Matplotlib :

- Matplotlib is a popular Python library used for creating static, interactive, and animated visualizations in various formats.
- It provides a wide range of tools for visualizing data, from simple line plots to complex 3D visualizations.
- Matplotlib is particularly useful for data exploration, analysis, and presentation.

Installing Matplotlib :

```
pip install matplotlib
```

In [2]:
```
!pip install matplotlib
```

...

In [1]:
```
import matplotlib
```

Some common charts that can be genereted using .

1. `Line Plot:` Display data points connected by lines to show trends.
2. `Scatter Plot:` Display individual data points as dots, useful for spotting relationships.
3. `Bar Plot:` Show categorical data using rectangular bars, good for comparisons.
4. `Histogram:` Visualize frequency distribution of continuous data.
5. `Pie Chart:` Display proportions of parts within a whole.
6. `Box Plot:` Visualize data distribution using quartiles.
7. `Heatmap:` Use colors to represent data values in a 2D matrix.
8. `Subplots:` Create multiple plots within a single figure.

In [1]:
```
# NUMERICAL DATA
# CATEGORICAL DATA
```

In [2]:
```
# import the necessities
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
train = pd.read_csv("Data/train.csv")
test = pd.read_csv("Data/test.csv")
df1 = pd.concat([train, test], ignore_index=True)
```

In [4]:
```
df1.head()
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Em
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	
4	5	0.0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	

In [5]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  1309 non-null   int64  
 1   Survived     891 non-null    float64
 2   Pclass       1309 non-null   int64  
 3   Name         1309 non-null   object 
 4   Sex          1309 non-null   object 
 5   Age          1046 non-null   float64
 6   SibSp        1309 non-null   int64  
 7   Parch        1309 non-null   int64  
 8   Ticket       1309 non-null   object 
 9   Fare         1308 non-null   float64
 10  Cabin        295 non-null    object 
 11  Embarked     1307 non-null   object 
dtypes: float64(3), int64(4), object(5)
memory usage: 122.8+ KB
```

In [6]:
```
# Loading 4th data
df4 = pd.read_csv("Data/sample.csv")
```

In [7]:
```
df4.head()
```

Out[7]:

	name	sex	age	height	weight
0	Aubrey	M	41.0	74	170
1	Ron	M	42.0	68	166
2	Carl	M	32.0	70	155
3	Antonio	M	39.0	72	167
4	Deborah	F	30.0	66	124

In [38]:
```
# Load data from seaborn
iris_data = sns.load_dataset("iris")
tips_data = sns.load_dataset("tips")
```

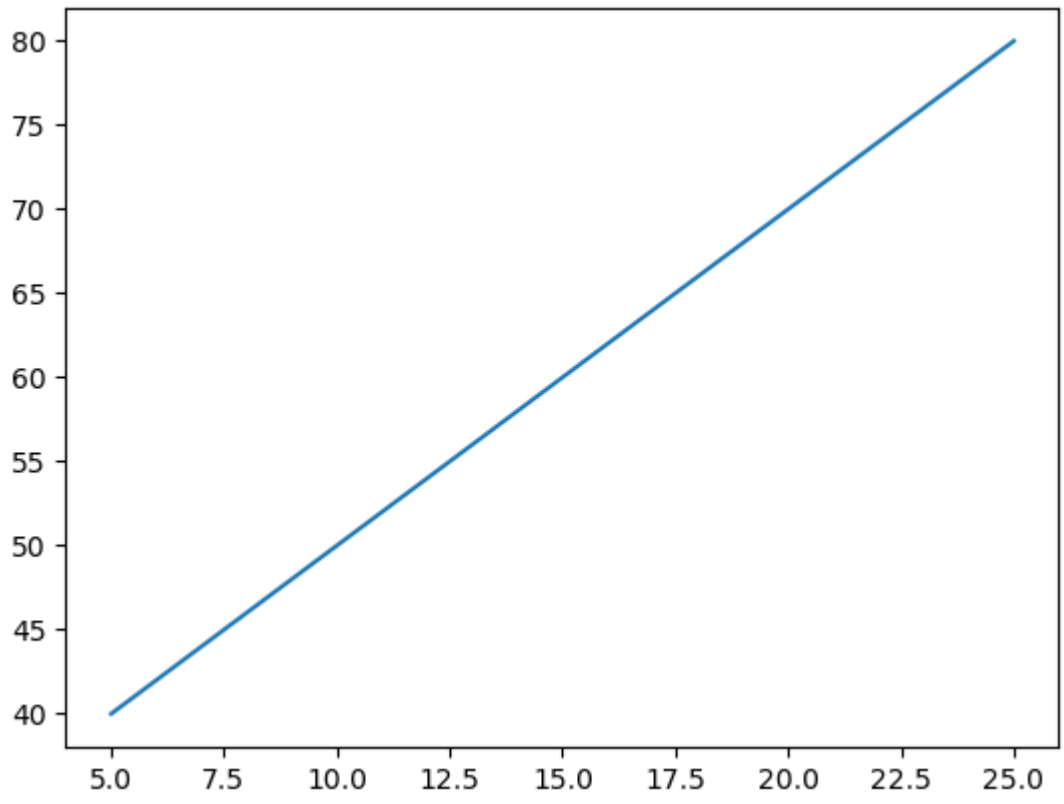
Line Plot :

- A line plot displays data points connected by lines, suitable for showing trends over continuous data points.
- Line plots are particularly useful for depicting data that is ordered chronologically or sequentially.
- General syntax:-

```
plt.plot(x, y)
```

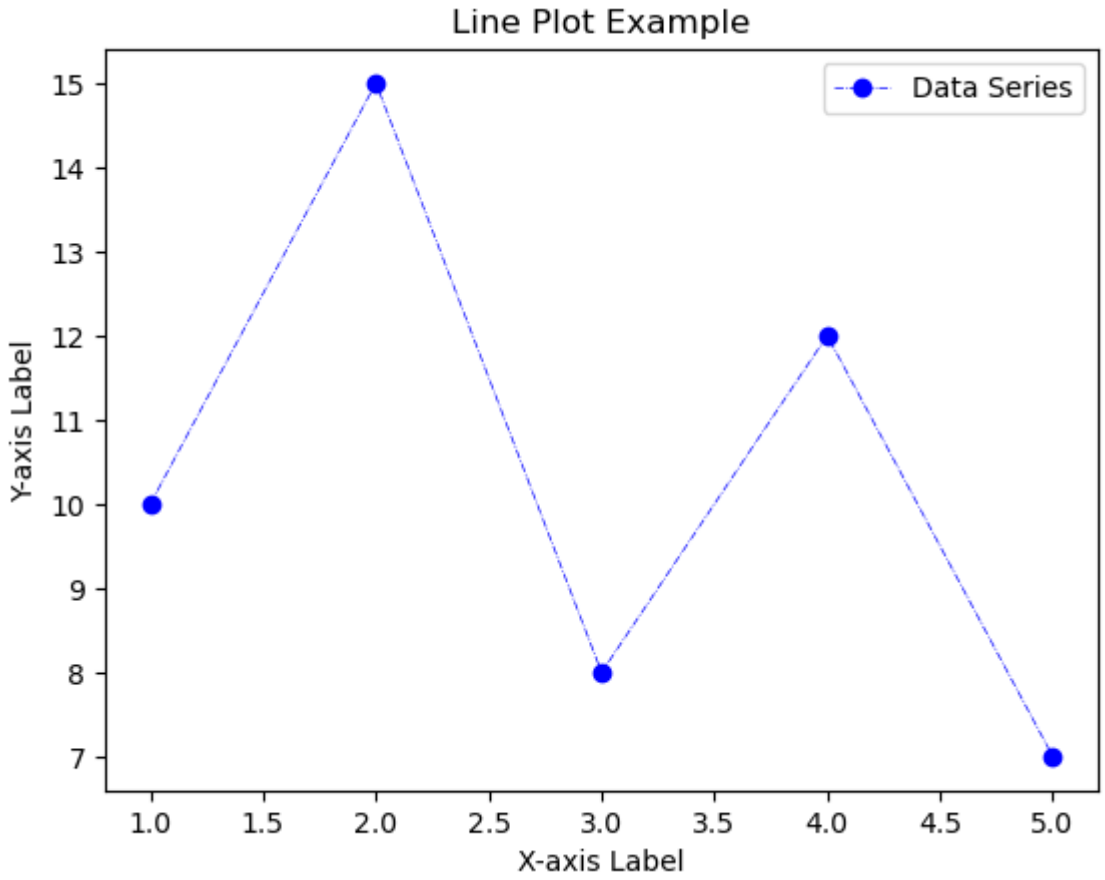
In [8]:
```
x = [5,10,15,20,25]
y = [40,50,60,70,80]
```

In [9]:
```
plt.plot(x,y);
```



In [10]:

```
# Sample data
x_values = [1, 2, 3, 4, 5]
y_values = [10, 15, 8, 12, 7]


plt.plot(x_values, y_values, marker='o',
         linestyle='dashdot', color='b',
         label='Data Series', linewidth = 0.5)
plt.xlabel('X-axis Label')
plt.ylabel('Y-axis Label')
plt.title('Line Plot Example')
plt.legend()
plt.show();

```



In [11]:

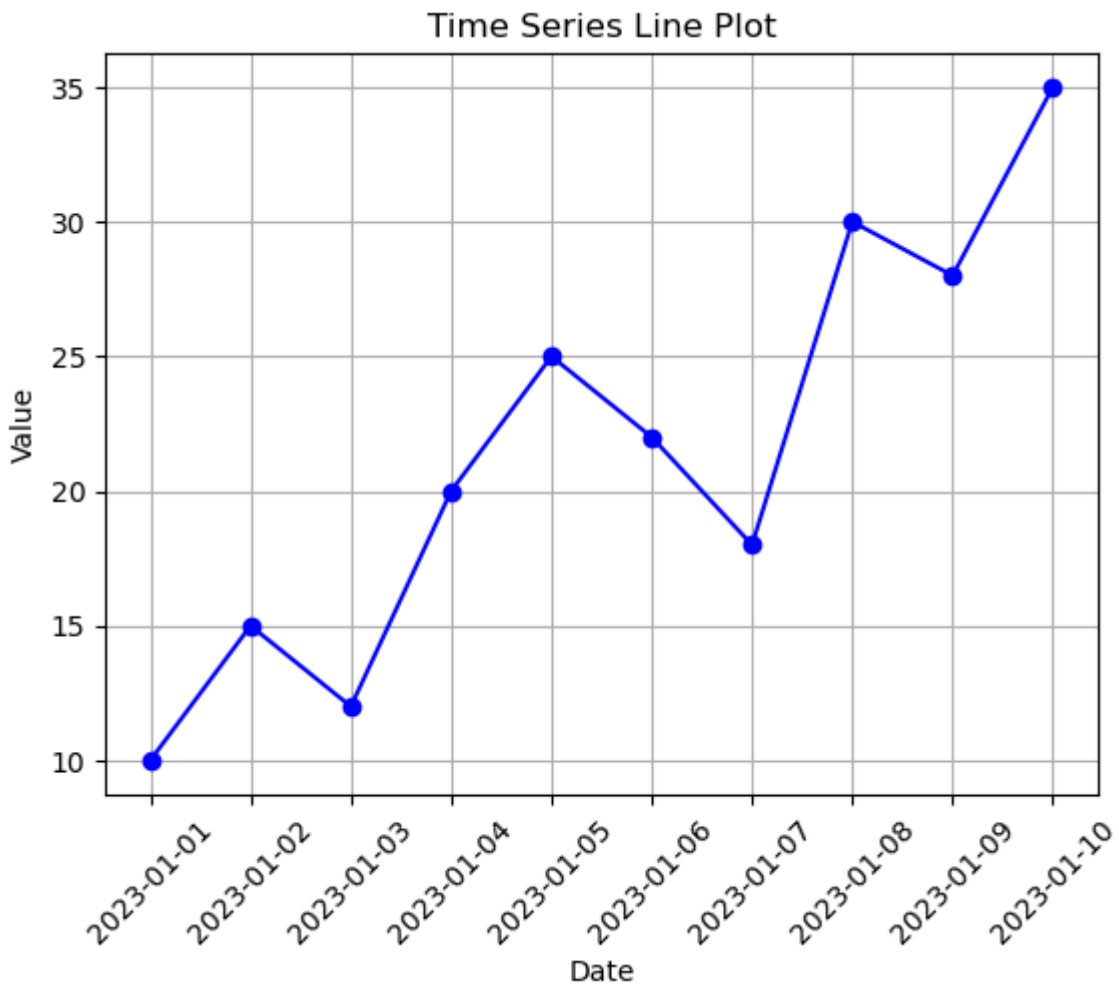
```
data = {
    'date': pd.date_range(start='2023-01-01', periods=10, freq='D'),
    'value': [10, 15, 12, 20, 25, 22, 18, 30, 28, 35]
}
time_series = pd.DataFrame(data)
time_series
```

Out[11]:

	date	value
0	2023-01-01	10
1	2023-01-02	15
2	2023-01-03	12
3	2023-01-04	20
4	2023-01-05	25
5	2023-01-06	22
6	2023-01-07	18
7	2023-01-08	30
8	2023-01-09	28
9	2023-01-10	35

In [54]:

```
plt.plot(time_series['date'], time_series['value'], marker="o", linestyle="-", c
plt.title('Time Series Line Plot')
plt.xlabel('Date')
plt.ylabel('Value')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

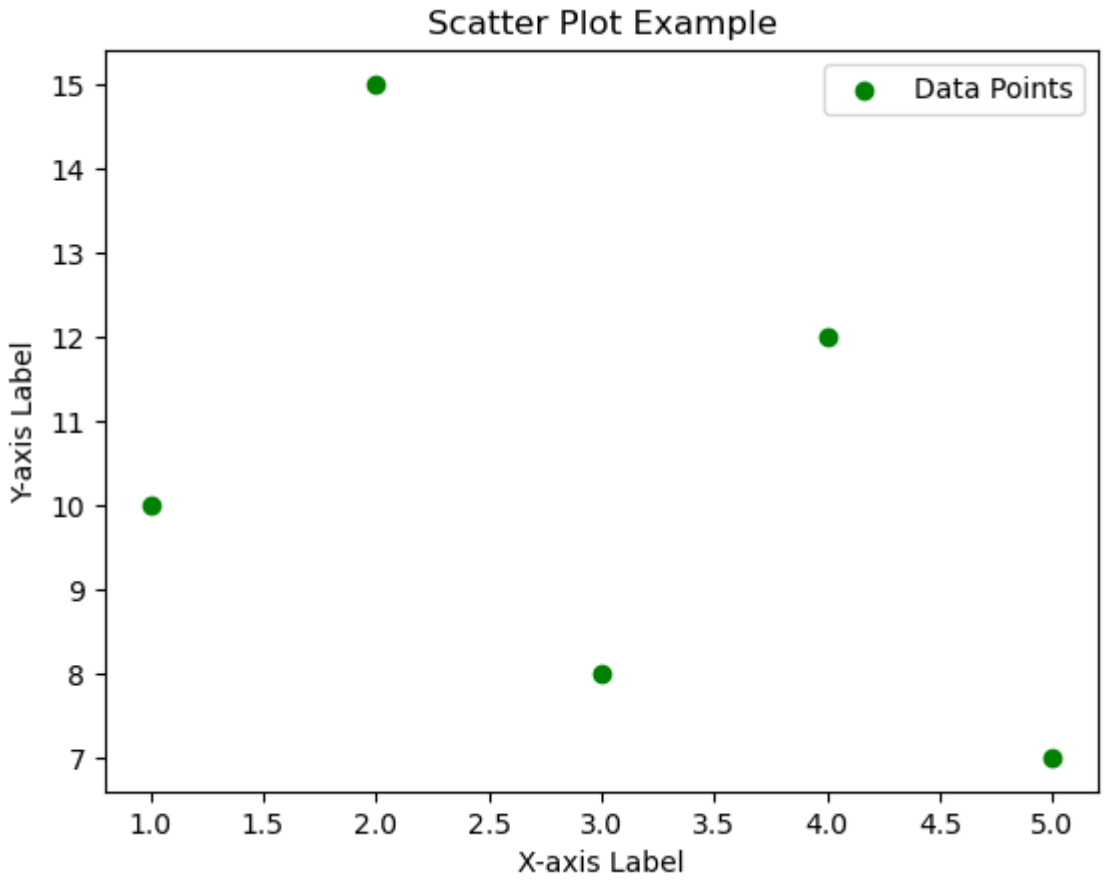


Scatter Plot:

- Scatter plots are used to display individual data points as dots.
- Great for showing relationships between two variables.

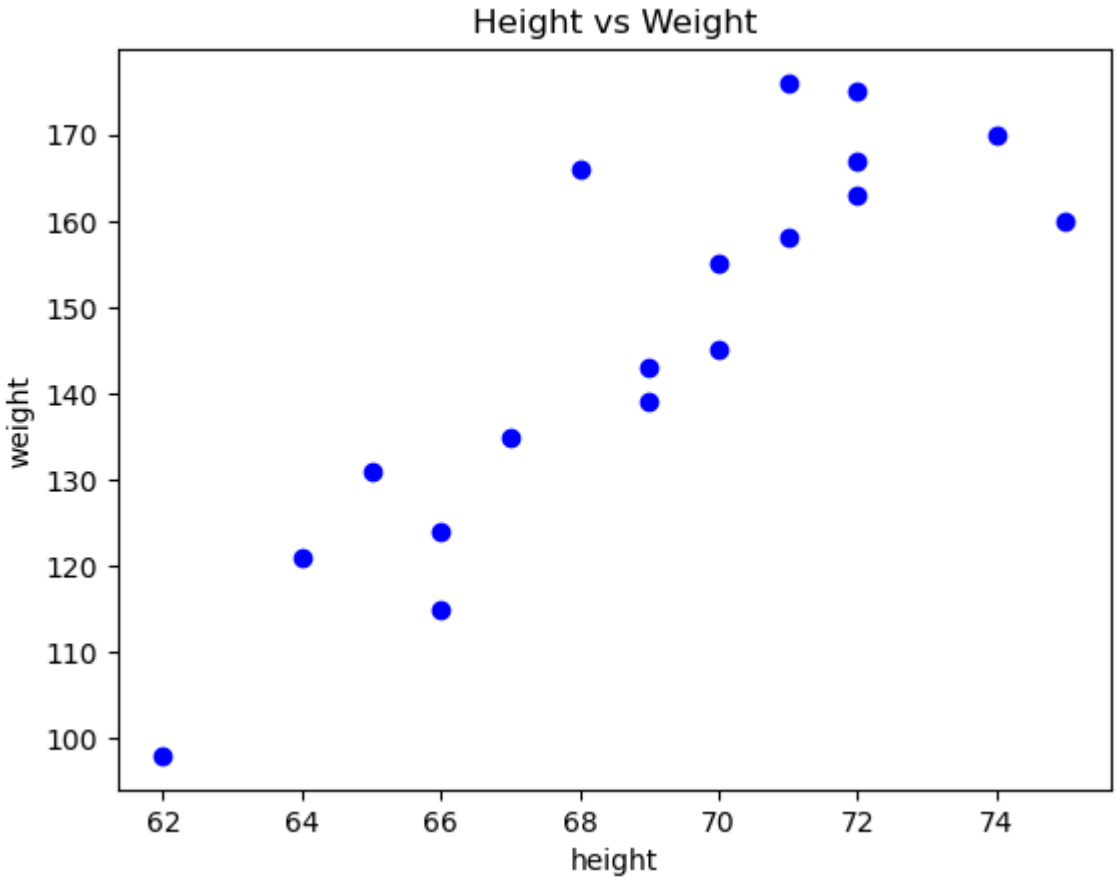
In [12]:

```
# Sample data
x_values = [1, 2, 3, 4, 5]
y_values = [10, 15, 8, 12, 7]

plt.scatter(x_values, y_values, color='g', marker='o', label='Data Points')

plt.xlabel('X-axis Label')
plt.ylabel('Y-axis Label')
plt.title('Scatter Plot Example')

plt.legend()

plt.show();


```



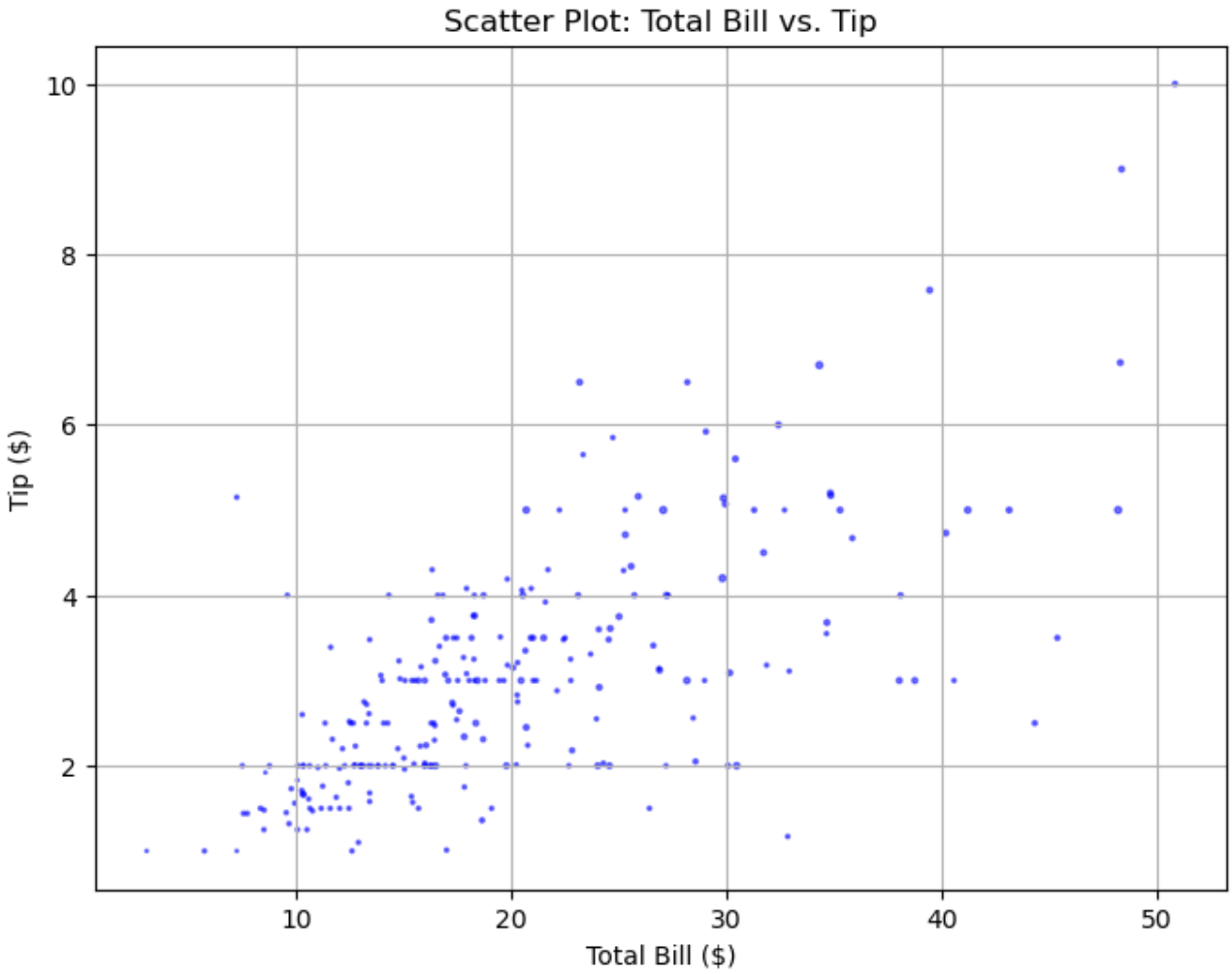
In [13]:

```
plt.scatter(df4.height, df4.weight, c='b')
plt.xlabel('height')
plt.ylabel('weight')
plt.title('Height vs Weight')
plt.show()

```



In [56]:

```
plt.figure(figsize=(8, 6))
plt.scatter(tips_data["total_bill"], tips_data["tip"], color='blue', alpha=0.5,s
plt.title('Scatter Plot: Total Bill vs. Tip')
plt.xlabel('Total Bill ($)')
plt.ylabel('Tip ($)')
plt.grid(True)
plt.show()
```

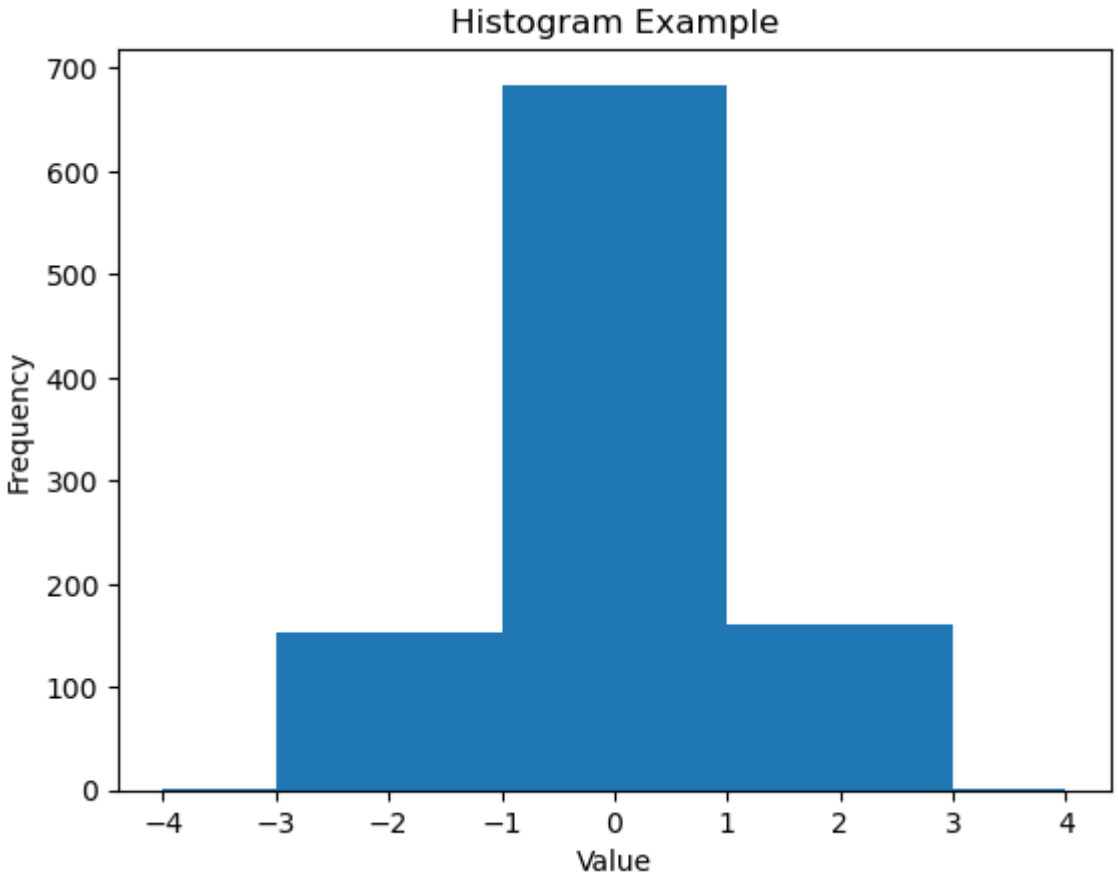


Histogram:

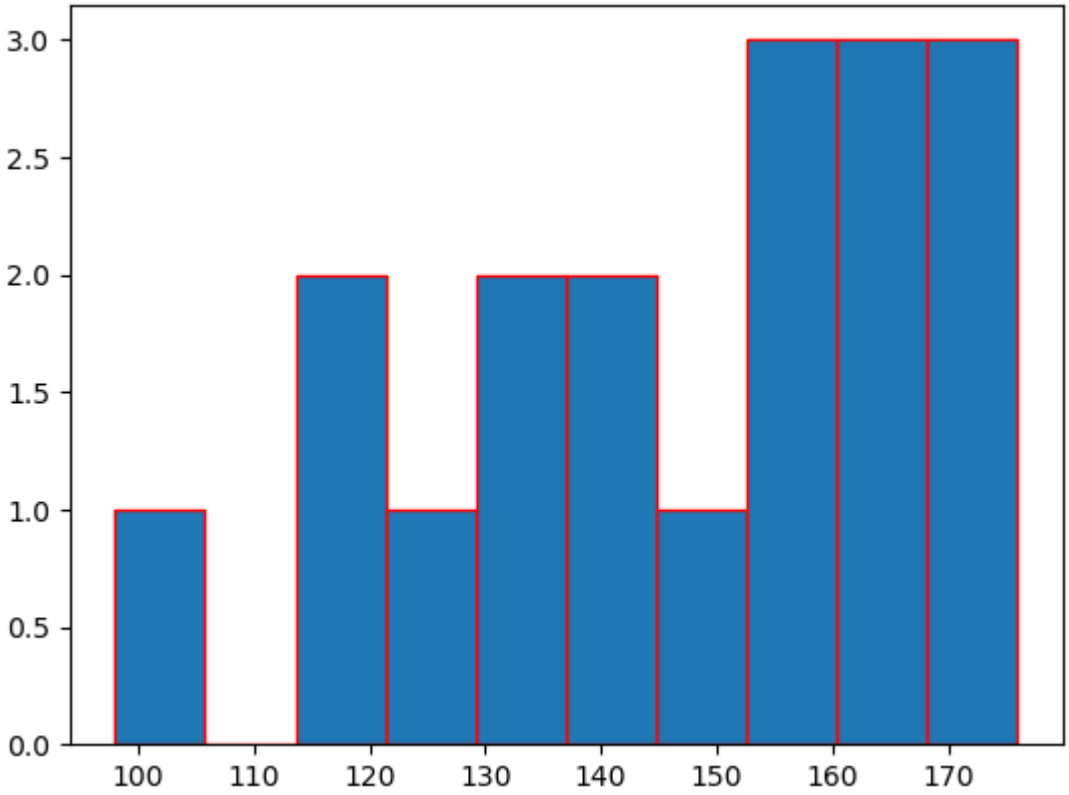
- A histogram is a graphical representation of the distribution of a dataset.
- It displays the frequency or count of data points within specified intervals, often called "bins," along the x-axis.
- The y-axis represents the frequency or count of data points falling within each bin.
- Histograms are particularly useful for visualizing the underlying distribution of continuous or discrete data.
- for Univariate Analysis.

In [14]:

```
# Generate random data
data = np.random.randn(1000)  # Sample data from a normal distribution

# Create a histogram
plt.hist(data,bins=[-4,-3,-1,1,3,4])

plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Histogram Example')

plt.show()
```



In [15]:
```
1 plt.hist(df4.weight,edgecolor = 'red');
```

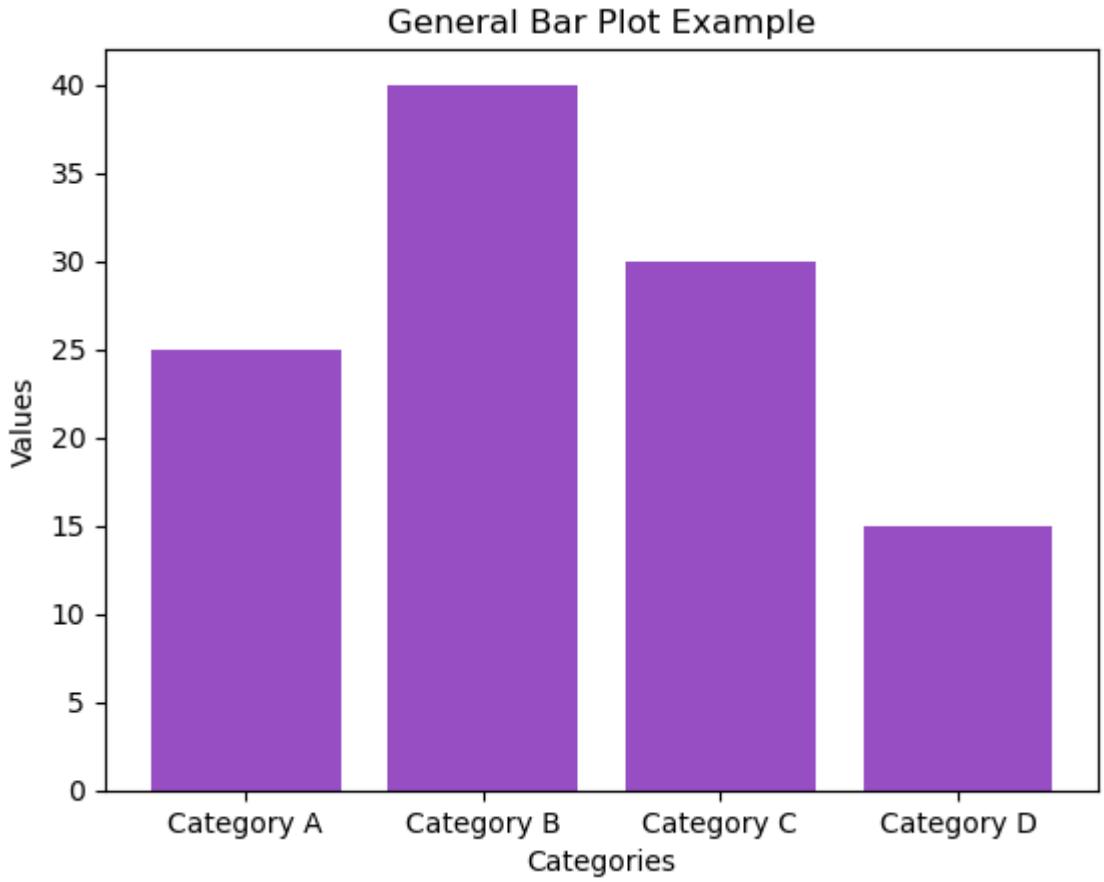


Bar Plot:

- A bar plot, also known as a bar chart or bar graph, is a data visualization tool used to represent categorical data.
- It displays the values of different categories as rectangular bars with lengths proportional to the values they represent.
- Bar plots are commonly used to compare and display the distribution, frequency, or relative size of different categories or groups.

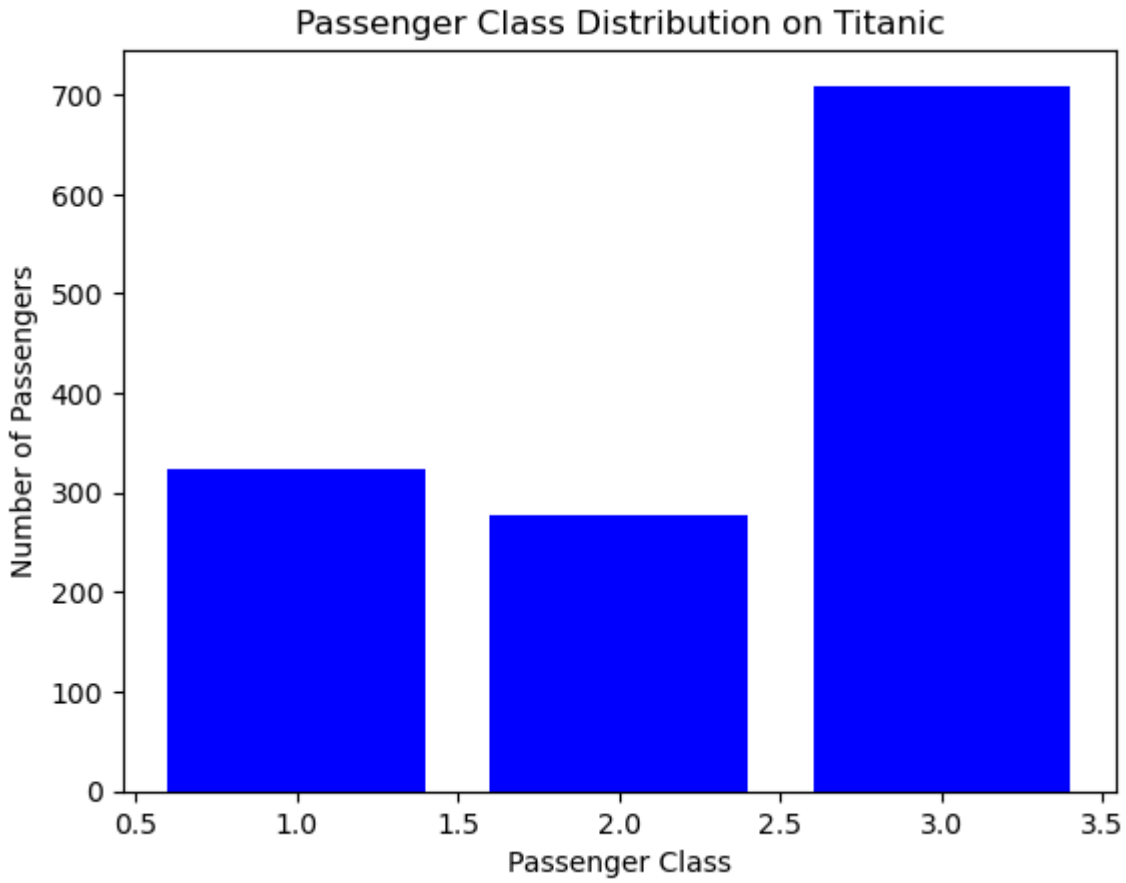
In [16]:

```
# Sample data
categories = ['Category A', 'Category B', 'Category C', 'Category D']
values = [25, 40, 30, 15]

plt.bar(categories, values, color='#974EC3')

plt.xlabel('Categories')
plt.ylabel('Values')
plt.title('General Bar Plot Example')
# plt.grid(axis='y')

plt.show()

```



In [17]:

```
class_counts = df1['Pclass'].value_counts().sort_index()

# Create a bar plot
plt.bar(class_counts.index, class_counts.values, color='blue')

plt.xlabel('Passenger Class')
plt.ylabel('Number of Passengers')
plt.title('Passenger Class Distribution on Titanic')

plt.show();
```



In [18]:

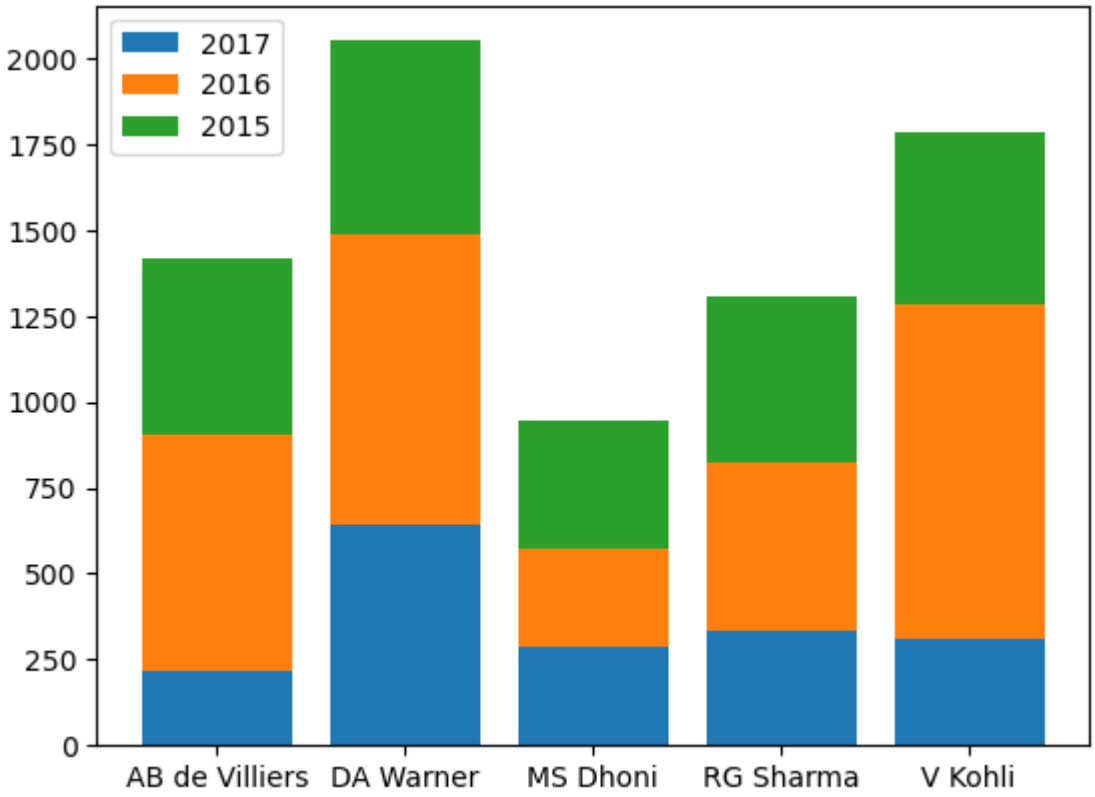
```
# stacked bar chart
batsman = pd.read_csv('Data/batsman_season_record.csv')
batsman
```

Out[18]:

	batsman	2015	2016	2017
0	AB de Villiers	513	687	216
1	DA Warner	562	848	641
2	MS Dhoni	372	284	290
3	RG Sharma	482	489	333
4	V Kohli	505	973	308

In [19]:

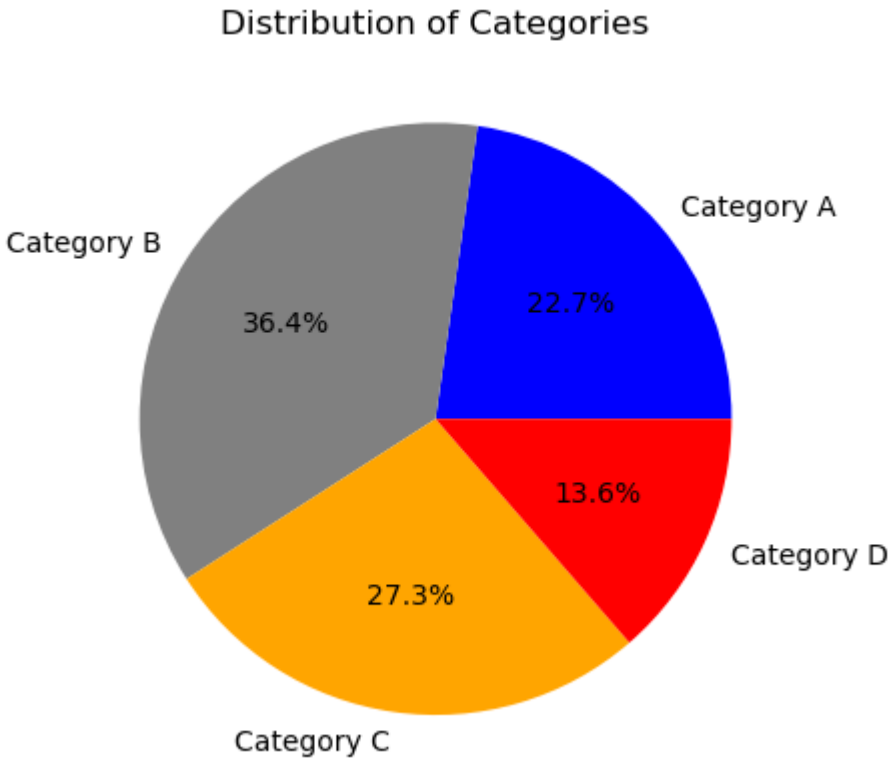
```
plt.bar(batsman['batsman'],batsman['2017'],label = '2017')
plt.bar(batsman['batsman'],batsman['2016'],bottom=batsman['2017'],label='2016')
plt.bar(batsman['batsman'],batsman['2015'],bottom= batsman['2016']+batsman['2017
plt.legend();
```



PIE CHART:-

- A circular graphical representation used to visualize the distribution of data within different categories or parts of a whole.
- It is useful when you want to show how a single entity is divided into smaller components, and you want to emphasize the proportions or percentages of those components.

In [20]:
```
# Sample data
labels = ['Category A', 'Category B', 'Category C', 'Category D']
sizes = [25, 40, 30, 15]
colors = ['blue', 'grey', 'orange', 'red']

plt.pie(sizes, labels=labels, colors=colors,autopct='%1.1f%%') #explode/shadow
plt.title('Distribution of Categories')

plt.show()

```

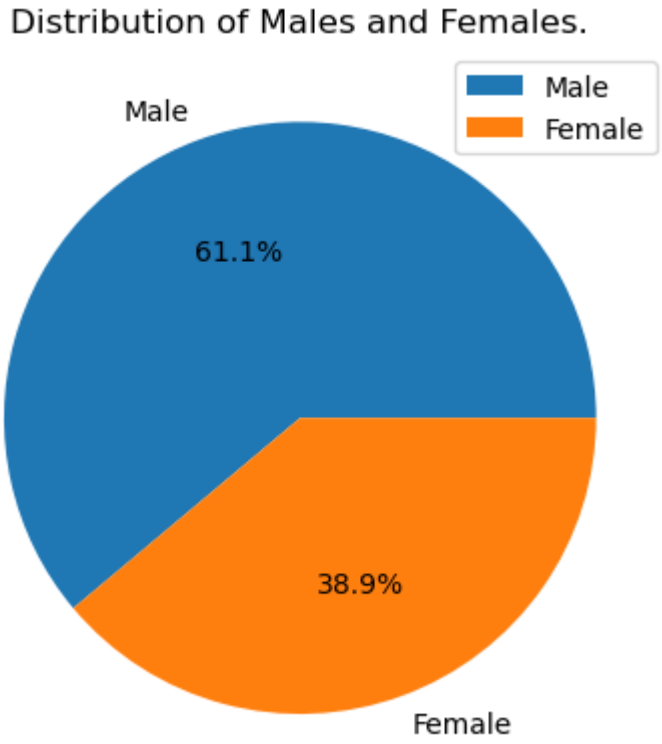


In [21]:
```
df4['sex'].value_counts()
```

Out[21]:
```
M    11
F     7
Name: sex, dtype: int64
```

In [22]:

```
plt.pie(df4['sex'].value_counts(),labels=['Male','Female'],autopct="%0.1f%%")
plt.legend()
plt.title("Distribution of Males and Females.")
plt.show()
```

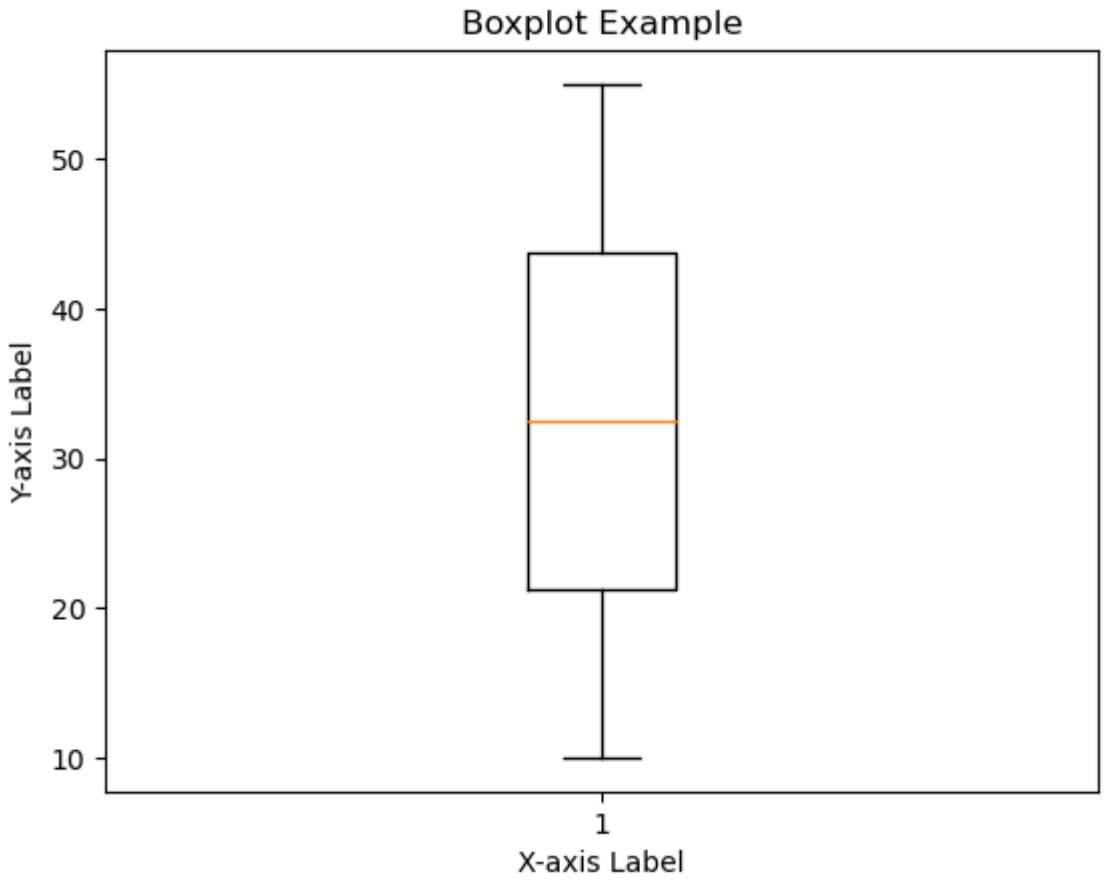


BOXPLOTS:

- A box plot, also known as a box-and-whisker plot, is a graphical representation of the distribution of a dataset through its five-number summary: minimum, first quartile (25th percentile), median (50th percentile), third quartile (75th percentile), and maximum.
- It provides a compact way to visualize the central tendency, spread, and potential outliers of a dataset.

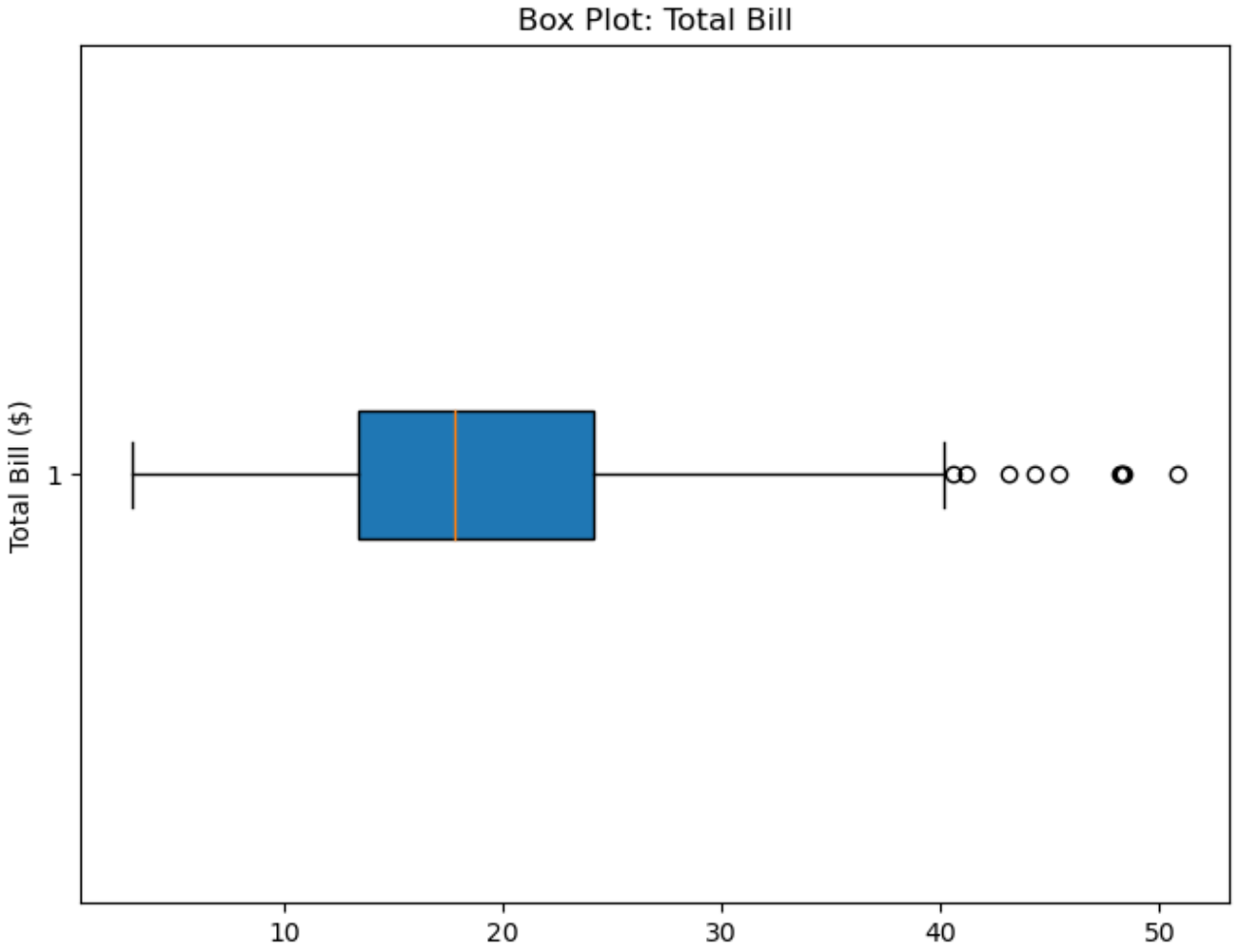
In [23]:

```python
import matplotlib.pyplot as plt

# Sample data
data = [10, 15, 20, 25, 30, 35, 40, 45, 50, 55]

# Create a boxplot
plt.boxplot(data)

# Add title and labels
plt.title('Boxplot Example')
plt.xlabel('X-axis Label')
plt.ylabel('Y-axis Label')

# Display the plot
plt.show()

```



In [48]:

```
plt.figure(figsize=(8, 6))
plt.boxplot(tips_data['total_bill'],vert=False, patch_artist=True)
plt.title('Box Plot: Total Bill')
plt.ylabel('Total Bill ($)')
# plt.xticks([1], ['Total Bill'], rotation=45)
plt.show()
```



Adding Details to plot:

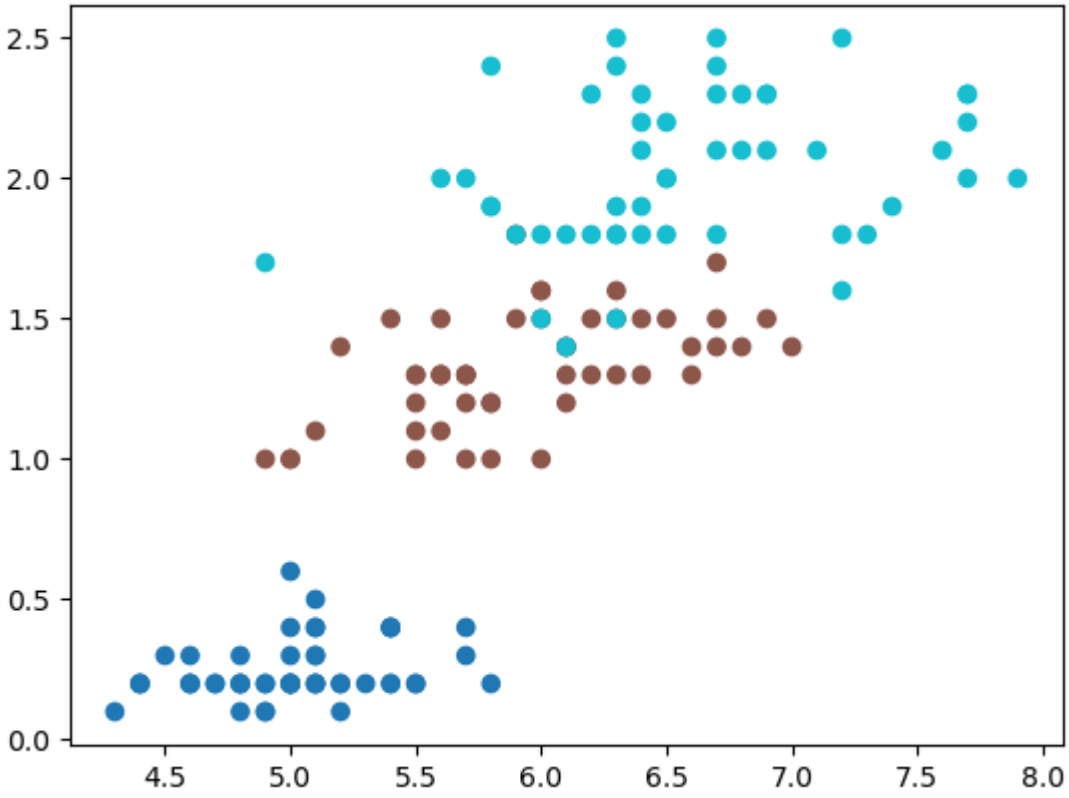
- `plt.title('Chart Title')`: Adds a title to your chart.
- `plt.xlabel('X Label')` and `plt.ylabel('Y Label')`: Adds labels to the x-axis and y-axis, respectively.
- `plt.legend()`: Adds a legend to differentiate between multiple data series.
- `plt.grid()`: Adds grid lines to your chart for better readability.
- `plt.xticks(ticks, labels)`: Sets custom tick locations and labels on the x-axis.
- `plt.yticks(ticks, labels)`: Sets custom tick locations and labels on the y-axis.
- You can customize colors, line styles, and marker styles using parameters like `color`, `linestyle`, and `marker` in plotting functions.
- You can adjust the transparency of chart elements using the `alpha` parameter.
- `plt.xlim(min_value, max_value)` and `plt.ylim(min_value, max_value)`: Set custom axis limits.
- `plt.text(x, y, 'Text', fontsize=12, color='red')`: Adds text to specified coordinates with custom font size and color.

In [24]:
```
# plt.style.available
# plt.style.use("Solarize_Light2")
# saving a plot [plt.savefig()]
```

In [25]:
```
# cmap
# colorbar
```

In [26]:
```
iris_data['species']= iris_data['species'].replace({'setosa':0,'versicolor':1,'v
```

In [27]:
```
plt.scatter(iris_data['sepal_length'],iris_data['petal_width'],
            c=iris_data['species'],
            cmap= 'tab10')
# plt.colorbar()
plt.show();
```



Plot size:-

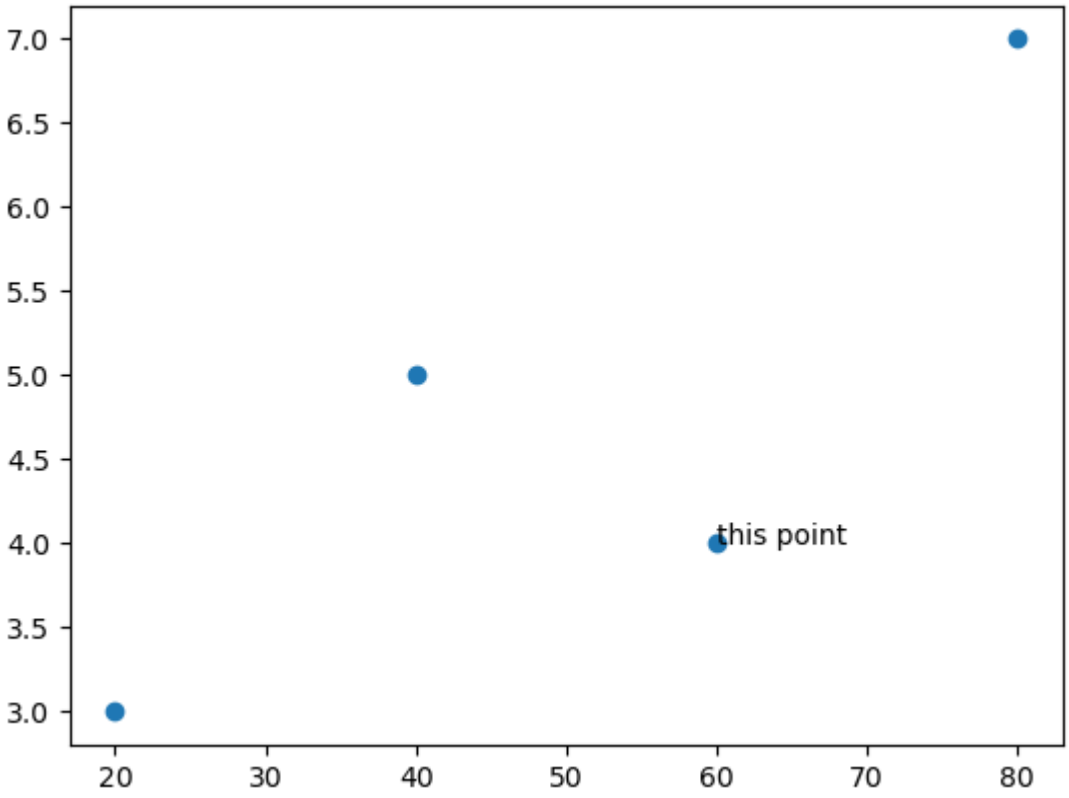
In [28]:
```
# plt.figure(figsize=(12,6))
```

Annotations

In [29]:
```
#plt.text
```

In [30]:
```
x = [20,40,60,80]
y = [3,5,4,7]
plt.scatter(x,y)
plt.text(60,4,'this point')
```

Out[30]:
```
Text(60, 4, 'this point')
```

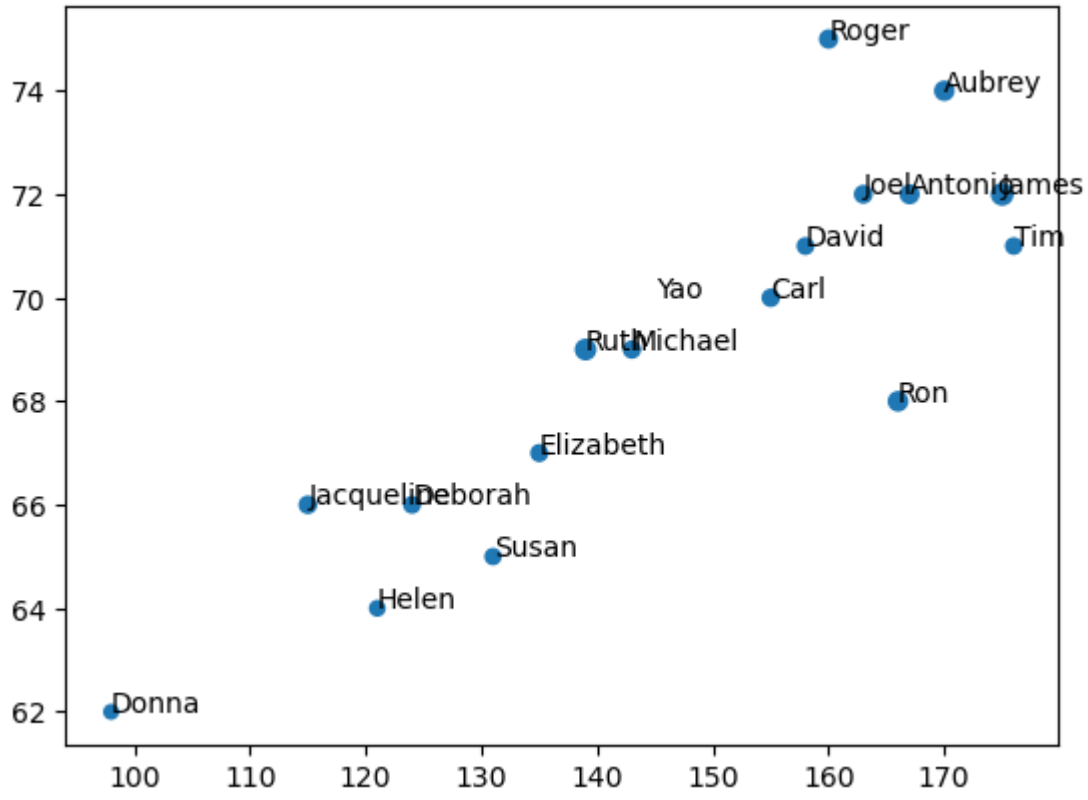


In [31]:
```
df4.head()
```

Out[31]:

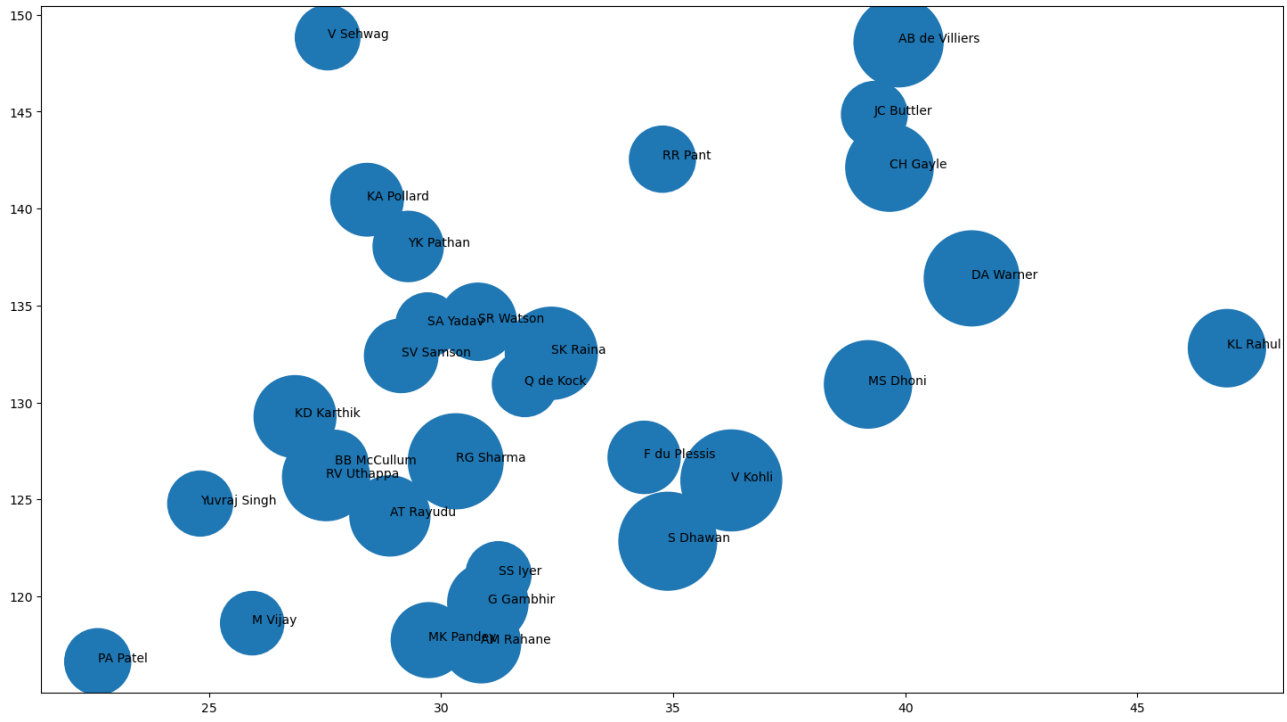
	name	sex	age	height	weight
0	Aubrey	M	41.0	74	170
1	Ron	M	42.0	68	166
2	Carl	M	32.0	70	155
3	Antonio	M	39.0	72	167
4	Deborah	F	30.0	66	124

In [32]:
```
plt.scatter(df4['weight'],df4['height'],s= df4['age'])
for i in range(df4.shape[0]):
    plt.text(df4['weight'].values[i],df4['height'].values[i],df4['name'].values[
```



In [33]:
```
batters = pd.read_csv('Data/batter.csv')
batters_df = batters.head(30)
```

In [34]:
```
plt.figure(figsize=(18,10))
plt.scatter(batters_df['avg'],batters_df['strike_rate'],s=batters_df['runs'])

for i in range(batters_df.shape[0]):
    plt.text(batters_df['avg'].values[i],batters['strike_rate'].values[i],batter
```

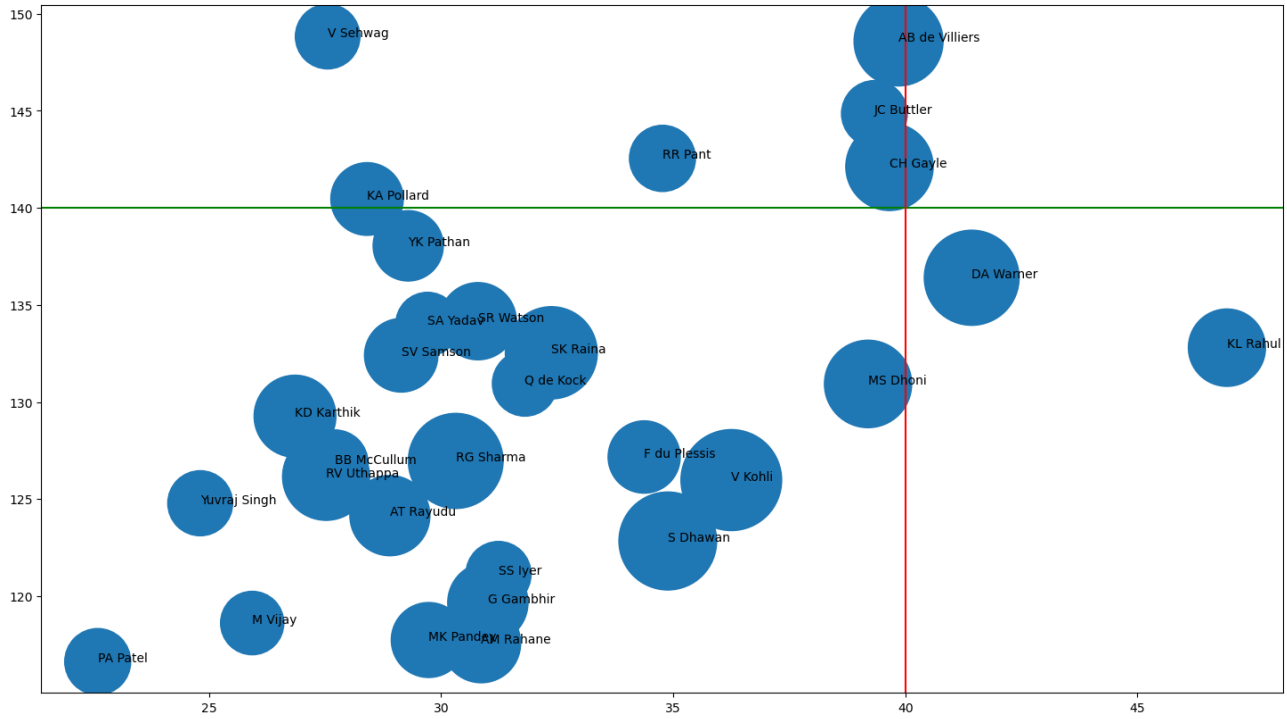


ADDING LINES

- plt.axhline()
- plt.axvlines()

In [35]:

```
1  plt.figure(figsize=(18,10))
2  plt.scatter(batters_df['avg'],batters_df['strike_rate'],s=batters_df['runs'])
3
4  plt.axhline(140,color='green')
5  plt.axvline(40,color = 'red')
6  for i in range(batters_df.shape[0]):
7      plt.text(batters_df['avg'].values[i],batters['strike_rate'].values[i],batter
8
```



SUBPLOTS :

In [36]:
```
1 tips_data
```

Out[36]:

	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
...	...	...	...	...	...	...	...
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

244 rows × 7 columns

In [37]:

```
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(10, 6))

# Generate some sample data
x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)
y3 = x ** 2
y4 = np.exp(x)

# Plot data on each subplot
axes[0, 0].scatter(tips_data['total_bill'],tips_data['tip'])
axes[0, 0].set_title('Total bill vs tips')

axes[0, 1].hist(tips_data['total_bill'])
axes[0, 1].set_title('Distribution of Total Bill column')

axes[1, 0].bar(tips_data['day'],tips_data['total_bill'])
axes[1, 0].set_title('Day wise Total bill')

axes[1, 1].pie(tips_data['sex'].value_counts(),labels=['Male','Female'],autopct=
axes[1, 1].set_title('Sex distribution.')
axes[1, 1].legend( loc="upper right")

# Adjust layout
plt.tight_layout()

# Show the subplots
plt.show()

```