Practical File Internet of Things (IoT)

Bachelor of Computer Applications (BCA)

To Guru Gobind Singh Indraprastha University

Submitted to:

Dr. Anu Taneja

(Professor)

Submitted By:

Aditya Kumar

Roll No. 04611102021

Banarsidas Chandiwala Institute of Information Technology

New Delhi – 110019

Batch (2021-2024)

BCA (372)

1. WAP to implement data preprocessing on student's dataset.

• Handle missing data using different methods

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

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

	roll no	name	marks
0	1.0	a	63.0
1	2.0	NaN	25.0
2	NaN	NaN	NaN
3	4.0	d	14.0
4	5.0	e	NaN
5	6.0	NaN	85.0
6	7.0	g	NaN
7	8.0	NaN	NaN
8	9.0	NaN	95.0
9	10.0	NaN	100.0

a) Delete the row containing null values

```
[3] df.dropna()
```

	roll no	name	marks
0	1.0	a	63.0
3	4.0	d	14.0

b) Delete the row containing all null values

```
df.dropna(how='all')
```

	roll no	name	marks
0	1.0	a	63.0
1	2.0	NaN	25.0
3	4.0	d	14.0
4	5.0	e	NaN
5	6.0	NaN	85.0
6	7.0	g	NaN
7	8.0	NaN	NaN
8	9.0	NaN	95.0
9	10.0	NaN	100.0

c) Replace the null values using forward method

```
[5] df.fillna(method='ffill')
```

	roll no	name	marks
0	1.0	a	63.0
1	2.0	a	25.0
2	2.0	a	25.0
3	4.0	d	14.0
4	5.0	e	14.0
5	6.0	e	85.0
6	7.0	g	85.0
7	8.0	g	85.0
8	9.0	g	95.0
9	10.0	g	100.0

d) Replace the null values using backward method

```
df.fillna(method='bfill')
```

	roll no	name	marks
0	1.0	a	63.0
1	2.0	d	25.0
2	4.0	d	14.0
3	4.0	d	14.0
4	5.0	e	85.0
5	6.0	g	85.0
6	7.0	g	95.0
7	8.0	NaN	95.0
8	9.0	NaN	95.0
9	10.0	NaN	100.0

e) Replace with constant value

```
df.fillna(0)
```

	roll no	name	marks
0	1.0	a	63.0
1	2.0	0	25.0
2	0.0	0	0.0
3	4.0	d	14.0
4	5.0	e	0.0
5	6.0	0	85.0
6	7.0	g	0.0
7	8.0	0	0.0
8	9.0	0	95.0
9	10.0	0	100.0

f) Replace with central tendency measures –mean, mode, medium

```
df.fillna(df.median(numeric_only=True))
```

	roll no	name	marks
0	1.0	a	63.0
1	2.0	NaN	25.0
2	6.0	NaN	74.0
3	4.0	d	14.0
4	5.0	e	74.0
5	6.0	NaN	85.0
6	7.0	g	74.0
7	8.0	NaN	74.0
8	9.0	NaN	95.0
9	10.0	NaN	100.0

```
df.fillna(df.mode(numeric_only=True))
```

	roll no	name	marks
0	1.0	a	63.0
1	2.0	NaN	25.0
2	4.0	NaN	63.0
3	4.0	d	14.0
4	5.0	e	95.0
5	6.0	NaN	85.0
6	7.0	g	NaN
7	8.0	NaN	NaN
8	9.0	NaN	95.0
9	10.0	NaN	100.0

```
[8] df.fillna(df.mean(numeric_only=True))
```

	roll no	name	marks
0	1.000000	a	63.000000
1	2.000000	NaN	25.000000
2	5.777778	NaN	63.666667
3	4.000000	d	14.000000
4	5.000000	e	63.666667
5	6.000000	NaN	85.000000
6	7.000000	g	63.666667
7	8.000000	NaN	63.666667
8	9.000000	NaN	95.000000
9	10.000000	NaN	100.000000

2. Implement data preprocessing on the given e commerce dataset:

```
dataset = pd.read_csv('ecommerce.csv')
dataset
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

A) describe the data

a) Head

```
dataset.head()
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes

b) Tail

```
dataset.tail()
```

	Country	Age	Salary	Purchased
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

c) Shape

```
1  dataset.shape
```

```
(10, 4)
```

d) Info

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Country    10 non-null     object 
 1   Age        9 non-null      float64
 2   Salary     9 non-null      float64
 3   Purchased  10 non-null     object 
dtypes: float64(2), object(2)
memory usage: 448.0+ bytes
```

B) Check whether null values are present or not and display column wise

a) Also

```
dataset.isnull().sum()
```

```
Country      0
Age          1
Salary       1
Purchased    0
dtype: int64
```

C) extract the independent and dependent variables

```
x = dataset.iloc[:,:-1]
y = dataset.iloc[:, -1:]
```

```
print("ind variable: \n", x)
print("dep variable: \n",y)
```

```
ind variable:
    Country   Age   Salary
0   France  44.0  72000.0
1    Spain  27.0  48000.0
2  Germany  30.0  54000.0
3    Spain  38.0  61000.0
4  Germany  40.0  58000.0
5   France  35.0  58000.0
6    Spain  48.0  52000.0
7   France  48.0  79000.0
8  Germany  50.0  83000.0
9   France  37.0  67000.0
dep variable:
   Purchased
0         No
1        Yes
2         No
3         No
4        Yes
5        Yes
6         No
7        Yes
8         No
9        Yes
```

D) Handle the missing data.

```
[ ]  dataset.fillna(method='ffill')
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	61000.0	Yes
5	France	35.0	58000.0	Yes
6	Spain	35.0	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

E) split into training and test data

```
[ ] x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 0)
```

```
print("Ind train: \n", x_train)
print("Ind test: \n",x_test)
print("Dep train: \n",y_train)
print("Dep test: \n",y_test)
```

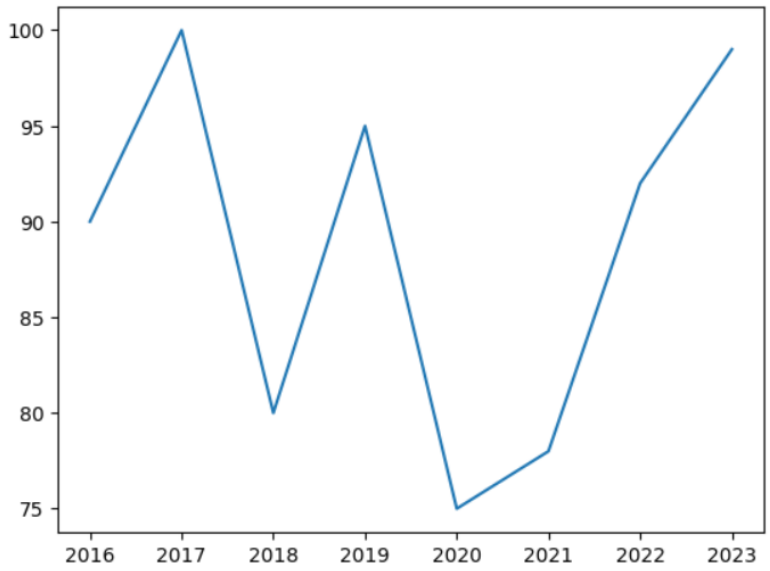
```
Ind train:
    Country   Age   Salary
4  Germany  40.0      NaN
9   France  37.0  67000.0
1    Spain  27.0  48000.0
6    Spain   NaN  52000.0
7   France  48.0  79000.0
3    Spain  38.0  61000.0
0   France  44.0  72000.0
5   France  35.0  58000.0
Ind test:
    Country   Age   Salary
2  Germany  30.0  54000.0
8  Germany  50.0  83000.0
Dep train:
   Purchased
4        Yes
9        Yes
1        Yes
6         No
7        Yes
3         No
0         No
5        Yes
Dep test:
   Purchased
2         No
8         No
```

3. WAP to implement line plots by comparing performances of computer science subject of different years.

a. Draw a basic line plot using plot()

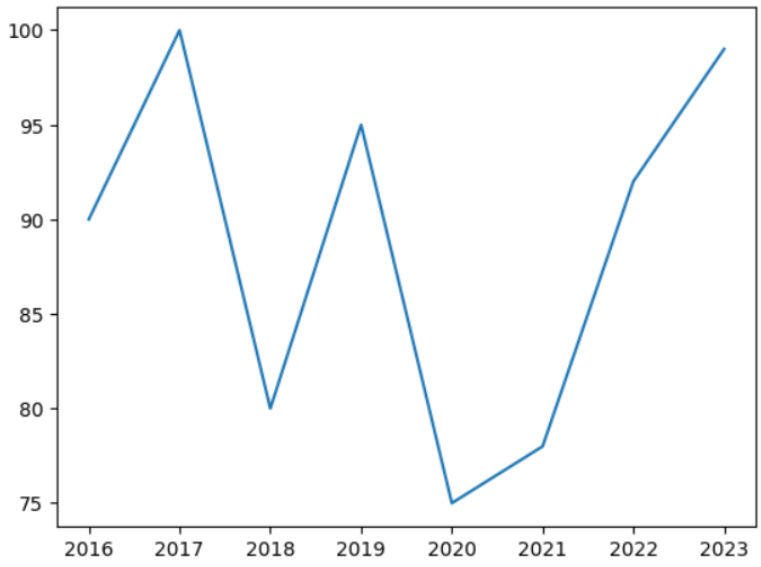
```
year = np.array([2016,2017,2018,2019,2020,2021,2022,2023])
csResult = np.array([90,100,80,95,75,78,92,99])

plt.plot(year, csResult)
```

```
[<matplotlib.lines.Line2D at 0x7e281fa7ad40>]
```



b. Display a line plot using show()

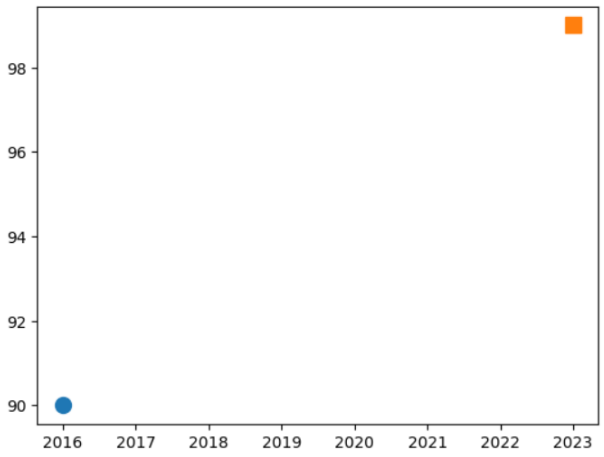
```
[85] plt.plot(year, csResult)
     plt.show()
```



c. Display markers (initial and final endpoints)

```
plt.plot(year[0], csResult[0], marker="o", markersize=10)
plt.plot(year[-1], csResult[-1], marker="s", markersize=10)
```

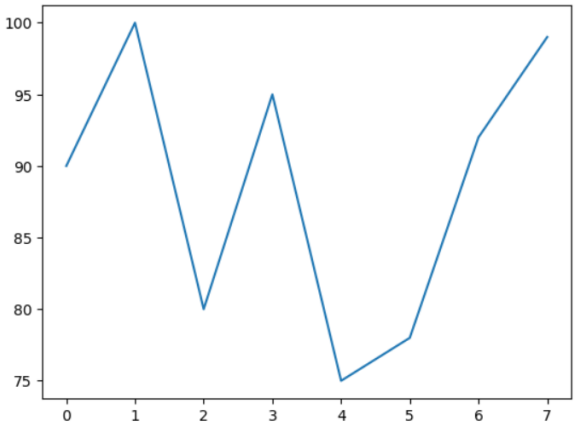
```
[<matplotlib.lines.Line2D at 0x7e281f8c0760>]
```



d. Take only y data

```
plt.plot(csResult)
```

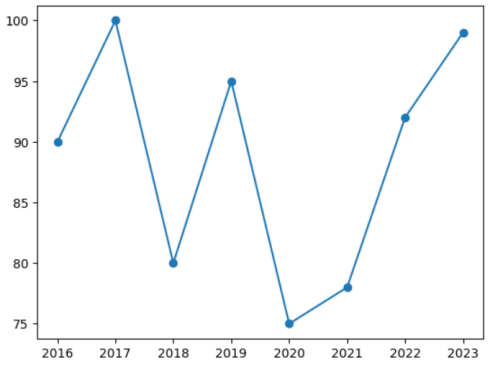
```
[<matplotlib.lines.Line2D at 0x7e281f86d4e0>]
```



e. Display markers on every point

```
plt.plot(year, csResult, marker="o")
```

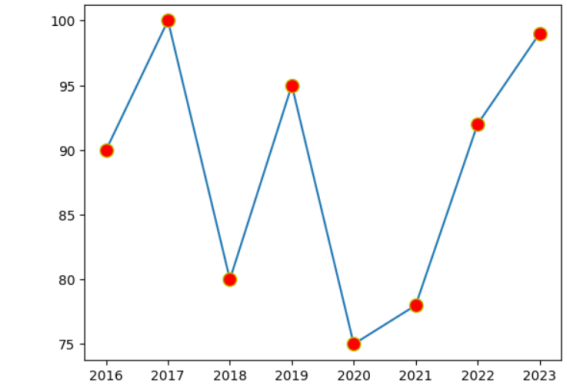
```
[<matplotlib.lines.Line2D at 0x7e281f837730>]
```



f. Set the marker properties : set the marker color,size,edge color,face color

```
plt.plot(year, csResult, marker="o", markersize=10, mfc='red', mec='y')
```

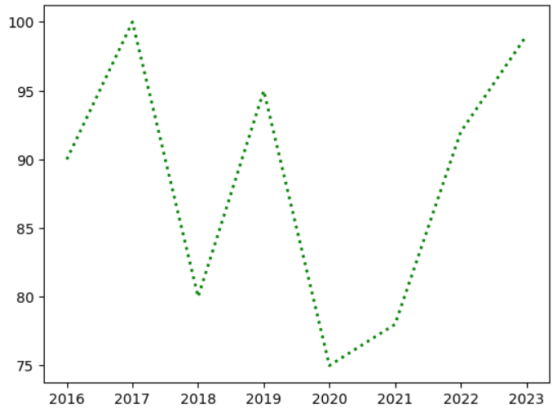
```
[<matplotlib.lines.Line2D at 0x7e281f9aa320>]
```



g. Set the line properties: line style,line color,line width

```
[90] plt.plot(year, csResult, linestyle='dotted', color='g', linewidth=2)
```

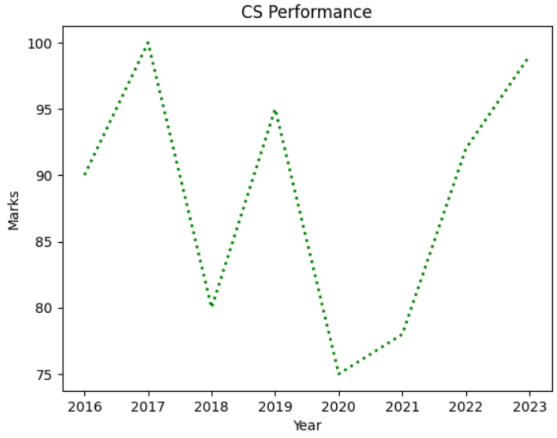
```
[<matplotlib.lines.Line2D at 0x7e281f574370>]
```



h. Set the x axis label, y axis label and title of the graph

```
[91] plt.plot(year, csResult, linestyle='dotted', color='g', linewidth=2)
     plt.title("CS Performance")
     plt.xlabel("Year")
     plt.ylabel("Marks")
```

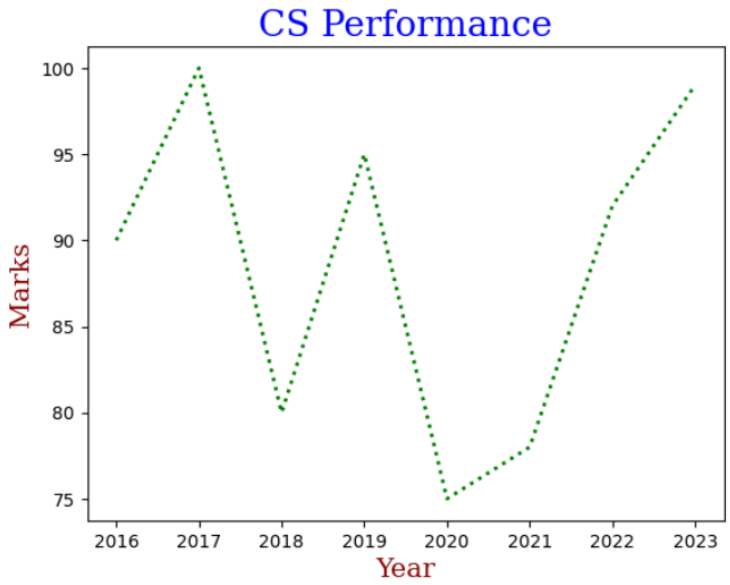
```
Text(0, 0.5, 'Marks')
```



i. Set the font properties for title and label: font family, color and size

```
font1 = {'family':'serif','color':'blue','size':20}
font2 = {'family':'serif','color':'darkred','size':15}

plt.plot(year, csResult, linestyle='dotted', color='g', linewidth=2)
plt.title("CS Performance", fontdict=font1)
plt.xlabel("Year", fontdict=font2)
plt.ylabel("Marks", fontdict=font2)
```

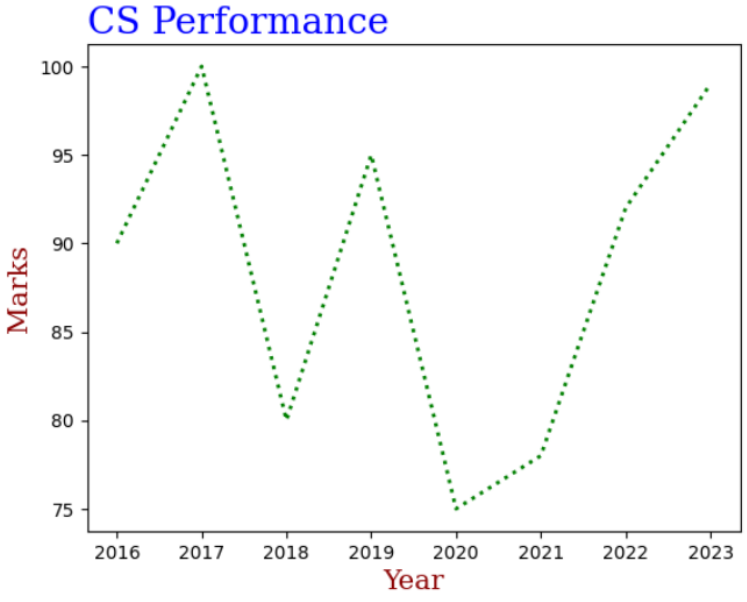
```
Text(0, 0.5, 'Marks')
```



j. Change the position of title

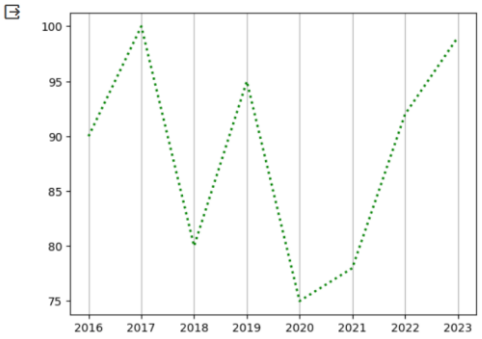
```
plt.plot(year, csResult, linestyle='dotted', color='g', linewidth=2)
plt.title("CS Performance", fontdict=font1, loc='left')
plt.xlabel("Year", fontdict=font2)
plt.ylabel("Marks", fontdict=font2)
```

```
Text(0, 0.5, 'Marks')
```

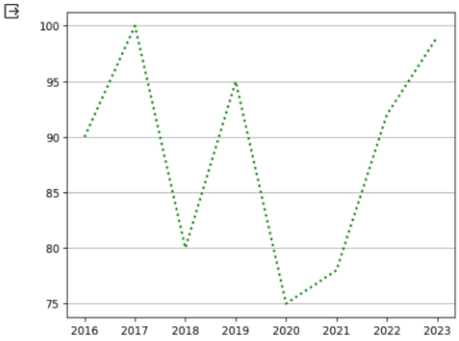


k. Add gridline to the plot(only x axis, only y axis, both)

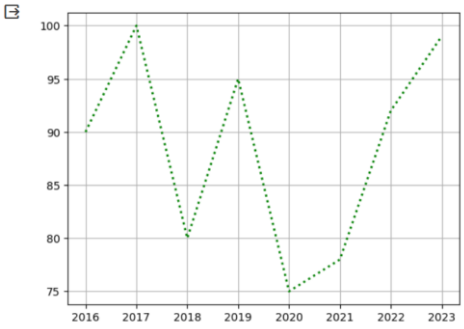
```
plt.plot(year, csResult, linestyle='dotted', color='g', linewidth=2)
plt.grid(axis="x")
```




```
plt.plot(year, csResult, linestyle='dotted', color='g', linewidth=2)
plt.grid(axis="y")
```

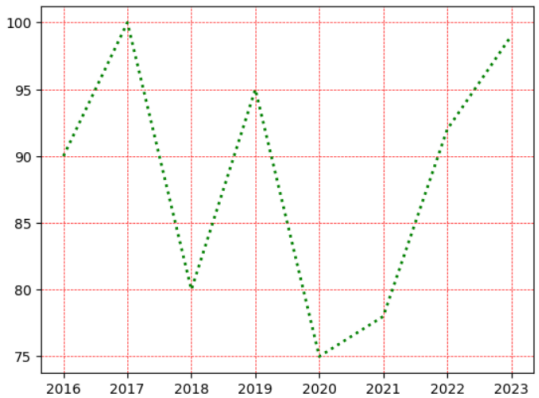



```
plt.plot(year, csResult, linestyle='dotted', color='g', linewidth=2)
plt.grid()
```



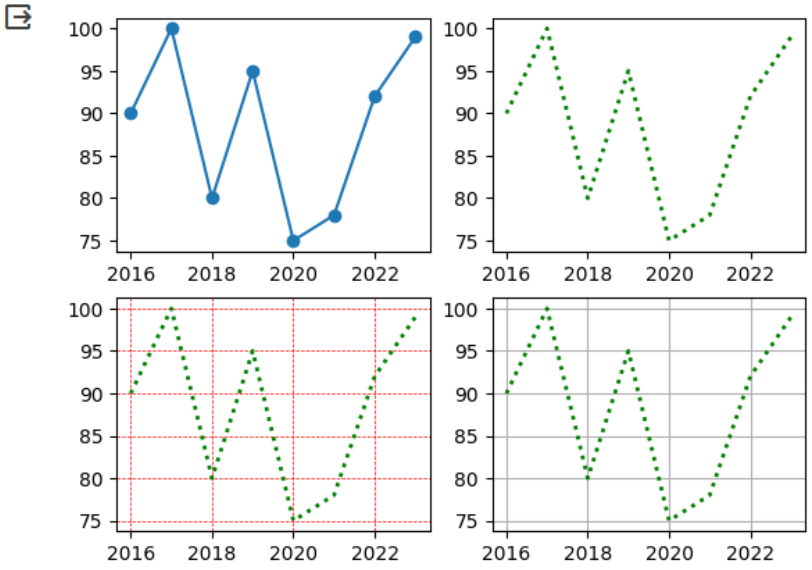
l. Set grid properties : grid color, grid linestyle, grid line width

```
[97] plt.plot(year, csResult, linestyle='dotted', color='g', linewidth=2)
     plt.grid(color="red", linestyle="--", linewidth=0.5)
```



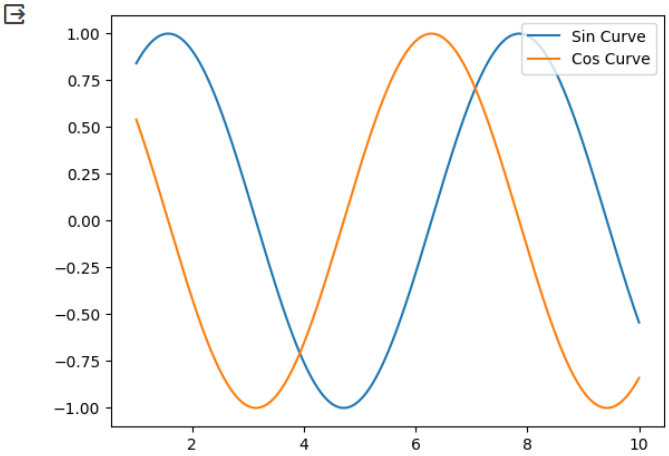
m. Display multiple plots using subplot()

```
plt.subplot(2, 2, 2)
plt.plot(year, csResult, linestyle='dotted', color='g', linewidth=2)

plt.subplot(2, 2, 3)
plt.plot(year, csResult, linestyle='dotted', color='g', linewidth=2)
plt.grid(color="red", linestyle="--", linewidth=0.5)

plt.subplot(2, 2, 4)
plt.plot(year, csResult, linestyle='dotted', color='g', linewidth=2)
plt.grid()

plt.show()
```



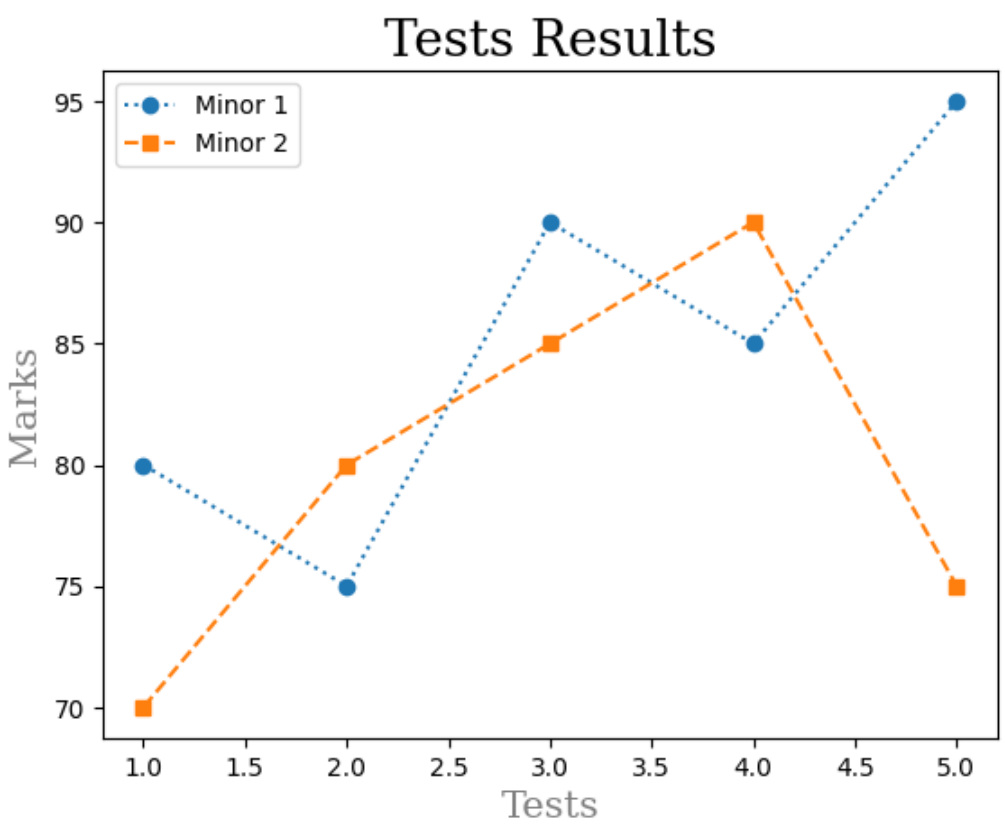
n. Display multiple lineplots in a single plot for comparison

```
x = np.arange(1,10,0.001)

sinY = np.sin(x)
cosY = np.cos(x)

plt.plot(x, sinY, label="Sin Curve")
plt.plot(x, cosY, label="Cos Curve")
plt.legend(loc='upper right')
plt.show()
```



4. WAP to plot a line chart of student's performance in 5 tests of minor1 and minor2 by setting the line style and marker style.

```
minor1 = np.array([80, 75, 90, 85, 95])
minor2 = np.array([70, 80, 85, 90, 75])
tests = np.array([1, 2, 3, 4, 5])

font1 = {"family": "serif", "color": "black", "size": 20}
font2 = {"family": "serif", "color": "grey", "size": 15}

plt.plot(tests, minor1, linestyle='dotted', marker='o', label="Minor 1")
plt.plot(tests, minor2, linestyle='dashed', marker='s', label="Minor 2")
plt.title("Tests Results", fontdict=font1)
plt.xlabel("Tests", fontdict=font2)
plt.ylabel("Marks", fontdict=font2)
plt.legend()
plt.show()
```



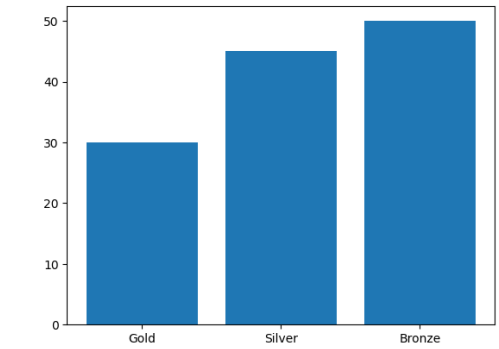
5. WAP to implement a bar graph showing the no. of medals in common

wealth game.

a. Create a basic bar graph using bar()

```
x = np.array(["Gold", "Silver", "Bronze"])
y = np.array([30,45,50])

plt.bar(x, y)
```

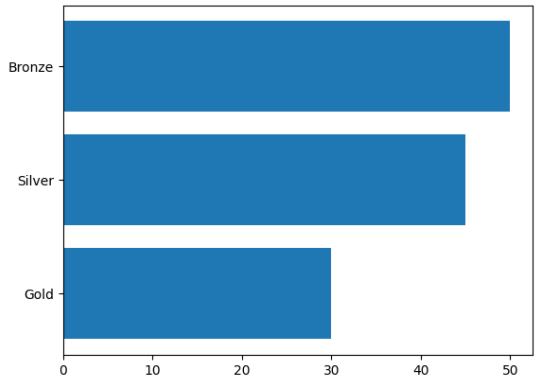
```
<BarContainer object of 3 artists>
```



b. Draw a bar graph horizontally using barh()

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

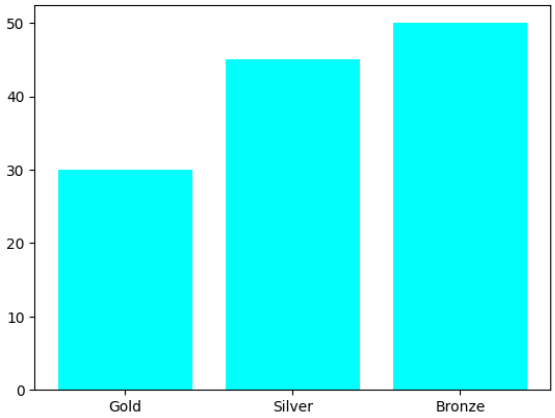
```
<BarContainer object of 3 artists>
```



c. change color of bar

```
plt.bar(x, y, color="cyan")
```

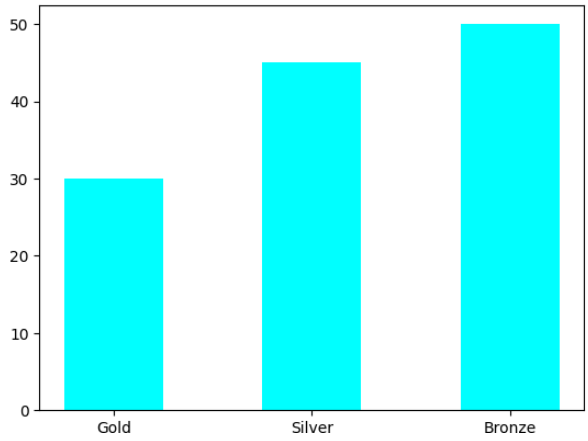
```
<BarContainer object of 3 artists>
```



d. change width of bar

```
plt.bar(x, y, color="cyan", width=0.5)
```

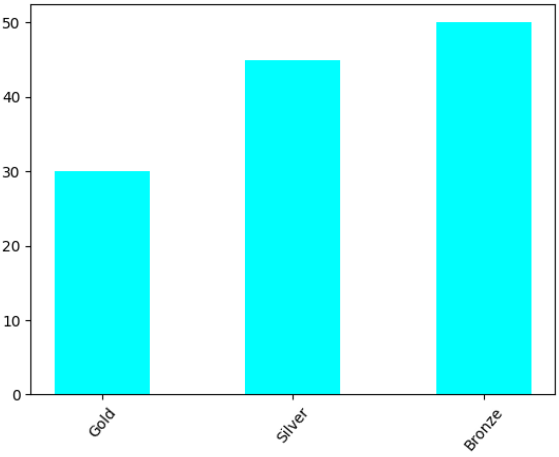
```
<BarContainer object of 3 artists>
```



e. change the alignment of the labels

```
plt.bar(x, y, color="cyan", width=0.5)
plt.xticks(rotation=50)
```

```
([0, 1, 2], [Text(0, 0, 'Gold'), Text(1, 0, 'Silver'), Text(2, 0, 'Bronze')])
```



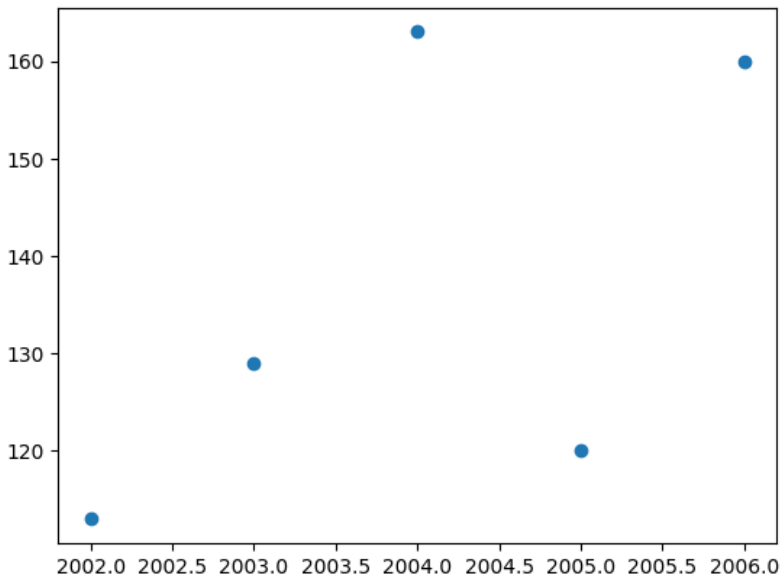
6. Draw a scatter plot that shows the stock trend for 5 years for TATA and

Reliance company. Use the following properties:-

a. Draw the basic scatter graph using scatter()

```
year = np.array([2002,2003,2004,2005,2006])
tataStock = np.random.randint(100,200,size=5)
relianceStock = np.random.randint(100,200,size=5)

plt.scatter(year, tataStock)
```

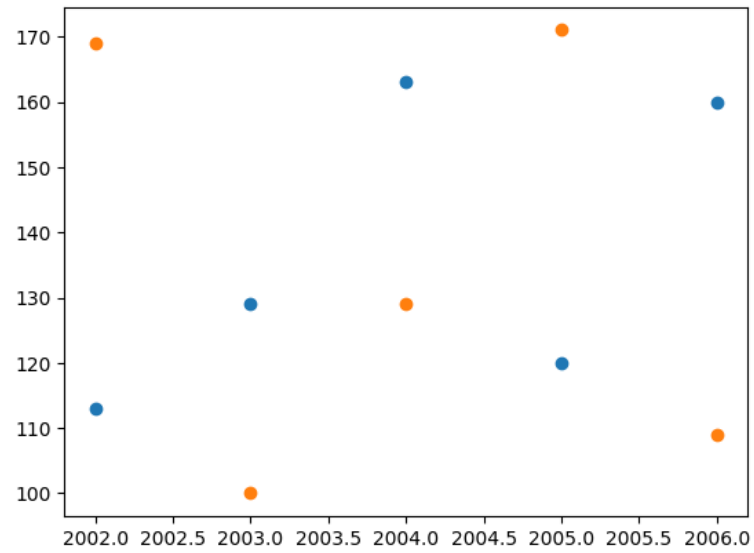
```
<matplotlib.collections.PathCollection at 0x7e281e9e2c20>
```



b. Compare two plots in a single plot

```
plt.scatter(year, tataStock)
plt.scatter(year, relianceStock)
```

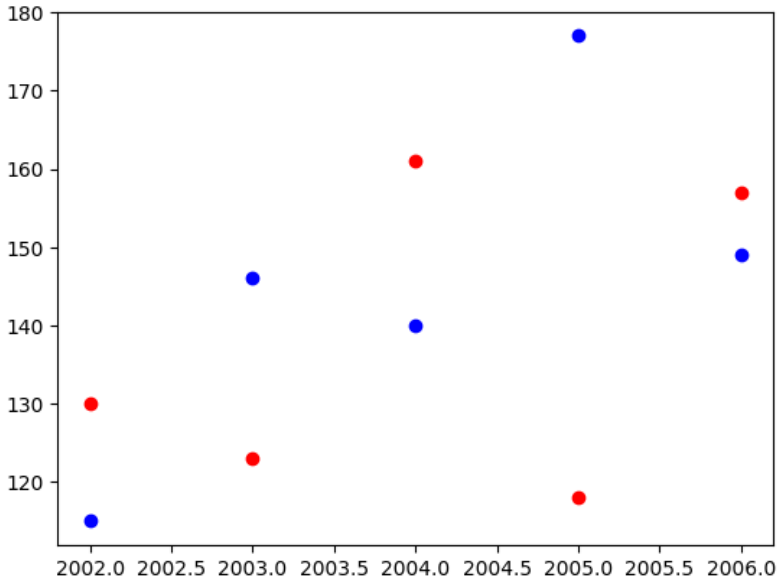
```
<matplotlib.collections.PathCollection at 0x7e281e86a800>
```



c. Set the color of both plots.

```
plt.scatter(year, tataStock, color="red")
plt.scatter(year, relianceStock, color="blue")
```

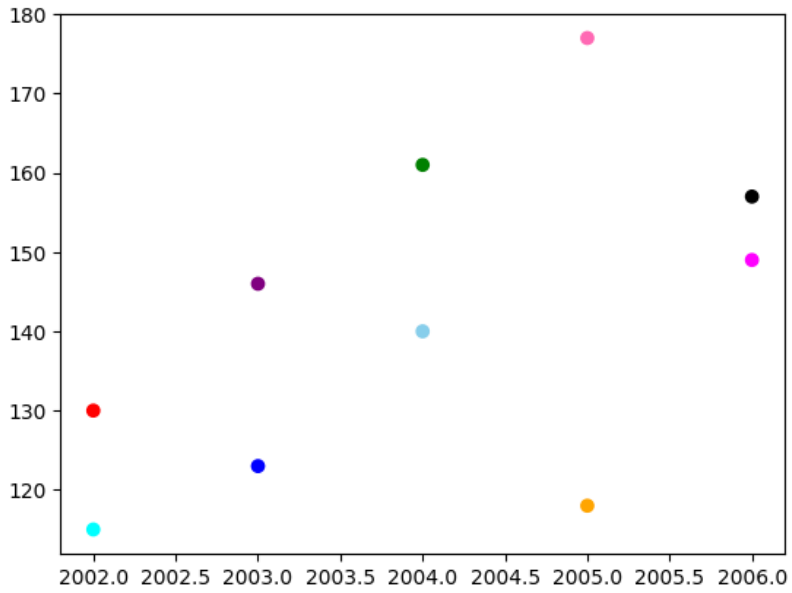
```
<matplotlib.collections.PathCollection at 0x7e281f00d090>
```



d. Set the different color of every dot.

```
color1 = ['red', 'blue', 'green', 'orange', 'black']
plt.scatter(year, tataStock, color=color1)

color2 = ['cyan', 'purple', 'skyblue', 'hotpink', 'magenta']
plt.scatter(year, relianceStock, color=color2)
```

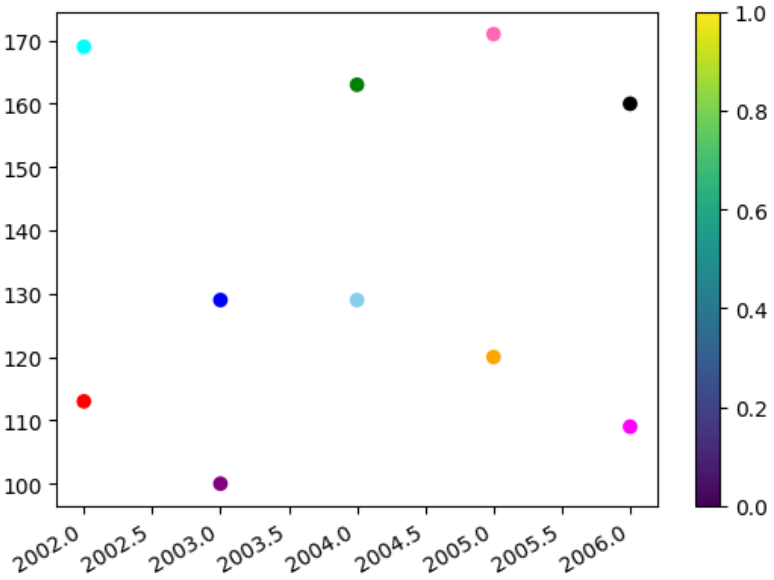
```
<matplotlib.collections.PathCollection at 0x7e281ee8c520>
```



e. Set colormap and display it using colorbar().

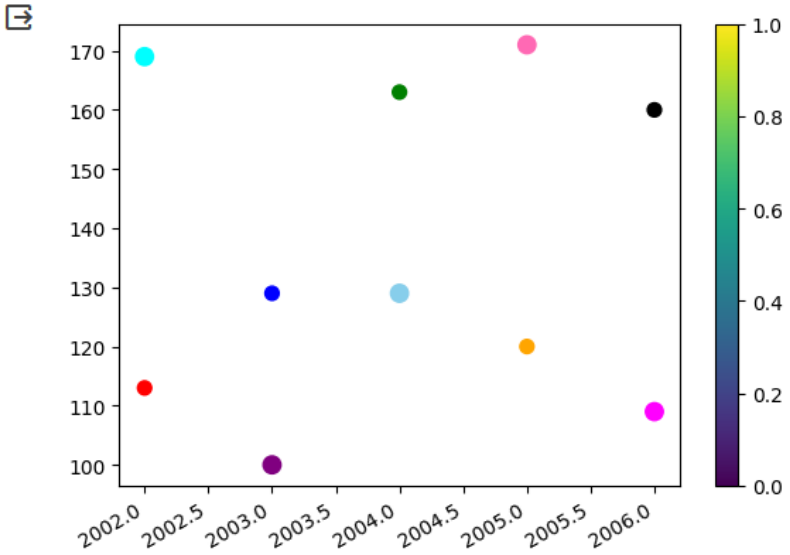
```
color1 = ['red', 'blue', 'green', 'orange', 'black']
plt.scatter(year, tataStock, color=color1)

color2 = ['cyan', 'purple', 'skyblue', 'hotpink', 'magenta']
plt.scatter(year, relianceStock, color=color2)
plt.gcf().autofmt_xdate()

plt.colorbar()
```

```
<matplotlib.colorbar.Colorbar at 0x7e281e5e3700>
```



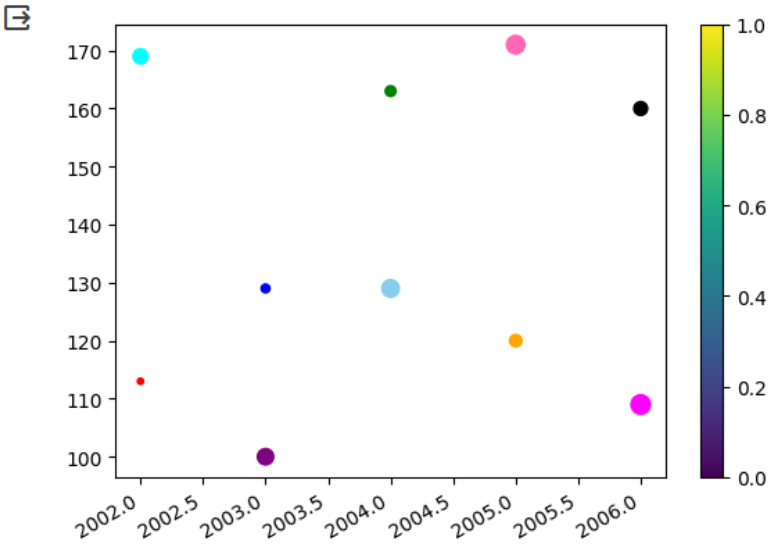
f. Set size of the dot.

```
color1 = ['red', 'blue', 'green', 'orange', 'black']
plt.scatter(year, tataStock, color=color1, s=50)

color2 = ['cyan', 'purple', 'skyblue', 'hotpink', 'magenta']
plt.scatter(year, relianceStock, color=color2, s=80)
plt.colorbar()
plt.gcf().autofmt_xdate()
```



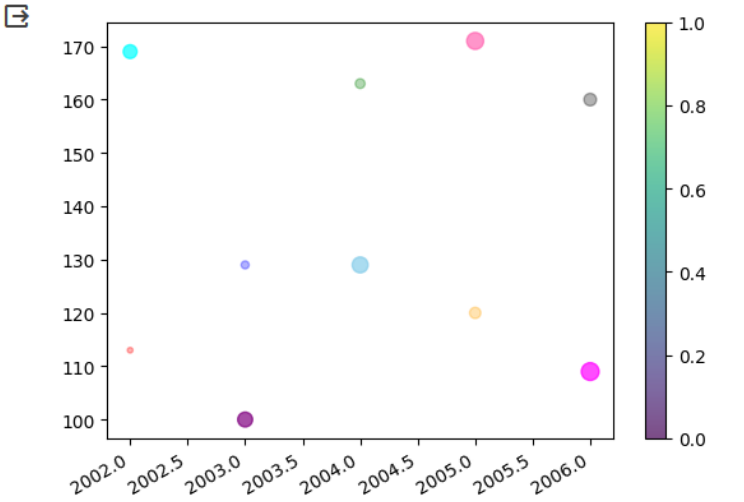
g. Set the different size of every dot.

```
color1 = ['red', 'blue', 'green', 'orange', 'black']
size1 = [10,20,30,40,50]
plt.scatter(year, tataStock, color=color1, s=size1)

color2 = ['cyan', 'purple', 'skyblue', 'hotpink', 'magenta']
size2 = [60,70,80,90,100]
plt.scatter(year, relianceStock, color=color2, s=size2)
plt.colorbar()
plt.gcf().autofmt_xdate()
```



g. Set the transparency of dot.

```
color1 = ['red', 'blue', 'green', 'orange', 'black']
size1 = [10,20,30,40,50]
plt.scatter(year, tataStock, color=color1, s=size1, alpha=0.3)

color2 = ['cyan', 'purple', 'skyblue', 'hotpink', 'magenta']
size2 = [60,70,80,90,100]
plt.scatter(year, relianceStock, color=color2, s=size2, alpha=0.7)
plt.colorbar()
plt.gcf().autofmt_xdate()
```



7. Draw a pie chart that shows the sales of a different fruit in a day for a

shop: 30% banana, 20% other, 15% orange, 10% guava, 25% mango

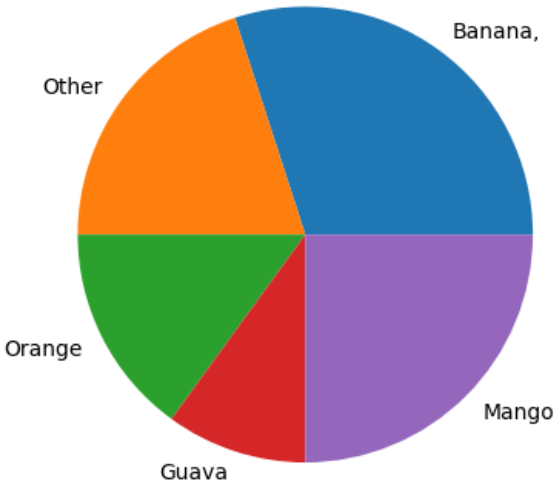
Use the following properties:

a. Draw basic pie chart using pie().

```
data = np.array([30,20,15,10,25])

plt.pie(data)
plt.show()
```

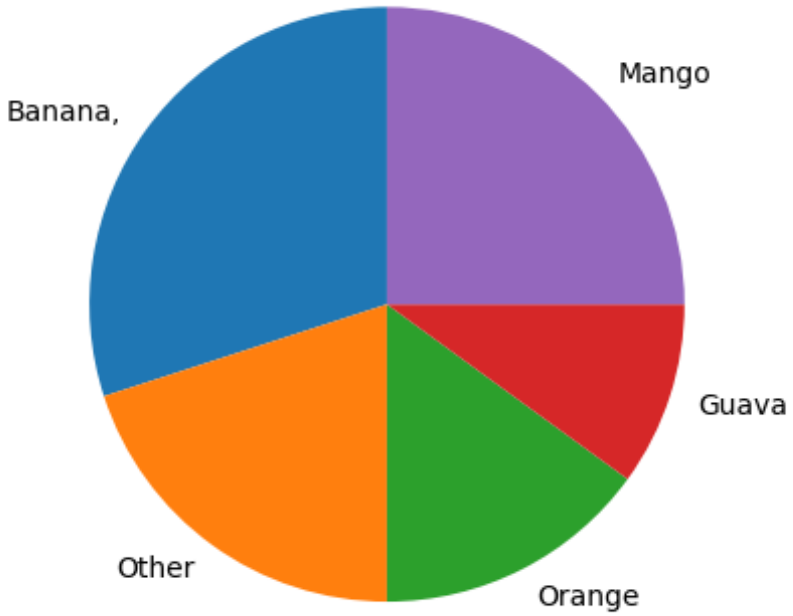
b. Add labels to wedges.

```
[142] fruits = np.array(['Banana,', 'Other', 'Orange', 'Guava', 'Mango'])

      plt.pie(data, labels=fruits)
      plt.show()
```



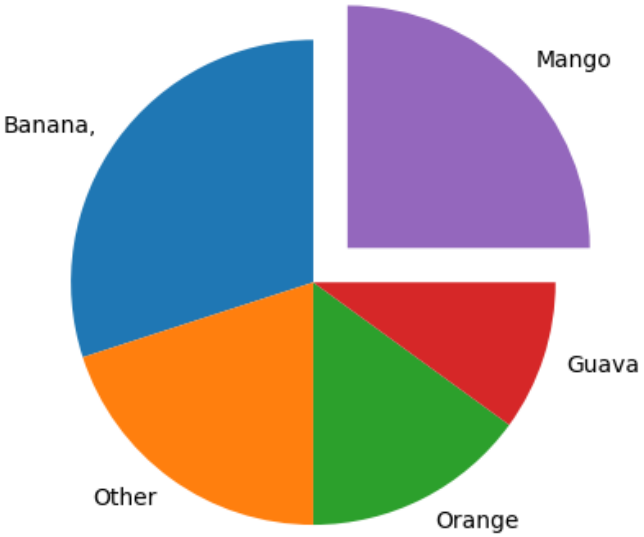
c. Change the start angle of any wedge.

```
[145] plt.pie(data, labels=fruits, startangle=90)
      plt.show()
```



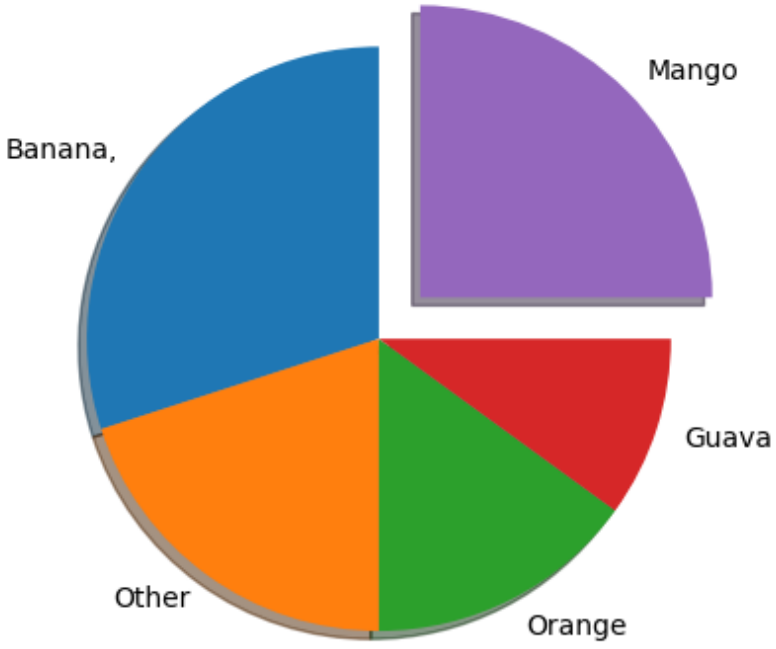
d. Displace the centre of wedge.

```
[147] exp = [0, 0, 0, 0, 0.2]
      plt.pie(data, labels=fruits, startangle=90, explode=exp)
      plt.show()
```



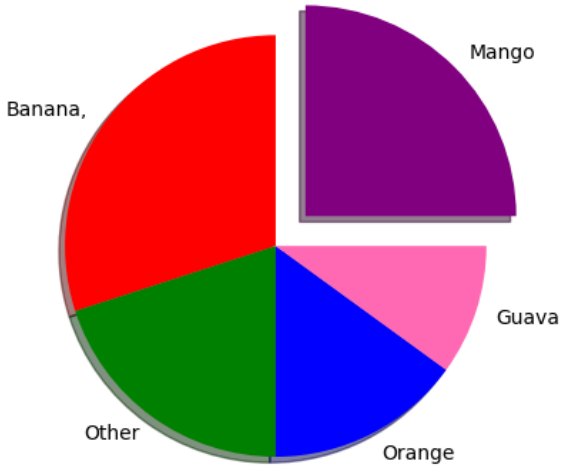
e. Add shadow to slice wedges.

```
exp = [0, 0, 0, 0, 0.2]
plt.pie(data, labels=fruits, startangle=90, explode=exp, shadow=True)
plt.show()
```



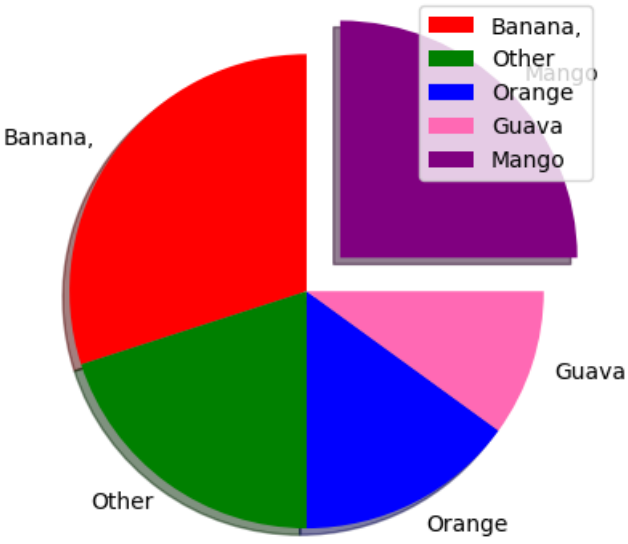
f. Change the colour of wedges

```
[151] col = ['red', 'green', 'blue', 'hotpink', 'purple']
      exp = [0, 0, 0, 0, 0.2]
      plt.pie(data, labels=fruits, startangle=90, explode=exp, shadow=True, colors=col)
      plt.show()
```



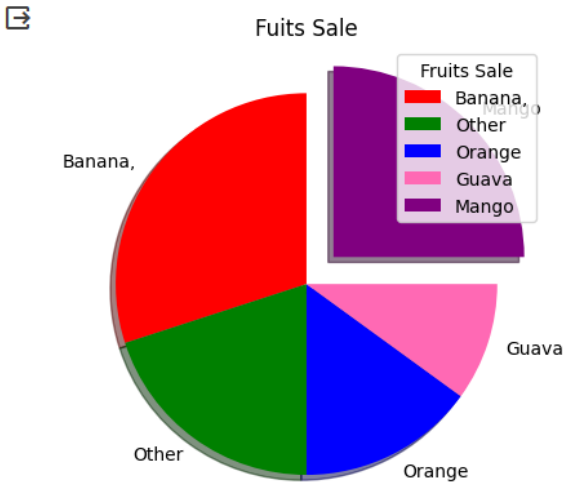
g. Add legend.

```
col = ['red', 'green', 'blue', 'hotpink', 'purple']
exp = [0, 0, 0, 0, 0.2]

plt.pie(data, labels=fruits, startangle=90, explode=exp, shadow=True, colors=col)

plt.legend()
plt.show()
```



h. Add legend with header.

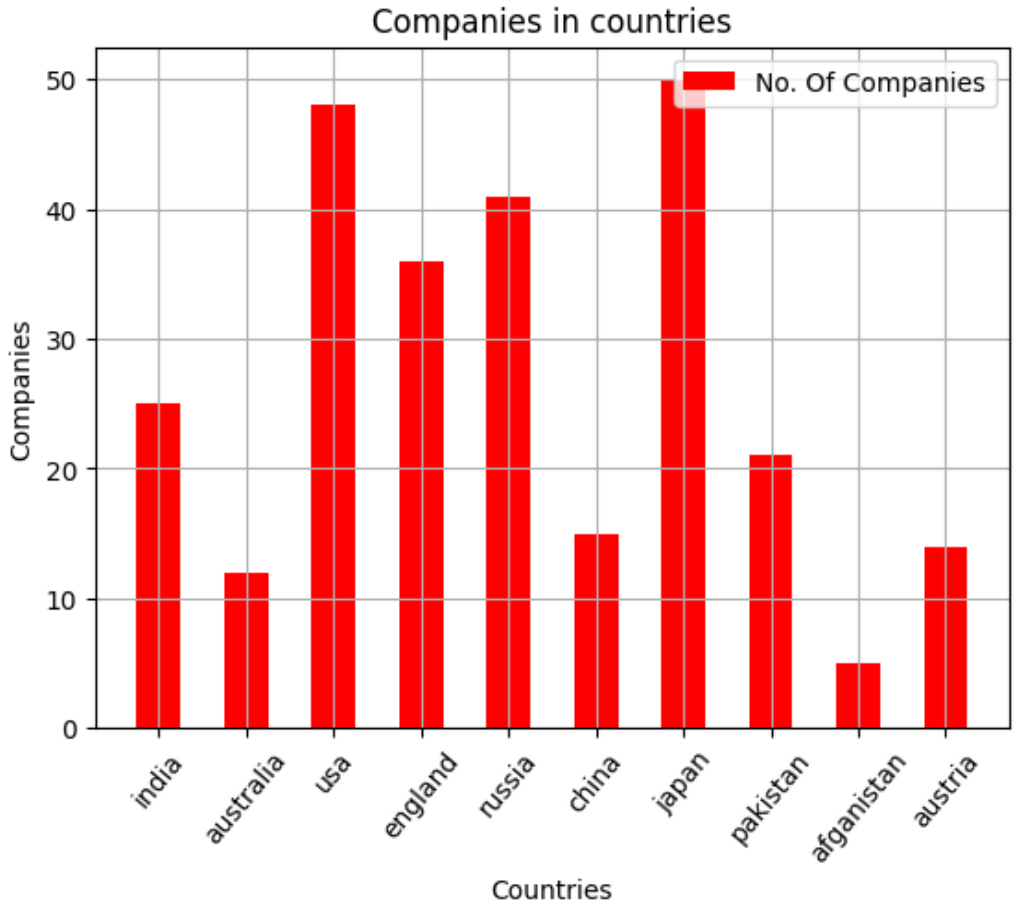
```
col = ['red', 'green', 'blue', 'hotpink', 'purple']
exp = [0, 0, 0, 0, 0.2]

plt.pie(data, labels=fruits, startangle=90, explode=exp, shadow=True, colors=col)

plt.title("Fuits Sale")
plt.legend(title="Fruits Sale")
plt.show()
```



8. Create a csv file displaying countries and no. Of companies in each

country. Display a bar graph from this csv file and style the bar graph too.

```
df = pd.read_csv('countries.csv')

x = df['country']
y = df['companies']

plt.bar(x, y, color = 'red', label='No. Of Companies', width=0.5)
plt.title("Companies in countries")
plt.xlabel("Countries")
plt.ylabel("Companies")
plt.grid()
plt.legend()
plt.xticks(rotation=50)
plt.show()
```



9. WAP to combine two data frames (data contained in two csv files – f1.csv country, no. Of companies and f2.csv – country, no. Of employees) Using merge (), join (), concat ().

DATA:

```
[44] df1 = pd.read_csv('f1.csv')
     df2 = pd.read_csv('f2.csv')
```

merge()

```
df3 = df1.merge(df2)
df3
```

	country	companies	employe
0	india	25	50
1	australia	12	23
2	usa	48	89
3	england	36	14
4	russia	41	85
5	china	15	22
6	japan	50	26
7	pakistan	21	52
8	afganistan	5	48
9	austria	14	78

join()

```
df3 = df1.join(df2.set_index('country'), on='country')
df3
```

	country	companies	employe
0	india	25	50
1	australia	12	23
2	usa	48	89
3	england	36	14
4	russia	41	85
5	china	15	22
6	japan	50	26
7	pakistan	21	52
8	afganistan	5	48
9	austria	14	78

concat()

```
[47] df4 = pd.concat([df1, df2], axis=1)
     df4
```

	country	companies	country	employe
0	india	25	india	50
1	australia	12	australia	23
2	usa	48	usa	89
3	england	36	england	14
4	russia	41	russia	85
5	china	15	china	22
6	japan	50	japan	26
7	pakistan	21	pakistan	52
8	afganistan	5	afganistan	48
9	austria	14	austria	78

9. Implement these graphs using seaborn library:-

a) **Lineplot**
b) **Distplot**
c) **Lmplot**
d) **Countplot**
e) **Relplot**
f) **Displot**
g) **Catplot**

Ouput:

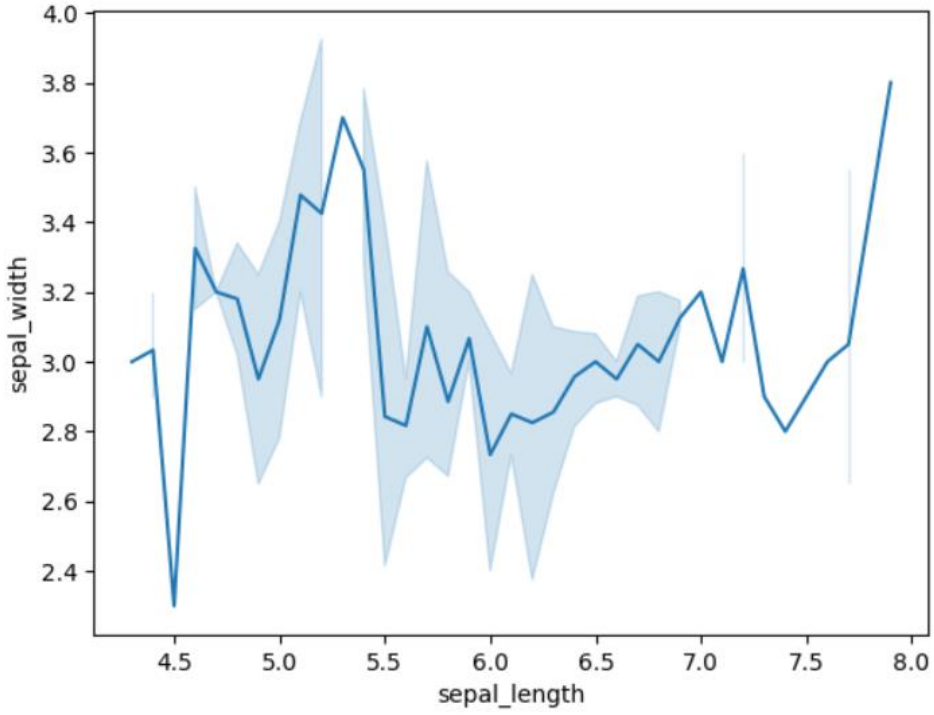
```
[2] import seaborn as sns
    import matplotlib.pyplot as plt
```

```
data = sns.load_dataset('tips')
```

a) Lineplot

```
sns.lineplot(data=d, x = 'sepal_length', y = 'sepal_width')
```

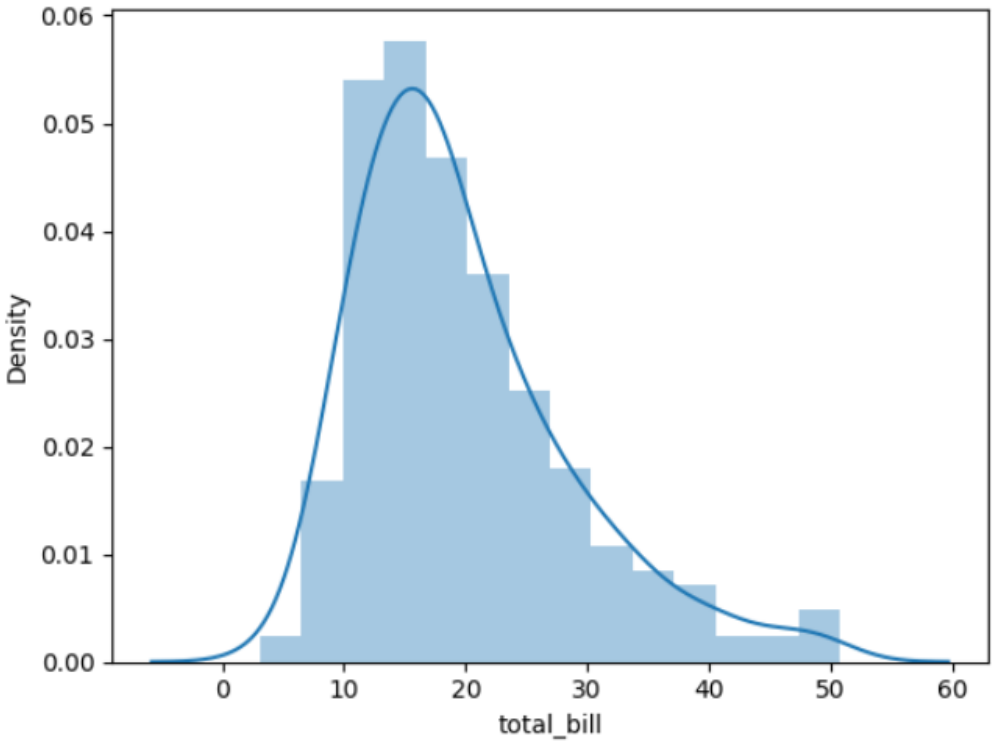
```
<Axes: xlabel='sepal_length', ylabel='sepal_width'>
```



b) Distplot

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

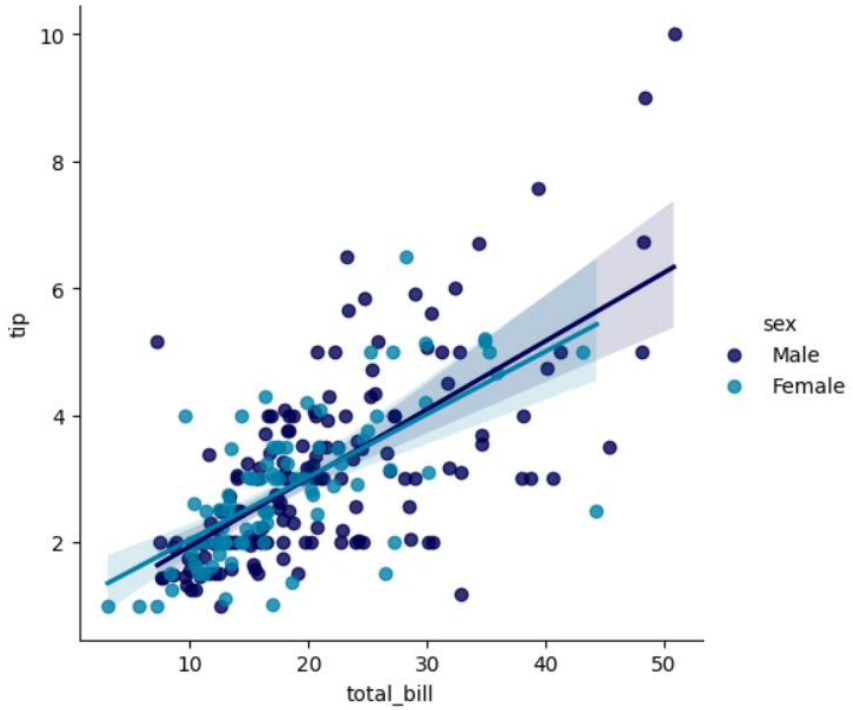
```
<ipython-input-12-86406deabd62>:1: UserWarning:

`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(data['total_bill'])
<Axes: xlabel='total_bill', ylabel='Density'>
```



c) Implot

```
sns.lmplot(x='total_bill', y='tip', data=data, hue='sex', palette='ocean')
```

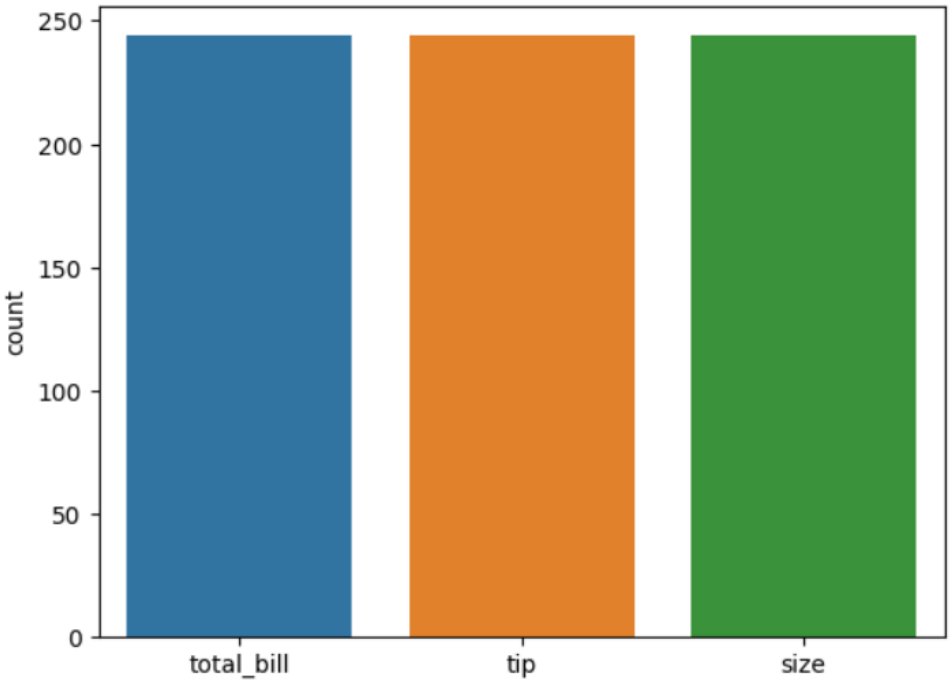
```
<seaborn.axisgrid.FacetGrid at 0x7bb23971bd60>
```



d) Countplot

```
[14] sns.countplot(data=data)
```

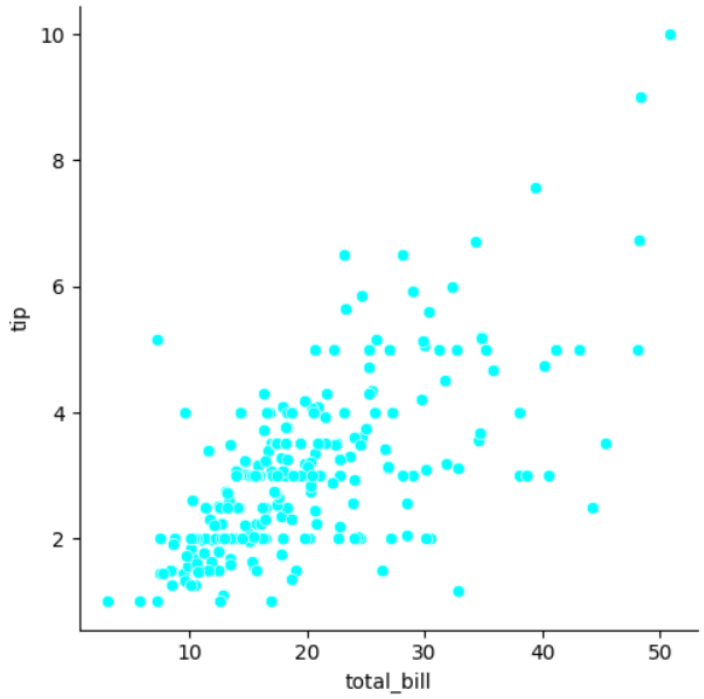
```
<Axes: ylabel='count'>
```



e) Relplot

```
sns.relplot(x="total_bill", y='tip', data = data, kind="scatter", color='cyan')
```

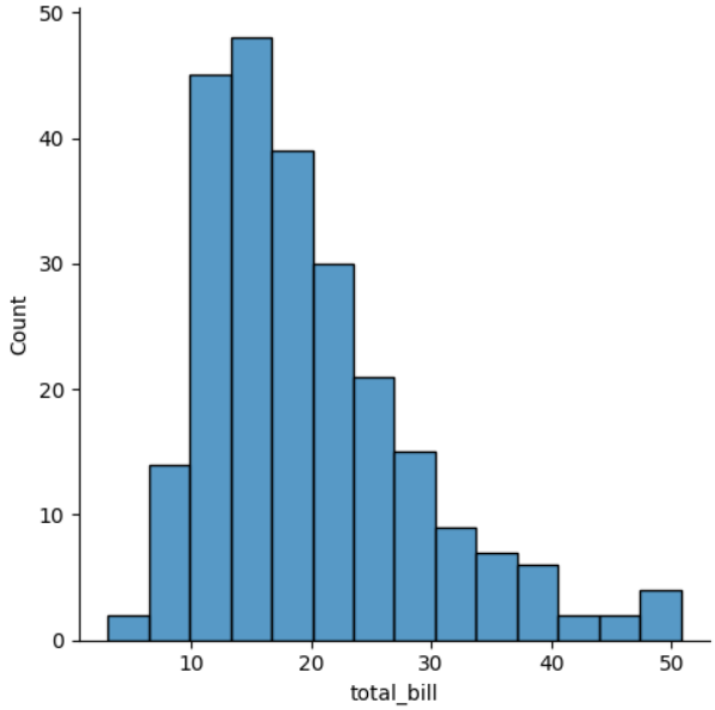
```
<seaborn.axisgrid.FacetGrid at 0x7bb2399febc0>
```



f) Displot

```
sns.displot(data['total_bill'])
```

```
<seaborn.axisgrid.FacetGrid at 0x7bb23b33a710>
```



g) Catplot

```
sns.catplot(x="day", y="total_bill", data=data, kind="violin")
```

```
<seaborn.axisgrid.FacetGrid at 0x7bb2398e5690>
```