# Name – Mahek Dosi

# PRN - 202201040179

Batch – H4

Roll No - 874

import pandas as pd import numpy as np

import matplotlib.pyplot as plt

from pandas import Series, DataFrame

# Reading the tips.csv file df1=pd.read\_csv('/content/tips.csv')

df1.head()

1 to 5 of 5 entries 

Filter

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **index** | **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 | 3.5 | 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 |

Show per page

25

Like what you see? Visit the [data table notebook](https://colab.research.google.com/notebooks/data_table.ipynb) to learn more about interactive tables.



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| df1.tail() | **total\_bill** | **tip** | **sex** | **smoker** | **day** | **time** | **size** |
| **239** | 29.03 | 5.92 | Male | No | Sat | Dinner | 3 |
| **240** | 27.18 | 2.00 | Female | Yes | Sat | Dinner | 2 |
| **241** | 22.67 | 2.00 | Male | Yes | Sat | Dinner | 2 |
| **242** | 17.82 | 1.75 | Male | No | Sat | Dinner | 2 |
| **243** | 18.78 | 3.00 | Female | No | Thur | Dinner | 2 |

df1.columns

Index(['total\_bill', 'tip', 'sex', 'smoker', 'day', 'time', 'size'], dtype='object')

df1.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 244 entries, 0 to 243 Data columns (total 7 columns):

# Column Non-Null Count Dtype

---

------

--------------

-----

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

df1.describe()

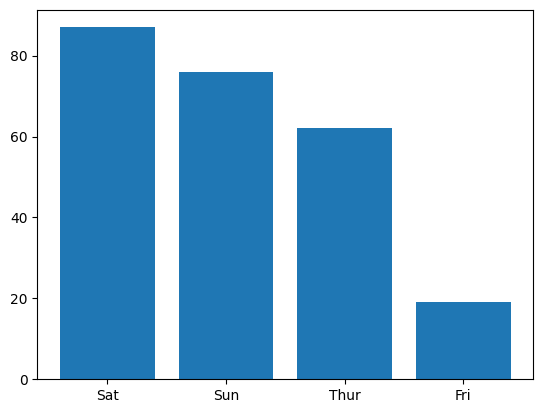
**50%** 17.795000 2.900000 2.000000

**75%** 24.127500 3.562500 3.000000

**max** 50.810000 10.000000 6.000000



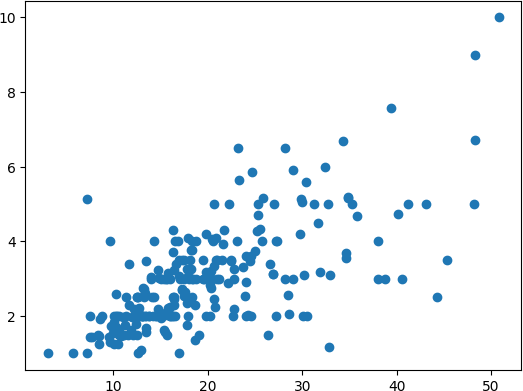
|  |  |  |  |
| --- | --- | --- | --- |
|  | **total\_bill** | **tip** | **size** |
| **count** | 244.000000 | 244.000000 | 244.000000 |
| **mean** | 19.785943 | 2.998279 | 2.569672 |
| **std** | 8.902412 | 1.383638 | 0.951100 |
| **min** | 3.070000 | 1.000000 | 1.000000 |
| **25%** | 13.347500 | 2.000000 | 2.000000 |



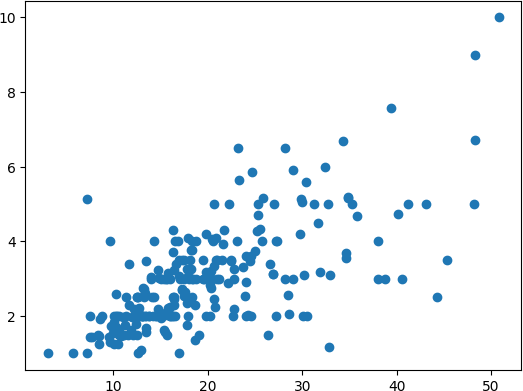
a=pd.DataFrame(df1['day'].value\_counts()) a.reset\_index(inplace=True) plt.bar(a['index'],a['day'])

<BarContainer object of 4 artists>

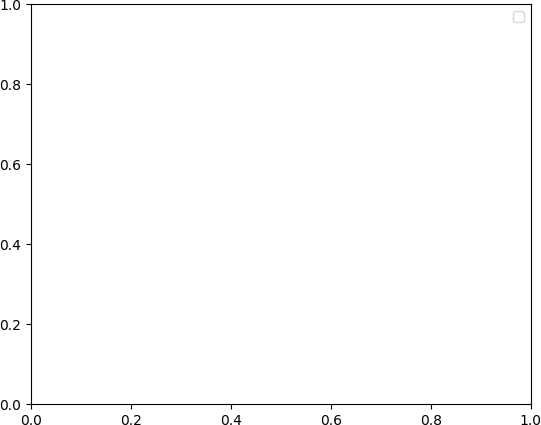
plt.scatter(df1['total\_bill'],df1['tip']) plt.show()



plt.scatter(x='total\_bill',y='tip',data=df1) fig=plt.figure(figsize=(5,4)) ax=fig.add\_axes([1,1,1,1]) ax.legend(labels=('sun','mon','tue')) plt.show()



#Different types of Matplotlib Plots #bar chart



import matplotlib.pyplot as plt import pandas as pd

# Reading the tips.csv file

data = pd.read\_csv('/content/tips.csv')

# initializing the data x = data['day']

y = data['total\_bill']

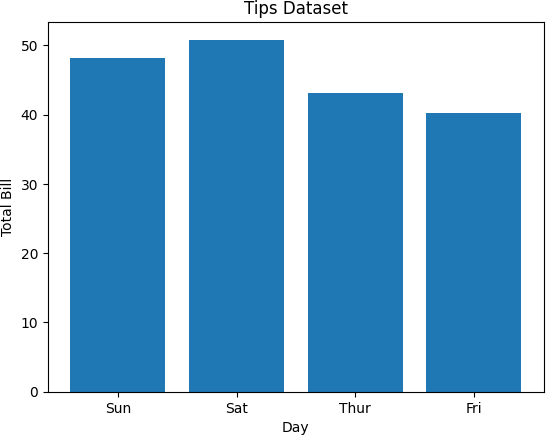
# plotting the data plt.bar(x, y)

# Adding title to the plot plt.title("Tips Dataset")

# Adding label on the y-axis plt.ylabel('Total Bill')

# Adding label on the x-axis plt.xlabel('Day')

plt.show()



Customization that is available for the Bar Chart – color: For the bar faces

edgecolor: Color of edges of the bar linewidth: Width of the bar edges width: Width of the bar

import matplotlib.pyplot as plt import pandas as pd

# initializing the data x = data['day']

y = data['total\_bill']

# plotting the data

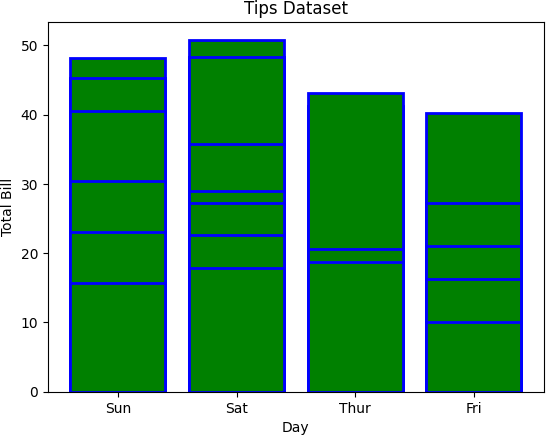
plt.bar(x, y, color='green', edgecolor='blue', linewidth=2)

# Adding title to the plot plt.title("Tips Dataset")

# Adding label on the y-axis plt.ylabel('Total Bill')

# Adding label on the x-axis plt.xlabel('Day')

plt.show()



## Histogram A histogram is basically used to represent data provided in a form of some groups. It is a type of bar plot where the X-axis represents the bin ranges while the Y-axis gives information about frequency. The hist() function is used to compute and create histogram of x.

import matplotlib.pyplot as plt import pandas as pd

# initializing the data x = data['total\_bill']

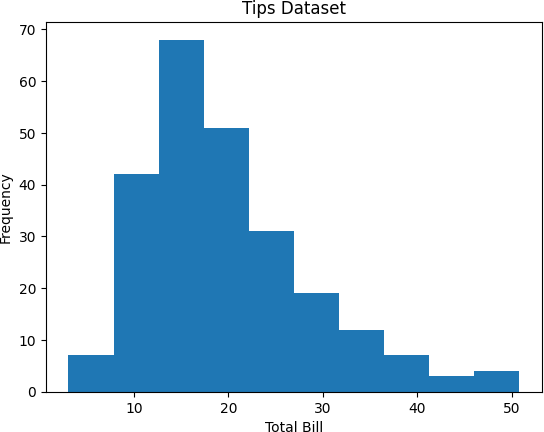
# plotting the data plt.hist(x)

# Adding title to the plot plt.title("Tips Dataset")

# Adding label on the y-axis plt.ylabel('Frequency')

# Adding label on the x-axis plt.xlabel('Total Bill')

plt.show()



## Customization that is available for the Histogram –

bins: Number of equal-width bins color: For changing the face color edgecolor: Color of the edges linestyle: For the edgelines alpha: blending value, between 0 (transparent) and 1 (opaque)

import matplotlib.pyplot as plt import pandas as pd

# initializing the data x = data['total\_bill']

# plotting the data

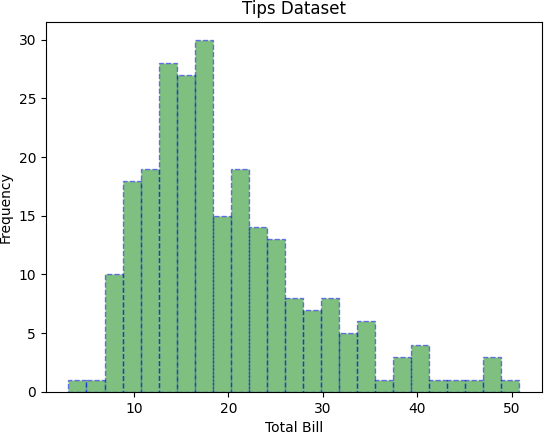
plt.hist(x, bins=25, color='green', edgecolor='blue', linestyle='--', alpha=0.5)

# Adding title to the plot plt.title("Tips Dataset")

# Adding label on the y-axis plt.ylabel('Frequency')

# Adding label on the x-axis plt.xlabel('Total Bill')

plt.show()



## Scatter Plot Scatter plots are used to observe relationships between variables. The scatter() method in the matplotlib library is used to draw a scatter plot.

import matplotlib.pyplot as plt import pandas as pd

# initializing the data x = data['day']

y = data['total\_bill']

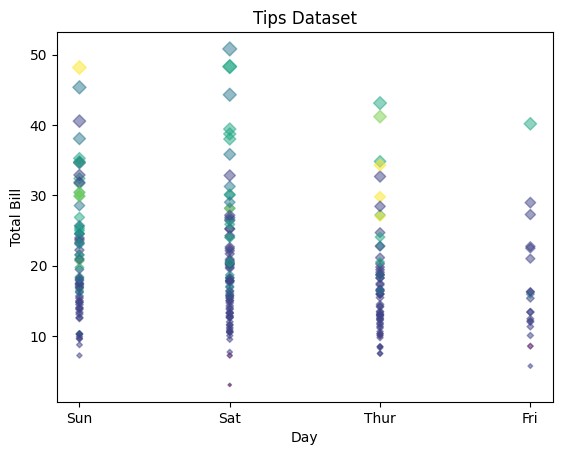
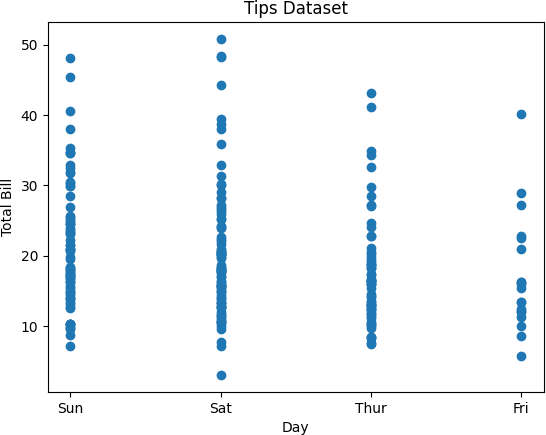
# plotting the data plt.scatter(x, y)

# Adding title to the plot plt.title("Tips Dataset")

# Adding label on the y-axis plt.ylabel('Total Bill')

# Adding label on the x-axis plt.xlabel('Day')

plt.show()



## Customizations that are available for the scatter plot are –

s: marker size (can be scalar or array of size equal to size of x or y) c: color of sequence of colors for markers

## marker: marker style

linewidths: width of marker border edgecolor: marker border color

## alpha: blending value, between 0 (transparent) and 1 (opaque)

import matplotlib.pyplot as plt import pandas as pd

# initializing the data x = data['day']

y = data['total\_bill']

# plotting the data

plt.scatter(x, y, c=data['size'], s=data['total\_bill'], marker='D', alpha=0.5)

# Adding title to the plot plt.title("Tips Dataset")

# Adding label on the y-axis plt.ylabel('Total Bill')

# Adding label on the x-axis plt.xlabel('Day')

plt.show()

## Pie Chart Pie chart is a circular chart used to display only one series of data. The area of slices of the pie represents the percentage of the parts of the data. The slices of pie are called wedges. It can be created using the pie() method.

import matplotlib.pyplot as plt import pandas as pd

# initializing the data

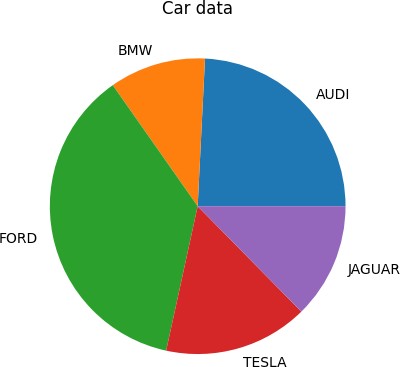
cars = ['AUDI', 'BMW', 'FORD', 'TESLA', 'JAGUAR',]

data = [23, 10, 35, 15, 12]

# plotting the data plt.pie(data, labels=cars)

# Adding title to the plot plt.title("Car data")

plt.show()



## Customizations that are available for the Pie chart are –

explode: Moving the wedges of the plot autopct: Label the wedge with their numerical value. color: Attribute is used to provide color to the wedges. shadow: Used to create shadow of wedge.

import matplotlib.pyplot as plt import pandas as pd

# initializing the data

cars = ['AUDI', 'BMW', 'FORD', 'TESLA', 'JAGUAR',]

data = [23, 13, 35, 15, 12]

explode = [0.1, 0.5, 0, 0, 0]

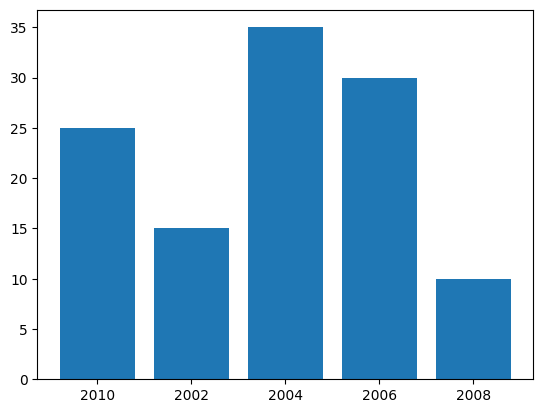
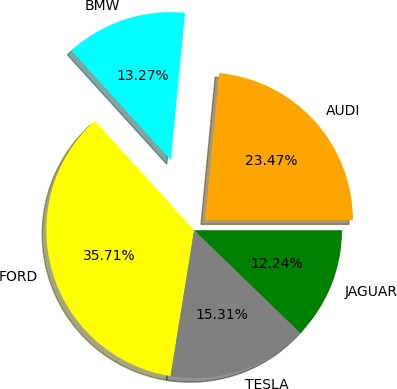
colors = ( "orange", "cyan", "yellow", "grey", "green",)

# plotting the data

plt.pie(data, labels=cars, explode=explode, autopct='%1.2f%%', colors=colors, shadow=True)

plt.show()

Saving a Plot



## For saving a plot in a file on storage disk, savefig() method is used. A file can be saved in many formats like .png, .jpg, .pdf, etc.

import matplotlib.pyplot as plt

# Creating data

year = ['2010', '2002', '2004', '2006', '2008']

production = [25, 15, 35, 30, 10]

# Plotting barchart plt.bar(year, production)

# Saving the figure. plt.savefig("output.jpg")

# Saving figure by changing parameter values plt.savefig("output1", facecolor='y', bbox\_inches="tight",

pad\_inches=0.3, transparent=True)

0s completed at 2:44 PM