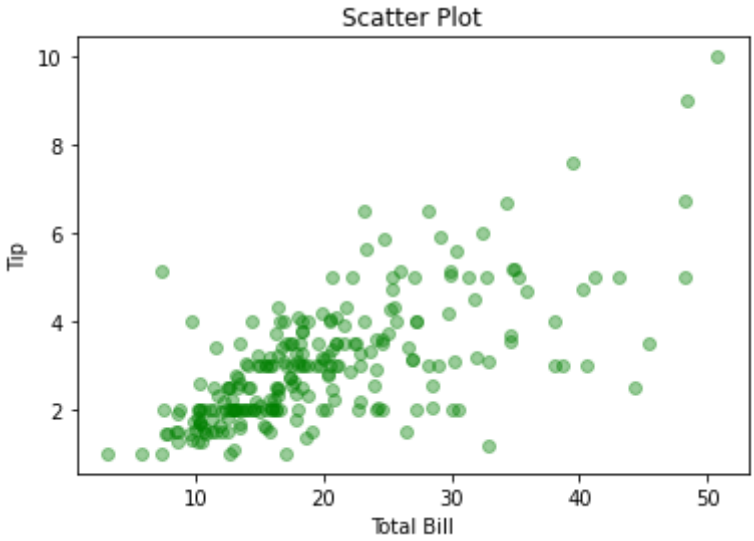
Out[23]:
```
Text(0, 0.5, 'Tip')
```



We can add different colors, opacity, and shape of data points. Let's add these customizations in the above plot.

In [10]:
```
x = data['total_bill']
y = data['tip']

# plot a scatter plot
plt.scatter(x,y, c = 'green', alpha = 0.4 )

# add the axes labels to the plot
plt.title('Scatter Plot')
plt.xlabel('Total Bill')
plt.ylabel ('Tip')
```

Out[10]:
```
Text(0, 0.5, 'Tip')
```



2.2 Bar Plot

- A bar plot is used to display categorical data with bars with lengths proportional to the values that they represent. The comparison between different categories of a categorical variable can be done by studying a bar plot.
- In the vertical bar plot, the X-axis displays the categorical variable and Y-axis contains the values corresponding to different categories.

In [11]:
```
data.head()
```

Out[11]:

	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

In [13]:
```
# the variable 'smoker' is categorical
# check categories in the variable
set(data['smoker']), set(data['day'])
```

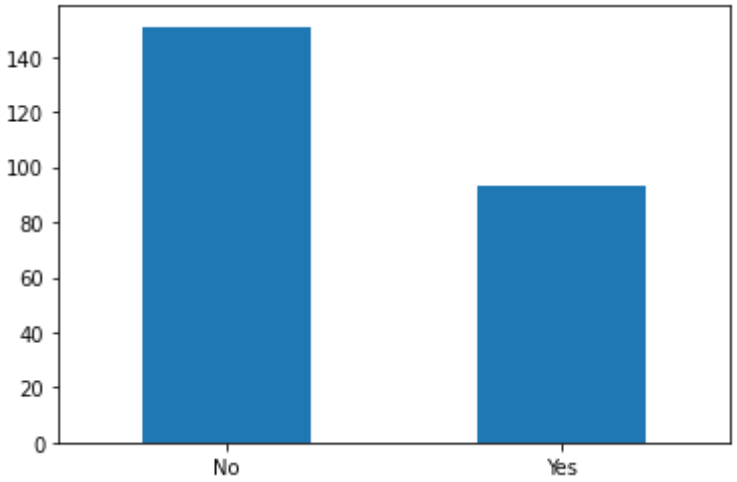
Out[13]:
```
({'No', 'Yes'}, {'Fri', 'Sat', 'Sun', 'Thur'})
```

In [16]:
```
# bar plot to get the count of smokers and non-smokers in the data
data['smoker'].value_counts()
```

Out[16]:
```
No     151
Yes     93
Name: smoker, dtype: int64
```

In [23]:
```
data['smoker'].value_counts().plot(kind = 'bar', rot = 0)
```

Out[23]:
```
<AxesSubplot:>
```



Observations

- From the bar plot, it can be interpreted that the proportion of non-smokers is more in the data

2.3 Pie Plot

Pie plot is a graphical representation of univariate data. It is a circular graph divided into slices displaying the numerical proportion. For the categorical variable, each slice of the pie plot corresponds to each of the categories.

In [24]:
```
data.head()
```

Out[24]:

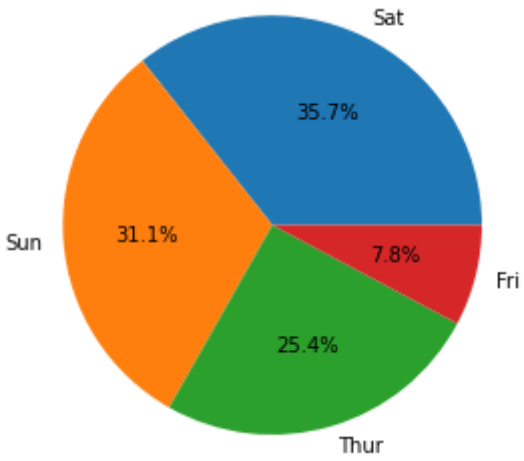
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [25]:
```
# categories in the 'day' variable
data['day'].value_counts()
```

Out[25]:
```
Sat     87
Sun     76
Thur    62
Fri     19
Name: day, dtype: int64
```

In [63]:
```
# plot the occurrence of different days in the dataset
```

```
# 'autopct' displays the percentage upto 1 decimal place
# 'radius' sets the radius of the pie plot
plt.pie(data.day.value_counts(), autopct = '%.1f%%', radius = 1.2, labels = ['S


# display the plot
plt.show()
```

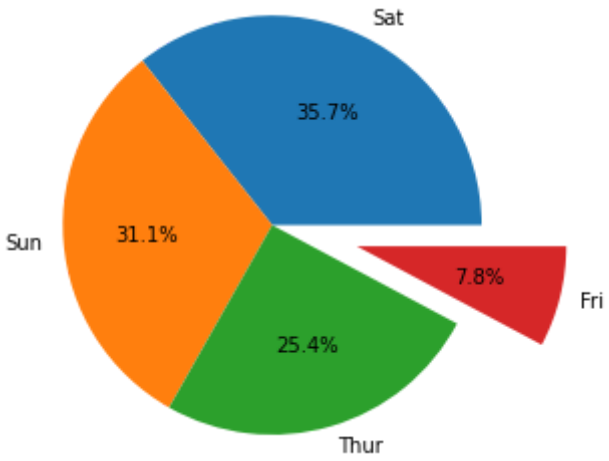


Observations

- From the above pie plot, it can be seen that the data has a high proportion for Saturday followed by Sunday.

Exploded pie plot is a plot in which one or more sectors are separated from the disc. Use `explode = [0,0,0,0.5]`

In [64]:
```
# plot the occurrence of different days in the dataset

# 'autopct' displays the percentage upto 1 decimal place
# 'radius' sets the radius of the pie plot
plt.pie(data.day.value_counts(), autopct = '%.1f%%', radius = 1.2, labels = ['S


# display the plot
plt.show()
```



2.4 Histogram

A histogram is used to display the distribution and spread of the continuous variable. One axis represents the range of variable and the other axis shows the frequency of the data points.

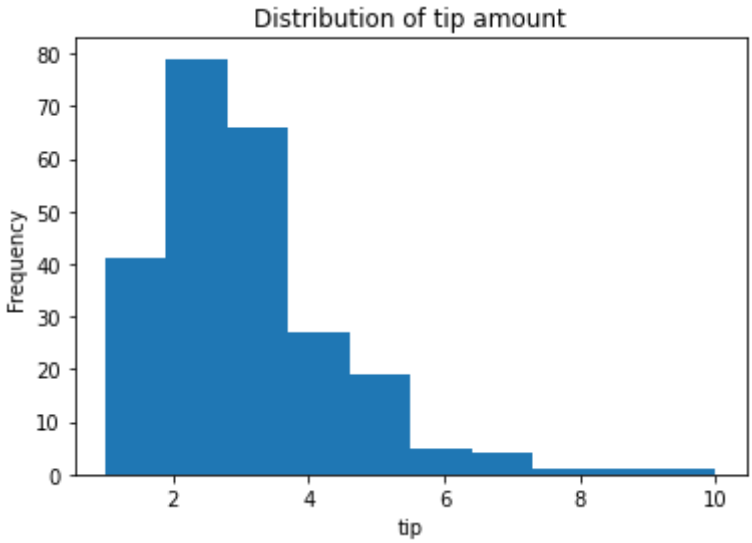
In tips dataset, 'tip' is the continuous variable. Let's plot the histogram to understand the distribution of the variable.

In [34]:
```
data.head()
```

Out[34]:

	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

In [65]:
```
# plot the histogram
plt.hist(data['tip'])

# add the graph title and axes labels
plt.title('Distribution of tip amount')
plt.xlabel('tip')
plt.ylabel('Frequency')

# display the plot
plt.show()
```



Observations

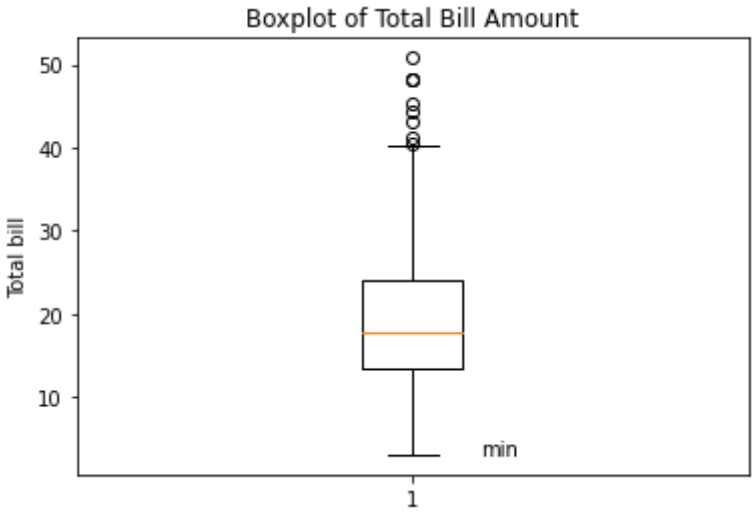
- From the above plot, we can see that the tip amount is positively skewed.

2.5 Box Plot

Boxplot is a way to visualize the five-number summary of the variable. The five-number summary includes the numerical quantities like minimum, first quartile (Q1), median (Q2), third quartile (Q3), and maximum. Boxplot gives information about the outliers in the data. Detecting and removing outliers is one of the most important steps in exploratory data analysis. Boxplots also tells about the distribution of the data.

Plot the boxplot of `'total_bill'` to check the distribution and presence of outliers in the variable.

In [66]:
```
# plot a distribution of total bill
plt.boxplot(data['total_bill'])

# sns.boxplot(x='sex', y='tip', data=df)

# add labels for five number summary
plt.text(x = 1.1, y = data['total_bill'].min(), s ='min')
# plt.text(x = 1.1, y = data.total_bill.quantile(0.25), s ='Q1')
# plt.text(x = 1.1, y = data['total_bill'].median(), s ='meadian (Q2)')
# plt.text(x = 1.1, y = data.total_bill.quantile(0.75), s ='Q3')
# plt.text(x = 1.1, y = data['total_bill'].max(), s ='max')


# add the graph title and axes labels
plt.title('Boxplot of Total Bill Amount')
plt.ylabel('Total bill')

# display the plot
plt.show()
```



2.6 Subplots

`pyplot.subplot()` is a function in the matplotlib library that creates multiple subplots in a single figure. It takes three integer arguments: `numrows`, `numcols`, and

`plot_number` where `plot_number` ranges from 1 to numrows * numcols

In [ ]:

```
# If you need multiple plots in the same canvas
# plt.subplot(no. of rows, no. of columns, position of the chart)
plt.figure(figsize = (10,10))

plt.subplot(2,2,1)
sns.heatmap(data.corr(), annot =True, cmap = 'YlGnBu')

plt.subplot(2,2,2)
sns.countplot(x = 'day', data=data, hue='time')

plt.subplot(2,2,3)
sns.distplot(data['total_bill'],hist=False)
```