```
!pip install numpy
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.23.5)
```

```
!pip install pandas
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.4)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.23.5)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas)
```

```
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.15.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.5.2
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tens
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (16.0.6)
Requirement already satisfied: ml-dtypes~=0.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.23.5
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from ten
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.60.1)
Requirement already satisfied: tensorboard<2.16,>=2.15 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.1
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tenso
Requirement already satisfied: keras<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->te
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorboard
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15-
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from te
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15-
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->ten
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oau
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensor
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tenso
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0
```

```
!pip install scipy
```

```
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (1.11.4)
Requirement already satisfied: numpy<1.28.0,>=1.21.6 in /usr/local/lib/python3.10/dist-packages (from scipy) (1.23.5)
```

```
!pip install matplotlib
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.47.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.23.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (23.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotli
```

```
!pip install seaborn
```

```
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.13.1)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.10/dist-packages (from seaborn) (1.23.5)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dist-packages (from seaborn) (1.5.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.10/dist-packages (from seaborn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->se
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4-
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->s
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seaborn) (2023
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotli
```

```
!pip install -U scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Collecting scikit-learn
  Downloading scikit_learn-1.4.0-1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.1/12.1 MB 47.6 MB/s eta 0:00:00
Requirement already satisfied: numpy<2.0,>=1.19.5 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.23.5
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.2.
Installing collected packages: scikit-learn
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 1.2.2
    Uninstalling scikit-learn-1.2.2:
      Successfully uninstalled scikit-learn-1.2.2
Successfully installed scikit-learn-1.4.0
```

```
import pandas as pd


indian_names_dict = {
    'Name': ['Aarav', 'Aanya', 'Arjun', 'Divya', 'Rohan', 'Isha', 'Kabir', 'Anika'],
    'Age': [25, 22, 28, 24, 30, 26, 29, 23],
    'City': ['Mumbai', 'Delhi', 'Bangalore', 'Chennai', 'Kolkata', 'Hyderabad', 'Ahmedabad', 'Pune']
}


df = pd.DataFrame(indian_names_dict)


print(df)
```

```
     Name  Age       City
0  Aarav   25     Mumbai
1  Aanya   22      Delhi
2  Arjun   28  Bangalore
3  Divya   24    Chennai
4  Rohan   30    Kolkata
5   Isha   26  Hyderabad
6  Kabir   29  Ahmedabad
7  Anika   23       Pune
```

Start coding or generate with AI.

```
import pandas as pd


df = pd.read_csv('/content/demo.csv')
df
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75
...	...	...	...	...	...	...	...	...
995	female	group E	master's degree	standard	completed	88	99	95
996	male	group C	high school	free/reduced	none	62	55	55
997	female	group C	high school	free/reduced	completed	59	71	65
998	female	group D	some college	standard	completed	68	78	77
999	female	group D	some college	free/reduced	none	77	86	86

1000 rows × 8 columns

```
# Check for missing values using pandas isnull()
missing_values = df.isnull().sum()
missing_values
```

```
gender                       0
race/ethnicity               0
parental level of education  0
lunch                        0
test preparation course      0
math score                   0
reading score                0
writing score                0
dtype: int64
```

```
# Describe function to get initial statistics
initial_statistics = df.describe()
initial_statistics
```

	math score	reading score	writing score
count	1000.00000	1000.000000	1000.000000
mean	66.08900	69.169000	68.054000
std	15.16308	14.600192	15.195657
min	0.00000	17.000000	10.000000
25%	57.00000	59.000000	57.750000
50%	66.00000	70.000000	69.000000
75%	77.00000	79.000000	79.000000
max	100.00000	100.000000	100.000000

```
# Variable descriptions and types
variable_descriptions = df.dtypes

print("\nVariable Descriptions and Types:")
print(variable_descriptions)
```

```
Variable Descriptions and Types:
gender                       object
race/ethnicity               object
parental level of education  object
lunch                        object
test preparation course      object
math score                    int64
reading score                 int64
```

```
writing score                  int64
dtype: object
```

```
dataframe_dimensions = df.shape
print("\nData Frame Dimensions:")
print(dataframe_dimensions)
```

```

Data Frame Dimensions:
(1000, 8)
```

```
from sklearn.preprocessing import LabelEncoder
```

```
label_encoder = LabelEncoder()
df['gender'] = label_encoder.fit_transform(df['gender'])
print("\nModified DataFrame:")
print(df)
```

```

Modified DataFrame:
     gender race/ethnicity parental level of education         lunch  \
0         0        group B           bachelor's degree      standard
1         0        group C                some college      standard
2         0        group B             master's degree      standard
3         1        group A          associate's degree  free/reduced
4         1        group C                some college      standard
..      ...            ...                         ...           ...
995       0        group E             master's degree      standard
996       1        group C                 high school  free/reduced
997       0        group C                 high school  free/reduced
998       0        group D                some college      standard
999       0        group D                some college  free/reduced

    test preparation course  math score  reading score  writing score
0                      none          72             72             74
1                 completed          69             90             88
2                      none          90             95             93
3                      none          47             57             44
4                      none          76             78             75
..                      ...         ...            ...            ...
995               completed          88             99             95
996                    none          62             55             55
997               completed          59             71             65
998               completed          68             78             77
999                    none          77             86             86

[1000 rows x 8 columns]
```

```
Start coding or generate with AI.
```

Data Science Practical 3

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
```

```
df = pd.read_csv("xAPI-Edu-Data.csv")
df.head(10)
```

	gender	NationalITy	PlaceofBirth	StageID	GradeID	SectionID	Topic	Semester	Relation	raisedhands	VisITedResources	Annc
0	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	15	16	
1	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	20	20	
2	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	10	7	
3	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	30	25	
4	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	40	50	
5	F	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	42	30	
6	M	KW	KuwaIT	MiddleSchool	G-07	A	Math	F	Father	35	12	
7	M	KW	KuwaIT	MiddleSchool	G-07	A	Math	F	Father	50	10	
8	F	KW	KuwaIT	MiddleSchool	G-07	A	Math	F	Father	12	21	
9	F	KW	KuwaIT	MiddleSchool	G-07	B	IT	F	Father	70	80	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 480 entries, 0 to 479
Data columns (total 17 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   gender                    480 non-null    object
 1   NationalITy               480 non-null    object
 2   PlaceofBirth              480 non-null    object
 3   StageID                   480 non-null    object
 4   GradeID                   480 non-null    object
 5   SectionID                 480 non-null    object
 6   Topic                     480 non-null    object
 7   Semester                  480 non-null    object
 8   Relation                  480 non-null    object
 9   raisedhands               480 non-null    int64
 10  VisITedResources          480 non-null    int64
 11  AnnouncementsView         480 non-null    int64
 12  Discussion                480 non-null    int64
 13  ParentAnsweringSurvey     480 non-null    object
 14  ParentschoolSatisfaction  480 non-null    object
 15  StudentAbsenceDays        480 non-null    object
 16  Class                     480 non-null    object
dtypes: int64(4), object(13)
memory usage: 63.9+ KB
```

```
df.isnull().sum()
```

```
gender                      0
NationalITy                 0
PlaceofBirth                0
StageID                     0
GradeID                     0
SectionID                   0
Topic                       0
Semester                    0
Relation                    0
raisedhands                 0
VisITedResources            0
AnnouncementsView           0
Discussion                  0
ParentAnsweringSurvey       0
ParentschoolSatisfaction    0
StudentAbsenceDays          0
Class                       0
dtype: int64
```

```
le = LabelEncoder()
```

```
df['gender'] = le.fit_transform(df['gender'])
```

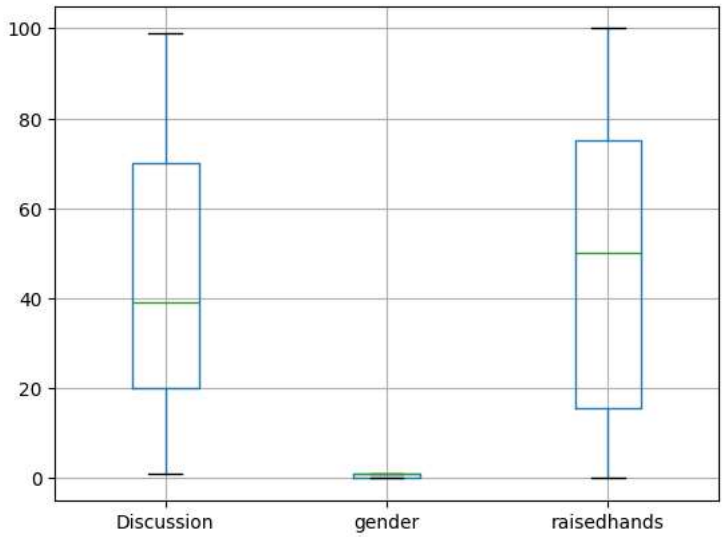
```
df
```

	gender	NationalITy	PlaceofBirth	StageID	GradeID	SectionID	Topic	Semester	Relation	raisedhands	VisITedResources
0	1	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	15	16
1	1	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	20	20
2	1	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	10	7
3	1	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	30	25
4	1	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	40	50
...	...	...	...	...	...	...	...	...	...	...	...
475	0	Jordan	Jordan	MiddleSchool	G-08	A	Chemistry	S	Father	5	4
476	0	Jordan	Jordan	MiddleSchool	G-08	A	Geology	F	Father	50	77
477	0	Jordan	Jordan	MiddleSchool	G-08	A	Geology	S	Father	55	74
478	0	Jordan	Jordan	MiddleSchool	G-08	A	History	F	Father	30	17
479	0	Jordan	Jordan	MiddleSchool	G-08	A	History	S	Father	35	14

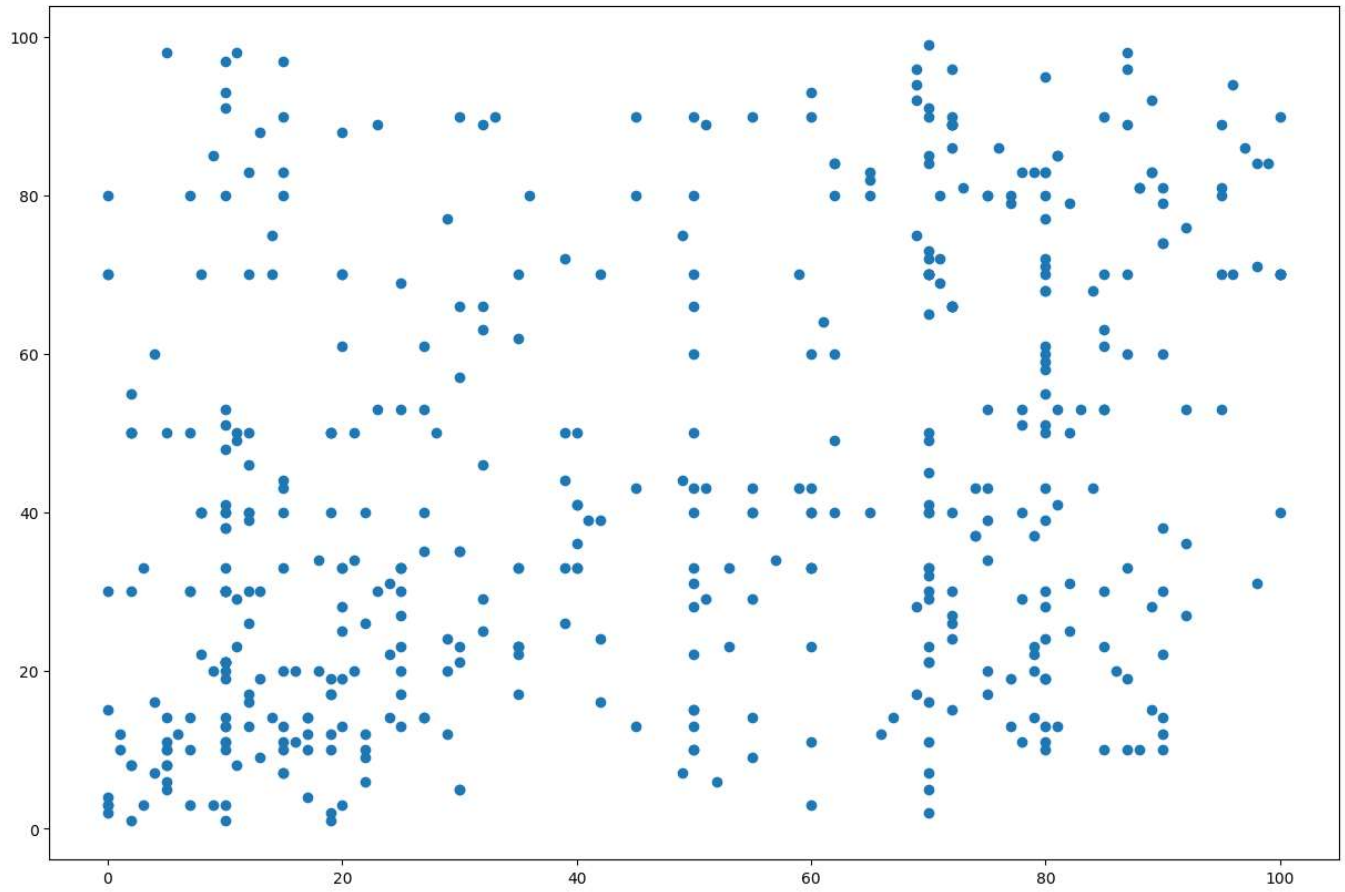
480 rows × 17 columns

```
# Using boxplot
col = ['Discussion','gender','raisedhands']
df.boxplot(col)

print(np.where(df['Discussion']>90))

print(np.where(df['raisedhands']>90))
```

```
(array([ 18,  19, 155, 180, 218, 240, 247, 252, 258, 372, 373, 378, 380,
       381, 386, 433, 463, 465], dtype=int64),)
(array([ 95, 138, 139, 146, 149, 152, 192, 196, 218, 239, 271, 274, 277,
       282, 283, 296, 308, 368, 404, 416, 419, 432], dtype=int64),)
```




```
# Using scatterplot
fig, ax=plt.subplots(figsize=(15,10))
ax.scatter(df['raisedhands'],df['Discussion'])
plt.show()
```



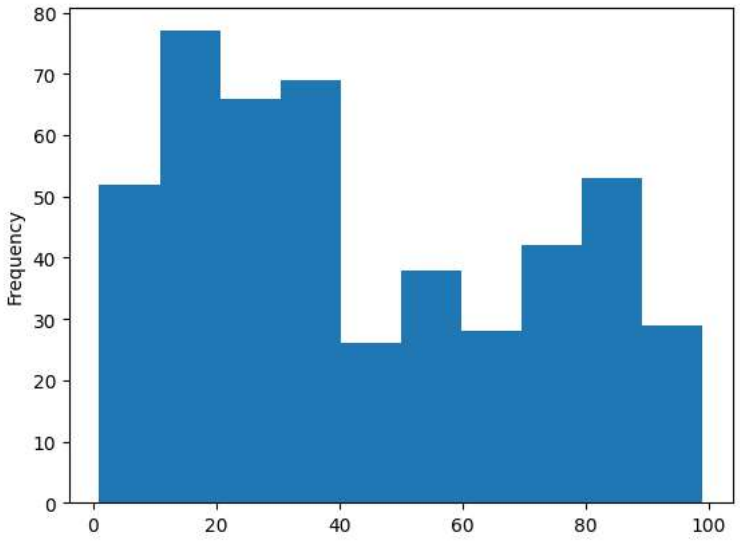

```
print(np.where((df['raisedhands']<50)  &  (df['Discussion']>1)))
```

```
(array([  0,   1,   2,   3,   4,   5,   6,   8,  11,  12,  13,  15,  16,
         21,  22,  23,  24,  25,  26,  27,  29,  30,  31,  32,  33,  34,
         35,  36,  37,  38,  40,  41,  42,  43,  44,  45,  46,  48,  49,
         50,  51,  53,  54,  55,  57,  58,  59,  60,  63,  64,  65,  66,
         69,  70,  71,  72,  73,  74,  76,  77,  78,  80,  81,  82,  83,
         85,  87,  88,  89,  90,  96,  97,  98, 102, 103, 104, 105, 106,
        108, 112, 113, 114, 115, 117, 118, 120, 121, 124, 125, 126, 128,
        130, 132, 133, 140, 141, 142, 144, 147, 151, 153, 158, 169, 170,
        172, 173, 175, 177, 179, 183, 184, 190, 191, 195, 198, 199, 200,
        201, 202, 203, 204, 207, 208, 210, 211, 213, 214, 215, 216, 226,
        227, 229, 231, 232, 233, 234, 235, 242, 243, 250, 251, 253, 255,
        260, 262, 263, 266, 267, 268, 269, 272, 273, 284, 285, 290, 291,
        298, 299, 300, 301, 302, 303, 304, 305, 310, 311, 322, 323, 324,
        325, 326, 327, 330, 331, 332, 333, 334, 335, 340, 341, 342, 343,
        344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356,
        357, 360, 361, 366, 367, 370, 371, 372, 373, 374, 375, 376, 377,
        378, 379, 380, 381, 386, 387, 388, 389, 400, 401, 407, 415, 428,
        429, 450, 451, 452, 453, 454, 455, 468, 469, 474, 475, 478, 479],
       dtype=int64),)
```

Plotting histogram

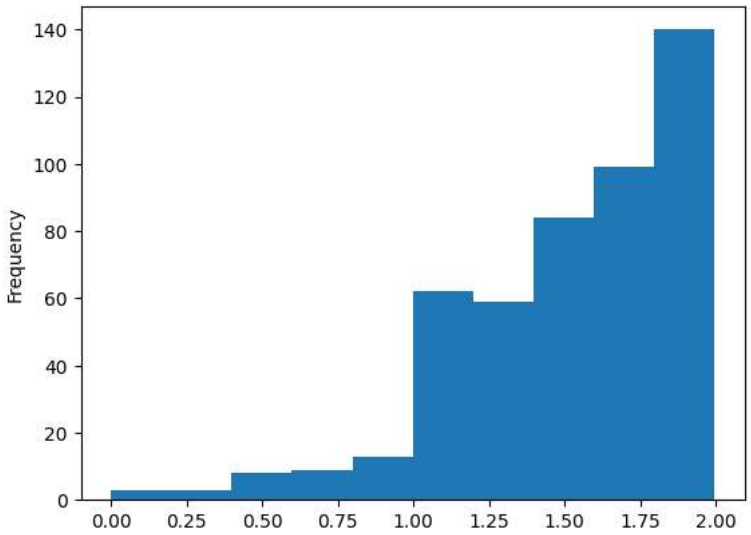
```
df['Discussion'].plot(kind='hist')
```

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




```
df['log_dis'] = np.log10(df['Discussion'])
df['log_dis'].plot(kind = 'hist')
```

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



```
import pandas as pd
```

Problem Statement -2 - Iris.csv

```
df=pd.read_csv("/content/Iris.csv")
df
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

```
features=['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']
for i in features:
  a=df[i].mean()
  print("The mean of",i ,"is",a)

for i in features:
  a=df[i].std()
  print("The standar deviation of",i ,"is",a)

for i in features:
  a=df[i].median()
  print("The median  of",i ,"is",a)
```

```
The mean of SepalLengthCm is SepalLengthCm    5.843333
dtype: float64
The mean of SepalWidthCm is SepalWidthCm    3.054
dtype: float64
The mean of PetalLengthCm is PetalLengthCm    3.758667
dtype: float64
The mean of PetalWidthCm is PetalWidthCm    1.198667
dtype: float64
The standar deviation of SepalLengthCm is 0.828066127977863
The standar deviation of SepalWidthCm is 0.4335943113621737
The standar deviation of PetalLengthCm is 1.7644204199522626
The standar deviation of PetalWidthCm is 0.7631607417008411
The median  of SepalLengthCm is 5.8
The median  of SepalWidthCm is 3.0
The median  of PetalLengthCm is 4.35
The median  of PetalWidthCm is 1.3
```

```
df['SepalLengthCm'].mean()
```

```
5.843333333333334
```

```
ad=df.describe()
ad
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

```
df.quantile()
```

```
<ipython-input-133-431199b824ed>:1: FutureWarning: The default value of numeric_only in DataFrame.quantile is dep
  df.quantile()
Id               75.50
SepalLengthCm     5.80
SepalWidthCm      3.00
PetalLengthCm     4.35
PetalWidthCm      1.30
Name: 0.5, dtype: float64
```

```
setosa_data = df[df['Species'] == 'Iris-setosa']
versicolor_data = df[df['Species'] == 'Iris-versicolor']
virginica_data = df[df['Species'] == 'Iris-virginica']
```

```
def display_statistics(data, species_name):
    print(f"Statistics for {species_name}:\n")
    print("Percentiles:")
    print(data.describe(percentiles=[.25, .50, .75]))
    print("\nMean:")
    print(data.mean())
    print("\nStandard Deviation:")
    print(data.std())
    print("\n")
```

```
display_statistics(setosa_data, 'Iris-setosa')
display_statistics(versicolor_data, 'Iris-versicolor')
display_statistics(virginica_data, 'Iris-virginica')
```

```
Statistics for Iris-setosa:

Percentiles:
              Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count  50.00000       50.00000     50.000000      50.000000      50.00000
mean   25.50000        5.00600      3.418000       1.464000       0.24400
std    14.57738        0.35249      0.381024       0.173511       0.10721
min     1.00000        4.30000      2.300000       1.000000       0.10000
25%    13.25000        4.80000      3.125000       1.400000       0.20000
50%    25.50000        5.00000      3.400000       1.500000       0.20000
75%    37.75000        5.20000      3.675000       1.575000       0.30000
max    50.00000        5.80000      4.400000       1.900000       0.60000

Mean:
```

```
Id               25.500
SepalLengthCm     5.006
SepalWidthCm      3.418
PetalLengthCm     1.464
PetalWidthCm      0.244
dtype: float64

Standard Deviation:
Id               14.577380
SepalLengthCm     0.352490
SepalWidthCm      0.381024
PetalLengthCm     0.173511
PetalWidthCm      0.107210
dtype: float64


Statistics for Iris-versicolor:

Percentiles:
                Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count   50.00000      50.000000     50.000000      50.000000     50.000000
mean    75.50000       5.936000      2.770000       4.260000      1.326000
std     14.57738       0.516171      0.313798       0.469911      0.197753
min     51.00000       4.900000      2.000000       3.000000      1.000000
25%     63.25000       5.600000      2.525000       4.000000      1.200000
50%     75.50000       5.900000      2.800000       4.350000      1.300000
75%     87.75000       6.300000      3.000000       4.600000      1.500000
max    100.00000       7.000000      3.400000       5.100000      1.800000

Mean:
Id               75.500
SepalLengthCm     5.936
SepalWidthCm      2.770
PetalLengthCm     4.260
PetalWidthCm      1.326
dtype: float64

Standard Deviation:
Id               14.577380
SepalLengthCm     0.516171
SepalWidthCm      0.313798
PetalLengthCm     0.469911
PetalWidthCm      0.197753
```

Problem Statement - 1 - data.csv - wine.csv

```
df=pd.read_csv('/content/wine.csv')
df.head(7)
```

	Wine	Alcohol	Malic.acid	Ash	Acl	Mg	Phenols	Flavanoids	Nonflavanoid.phenols	Proanth	Color.int	Hue	OD
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93
5	1	14.20	1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	6.75	1.05	2.85
6	1	14.39	1.87	2.45	14.6	96	2.50	2.52	0.30	1.98	5.25	1.02	3.58

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Wine                  178 non-null    int64
 1   Alcohol               178 non-null    float64
 2   Malic.acid            178 non-null    float64
 3   Ash                   178 non-null    float64
 4   Acl                   178 non-null    float64
 5   Mg                    178 non-null    int64
 6   Phenols               178 non-null    float64
```

```
 7   Flavanoids            178 non-null    float64
 8   Nonflavanoid.phenols  178 non-null    float64
 9   Proanth               178 non-null    float64
 10  Color.int             178 non-null    float64
 11  Hue                   178 non-null    float64
 12  OD                    178 non-null    float64
 13  Proline               178 non-null    int64
dtypes: float64(11), int64(3)
memory usage: 19.6 KB
```

```
df.describe()
```

	Wine	Alcohol	Malic.acid	Ash	Acl	Mg	Phenols	Flavanoids	Nonflavanoid.phenols
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000
mean	1.938202	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.029270	0.361854
std	0.775035	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	0.998859	0.124453
min	1.000000	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.340000	0.130000
25%	1.000000	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.205000	0.270000
50%	2.000000	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.135000	0.340000
75%	3.000000	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	2.875000	0.437500
max	3.000000	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.080000	0.660000

```
df.mean()
```

```
Wine                     1.938202
Alcohol                 13.000618
Malic.acid               2.336348
Ash                      2.366517
Acl                     19.494944
Mg                      99.741573
Phenols                  2.295112
Flavanoids               2.029270
Nonflavanoid.phenols     0.361854
Proanth                  1.590899
Color.int                5.058090
Hue                      0.957449
OD                       2.611685
Proline                746.893258
dtype: float64
```

```
a=df.loc[:,'Ash'].mean()
print("The mean of the Ash is",a)
```

```
The mean of the Ash is 2.3665168539325845
```

```
df.mean(axis=1)[0:4]
#doubt
```

```
0     89.000000
1     85.364286
2     95.915714
3    117.963571
dtype: float64
```

```
a=df['Ash']

c=a.to_list()
sum=0
for i in a:
  sum=sum+i

print(sum)
b=len(c)
print("the total number of values is",b)
d=(sum/b)
print("the average is",d)
```

```
421.2400000000002
the total number of values is 178
the average is 2.3665168539325854
```

```
df.median()
```

```
Wine                     2.000
Alcohol                 13.050
Malic.acid               1.865
Ash                      2.360
Acl                     19.500
Mg                      98.000
Phenols                  2.355
Flavanoids               2.135
Nonflavanoid.phenols     0.340
Proanth                  1.555
Color.int                4.690
Hue                      0.965
OD                       2.780
Proline                673.500
dtype: float64
```

```
a=df['Mg']
b=a.to_list()

b.sort()
b
c=len(b)
c
```

```
178
```

```
# Assuming df is your DataFrame

a = df['Ash']

total_sum = a.sum()  # Use the sum() method of the Series
total_values = len(a)
average = total_sum / total_values

print("The sum is:", total_sum)
print("The total number of values is:", total_values)
print("The average is:", average)
```

```
The sum is: 421.24
The total number of values is: 178
The average is: 2.3665168539325845
```

```
import pandas as pd
from sklearn.datasets import load_wine


wine_data = load_wine()
features = wine_data['data']
target = wine_data['target']
feature_names = wine_data['feature_names']
target_names = wine_data['target_names']


df = pd.DataFrame(features, columns=feature_names)
df['class'] = target_names[target]


def display_grouped_statistics(data, categorical_column):
    grouped_stats = data.groupby(categorical_column).describe().transpose()
    return grouped_stats


categorical_column = 'class'

grouped_statistics = display_grouped_statistics(df, categorical_column)
print(grouped_statistics)
```

```
class              class_0     class_1     class_2
alcohol count    59.000000   71.000000   48.000000
        mean     13.744746   12.278732   13.153750
        std       0.462125    0.537964    0.530241
```

```
              min      12.850000   11.030000   12.200000
              25%      13.400000   11.915000   12.805000
     ...                     ...         ...         ...
     proline min     680.000000  278.000000  415.000000
              25%     987.500000  406.500000  545.000000
              50%    1095.000000  495.000000  627.500000
              75%    1280.000000  625.000000  695.000000
              max    1680.000000  985.000000  880.000000

     [104 rows x 3 columns]
```

```
import pandas as pd
a = df['Mg']
b = a.to_list()

b.sort()
c = len(b)

if c % 2 == 0:
    median_index = c // 2
    median = (b[median_index - 1] + b[median_index]) / 2
    print("The median is:", median)
else:
    median_index = c // 2
    median = b[median_index]
    print("The median is:", median)
```

```
     The median is: 98.0
```

```
a=df.loc[:,'Proanth'].median()
a
```

```
     1.5550000000000002
```

```
df.mode()
```

	Wine	Alcohol	Malic.acid	Ash	Acl	Mg	Phenols	Flavanoids	Nonflavanoid
0	2.0	12.37	1.73	2.28	20.0	88.0	2.2	2.65	
1	NaN	13.05	NaN	2.30	NaN	NaN	NaN	NaN	
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

```
# #example for undestanding the mode

# import pandas as pd

# data = {'Column': [1, 2, 2, 3, 3, 4]}
# df = pd.DataFrame(data)

# modes = df['Column'].mode()
# print("Modes:")
# print(modes)
```

in the mode , if there are 3 values then there is a tie between 3

```
import pandas as pd
a=df['Mg']
b=a.to_list()
# print(b)
d=[]
for i in b:
  c=b.count(i)
  d.append(c)
print(max(d))
```

```
     13
```

```
df.min()
```

```
Wine                     1.00
Alcohol                 11.03
Malic.acid               0.74
Ash                      1.36
Acl                     10.60
Mg                      70.00
Phenols                  0.98
Flavanoids               0.34
Nonflavanoid.phenols     0.13
Proanth                  0.41
Color.int                1.28
Hue                      0.48
OD                       1.27
Proline                278.00
dtype: float64
```

```
df.max()
```

```
Wine                     3.00
Alcohol                 14.83
Malic.acid               5.80
Ash                      3.23
Acl                     30.00
Mg                     162.00
Phenols                  3.88
Flavanoids               5.08
Nonflavanoid.phenols     0.66
Proanth                  3.58
Color.int               13.00
Hue                      1.71
OD                       4.00
Proline                1680.00
dtype: float64
```

```
a=df['Proline']
b=a.to_list()
max=0
for i in b:
  if i >= max:
    max=i
print("the maximum value is",max)
```

```
the maximum value is 1680
```

```
a=df['Proline']
b=a.to_list()
min=9999
for i in b:
  if i <= min:
    min=i
print("the minimum value is",min)
```

```
the minimum value is 0
```

```
df.min()
```

```
Wine                     1.00
Alcohol                 11.03
Malic.acid               0.74
Ash                      1.36
Acl                     10.60
Mg                      70.00
Phenols                  0.98
Flavanoids               0.34
Nonflavanoid.phenols     0.13
Proanth                  0.41
Color.int                1.28
Hue                      0.48
OD                       1.27
Proline                278.00
dtype: float64
```

```
a = df['Proline']
b = a.to_list()

if len(b) > 0:
    min_value = b[0]

    for i in b:
        if i < min_value:
            min_value = i

    print("The minimum value is", min_value)
else:
    print("The list is empty.")
```

```
df.std()
```

```
Wine                     0.775035
Alcohol                  0.811827
Malic.acid               1.117146
Ash                      0.274344
Acl                      3.339564
Mg                      14.282484
Phenols                  0.625851
Flavanoids               0.998859
Nonflavanoid.phenols     0.124453
Proanth                  0.572359
Color.int                2.318286
Hue                      0.228572
OD                       0.709990
Proline                314.907474
dtype: float64
```

```
df['Proline'].std()
```

```
314.9074742768491
```

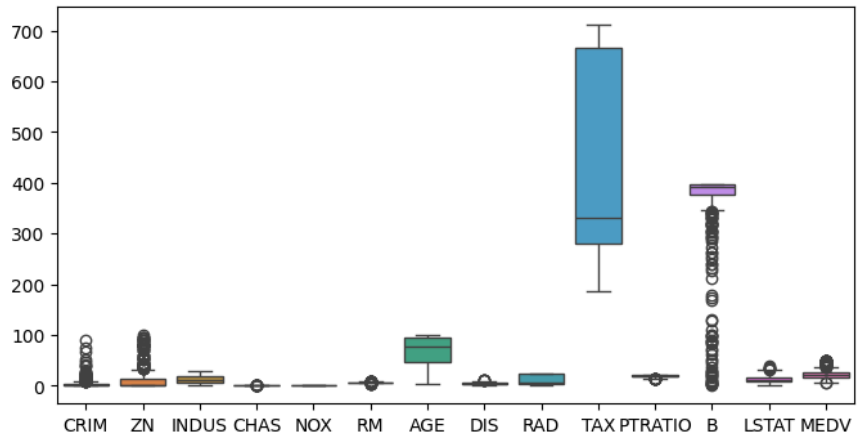
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.linear_model import LinearRegression


df = pd.read_csv("HousingData.csv")


df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     486 non-null    float64
 1   ZN       486 non-null    float64
 2   INDUS    486 non-null    float64
 3   CHAS     486 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      486 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64  
 9   TAX      506 non-null    int64  
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    486 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
```

```
df.isnull().sum()
```

```
CRIM       20
ZN         20
INDUS      20
CHAS       20
NOX         0
RM          0
AGE        20
DIS         0
RAD         0
TAX         0
PTRATIO     0
B           0
LSTAT      20
MEDV        0
dtype: int64
```

```
plt.figure(figsize=[8,4])
sns.boxplot(data=df)
plt.show()
```




```
df1 = df[(df['RM'] < 7)]
df = df1[(df1['RM']>5.2)]

df = df[(df['DIS'] < 9.2)]

df = df[(df['LSTAT'] < 29)]
```

```
df.shape[0] - df1.shape[0]
```

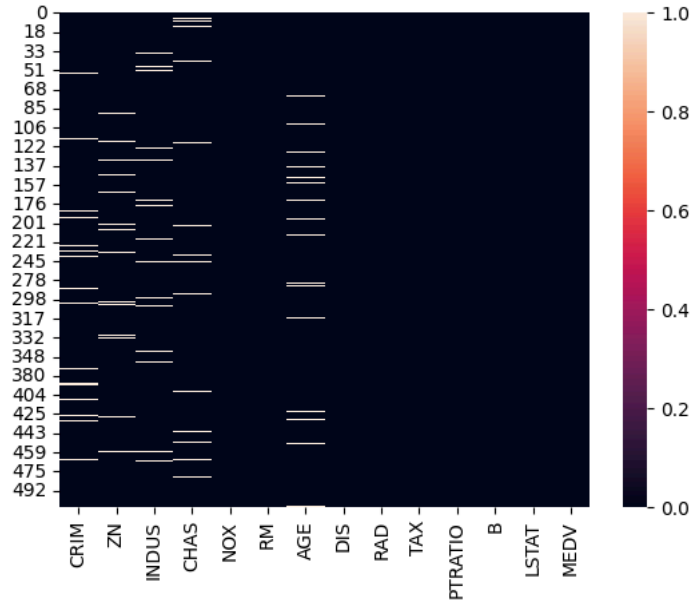
```
-54
```

```
df.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AG
count	371.000000	373.000000	372.000000	374.000000	388.000000	388.000000	372.00000
mean	3.359170	8.320375	11.719704	0.066845	0.556551	6.178562	69.15967
std	8.820305	19.321833	6.603709	0.250088	0.112910	0.387884	27.30515
min	0.006320	0.000000	0.740000	0.000000	0.385000	5.272000	2.90000
25%	0.090665	0.000000	6.035000	0.000000	0.459500	5.895750	46.60000
50%	0.239120	0.000000	9.900000	0.000000	0.538000	6.162000	77.50000
75%	3.428030	0.000000	18.100000	0.000000	0.624000	6.439000	93.92500
max	88.976200	100.000000	27.740000	1.000000	0.871000	6.998000	100.00000

```
sns.heatmap(df.isnull())
```

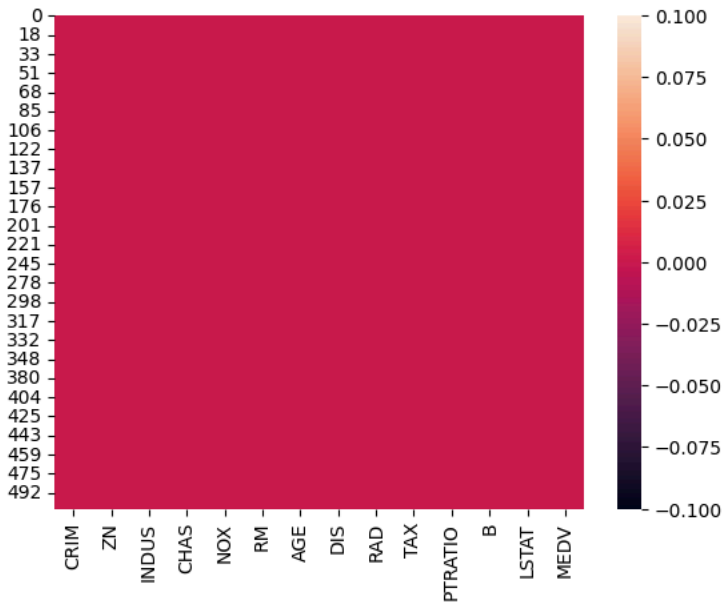
```
<Axes: >
```




```
df['CRIM'].fillna(value=df['CRIM'].mean(),inplace=True)
df['ZN'].fillna(value=df['ZN'].mean(),inplace=True)
df['INDUS'].fillna(value=df['INDUS'].mean(),inplace=True)
df['CHAS'].fillna(value=df['CHAS'].mean(),inplace=True)
df['AGE'].fillna(value=df['AGE'].mean(),inplace=True)
df['LSTAT'].fillna(value=df['LSTAT'].mean(),inplace=True)
```

```
sns.heatmap(df.isnull())
```

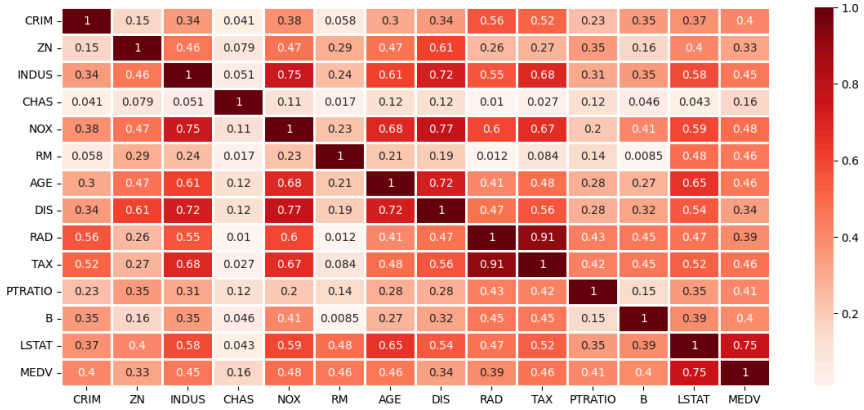
```
<Axes: >
```




```
df.isnull().sum()
```

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

```
plt.figure(figsize=(14,6))
corr=abs(df.corr())
sns.heatmap(corr,annot=True,linewidth=1,cmap="Reds")
plt.show()
```




```
df.count()
```

```
CRIM       388
ZN         388
```

```
INDUS      388
CHAS       388
NOX        388
RM         388
AGE        388
DIS        388
RAD        388
TAX        388
PTRATIO    388
B          388
LSTAT      388
MEDV       388
dtype: int64
```

```
df.dtypes
```

```
CRIM       float64
ZN         float64
INDUS      float64
CHAS       float64
NOX        float64
RM         float64
AGE        float64
DIS        float64
RAD          int64
TAX          int64
PTRATIO    float64
B          float64
LSTAT      float64
MEDV       float64
dtype: object
```

```
X = df.drop('MEDV', axis='columns')
y = df.MEDV
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=30)
```

```
from sklearn.linear_model import LinearRegression
clf = LinearRegression()
clf.fit(X_train, y_train)
clf.score(X_test, y_test)
```

```
0.585220657973823
```

SL - 3 PRACTICAL 5 DATA ANALYTICS 2 - SOCIAL NETWROK ADS

```
import pandas as pd
import numpy as np
from  sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```
df=pd.read_csv("/content/Social_Network_Ads.csv")
```

```
df.describe()
```

	User ID	Age	EstimatedSalary	Purchased
count	4.000000e+02	400.000000	400.000000	400.000000
mean	1.569154e+07	37.655000	69742.500000	0.357500
std	7.165832e+04	10.482877	34096.960282	0.479864
min	1.556669e+07	18.000000	15000.000000	0.000000
25%	1.562676e+07	29.750000	43000.000000	0.000000
50%	1.569434e+07	37.000000	70000.000000	0.000000
75%	1.575036e+07	46.000000	88000.000000	1.000000
max	1.581524e+07	60.000000	150000.000000	1.000000

```
df.isnull().sum()
```

```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

```
df.shape
```

```
(400, 5)
```

```
X = df.iloc[:, [2, 3]]
y = df.iloc[:, 4]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
from sklearn.preprocessing import StandardScaler
sc= StandardScaler()
X_train = sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

```
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
print(f"Accuracy: {accuracy}")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(classification_rep)
```

```
Accuracy: 0.8375
Confusion Matrix:
[[49  0]
 [13 18]]
Classification Report:
              precision    recall  f1-score   support

           0       0.79      1.00      0.88        49
           1       1.00      0.58      0.73        31

    accuracy                           0.84        80
   macro avg       0.90      0.79      0.81        80
weighted avg       0.87      0.84      0.83        80
```

```
tp,fn,fp,tn=confusion_matrix(y_test,y_pred,labels=[0,1]).reshape(-1)
print('Output Vlaues: \n',tp,fn,fp,tn )
```

```
Output Vlaues:
 49 0 13 18
```

```
accuracy_cm = (tp+tn)/(tp+fp+tn+fn)
precision_cm = tp/ (tp+fp)
recall_cm = tp/ (tp+fn)
f1_score = 2/ ((1/recall_cm)+(1/precision_cm))
```

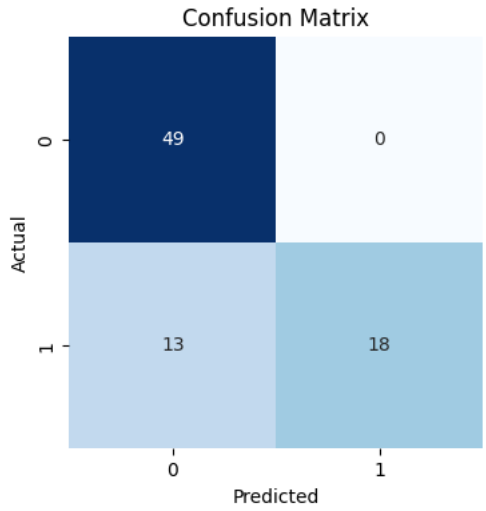
```
print("Accuracy : ",accuracy_cm)

print("Precision : ",precision_cm)

print("Recall :",recall_cm)

print("F1-Score : ",f1_score)
```

```
Accuracy :  0.8375
Precision :  0.7903225806451613
Recall : 1.0
F1-Score :  0.8828828828828829
```

```
Error_rate = (fp+fn)/(tp+fp+tn+fn)
print("Error Rate:", Error_rate)
```

```
Error Rate: 0.1625
```

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(4, 4))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```



Start coding or generate with AI.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix,accuracy_score, precision_score, recall_score, f1_score


from sklearn.datasets import load_iris
iris = load_iris()
iris_data = pd.DataFrame(data=np.c_[iris['data'], iris['target']], columns=iris['feature_names'] + ['target'])



X = iris_data.drop('target', axis=1)
y = iris_data['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)




naive_bayes_classifier = GaussianNB()
naive_bayes_classifier.fit(X_train, y_train)
```

▾ GaussianNB
GaussianNB()

```
y_pred = naive_bayes_classifier.predict(X_test)



conf_matrix = confusion_matrix(y_test, y_pred)
conf_matrix
```

```
array([[10,  0,  0],
       [ 0,  9,  0],
       [ 0,  0, 11]])
```

```
TP = conf_matrix[1, 1]  # True Positive
FP = conf_matrix[0, 1]  # False Positive
TN = conf_matrix[0, 0]  # True Negative
FN = conf_matrix[1, 0]  # False Negative

# Compute performance metrics
accuracy = accuracy_score(y_test, y_pred)
error_rate = 1 - accuracy
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Print the results
print("Confusion Matrix:")
print(conf_matrix)
print("\nTrue Positive (TP):", TP)
print("False Positive (FP):", FP)
print("True Negative (TN):", TN)
print("False Negative (FN):", FN)
print("\nAccuracy:", accuracy)
print("Error Rate:", error_rate)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]

True Positive (TP): 9
False Positive (FP): 0
True Negative (TN): 10
False Negative (FN): 0

Accuracy: 1.0
Error Rate: 0.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
```

Tokenization

In [1]:
```
import nltk
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
nltk.download('stopwords')
from nltk import sent_tokenize
from nltk import word_tokenize
from nltk.corpus import stopwords
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\RBS\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\RBS\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\wordnet.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\RBS\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping taggers\averaged_perceptron_tagger.zip.
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\RBS\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

In [2]:
```
text='Real madrid is set to win the UCL for the season . Benzema might win
```

In [3]:
```
tokens_sents = nltk.sent_tokenize(text)
print(tokens_sents)
```

```
['Real madrid is set to win the UCL for the season .', 'Benzema might win
Balon dor .', 'Salah might be the runner up']
```

In [4]:
```
tokens_words = nltk.word_tokenize(text)
print(tokens_words)
```

```
['Real', 'madrid', 'is', 'set', 'to', 'win', 'the', 'UCL', 'for', 'the',
'season', '.', 'Benzema', 'might', 'win', 'Balon', 'dor', '.', 'Salah', 'm
ight', 'be', 'the', 'runner', 'up']
```

In [5]:
```
from nltk.stem import PorterStemmer
from nltk.stem.snowball import SnowballStemmer
from nltk.stem import LancasterStemmer
```

In [9]:
```
stem=[]
for i in tokens_words:
  ps = PorterStemmer()
  stem_word= ps.stem(i)
  stem.append(stem_word)
print(stem)
```

```
['real', 'madrid', 'is', 'set', 'to', 'win', 'the', 'ucl', 'for', 'the',
'season', '.', 'benzema', 'might', 'win', 'balon', 'dor', '.', 'salah', 'm
ight', 'be', 'the', 'runner', 'up']
```

Lemmatization

In [10]:
```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
```

In [11]:
```
lemmatized_output = ' '.join([lemmatizer.lemmatize(w) for w in stem])
print(lemmatized_output)
```

```
real madrid is set to win the ucl for the season . benzema might win balon
dor . salah might be the runner up
```

In [12]:
```
leme=[]
for i in stem:
  lemetized_word=lemmatizer.lemmatize(i)
  leme.append(lemetized_word)
print(leme)
```

```
['real', 'madrid', 'is', 'set', 'to', 'win', 'the', 'ucl', 'for', 'the',
'season', '.', 'benzema', 'might', 'win', 'balon', 'dor', '.', 'salah', 'm
ight', 'be', 'the', 'runner', 'up']
```

Part of Speech Tagging

In [13]:
```
print("Parts of Speech: ",nltk.pos_tag(leme))
```

```
Parts of Speech:  [('real', 'JJ'), ('madrid', 'NN'), ('is', 'VBZ'), ('se
t', 'VBN'), ('to', 'TO'), ('win', 'VB'), ('the', 'DT'), ('ucl', 'NN'), ('f
or', 'IN'), ('the', 'DT'), ('season', 'NN'), ('.', '.'), ('benzema', 'N
N'), ('might', 'MD'), ('win', 'VB'), ('balon', 'NN'), ('dor', 'NN'), ('.',
'.'), ('salah', 'NN'), ('might', 'MD'), ('be', 'VB'), ('the', 'DT'), ('run
ner', 'NN'), ('up', 'RP')]
```

Stop Word

In [14]:
```
sw_nltk = stopwords.words('english')
print(sw_nltk)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "yo
u're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselv
es', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'hersel
f', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'the
mselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'thes
e', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'hav
e', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'th
e', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'a
t', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through',
'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down',
'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'on
ce', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',
'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'no
t', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can',
'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll',
'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't",
'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't",
'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',
"mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't",
'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "would
n't"]
```

In [15]:
```
words = [word for word in text.split() if word.lower() not in sw_nltk]
new_text = " ".join(words)
print(new_text)
```

```
Real madrid set win UCL season . Benzema might win Balon dor . Salah might
runner
```

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

In [2]:
```
data = pd.read_csv('https://raw.githubusercontent.com/dphi-official/Dataset
data
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fa
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25(
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.28:
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92:
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.10(
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05(
...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00(
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00(
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45(
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00(
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75(

891 rows × 12 columns

In [3]:
```
data.shape
```

Out[3]:
```
(891, 12)
```

In [4]:
```
data.describe()
```

Out[4]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [5]:
```
data.describe(include = 'object')
```

Out[5]:

	Name	Sex	Ticket	Cabin	Embarked
count	891	891	891	204	889
unique	891	2	681	147	3
top	Mangan, Miss. Mary	male	347082	G6	S
freq	1	577	7	4	644

In [6]:
```
data.isnull().sum()
```

Out[6]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [7]:
```
data['Age'] = data['Age'].fillna(np.mean(data['Age']))
```

In [8]:
```
data['Cabin'] = data['Cabin'].fillna(data['Cabin'].mode()[0])
```

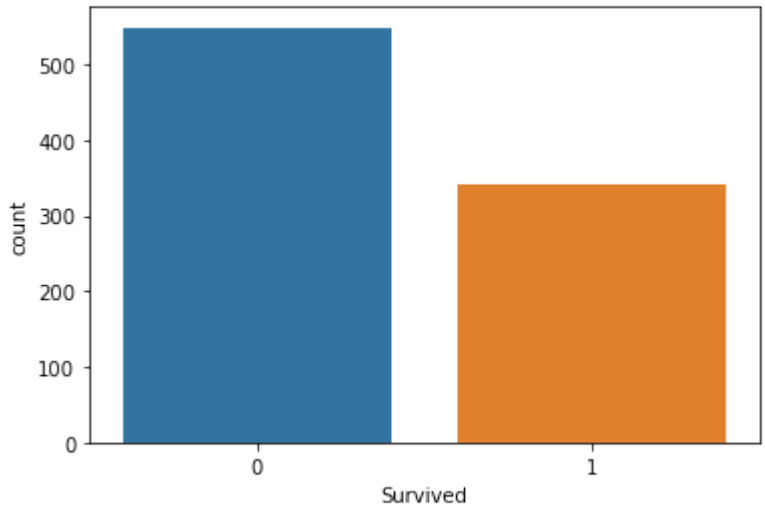
In [9]:
```
data['Embarked'] = data['Embarked'].fillna(data['Embarked'].mode()[0])
```

In [10]:
```
data.isnull().sum()
```

Out[10]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

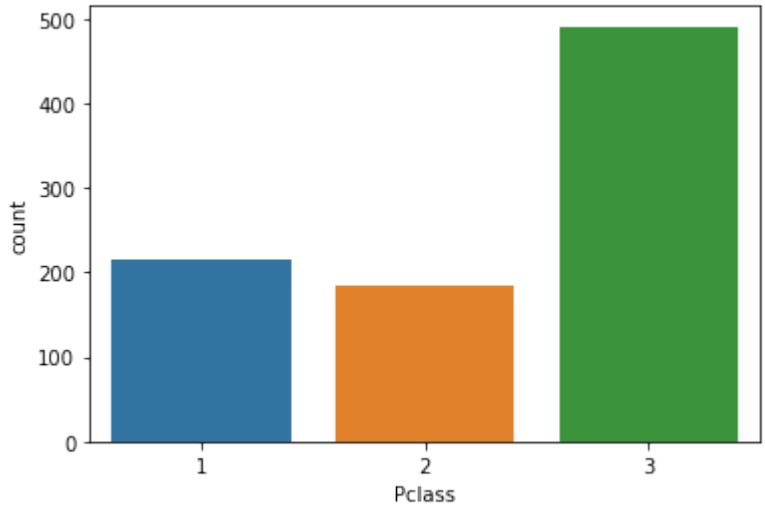
In [11]:
```
sns.countplot(data['Survived'])
```

Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b5c823e7c0>
```



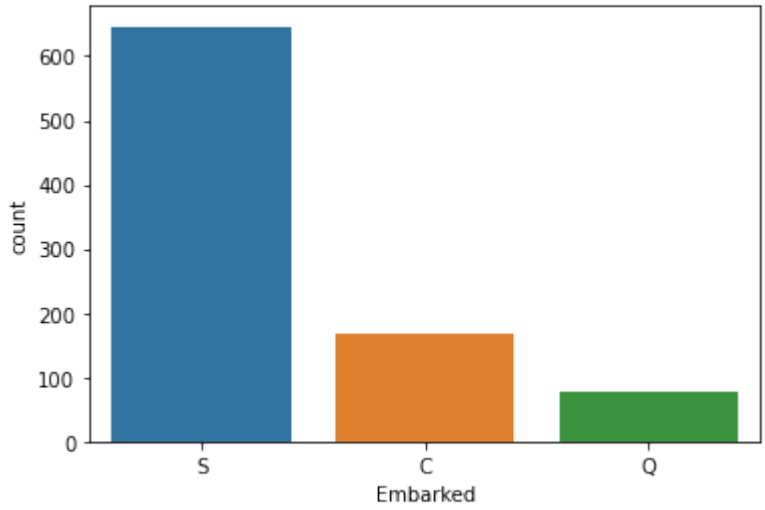
In [12]:
```
sns.countplot(data['Pclass'])
```

Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b5c8997c70>
```



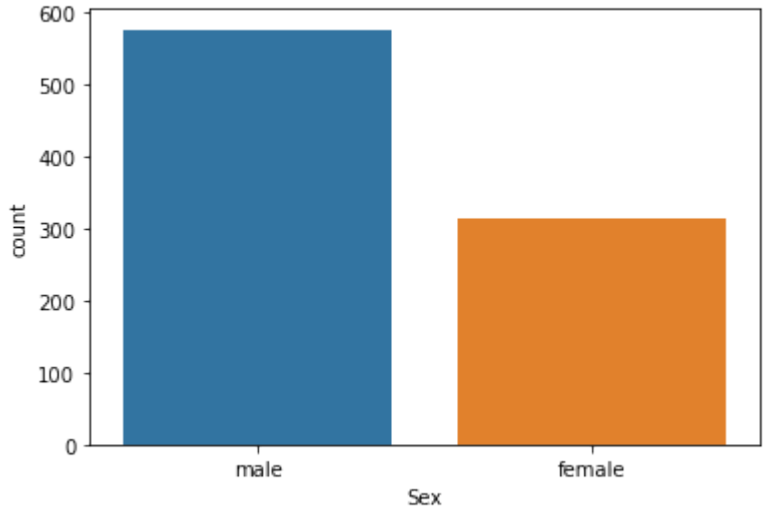
In [13]:
```
sns.countplot(data['Embarked'])
```

Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b5c89f9910>
```



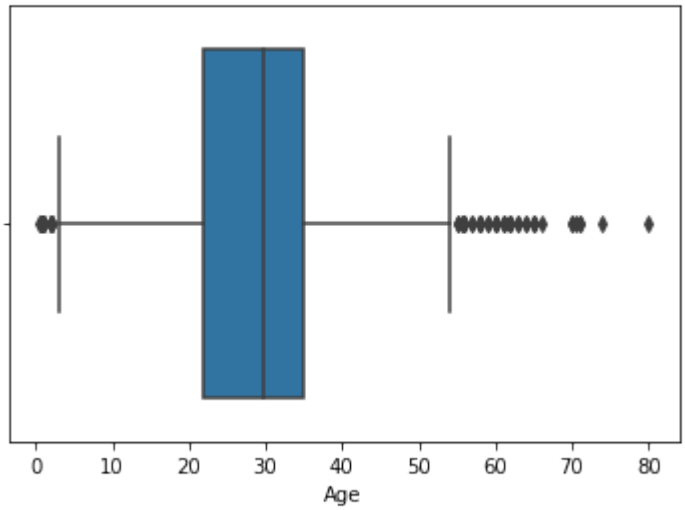
In [14]:
```
sns.countplot(data['Sex'])
```

Out[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b5c8a4e8b0>
```



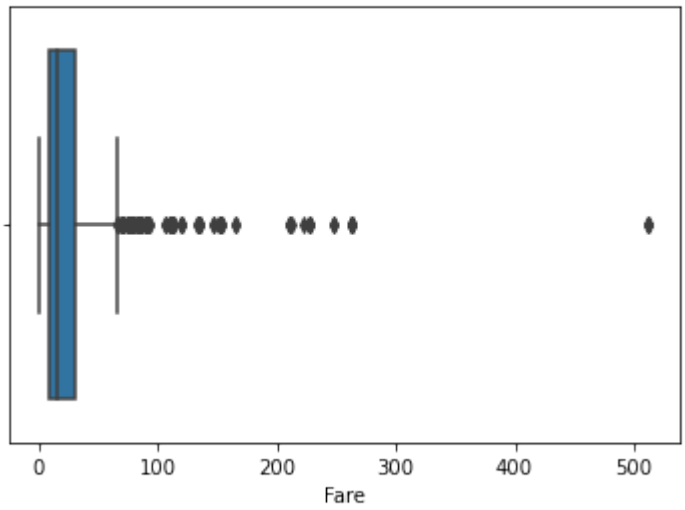
In [15]:
```
sns.boxplot(data['Age'])
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b5c8a9c490>
```



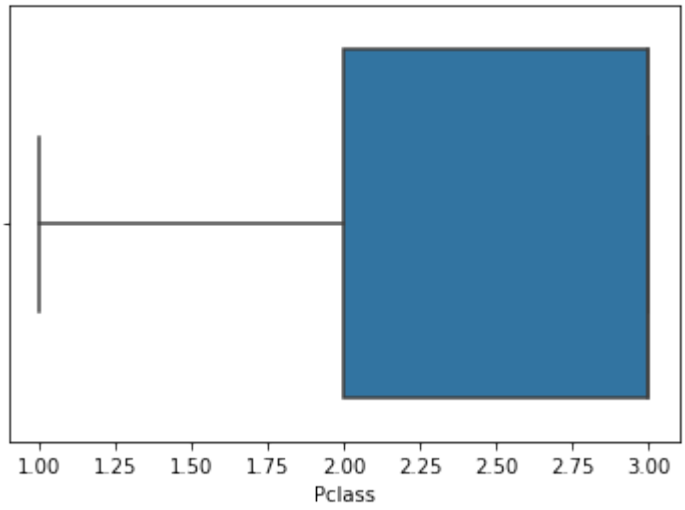
In [16]:
```
sns.boxplot(data['Fare'])
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b5c8aed7c0>
```



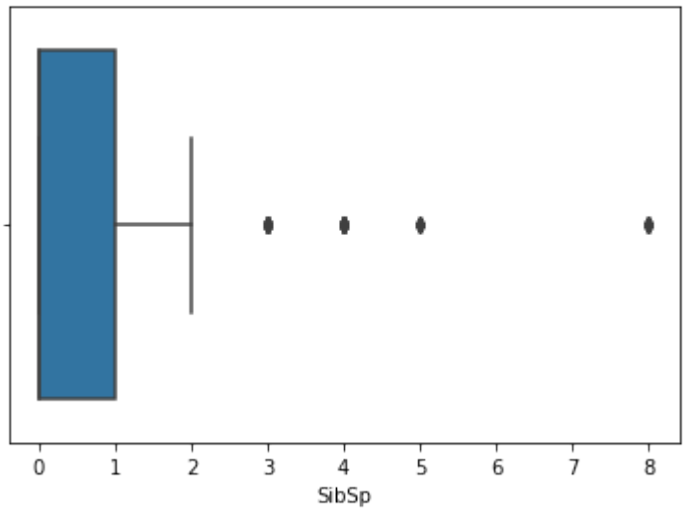
In [17]:
```
sns.boxplot(data['Pclass'])
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b5c8b52c10>
```



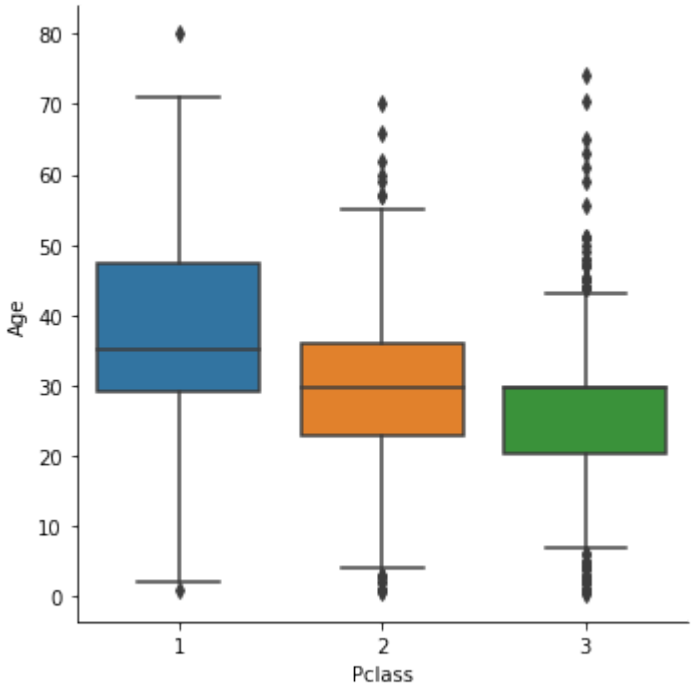
In [18]:
```
sns.boxplot(data['SibSp'])
```

Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b5c8b9f940>
```



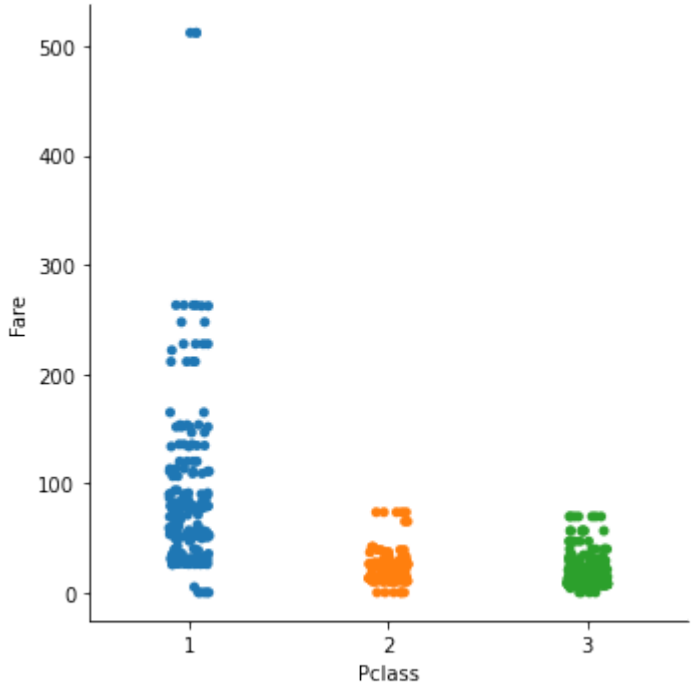
In [19]:
```
sns.catplot(x= 'Pclass', y = 'Age', data=data, kind = 'box')
```

Out[19]:
```
<seaborn.axisgrid.FacetGrid at 0x1b5c8affca0>
```



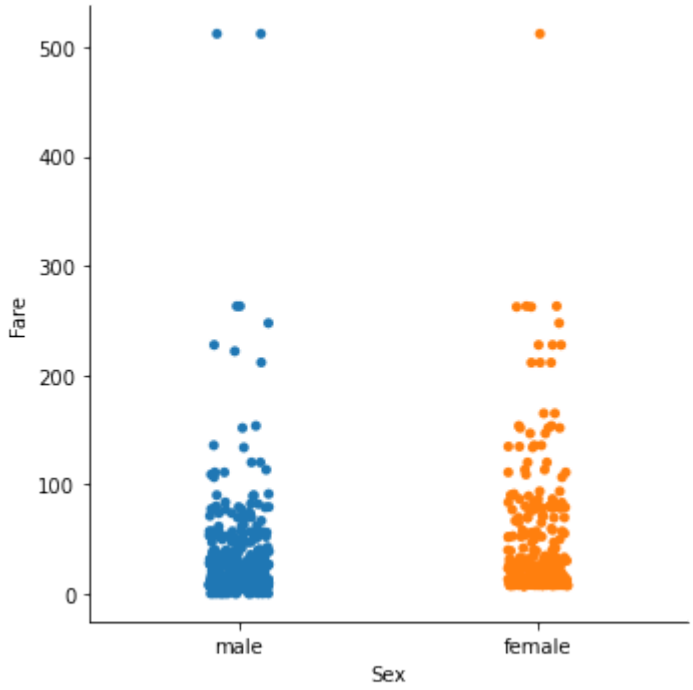
In [20]:
```
sns.catplot(x= 'Pclass', y = 'Fare', data=data, kind = 'strip')
```

Out[20]:
```
<seaborn.axisgrid.FacetGrid at 0x1b5c8ca3ee0>
```



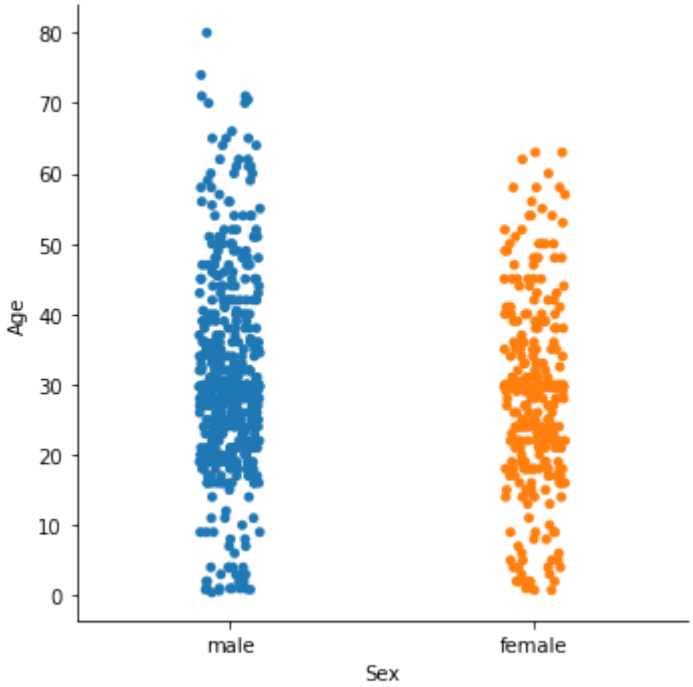
In [21]:
```
sns.catplot(x= 'Sex', y = 'Fare', data=data, kind = 'strip')
```

Out[21]:
```
<seaborn.axisgrid.FacetGrid at 0x1b5c8aed9a0>
```



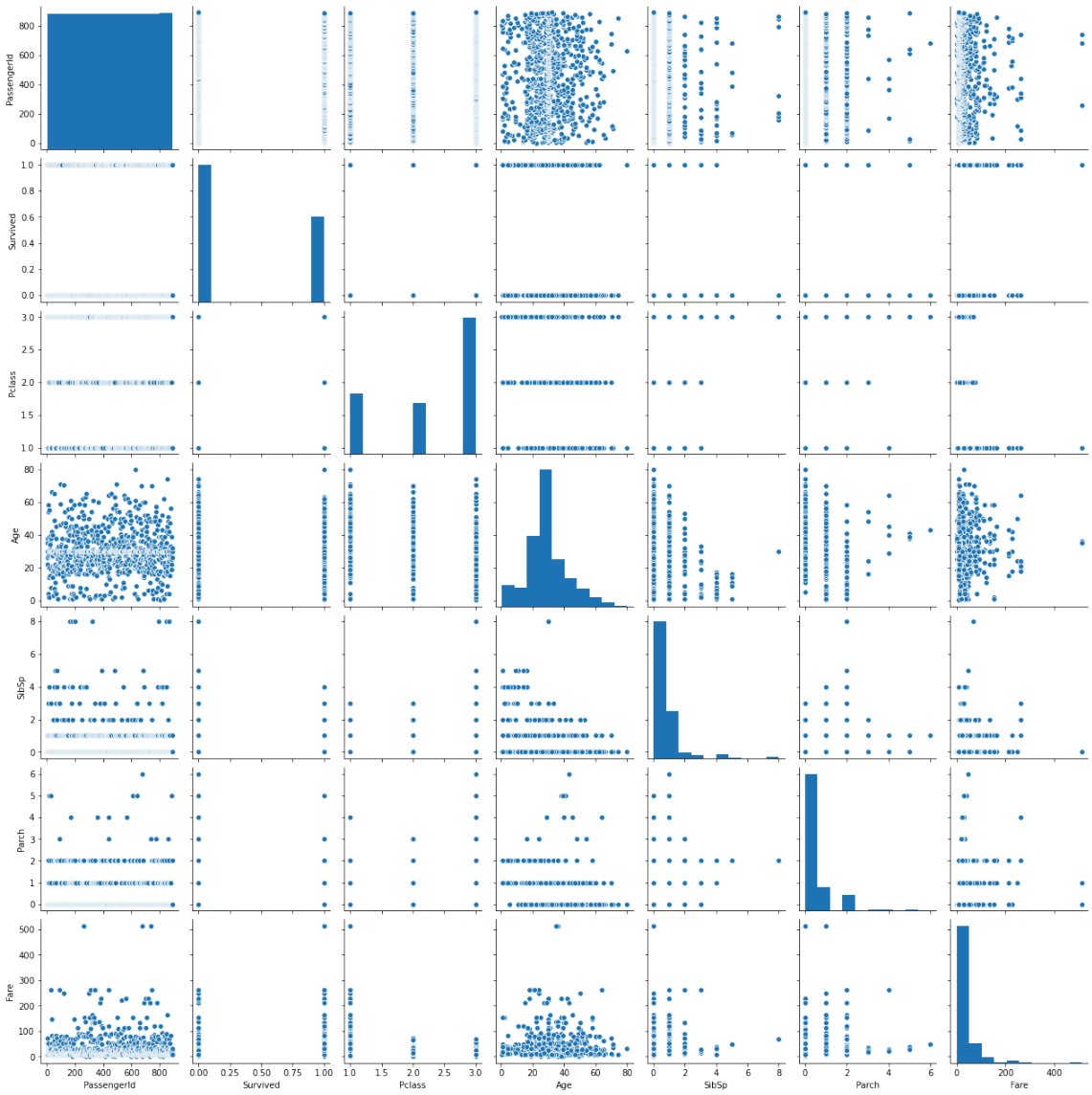
In [22]:
```
sns.catplot(x= 'Sex', y = 'Age', data=data, kind = 'strip')
```

Out[22]:
```
<seaborn.axisgrid.FacetGrid at 0x1b5c8d515e0>
```



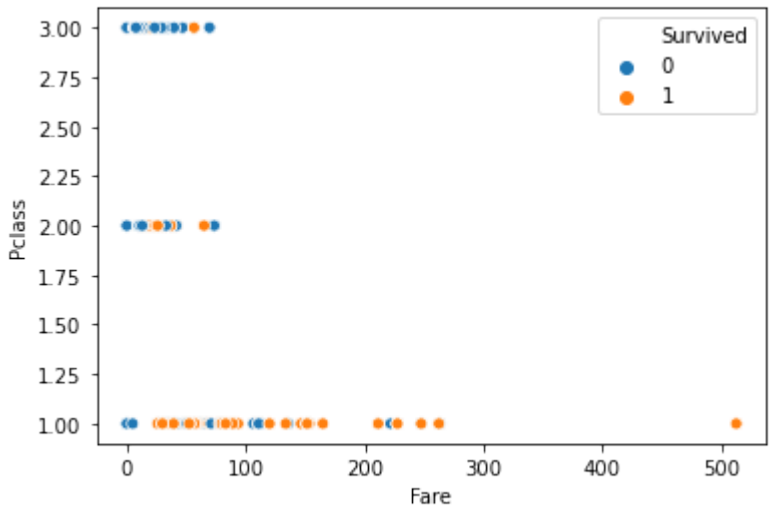
In [23]:
```
sns.pairplot(data)
```

Out[23]:
```
<seaborn.axisgrid.PairGrid at 0x1b5c8d9c490>
```



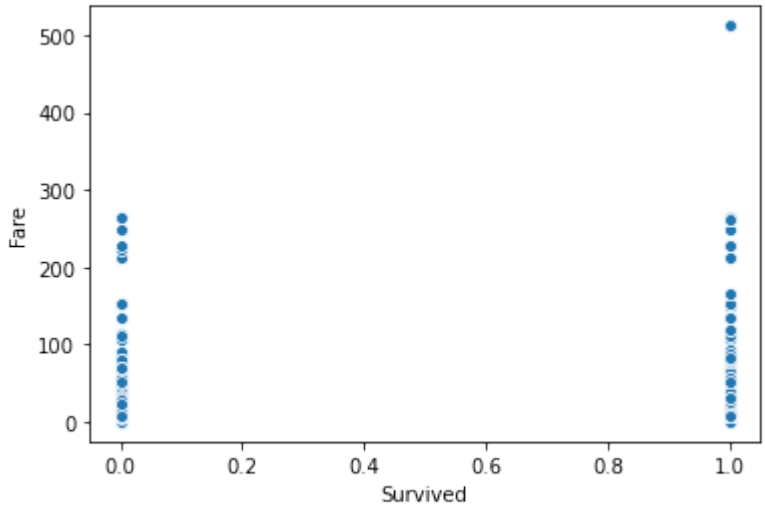
In [24]:
```
sns.scatterplot(x = 'Fare', y = 'Pclass', hue = 'Survived', data = data)
```

Out[24]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b5cae10190>
```



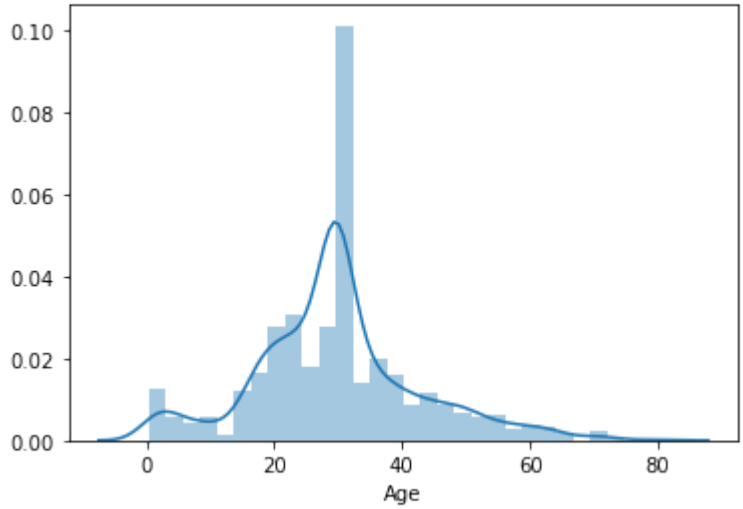
In [25]:
```
sns.scatterplot(x = 'Survived', y = 'Fare', data = data)
```

Out[25]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b5c8d84820>
```



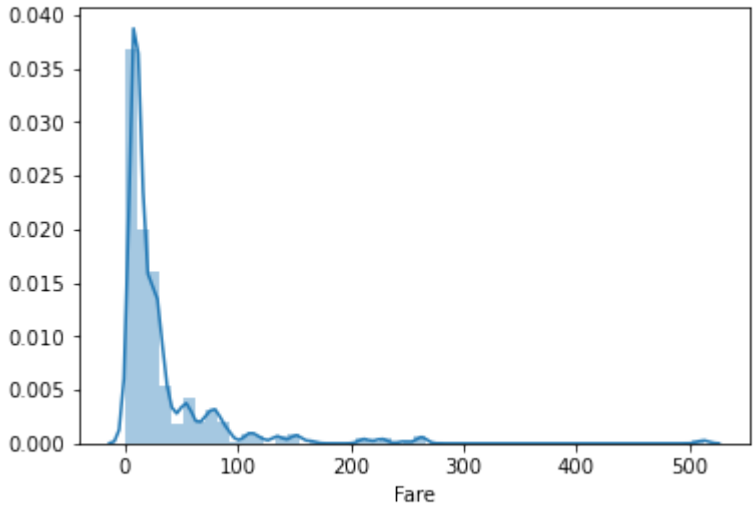
In [26]:
```
sns.distplot(data['Age'])
```

Out[26]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b5cba89460>
```



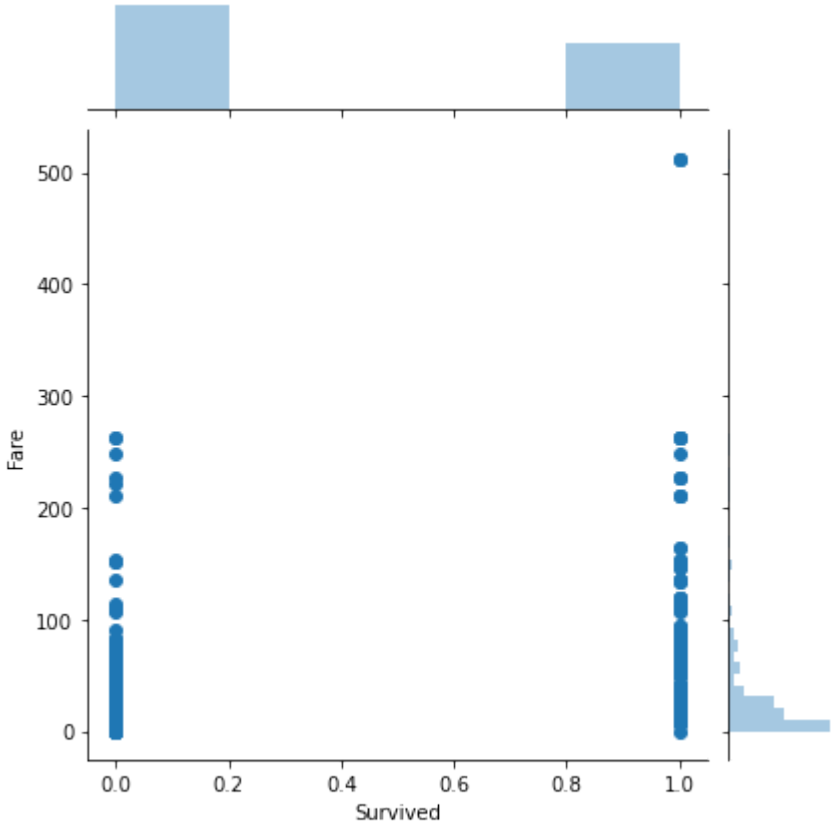
In [27]:
```
sns.distplot(data['Fare'])
```

Out[27]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b5cbb34f40>
```



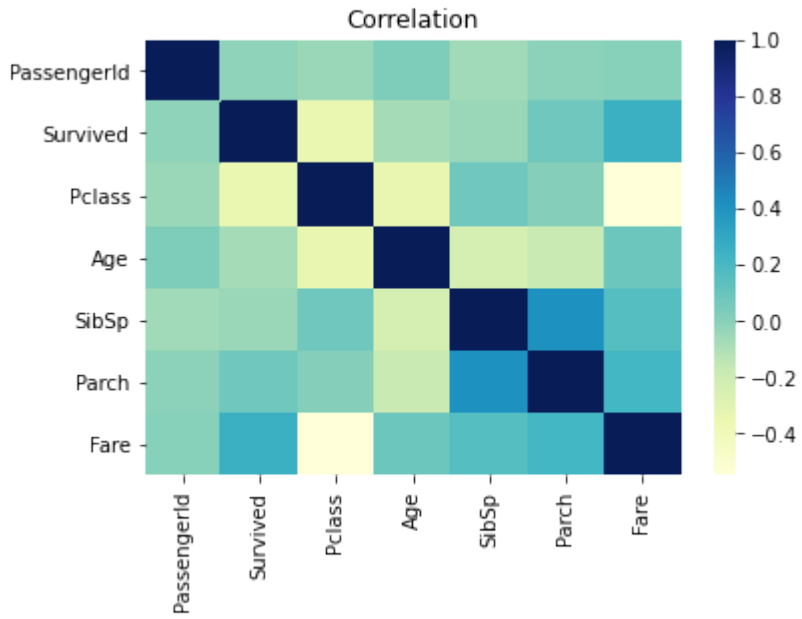
In [28]:
```
sns.jointplot(x = "Survived", y = "Fare", kind = "scatter", data = data)
```

Out[28]:
```
<seaborn.axisgrid.JointGrid at 0x1b5cbbf62b0>
```



In [29]:
```
tc = data.corr()
sns.heatmap(tc, cmap="YlGnBu")
plt.title('Correlation')
```

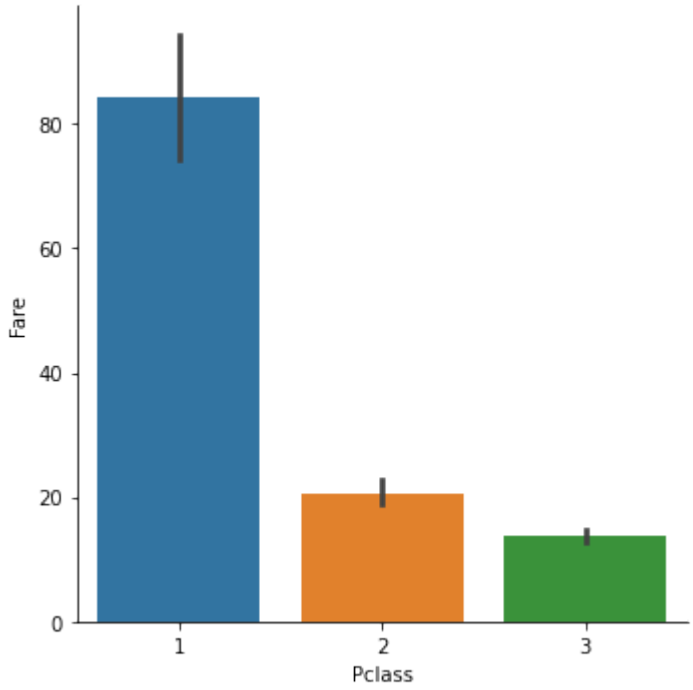
Out[29]:
```
Text(0.5, 1.0, 'Correlation')
```



Price of Ticket for each passenger is distributed

In [33]:
```
sns.catplot(x='Pclass', y='Fare', data=data, kind='bar')
```

Out[33]:
```
<seaborn.axisgrid.FacetGrid at 0x1b5ca67bcd0>
```



ASSIGNMENT-9

1. Use the inbuilt dataset 'titanic' as used in the above problem. Plot a box plot for distribution of age with respect to each gender along with the information about whether they survived or not. (Column names : 'sex' and 'age')
2. Write observations on the inference from the above statistics.

In [16]:
```
#importing required library
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

#loading dataset
data = pd.read_csv('https://raw.githubusercontent.com/dphi-official/Dataset
```

In [8]:
```
data.head()
```

Out[8]:

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

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

Out[12]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [9]:
```
data.isnull().sum()
```

Out[9]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

Here, we can see there are Null values in the dataset. Hence, we need to replace these values by mean (in case of numerical variables) or mode (in case of categorical variables)

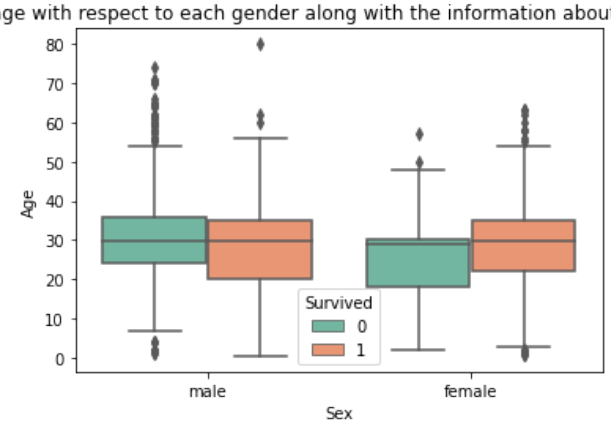
In [19]:
```
data['Age'] = data['Age'].fillna(np.mean(data['Age']))
data['Cabin'] = data['Cabin'].fillna(data['Cabin'].mode()[0])
data['Embarked'] = data['Embarked'].fillna(data['Embarked'].mode()[0])
```

In [20]:
```
data.isnull().sum()
```

Out[20]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

In [22]:
```
sns.boxplot(data['Sex'], data["Age"], data["Survived"], palette = 'Set2').s
plt.show()
```

Plot for distribution of age with respect to each gender along with the information about whether they survived or not



In [ ]:
```

```

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

In [3]:
```
data = pd.read_csv('https://gist.githubusercontent.com/curran/a08a1080b8834
data
```

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

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

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [5]:
```
data.describe()
```

Out[5]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [6]:
```
data.describe(include = 'object')
```

Out[6]:

	species
count	150
unique	3
top	setosa
freq	50

In [7]:
```
data.isnull().sum()
```

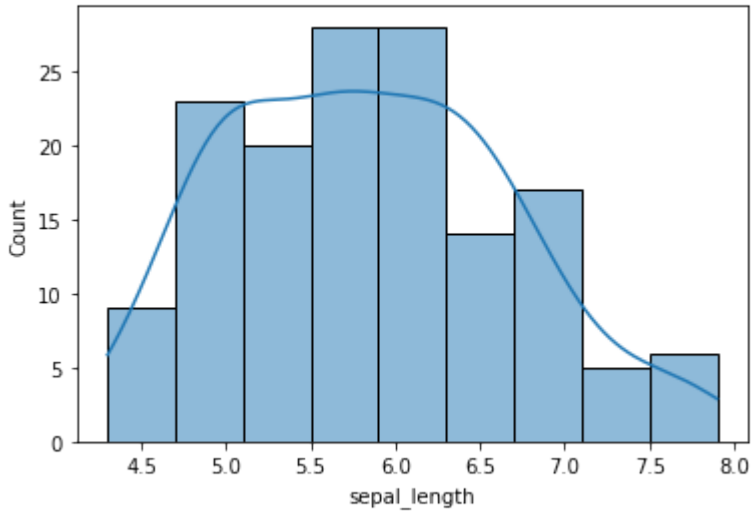
Out[7]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

In [8]:
```
print("\n\nThe features in the dataset are as follows : ")
print("1. Sepal length : ", data['sepal_length'].dtype)
print("2. Sepal width : ", data['sepal_width'].dtype)
print("3. Petal length : ", data['petal_length'].dtype)
print("4. Petal width : ", data['petal_width'].dtype)
print("5. Species : ", data['species'].dtype)
```

```
The features in the dataset are as follows :
1. Sepal length :  float64
2. Sepal width :  float64
3. Petal length :  float64
4. Petal width :  float64
5. Species :  object
```

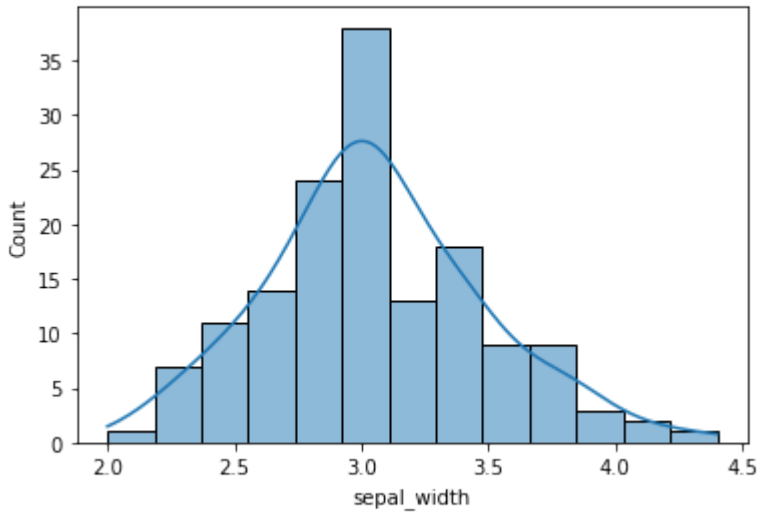
In [9]:
```
sns.histplot(x = data['sepal_length'], kde=True)
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe839f4d9d0>
```



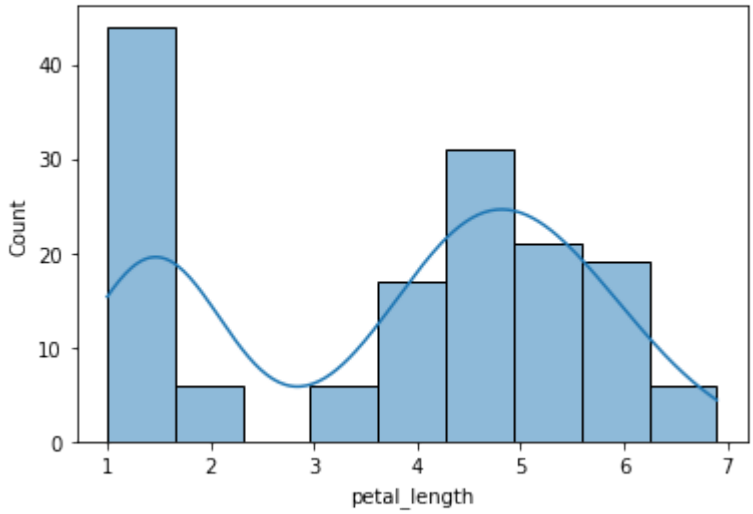
In [10]:
```
sns.histplot(x = data['sepal_width'], kde=True)
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe839343e90>
```



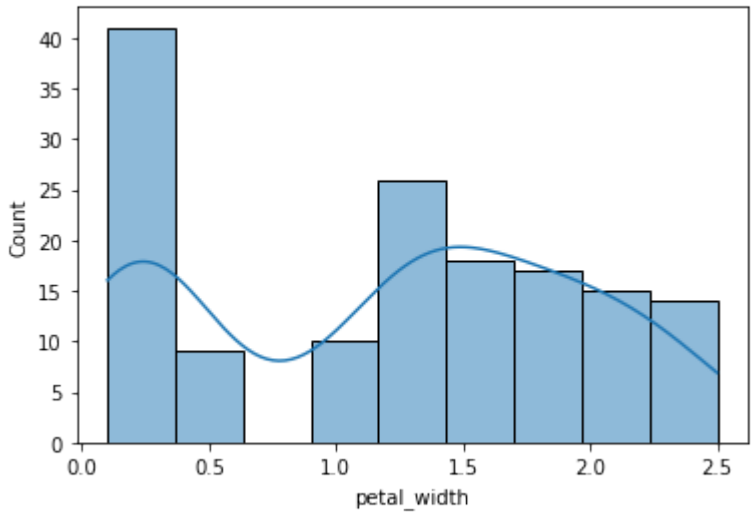
In [11]:
```
sns.histplot(x = data['petal_length'], kde=True)
```

Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe836d341d0>
```



In [12]:
```
sns.histplot(x = data['petal_width'], kde=True)
```

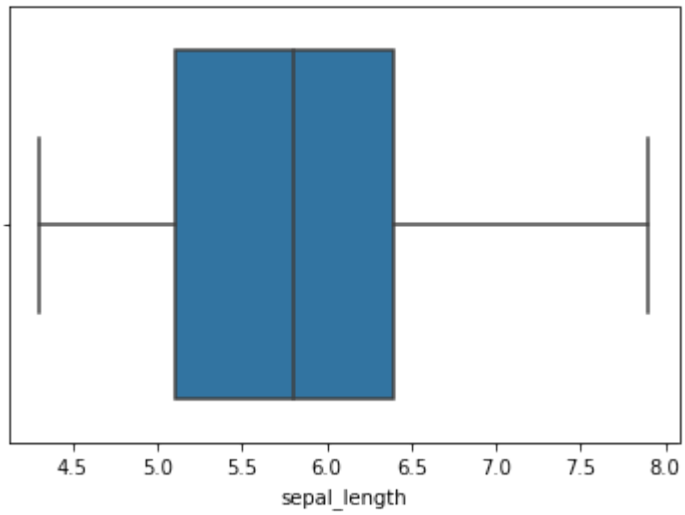
Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe836c64f50>
```



In [13]:
```
sns.boxplot(data['sepal_length'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWa
rning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argum
ents without an explicit keyword will result in an error or misinterpretat
ion.
  FutureWarning
```

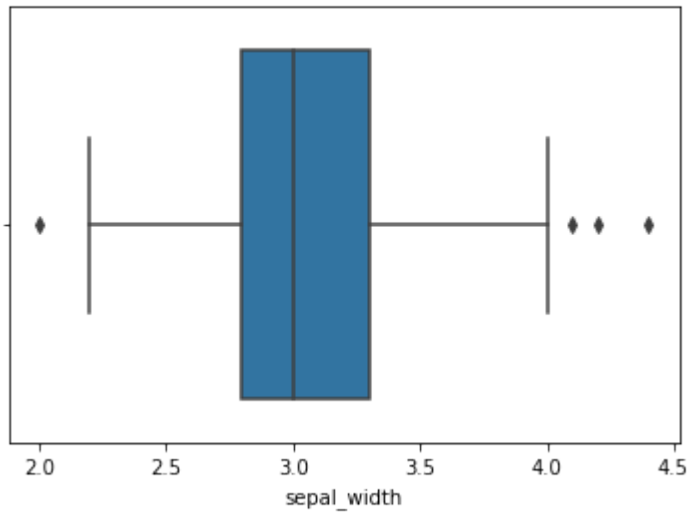
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe836b8a8d0>



In [14]:
```
sns.boxplot(data['sepal_width'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWa
rning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argum
ents without an explicit keyword will result in an error or misinterpretat
ion.
  FutureWarning
```

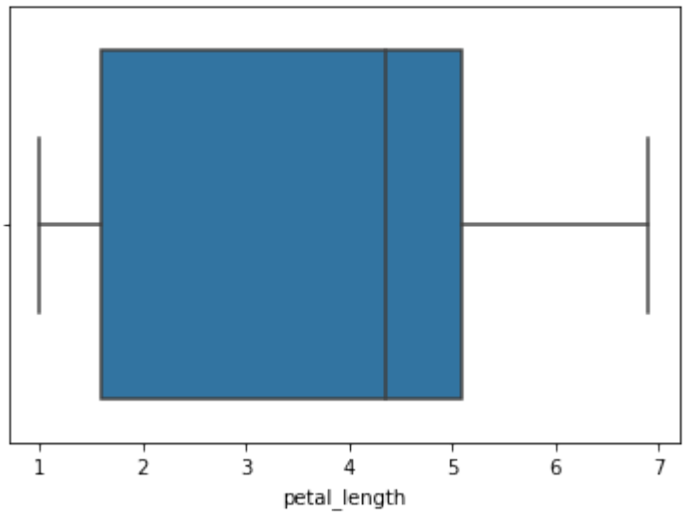
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe836c79ed0>



In [15]:
```
sns.boxplot(data['petal_length'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWa
rning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argum
ents without an explicit keyword will result in an error or misinterpretat
ion.
  FutureWarning
```

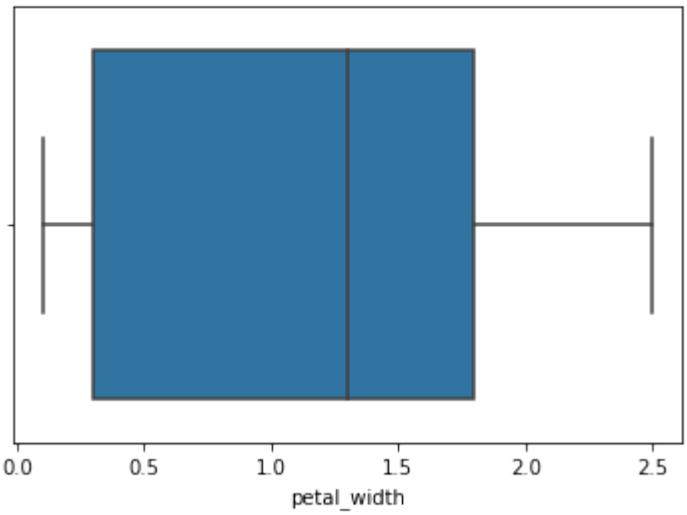
Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe836bf8290>
```



In [16]:
```
sns.boxplot(data['petal_width'])
```

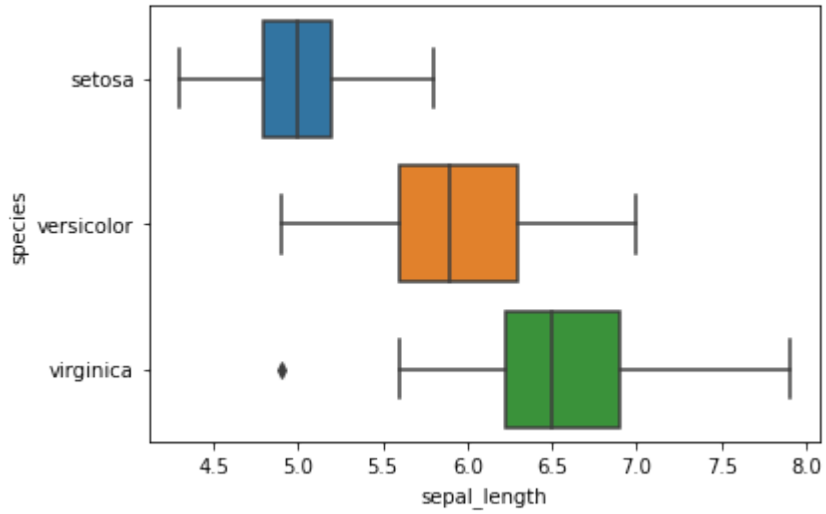
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWa
rning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argum
ents without an explicit keyword will result in an error or misinterpretat
ion.
  FutureWarning
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe836a5f850>
```



In [17]:
```
sns.boxplot(x='sepal_length',y='species',data=data)
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe836a3ca90>
```



In [18]:
```
sns.boxplot(x='petal_length',y='species',data=data)
```

Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe83696b950>
```