```
EX 1.a BASIC PRACTICE EXPERIMENTS
REG NO:230701027
STUDENT NAME:AMIRTHAVARSHINI R U
DATE:30/07/2024
```

In [7]:
```
import pandas as pd
```

In [8]:
```
import matplotlib.pyplot as plt
```

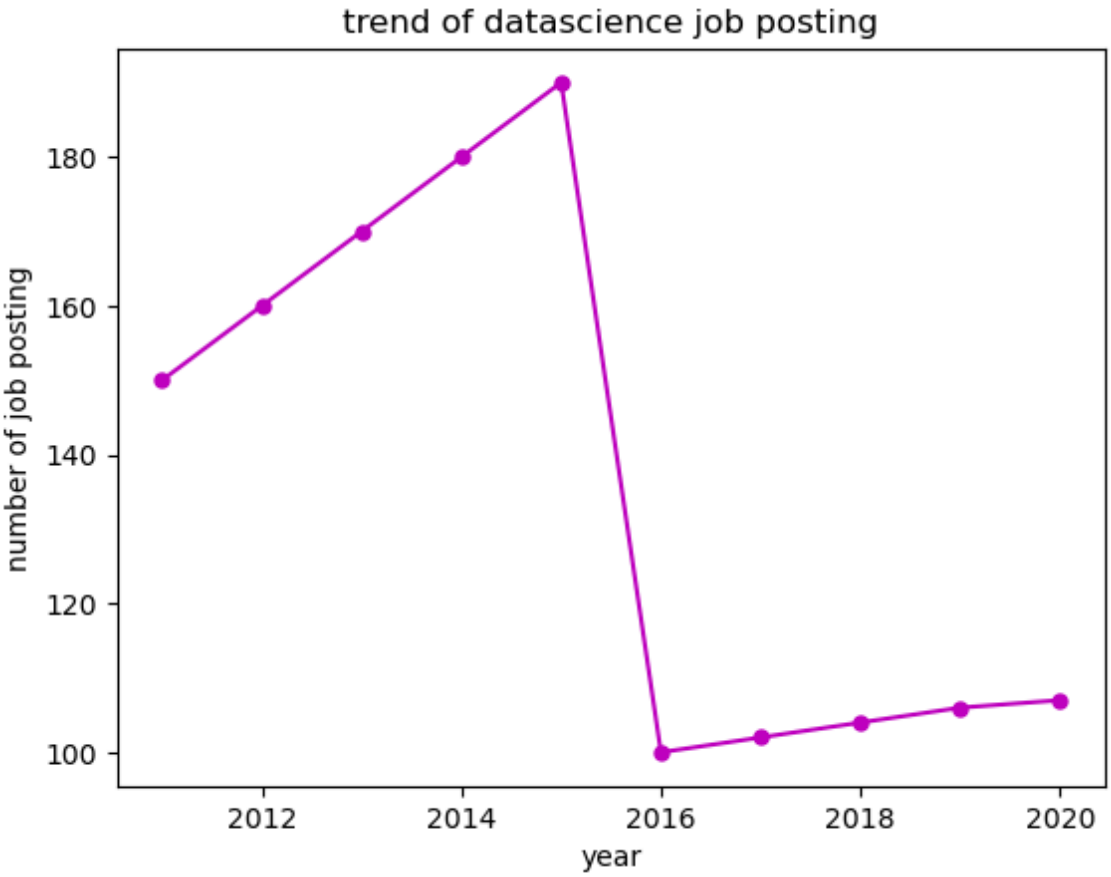
In [9]:
```
data = { 'year':list(range(2011,2021)),'job posting':[150,160,170,180,
df = pd.DataFrame(data)
```

In [10]:
```
df
```

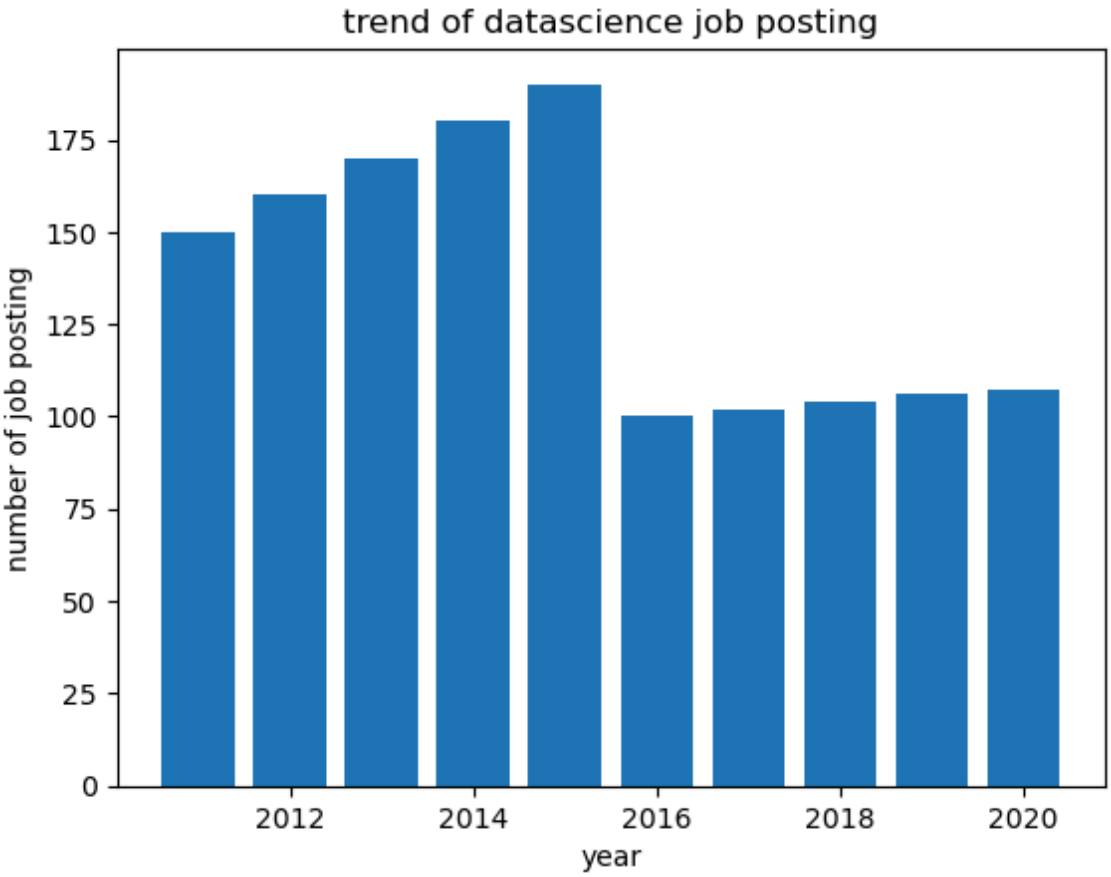
Out[10]:

	year	job posting
0	2011	150
1	2012	160
2	2013	170
3	2014	180
4	2015	190
5	2016	100
6	2017	102
7	2018	104
8	2019	106
9	2020	107

In [11]:
```
plt.plot(df['year'],df['job posting'],marker='o',color='m',ms=5)
plt.title("trend of datascience job posting")
plt.xlabel('year')
plt.ylabel('number of job posting')
plt.show()
```



In [12]:
```
plt.bar(df['year'],df['job posting'])
plt.title("trend of datascience job posting")
plt.xlabel('year')
plt.ylabel('number of job posting')
plt.show()
```



In [ ]:
```

```

```
EX 1.B NUMPY BUILDINH FUNCTIONS
REG NO:230701027
STUDENT NAME:AMIRTHAVARSHINI R U
DATE:30/07/2024
```

In [1]:
```
import numpy as np

num=np.random.randint(1,100,9)
print(num)
```

```
[15  6  8 40 24 45  2 41 68]
```

In [2]:
```
num=np.sqrt(num)
print(num)
```

```
[3.87298335 2.44948974 2.82842712 6.32455532 4.89897949 6.70820393
 1.41421356 6.40312424 8.24621125]
```

In [3]:
```
num=num.reshape(3,3)
print(num)
```

```
[[3.87298335 2.44948974 2.82842712]
 [6.32455532 4.89897949 6.70820393]
 [1.41421356 6.40312424 8.24621125]]
```

In [4]:
```
num.ndim
```

Out[4]: 2

In [5]:
```
num=num.ravel()
print(num)
```

```
[3.87298335 2.44948974 2.82842712 6.32455532 4.89897949 6.70820393
 1.41421356 6.40312424 8.24621125]
```

In [6]:
```
num.ndim
```

Out[6]: 1

In [7]:
```
num=num.reshape(3,3)
print(num)
```

```
[[3.87298335 2.44948974 2.82842712]
 [6.32455532 4.89897949 6.70820393]
 [1.41421356 6.40312424 8.24621125]]
```

In [8]:
```
num[1:1,1:1]
```

Out[8]: array([], shape=(0, 0), dtype=float64)

In [9]:
```
1 num[1:3,1:3]
```

Out[9]:
```
array([[4.89897949, 6.70820393],
       [6.40312424, 8.24621125]])
```

```
EX 2 OUTLIERS DETECTION
REG NO:230701027
STUDENT NAME:AMIRTHAVARSHINI R U
DATE:30/07/2024
```

In [1]:
```
import numpy as np
from scipy import stats
import seaborn as sns

num=np.random.randint(1,100,9)
print(num)
```

```
[58 94 62 22  3 53  2 80 50]
```

In [2]:
```
np.mean(num)
```

Out[2]:
```
47.111111111111114
```

In [3]:
```
np.median(num)
```

Out[3]:
```
53.0
```

In [4]:
```
stats.mode(num)
```

Out[4]:
```
ModeResult(mode=2, count=1)
```

In [5]:
```
def out(num):
    q1=np.percentile(num,25)
    q3=np.percentile(num,75)
    iqr=q3-q1
    lb=q1-(1.5*iqr)
    ub=q3+(1.5*iqr)
    return lb,ub
```

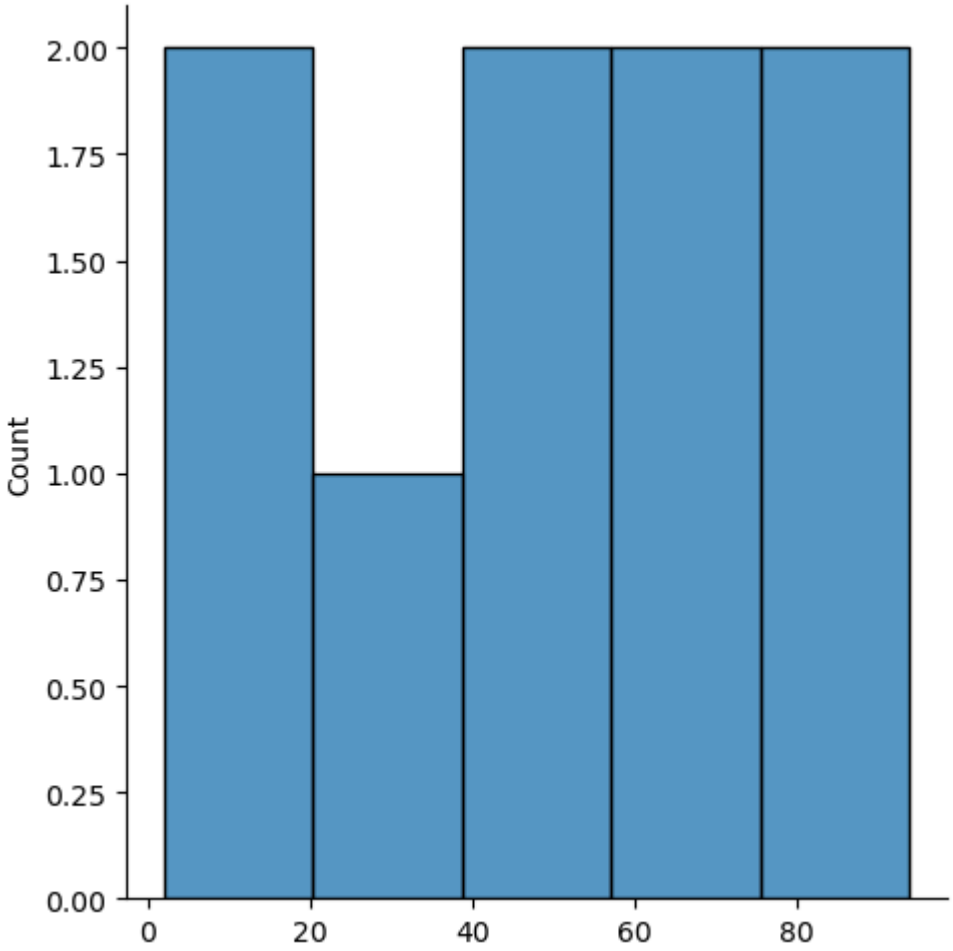
In [6]:
```
lb,ub=out(num)
print(lb)
print(ub) #range for the non outliers
```

```
-38.0
122.0
```

In [7]:
```
1 sns.displot(num)
```

```
C:\Users\amirt_erk6tkl\anaconda3\hi\Lib\site-packages\seaborn\axisgrid.p
y:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

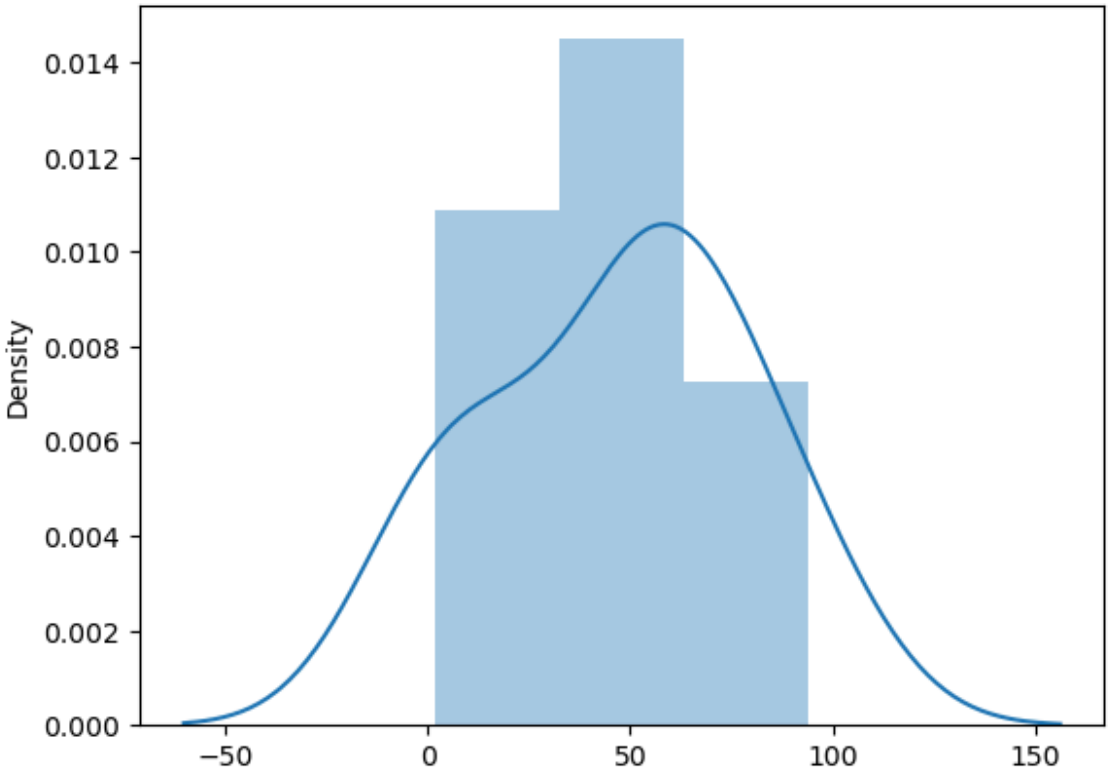
Out[7]: <seaborn.axisgrid.FacetGrid at 0x1a3bbd0e690>



In [8]:
```
1 sns.distplot(num)
```

```
C:\Users\amirt_erk6tkl\AppData\Local\Temp\ipykernel_26308\702644673.py: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 w
ith
similar flexibility) or `histplot` (an axes-level function for histogram
s).

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

  sns.distplot(num)
```

Out[8]:
```
<Axes: ylabel='Density'>
```



In [9]:
```
1 new_num=num[(lb<num) & (num<ub)] #return a boolean array which satisfi
2 #num[(lb<num) & (num<ub)]  this gives the numbers that are true that i
3 print(new_num)
```

```
[58 94 62 22  3 53  2 80 50]
```

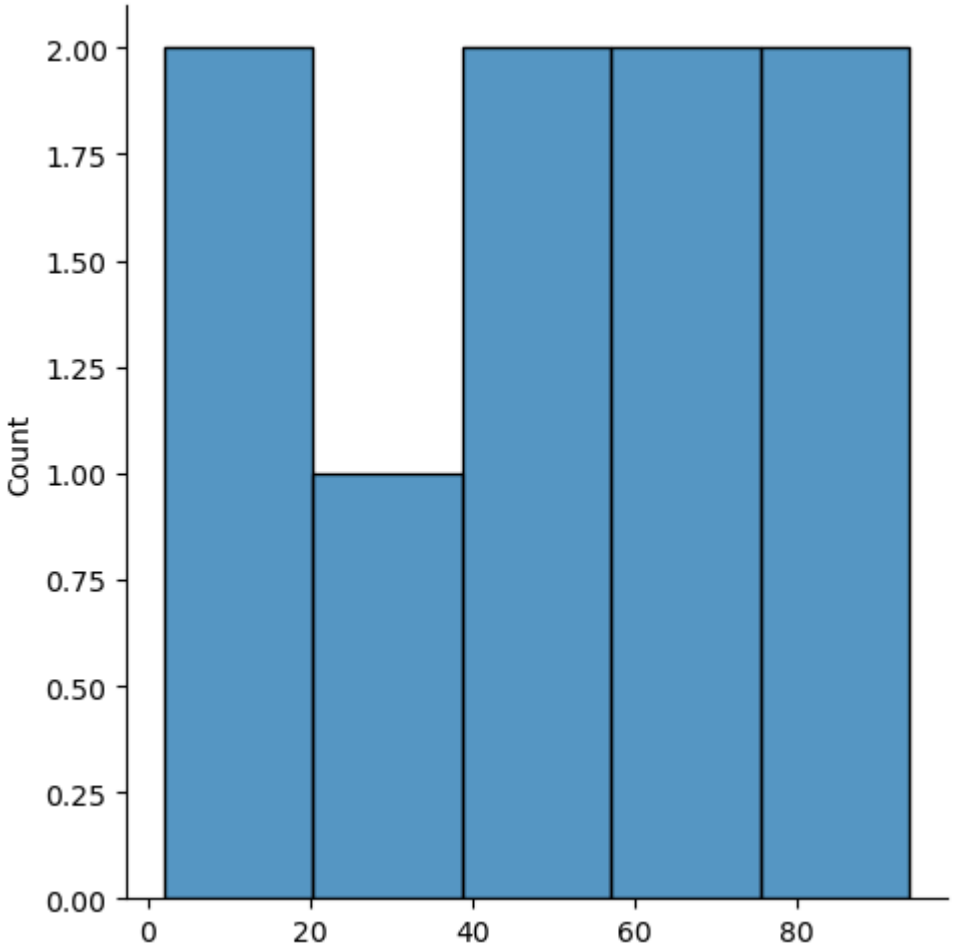
In [10]:
```
1 lb1,lb2=out(new_num)
2 print(lb1)
3 print(lb2)
```

```
-38.0
122.0
```

In [11]:
```
1 sns.displot(new_num)
```

```
C:\Users\amirt_erk6tkl\anaconda3\hi\Lib\site-packages\seaborn\axisgrid.p
y:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

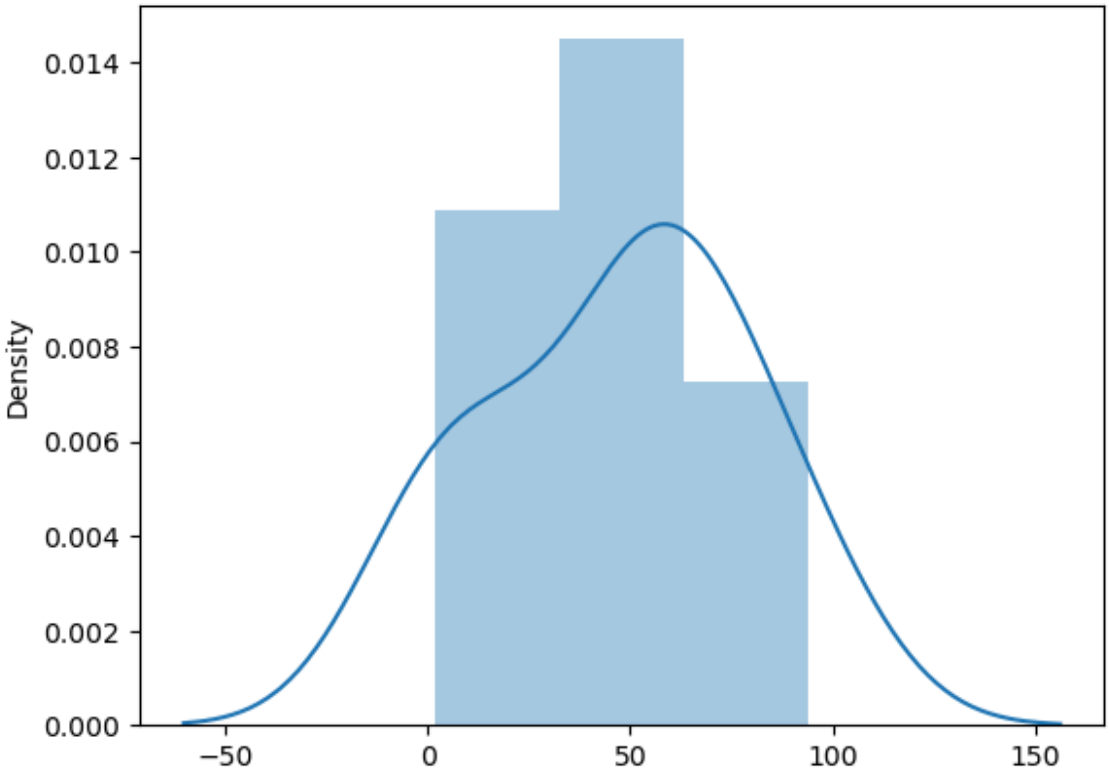
Out[11]: <seaborn.axisgrid.FacetGrid at 0x1a3bc42b350>



In [12]:
```
1 sns.distplot(new_num)
```

```
C:\Users\amirt_erk6tkl\AppData\Local\Temp\ipykernel_26308\1576028879.py:
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 w
ith
similar flexibility) or `histplot` (an axes-level function for histogram
s).

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

  sns.distplot(new_num)
```

Out[12]: <Axes: ylabel='Density'>



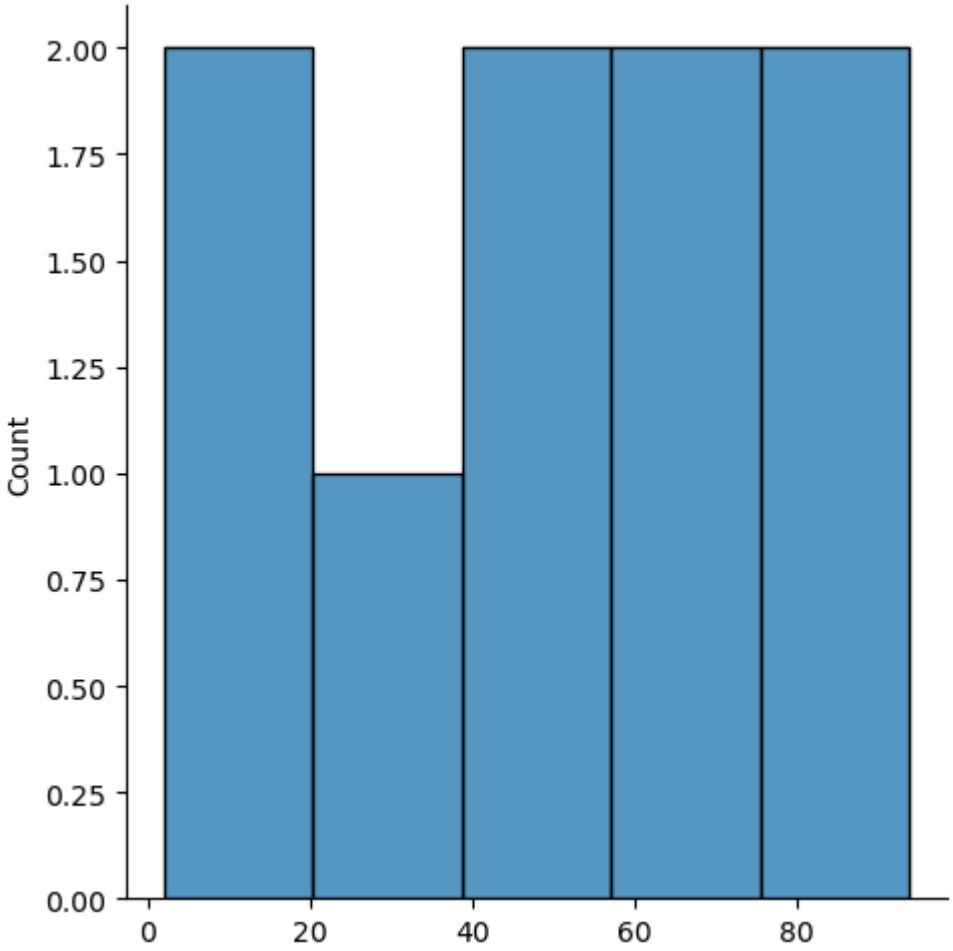
In [13]:
```
1 final_num=new_num[(lb<num) & (num<ub)]
2 #2 times verification
3 print(final_num)
```

```
[58 94 62 22  3 53  2 80 50]
```

In [14]:
```
1 sns.displot(final_num)
```

```
C:\Users\amirt_erk6tkl\anaconda3\hi\Lib\site-packages\seaborn\axisgrid.p
y:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

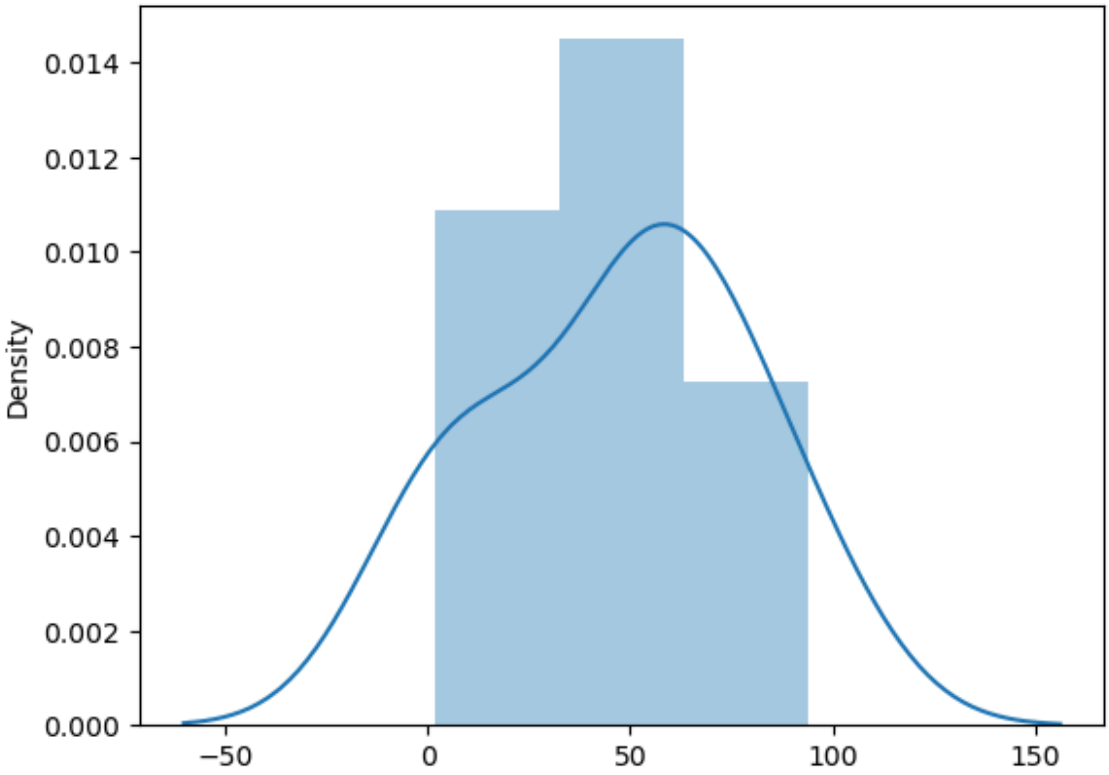
Out[14]: <seaborn.axisgrid.FacetGrid at 0x1a3bc59a610>



In [15]:
```
1 sns.distplot(final_num)
```

```
C:\Users\amirt_erk6tkl\AppData\Local\Temp\ipykernel_26308\1910024771.py:
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 w
ith
similar flexibility) or `histplot` (an axes-level function for histogram
s).

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

  sns.distplot(final_num)
```

Out[15]:
```
<Axes: ylabel='Density'>
```



In [16]:
```
1 print(final_num)
```

```
[58 94 62 22  3 53  2 80 50]
```

In [ ]:
```
1
```

```
EX 3 HANDLE MISSING VALUES
REG NO:230701027
STUDENT NAME:AMIRTHAVARSHINI R U
DATE:13/08/2024
```

```
info
find duplicate...drop duplicate
drop duplicate column
find its length set index (after removing)

finding negative values and filling with null
finding different values and replacing it (categorical)
find null...and its sum...fill null with mean(conti) mode(categori) median(discrete)

note:
    axis=0 row
    axis=1 column
    df.replace for - rows la
    df.hotel.replace and df["hotel"].replace -for columns la do ...specific column
    df.hotel.replace(df.hotel>1)...inside df.hotel is for checking all entries in df.hotel column


    df.hotel.replace....[doing replace only at hotel column]
    df.hotel.iloc(df.hotel,....)....[doing some operationn in hotel column and doing that for all elements in that column by iterating

```

In [1]:

```
import numpy as np
import pandas as pd

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

In [2]:
```
1 df
```

Out[2]:

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	Estimat
0	1	20-25	4	Ibis	veg	1300	2	
1	2	30-35	5	LemonTree	Non-Veg	2000	3	
2	3	25-30	6	RedFox	Veg	1322	2	
3	4	20-25	-1	LemonTree	Veg	1234	2	
4	5	35+	3	Ibis	Vegetarian	989	2	
5	6	35+	3	Ibys	Non-Veg	1909	2	
6	7	35+	4	RedFox	Vegetarian	1000	-1	
7	8	20-25	7	LemonTree	Veg	2999	-10	
8	9	25-30	2	Ibis	Non-Veg	3456	3	
9	9	25-30	2	Ibis	Non-Veg	3456	3	
10	10	30-35	5	RedFox	non-Veg	-6755	4	

In [3]:
```
1 df.duplicated()
```

Out[3]:
```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9      True
10    False
dtype: bool
```

In [4]:
```
1 df.drop_duplicates(inplace=True)
2
```

In [5]:
```
1 df
```

Out[5]:

	CustomerID	Age_Group	Rating(1-5)	Hotel	FoodPreference	Bill	NoOfPax	Estimat
0	1	20-25	4	Ibis	veg	1300	2	
1	2	30-35	5	LemonTree	Non-Veg	2000	3	
2	3	25-30	6	RedFox	Veg	1322	2	
3	4	20-25	-1	LemonTree	Veg	1234	2	
4	5	35+	3	Ibis	Vegetarian	989	2	
5	6	35+	3	Ibys	Non-Veg	1909	2	
6	7	35+	4	RedFox	Vegetarian	1000	-1	
7	8	20-25	7	LemonTree	Veg	2999	-10	
8	9	25-30	2	Ibis	Non-Veg	3456	3	
10	10	30-35	5	RedFox	non-Veg	-6755	4	

In [7]:
```
1 df.NoOfPox.loc(df["NoOfPox"]>0)==np.nan
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_16376\3044823116.py in ?()
----> 1 df.NoOfPox.loc(df["NoOfPox"]>0)==np.nan

~\anaconda3\hi\Lib\site-packages\pandas\core\generic.py in ?(self, name)
   5985             and name not in self._accessors
   5986             and self._info_axis._can_hold_identifiers_and_holds_name(name)
   5987         ):
   5988             return self[name]
-> 5989         return object.__getattribute__(self, name)

AttributeError: 'DataFrame' object has no attribute 'NoOfPox'
```

In [ ]:
```
1
```

```
EX 4 DATAPREPROCESSING
REG NO:230701027
STUDENT NAME:AMIRTHAVARSHINI R U
DATE:27/08/2024
```

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
file_path=r'D:\230701027\230701027\230701027\Students_data.csv'
```

In [3]:
```
df=pd.read_csv(file_path)
```

In [4]:
```
df
```

Out[4]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average	Date
0	1	arul	26	maths	A	1.0	chennai	98	100	NaN	12-11-2003
1	2	anu	27	physics	B	2.0	tiruvallur	99	99	NaN	13-11-2003
2	3	sam	29	chemistry	A	1.0	kanchi	97	98	NaN	14-11-2003
3	4	kaaviya	28	social	B	3.0	chennai	87	97	NaN	15-11-2003
4	5	anirudh	25	NaN	C	NaN	tpt	85	95	NaN	16-11-2003
5	6	amir	28	maths	D	2.0	chennai	97	87	NaN	17-11-2003
6	7	arav	30	NaN	A	NaN	salem	99	89	NaN	18-11-2003
7	8	ajeesh	29	tamil	B	2.0	cmbr	86	80	NaN	19-11-2003
8	9	arit	27	english	E	3.0	karur	85	83	NaN	20-11-2003
9	10	ajay	25	botany	D	3.0	chennai	89	98	NaN	21-11-2003
10	11	aks	28	zoology	A	NaN	tirunelveli	91	90	NaN	22-11-2003
11	12	ash	30	computer science	B	NaN	chennai	87	93	NaN	23-11-2003

In [5]:
```
df.head()
```

Out[5]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average	Date
0	1	arul	26	maths	A	1.0	chennai	98	100	NaN	12-11-2003
1	2	anu	27	physics	B	2.0	tiruvallur	99	99	NaN	13-11-2003
2	3	sam	29	chemistry	A	1.0	kanchi	97	98	NaN	14-11-2003
3	4	kaaviya	28	social	B	3.0	chennai	87	97	NaN	15-11-2003
4	5	anirudh	25	NaN	C	NaN	tpt	85	95	NaN	16-11-2003

In [6]:
```
1 df.head(8)
```

Out[6]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average	Date
0	1	arul	26	maths	A	1.0	chennai	98	100	NaN	12-11-2003
1	2	anu	27	physics	B	2.0	tiruvallur	99	99	NaN	13-11-2003
2	3	sam	29	chemistry	A	1.0	kanchi	97	98	NaN	14-11-2003
3	4	kaaviya	28	social	B	3.0	chennai	87	97	NaN	15-11-2003
4	5	anirudh	25	NaN	C	NaN	tpt	85	95	NaN	16-11-2003
5	6	amir	28	maths	D	2.0	chennai	97	87	NaN	17-11-2003
6	7	arav	30	NaN	A	NaN	salem	99	89	NaN	18-11-2003
7	8	ajeesh	29	tamil	B	2.0	cmbr	86	80	NaN	19-11-2003

In [7]:
```
1 df.tail()
```

Out[7]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average	Date
7	8	ajeesh	29	tamil	B	2.0	cmbr	86	80	NaN	19-11-2003
8	9	arit	27	english	E	3.0	karur	85	83	NaN	20-11-2003
9	10	ajay	25	botany	D	3.0	chennai	89	98	NaN	21-11-2003
10	11	aks	28	zoology	A	NaN	tirunelveli	91	90	NaN	22-11-2003
11	12	ash	30	computer science	B	NaN	chennai	87	93	NaN	23-11-2003

In [8]:
```
1 df.tail(8)
```

Out[8]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average	Date
4	5	anirudh	25	NaN	C	NaN	tpt	85	95	NaN	16-11-2003
5	6	amir	28	maths	D	2.0	chennai	97	87	NaN	17-11-2003
6	7	arav	30	NaN	A	NaN	salem	99	89	NaN	18-11-2003
7	8	ajeesh	29	tamil	B	2.0	cmbr	86	80	NaN	19-11-2003
8	9	arit	27	english	E	3.0	karur	85	83	NaN	20-11-2003
9	10	ajay	25	botany	D	3.0	chennai	89	98	NaN	21-11-2003
10	11	aks	28	zoology	A	NaN	tirunelveli	91	90	NaN	22-11-2003
11	12	ash	30	computer science	B	NaN	chennai	87	93	NaN	23-11-2003

In [9]:
```
1 df['Age']
```

Out[9]:
```
0     26
1     27
2     29
3     28
4     25
5     28
6     30
7     29
8     27
9     25
10    28
11    30
Name: Age, dtype: int64
```

In [10]:
```
1 df['City']
```

Out[10]:
```
0         chennai
1      tiruvallur
2          kanchi
3         chennai
4             tpt
5         chennai
6           salem
7            cmbr
8           karur
9         chennai
10    tirunelveli
11        chennai
Name: City, dtype: object
```

In [11]:
```
1 df.isnull().sum()
```

Out[11]:
```
ID          0
Name        0
Age         0
Centums     2
Section     0
Rank        4
City        0
Maths       0
Science     0
Average    12
Date        0
dtype: int64
```

In [12]:
```
1 df.dropna(subset=['Rank','Centums'],inplace=True)
```

In [13]:
```
1 df
```

Out[13]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average	Date
0	1	arul	26	maths	A	1.0	chennai	98	100	NaN	12-11-2003
1	2	anu	27	physics	B	2.0	tiruvallur	99	99	NaN	13-11-2003
2	3	sam	29	chemistry	A	1.0	kanchi	97	98	NaN	14-11-2003
3	4	kaaviya	28	social	B	3.0	chennai	87	97	NaN	15-11-2003
5	6	amir	28	maths	D	2.0	chennai	97	87	NaN	17-11-2003
7	8	ajeesh	29	tamil	B	2.0	cmbr	86	80	NaN	19-11-2003
8	9	arit	27	english	E	3.0	karur	85	83	NaN	20-11-2003
9	10	ajay	25	botany	D	3.0	chennai	89	98	NaN	21-11-2003

In [14]:
```
df['Rank'].fillna(2,inplace=True)
```

In [15]:
```
df
```

Out[15]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average	Date
0	1	arul	26	maths	A	1.0	chennai	98	100	NaN	12-11-2003
1	2	anu	27	physics	B	2.0	tiruvallur	99	99	NaN	13-11-2003
2	3	sam	29	chemistry	A	1.0	kanchi	97	98	NaN	14-11-2003
3	4	kaaviya	28	social	B	3.0	chennai	87	97	NaN	15-11-2003
5	6	amir	28	maths	D	2.0	chennai	97	87	NaN	17-11-2003
7	8	ajeesh	29	tamil	B	2.0	cmbr	86	80	NaN	19-11-2003
8	9	arit	27	english	E	3.0	karur	85	83	NaN	20-11-2003
9	10	ajay	25	botany	D	3.0	chennai	89	98	NaN	21-11-2003

In [16]:
```
a=df.dropna() #does not alters original dataframe
```

In [17]:
```
print(a.to_string())
```

```
Empty DataFrame
Columns: [ID, Name, Age, Centums, Section, Rank, City, Maths, Science, Av
erage, Date]
Index: []
```

In [18]:
```
df.dropna(inplace=True) #alters original dataframe
```

In [19]:
```
df['Maths'].mean()
```

Out[19]: nan

In [20]:
```
df['Science'].mean()
```

Out[20]: nan

In [21]:
```
df['Maths'].median() #after sorted in ascending order...finding the mi
```

Out[21]: nan

In [22]:
```
df['Science'].median() #after sorted in ascending order...finding the
```

Out[22]: nan

In [23]:
```
df['Maths'].mode() #all below four occurse equallyy 2 times
```

Out[23]:
```
Series([], Name: Maths, dtype: int64)
```

In [24]:
```
df['Science'].mode() #most frequently present
```

Out[24]:
```
Series([], Name: Science, dtype: int64)
```

In [25]:
```
df['Average'].fillna(df['Maths'].mean(),inplace=True)
```

In [26]:
```
df
```

Out[26]:

ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average	Date

In [27]:
```
df['Average'].fillna(df['Science'].mean(),inplace=True)
```

In [28]:
```
df
```

Out[28]:

ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average	Date

In [29]:
```
df.dtypes
```

Out[29]:
```
ID           int64
Name        object
Age          int64
Centums     object
Section     object
Rank       float64
City        object
Maths        int64
Science      int64
Average    float64
Date        object
dtype: object
```

In [30]:
```
df['Age'] = df['Age'].astype(int)
df['Rank'] = df['Rank'].astype(int)
df['Maths'] = df['Maths'].astype(float)
df['Science'] = df['Science'].astype(float)

```

In [31]:
```
df
```

Out[31]:

ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average	Date

In [32]:
```
1 df.dtypes
```

Out[32]:
```
ID           int64
Name        object
Age          int32
Centums     object
Section     object
Rank         int32
City        object
Maths      float64
Science    float64
Average    float64
Date        object
dtype: object
```

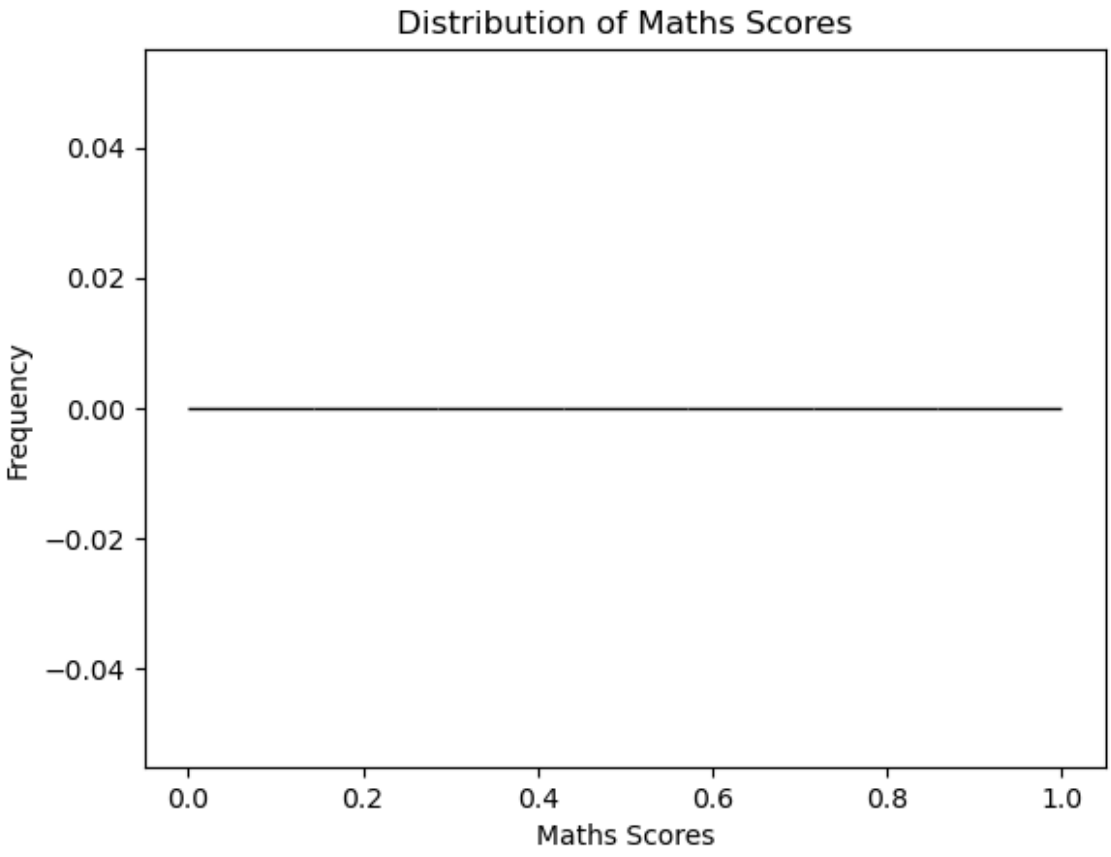
In [33]:
```
1 df.describe()
```

Out[33]:

	ID	Age	Rank	Maths	Science	Average
count	0.0	0.0	0.0	0.0	0.0	0.0
mean	NaN	NaN	NaN	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN	NaN

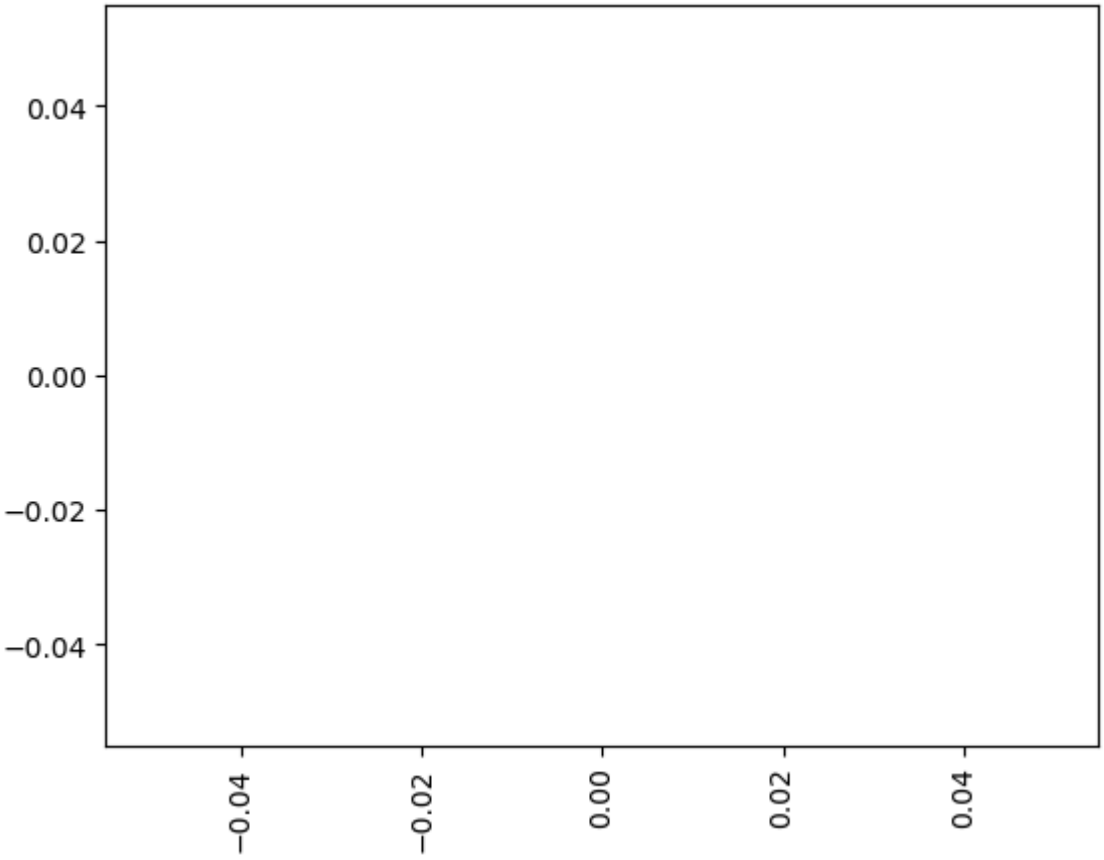
In [34]:

```
plt.hist(df['Maths'], bins=7,color='blue', edgecolor='black')
plt.title('Distribution of Maths Scores')
plt.xlabel('Maths Scores')
plt.ylabel('Frequency')
plt.show()
#total range = max score - min score
#bins means wanting the total range in the specified interval in such
#here 14 is the total range
#when bins is 7 ...so there will be 7 intervals
#if bins is 1...then there will be a one range from 85 to 99
#total range/given bins = range btwn each interval
```



In [35]:
```
df.groupby('Section')['Maths'].mean().plot(kind='bar', color='green')
plt.title('Average Maths Scores by Section')
plt.xlabel('Section')
plt.ylabel('Average Maths Score')
plt.show()
#bar chart for the new dataframe which is grouped based on section (ave
```

```
---------------------------------------------------------------------
--
IndexError                                Traceback (most recent call las
t)
Cell In[35], line 1
----> 1 df.groupby('Section')['Maths'].mean().plot(kind='bar', color='gre
en')
      2 plt.title('Average Maths Scores by Section')
      3 plt.xlabel('Section')

File ~\anaconda3\hi\Lib\site-packages\pandas\plotting\_core.py:975, in Pl
otAccessor.__call__(self, *args, **kwargs)
    972             label_name = label_kw or data.columns
    973             data.columns = label_name
--> 975 return plot_backend.plot(data, kind=kind, **kwargs)

File ~\anaconda3\hi\Lib\site-packages\pandas\plotting\_matplotlib\__init_
_.py:71, in plot(data, kind, **kwargs)
     69         kwargs["ax"] = getattr(ax, "left_ax", ax)
     70 plot_obj = PLOT_CLASSES[kind](data, **kwargs)
---> 71 plot_obj.generate()
     72 plot_obj.draw()
     73 return plot_obj.result

File ~\anaconda3\hi\Lib\site-packages\pandas\plotting\_matplotlib\core.p
y:455, in MPLPlot.generate(self)
    453 for ax in self.axes:
    454     self._post_plot_logic_common(ax, self.data)
--> 455     self._post_plot_logic(ax, self.data)

File ~\anaconda3\hi\Lib\site-packages\pandas\plotting\_matplotlib\core.p
y:1750, in BarPlot._post_plot_logic(self, ax, data)
   1747 else:
   1748     str_index = [pprint_thing(key) for key in range(data.shape
[0])]
-> 1750 s_edge = self.ax_pos[0] - 0.25 + self.lim_offset
   1751 e_edge = self.ax_pos[-1] + 0.25 + self.bar_width + self.lim_offse
t
   1753 self._decorate_ticks(ax, self._get_index_name(), str_index, s_edg
e, e_edge)

IndexError: index 0 is out of bounds for axis 0 with size 0
```



In [ ]:
```
df.groupby('Section')['Maths'].mean()
df.plot(kind='bar', color='green')
plt.title('Students Data')
plt.xlabel('ID')
plt.ylabel('Range')
plt.show()
#bar plot based on entire dataframe (all attributes)

```

In [ ]:
```
plt.scatter(df['Maths'], df['Science'], color='purple')
plt.title('Maths vs. Science Scores')
plt.xlabel('Maths Scores')
plt.ylabel('Science Scores')
plt.show()
#normal like plotting points in x and y axis
```

In [ ]:
```
df['Maths'].value_counts().plot(kind='pie', autopct='%.2f%%')
plt.title('Maths Marks')
plt.show()
#counts no.of occurence for each value in Maths attribute according to
```

In [ ]:
```
df['Section'].value_counts().plot(kind='pie', autopct='%.2f%%') #autop
plt.title('Proportion of Students in Each Section')
plt.show()
#no of occurence of each section which means gives no. of students in
```

In [ ]:
```
df['Name'].value_counts().plot(kind='pie', autopct='%.2f%%')
plt.title('Students')
plt.show()
#all names are occured only once...so equal percentage
```

In [ ]:
```
sns.boxplot(x='Section', y='Maths', data=df)
plt.title('Box Plot of Maths Scores by Section')
plt.xlabel('Section')
plt.ylabel('Maths Scores')
plt.show()
#its based on IQR
```

In [ ]:
```
sns.heatmap(df[['Maths', 'Science', 'Average']].corr(),annot=True, cma
plt.title('Correlation Heatmap')
plt.show()
#corr means correlation gives matrix
#corr is found using certain formula
#annot to display the correlation coefficients inside the cell
#1 indicates a perfect positive correlation,0 indicates no correlation
#cool colors represent negative correlations, and warm colors represen
```

In [ ]:
```
plt.hist(df['Maths'])
plt.show()
#default bins is 10
```

In [ ]:
```

```

In [36]:
```
file_path=r'D:\230701027\230701027\230701027\Students_data.csv'
```

In [37]:
```
df=pd.read_csv(file_path)
```

In [38]:
```
df
```

Out[38]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average	Date
0	1	arul	26	maths	A	1.0	chennai	98	100	NaN	12-11-2003
1	2	anu	27	physics	B	2.0	tiruvallur	99	99	NaN	13-11-2003
2	3	sam	29	chemistry	A	1.0	kanchi	97	98	NaN	14-11-2003
3	4	kaaviya	28	social	B	3.0	chennai	87	97	NaN	15-11-2003
4	5	anirudh	25	NaN	C	NaN	tpt	85	95	NaN	16-11-2003
5	6	amir	28	maths	D	2.0	chennai	97	87	NaN	17-11-2003
6	7	arav	30	NaN	A	NaN	salem	99	89	NaN	18-11-2003
7	8	ajeesh	29	tamil	B	2.0	cmbr	86	80	NaN	19-11-2003
8	9	arit	27	english	E	3.0	karur	85	83	NaN	20-11-2003
9	10	ajay	25	botany	D	3.0	chennai	89	98	NaN	21-11-2003
10	11	aks	28	zoology	A	NaN	tirunelveli	91	90	NaN	22-11-2003
11	12	ash	30	computer science	B	NaN	chennai	87	93	NaN	23-11-2003

In [39]:
```
df['Average'].fillna(df['Maths'].mean(),inplace=True)
```

In [40]:
```
1 df.dropna()
```

Out[40]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average	Date
0	1	arul	26	maths	A	1.0	chennai	98	100	91.666667	12-11-2003
1	2	anu	27	physics	B	2.0	tiruvallur	99	99	91.666667	13-11-2003
2	3	sam	29	chemistry	A	1.0	kanchi	97	98	91.666667	14-11-2003
3	4	kaaviya	28	social	B	3.0	chennai	87	97	91.666667	15-11-2003
5	6	amir	28	maths	D	2.0	chennai	97	87	91.666667	17-11-2003
7	8	ajeesh	29	tamil	B	2.0	cmbr	86	80	91.666667	19-11-2003
8	9	arit	27	english	E	3.0	karur	85	83	91.666667	20-11-2003
9	10	ajay	25	botany	D	3.0	chennai	89	98	91.666667	21-11-2003

In [41]:
```
df
```

Out[41]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average	Date
0	1	arul	26	maths	A	1.0	chennai	98	100	91.666667	12-11-2003
1	2	anu	27	physics	B	2.0	tiruvallur	99	99	91.666667	13-11-2003
2	3	sam	29	chemistry	A	1.0	kanchi	97	98	91.666667	14-11-2003
3	4	kaaviya	28	social	B	3.0	chennai	87	97	91.666667	15-11-2003
4	5	anirudh	25	NaN	C	NaN	tpt	85	95	91.666667	16-11-2003
5	6	amir	28	maths	D	2.0	chennai	97	87	91.666667	17-11-2003
6	7	arav	30	NaN	A	NaN	salem	99	89	91.666667	18-11-2003
7	8	ajeesh	29	tamil	B	2.0	cmbr	86	80	91.666667	19-11-2003
8	9	arit	27	english	E	3.0	karur	85	83	91.666667	20-11-2003
9	10	ajay	25	botany	D	3.0	chennai	89	98	91.666667	21-11-2003
10	11	aks	28	zoology	A	NaN	tirunelveli	91	90	91.666667	22-11-2003
11	12	ash	30	computer science	B	NaN	chennai	87	93	91.666667	23-11-2003

In [42]:
```
df.dropna(inplace=True)
```

In [43]:
```
date='23/7/2003'
```

In [44]:
```
date=pd.to_datetime(date)
```

```
C:\Users\amirt_erk6tkl\AppData\Local\Temp\ipykernel_26580\1716004735.py:
1: UserWarning: Parsing dates in %d/%m/%Y format when dayfirst=False (the
default) was specified. Pass `dayfirst=True` or specify a format to silen
ce this warning.
  date=pd.to_datetime(date)
```

In [45]:
```
1 date
```

Out[45]:
```
Timestamp('2003-07-23 00:00:00')
```

In [46]:
```
1 df['Age']
```

Out[46]:
```
0    26
1    27
2    29
3    28
5    28
7    29
8    27
9    25
Name: Age, dtype: int64
```

In [47]:
```
1 df['Work']=np.nan
```

In [48]:
```
1 df
```

Out[48]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average	Date	W
0	1	arul	26	maths	A	1.0	chennai	98	100	91.666667	12-11-2003	N
1	2	anu	27	physics	B	2.0	tiruvallur	99	99	91.666667	13-11-2003	N
2	3	sam	29	chemistry	A	1.0	kanchi	97	98	91.666667	14-11-2003	N
3	4	kaaviya	28	social	B	3.0	chennai	87	97	91.666667	15-11-2003	N
5	6	amir	28	maths	D	2.0	chennai	97	87	91.666667	17-11-2003	N
7	8	ajeesh	29	tamil	B	2.0	cmbr	86	80	91.666667	19-11-2003	N
8	9	arit	27	english	E	3.0	karur	85	83	91.666667	20-11-2003	N
9	10	ajay	25	botany	D	3.0	chennai	89	98	91.666667	21-11-2003	N

In [49]:
```
df.loc[0,'Work']=5
df.loc[2,'Work']=10
df.loc[3,'Work']=15
df.loc[7,'Work']=20
df.loc[8,'Work']=25

```

In [50]:
```
df
```

Out[50]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average	Date	W
0	1	arul	26	maths	A	1.0	chennai	98	100	91.666667	12-11-2003	
1	2	anu	27	physics	B	2.0	tiruvallur	99	99	91.666667	13-11-2003	N
2	3	sam	29	chemistry	A	1.0	kanchi	97	98	91.666667	14-11-2003	
3	4	kaaviya	28	social	B	3.0	chennai	87	97	91.666667	15-11-2003	
5	6	amir	28	maths	D	2.0	chennai	97	87	91.666667	17-11-2003	N
7	8	ajeesh	29	tamil	B	2.0	cmbr	86	80	91.666667	19-11-2003	
8	9	arit	27	english	E	3.0	karur	85	83	91.666667	20-11-2003	
9	10	ajay	25	botany	D	3.0	chennai	89	98	91.666667	21-11-2003	N

In [51]:
```
df['Work'].fillna(8,inplace=True)

```

In [52]:
```
1 df
```

Out[52]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average	Date	W
0	1	arul	26	maths	A	1.0	chennai	98	100	91.666667	12-11-2003	
1	2	anu	27	physics	B	2.0	tiruvallur	99	99	91.666667	13-11-2003	
2	3	sam	29	chemistry	A	1.0	kanchi	97	98	91.666667	14-11-2003	
3	4	kaaviya	28	social	B	3.0	chennai	87	97	91.666667	15-11-2003	
5	6	amir	28	maths	D	2.0	chennai	97	87	91.666667	17-11-2003	
7	8	ajeesh	29	tamil	B	2.0	cmbr	86	80	91.666667	19-11-2003	
8	9	arit	27	english	E	3.0	karur	85	83	91.666667	20-11-2003	
9	10	ajay	25	botany	D	3.0	chennai	89	98	91.666667	21-11-2003	

In [53]:
```
1 df.loc[8,'Work']=np.nan
2
```

In [54]:
```
df
```

Out[54]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average	Date	W
0	1	arul	26	maths	A	1.0	chennai	98	100	91.666667	12-11-2003	
1	2	anu	27	physics	B	2.0	tiruvallur	99	99	91.666667	13-11-2003	
2	3	sam	29	chemistry	A	1.0	kanchi	97	98	91.666667	14-11-2003	
3	4	kaaviya	28	social	B	3.0	chennai	87	97	91.666667	15-11-2003	
5	6	amir	28	maths	D	2.0	chennai	97	87	91.666667	17-11-2003	
7	8	ajeesh	29	tamil	B	2.0	cmbr	86	80	91.666667	19-11-2003	
8	9	arit	27	english	E	3.0	karur	85	83	91.666667	20-11-2003	
9	10	ajay	25	botany	D	3.0	chennai	89	98	91.666667	21-11-2003	

In [55]:
```
df.dropna(axis=1)
```

Out[55]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average	Date
0	1	arul	26	maths	A	1.0	chennai	98	100	91.666667	12-11-2003
1	2	anu	27	physics	B	2.0	tiruvallur	99	99	91.666667	13-11-2003
2	3	sam	29	chemistry	A	1.0	kanchi	97	98	91.666667	14-11-2003
3	4	kaaviya	28	social	B	3.0	chennai	87	97	91.666667	15-11-2003
5	6	amir	28	maths	D	2.0	chennai	97	87	91.666667	17-11-2003
7	8	ajeesh	29	tamil	B	2.0	cmbr	86	80	91.666667	19-11-2003
8	9	arit	27	english	E	3.0	karur	85	83	91.666667	20-11-2003
9	10	ajay	25	botany	D	3.0	chennai	89	98	91.666667	21-11-2003

In [56]:
```
df.dropna(inplace=True,axis=1)
```

In [57]:
```
df
```

Out[57]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average	Date
0	1	arul	26	maths	A	1.0	chennai	98	100	91.666667	12-11-2003
1	2	anu	27	physics	B	2.0	tiruvallur	99	99	91.666667	13-11-2003
2	3	sam	29	chemistry	A	1.0	kanchi	97	98	91.666667	14-11-2003
3	4	kaaviya	28	social	B	3.0	chennai	87	97	91.666667	15-11-2003
5	6	amir	28	maths	D	2.0	chennai	97	87	91.666667	17-11-2003
7	8	ajeesh	29	tamil	B	2.0	cmbr	86	80	91.666667	19-11-2003
8	9	arit	27	english	E	3.0	karur	85	83	91.666667	20-11-2003
9	10	ajay	25	botany	D	3.0	chennai	89	98	91.666667	21-11-2003

In [58]:
```
df['Science'].mode()
```

Out[58]:
```
0    98
Name: Science, dtype: int64
```

In [59]:
```
df['Maths'].mode()[0]
```

Out[59]:
```
97
```

In [60]:
```
df['Maths'].median()
```

Out[60]:
```
93.0
```

In [61]:
```
for i in df.index:
    print(df.loc[i,'Maths'])
```

```
98
99
97
87
97
86
85
89
```

In [62]:
```
df.loc[4,'Maths']=np.nan
df.loc[6,'Maths']=np.nan
```

In [63]:
```
1 df
```

Out[63]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average	Date
0	1.0	arul	26.0	maths	A	1.0	chennai	98.0	100.0	91.666667	12-11-2003
1	2.0	anu	27.0	physics	B	2.0	tiruvallur	99.0	99.0	91.666667	13-11-2003
2	3.0	sam	29.0	chemistry	A	1.0	kanchi	97.0	98.0	91.666667	14-11-2003
3	4.0	kaaviya	28.0	social	B	3.0	chennai	87.0	97.0	91.666667	15-11-2003
5	6.0	amir	28.0	maths	D	2.0	chennai	97.0	87.0	91.666667	17-11-2003
7	8.0	ajeesh	29.0	tamil	B	2.0	cmbr	86.0	80.0	91.666667	19-11-2003
8	9.0	arit	27.0	english	E	3.0	karur	85.0	83.0	91.666667	20-11-2003
9	10.0	ajay	25.0	botany	D	3.0	chennai	89.0	98.0	91.666667	21-11-2003
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [64]:
```
1 df.duplicated()
```

Out[64]:
```
0    False
1    False
2    False
3    False
5    False
7    False
8    False
9    False
4    False
6     True
dtype: bool
```

In [65]:
```
1 df
```

Out[65]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average	Date
0	1.0	arul	26.0	maths	A	1.0	chennai	98.0	100.0	91.666667	12-11-2003
1	2.0	anu	27.0	physics	B	2.0	tiruvallur	99.0	99.0	91.666667	13-11-2003
2	3.0	sam	29.0	chemistry	A	1.0	kanchi	97.0	98.0	91.666667	14-11-2003
3	4.0	kaaviya	28.0	social	B	3.0	chennai	87.0	97.0	91.666667	15-11-2003
5	6.0	amir	28.0	maths	D	2.0	chennai	97.0	87.0	91.666667	17-11-2003
7	8.0	ajeesh	29.0	tamil	B	2.0	cmbr	86.0	80.0	91.666667	19-11-2003
8	9.0	arit	27.0	english	E	3.0	karur	85.0	83.0	91.666667	20-11-2003
9	10.0	ajay	25.0	botany	D	3.0	chennai	89.0	98.0	91.666667	21-11-2003
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [66]:
```
df.drop_duplicates()
```

Out[66]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average	Date
0	1.0	arul	26.0	maths	A	1.0	chennai	98.0	100.0	91.666667	12-11-2003
1	2.0	anu	27.0	physics	B	2.0	tiruvallur	99.0	99.0	91.666667	13-11-2003
2	3.0	sam	29.0	chemistry	A	1.0	kanchi	97.0	98.0	91.666667	14-11-2003
3	4.0	kaaviya	28.0	social	B	3.0	chennai	87.0	97.0	91.666667	15-11-2003
5	6.0	amir	28.0	maths	D	2.0	chennai	97.0	87.0	91.666667	17-11-2003
7	8.0	ajeesh	29.0	tamil	B	2.0	cmbr	86.0	80.0	91.666667	19-11-2003
8	9.0	arit	27.0	english	E	3.0	karur	85.0	83.0	91.666667	20-11-2003
9	10.0	ajay	25.0	botany	D	3.0	chennai	89.0	98.0	91.666667	21-11-2003
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [67]:
```
df.dropna(inplace=True)
```

In [68]:
```
df
```

Out[68]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average	Date
0	1.0	arul	26.0	maths	A	1.0	chennai	98.0	100.0	91.666667	12-11-2003
1	2.0	anu	27.0	physics	B	2.0	tiruvallur	99.0	99.0	91.666667	13-11-2003
2	3.0	sam	29.0	chemistry	A	1.0	kanchi	97.0	98.0	91.666667	14-11-2003
3	4.0	kaaviya	28.0	social	B	3.0	chennai	87.0	97.0	91.666667	15-11-2003
5	6.0	amir	28.0	maths	D	2.0	chennai	97.0	87.0	91.666667	17-11-2003
7	8.0	ajeesh	29.0	tamil	B	2.0	cmbr	86.0	80.0	91.666667	19-11-2003
8	9.0	arit	27.0	english	E	3.0	karur	85.0	83.0	91.666667	20-11-2003
9	10.0	ajay	25.0	botany	D	3.0	chennai	89.0	98.0	91.666667	21-11-2003

In [69]:
```
df[['Maths','Science','Average']].corr()
```

Out[69]:

	Maths	Science	Average
Maths	1.000000	0.566497	NaN
Science	0.566497	1.000000	NaN
Average	NaN	NaN	NaN

In [70]:
```
df.drop(1,inplace=True)
```

In [71]:
```
df.drop('Date',axis=1,inplace=True)
```

In [72]:
```
df
```

Out[72]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average
0	1.0	arul	26.0	maths	A	1.0	chennai	98.0	100.0	91.666667
2	3.0	sam	29.0	chemistry	A	1.0	kanchi	97.0	98.0	91.666667
3	4.0	kaaviya	28.0	social	B	3.0	chennai	87.0	97.0	91.666667
5	6.0	amir	28.0	maths	D	2.0	chennai	97.0	87.0	91.666667
7	8.0	ajeesh	29.0	tamil	B	2.0	cmbr	86.0	80.0	91.666667
8	9.0	arit	27.0	english	E	3.0	karur	85.0	83.0	91.666667
9	10.0	ajay	25.0	botany	D	3.0	chennai	89.0	98.0	91.666667

In [73]:
```
df.drop(7,inplace=True)
```

In [74]:
```
df
```

Out[74]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average
0	1.0	arul	26.0	maths	A	1.0	chennai	98.0	100.0	91.666667
2	3.0	sam	29.0	chemistry	A	1.0	kanchi	97.0	98.0	91.666667
3	4.0	kaaviya	28.0	social	B	3.0	chennai	87.0	97.0	91.666667
5	6.0	amir	28.0	maths	D	2.0	chennai	97.0	87.0	91.666667
8	9.0	arit	27.0	english	E	3.0	karur	85.0	83.0	91.666667
9	10.0	ajay	25.0	botany	D	3.0	chennai	89.0	98.0	91.666667

In [75]:
```
a=pd.DataFrame({'Average':5,'Rank':1.0},index=[4])
```

In [76]:
```
df=pd.concat([df,a])
```

In [77]:
```
df
```

Out[77]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average
0	1.0	arul	26.0	maths	A	1.0	chennai	98.0	100.0	91.666667
2	3.0	sam	29.0	chemistry	A	1.0	kanchi	97.0	98.0	91.666667
3	4.0	kaaviya	28.0	social	B	3.0	chennai	87.0	97.0	91.666667
5	6.0	amir	28.0	maths	D	2.0	chennai	97.0	87.0	91.666667
8	9.0	arit	27.0	english	E	3.0	karur	85.0	83.0	91.666667
9	10.0	ajay	25.0	botany	D	3.0	chennai	89.0	98.0	91.666667
4	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	5.000000

In [78]:
```
1 df[0:9:2]
```

Out[78]:

	ID	Name	Age	Centums	Section	Rank	City	Maths	Science	Average
0	1.0	arul	26.0	maths	A	1.0	chennai	98.0	100.0	91.666667
3	4.0	kaaviya	28.0	social	B	3.0	chennai	87.0	97.0	91.666667
8	9.0	arit	27.0	english	E	3.0	karur	85.0	83.0	91.666667
4	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	5.000000

In [79]:
```
1 df[['City']][0:2]
```

Out[79]:

	City
0	chennai
2	kanchi

In [ ]:
```
1
```

```
EX 5 EXPLORATORY DATA ANALYSIS
REG NO:230701027
STUDENT NAME:AMIRTHAVARSHINI R U
DATE:03/09/2024
```

In [12]:
```
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline #especially for jupiter notebook
```

```
UsageError: unrecognized arguments: #especially for jupiter notebook
```

In [13]:
```
tips=sns.load_dataset('tips')
```

In [14]:
```
tips.head
```

Out[14]:
```
<bound method NDFrame.head of      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 x 7 columns]>
```

In [15]:
```
tips.head()
```

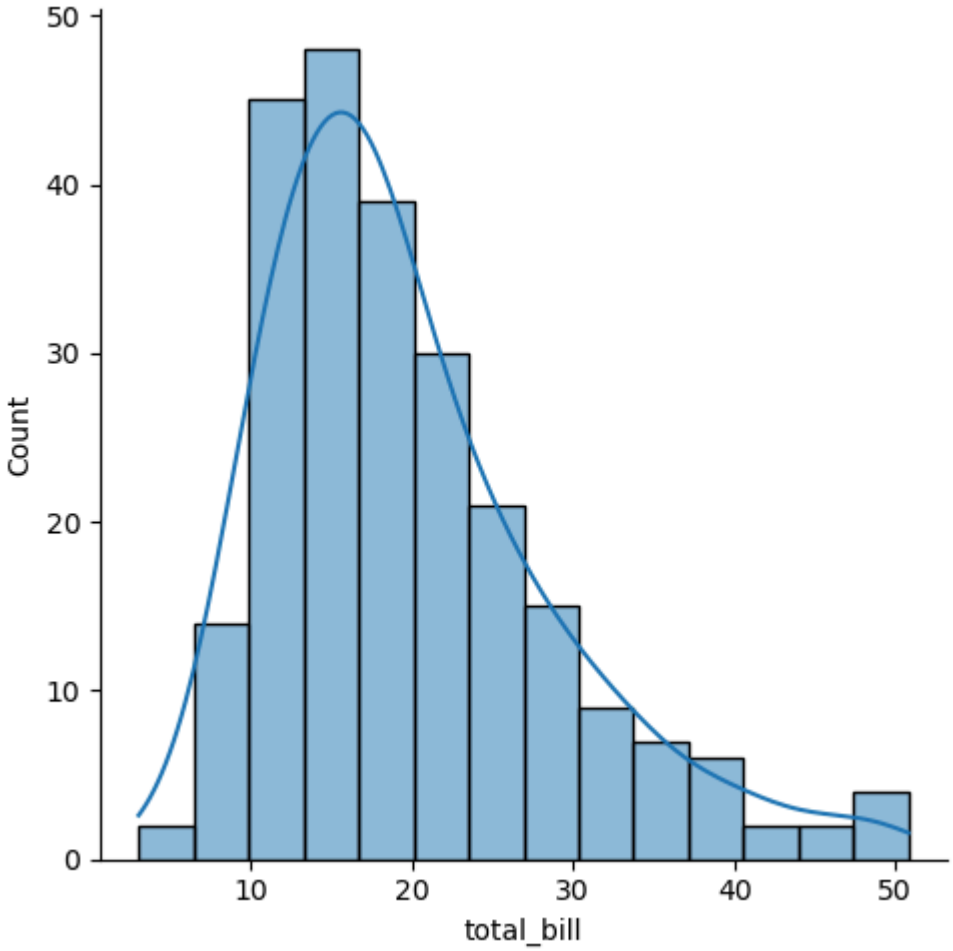
Out[15]:

	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 [16]:
```
1 sns.displot(tips.total_bill,kde=True)
```

```
C:\Users\amirt_erk6tkl\anaconda3\hi\Lib\site-packages\seaborn\axisgrid.p
y:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

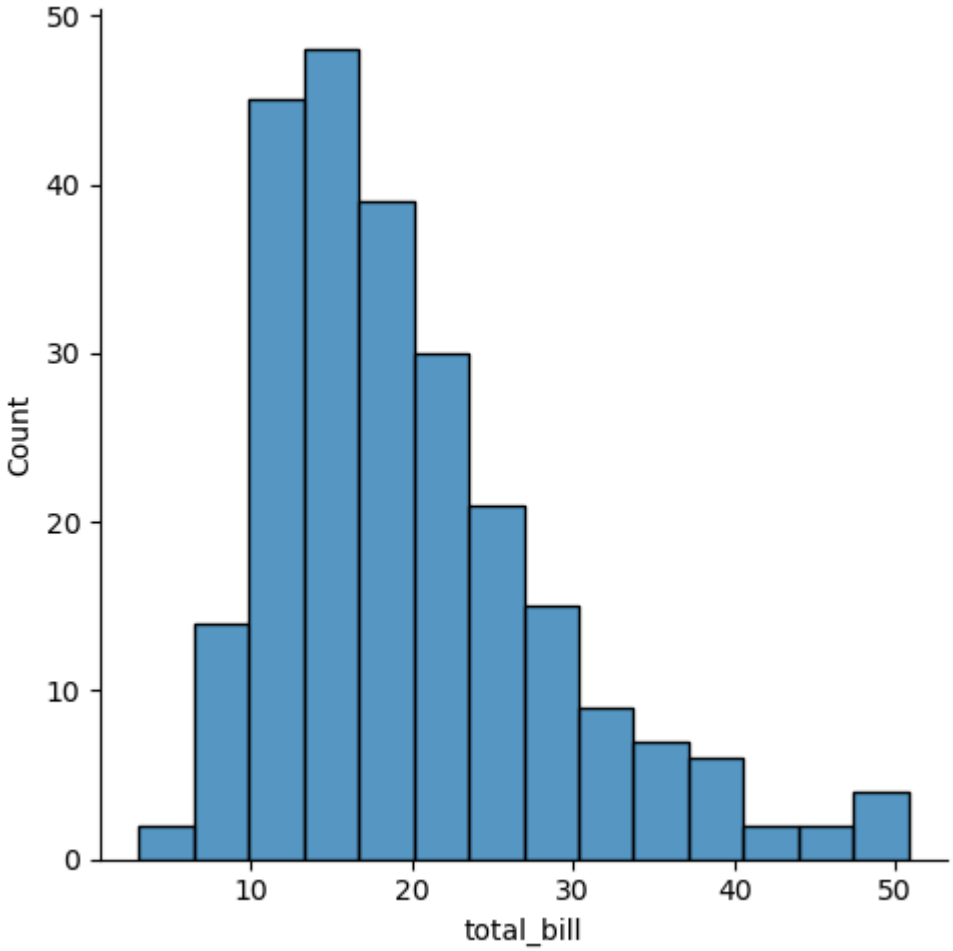
Out[16]:
```
<seaborn.axisgrid.FacetGrid at 0x14ac9e0d250>
```



In [17]:
```
1 sns.displot(tips.total_bill,kde=False)
```

```
C:\Users\amirt_erk6tkl\anaconda3\hi\Lib\site-packages\seaborn\axisgrid.p
y:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

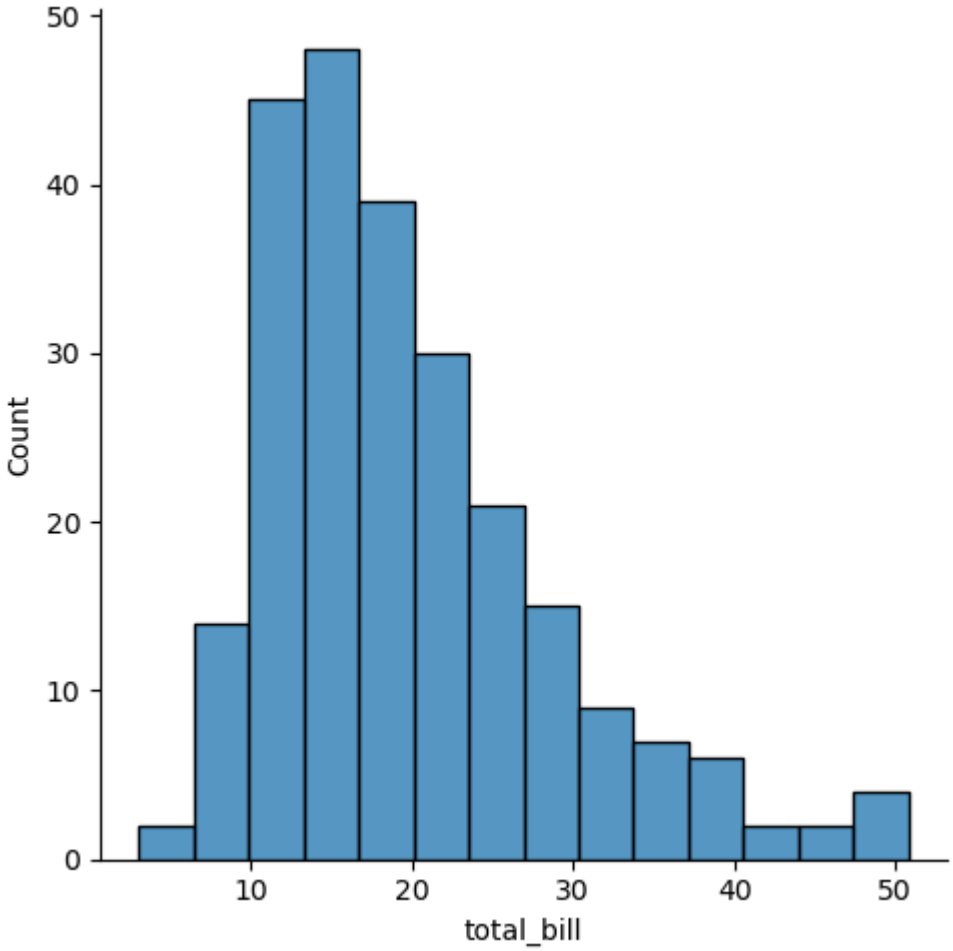
Out[17]:
```
<seaborn.axisgrid.FacetGrid at 0x14ac95d0f50>
```



In [18]:
```
1 sns.displot(tips.total_bill) #default kde=False ,where kde stands for
```

```
C:\Users\amirt_erk6tkl\anaconda3\hi\Lib\site-packages\seaborn\axisgrid.p
y:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[18]:
```
<seaborn.axisgrid.FacetGrid at 0x14acd0c4c10>
```

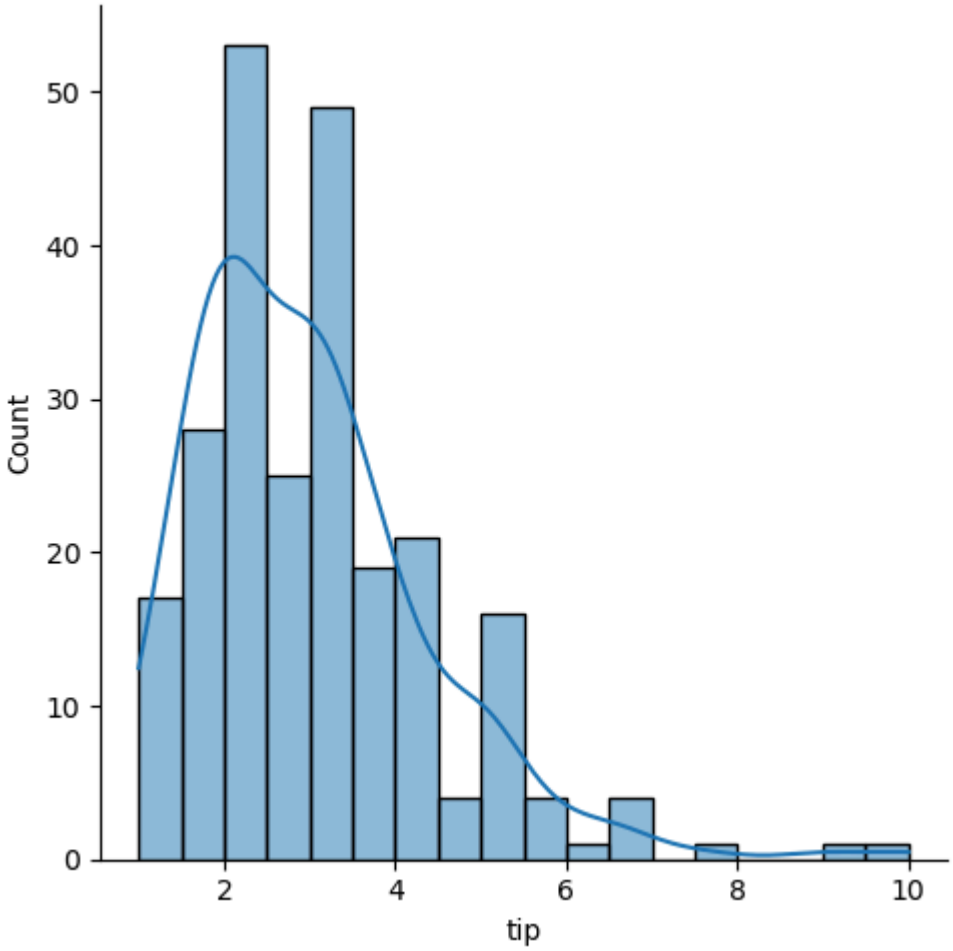


The total_bill is left skewed

In [19]:
```
1 sns.displot(tips.tip,kde=True)
```

```
C:\Users\amirt_erk6tkl\anaconda3\hi\Lib\site-packages\seaborn\axisgrid.p
y:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

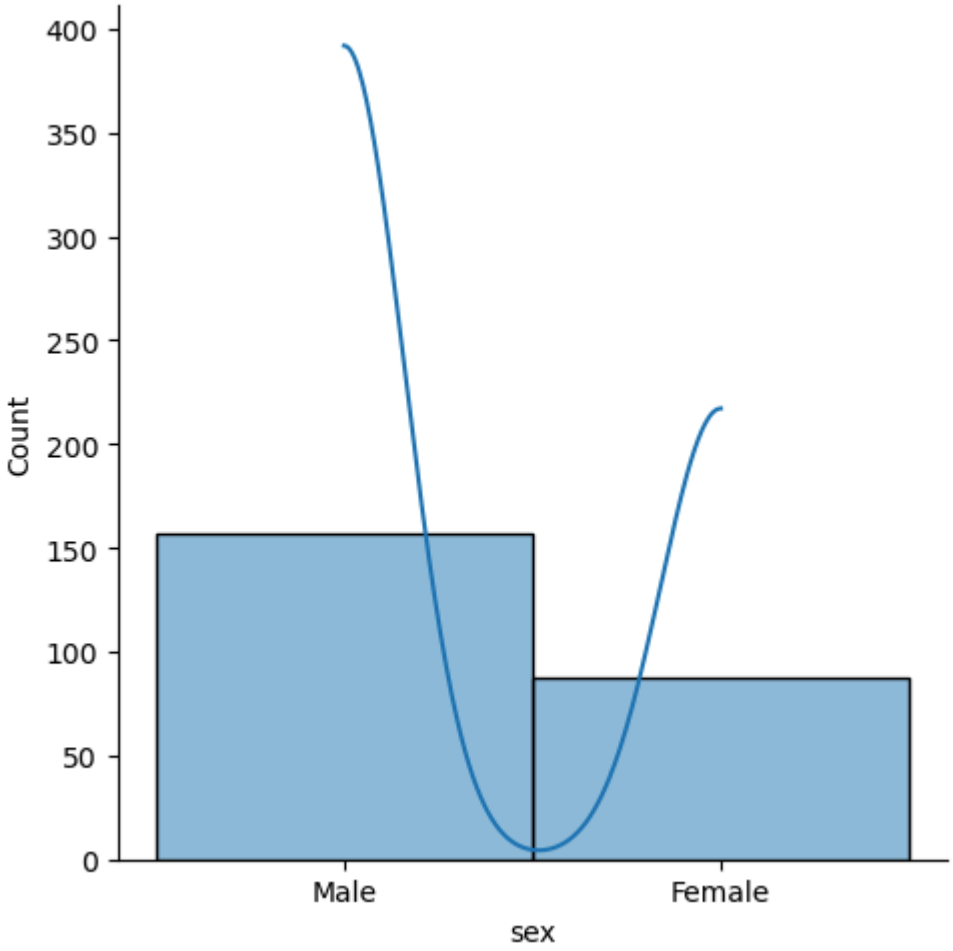
Out[19]:
```
<seaborn.axisgrid.FacetGrid at 0x14ac9f9b5d0>
```



In [20]:
```
1 sns.displot(tips.sex,kde=True)
```

```
C:\Users\amirt_erk6tkl\anaconda3\hi\Lib\site-packages\seaborn\axisgrid.p
y:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

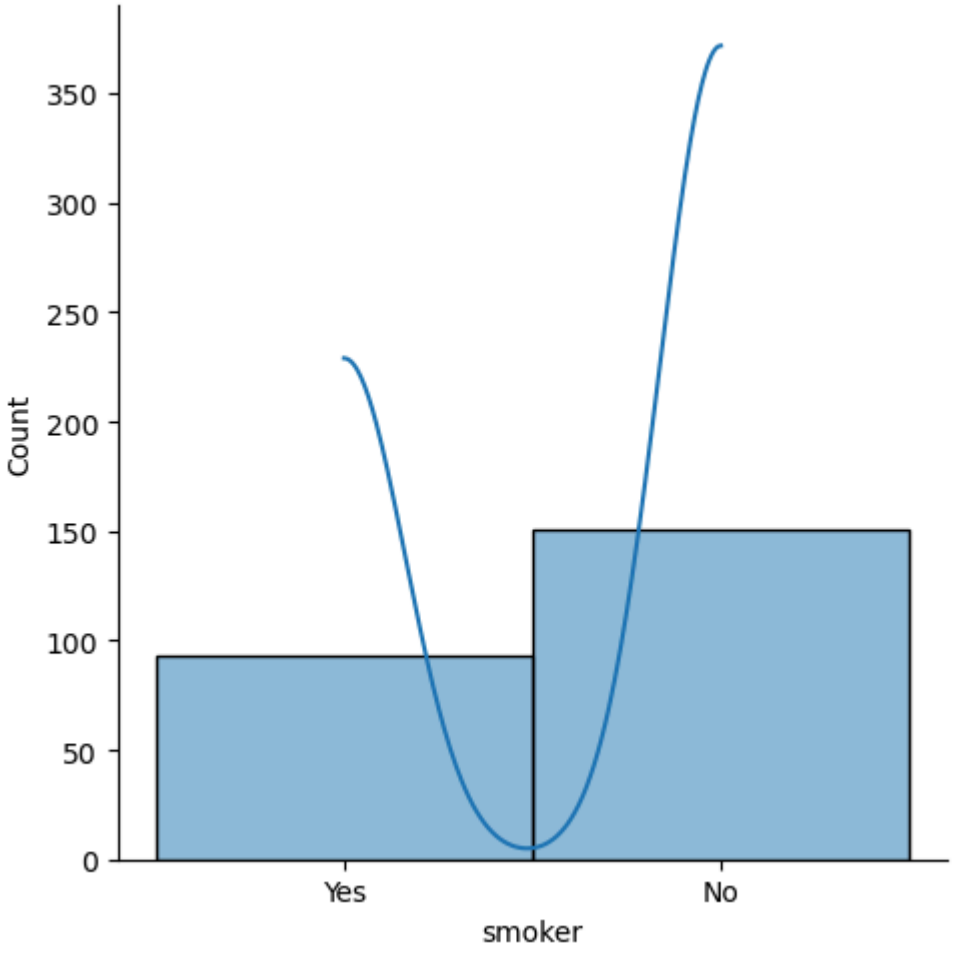
Out[20]: <seaborn.axisgrid.FacetGrid at 0x14acd122690>



In [21]:
```
1 sns.displot(tips.smoker,kde=True)
```

```
C:\Users\amirt_erk6tkl\anaconda3\hi\Lib\site-packages\seaborn\axisgrid.p
y:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

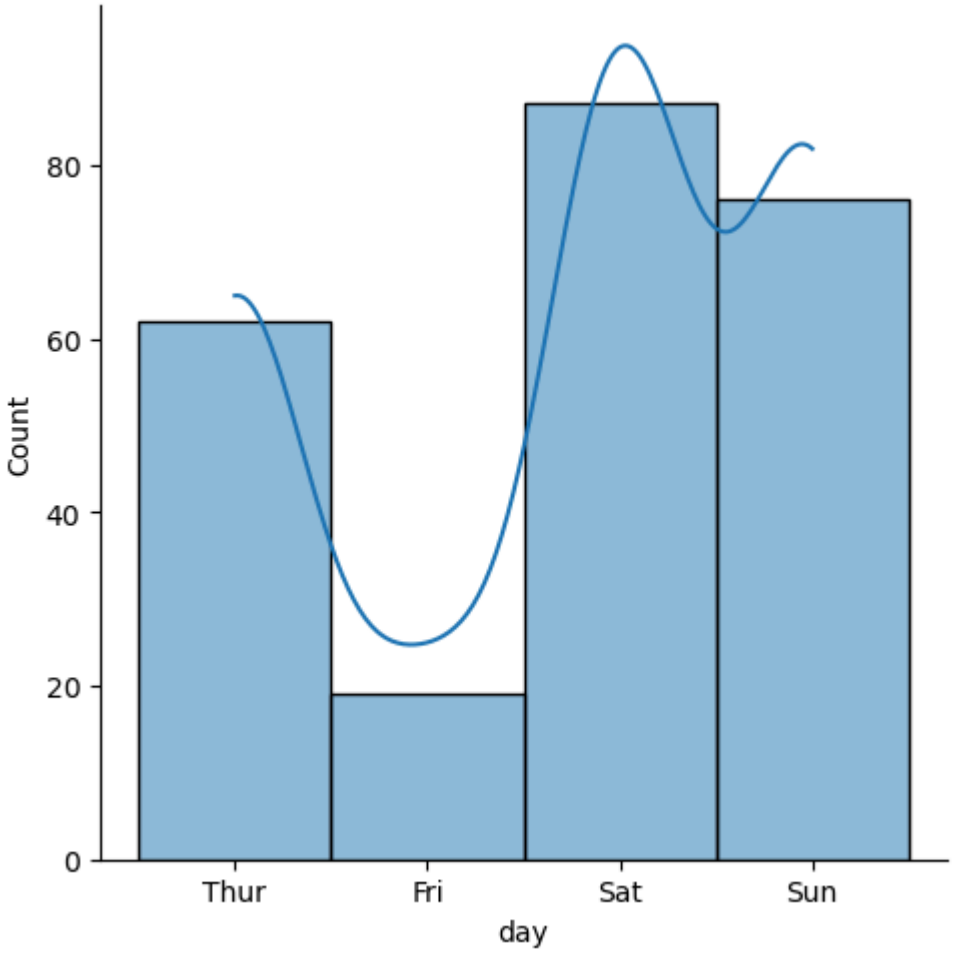
Out[21]: <seaborn.axisgrid.FacetGrid at 0x14ac9d97e10>



In [22]:
```
1 sns.displot(tips.day,kde=True)
```

```
C:\Users\amirt_erk6tkl\anaconda3\hi\Lib\site-packages\seaborn\axisgrid.p
y:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

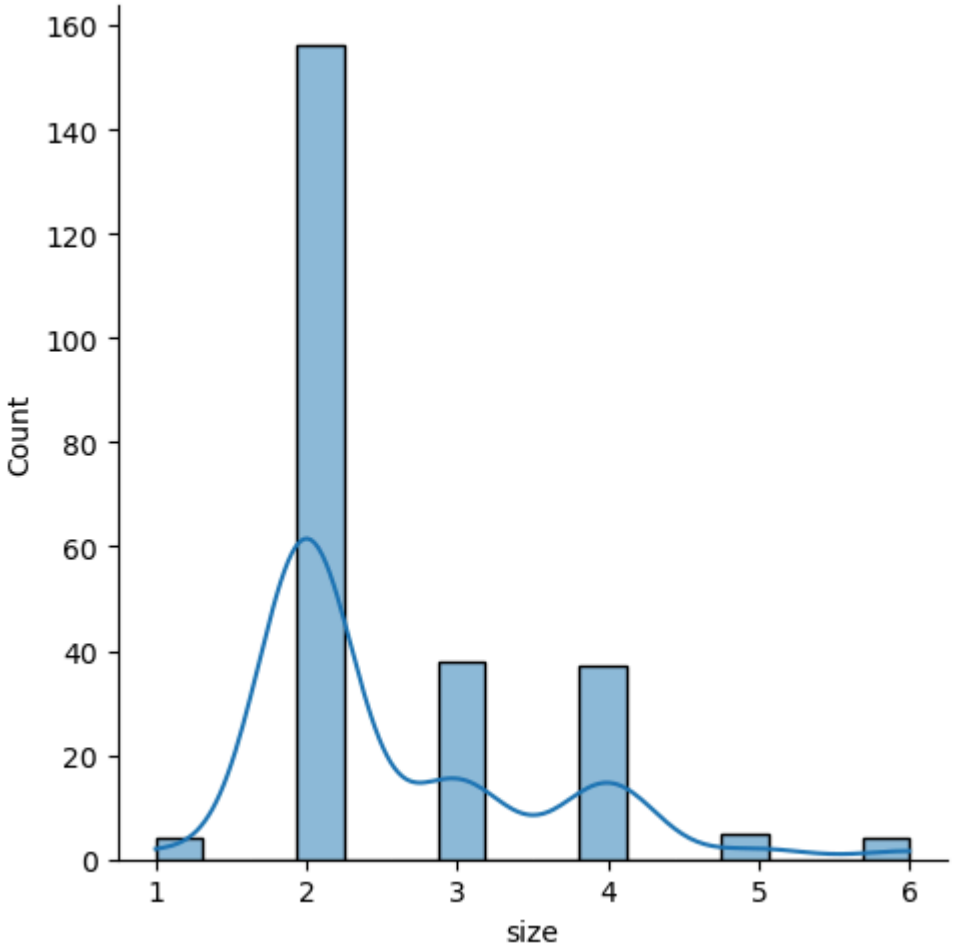
Out[22]: `<seaborn.axisgrid.FacetGrid at 0x14ac9e75790>`



In [23]:
```
1 sns.displot(tips['size'],kde=True)
```

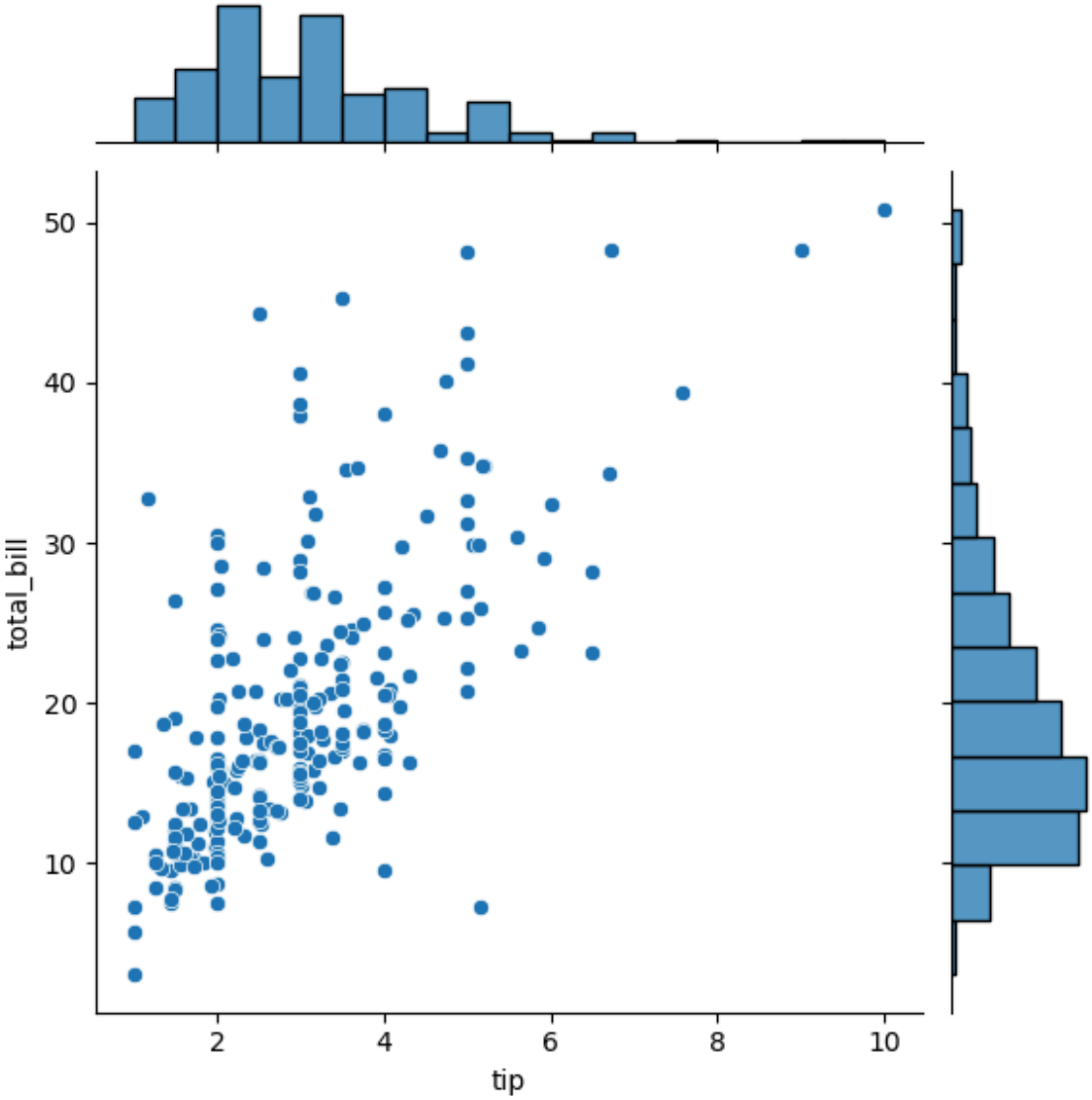
```
C:\Users\amirt_erk6tkl\anaconda3\hi\Lib\site-packages\seaborn\axisgrid.p
y:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[23]: <seaborn.axisgrid.FacetGrid at 0x14ac9db1e90>



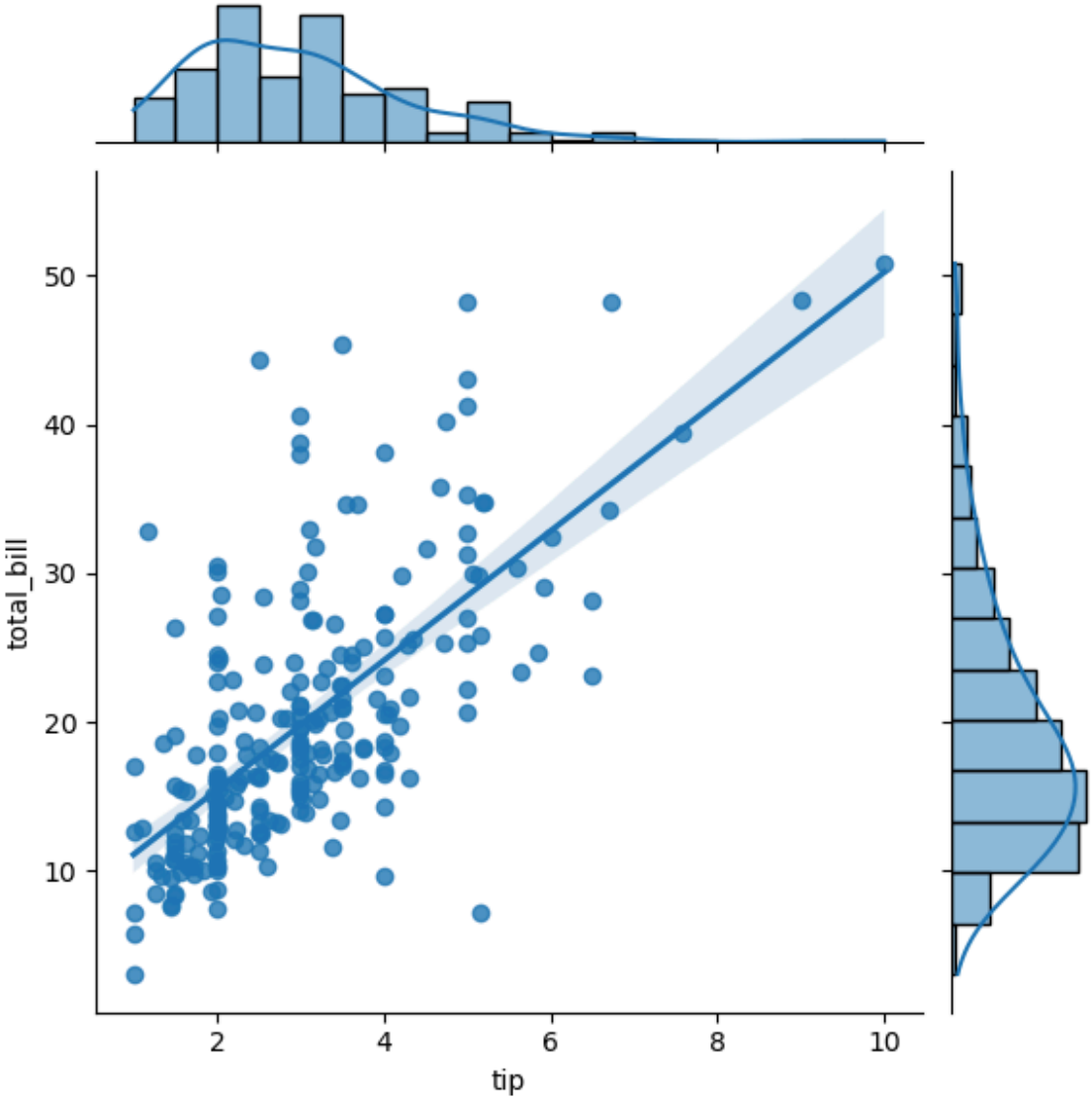
In [24]:
```
1 sns.jointplot(x=tips.tip,y=tips.total_bill) #does not have kde argumen
```

Out[24]: <seaborn.axisgrid.JointGrid at 0x14ace2c5a90>



In [25]:
```
1 sns.jointplot(x=tips.tip,y=tips.total_bill,kind='reg')   # here "reg"
```

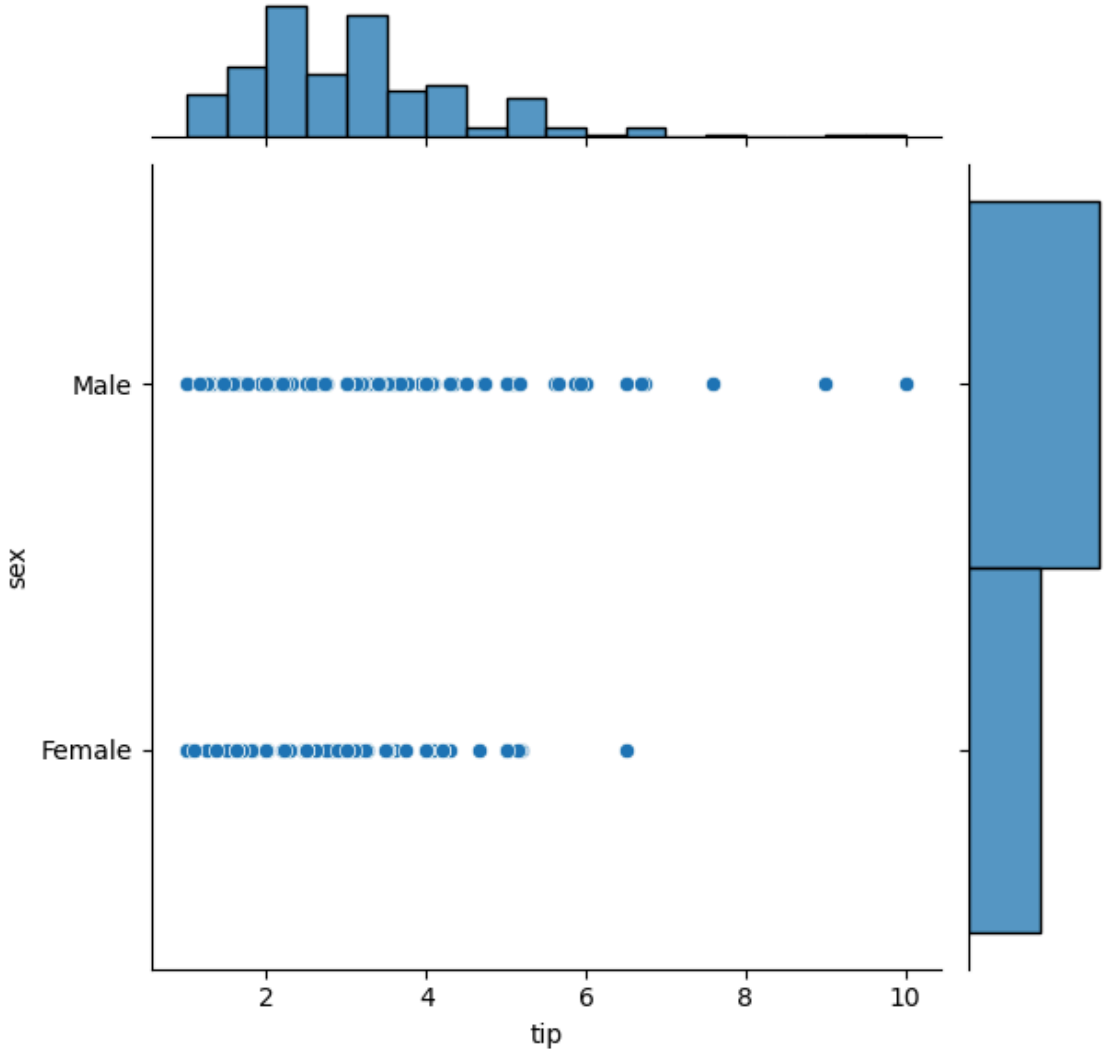
Out[25]: <seaborn.axisgrid.JointGrid at 0x14ace45aad0>



This jointplot shows the distribution

In [26]:
```
1 sns.jointplot(x=tips.tip,y=tips.sex)
```

Out[26]: <seaborn.axisgrid.JointGrid at 0x14ace9e7dd0>



In [27]:
```
1 sns.jointplot(x=tips.tip,y=tips.sex,kind='reg') #jointplot is only for
```

```
---------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[27], line 1
----> 1 sns.jointplot(x=tips.tip,y=tips.sex,kind='reg')

File ~\anaconda3\hi\Lib\site-packages\seaborn\axisgrid.py:2324, in joi
ntplot(data, x, y, hue, kind, height, ratio, space, dropna, xlim, yli
m, color, palette, hue_order, hue_norm, marginal_ticks, joint_kws, mar
ginal_kws, **kwargs)
   2321     grid.plot_marginals(histplot, **marginal_kws)
   2323     joint_kws.setdefault("color", color)
-> 2324     grid.plot_joint(regplot, **joint_kws)
   2326 elif kind.startswith("resid"):
   2328     joint_kws.setdefault("color", color)

File ~\anaconda3\hi\Lib\site-packages\seaborn\axisgrid.py:1826, in Joi
ntGrid.plot_joint(self, func, **kwargs)
```

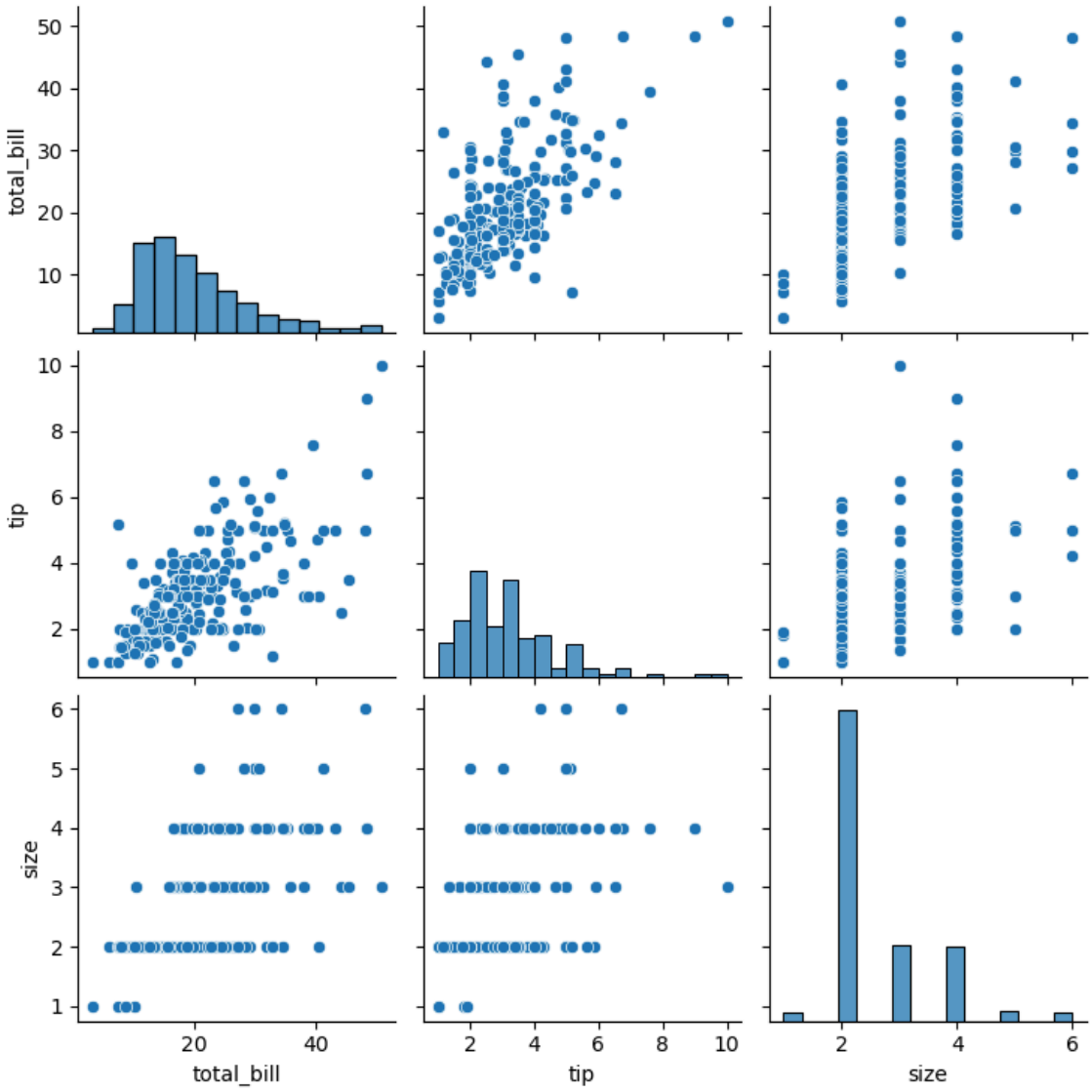
In [ ]:
```
1 sns.jointplot(x=tips.tip,y=tips['size'],kind='reg')
```

In [ ]:
```
1 sns.jointplot(x=tips.tip,y=tips.total_bill,kind='hex')
```

In [28]:
```
1 sns.pairplot(tips) #does not give plot for categorical data of type st
```

```
C:\Users\amirt_erk6tkl\anaconda3\hi\Lib\site-packages\seaborn\axisgrid.p
y:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[28]: <seaborn.axisgrid.PairGrid at 0x14ad0d784d0>



In [29]:
```
1 tips.time.value_counts()
```

Out[29]:
```
time
Dinner    176
Lunch      68
Name: count, dtype: int64
```

In [30]:
```
1 tips.sex.value_counts()
```

Out[30]:
```
sex
Male      157
Female     87
Name: count, dtype: int64
```

In [31]:
```
1 tips.total_bill.value_counts()
```

Out[31]:
```
total_bill
13.42    3
13.81    2
15.98    2
17.92    2
10.07    2
        ..
24.71    1
21.16    1
28.97    1
22.49    1
18.78    1
Name: count, Length: 229, dtype: int64
```

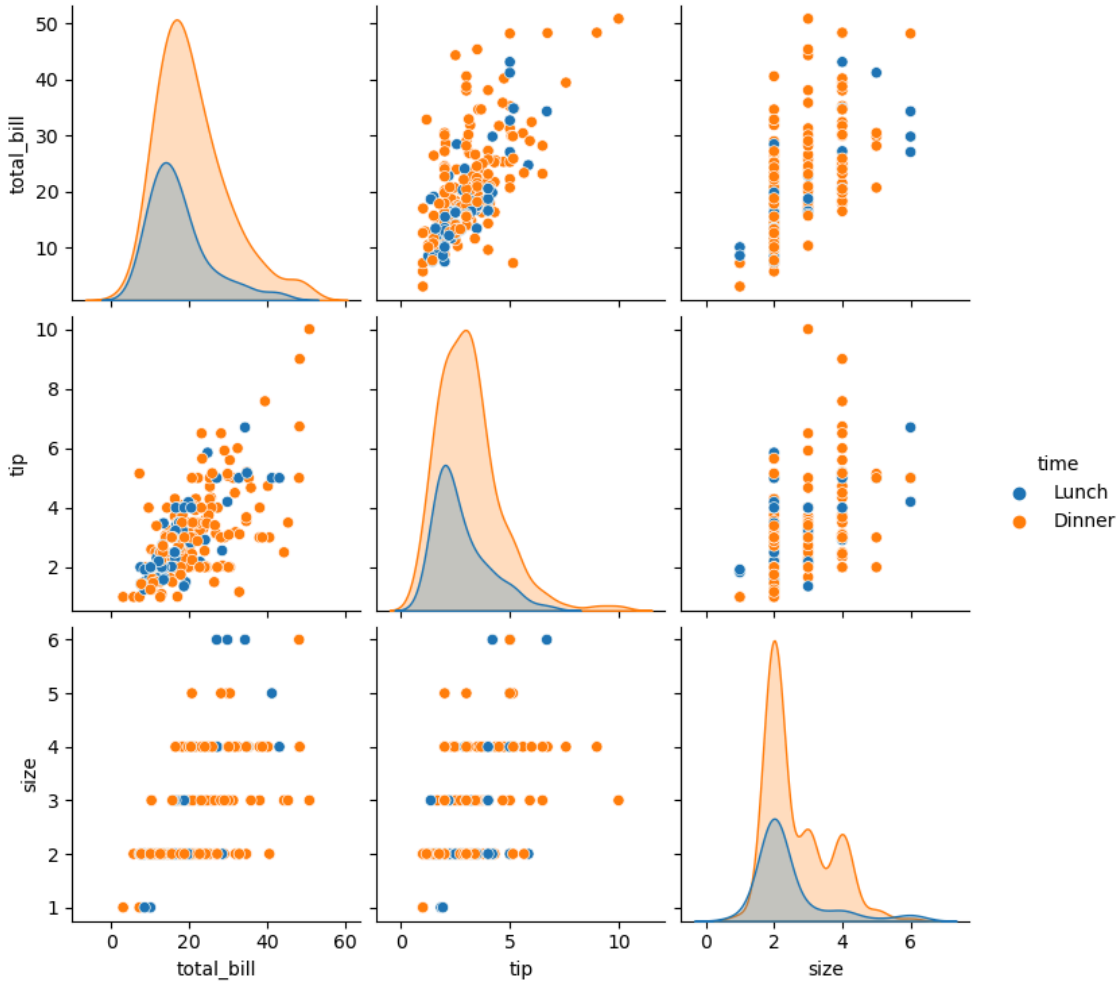
In [32]:
```
1 tips.tip.value_counts()
```

Out[32]:
```
tip
2.00    33
3.00    23
4.00    12
5.00    10
2.50    10
        ..
4.34     1
1.56     1
5.20     1
2.60     1
1.75     1
Name: count, Length: 123, dtype: int64
```

In [33]:
```
1 sns.pairplot(tips,hue='time') # This is good for ploting all numerical
```

```
C:\Users\amirt_erk6tkl\anaconda3\hi\Lib\site-packages\seaborn\axisgrid.p
y:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

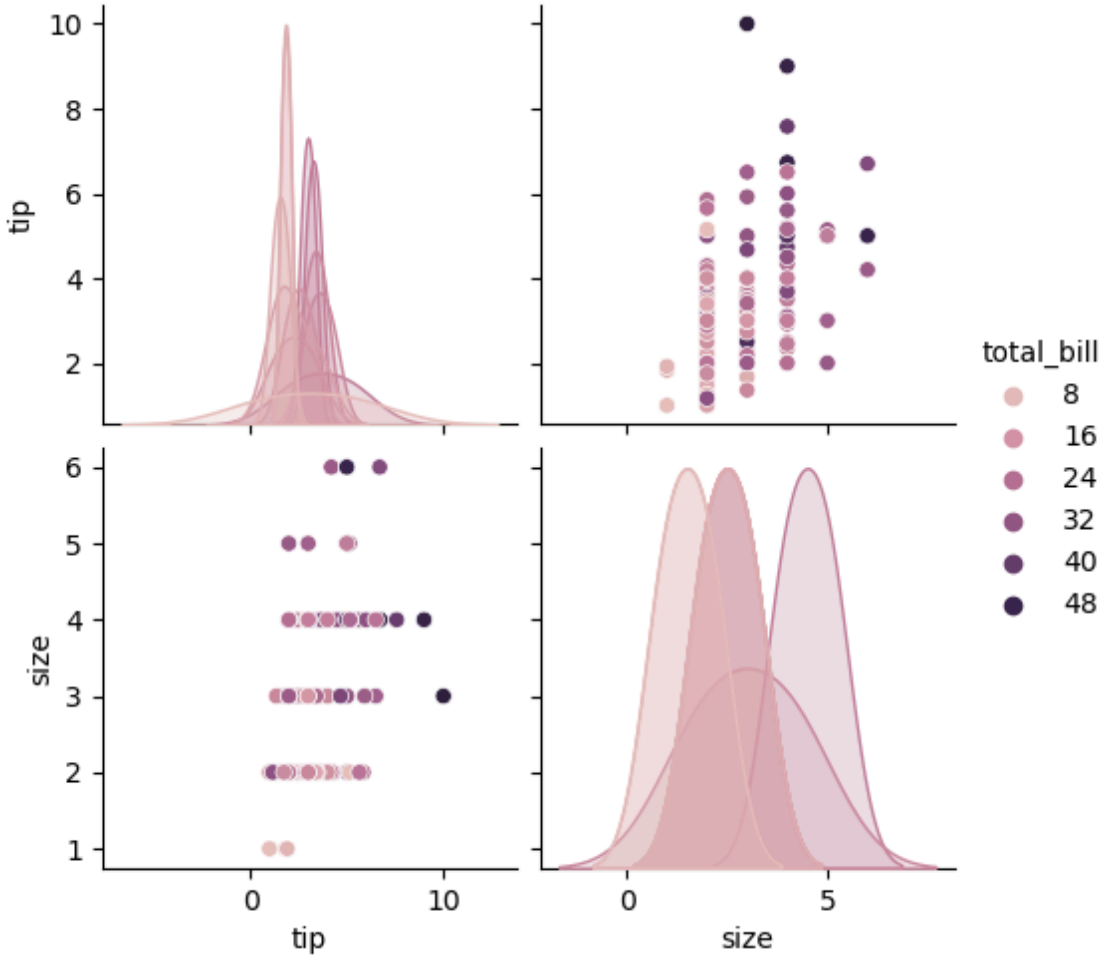
Out[33]: <seaborn.axisgrid.PairGrid at 0x14ac9e84190>



In [34]:
```
1 sns.pairplot(tips,hue='total_bill')
```

```
C:\Users\amirt_erk6tkl\anaconda3\hi\Lib\site-packages\seaborn\axisgrid.p
y:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

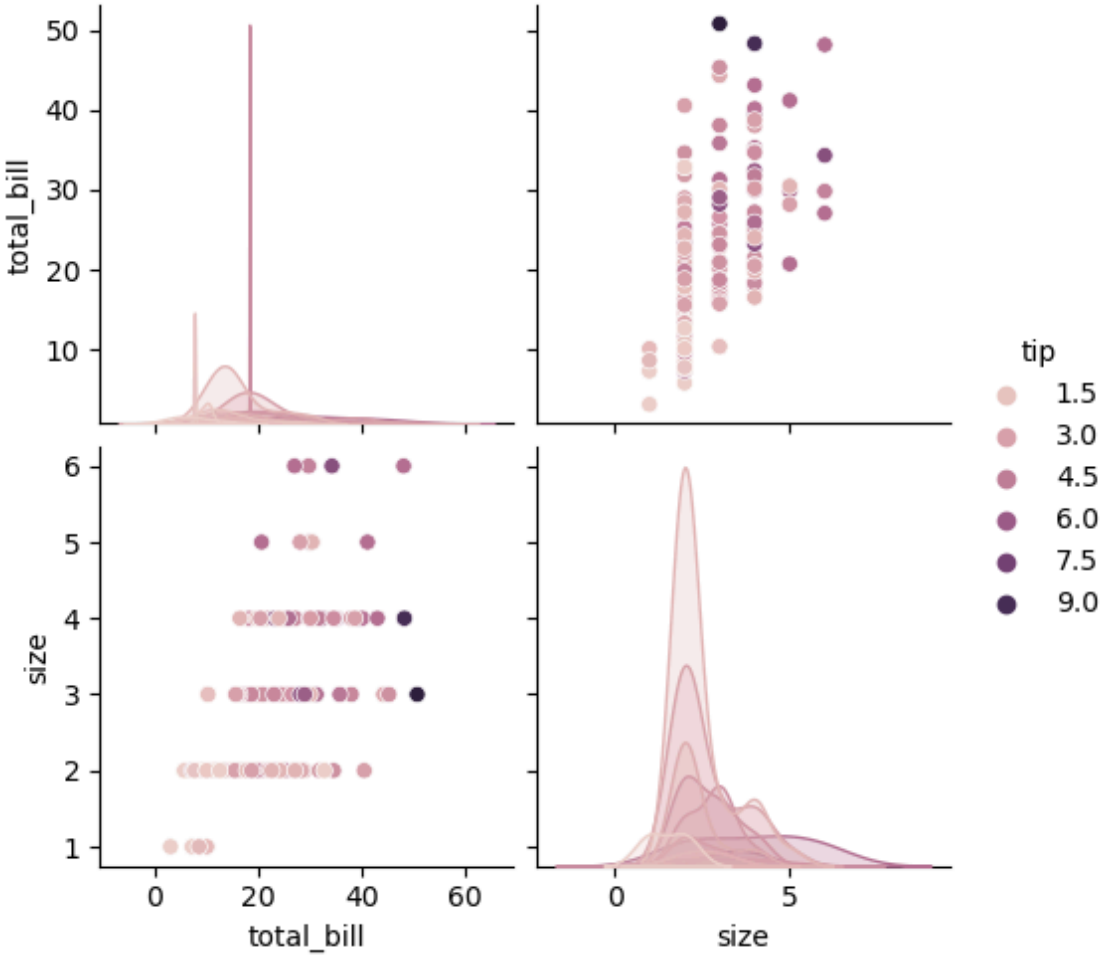
Out[34]: <seaborn.axisgrid.PairGrid at 0x14ad1e17150>



In [35]:
```
1 sns.pairplot(tips,hue='tip')
```

```
C:\Users\amirt_erk6tkl\anaconda3\hi\Lib\site-packages\seaborn\axisgrid.p
y:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

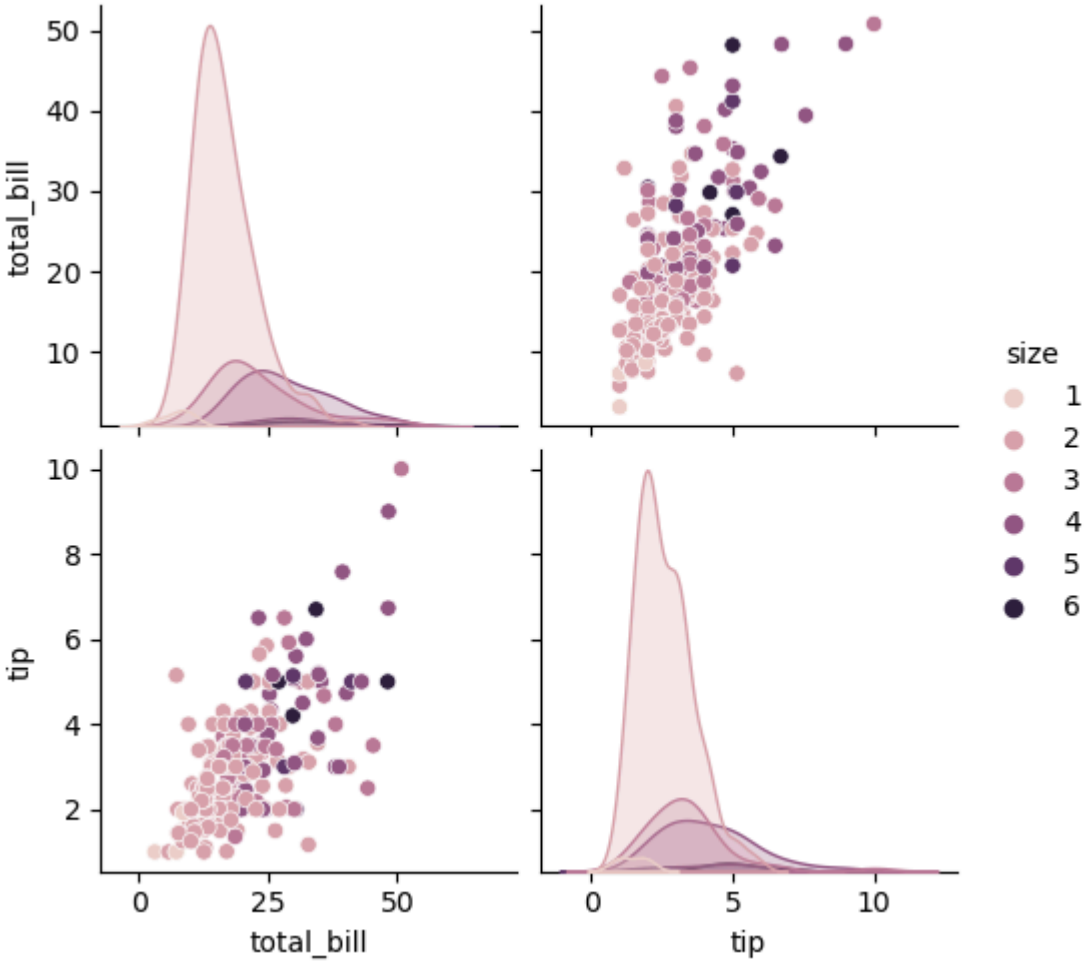
Out[35]: <seaborn.axisgrid.PairGrid at 0x14ad23d1c10>



In [36]:
```
1 sns.pairplot(tips,hue='size')
```

```
C:\Users\amirt_erk6tkl\anaconda3\hi\Lib\site-packages\seaborn\axisgrid.p
y:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

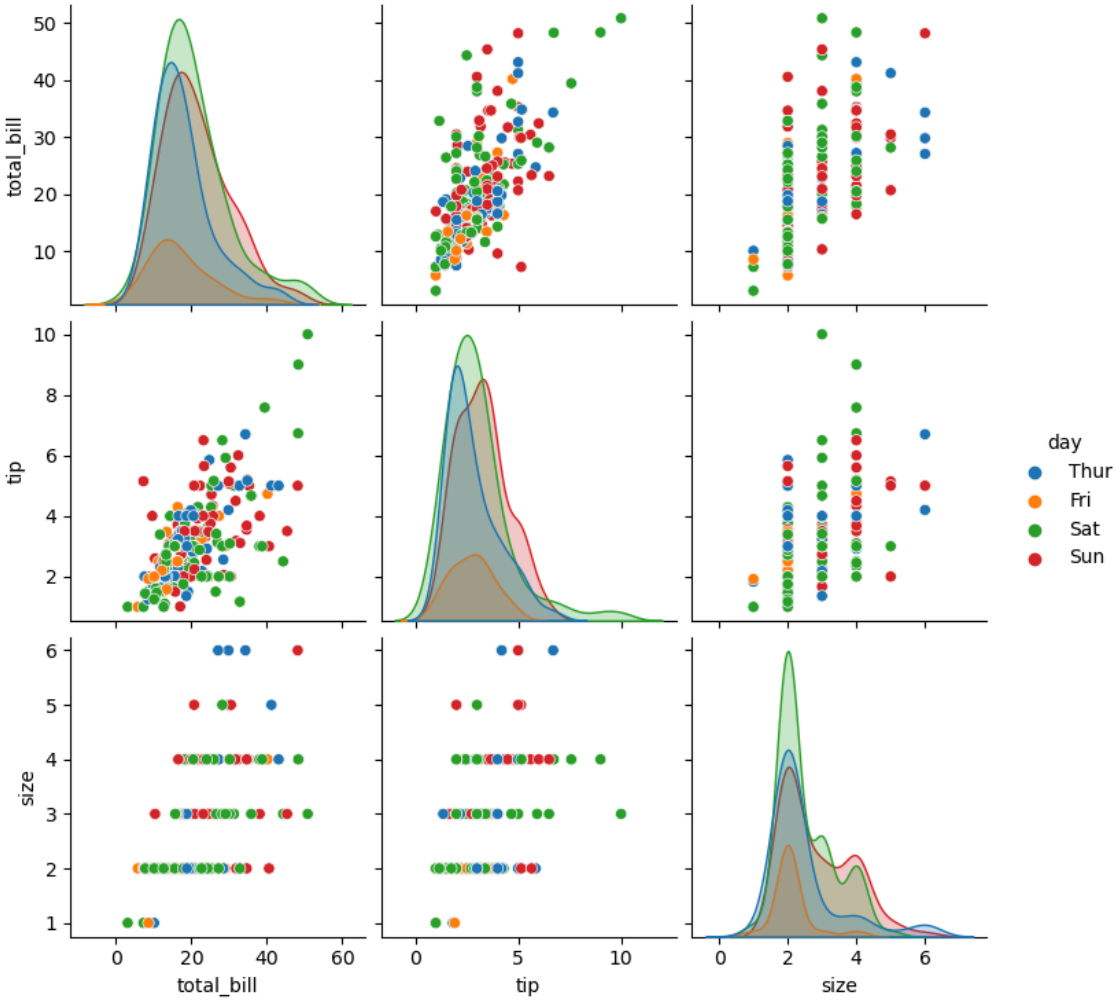
Out[36]: <seaborn.axisgrid.PairGrid at 0x14ad2ccf150>



In [37]:
```
1 sns.pairplot(tips,hue='day')
```

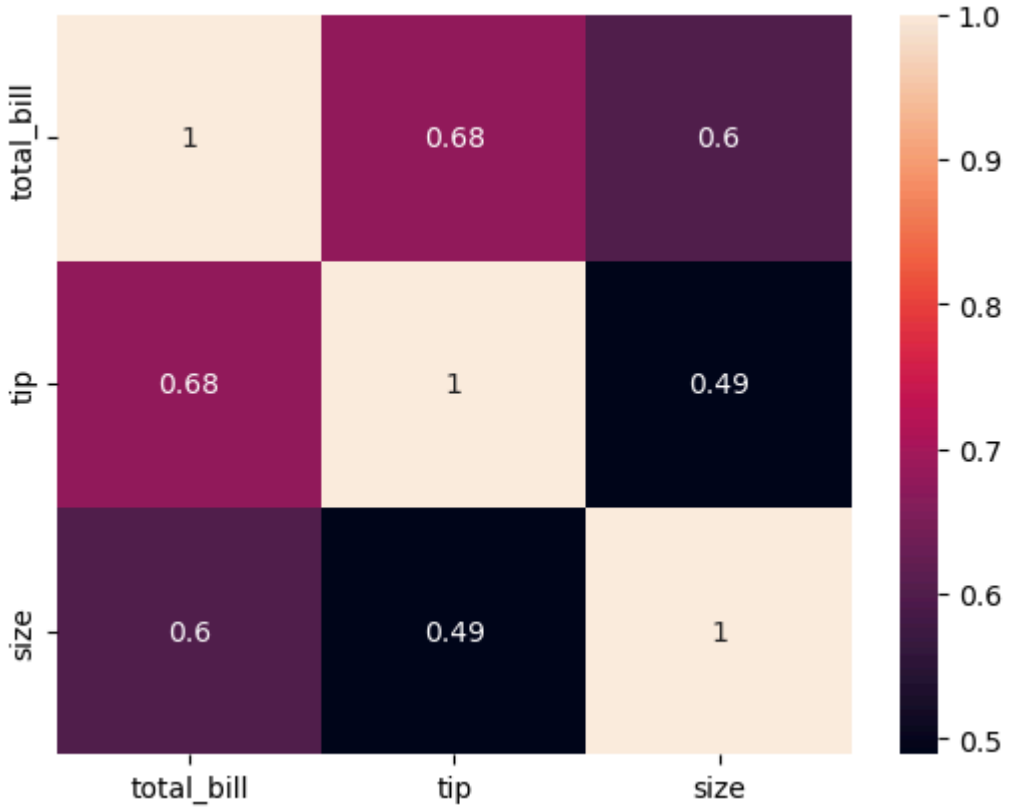
```
C:\Users\amirt_erk6tkl\anaconda3\hi\Lib\site-packages\seaborn\axisgrid.p
y:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[37]: <seaborn.axisgrid.PairGrid at 0x14ad31a1c10>



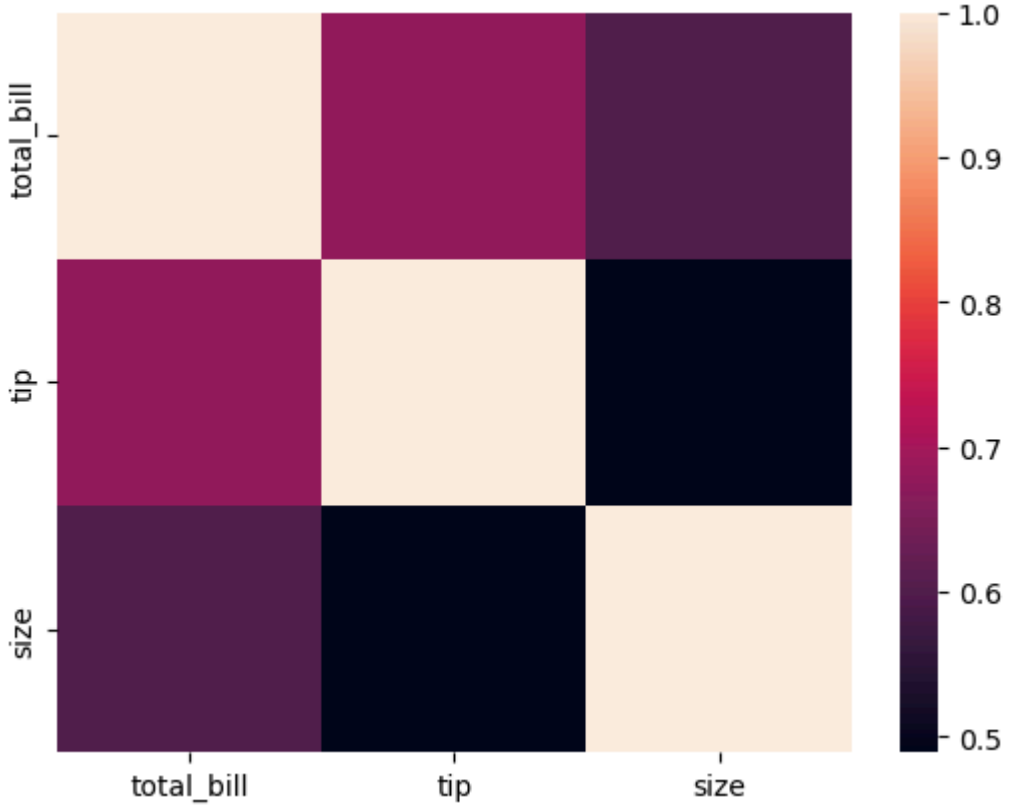
In [38]:
```
1 sns.heatmap(tips.corr(numeric_only=True),annot=True)
```

Out[38]: <Axes: >



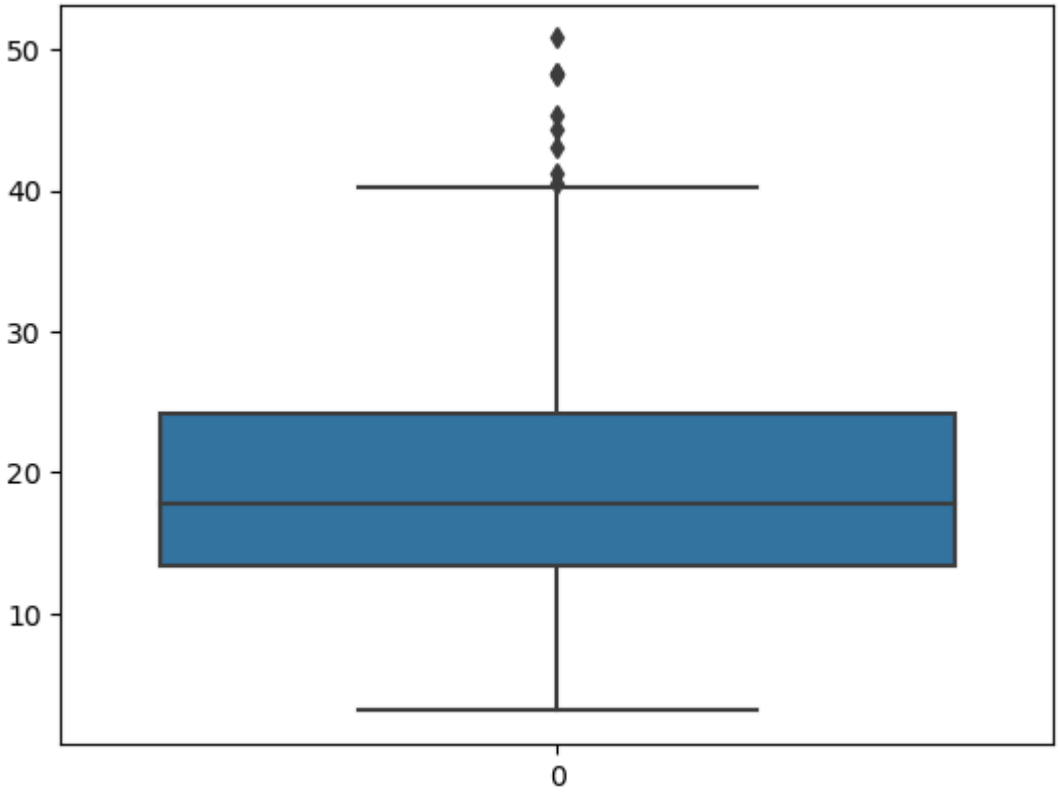
In [41]:
```
1 sns.heatmap(tips.corr(numeric_only=True),annot=False)
```

Out[41]: <Axes: >



In [43]:
```
1 sns.boxplot(tips.total_bill)
```

Out[43]: <Axes: >



In [ ]:
```
1
```

```
EX 6 FEATURE EXTRACTION
REG NO:230701027
STUDENT NAME:AMIRTHAVARSHINI R U
DATE:10/09/2024
```

In [*]:
```
import string
import nltk
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer, TfidfTran
from sklearn.naive_bayes import MultinomialNB
```

In [*]:
```
nltk.download('stopwords')
```

In [*]:
```
def textPreprocessing(data):
    if not isinstance(data, str):
        return ""
    remove_pun = [c for c in data if c not in string.punctuation]
    sentences = ''.join(remove_pun)
    words = sentences.split()
    return ' '.join(words)
```

In [*]:
```
file_path = r"spam.csv"
df = pd.read_csv(file_path, sep='\t', names=['label', 'message'], enco
df['message'] = df['message'].astype(str)
```

In [*]:
```
wordVector = CountVectorizer(analyzer=textPreprocessing)
finalWordVector = wordVector.fit(df['message'])
print(finalWordVector.vocabulary_)
bow = finalWordVector.transform(df['message'])

print(bow)
```

In [*]:
```
tfidfObject = TfidfTransformer().fit(bow)
final_feature = tfidfObject.transform(bow)
```

In [*]:
```
model = MultinomialNB()
model.fit(final_feature, df['label'])
```

In [*]:
```
score = model.score(final_feature, df['label'])
print("Model Accuracy: ", score)
```

In [*]:
```
inputSMS = input("Enter the SMS Content: ")
preprocessText = textPreprocessing(inputSMS)
```

In [*]:
```
vector = finalWordVector.transform([preprocessText])
finalFeature = tfidfObject.transform(vector)
```

In [*]:
```
pred = model.predict(finalFeature)[0]
print("Given SMS is", pred)
```

```
EX 7 RANDOM SAMPLING
REG NO:230701027
STUDENT NAME:AMIRTHAVARSHINI R U
DATE:10/09/2024
```

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt

population_mean = 50
population_std = 10
population_size = 100000
population = np.random.normal(population_mean, population_std, populat
```

In [3]:
```
sample_sizes = [30, 50, 100]
num_samples = 1000

sample_means = {}
```

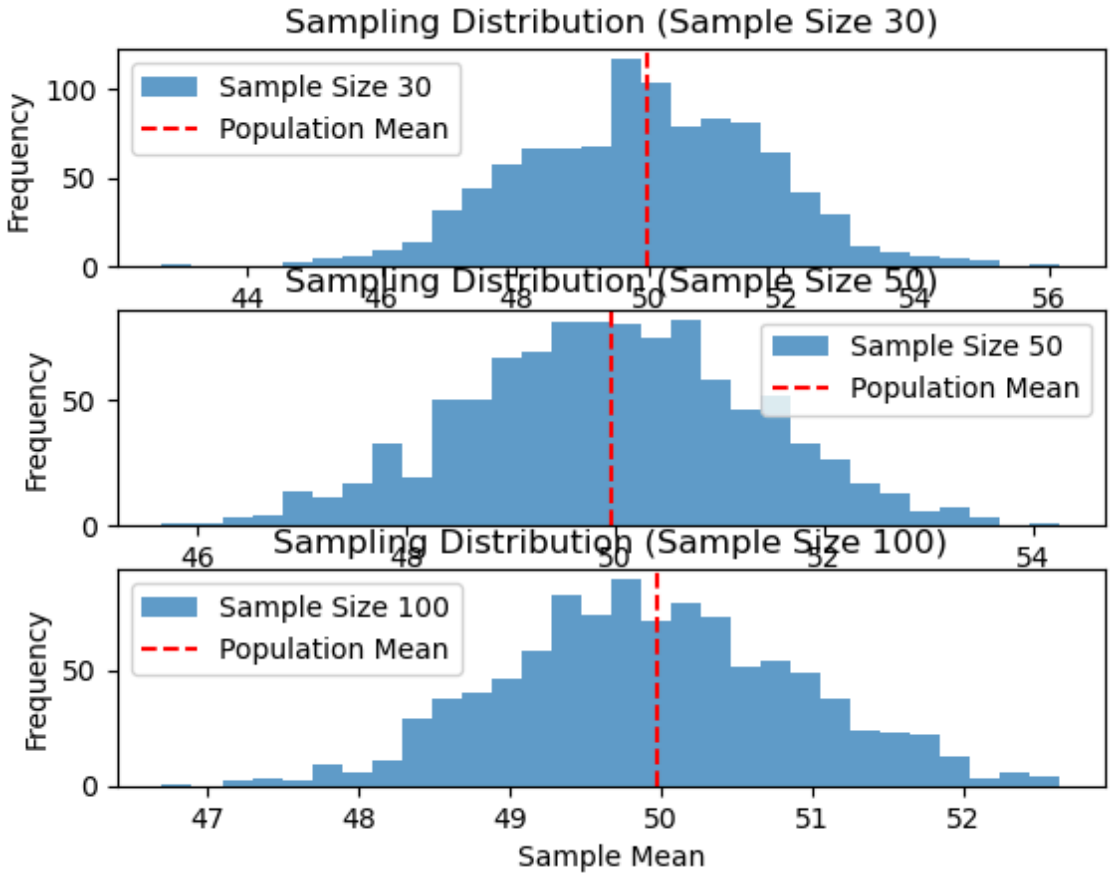
In [4]:
```
for size in sample_sizes:
    sample_means[size] = []
    for _ in range(num_samples):
        sample = np.random.choice(population, size=size, replace=False
        sample_means[size].append(np.mean(sample))
```

In [5]:
```
plt.figure(figsize=(12, 8))
```

Out[5]:
```
<Figure size 1200x800 with 0 Axes>
```


<Figure size 1200x800 with 0 Axes>

In [6]:
```
for i, size in enumerate(sample_sizes):
    plt.subplot(len(sample_sizes), 1, i+1)
    plt.hist(sample_means[size], bins=30, alpha=0.7, label=f'Sample Si
    plt.axvline(np.mean(population), color='red', linestyle='dashed',
    plt.title(f'Sampling Distribution (Sample Size {size})')
    plt.xlabel('Sample Mean')
    plt.ylabel('Frequency')
    plt.legend()


```



In [7]:
```
plt.tight_layout()
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```

In [ ]:
```

```

```
EX 8 Z TEST
REG NO:230701027
STUDENT NAME:AMIRTHAVARSHINI R U
DATE:08/10/2024
```

In [1]:
```
import math
import numpy as np
from statsmodels.stats.weightstats import ztest
from scipy.stats import norm

sample_marks = [650,730,510,670,480,800,690,530,590,620,710,670,640,78
```

In [2]:
```
sample_mean = np.mean(sample_marks)
sample_size = np.count_nonzero(sample_marks)
population_mean = 600
population_std = 100
alpha = 0.05
```

In [3]:
```
z_score = (sample_mean-population_mean)/(population_std/math.sqrt(samp
critical_value = 1.645 # from z table
```

In [4]:
```
if(z_score<critical_value):
    print('H0 is accepted!')
else:
    print('H0 is rejected. \nH1 is accepted!')
```

```
Null hypothesis is rejected.
Alternate hypothesis is accepted!
```

In [5]:
```
ztest_score, pval = ztest(sample_marks,value=population_mean,alternati
print('Z-test Score:',ztest_score,'\nP-value:',pval)
if(pval>alpha):
     print('Null hypothesis is accepted!')
else:
    print('Null hypothesis is rejected. \nAlternate hypothesis is acce
```

```
Z-test Score: 1.831744911595958
P-value: 0.03349471703839336
Null hypothesis is rejected.
Alternate hypothesis is accepted!
```

In [6]:
```
def ztest(x,mu,sigma,n):
    deno = sigma/math.sqrt(n)
    z = (x-mu)/deno
    p = 2*(1-norm.cdf(abs(z)))
    return z,p

s_mean = np.mean(sample_marks)
p_mean = 600
p_std = 100
s_size = np.count_nonzero(sample_marks)

ztest(s_mean,p_mean,p_std,s_size)

ztest(641,600,100,20)

```

Out[6]: (1.8335757415498277, 0.06671699590108493)

In [ ]:
```

```

```
EX 9 T TEST
REG NO:230701027
STUDENT NAME:AMIRTHAVARSHINI R U
DATE:08/10/2024
```

In [1]:
```
import numpy as np
from scipy import stats

student_scores = np.array([72, 89, 65, 73, 79, 84, 63, 76, 85, 75])


mu = 70
```

In [2]:
```
t_stat, p_value = stats.ttest_1samp(student_scores, mu)
```

In [3]:
```
t_stat
```

Out[3]: 2.2894683580127317

In [4]:
```
p_value
```

Out[4]: 0.047816221110566944

In [5]:
```
alpha = 0.05
```

In [6]:
```
if p_value < alpha:
    print("Reject the H0; there is a significant difference.")
else:
    print("Fail to reject the H0; there is no significant difference.")

```

```
Reject the H0; there is a significant difference.
```

In [ ]:
```

```

```
EX 10 ANOVA TEST
REG NO:230701027
STUDENT NAME:AMIRTHAVARSHINI R U
DATE:22/10/2024
```

In [1]:
```
import numpy as np
from scipy.stats import f_oneway
```

In [2]:
```
np.random.seed(42)
```

In [3]:
```
method_A_scores = np.random.normal(loc=80, scale=10, size=30)
method_B_scores = np.random.normal(loc=85, scale=10, size=30)
method_C_scores = np.random.normal(loc=90, scale=10, size=30)
```

In [4]:
```
f_statistic, p_value = f_oneway(method_A_scores, method_B_scores, method_
```

In [5]:
```
print("F-Statistic:", f_statistic)
print("P-Value:", p_value)
```

```
F-Statistic: 12.20952551797281
P-Value: 2.1200748140507065e-05
```

In [ ]:
```
```

```
EX 11 LINEAR REGRESSION
REG NO:230701027
STUDENT NAME:AMIRTHAVARSHINI R U
DATE:29/10/2024
```

In [7]:
```
#Supervised learning
#regression
import numpy as np
import pandas as pd
df=pd.read_csv('Salary_data.csv')
df
#feature-input
#label-output
```

Out[7]:

	YearsExperience	Salary
0	1.1	39343
1	1.3	46205
2	1.5	37731
3	2.0	43525
4	2.2	39891
5	2.9	56642
6	3.0	60150
7	3.2	54445
8	3.2	64445
9	3.7	57189
10	3.9	63218
11	4.0	55794
12	4.0	56957
13	4.1	57081
14	4.5	61111
15	4.9	67938
16	5.1	66029
17	5.3	83088
18	5.9	81363
19	6.0	93940
20	6.8	91738
21	7.1	98273
22	7.9	101302
23	8.2	113812
24	8.7	109431
25	9.0	105582
26	9.5	116969
27	9.6	112635
28	10.3	122391
29	10.5	121872

In [8]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     int64
dtypes: float64(1), int64(1)
memory usage: 612.0 bytes
```

In [9]:
```
1 df.dropna(inplace=True)
```

In [10]:
```
1 df
```

Out[10]:

	YearsExperience	Salary
0	1.1	39343
1	1.3	46205
2	1.5	37731
3	2.0	43525
4	2.2	39891
5	2.9	56642
6	3.0	60150
7	3.2	54445
8	3.2	64445
9	3.7	57189
10	3.9	63218
11	4.0	55794
12	4.0	56957
13	4.1	57081
14	4.5	61111
15	4.9	67938
16	5.1	66029
17	5.3	83088
18	5.9	81363
19	6.0	93940
20	6.8	91738
21	7.1	98273
22	7.9	101302
23	8.2	113812
24	8.7	109431
25	9.0	105582
26	9.5	116969
27	9.6	112635
28	10.3	122391
29	10.5	121872

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 612.0 bytes
```

In [12]:
```
df.describe()
```

Out[12]:

	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

In [13]:
```
features=df.iloc[:,[0]].values #first column
label=df.iloc[:,[1]].values #second column
```

In [14]:
```
from sklearn.model_selection import train_test_split #lib for ml
x_train,x_test,y_train,y_test=train_test_split(features,label,test_siz
#x-indep y-dep
#20% test 80% train
```

In [15]:
```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(x_train,y_train)
#model is created
```

Out[15]:

▾ LinearRegression

LinearRegression()

In [16]:
```
model.score(x_train,y_train)
#accuracy for training dataset
```

Out[16]:
```
0.9603182547438908
```

In [17]:
```
model.score(x_test,y_test)
#accuracy for testing dataset
```

Out[17]: 0.9184170849214232

In [18]:
```
model.coef_
#slope
```

Out[18]: array([[9281.30847068]])

In [19]:
```
model.intercept_
# y intercept
```

Out[19]: array([27166.73682891])

In [20]:
```
import pickle #to convert memory object into file
pickle.dump(model,open('SalaryPred.model','wb'))
#creates modelproduct
```

In [21]:
```
model=pickle.load(open('SalaryPred.model','rb'))
```

In [22]:
```
yr_of_exp=float(input("Enter Years Of Experience"))
yr_of_exp_NP=np.array([[yr_of_exp]])
Salary=model.predict(yr_of_exp_NP)
```

```
Enter Years Of Experience42
```

In [23]:
```
print("Estimated Salary for {} years of experience is {}:".format(yr_o
```

```
Estimated Salary for 42.0 years of experience is [[416981.69259751]]:
```

In [ ]:
```

```

```
EX 12 LOGISTIC REGRESSION
REG NO:230701027
STUDENT NAME:AMIRTHAVARSHINI R U
DATE:05/11/2024
```

In [1]:
```
import numpy as np
import pandas as pd
df=pd.read_csv('Social_Network_Ads.csv.csv')
df
```

Out[1]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

In [2]:
```
df.head()
```

Out[2]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

In [3]:
```
1 df.head(15)
```

Out[3]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0
10	15570769	Female	26	80000	0
11	15606274	Female	26	52000	0
12	15746139	Male	20	86000	0
13	15704987	Male	32	18000	0
14	15628972	Male	18	82000	0

In [4]:
```
1 df.tail()
```

Out[4]:

	User ID	Gender	Age	EstimatedSalary	Purchased
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

In [5]:
```
1 df.tail(20)
```

Out[5]:

	User ID	Gender	Age	EstimatedSalary	Purchased
380	15683758	Male	42	64000	0
381	15670615	Male	48	33000	1
382	15715622	Female	44	139000	1
383	15707634	Male	49	28000	1
384	15806901	Female	57	33000	1
385	15775335	Male	56	60000	1
386	15724150	Female	49	39000	1
387	15627220	Male	39	71000	0
388	15672330	Male	47	34000	1
389	15668521	Female	48	35000	1
390	15807837	Male	48	33000	1
391	15592570	Male	47	23000	1
392	15748589	Female	45	45000	1
393	15635893	Male	60	42000	1
394	15757632	Female	39	59000	0
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

In [6]:
```
1 features=df.iloc[:,[2,3]].values #3rd 4th column #selecting only neces
2 label=df.iloc[:,4].values #5th column
3
```

In [7]:
```
1 features
```

Out[7]:
```
array([[    19,  19000],
       [    35,  20000],
       [    26,  43000],
       [    27,  57000],
       [    19,  76000],
       [    27,  58000],
       [    27,  84000],
       [    32, 150000],
       [    25,  33000],
       [    35,  65000],
       [    26,  80000],
       [    26,  52000],
       [    20,  86000],
       [    32,  18000],
       [    18,  82000],
       [    29,  80000],
       [    47,  25000],
       [    45,  26000],
       [    46,  28000],
```

In [8]:
```
1 label
```

Out[8]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0,
       1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0,
       1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 0, 1], dtype=int64)
```

In [9]:
```
1 from sklearn.model_selection import train_test_split
2 from sklearn.linear_model import LogisticRegression
```

In [10]:

```
for i in range(1,401):
    x_train,x_test,y_train,y_test=train_test_split(features,label,test_
    model=LogisticRegression()
    model.fit(x_train,y_train)
    train_score=model.score(x_train,y_train)
    test_score=model.score(x_test,y_test)
    if test_score > train_score:
        print("Test {} Train{} Random State {}".format(test_score,trai
```

```
Test 0.6875 Train0.63125 Random State 3
Test 0.7375 Train0.61875 Random State 4
Test 0.6625 Train0.6375 Random State 5
Test 0.65 Train0.640625 Random State 6
Test 0.675 Train0.634375 Random State 7
Test 0.675 Train0.634375 Random State 8
Test 0.65 Train0.640625 Random State 10
Test 0.6625 Train0.6375 Random State 11
Test 0.7125 Train0.625 Random State 13
Test 0.675 Train0.634375 Random State 16
Test 0.7 Train0.628125 Random State 17
Test 0.7 Train0.628125 Random State 21
Test 0.65 Train0.640625 Random State 24
Test 0.6625 Train0.6375 Random State 25
Test 0.75 Train0.615625 Random State 26
Test 0.675 Train0.634375 Random State 27
Test 0.7 Train0.628125 Random State 28
Test 0.6875 Train0.63125 Random State 29
Test 0.6875 Train0.63125 Random State 31
```

In [ ]:

```
1
```