In [52]:
```python
import pandas as pd
df_titanic=pd.read_csv(r"C:\Users\HP\Downloads\titanic.csv")
df_titanic
```

Out[52]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

In [53]:
```python
import matplotlib.pyplot as plt
class1=df_titanic[df_titanic['Pclass']==1]['Age'].dropna()
```

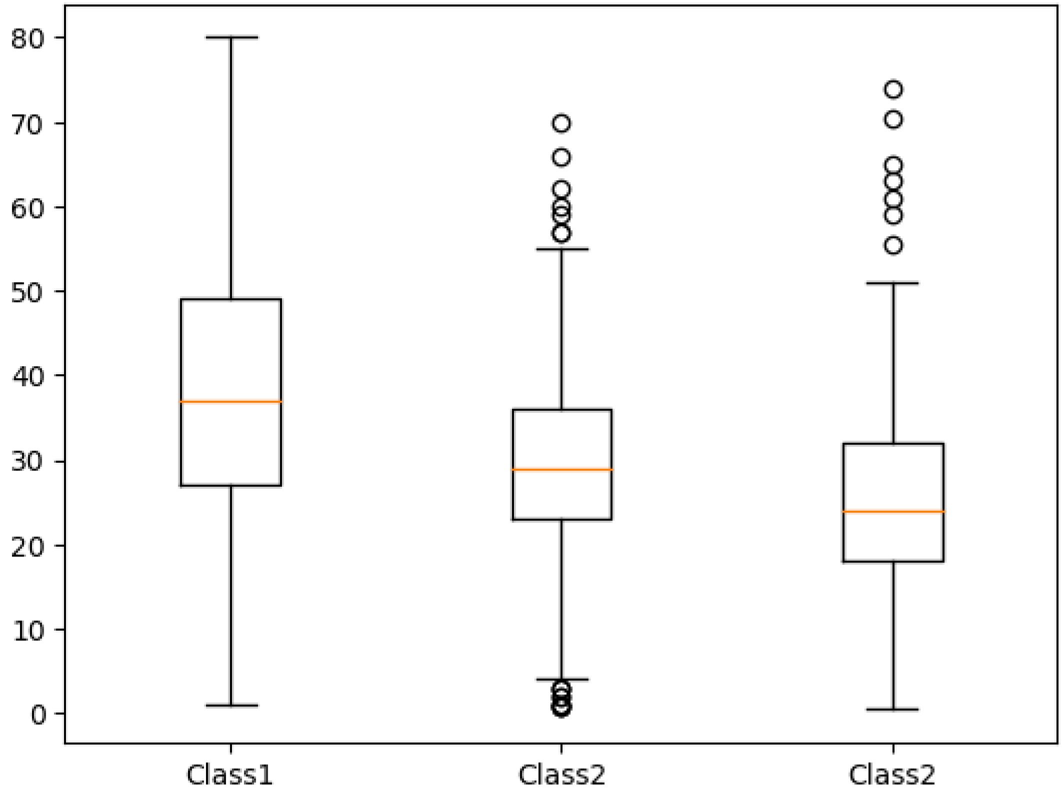
In [54]:
```python
class2=df_titanic[df_titanic['Pclass']==2]['Age'].dropna()
```

In [55]:
```python
class3=df_titanic[df_titanic['Pclass']==3]['Age'].dropna()
```

In [56]:
```python
l1=[class1,class2,class3]
plt.boxplot(l1,labels=["Class1","Class2","Class2"])
```

Out[56]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x1ef9f33d330>,
  <matplotlib.lines.Line2D at 0x1ef9f33d5d0>,
  <matplotlib.lines.Line2D at 0x1ef9f13c6d0>,
  <matplotlib.lines.Line2D at 0x1ef9f33e1a0>,
  <matplotlib.lines.Line2D at 0x1ef9f33f160>,
  <matplotlib.lines.Line2D at 0x1ef9f33f400>],
 'caps': [<matplotlib.lines.Line2D at 0x1ef9f33d870>,
  <matplotlib.lines.Line2D at 0x1ef9f33db10>,
  <matplotlib.lines.Line2D at 0x1ef9f33e440>,
  <matplotlib.lines.Line2D at 0x1ef9f33e6e0>,
  <matplotlib.lines.Line2D at 0x1ef9f33f6a0>,
  <matplotlib.lines.Line2D at 0x1ef9f33f940>],
 'boxes': [<matplotlib.lines.Line2D at 0x1ef9f33d090>,
  <matplotlib.lines.Line2D at 0x1ef9f104cd0>,
  <matplotlib.lines.Line2D at 0x1ef9f33eec0>],
 'medians': [<matplotlib.lines.Line2D at 0x1ef9f33ddb0>,
  <matplotlib.lines.Line2D at 0x1ef9f33e980>,
  <matplotlib.lines.Line2D at 0x1ef9f33fbe0>],
 'fliers': [<matplotlib.lines.Line2D at 0x1ef9f33e050>,
  <matplotlib.lines.Line2D at 0x1ef9f33ec20>,
  <matplotlib.lines.Line2D at 0x1ef9f33fe80>],
 'means': []}
```



In [57]:

```
df_titanic.rename(columns={"Sex":"Gender"},inplace=True)
df_titanic
```

Out[57]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

In [58]:
```
df_titanic['Gender']=df_titanic['Gender'].map({'male':0,'female':1})
df_titanic
```

Out[58]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	0	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	1	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	0	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	0	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	1	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	1	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	0	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	0	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

In [59]:
```
m=(df_titanic['Age']<25) & (df_titanic['Gender']==1)
```

In [60]:
```
df_titanic[m]
```

Out[60]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	1	14.0	1	0	237736	30.0708	NaN	C
10	11	1	3	Sandstrom, Miss. Marguerite Rut	1	4.0	1	1	PP 9549	16.7000	G6	S
14	15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	1	14.0	0	0	350406	7.8542	NaN	S
22	23	1	3	McGowan, Miss. Anna "Annie"	1	15.0	0	0	330923	8.0292	NaN	Q
24	25	0	3	Palsson, Miss. Torborg Danira	1	8.0	3	1	349909	21.0750	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
855	856	1	3	Aks, Mrs. Sam (Leah Rosen)	1	18.0	0	1	392091	9.3500	NaN	S
858	859	1	3	Baclini, Mrs. Solomon (Latifa Qurban)	1	24.0	0	3	2666	19.2583	NaN	C
875	876	1	3	Najib, Miss. Adele Kiamie "Jane"	1	15.0	0	0	2667	7.2250	NaN	C
882	883	0	3	Dahlberg, Miss. Gerda Ulrika	1	22.0	0	0	7552	10.5167	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	1	19.0	0	0	112053	30.0000	B42	S

117 rows × 12 columns

In [61]:

```
#number of males survived
a=(df_titanic['Survived']==1) & (df_titanic['Gender']==1)
print(len(df_titanic[a]))
df_titanic[a]
```

233

Out[61]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	1	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	35.0	1	0	113803	53.1000	C123	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	1	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	1	14.0	1	0	237736	30.0708	NaN	C
...	...	...	...	...	...	...	...	...	...	...	...	...
874	875	1	2	Abelson, Mrs. Samuel (Hannah Wizosky)	1	28.0	1	0	P/PP 3381	24.0000	NaN	C
875	876	1	3	Najib, Miss. Adele Kiamie "Jane"	1	15.0	0	0	2667	7.2250	NaN	C
879	880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	1	56.0	0	1	11767	83.1583	C50	C
880	881	1	2	Shelley, Mrs. William (Imanita Parrish Hall)	1	25.0	0	1	230433	26.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	1	19.0	0	0	112053	30.0000	B42	S

233 rows × 12 columns

In [62]:

```
b=(df_titanic['Survived']==1) & (df_titanic['Gender']==0)
```

In [63]:

```
#number of males survived

b=(df_titanic['Survived']==1) & (df_titanic['Gender']==0)

print(len(df_titanic[b]))

df_titanic[b]
```

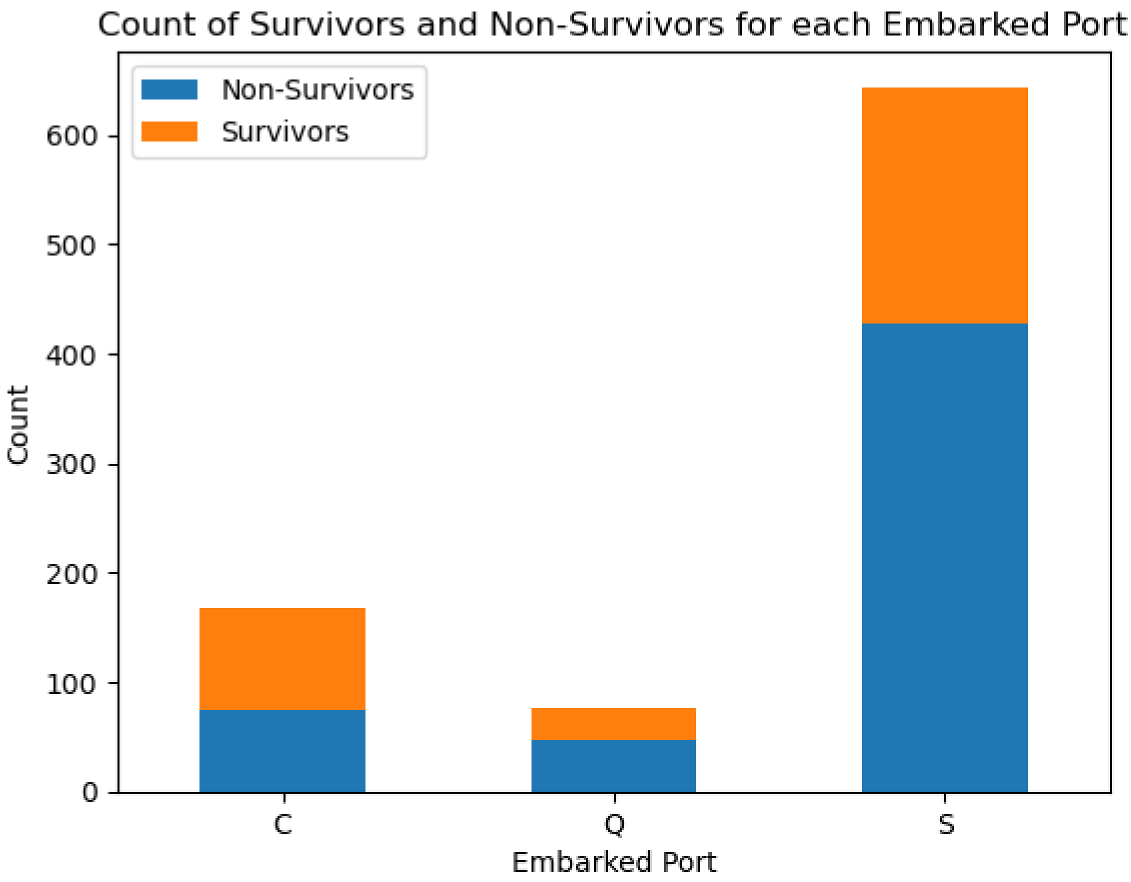
```
109
```

Out[63]:

	PassengerId	Survived	Pclass	Name	Gender	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
17	18	1	2	Williams, Mr. Charles Eugene	0	NaN	0	0	244373	13.0000	NaN	S
21	22	1	2	Beesley, Mr. Lawrence	0	34.0	0	0	248698	13.0000	D56	S
23	24	1	1	Sloper, Mr. William Thompson	0	28.0	0	0	113788	35.5000	A6	S
36	37	1	3	Mamee, Mr. Hanna	0	NaN	0	0	2677	7.2292	NaN	C
55	56	1	1	Woolner, Mr. Hugh	0	NaN	0	0	19947	35.5000	C52	S
...	...	...	...	...	...	...	...	...	...	...	...	...
838	839	1	3	Chip, Mr. Chang	0	32.0	0	0	1601	56.4958	NaN	S
839	840	1	1	Marechal, Mr. Pierre	0	NaN	0	0	11774	29.7000	C47	C
857	858	1	1	Daly, Mr. Peter Denis	0	51.0	0	0	113055	26.5500	E17	S
869	870	1	3	Johnson, Master. Harold Theodor	0	4.0	1	1	347742	11.1333	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	0	26.0	0	0	111369	30.0000	C148	C

109 rows × 12 columns

In [64]:
```
embarked_survived_count = df_titanic.groupby(['Embarked','Survived']).size()
embarked_survived_count.plot(kind='bar',stacked=True)
plt.title('Count of Survivors and Non-Survivors for each Embarked Port')
plt.xlabel('Embarked Port')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.legend(['Non-Survivors','Survivors'])
plt.show()
```



In [65]:
```
import seaborn as sns
tips=sns.load_dataset('tips')
tips
```

Out[65]:

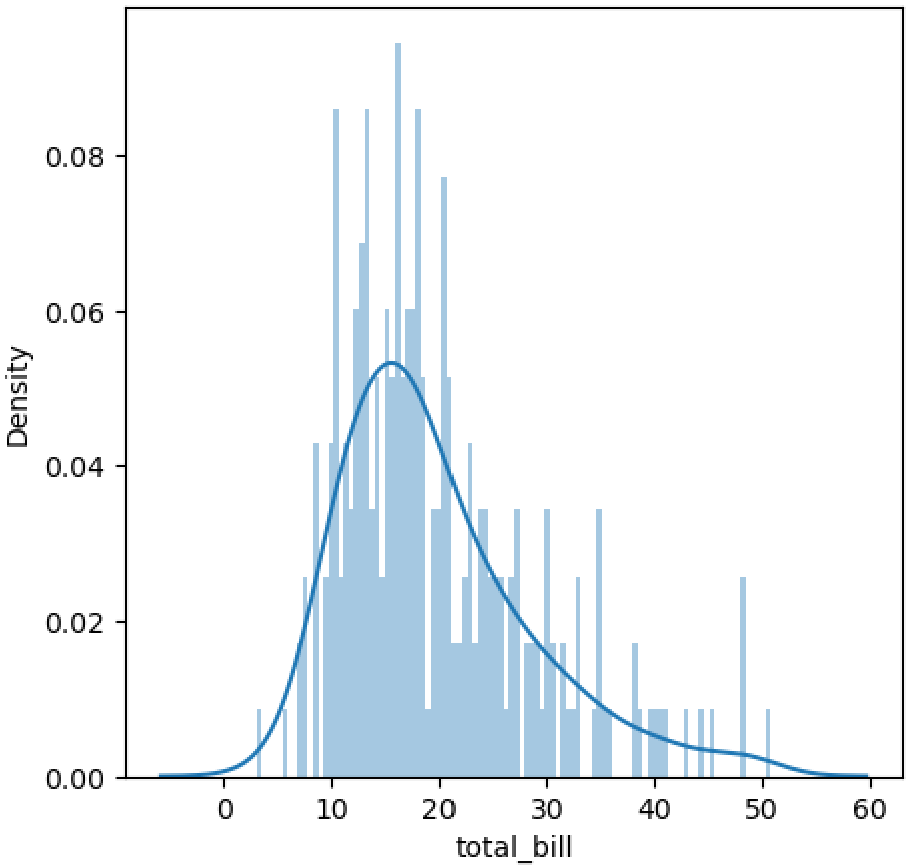
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

In [66]:
```
#-----> distplot
#It will  take only one column

plt.figure(figsize=(5,5))

sns.distplot(tips['total_bill'],bins=100)   #kde=True by default -->kde is u
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_19832\1281462628.py:6: UserWarnin
g:

`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 wit
h
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(tips['total_bill'],bins=100)   #kde=True by default -->kde i
s used for graph
```

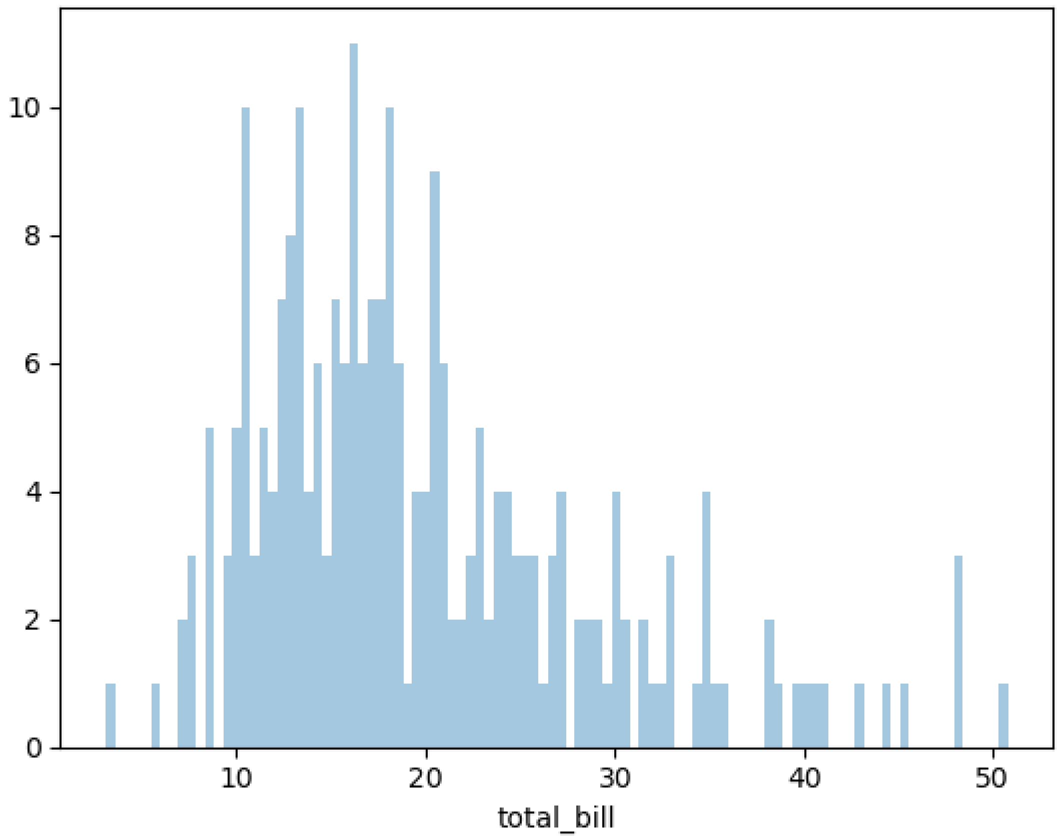
Out[66]:
```
<Axes: xlabel='total_bill', ylabel='Density'>
```



In [67]:
```
sns.distplot(tips['total_bill'],bins=100,kde=False)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_19832\3888612417.py:1: UserWarnin
g:

`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 wit
h
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(tips['total_bill'],bins=100,kde=False)
```

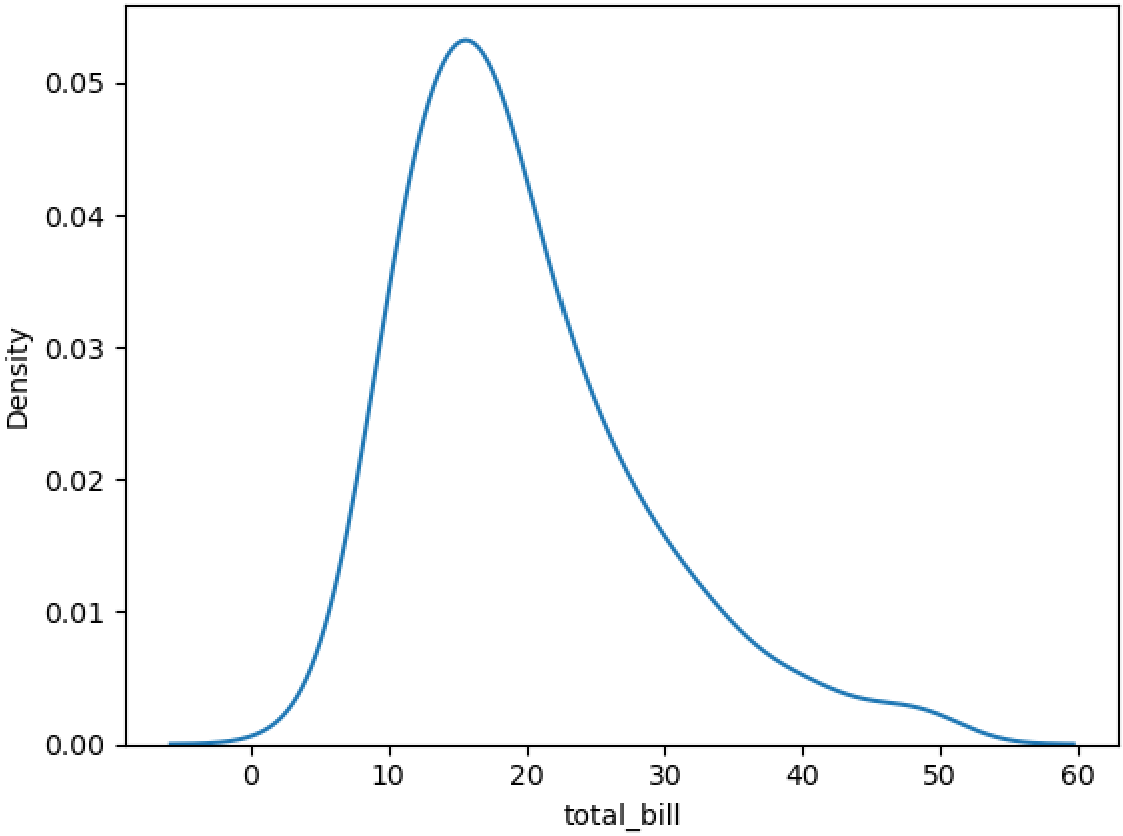
Out[67]:
```
<Axes: xlabel='total_bill'>
```



In [68]:
```python
sns.distplot(tips['total_bill'],bins=100,kde=True,hist=False)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_19832\951507828.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 wit
h
similar flexibility) or `kdeplot` (an axes-level function for kernel densit
y plots).

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

  sns.distplot(tips['total_bill'],bins=100,kde=True,hist=False)
```

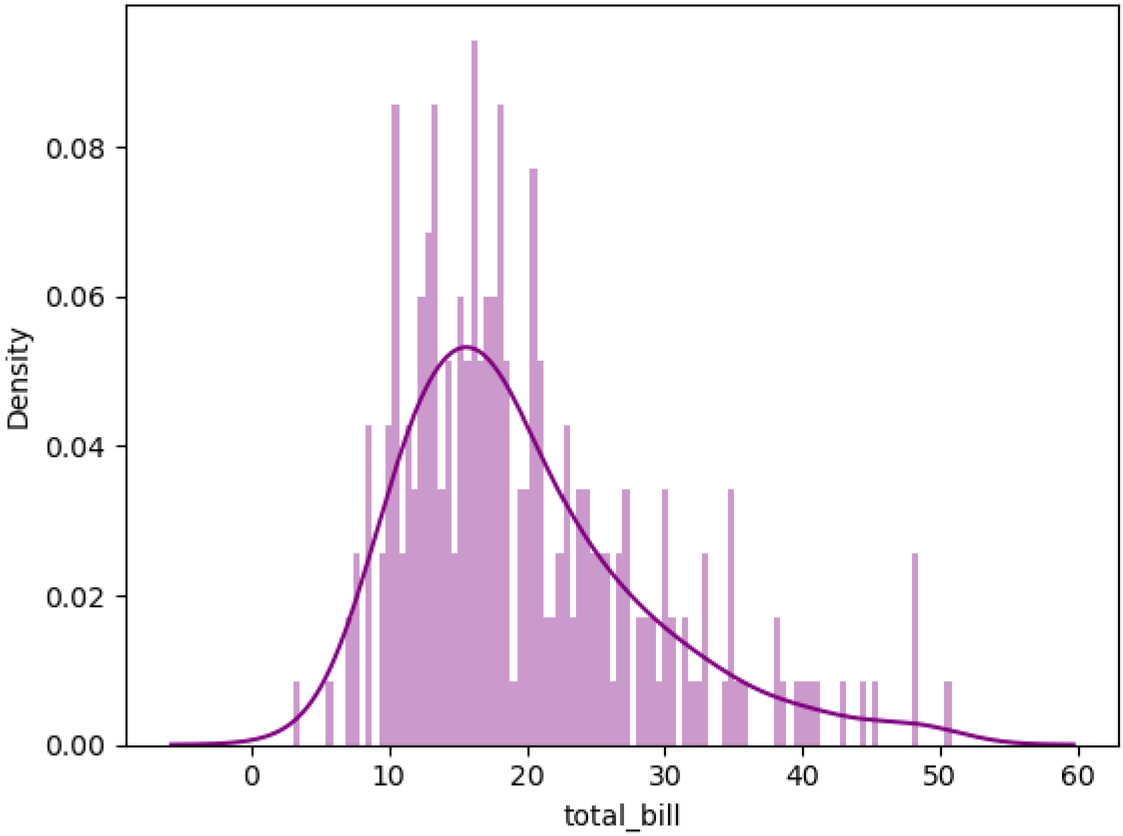
Out[68]:
```
<Axes: xlabel='total_bill', ylabel='Density'>
```



In [69]:
```
sns.distplot(tips['total_bill'],bins=100,hist=True,color='purple')
```

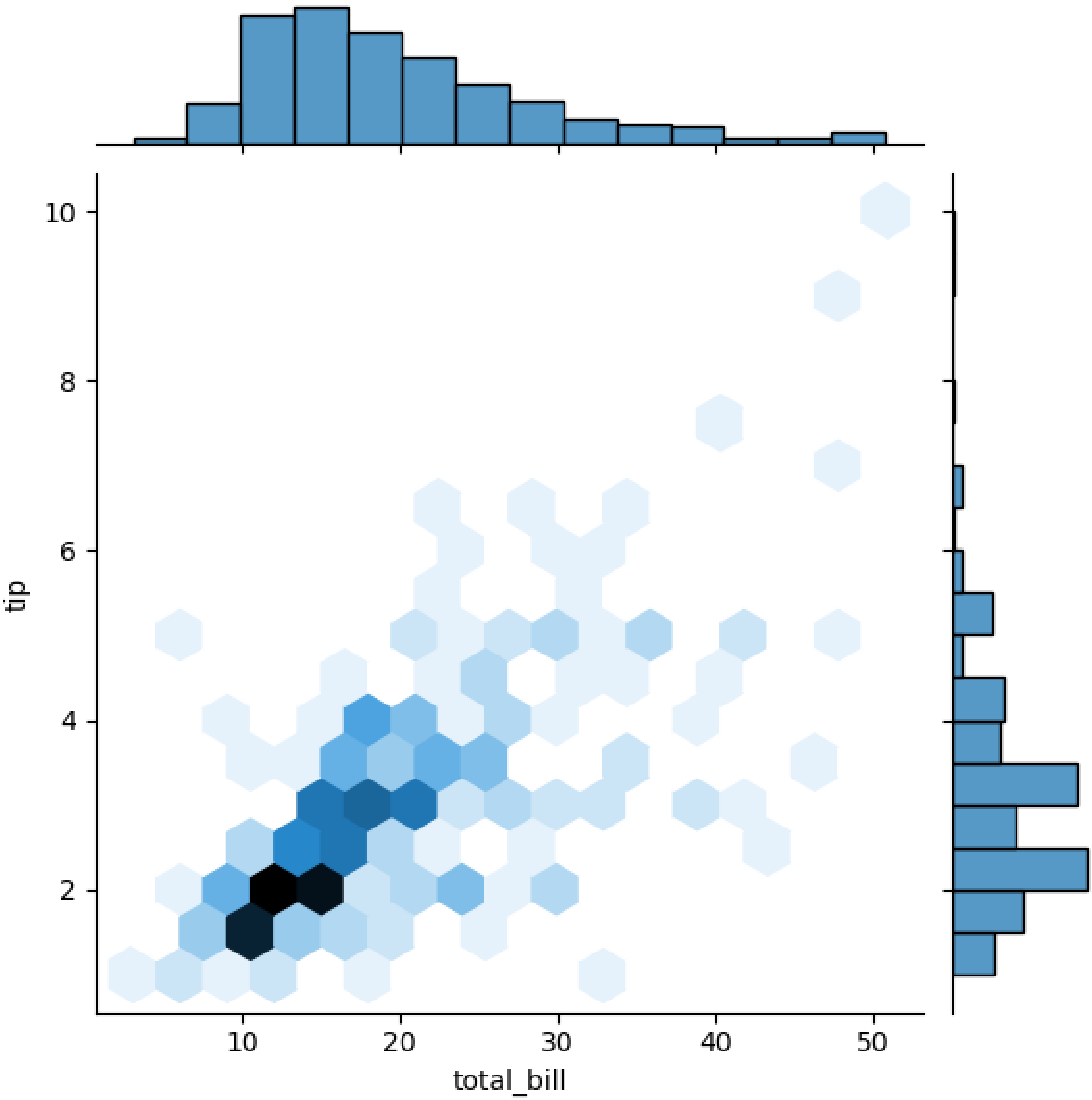
```
C:\Users\HP\AppData\Local\Temp\ipykernel_19832\3206045060.py:1: UserWarnin
g:

`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 wit
h
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(tips['total_bill'],bins=100,hist=True,color='purple')
```

Out[69]:
```
<Axes: xlabel='total_bill', ylabel='Density'>
```



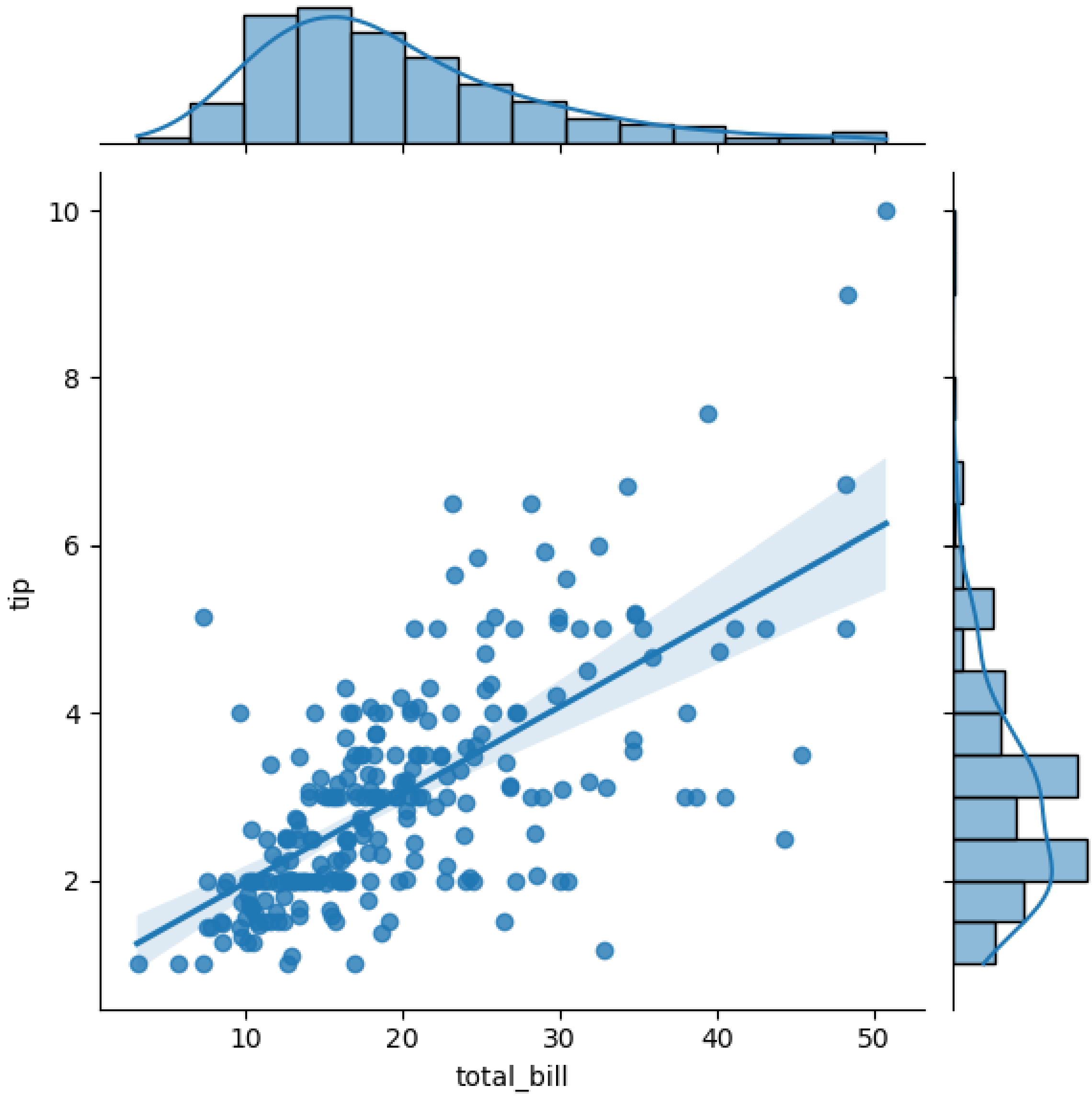
In [70]:
```
#jointplot()
sns.jointplot(x='total_bill',y='tip',data=tips,kind='hex')
```

Out[70]: <seaborn.axisgrid.JointGrid at 0x1ef9f73add0>



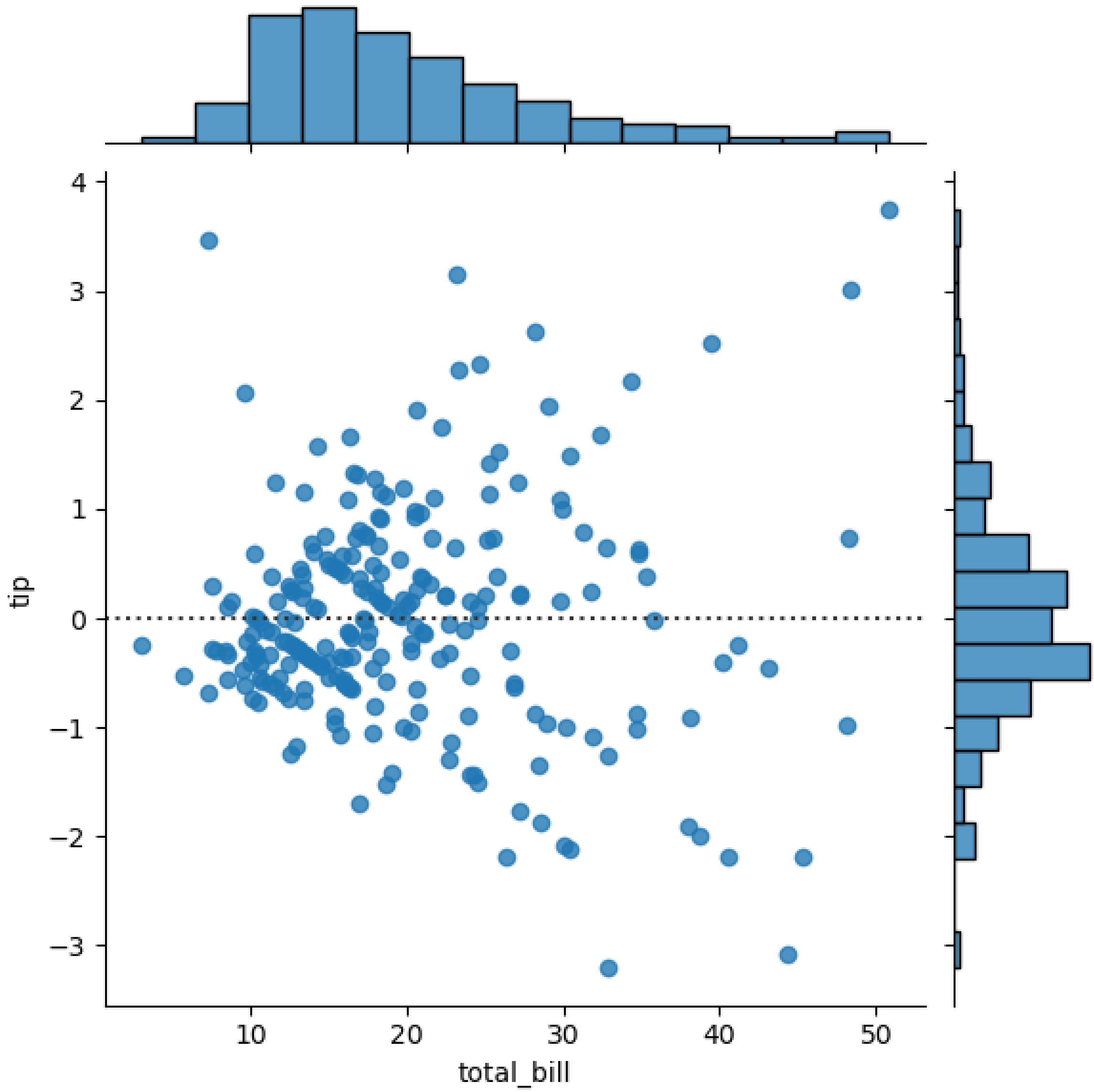
In [71]:
```
sns.jointplot(x='total_bill',y='tip',data=tips,kind='reg')
```

Out[71]:
```
<seaborn.axisgrid.JointGrid at 0x1ef9f852f20>
```



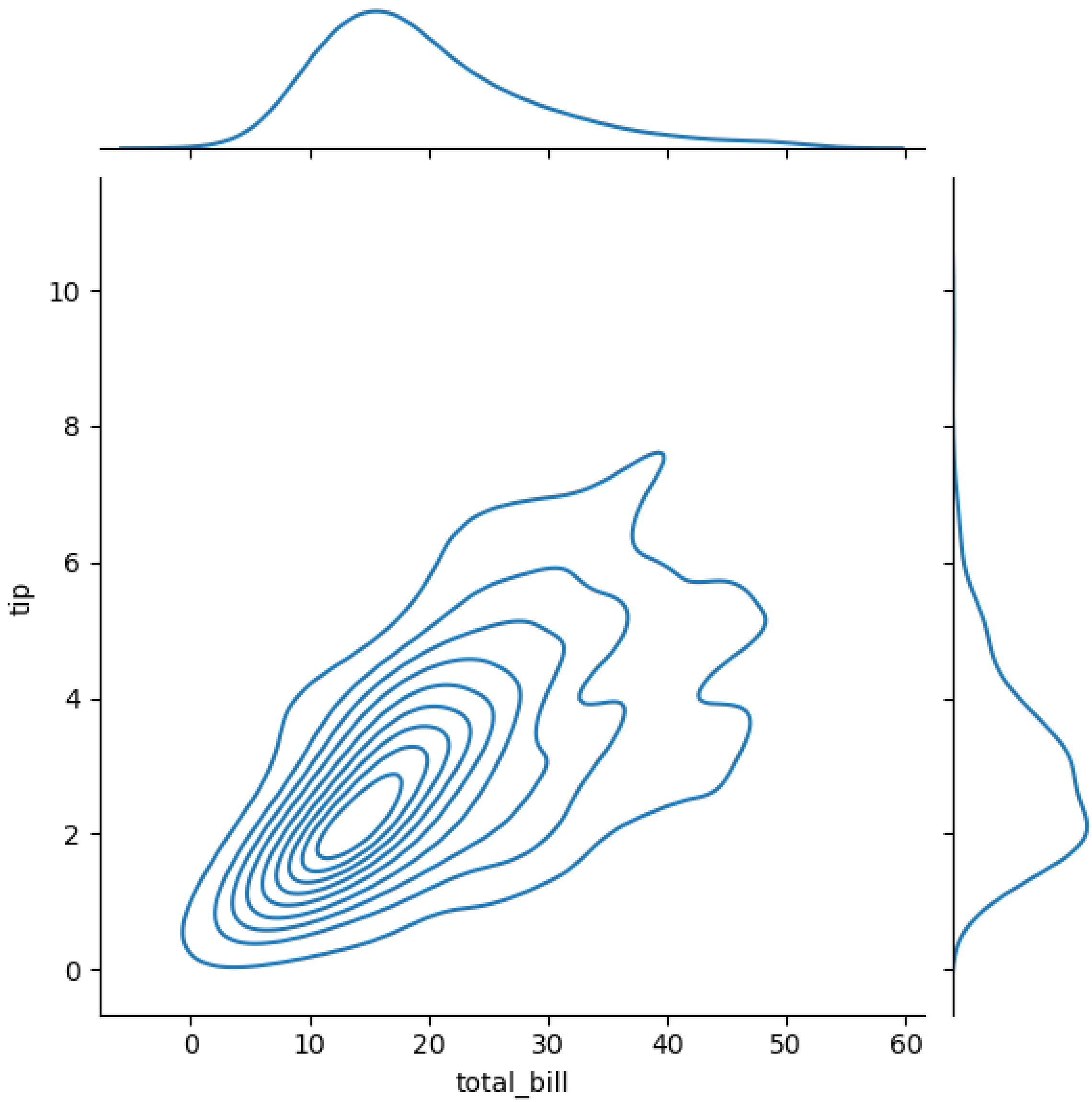
In [72]:
```
sns.jointplot(x='total_bill',y='tip',data=tips,kind='resid')
```

Out[72]:
```
<seaborn.axisgrid.JointGrid at 0x1ef9fe46d40>
```



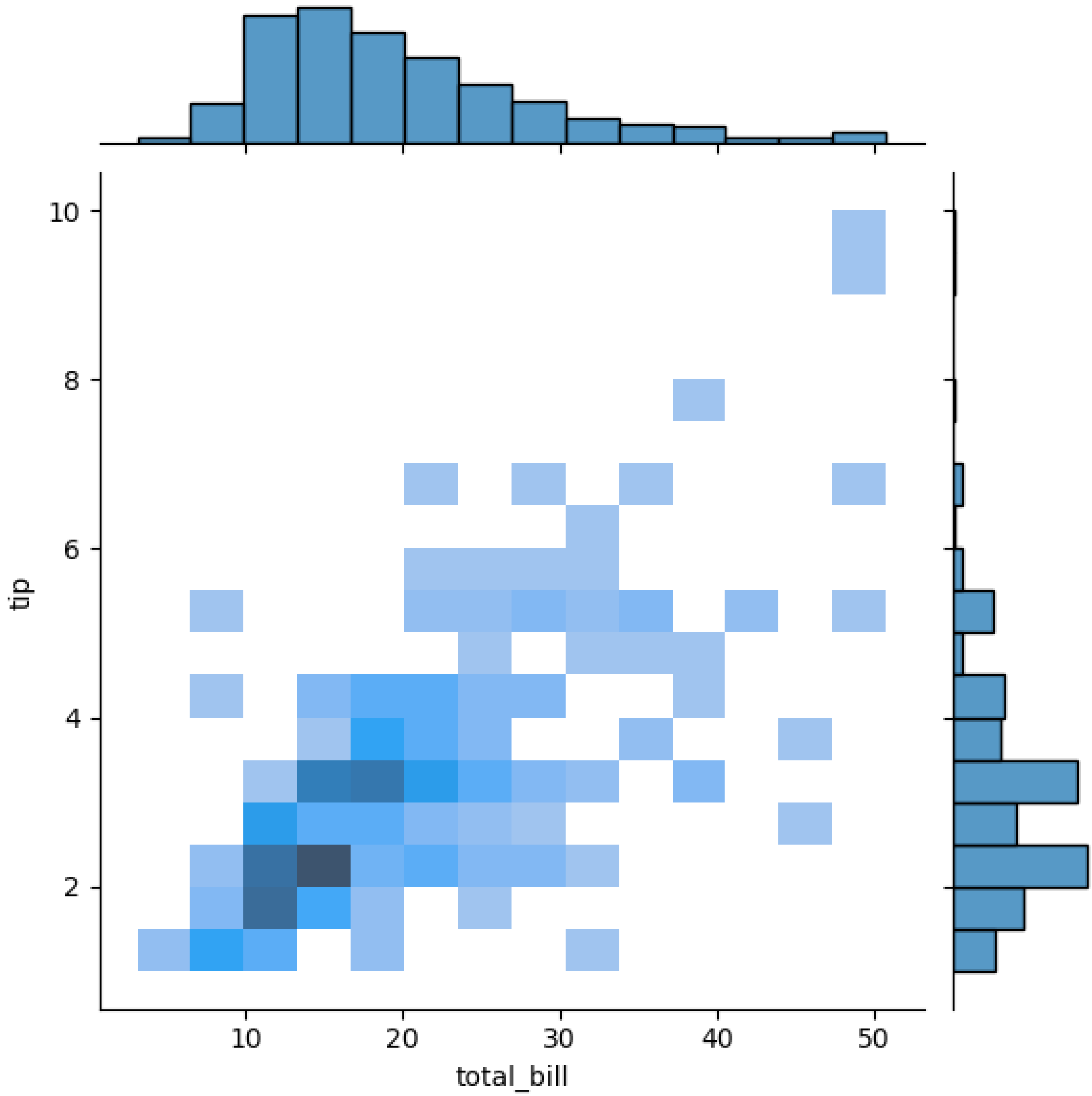
In [73]:
```
sns.jointplot(x='total_bill',y='tip',data=tips,kind='kde')
```

Out[73]: <seaborn.axisgrid.JointGrid at 0x1ef9fe47f10>

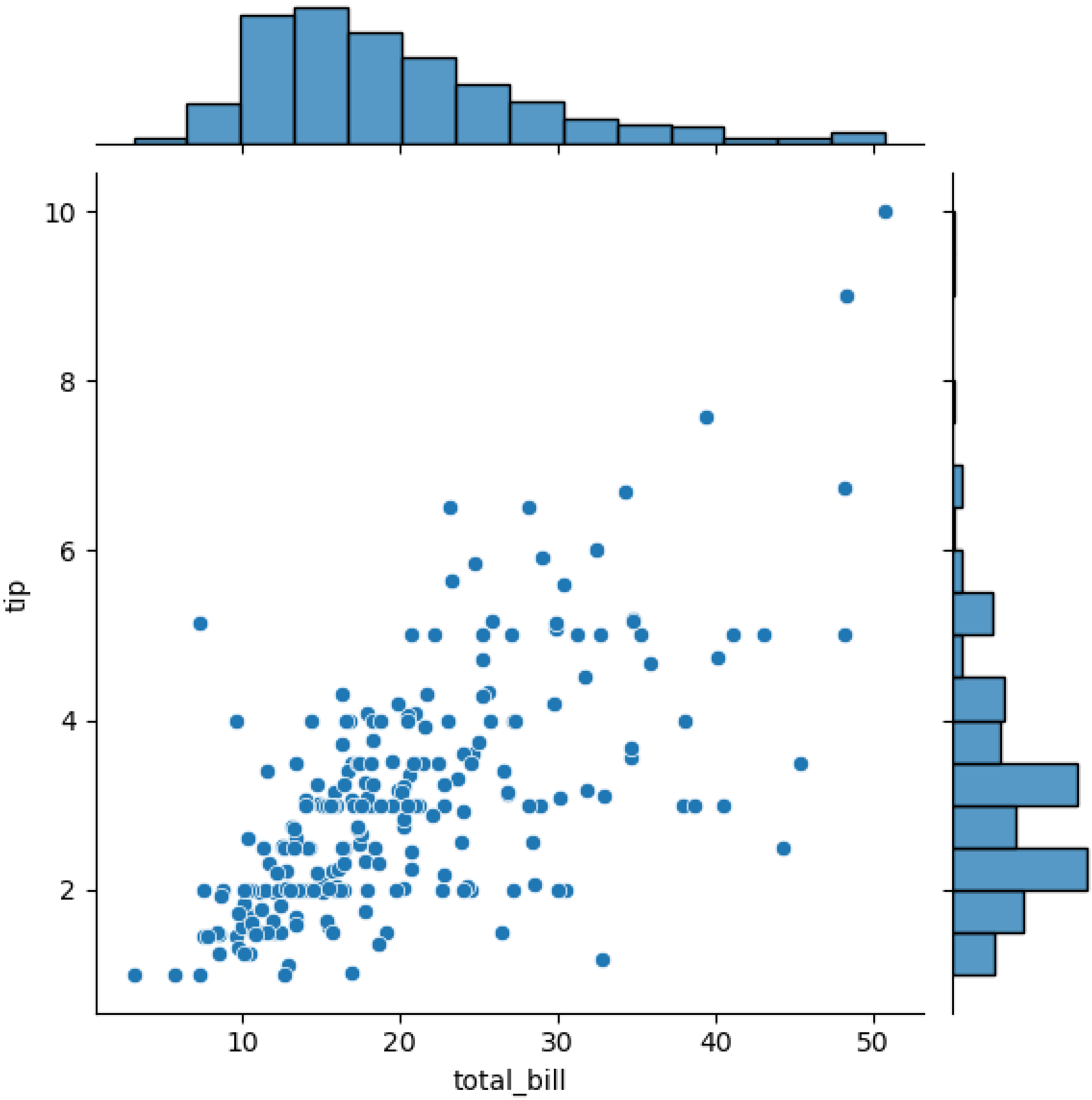


In [74]:
```
sns.jointplot(x='total_bill',y='tip',data=tips,kind='hist')
```

Out[74]:
```
<seaborn.axisgrid.JointGrid at 0x1ef9f87fdc0>
```

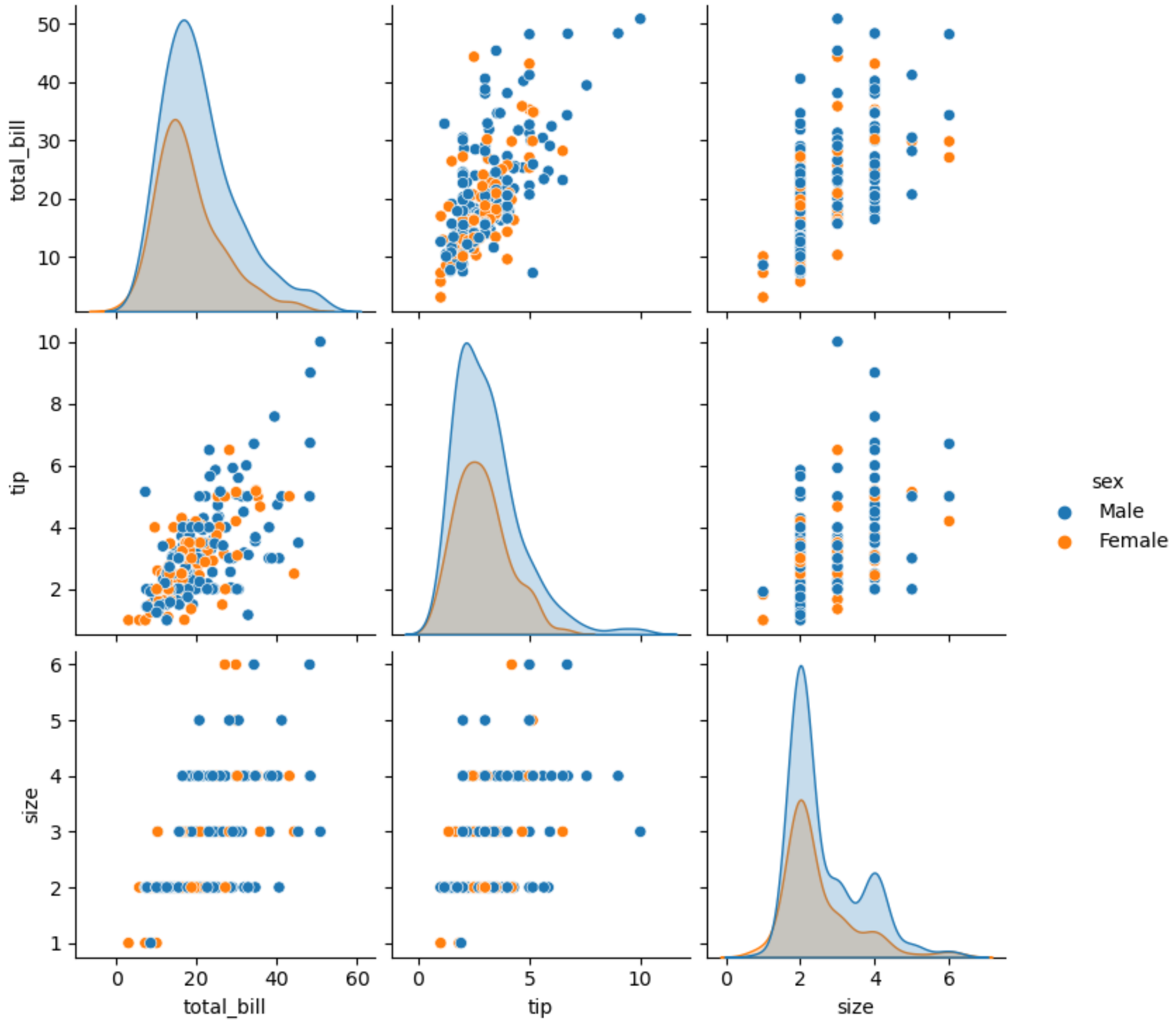


In [75]:
```
sns.jointplot(x='total_bill',y='tip',data=tips,kind='scatter')
```
Out[75]:
```
<seaborn.axisgrid.JointGrid at 0x1efa1a96e30>
```



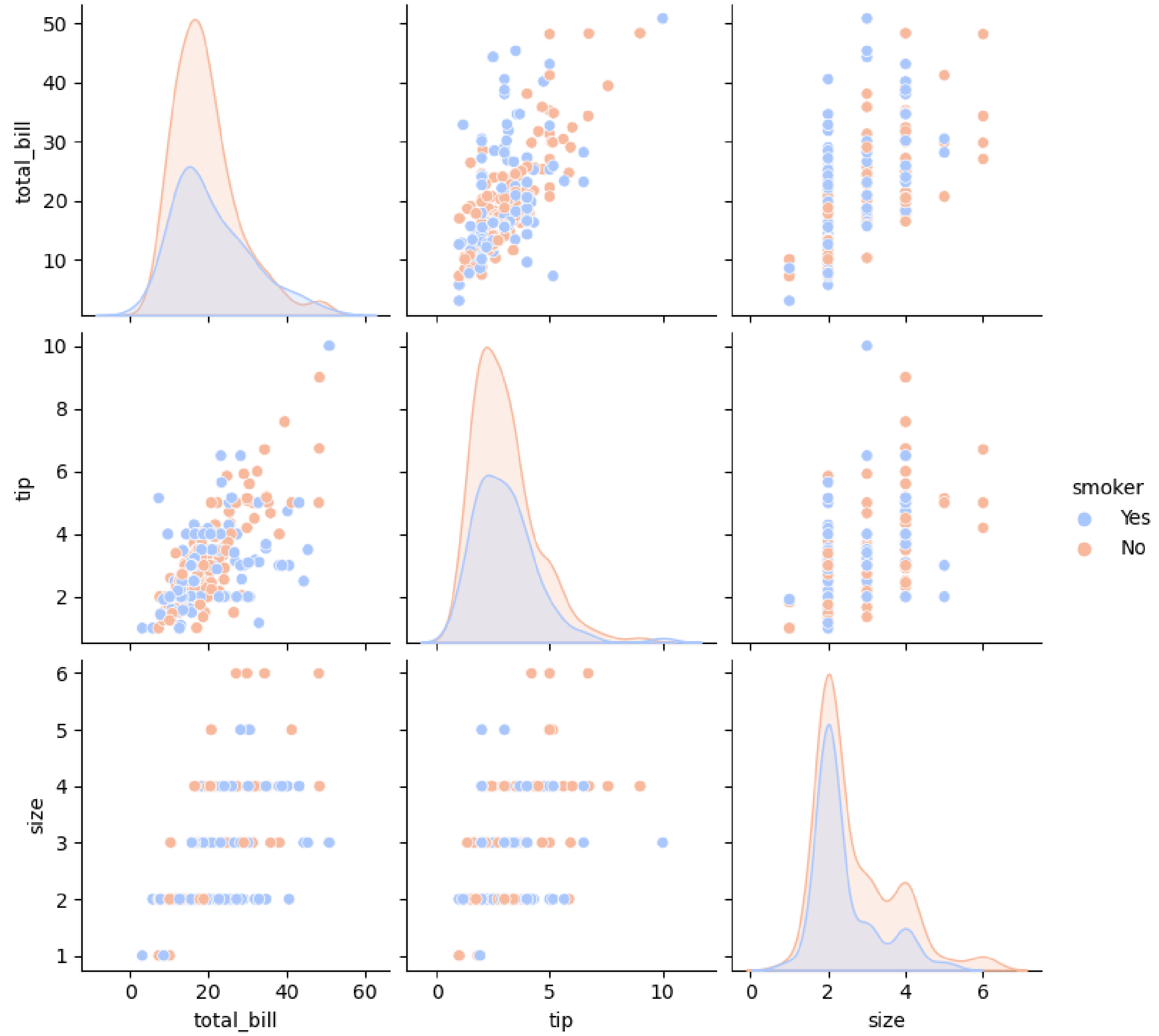
In [76]:
```
sns.pairplot(tips,hue="sex")
```

Out[76]: <seaborn.axisgrid.PairGrid at 0x1efa1ac4c70>



In [77]:
```
sns.pairplot(tips,hue="smoker",palette='coolwarm')
```

Out[77]:
```
<seaborn.axisgrid.PairGrid at 0x1ef9f7ebf10>
```



In [78]:
```
#logistic regression

# Logistic regression is a statitical method used to model the relationship
#a binary dependent variable and one or more independent variables

#in logistic regression, the dependent variabl is a binary,meaning it can or
#on two values,labelled as 0 or 1

#The dependent variables can be either continous or categorical
```

In [79]:
```
import numpy as np
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_s
```

In [80]:
```
y_pred=np.array([0.3,0.6,0.8,0.2,0.4,0.9,0.1,0.7,0.5,0.6])
y_true=np.array([0,1,1,0,0,1,0,1,1,1])
```

In [81]:
```
#Accuracy
#Accuracy measures the percentage of correctly classified instance of all ir
#instances out of all instance

accuracy=accuracy_score(y_true,np.round(y_pred))
accuracy
```

Out[81]:
```
0.9
```

In [82]:
```
# Precision
# Precision measured the proportion of true positive prediction out of all p
```

```
#precision=true positive/all positive
precision=precision_score(y_true,np.round(y_pred))
precision
```

Out[82]:

```
1.0
```

In [83]:

```
#Recall

# Recall measures the proportion of true positive prediction out of all actu

#recall=true positive/actual positive
recall=recall_score(y_true,np.round(y_pred))
recall
```

Out[83]:

```
0.8333333333333334
```

In [84]:

```
#f1_score

#IT is the mean of precision and recall

f1=f1_score(y_true,np.round(y_pred))
f1
```

Out[84]:

```
0.9090909090909091
```

In [85]:

```
#Confusion matrix

#It is a table gives the performance of a classification model

#It shows true positive,true negative ,false positive,false negative

matrix=confusion_matrix(y_true,np.round(y_pred))
matrix
```

Out[85]:

```
array([[4, 0],
       [1, 5]], dtype=int64)
```

In [86]:

```
# Eg:1

#Logistic regression
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

In [87]:

```
iris=load_iris()
iris
```

```
Out[87]: {'data': array([[5.1, 3.5, 1.4, 0.2],
                [4.9, 3. , 1.4, 0.2],
                [4.7, 3.2, 1.3, 0.2],
                [4.6, 3.1, 1.5, 0.2],
                [5. , 3.6, 1.4, 0.2],
                [5.4, 3.9, 1.7, 0.4],
                [4.6, 3.4, 1.4, 0.3],
                [5. , 3.4, 1.5, 0.2],
                [4.4, 2.9, 1.4, 0.2],
                [4.9, 3.1, 1.5, 0.1],
                [5.4, 3.7, 1.5, 0.2],
                [4.8, 3.4, 1.6, 0.2],
                [4.8, 3. , 1.4, 0.1],
                [4.3, 3. , 1.1, 0.1],
                [5.8, 4. , 1.2, 0.2],
                [5.7, 4.4, 1.5, 0.4],
                [5.4, 3.9, 1.3, 0.4],
                [5.1, 3.5, 1.4, 0.3],
                [5.7, 3.8, 1.7, 0.3],
                [5.1, 3.8, 1.5, 0.3],
                [5.4, 3.4, 1.7, 0.2],
                [5.1, 3.7, 1.5, 0.4],
                [4.6, 3.6, 1. , 0.2],
                [5.1, 3.3, 1.7, 0.5],
                [4.8, 3.4, 1.9, 0.2],
                [5. , 3. , 1.6, 0.2],
                [5. , 3.4, 1.6, 0.4],
                [5.2, 3.5, 1.5, 0.2],
                [5.2, 3.4, 1.4, 0.2],
                [4.7, 3.2, 1.6, 0.2],
                [4.8, 3.1, 1.6, 0.2],
                [5.4, 3.4, 1.5, 0.4],
                [5.2, 4.1, 1.5, 0.1],
                [5.5, 4.2, 1.4, 0.2],
                [4.9, 3.1, 1.5, 0.2],
                [5. , 3.2, 1.2, 0.2],
                [5.5, 3.5, 1.3, 0.2],
                [4.9, 3.6, 1.4, 0.1],
                [4.4, 3. , 1.3, 0.2],
                [5.1, 3.4, 1.5, 0.2],
                [5. , 3.5, 1.3, 0.3],
                [4.5, 2.3, 1.3, 0.3],
                [4.4, 3.2, 1.3, 0.2],
                [5. , 3.5, 1.6, 0.6],
                [5.1, 3.8, 1.9, 0.4],
                [4.8, 3. , 1.4, 0.3],
                [5.1, 3.8, 1.6, 0.2],
                [4.6, 3.2, 1.4, 0.2],
                [5.3, 3.7, 1.5, 0.2],
                [5. , 3.3, 1.4, 0.2],
                [7. , 3.2, 4.7, 1.4],
                [6.4, 3.2, 4.5, 1.5],
                [6.9, 3.1, 4.9, 1.5],
                [5.5, 2.3, 4. , 1.3],
                [6.5, 2.8, 4.6, 1.5],
                [5.7, 2.8, 4.5, 1.3],
                [6.3, 3.3, 4.7, 1.6],
                [4.9, 2.4, 3.3, 1. ],
                [6.6, 2.9, 4.6, 1.3],
                [5.2, 2.7, 3.9, 1.4],
                [5. , 2. , 3.5, 1. ],
                [5.9, 3. , 4.2, 1.5],
                [6. , 2.2, 4. , 1. ],
                [6.1, 2.9, 4.7, 1.4],
```

```
[5.6, 2.9, 3.6, 1.3],
[6.7, 3.1, 4.4, 1.4],
[5.6, 3. , 4.5, 1.5],
[5.8, 2.7, 4.1, 1. ],
[6.2, 2.2, 4.5, 1.5],
[5.6, 2.5, 3.9, 1.1],
[5.9, 3.2, 4.8, 1.8],
[6.1, 2.8, 4. , 1.3],
[6.3, 2.5, 4.9, 1.5],
[6.1, 2.8, 4.7, 1.2],
[6.4, 2.9, 4.3, 1.3],
[6.6, 3. , 4.4, 1.4],
[6.8, 2.8, 4.8, 1.4],
[6.7, 3. , 5. , 1.7],
[6. , 2.9, 4.5, 1.5],
[5.7, 2.6, 3.5, 1. ],
[5.5, 2.4, 3.8, 1.1],
[5.5, 2.4, 3.7, 1. ],
[5.8, 2.7, 3.9, 1.2],
[6. , 2.7, 5.1, 1.6],
[5.4, 3. , 4.5, 1.5],
[6. , 3.4, 4.5, 1.6],
[6.7, 3.1, 4.7, 1.5],
[6.3, 2.3, 4.4, 1.3],
[5.6, 3. , 4.1, 1.3],
[5.5, 2.5, 4. , 1.3],
[5.5, 2.6, 4.4, 1.2],
[6.1, 3. , 4.6, 1.4],
[5.8, 2.6, 4. , 1.2],
[5. , 2.3, 3.3, 1. ],
[5.6, 2.7, 4.2, 1.3],
[5.7, 3. , 4.2, 1.2],
[5.7, 2.9, 4.2, 1.3],
[6.2, 2.9, 4.3, 1.3],
[5.1, 2.5, 3. , 1.1],
[5.7, 2.8, 4.1, 1.3],
[6.3, 3.3, 6. , 2.5],
[5.8, 2.7, 5.1, 1.9],
[7.1, 3. , 5.9, 2.1],
[6.3, 2.9, 5.6, 1.8],
[6.5, 3. , 5.8, 2.2],
[7.6, 3. , 6.6, 2.1],
[4.9, 2.5, 4.5, 1.7],
[7.3, 2.9, 6.3, 1.8],
[6.7, 2.5, 5.8, 1.8],
[7.2, 3.6, 6.1, 2.5],
[6.5, 3.2, 5.1, 2. ],
[6.4, 2.7, 5.3, 1.9],
[6.8, 3. , 5.5, 2.1],
[5.7, 2.5, 5. , 2. ],
[5.8, 2.8, 5.1, 2.4],
[6.4, 3.2, 5.3, 2.3],
[6.5, 3. , 5.5, 1.8],
[7.7, 3.8, 6.7, 2.2],
[7.7, 2.6, 6.9, 2.3],
[6. , 2.2, 5. , 1.5],
[6.9, 3.2, 5.7, 2.3],
[5.6, 2.8, 4.9, 2. ],
[7.7, 2.8, 6.7, 2. ],
[6.3, 2.7, 4.9, 1.8],
[6.7, 3.3, 5.7, 2.1],
[7.2, 3.2, 6. , 1.8],
[6.2, 2.8, 4.8, 1.8],
[6.1, 3. , 4.9, 1.8],
```

```
        [6.4, 2.8, 5.6, 2.1],
        [7.2, 3. , 5.8, 1.6],
        [7.4, 2.8, 6.1, 1.9],
        [7.9, 3.8, 6.4, 2. ],
        [6.4, 2.8, 5.6, 2.2],
        [6.3, 2.8, 5.1, 1.5],
        [6.1, 2.6, 5.6, 1.4],
        [7.7, 3. , 6.1, 2.3],
        [6.3, 3.4, 5.6, 2.4],
        [6.4, 3.1, 5.5, 1.8],
        [6. , 3. , 4.8, 1.8],
        [6.9, 3.1, 5.4, 2.1],
        [6.7, 3.1, 5.6, 2.4],
        [6.9, 3.1, 5.1, 2.3],
        [5.8, 2.7, 5.1, 1.9],
        [6.8, 3.2, 5.9, 2.3],
        [6.7, 3.3, 5.7, 2.5],
        [6.7, 3. , 5.2, 2.3],
        [6.3, 2.5, 5. , 1.9],
        [6.5, 3. , 5.2, 2. ],
        [6.2, 3.4, 5.4, 2.3],
        [5.9, 3. , 5.1, 1.8]]),
 'target': array([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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]),
 'frame': None,
 'target_names': array(['setosa', 'versicolor', 'virginica'], dtype='<U1
0'),
 'DESCR': '.. _iris_dataset:\n\nIris plants dataset\n--------------------\n
\n**Data Set Characteristics:**\n\n    :Number of Instances: 150 (50 in eac
h of three classes)\n    :Number of Attributes: 4 numeric, predictive attri
butes and the class\n    :Attribute Information:\n        - sepal length in
cm\n        - sepal width in cm\n        - petal length in cm\n        - pe
tal width in cm\n        - class:\n                - Iris-Setosa\n
- Iris-Versicolour\n                - Iris-Virginica\n                \n
:Summary Statistics:\n\n    ============== ==== ==== ======= ===== ========
============\n                    Min  Max   Mean    SD   Class Correlation
\n    ============== ==== ==== ======= ===== ====================\n    sepa
l length:   4.3  7.9   5.84   0.83    0.7826\n    sepal width:    2.0  4.4
3.05   0.43   -0.4194\n    petal length:   1.0  6.9   3.76   1.76    0.9490
(high!)\n    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)\n
============== ==== ==== ======= ===== ====================\n\n    :Missing
Attribute Values: None\n    :Class Distribution: 33.3% for each of 3 classe
s.\n    :Creator: R.A. Fisher\n    :Donor: Michael Marshall (MARSHALL%PLU@i
o.arc.nasa.gov)\n    :Date: July, 1988\n\nThe famous Iris database, first u
sed by Sir R.A. Fisher. The dataset is taken\nfrom Fisher\'s paper. Note th
at it\'s the same as in R, but not as in the UCI\nMachine Learning Reposito
ry, which has two wrong data points.\n\nThis is perhaps the best known data
base to be found in the\npattern recognition literature.  Fisher\'s paper i
s a classic in the field and\nis referenced frequently to this day.  (See D
uda & Hart, for example.)  The\ndata set contains 3 classes of 50 instances
each, where each class refers to a\ntype of iris plant.  One class is linea
rly separable from the other 2; the\nlatter are NOT linearly separable from
each other.\n\n.. topic:: References\n\n   - Fisher, R.A. "The use of multi
ple measurements in taxonomic problems"\n     Annual Eugenics, 7, Part II,
179-188 (1936); also in "Contributions to\n     Mathematical Statistics" (J
ohn Wiley, NY, 1950).\n   - Duda, R.O., & Hart, P.E. (1973) Pattern Classif
ication and Scene Analysis.\n     (Q327.D83) John Wiley & Sons.  ISBN 0-471
-22361-1.  See page 218.\n   - Dasarathy, B.V. (1980) "Nosing Around the Ne
```

```
ighborhood: A New System\n     Structure and Classification Rule for Recogn
ition in Partially Exposed\n     Environments".  IEEE Transactions on Patte
rn Analysis and Machine\n     Intelligence, Vol. PAMI-2, No. 1, 67-71.\n
- Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transaction
s\n     on Information Theory, May 1972, 431-433.\n    - See also: 1988 MLC
Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II\n     conceptual cluste
ring system finds 3 classes in the data.\n    - Many, many more ...',
 'feature_names': ['sepal length (cm)',
  'sepal width (cm)',
  'petal length (cm)',
  'petal width (cm)'],
 'filename': 'iris.csv',
 'data_module': 'sklearn.datasets.data'}
```

In [88]:

```
import pandas as pd
iris_df=pd.DataFrame(data=iris.data,columns=iris.feature_names)
iris_df['target']=iris.target
iris_df['target_names']=iris.target_names[iris.target]
iris_df
```

Out[88]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	target_names
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5.0	3.6	1.4	0.2	0	setosa
...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2	virginica
146	6.3	2.5	5.0	1.9	2	virginica
147	6.5	3.0	5.2	2.0	2	virginica
148	6.2	3.4	5.4	2.3	2	virginica
149	5.9	3.0	5.1	1.8	2	virginica

150 rows × 6 columns

In [89]:

```
# seperate dependent variable and independent variable

x=iris.data
x
```

Out[89]:
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
       [5.2, 3.5, 1.5, 0.2],
       [5.2, 3.4, 1.4, 0.2],
       [4.7, 3.2, 1.6, 0.2],
       [4.8, 3.1, 1.6, 0.2],
       [5.4, 3.4, 1.5, 0.4],
       [5.2, 4.1, 1.5, 0.1],
       [5.5, 4.2, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.2],
       [5. , 3.2, 1.2, 0.2],
       [5.5, 3.5, 1.3, 0.2],
       [4.9, 3.6, 1.4, 0.1],
       [4.4, 3. , 1.3, 0.2],
       [5.1, 3.4, 1.5, 0.2],
       [5. , 3.5, 1.3, 0.3],
       [4.5, 2.3, 1.3, 0.3],
       [4.4, 3.2, 1.3, 0.2],
       [5. , 3.5, 1.6, 0.6],
       [5.1, 3.8, 1.9, 0.4],
       [4.8, 3. , 1.4, 0.3],
       [5.1, 3.8, 1.6, 0.2],
       [4.6, 3.2, 1.4, 0.2],
       [5.3, 3.7, 1.5, 0.2],
       [5. , 3.3, 1.4, 0.2],
       [7. , 3.2, 4.7, 1.4],
       [6.4, 3.2, 4.5, 1.5],
       [6.9, 3.1, 4.9, 1.5],
       [5.5, 2.3, 4. , 1.3],
       [6.5, 2.8, 4.6, 1.5],
       [5.7, 2.8, 4.5, 1.3],
       [6.3, 3.3, 4.7, 1.6],
       [4.9, 2.4, 3.3, 1. ],
       [6.6, 2.9, 4.6, 1.3],
       [5.2, 2.7, 3.9, 1.4],
       [5. , 2. , 3.5, 1. ],
       [5.9, 3. , 4.2, 1.5],
       [6. , 2.2, 4. , 1. ],
       [6.1, 2.9, 4.7, 1.4],
```

```
[5.6, 2.9, 3.6, 1.3],
[6.7, 3.1, 4.4, 1.4],
[5.6, 3. , 4.5, 1.5],
[5.8, 2.7, 4.1, 1. ],
[6.2, 2.2, 4.5, 1.5],
[5.6, 2.5, 3.9, 1.1],
[5.9, 3.2, 4.8, 1.8],
[6.1, 2.8, 4. , 1.3],
[6.3, 2.5, 4.9, 1.5],
[6.1, 2.8, 4.7, 1.2],
[6.4, 2.9, 4.3, 1.3],
[6.6, 3. , 4.4, 1.4],
[6.8, 2.8, 4.8, 1.4],
[6.7, 3. , 5. , 1.7],
[6. , 2.9, 4.5, 1.5],
[5.7, 2.6, 3.5, 1. ],
[5.5, 2.4, 3.8, 1.1],
[5.5, 2.4, 3.7, 1. ],
[5.8, 2.7, 3.9, 1.2],
[6. , 2.7, 5.1, 1.6],
[5.4, 3. , 4.5, 1.5],
[6. , 3.4, 4.5, 1.6],
[6.7, 3.1, 4.7, 1.5],
[6.3, 2.3, 4.4, 1.3],
[5.6, 3. , 4.1, 1.3],
[5.5, 2.5, 4. , 1.3],
[5.5, 2.6, 4.4, 1.2],
[6.1, 3. , 4.6, 1.4],
[5.8, 2.6, 4. , 1.2],
[5. , 2.3, 3.3, 1. ],
[5.6, 2.7, 4.2, 1.3],
[5.7, 3. , 4.2, 1.2],
[5.7, 2.9, 4.2, 1.3],
[6.2, 2.9, 4.3, 1.3],
[5.1, 2.5, 3. , 1.1],
[5.7, 2.8, 4.1, 1.3],
[6.3, 3.3, 6. , 2.5],
[5.8, 2.7, 5.1, 1.9],
[7.1, 3. , 5.9, 2.1],
[6.3, 2.9, 5.6, 1.8],
[6.5, 3. , 5.8, 2.2],
[7.6, 3. , 6.6, 2.1],
[4.9, 2.5, 4.5, 1.7],
[7.3, 2.9, 6.3, 1.8],
[6.7, 2.5, 5.8, 1.8],
[7.2, 3.6, 6.1, 2.5],
[6.5, 3.2, 5.1, 2. ],
[6.4, 2.7, 5.3, 1.9],
[6.8, 3. , 5.5, 2.1],
[5.7, 2.5, 5. , 2. ],
[5.8, 2.8, 5.1, 2.4],
[6.4, 3.2, 5.3, 2.3],
[6.5, 3. , 5.5, 1.8],
[7.7, 3.8, 6.7, 2.2],
[7.7, 2.6, 6.9, 2.3],
[6. , 2.2, 5. , 1.5],
[6.9, 3.2, 5.7, 2.3],
[5.6, 2.8, 4.9, 2. ],
[7.7, 2.8, 6.7, 2. ],
[6.3, 2.7, 4.9, 1.8],
[6.7, 3.3, 5.7, 2.1],
[7.2, 3.2, 6. , 1.8],
[6.2, 2.8, 4.8, 1.8],
[6.1, 3. , 4.9, 1.8],
```

```
       [6.4, 2.8, 5.6, 2.1],
       [7.2, 3. , 5.8, 1.6],
       [7.4, 2.8, 6.1, 1.9],
       [7.9, 3.8, 6.4, 2. ],
       [6.4, 2.8, 5.6, 2.2],
       [6.3, 2.8, 5.1, 1.5],
       [6.1, 2.6, 5.6, 1.4],
       [7.7, 3. , 6.1, 2.3],
       [6.3, 3.4, 5.6, 2.4],
       [6.4, 3.1, 5.5, 1.8],
       [6. , 3. , 4.8, 1.8],
       [6.9, 3.1, 5.4, 2.1],
       [6.7, 3.1, 5.6, 2.4],
       [6.9, 3.1, 5.1, 2.3],
       [5.8, 2.7, 5.1, 1.9],
       [6.8, 3.2, 5.9, 2.3],
       [6.7, 3.3, 5.7, 2.5],
       [6.7, 3. , 5.2, 2.3],
       [6.3, 2.5, 5. , 1.9],
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]])
```

In [90]:
```
y=iris.target
y
```

Out[90]:
```
array([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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

In [91]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_stat
```

In [92]:
```
x_train.shape
```

Out[92]:
```
(120, 4)
```

In [93]:
```
x_test.shape
```

Out[93]:
```
(30, 4)
```

In [94]:
```
clf=LogisticRegression()
clf
```

Out[94]:
```
▾ LogisticRegression
LogisticRegression()
```

In [95]:
```
# To train the algorithm

clf.fit(x_train,y_train)
```

Out[95]:
```
▾ LogisticRegression
LogisticRegression()
```

In [96]:
```
y_pred=clf.predict(x_test)
y_pred
```

Out[96]:
```
array([0, 0, 0, 2, 1, 2, 1, 1, 2, 0, 2, 0, 0, 2, 2, 1, 1, 1, 0, 2, 1, 0,
       1, 1, 1, 1, 1, 2, 0, 0])
```

In [97]:
```
accuracy=accuracy_score(y_test,y_pred)

accuracy
```

Out[97]:
```
1.0
```

In [98]:
```
import pandas as pd
df_1=pd.read_csv(r"C:\Users\HP\Desktop\d1.csv")
```

In [99]:
```
df_1
```

Out[99]:

	Gender	Height	Weight	Index
0	Male	174	96	4
1	Male	189	87	2
2	Female	185	110	4
3	Female	195	104	3
4	Male	149	61	3
...	...	...	...	...
495	Female	150	153	5
496	Female	184	121	4
497	Female	141	136	5
498	Male	150	95	5
499	Male	173	131	5

500 rows × 4 columns

In [100…
```
g=(df_1['Gender']).value_counts()
g
```

Out[100]:
```
Female    255
Male      245
Name: Gender, dtype: int64
```

In [101…
```
g.Male
```

Out[101]:
```
245
```

In [102…
```
g.Female
```

Out[102]:
```
255
```

In [103…
```
df_1['Gender']=df_1['Gender'].map({'Male':0,'Female':1})
```

In [104…
```
df_1
```

Out[104]:

	Gender	Height	Weight	Index
0	0	174	96	4
1	0	189	87	2
2	1	185	110	4
3	1	195	104	3
4	0	149	61	3
...	...	...	...	...
495	1	150	153	5
496	1	184	121	4
497	1	141	136	5
498	0	150	95	5
499	0	173	131	5

500 rows × 4 columns

In [105…

```
data=pd.get_dummies(df_1,columns=["Gender"],dtype=int,drop_first=True)
data
```

Out[105]:

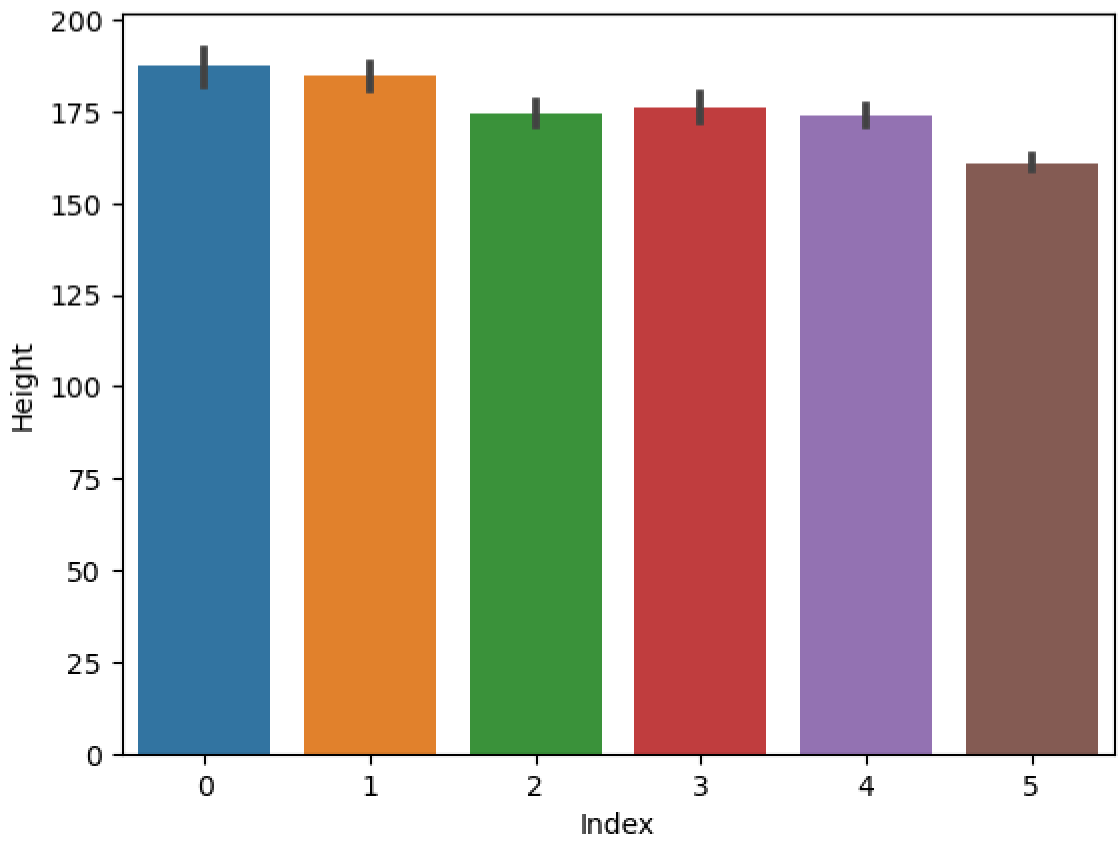
	Height	Weight	Index	Gender_1
0	174	96	4	0
1	189	87	2	0
2	185	110	4	1
3	195	104	3	1
4	149	61	3	0
...	...	...	...	...
495	150	153	5	1
496	184	121	4	1
497	141	136	5	1
498	150	95	5	0
499	173	131	5	0

500 rows × 4 columns

In [106…

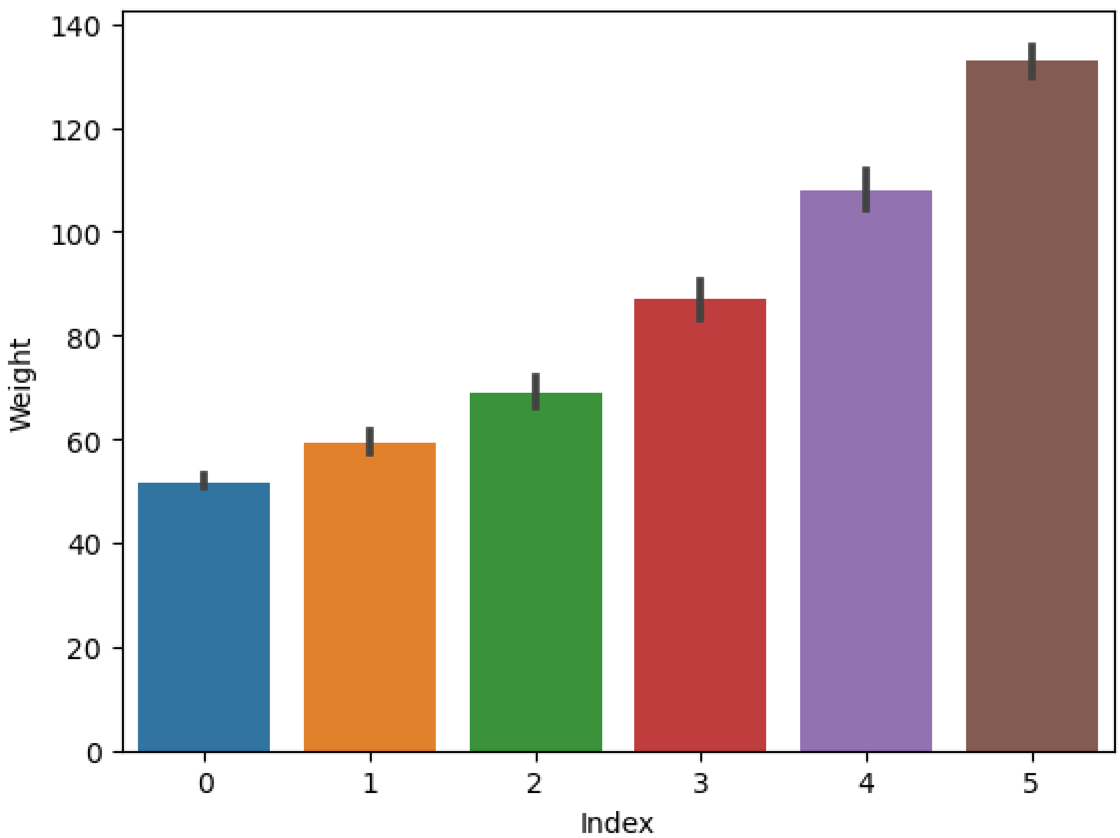
```
import seaborn as sns
#df=sns.load_dataset()
sns.barplot(y = 'Height',x ='Index',data=data)
```

Out[106]: <Axes: xlabel='Index', ylabel='Height'>



In [107…
```
sns.barplot(y = 'Weight',x ='Index',data=data)
```

Out[107]: <Axes: xlabel='Index', ylabel='Weight'>



In [108…
```
data
```

Out[108]:

	Height	Weight	Index	Gender_1
0	174	96	4	0
1	189	87	2	0
2	185	110	4	1
3	195	104	3	1
4	149	61	3	0
...	...	...	...	...
495	150	153	5	1
496	184	121	4	1
497	141	136	5	1
498	150	95	5	0
499	173	131	5	0

500 rows × 4 columns

In [109…

```
x=data.drop("Index",axis=1)
```

In [110…

```
x
```

Out[110]:

	Height	Weight	Gender_1
0	174	96	0
1	189	87	0
2	185	110	1
3	195	104	1
4	149	61	0
...	...	...	...
495	150	153	1
496	184	121	1
497	141	136	1
498	150	95	0
499	173	131	0

500 rows × 3 columns

In [111…

```
y=data['Index']
```

In [112…

```
y
```

Out[112]:

```
0      4
1      2
2      4
3      3
4      3
      ..
495    5
496    4
497    5
498    5
499    5
Name: Index, Length: 500, dtype: int64
```

In [ ]:

In [ ]:

In [113…
```
x.shape
```
Out[113]:
```
(500, 3)
```

In [114…
```
y.shape
```
Out[114]:
```
(500,)
```

In [ ]:

In [115…
```
x_train
```

```
Out[115]: array([[6.5, 3. , 5.8, 2.2],
       [5.5, 2.5, 4. , 1.3],
       [6.5, 3. , 5.5, 1.8],
       [5.8, 2.7, 3.9, 1.2],
       [6.8, 3. , 5.5, 2.1],
       [5.7, 2.8, 4.5, 1.3],
       [6.7, 3.1, 4.7, 1.5],
       [5.9, 3. , 4.2, 1.5],
       [5.6, 2.7, 4.2, 1.3],
       [7.7, 3. , 6.1, 2.3],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [4.7, 3.2, 1.6, 0.2],
       [6.7, 3. , 5. , 1.7],
       [5.6, 3. , 4.5, 1.5],
       [4.3, 3. , 1.1, 0.1],
       [7.1, 3. , 5.9, 2.1],
       [5.8, 2.7, 4.1, 1. ],
       [4.9, 3.1, 1.5, 0.2],
       [5.1, 2.5, 3. , 1.1],
       [5.6, 2.5, 3.9, 1.1],
       [5.1, 3.3, 1.7, 0.5],
       [5.8, 2.7, 5.1, 1.9],
       [5. , 3.6, 1.4, 0.2],
       [4.9, 2.4, 3.3, 1. ],
       [6.7, 2.5, 5.8, 1.8],
       [5.8, 2.6, 4. , 1.2],
       [4.9, 3.6, 1.4, 0.1],
       [5.1, 3.4, 1.5, 0.2],
       [6.1, 3. , 4.6, 1.4],
       [4.6, 3.4, 1.4, 0.3],
       [6.4, 3.2, 4.5, 1.5],
       [7.7, 2.6, 6.9, 2.3],
       [6.3, 3.4, 5.6, 2.4],
       [5.4, 3. , 4.5, 1.5],
       [5.8, 4. , 1.2, 0.2],
       [6. , 2.9, 4.5, 1.5],
       [4.6, 3.1, 1.5, 0.2],
       [5.8, 2.7, 5.1, 1.9],
       [6.9, 3.2, 5.7, 2.3],
       [6. , 3.4, 4.5, 1.6],
       [6.2, 3.4, 5.4, 2.3],
       [6.6, 2.9, 4.6, 1.3],
       [6.3, 3.3, 6. , 2.5],
       [4.7, 3.2, 1.3, 0.2],
       [4.8, 3. , 1.4, 0.3],
       [4.9, 3.1, 1.5, 0.1],
       [6.5, 2.8, 4.6, 1.5],
       [4.6, 3.2, 1.4, 0.2],
       [5.1, 3.8, 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
       [7.4, 2.8, 6.1, 1.9],
       [5.2, 3.5, 1.5, 0.2],
       [5.4, 3.4, 1.7, 0.2],
       [6. , 3. , 4.8, 1.8],
       [6.2, 2.8, 4.8, 1.8],
       [4.8, 3.1, 1.6, 0.2],
       [5. , 3.2, 1.2, 0.2],
       [7.2, 3.2, 6. , 1.8],
       [7.2, 3.6, 6.1, 2.5],
       [5.7, 2.5, 5. , 2. ],
       [4.8, 3.4, 1.9, 0.2],
       [5.7, 2.6, 3.5, 1. ],
       [6.8, 3.2, 5.9, 2.3],
```

```
       [5.1, 3.5, 1.4, 0.3],
       [4.8, 3. , 1.4, 0.1],
       [6. , 2.2, 5. , 1.5],
       [6.4, 2.8, 5.6, 2.1],
       [5.7, 4.4, 1.5, 0.4],
       [6.1, 2.8, 4. , 1.3],
       [5.7, 3.8, 1.7, 0.3],
       [4.9, 2.5, 4.5, 1.7],
       [7.7, 3.8, 6.7, 2.2],
       [4.4, 3. , 1.3, 0.2],
       [6.3, 2.9, 5.6, 1.8],
       [6.3, 3.3, 4.7, 1.6],
       [6.9, 3.1, 4.9, 1.5],
       [6.7, 3.3, 5.7, 2.1],
       [5. , 3.4, 1.5, 0.2],
       [6.9, 3.1, 5.4, 2.1],
       [5.2, 3.4, 1.4, 0.2],
       [5.7, 2.8, 4.1, 1.3],
       [6.3, 2.8, 5.1, 1.5],
       [5.5, 3.5, 1.3, 0.2],
       [6. , 2.2, 4. , 1. ],
       [4.4, 2.9, 1.4, 0.2],
       [6.7, 3.1, 5.6, 2.4],
       [6.1, 2.8, 4.7, 1.2],
       [7.6, 3. , 6.6, 2.1],
       [5.1, 3.5, 1.4, 0.2],
       [6.7, 3.3, 5.7, 2.5],
       [7.3, 2.9, 6.3, 1.8],
       [5.9, 3. , 5.1, 1.8],
       [6.8, 2.8, 4.8, 1.4],
       [5.4, 3.7, 1.5, 0.2],
       [5.1, 3.8, 1.5, 0.3],
       [5.6, 2.8, 4.9, 2. ],
       [6.3, 2.5, 4.9, 1.5],
       [5.1, 3.8, 1.9, 0.4],
       [5.2, 2.7, 3.9, 1.4],
       [7.9, 3.8, 6.4, 2. ],
       [6.4, 3.2, 5.3, 2.3],
       [6.4, 2.7, 5.3, 1.9],
       [6. , 2.7, 5.1, 1.6],
       [5. , 3.3, 1.4, 0.2],
       [6.9, 3.1, 5.1, 2.3],
       [5.4, 3.9, 1.7, 0.4],
       [6.5, 3.2, 5.1, 2. ],
       [5. , 2. , 3.5, 1. ],
       [6.7, 3. , 5.2, 2.3],
       [6.4, 2.8, 5.6, 2.2],
       [5. , 3.5, 1.3, 0.3],
       [6.4, 3.1, 5.5, 1.8],
       [6.6, 3. , 4.4, 1.4],
       [6.3, 2.3, 4.4, 1.3],
       [6.1, 2.9, 4.7, 1.4],
       [5.9, 3.2, 4.8, 1.8],
       [5.5, 2.4, 3.7, 1. ],
       [4.8, 3.4, 1.6, 0.2],
       [5.7, 3. , 4.2, 1.2]])
```

In [116…

```
x_test
```

Out[116]:
```
array([[5.5, 4.2, 1.4, 0.2],
       [5.4, 3.9, 1.3, 0.4],
       [5. , 3.5, 1.6, 0.6],
       [7.2, 3. , 5.8, 1.6],
       [7. , 3.2, 4.7, 1.4],
       [6.3, 2.7, 4.9, 1.8],
       [6.2, 2.2, 4.5, 1.5],
       [5.5, 2.3, 4. , 1.3],
       [6.3, 2.5, 5. , 1.9],
       [4.9, 3. , 1.4, 0.2],
       [6.5, 3. , 5.2, 2. ],
       [5.2, 4.1, 1.5, 0.1],
       [5.4, 3.4, 1.5, 0.4],
       [7.7, 2.8, 6.7, 2. ],
       [6.1, 3. , 4.9, 1.8],
       [6.4, 2.9, 4.3, 1.3],
       [5.6, 3. , 4.1, 1.3],
       [5.7, 2.9, 4.2, 1.3],
       [4.4, 3.2, 1.3, 0.2],
       [6.1, 2.6, 5.6, 1.4],
       [5.5, 2.4, 3.8, 1.1],
       [5.3, 3.7, 1.5, 0.2],
       [5.5, 2.6, 4.4, 1.2],
       [6.7, 3.1, 4.4, 1.4],
       [6.2, 2.9, 4.3, 1.3],
       [5.6, 2.9, 3.6, 1.3],
       [5. , 2.3, 3.3, 1. ],
       [5.8, 2.8, 5.1, 2.4],
       [5. , 3. , 1.6, 0.2],
       [4.5, 2.3, 1.3, 0.3]])
```

In [117…
```
y_train
```

Out[117]:
```
array([2, 1, 2, 1, 2, 1, 1, 1, 1, 2, 0, 0, 0, 1, 1, 0, 2, 1, 0, 1, 1, 0,
       2, 0, 1, 2, 1, 0, 0, 1, 0, 1, 2, 2, 1, 0, 1, 0, 2, 2, 1, 2, 1, 2,
       0, 0, 0, 1, 0, 0, 0, 2, 0, 0, 2, 2, 0, 0, 2, 2, 2, 0, 1, 2, 0, 0,
       2, 2, 0, 1, 0, 2, 2, 0, 2, 1, 1, 2, 0, 2, 0, 1, 2, 0, 1, 0, 2, 1,
       2, 0, 2, 2, 2, 1, 0, 0, 2, 1, 0, 1, 2, 2, 2, 1, 0, 2, 0, 2, 1, 2,
       2, 0, 2, 1, 1, 1, 1, 1, 0, 1])
```

In [118…
```
y_test
```

Out[118]:
```
array([0, 0, 0, 2, 1, 2, 1, 1, 2, 0, 2, 0, 0, 2, 2, 1, 1, 1, 0, 2, 1, 0,
       1, 1, 1, 1, 1, 2, 0, 0])
```

In [119…
```
y_test.shape
```

Out[119]:
```
(30,)
```

In [120…
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_stat
```

In [121…
```
from sklearn.linear_model import LogisticRegression
log_model=LogisticRegression()
```

In [122…
```
log_model
```

Out[122]:
```
▾ LogisticRegression
LogisticRegression()
```

In [123…

```python
#To train the DataSet
log_model.fit(x_train,y_train)
```

```
C:\Users\HP\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:4
58: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
ession
  n_iter_i = _check_optimize_result(
```

Out[123]:

```
▾ LogisticRegression
LogisticRegression()
```

In [124…

```python
pred=log_model.predict(x_test)
```

In [125…

```python
pred
```

Out[125]:

```
array([5, 4, 5, 2, 3, 3, 1, 4, 5, 4, 5, 3, 5, 3, 5, 2, 5, 5, 4, 5, 4, 5,
       4, 5, 2, 4, 3, 4, 5, 2, 5, 4, 4, 5, 4, 5, 0, 2, 5, 4, 3, 5, 4, 5,
       5, 5, 4, 2, 1, 3, 5, 5, 5, 4, 2, 2, 2, 5, 4, 5, 3, 3, 5, 5, 3, 5,
       4, 4, 4, 5, 5, 4, 5, 5, 1, 4, 3, 3, 5, 2, 2, 2, 5, 3, 5, 5, 5, 5,
       5, 2, 3, 5, 2, 4, 4, 0, 4, 5, 5, 5, 2, 4, 4, 5, 2, 3, 5, 5, 1, 1,
       5, 4, 5, 3, 5, 5, 5, 2, 4, 4, 5, 4, 4, 5, 4, 5, 5, 1, 4, 2, 5, 1,
       5, 4, 3, 5, 5, 5, 3, 5, 5, 4, 3, 5, 1, 5, 2, 4, 5, 5], dtype=int64)
```

In [126…

```python
from sklearn.metrics import accuracy_score

accuracy_score(y_test,pred)
```

Out[126]:

```
0.7666666666666667
```

In [127…

```python
# Linear Regression

# Data:

#X(Week)                                                     Y(Sales in Thousand)
#-----------------------------------------------------------------------------
#   1                                                                 1.2
#   2                                                                 1.8
#   3                                                                 2.5
#   4                                                                 3.2
#   5                                                                 3.8

# Linear Regression formula -->y=a0+a1*x

# a1 -->((meanof(x*y)) - (meanof(x)*meanof(y)))/meanof(x^2)-(meanof(x)^2)

# a0 -->mean(y) - a1*meanof(x)

#                  x            y            x^2                 x*y
#-----------------------------------------------------------------------------
#                  1            1.2          1                   1.2
#                  2            1.8          4                   3.6
#                  3            2.5          9                   7.5
#                  4            3.2          16                  12.8
#                  5            3.8          25                  19.0
```

```
#sum:                  15           12.5         55                 44.1
#Average               3            2.5          11                 8.82


#substitute in a1  -->>8.82-7.5/11-9  -->>1.32/2  -->>0.66

#a0 -->>2.5-0.66*3  -->>0.52


#Sales of 3rd week

#y -->>a0+(a1*x)

#y=0.54+(0.66*3)

#y=2.52

#The sales of the 7th week

#y=0.54+(0.66*7)

#y=5.17
```

In [128...

```
df_3=pd.read_csv(r"C:\Users\HP\Desktop\salary.csv")
```

In [129...

```
df_3
```

Out[129]:

	Unnamed: 0	YearsExperience	Salary
0	0	1.2	39344
1	1	1.4	46206
2	2	1.6	37732
3	3	2.1	43526
4	4	2.3	39892
5	5	3.0	56643
6	6	3.1	60151
7	7	3.3	54446
8	8	3.3	64446
9	9	3.8	57190
10	10	4.0	63219
11	11	4.1	55795
12	12	4.1	56958
13	13	4.2	57082
14	14	4.6	61112
15	15	5.0	67939
16	16	5.2	66030
17	17	5.4	83089
18	18	6.0	81364
19	19	6.1	93941
20	20	6.9	91739
21	21	7.2	98274
22	22	8.0	101303
23	23	8.3	113813
24	24	8.8	109432
25	25	9.1	105583
26	26	9.6	116970
27	27	9.7	112636
28	28	10.4	122392
29	29	10.6	121873

In [130…

```
df_3.shape
```

Out[130]:

```
(30, 3)
```

In [131…

```
df_3.isnull().sum()                #df_3.isnull().sum()
```

Out[131]:

```
Unnamed: 0         0
YearsExperience    0
Salary             0
dtype: int64
```

In [132…

```
df_3.isna().sum()
```

Out[132]:

```
Unnamed: 0         0
YearsExperience    0
Salary             0
dtype: int64
```

In [133…
```
x=df_3[['YearsExperience']]
```

In [134…
```
y=df_3[['Salary']]
```

In [135…
```
from sklearn.model_selection import train_test_split
```

In [136…
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.4,random_stat
x_train,x_test,y_train,y_test
```

Out[136]:
```
(    YearsExperience
 1               1.4
 19              6.1
 22              8.0
 12              4.1
 5               3.0
 14              4.6
 0               1.2
 21              7.2
 4               2.3
 8               3.3
 13              4.2
 9               3.8
 15              5.0
 29             10.6
 23              8.3
 6               3.1
 17              5.4
 11              4.1,
     YearsExperience
 20              6.9
 24              8.8
 7               3.3
 18              6.0
 2               1.6
 27              9.7
 26              9.6
 16              5.2
 25              9.1
 28             10.4
 10              4.0
 3               2.1,
     Salary
 1    46206
 19   93941
 22  101303
 12   56958
 5    56643
 14   61112
 0    39344
 21   98274
 4    39892
 8    64446
 13   57082
 9    57190
 15   67939
 29  121873
 23  113813
 6    60151
 17   83089
 11   55795,
     Salary
 20   91739
 24  109432
 7    54446
 18   81364
 2    37732
 27  112636
 26  116970
 16   66030
 25  105583
 28  122392
 10   63219
 3    43526)
```

In [273...
```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model
```
Out[273]:
```
▾ LinearRegression
LinearRegression()
```
In [138...
```
model.fit(x_test,y_test)
```
Out[138]:
```
▾ LinearRegression
LinearRegression()
```
In [139...
```
y_pred=model.predict(x_test)
y_pred
```
Out[139]:
```
array([[ 88594.87757335],
       [106682.10850323],
       [ 54324.33475883],
       [ 80027.24186972],
       [ 38141.02287419],
       [115249.74420686],
       [114297.78468424],
       [ 72411.56568871],
       [109537.98707111],
       [121913.46086524],
       [ 60988.05141721],
       [ 42900.82048732]])
```
In [140...
```
y_test
```
Out[140]:

	Salary
20	91739
24	109432
7	54446
18	81364
2	37732
27	112636
26	116970
16	66030
25	105583
28	122392
10	63219
3	43526

In [141...
```
import numpy as np
from sklearn.metrics import accuracy_score
acc=accuracy_score(y_test,np.round(y_pred))
acc
```
Out[141]:
```
0.0
```
In [142...
```
inputdata=[[18]]
prediction=model.predict(inputdata)
```

```
prediction
```

```
C:\Users\HP\anaconda3\lib\site-packages\sklearn\base.py:420: UserWarning: X
does not have valid feature names, but LinearRegression was fitted with fea
ture names
  warnings.warn(
```

Out[142]:

```
array([[194262.38458478]])
```

In [143...

```
from sklearn.metrics import mean_squared_error
mse=mean_squared_error(y_test,y_pred)
```

In [144...

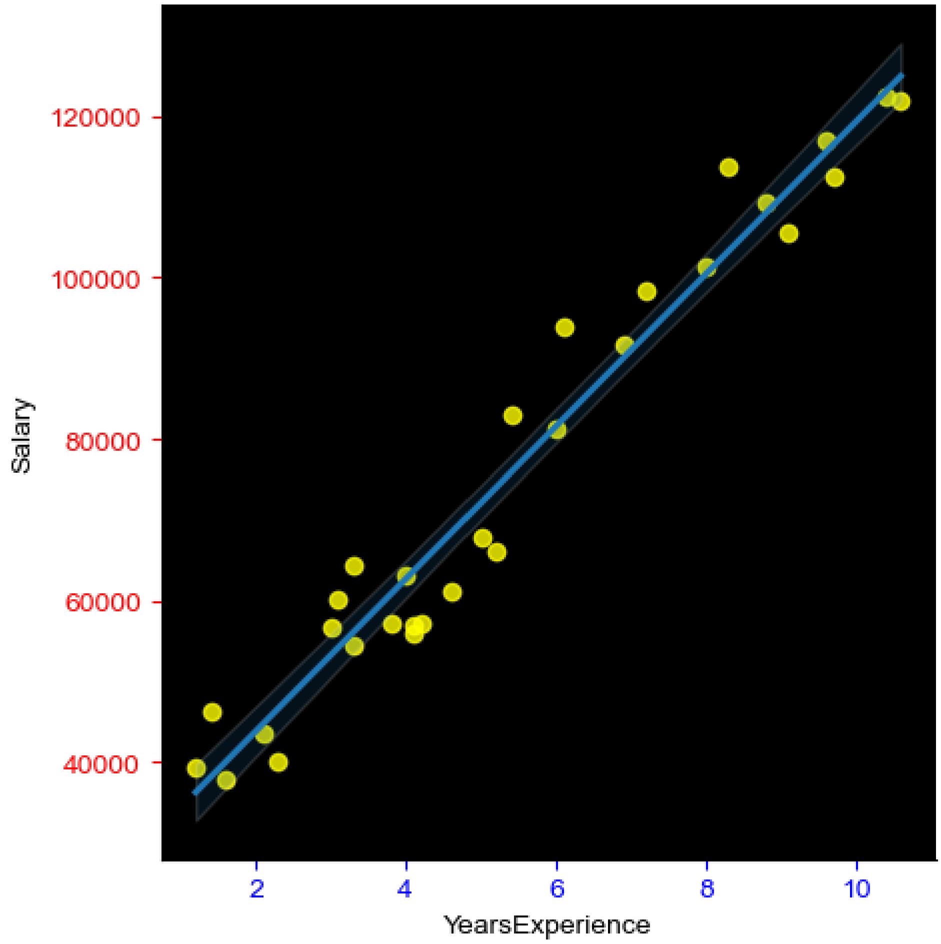
```
mse
```

Out[144]:

```
7946009.255793135
```

In [145...

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.lmplot(x='YearsExperience',y='Salary',data=df_3,scatter_kws={"color":"ye
ax=plt.gca()
sns.set_style("darkgrid")
plt.gca().set_facecolor('black')
ax.tick_params(axis='x',colors='blue')
ax.tick_params(axis='y',colors='red')
```



In [334...

```
#Project 3

import pandas as pd
df_4=pd.read_csv(r"C:\Users\HP\Desktop\student12.csv")
```

In [147...

```
df_4
```

Out[147]:

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
0	7	99	Yes	9	1	91
1	4	82	No	4	2	65
2	8	51	Yes	7	2	45
3	5	52	Yes	5	2	36
4	7	75	No	8	5	66
...	...	...	...	...	...	...
9995	1	49	Yes	4	2	23
9996	7	64	Yes	8	5	58
9997	6	83	Yes	8	5	74
9998	9	97	Yes	7	0	95
9999	7	74	No	8	1	64

10000 rows × 6 columns

In [148…

```
df_4.head()
```

Out[148]:

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
0	7	99	Yes	9	1	91
1	4	82	No	4	2	65
2	8	51	Yes	7	2	45
3	5	52	Yes	5	2	36
4	7	75	No	8	5	66

In [149…

```
df_4.shape
```

Out[149]:

```
(10000, 6)
```

In [150…

```
df_4.describe()
```

Out[150]:

	Hours Studied	Previous Scores	Sleep Hours	Sample Question Papers Practiced	Performance Index
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	4.992900	69.445700	6.530600	4.583300	55.224800
std	2.589309	17.343152	1.695863	2.867348	19.212558
min	1.000000	40.000000	4.000000	0.000000	10.000000
25%	3.000000	54.000000	5.000000	2.000000	40.000000
50%	5.000000	69.000000	7.000000	5.000000	55.000000
75%	7.000000	85.000000	8.000000	7.000000	71.000000
max	9.000000	99.000000	9.000000	9.000000	100.000000

In [151…

```
df_4.isna().sum()
```

Out[151]:

```
Hours Studied                       0
Previous Scores                     0
Extracurricular Activities          0
Sleep Hours                         0
Sample Question Papers Practiced    0
Performance Index                   0
dtype: int64
```

In [152…
```
# To check duplicate values

duplicate_rows=df_4.duplicated()
```

In [153…
```
df_4[duplicate_rows]
```

Out[153]:

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
915	9	52	No	5	9	48
1477	7	61	Yes	6	8	54
1601	5	99	No	7	5	89
1786	2	62	Yes	9	4	40
2026	5	87	Yes	6	7	74
...	...	...	...	...	...	...
9644	4	91	Yes	4	3	71
9940	8	95	No	5	2	90
9954	6	97	No	8	7	92
9966	1	41	No	7	3	12
9985	8	99	No	5	5	92

127 rows × 6 columns

In [154…
```
duplicate_rows.sum()
```

Out[154]:
```
127
```

In [155…
```
print("Before dropping duplicates:",df_4.shape)

df_4.drop_duplicates(inplace=True)

print("After dropping duplicates:",df_4.shape)
```

```
Before dropping duplicates: (10000, 6)
After dropping duplicates: (9873, 6)
```

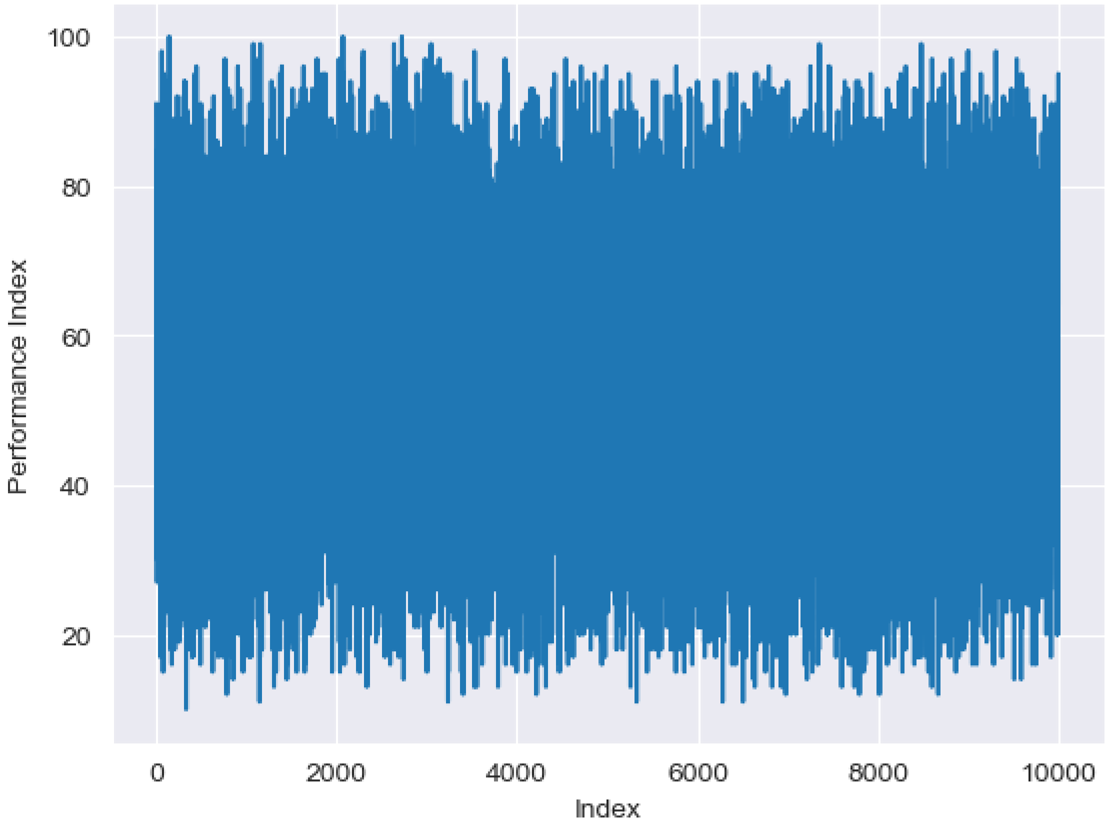
In [156…
```
#Based on index value try to check the performance

response=df_4["Performance Index"]
```

In [157…
```
response.dtype
```

Out[157]:
```
dtype('int64')
```

In [158…
```
import matplotlib.pyplot as plt
plt.plot(response.index,response)
plt.xlabel('Index')
plt.ylabel('Performance Index')
```

Out[158]:
```
Text(0, 0.5, 'Performance Index')
```

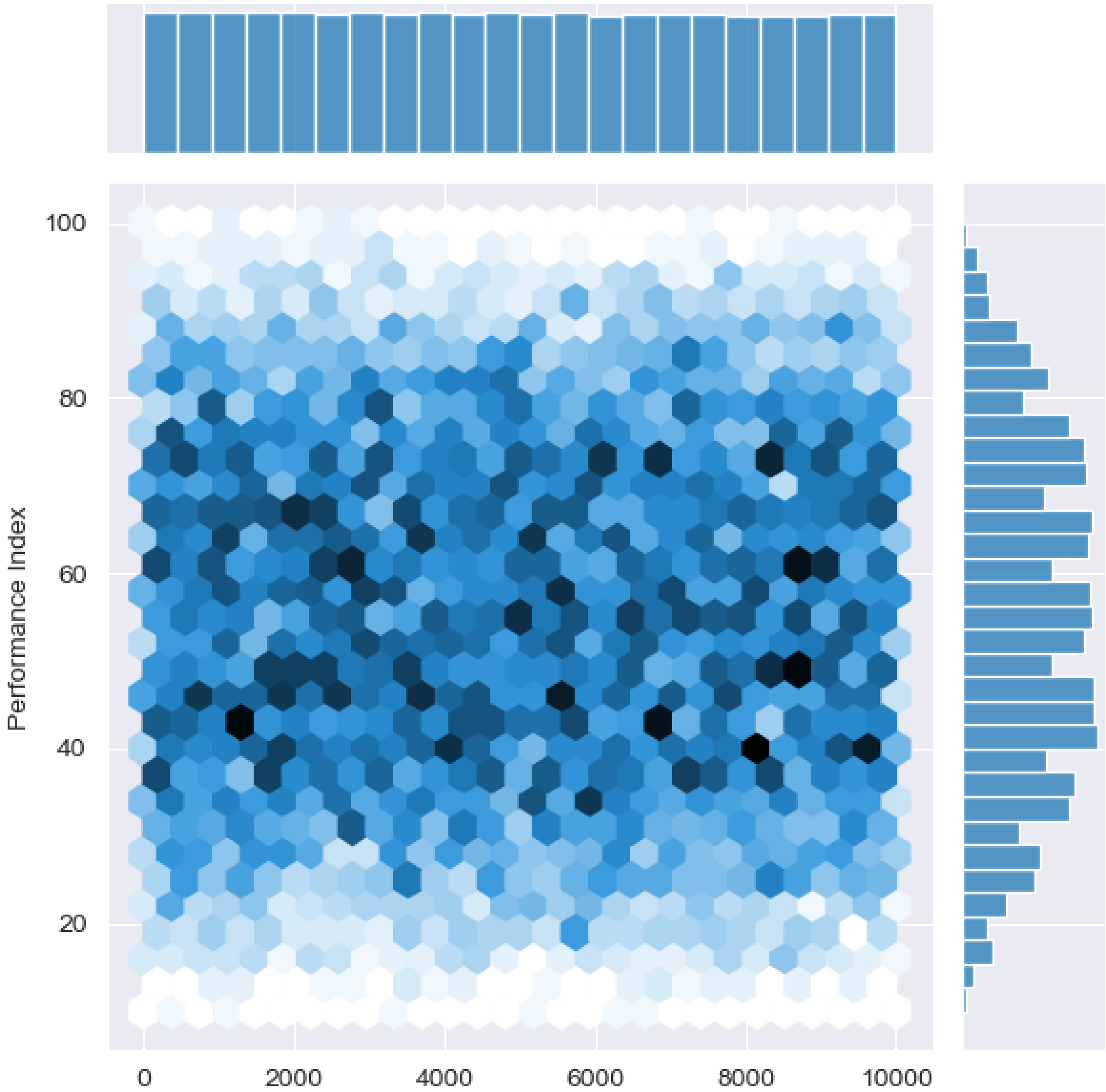


In [159…

```
sns.jointplot(x=response.index,y='Performance Index',data=df_4,kind='hex')
```

Out[159]:

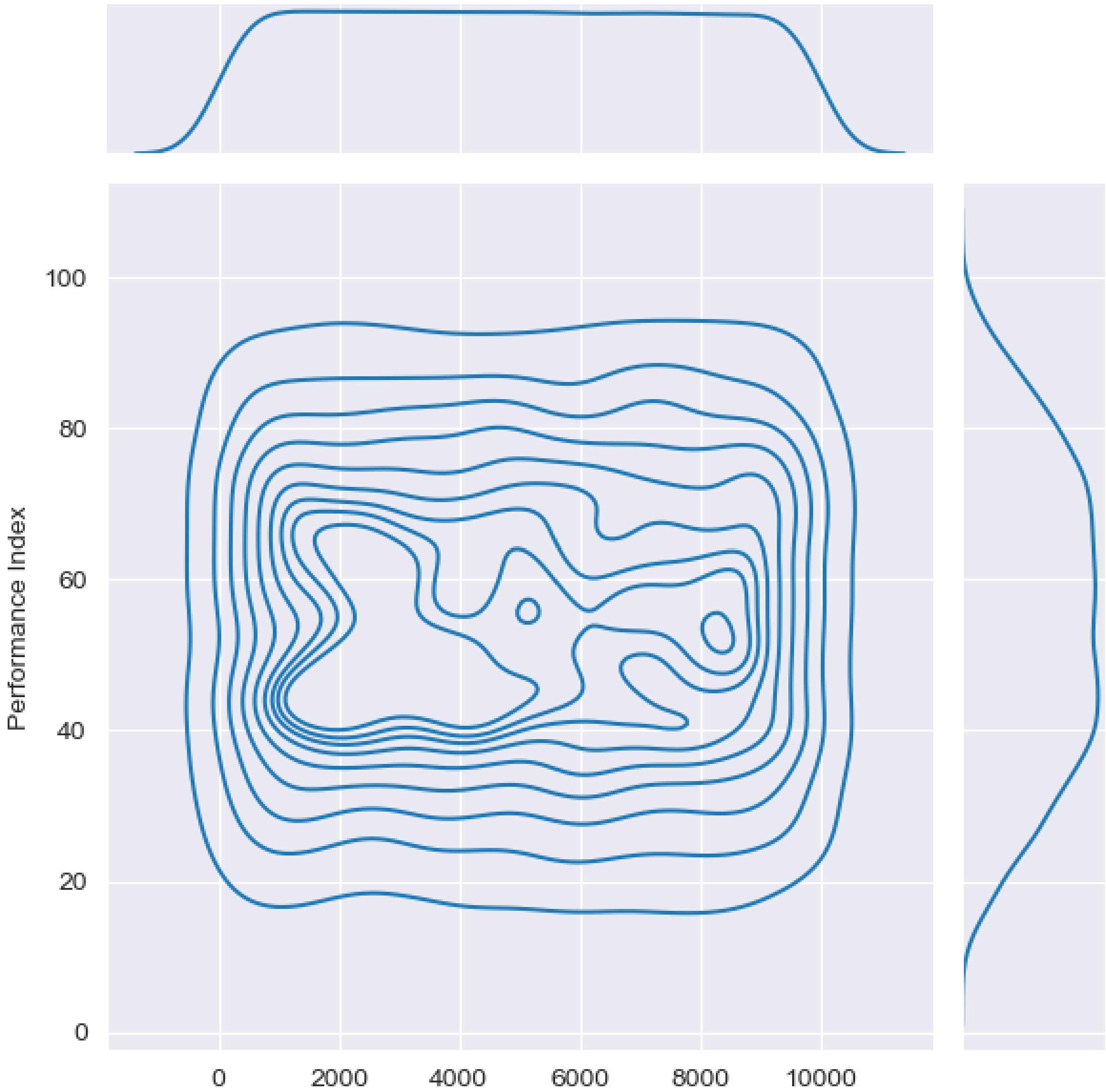
```
<seaborn.axisgrid.JointGrid at 0x1ef9264bf40>
```



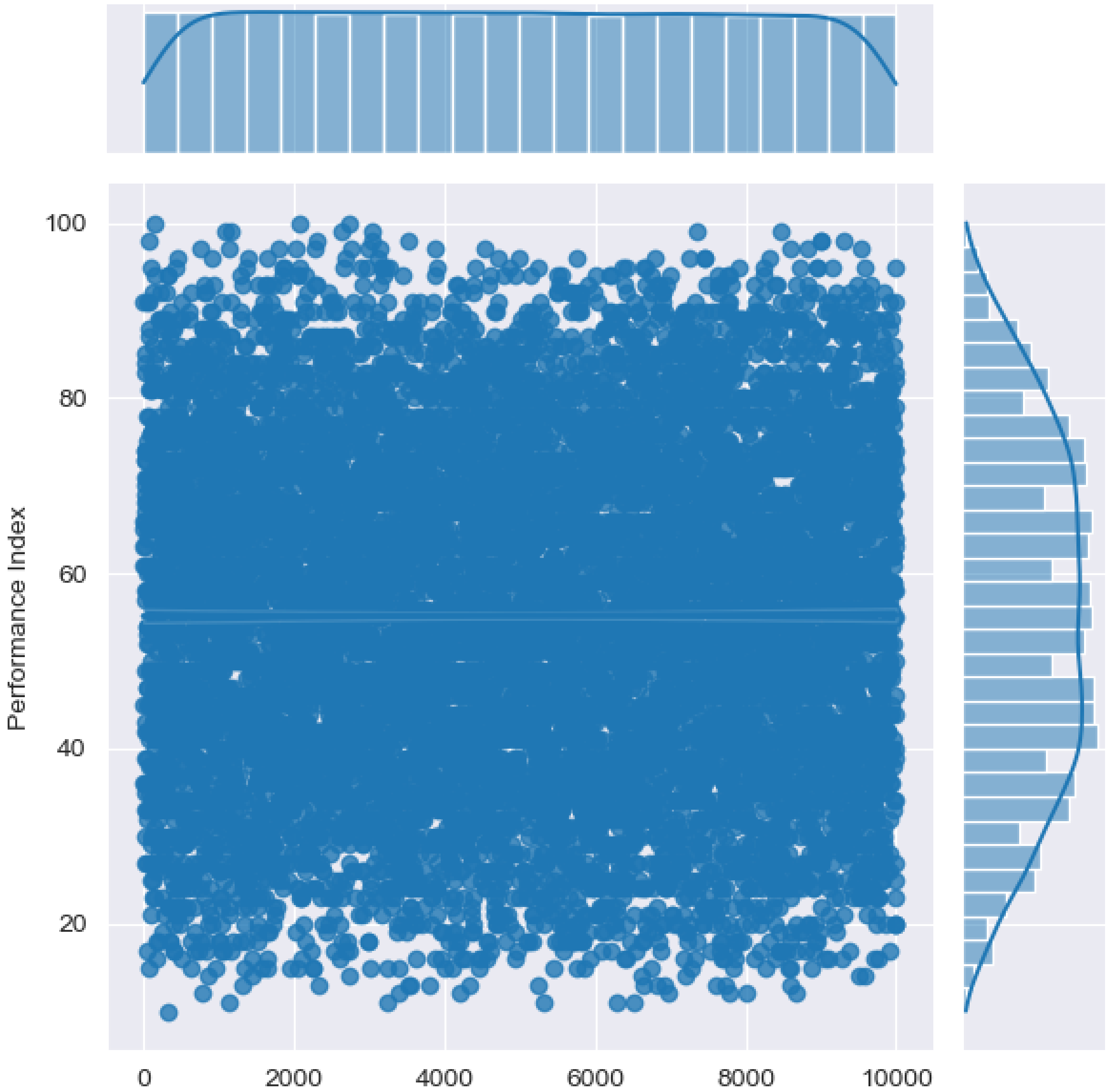
In [160…

```
sns.jointplot(x=response.index,y='Performance Index',data=df_4,kind='kde')
```

Out[160]: <seaborn.axisgrid.JointGrid at 0x1efa541bf40>

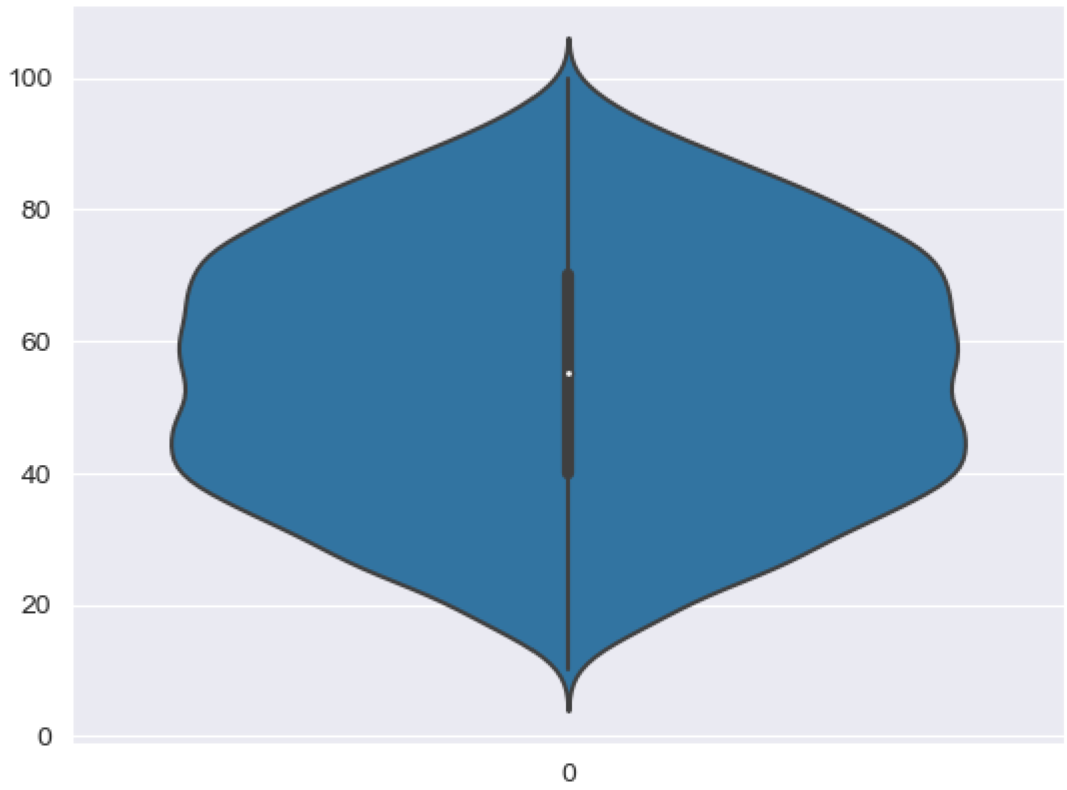



```
In [161…  sns.jointplot(x=response.index,y='Performance Index',data=df_4,kind='reg')
Out[161]: <seaborn.axisgrid.JointGrid at 0x1efa694a4d0>
```



In [162…
```
sns.violinplot(response)
```

Out[162]: <Axes: >

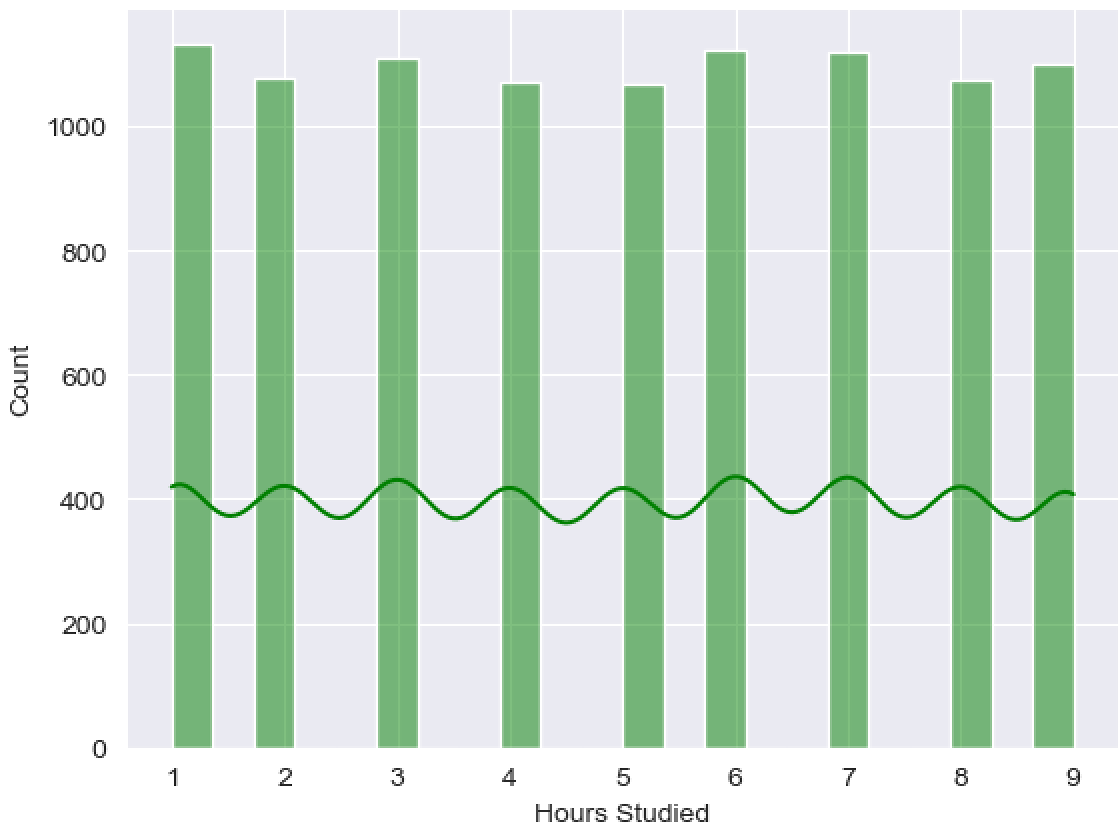


In [163…
```
a6=(df_4['Performance Index']).min()
a6
```
Out[163]:
```
10
```
In [164…
```
(df_4['Performance Index']==a6).sum()
```
Out[164]:
```
1
```
In [165…
```
b6=(df_4['Performance Index']).max()
b6
```
Out[165]:
```
100
```
In [166…
```
(df_4['Performance Index']==b6).sum()
```
Out[166]:
```
3
```
In [172…
```
#To get all the uniquee values

df_4['Hours Studied'].unique()
```
Out[172]:
```
array([7, 4, 8, 5, 3, 6, 2, 1, 9], dtype=int64)
```
In [173…
```
df_4['Hours Studied'].value_counts()
```

Out[173]:
```
1    1133
6    1122
7    1118
3    1110
9    1099
2    1077
8    1074
4    1071
5    1069
Name: Hours Studied, dtype: int64
```

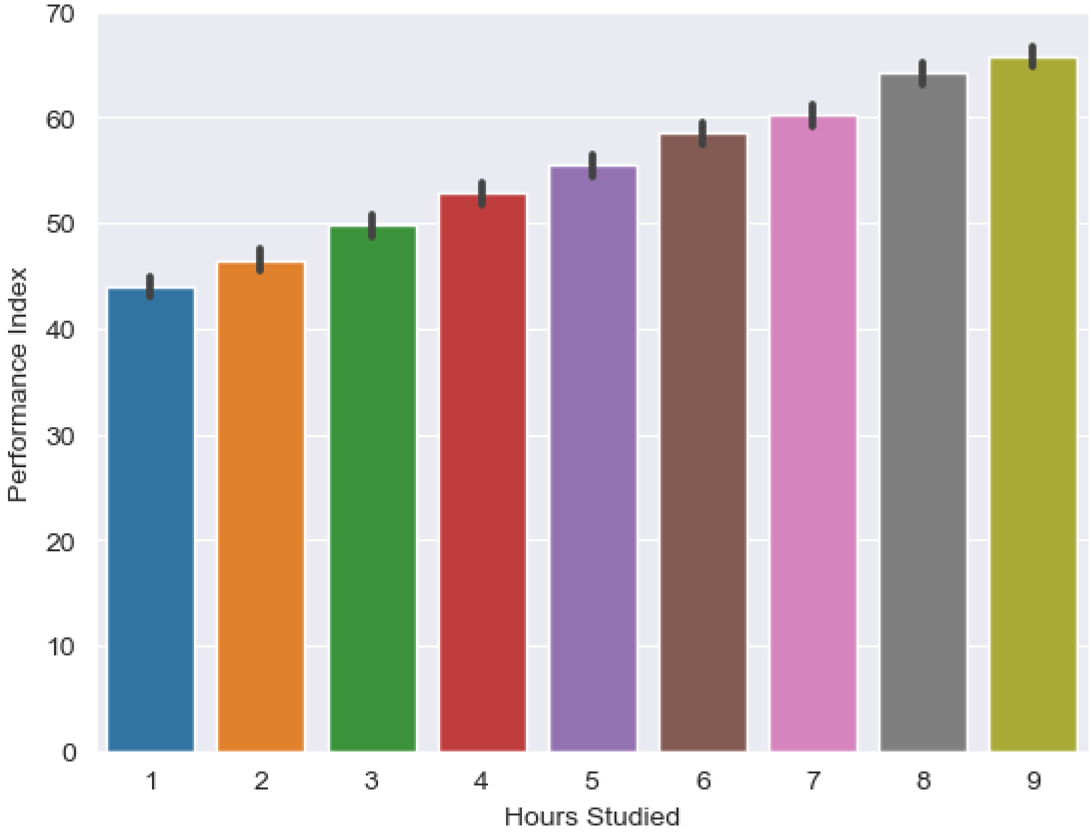
In [183…
```
# TO know how many students studied in each hour

x=df_4['Hours Studied']

sns.histplot(x,color='green',kde=True,)
```

Out[183]: <Axes: xlabel='Hours Studied', ylabel='Count'>



In [196…
```
sns.barplot(x='Hours Studied',y='Performance Index',data=df_4)
```

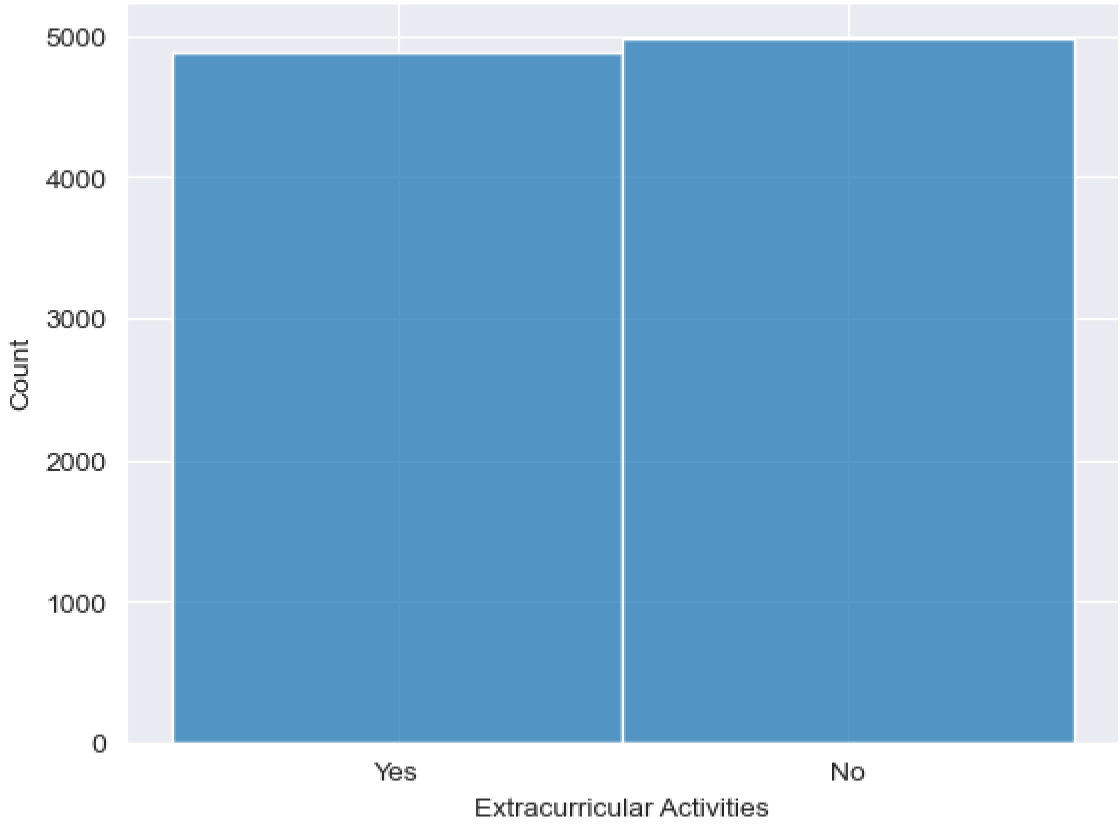
Out[196]: <Axes: xlabel='Hours Studied', ylabel='Performance Index'>



In [215…
```
a9=(df_4['Extracurricular Activities']=='Yes').sum()
a9
```
Out[215]:
```
4887
```

In [214…
```
b9=(df_4['Extracurricular Activities']=='No').sum()
b9
```
Out[214]:
```
4986
```

In [251…
```
sns.histplot(df_4['Extracurricular Activities'])
```
Out[251]:
```
<Axes: xlabel='Extracurricular Activities', ylabel='Count'>
```



In [241…

```
df_4.columns
```

Out[241]:

```
Index(['Hours Studied', 'Previous Scores', 'Extracurricular Activities',
       'Sleep Hours', 'Sample Question Papers Practiced', 'Performance Inde
x'],
      dtype='object')
```

In [ ]:

```
df_4=replace.
```

In [320…

```
y=df_4['Performance Index']
```

In [321…

```
y
```

Out[321]:

```
0       91
1       65
2       45
3       36
4       66
        ..
9995    23
9996    58
9997    74
9998    95
9999    64
Name: Performance Index, Length: 9873, dtype: int64
```

In [346…

```
x=df_4[['Hours Studied', 'Previous Scores', 'Extracurricular Activities','Sl

x
```

Out[346]:

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced
0	7	99	1	9	1
1	4	82	0	4	2
2	8	51	1	7	2
3	5	52	1	5	2
4	7	75	0	8	5
...	...	...	...	...	...
9995	1	49	1	4	2
9996	7	64	1	8	5
9997	6	83	1	8	5
9998	9	97	1	7	0
9999	7	74	0	8	1

10000 rows × 5 columns

In [347...

```
df_4['Extracurricular Activities']=df_4['Extracurricular Activities'].map({'
#df_4['Extracurricular Activities']=df_4['Extracurricular Activities'].apply
```

In [348...

```
df_4
```

Out[348]:

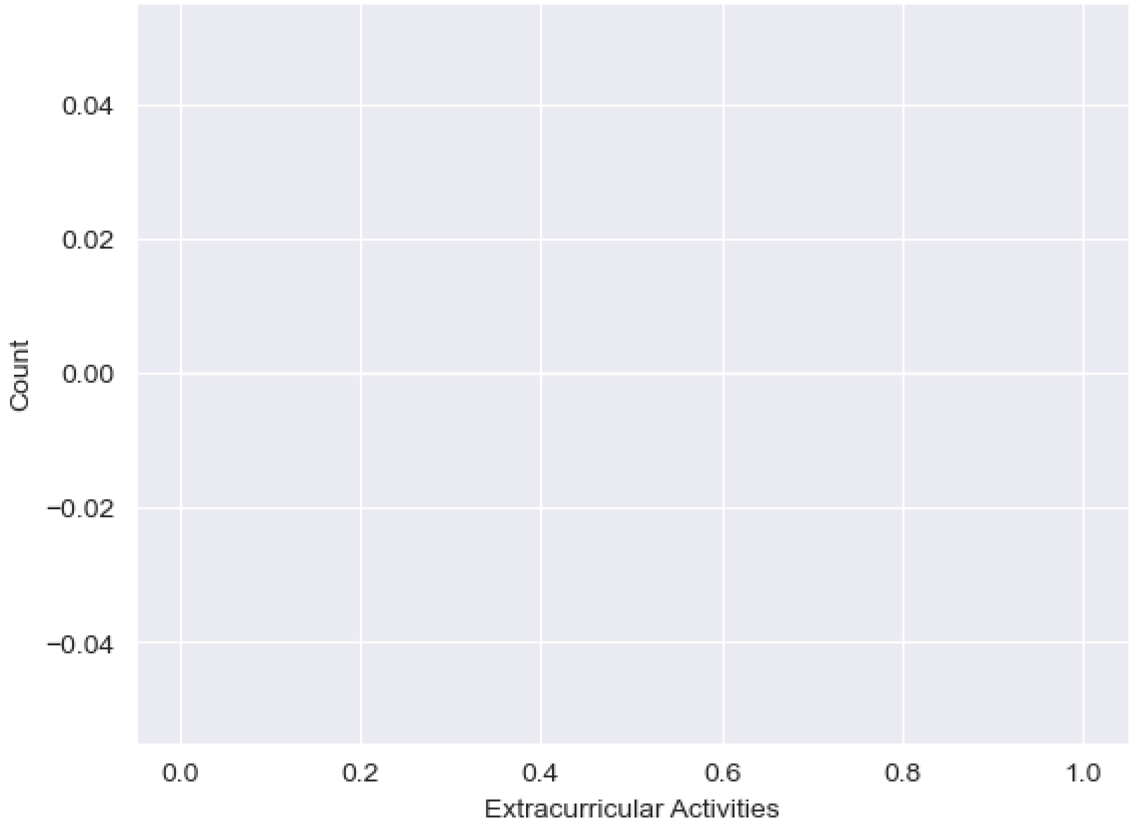
	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
0	7	99	NaN	9	1	91
1	4	82	NaN	4	2	65
2	8	51	NaN	7	2	45
3	5	52	NaN	5	2	36
4	7	75	NaN	8	5	66
...	...	...	...	...	...	...
9995	1	49	NaN	4	2	23
9996	7	64	NaN	8	5	58
9997	6	83	NaN	8	5	74
9998	9	97	NaN	7	0	95
9999	7	74	NaN	8	1	64

10000 rows × 6 columns

In [349...

```
sns.histplot(df_4['Extracurricular Activities'])
```

Out[349]: <Axes: xlabel='Extracurricular Activities', ylabel='Count'>



In [350…
```
df_4.head()
```

Out[350]:

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
0	7	99	NaN	9	1	91
1	4	82	NaN	4	2	65
2	8	51	NaN	7	2	45
3	5	52	NaN	5	2	36
4	7	75	NaN	8	5	66

In [383…
```
y=df_4['Performance Index']
```

In [384…
```
y
```

Out[384]:
```
0       91
1       65
2       45
3       36
4       66
        ..
9995    23
9996    58
9997    74
9998    95
9999    64
Name: Performance Index, Length: 10000, dtype: int64
```

In [385…
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.4,random_stat
from sklearn.linear_model import LinearRegression
model=LinearRegression()
```

In [386…
```
model
```

Out[386]:

```
▾ LinearRegression
LinearRegression()
```

In [387…

```
model.fit(x_train,y_train)
```

Out[387]:

```
▾ LinearRegression
LinearRegression()
```

In [392…

```
y_pred=model.predict(x_test)
```

In [393…

```
y_pred
```

Out[393]:

```
array([44.39182655, 96.13564305, 30.57978946, ..., 31.23011643,
       67.75874112, 28.10993149])
```

In [394…

```
from sklearn.metrics import accuracy_score

acc=accuracy_score(y_test,np.round(y_pred))

acc
```

Out[394]:

```
0.2015
```

In [395…

```
data=[[8,85,1,6,6]]
prediction=model.predict(data)
prediction
```

```
C:\Users\HP\anaconda3\lib\site-packages\sklearn\base.py:420: UserWarning: X
does not have valid feature names, but LinearRegression was fitted with fea
ture names
  warnings.warn(
```

Out[395]:

```
array([79.94425089])
```

In [396…

```
# Instead of linear regression
# Ridge
from sklearn.linear_model import Ridge
clf=Ridge()
clf.fit(x_train,y_train)
```

Out[396]:

```
▾ Ridge
Ridge()
```

In [397…

```
y_pred=clf.predict(x_test)
```

In [398…

```
y_pred
```

Out[398]:

```
array([44.39197551, 96.13511347, 30.5798681 , ..., 31.23037878,
       67.75871088, 28.11000109])
```

In [401…

```
clf.score(x_test,y_test)
```

Out[401]:

```
0.9888235045787498
```

In [402…

```
clf.score(x_train,y_train)
```

Out[402]: 0.9886995936043511

In [403…
```
df_5=pd.read_csv(r"C:\Users\HP\Desktop\fra.csv")
```

In [404…
```
df_5
```

Out[404]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	B
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.
2	1	48	1.0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.
3	0	61	3.0	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.
4	0	46	3.0	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23
...	...	...	...	...	...	...	...	...	...	...	...	...	
4233	1	50	1.0	1	1.0	0.0	0	1	0	313.0	179.0	92.0	25.
4234	1	51	3.0	1	43.0	0.0	0	0	0	207.0	126.5	80.0	19
4235	0	48	2.0	1	20.0	NaN	0	0	0	248.0	131.0	72.0	22.
4236	0	44	1.0	1	15.0	0.0	0	0	0	210.0	126.5	87.0	19
4237	0	52	2.0	0	0.0	0.0	0	0	0	269.0	133.5	83.0	21.

4238 rows × 16 columns

In [405…
```
df_5.columns
```

Out[405]:
```
Index(['male', 'age', 'education', 'currentSmoker', 'cigsPerDay', 'BPMeds',
       'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP',
       'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
      dtype='object')
```

In [413…
```
x=[['male', 'age', 'education', 'currentSmoker', 'cigsPerDay', 'BPMeds',
       'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP',
       'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD']]
x
```

Out[413]:
```
[['male',
  'age',
  'education',
  'currentSmoker',
  'cigsPerDay',
  'BPMeds',
  'prevalentStroke',
  'prevalentHyp',
  'diabetes',
  'totChol',
  'sysBP',
  'diaBP',
  'BMI',
  'heartRate',
  'glucose',
  'TenYearCHD']]
```

In [423…
```
df_5.isna()
```

Out[423]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	B
0	False	False	False	False	False	False	False	False	False	False	False	False	F
1	False	False	False	False	False	False	False	False	False	False	False	False	F
2	False	False	False	False	False	False	False	False	False	False	False	False	F
3	False	False	False	False	False	False	False	False	False	False	False	False	F
4	False	False	False	False	False	False	False	False	False	False	False	False	F
...	...	...	...	...	...	...	...	...	...	...	...	...	
4233	False	False	False	False	False	False	False	False	False	False	False	False	F
4234	False	False	False	False	False	False	False	False	False	False	False	False	F
4235	False	False	False	False	False	True	False	False	False	False	False	False	F
4236	False	False	False	False	False	False	False	False	False	False	False	False	F
4237	False	False	False	False	False	False	False	False	False	False	False	False	F

4238 rows × 16 columns

In [424…

```
df_5.fillna('zero')
```

Out[424]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	B
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.
2	1	48	1.0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.
3	0	61	3.0	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.
4	0	46	3.0	1	23.0	0.0	0	0	0	285.0	130.0	84.0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	
4233	1	50	1.0	1	1.0	0.0	0	1	0	313.0	179.0	92.0	25.
4234	1	51	3.0	1	43.0	0.0	0	0	0	207.0	126.5	80.0	19
4235	0	48	2.0	1	20.0	zero	0	0	0	248.0	131.0	72.0	2
4236	0	44	1.0	1	15.0	0.0	0	0	0	210.0	126.5	87.0	19
4237	0	52	2.0	0	0.0	0.0	0	0	0	269.0	133.5	83.0	21.

4238 rows × 16 columns

In [429…

```
(df_5['age']).max()
```

Out[429]: 70

In [430…

```
(df_5['age']).min()
```

Out[430]: 32

In [440…

```
len((df_5[df_5['currentSmoker']==1]))
```

Out[440]: 2094

In [ ]:

In [ ]: