In [2]:
```
import seaborn as sns
sns.get_dataset_names()
```

Out[2]:
```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'dowjones',
 'exercise',
 'flights',
 'fmri',
 'geyser',
 'glue',
 'healthexp',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'seaice',
 'taxis',
 'tips',
 'titanic',
 'anagrams',
 'anagrams',
 'anscombe',
 'anscombe',
 'attention',
 'attention',
 'brain_networks',
 'brain_networks',
 'car_crashes',
 'car_crashes',
 'diamonds',
 'diamonds',
 'dots',
 'dots',
 'dowjones',
 'dowjones',
 'exercise',
 'exercise',
 'flights',
 'flights',
 'fmri',
 'fmri',
 'geyser',
 'geyser',
 'glue',
 'glue',
 'healthexp',
 'healthexp',
 'iris',
 'iris',
 'mpg',
 'mpg',
 'penguins',
 'penguins',
 'planets',
 'planets',
 'seaice',
 'seaice',
 'taxis',
```

```
'taxis',
'tips',
'tips',
'titanic',
'titanic',
'anagrams',
'anscombe',
'attention',
'brain_networks',
'car_crashes',
'diamonds',
'dots',
'dowjones',
'exercise',
'flights',
'fmri',
'geyser',
'glue',
'healthexp',
'iris',
'mpg',
'penguins',
'planets',
'seaice',
'taxis',
'tips',
'titanic']
```

Practical no 1

In [3]:
```
df=sns.load_dataset("tips")
```

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

Out[4]:

	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 [5]:
```
import pandas as pd
```

In [6]:
```
df.head(5)
```

Out[6]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [7]:
```
df.tail(5)
```

Out[7]:

	total_bill	tip	sex	smoker	day	time	size
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

In [8]:
```
df.index
```

Out[8]:
```
RangeIndex(start=0, stop=244, step=1)
```

In [9]:
```
df.shape
```

Out[9]:
```
(244, 7)
```

In [10]:
```
df.dtypes
```

Out[10]:
```
total_bill     float64
tip            float64
sex           category
smoker        category
day           category
time          category
size             int64
dtype: object
```

In [11]:
```
df.columns.values
```

Out[11]:
```
array(['total_bill', 'tip', 'sex', 'smoker', 'day', 'time', 'size'],
      dtype=object)
```

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

Out[12]:

	total_bill	tip	size
count	244.000000	244.000000	244.000000
mean	19.785943	2.998279	2.569672
std	8.902412	1.383638	0.951100
min	3.070000	1.000000	1.000000
25%	13.347500	2.000000	2.000000
50%	17.795000	2.900000	2.000000
75%	24.127500	3.562500	3.000000
max	50.810000	10.000000	6.000000

In [13]:
```
df.sort_index(axis=1,ascending=False)
```

Out[13]:

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

244 rows × 7 columns

In [14]:
```
df.sort_values("total_bill")
```

Out[14]:

	total_bill	tip	sex	smoker	day	time	size
67	3.07	1.00	Female	Yes	Sat	Dinner	1
92	5.75	1.00	Female	Yes	Fri	Dinner	2
111	7.25	1.00	Female	No	Sat	Dinner	1
172	7.25	5.15	Male	Yes	Sun	Dinner	2
149	7.51	2.00	Male	No	Thur	Lunch	2
...	...	...	...	...	...	...	...
182	45.35	3.50	Male	Yes	Sun	Dinner	3
156	48.17	5.00	Male	No	Sun	Dinner	6
59	48.27	6.73	Male	No	Sat	Dinner	4
212	48.33	9.00	Male	No	Sat	Dinner	4
170	50.81	10.00	Male	Yes	Sat	Dinner	3

244 rows × 7 columns

In [15]:
```
df.iloc[5]
```

Out[15]:
```
total_bill     25.29
tip             4.71
sex             Male
smoker            No
day              Sun
time          Dinner
size               4
Name: 5, dtype: object
```

In [18]:
```
df.loc[:,["total_bill","tip"]]
```

Out[18]:

	total_bill	tip
0	16.99	1.01
1	10.34	1.66
2	21.01	3.50
3	23.68	3.31
4	24.59	3.61
...	...	...
239	29.03	5.92
240	27.18	2.00
241	22.67	2.00
242	17.82	1.75
243	18.78	3.00

244 rows × 2 columns

In [19]:
```
df.iloc[:6,:]
```

Out[19]:

	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
5	25.29	4.71	Male	No	Sun	Dinner	4

In [20]:
```
df.iloc[:10,:3]
```

Out[20]:

	total_bill	tip	sex
0	16.99	1.01	Female
1	10.34	1.66	Male
2	21.01	3.50	Male
3	23.68	3.31	Male
4	24.59	3.61	Female
5	25.29	4.71	Male
6	8.77	2.00	Male
7	26.88	3.12	Male
8	15.04	1.96	Male
9	14.78	3.23	Male

In [21]:
```
df.isnull()
```

Out[21]:

	total_bill	tip	sex	smoker	day	time	size
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
239	False	False	False	False	False	False	False
240	False	False	False	False	False	False	False
241	False	False	False	False	False	False	False
242	False	False	False	False	False	False	False
243	False	False	False	False	False	False	False

244 rows × 7 columns

In [22]:
```
df.isnull().any()
```

Out[22]:
```
total_bill    False
tip           False
sex           False
smoker        False
day           False
time          False
size          False
dtype: bool
```

In [23]:
```
df.isnull().sum()
```

Out[23]:
```
total_bill    0
tip           0
sex           0
smoker        0
day           0
time          0
size          0
dtype: int64
```

In [24]:
```
df.isna()
```

Out[24]:

	total_bill	tip	sex	smoker	day	time	size
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
239	False	False	False	False	False	False	False
240	False	False	False	False	False	False	False
241	False	False	False	False	False	False	False
242	False	False	False	False	False	False	False
243	False	False	False	False	False	False	False

244 rows × 7 columns

In [25]:
```
df.isna().sum()
```

Out[25]:
```
total_bill    0
tip           0
sex           0
smoker        0
day           0
time          0
size          0
dtype: int64
```

Practical no 2

In [26]:
```
import pandas as pd
import seaborn as sns
```

In [28]:
```
df=sns.load_dataset("tips")
```

In [29]:
```
df
```

Out[29]:

	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 [30]:
```
missing_values = ["Na", "na"]
```

In [31]:
```
missing_values
```

Out[31]:
```
['Na', 'na']
```

In [32]:
```
ndf=df
ndf.fillna(0)
```

Out[32]:

	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 [33]:
```
df['total_bill'] = df['total_bill'].fillna(df['total_bill'].mean())
```

In [34]:
```
df['total_bill']
```

Out[34]:
```
0      16.99
1      10.34
2      21.01
3      23.68
4      24.59
       ...
239    29.03
240    27.18
241    22.67
242    17.82
243    18.78
Name: total_bill, Length: 244, dtype: float64
```

In [36]:
```
ndf.dropna(how = 'all')
```

Out[36]:

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

244 rows × 7 columns

In [37]:
```
ndf.dropna(axis = 1)
```

Out[37]:

	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 [38]:
```
new_data = ndf.dropna(axis = 0, how ='any')
new_data
```

Out[38]:

	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

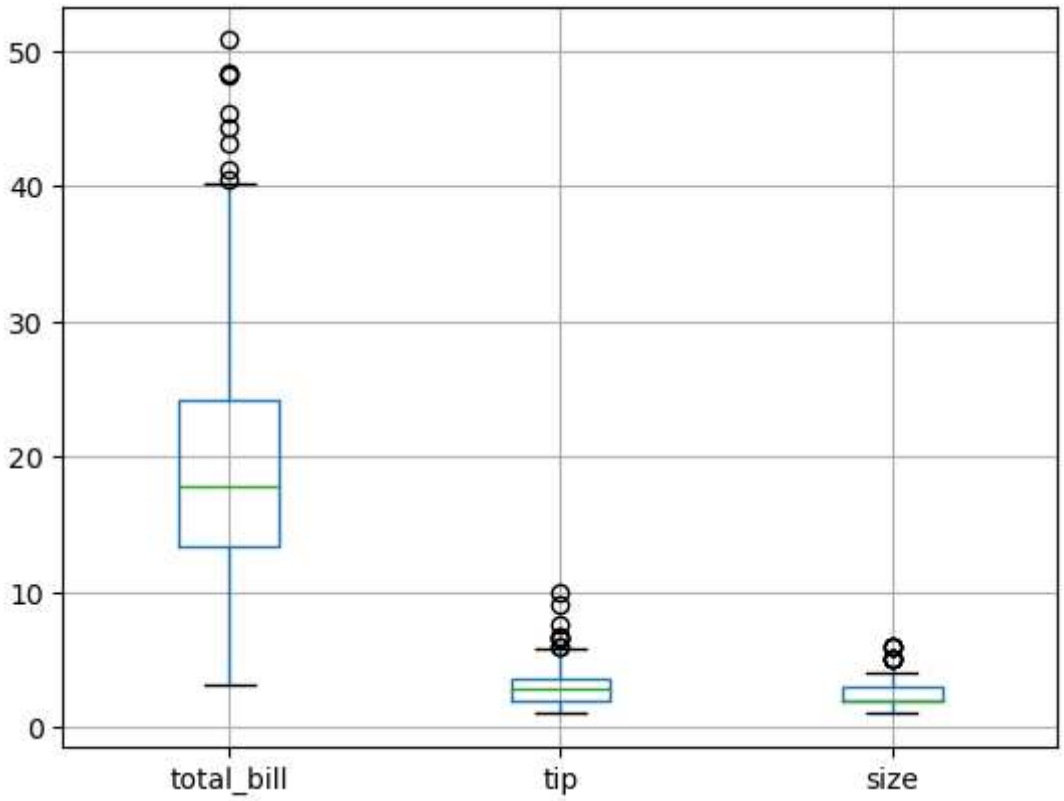
In [39]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['sex'] = le.fit_transform(df['sex'])
newdf=df
newdf
```

Out[39]:

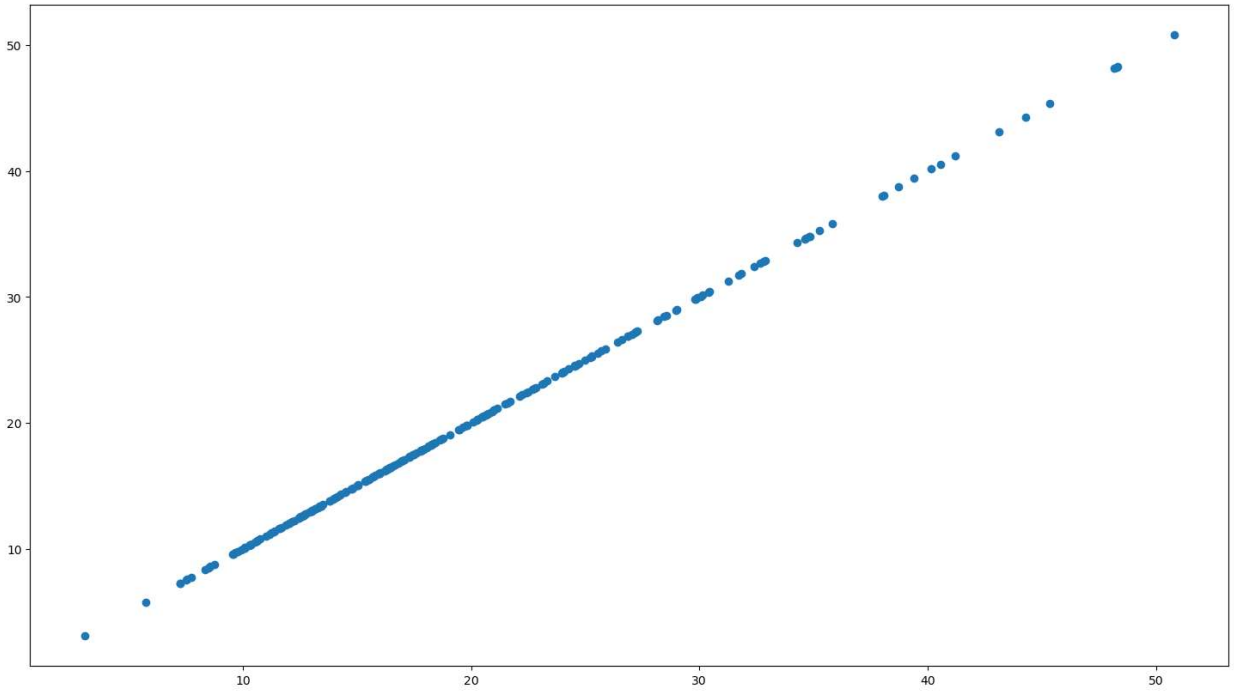
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	0	No	Sun	Dinner	2
1	10.34	1.66	1	No	Sun	Dinner	3
2	21.01	3.50	1	No	Sun	Dinner	3
3	23.68	3.31	1	No	Sun	Dinner	2
4	24.59	3.61	0	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	1	No	Sat	Dinner	3
240	27.18	2.00	0	Yes	Sat	Dinner	2
241	22.67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

244 rows × 7 columns

In [44]:
```
import matplotlib.pyplot as plt
col = ['total_bill','tip','size']
a=df.boxplot(col)
plt.show()
```



In [45]:
```
fig, ax = plt.subplots(figsize = (18,10))
ax.scatter(df['total_bill'], df['total_bill'])
plt.show()
```



In [48]:
```
import numpy as np
print(np.where((df['total_bill']<25) & (df['total_bill']>1)))
print(np.where((df['total_bill']>50) & (df['total_bill']<3)))
```

```
(array([  0,   1,   2,   3,   4,   6,   8,   9,  10,  12,  13,  14,  15,
        16,  17,  18,  19,  20,  21,  22,  24,  25,  26,  27,  28,  29,
        30,  31,  32,  33,  34,  35,  36,  37,  38,  40,  41,  42,  43,
        45,  46,  49,  50,  51,  53,  55,  58,  60,  61,  62,  63,  64,
        65,  66,  67,  68,  69,  70,  71,  74,  75,  76,  78,  79,  80,
        81,  82,  84,  86,  87,  88,  89,  91,  92,  93,  94,  97,  98,
        99, 100, 101, 103, 104, 105, 106, 108, 109, 110, 111, 113, 115,
       117, 118, 119, 120, 121, 122, 123, 124, 126, 127, 128, 129, 130,
       131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 144, 145, 146,
       147, 148, 149, 150, 151, 152, 153, 154, 158, 159, 160, 161, 162,
       163, 164, 165, 166, 168, 169, 171, 172, 174, 176, 177, 178, 181,
       183, 185, 186, 188, 189, 190, 191, 193, 194, 195, 196, 198, 199,
       200, 201, 202, 203, 204, 205, 208, 209, 213, 215, 217, 218, 220,
       221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233,
       234, 235, 236, 241, 242, 243], dtype=int64),)
(array([], dtype=int64),)
```

In [49]:
```
print(np.where(df['total_bill']>90))
print(np.where(df['sex']<25))
print(np.where(df['tip']<30))
```

```
(array([], dtype=int64),)
(array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,
        26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
        39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
        52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
        65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
        78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
        91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103,
       104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
       117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129,
       130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
       143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,
       156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168,
       169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181,
       182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194,
       195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,
       208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220,
       221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233,
       234, 235, 236, 237, 238, 239, 240, 241, 242, 243], dtype=int64),)
(array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,
        26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
        39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
        52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
        65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
        78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
        91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103,
       104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
       117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129,
       130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
       143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,
       156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168,
       169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181,
       182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194,
       195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,
       208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220,
       221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233,
       234, 235, 236, 237, 238, 239, 240, 241, 242, 243], dtype=int64),)
```

Practical no 3

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sb
```

In [2]:
```
sb.get_dataset_names()
```

Out[2]:
```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'dowjones',
 'exercise',
 'flights',
 'fmri',
 'geyser',
 'glue',
 'healthexp',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'seaice',
 'taxis',
 'tips',
 'titanic',
 'anagrams',
 'anagrams',
 'anscombe',
 'anscombe',
 'attention',
 'attention',
 'brain_networks',
 'brain_networks',
 'car_crashes',
 'car_crashes',
 'diamonds',
 'diamonds',
 'dots',
 'dots',
 'dowjones',
 'dowjones',
 'exercise',
 'exercise',
 'flights',
 'flights',
 'fmri',
 'fmri',
 'geyser',
 'geyser',
 'glue',
 'glue',
 'healthexp',
 'healthexp',
 'iris',
 'iris',
 'mpg',
 'mpg',
 'penguins',
 'penguins',
 'planets',
 'planets',
 'seaice',
 'seaice',
 'taxis',
```

```
 'taxis',
 'tips',
 'tips',
 'titanic',
 'titanic',
 'anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'dowjones',
 'exercise',
 'flights',
 'fmri',
 'geyser',
 'glue',
 'healthexp',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'seaice',
 'taxis',
 'tips',
 'titanic']
```

In [3]:
```
df = sb.load_dataset("iris")
```

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

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
...	...	...	...	...	...
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 [6]:
```
df.describe()
```

Out[6]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
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 [7]:
```
df.loc[:,'sepal_length'].mean()
```

Out[7]:
```
5.843333333333335
```

In [8]:
```
df.loc[:,'sepal_width'].mean()
```

Out[8]:
```
3.057333333333334
```

In [9]:
```
df.loc[:,'petal_length'].mean()
```

Out[9]:
```
3.7580000000000027
```

In [10]:
```
df.loc[:,'petal_width'].mean()
```

Out[10]:
```
1.199333333333334
```

In [11]:
```
df.loc[:,'sepal_length'].median()
```

Out[11]:
```
5.8
```

In [12]:
```
df.loc[:,'sepal_width'].median()
```

Out[12]:
```
3.0
```

In [13]:
```
df.loc[:,'petal_length'].median()
```

Out[13]:
```
4.35
```

In [14]:
```
df.loc[:,'petal_width'].median()
```

Out[14]:
```
1.3
```

In [15]:
```
df.loc[:,'sepal_length'].mode()
```

Out[15]:
```
0    5.0
Name: sepal_length, dtype: float64
```

In [16]:
```
df.loc[:,'sepal_width'].mode()
```

Out[16]:
```
0    3.0
Name: sepal_width, dtype: float64
```

In [17]:
```
df.loc[:,'petal_length'].mode()
```

Out[17]:
```
0    1.4
1    1.5
Name: petal_length, dtype: float64
```

In [18]:
```
df.loc[:,'petal_width'].mode()
```

Out[18]:
```
0    0.2
Name: petal_width, dtype: float64
```

In [19]:
```
df.loc[:,'petal_length'].std()
```

Out[19]:
```
1.7652982332594667
```

In [20]:
```
df.loc[:,'petal_width'].std()
```

Out[20]:
```
0.7622376689603465
```

In [21]:
```
df.loc[:,'sepal_length'].std()
```

Out[21]:
```
0.8280661279778629
```

In [23]:
```
df.loc[:,'sepal_width'].std()
```

Out[23]:
```
0.435866284936698
```

df.groupby(['species'])['sepal_length'].mean()

In [24]:
```
df.groupby(['species'])['sepal_length'].mean()
```

Out[24]:
```
species
setosa        5.006
versicolor    5.936
virginica     6.588
Name: sepal_length, dtype: float64
```

In [25]:
```
df.groupby(['species'])['sepal_width'].mean()
```

Out[25]:
```
species
setosa        3.428
versicolor    2.770
virginica     2.974
Name: sepal_width, dtype: float64
```

In [26]:
```
df.groupby(['species'])['petal_length'].mean()
```

Out[26]:
```
species
setosa        1.462
versicolor    4.260
virginica     5.552
Name: petal_length, dtype: float64
```

In [27]:
```
df.groupby(['species'])['petal_width'].mean()
```

Out[27]:
```
species
setosa        0.246
versicolor    1.326
virginica     2.026
Name: petal_width, dtype: float64
```

In [28]:
```
df.groupby(['species'])['sepal_length'].median()
```

Out[28]:
```
species
setosa        5.0
versicolor    5.9
virginica     6.5
Name: sepal_length, dtype: float64
```

In [29]:
```
df.groupby(['species'])['sepal_width'].median()
```

Out[29]:
```
species
setosa        3.4
versicolor    2.8
virginica     3.0
Name: sepal_width, dtype: float64
```

In [30]:
```
df.groupby(['species'])['petal_length'].median()
```

Out[30]:
```
species
setosa        1.50
versicolor    4.35
virginica     5.55
Name: petal_length, dtype: float64
```

In [31]:
```
df.groupby(['species'])['petal_width'].median()
```

Out[31]:
```
species
setosa        0.2
versicolor    1.3
virginica     2.0
Name: petal_width, dtype: float64
```

In [34]:
```
from sklearn import preprocessing
enc = preprocessing.OneHotEncoder()
enc_df = pd.DataFrame(enc.fit_transform(df[['species']]).toarray())
enc_df
```

Out[34]:

	0	1	2
0	1.0	0.0	0.0
1	1.0	0.0	0.0
2	1.0	0.0	0.0
3	1.0	0.0	0.0
4	1.0	0.0	0.0
...	...	...	...
145	0.0	0.0	1.0
146	0.0	0.0	1.0
147	0.0	0.0	1.0
148	0.0	0.0	1.0
149	0.0	0.0	1.0

150 rows × 3 columns

In [35]:
```
df101 = (df['species'] == 'setosa')
print(df101)
```

```
0       True
1       True
2       True
3       True
4       True
       ...
145    False
146    False
147    False
148    False
149    False
Name: species, Length: 150, dtype: bool
```

In [36]:
```
print("setosa")
print(df[df101].describe())
```

```
setosa
       sepal_length  sepal_width  petal_length  petal_width
count      50.00000    50.000000     50.000000    50.000000
mean        5.00600     3.428000      1.462000     0.246000
std         0.35249     0.379064      0.173664     0.105386
min         4.30000     2.300000      1.000000     0.100000
25%         4.80000     3.200000      1.400000     0.200000
50%         5.00000     3.400000      1.500000     0.200000
75%         5.20000     3.675000      1.575000     0.300000
max         5.80000     4.400000      1.900000     0.600000
```

In [37]:
```
df102 = (df['species'] == 'vericolor')
print("versicolor")
print(df[df102].describe())
```

```
versicolor
       sepal_length  sepal_width  petal_length  petal_width
count           0.0          0.0           0.0          0.0
mean            NaN          NaN           NaN          NaN
std             NaN          NaN           NaN          NaN
min             NaN          NaN           NaN          NaN
25%             NaN          NaN           NaN          NaN
50%             NaN          NaN           NaN          NaN
75%             NaN          NaN           NaN          NaN
max             NaN          NaN           NaN          NaN
```

In [38]:
```
df102 = (df['species'] == 'verginica')
print("verginica")
print(df[df102].describe())
```

```
verginica
       sepal_length  sepal_width  petal_length  petal_width
count           0.0          0.0           0.0          0.0
mean            NaN          NaN           NaN          NaN
std             NaN          NaN           NaN          NaN
min             NaN          NaN           NaN          NaN
25%             NaN          NaN           NaN          NaN
50%             NaN          NaN           NaN          NaN
75%             NaN          NaN           NaN          NaN
max             NaN          NaN           NaN          NaN
```

In [40]:
```
#Pratical No 4
```

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

In [2]:
```
x=np.array([95,85,80,70,60])
y=np.array([85,95,70,65,70])
```

In [3]:
```
model= np.polyfit(x, y, 1)
```

In [4]:
```
model
```

Out[4]:
```
array([ 0.64383562, 26.78082192])
```

In [5]:
```
predict = np.poly1d(model)
predict(65)
```

Out[5]:
```
68.63013698630137
```

In [6]:
```
y_pred= predict(x)
y_pred
```

Out[6]:
```
array([87.94520548, 81.50684932, 78.28767123, 71.84931507, 65.4109589 ])
```

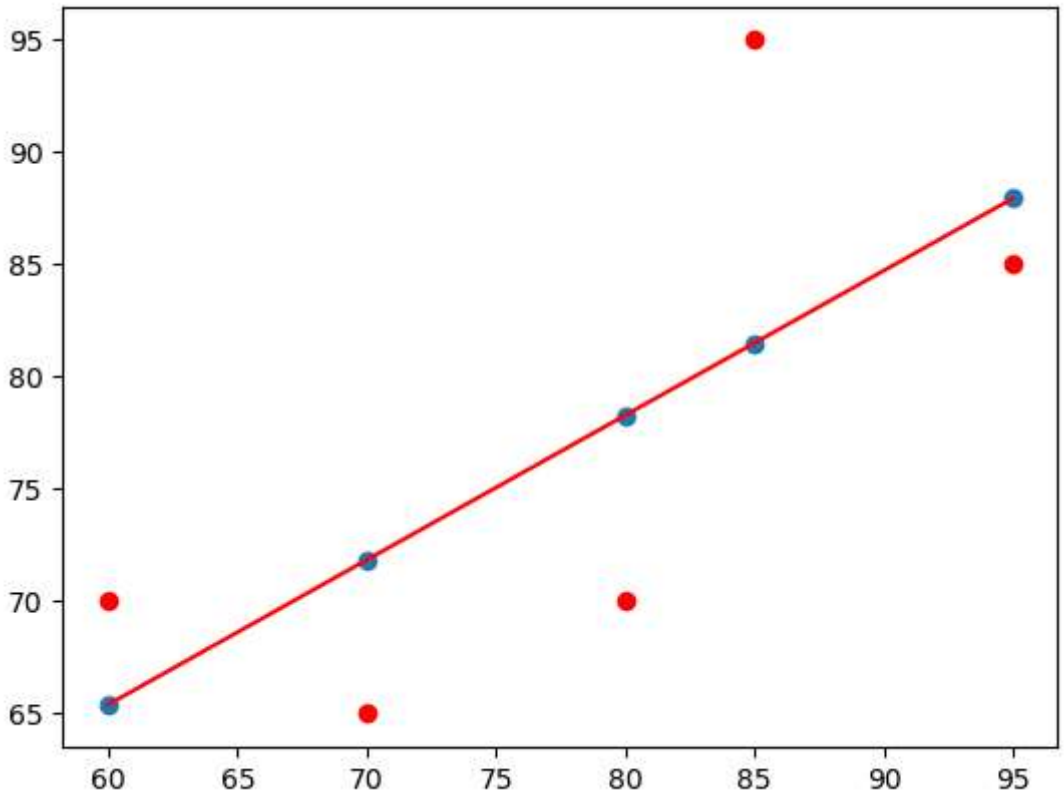
In [7]:
```
from sklearn.metrics import r2_score
r2_score(y, y_pred)
```

Out[7]:
```
0.4803218090889326
```

In [8]:
```
y_line = model[1] + model[0]* x
```

```
plt.plot(x, y_line, c = 'r')
plt.scatter(x, y_pred)
plt.scatter(x,y,c='r')
```

Out[8]: <matplotlib.collections.PathCollection at 0x189e3b01700>



In [9]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [10]:
```
from sklearn.datasets import fetch_california_housing
housing = fetch_california_housing()
```

In [11]:
```
data = pd.DataFrame(housing.data)
```

In [12]:
```
data.columns = housing.feature_names
data.head()
```

Out[12]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25

In [13]:
```
data['PRICE'] = housing.target
```

In [14]:
```
data.isnull().sum()
```

Out[14]:
```
MedInc        0
HouseAge      0
AveRooms      0
AveBedrms     0
Population    0
AveOccup      0
Latitude      0
Longitude     0
PRICE         0
dtype: int64
```

In [15]:
```
x = data.drop(['PRICE'], axis = 1)
y = data['PRICE']
```

In [18]:
```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size =0.2,random_state = 0)
```

In [19]:
```
import sklearn
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
model=lm.fit(xtrain, ytrain)
```

In [20]:
```
ytrain_pred = lm.predict(xtrain)
ytest_pred = lm.predict(xtest)
```

In [21]:
```
df=pd.DataFrame(ytrain_pred,ytrain)
df=pd.DataFrame(ytest_pred,ytest)
```

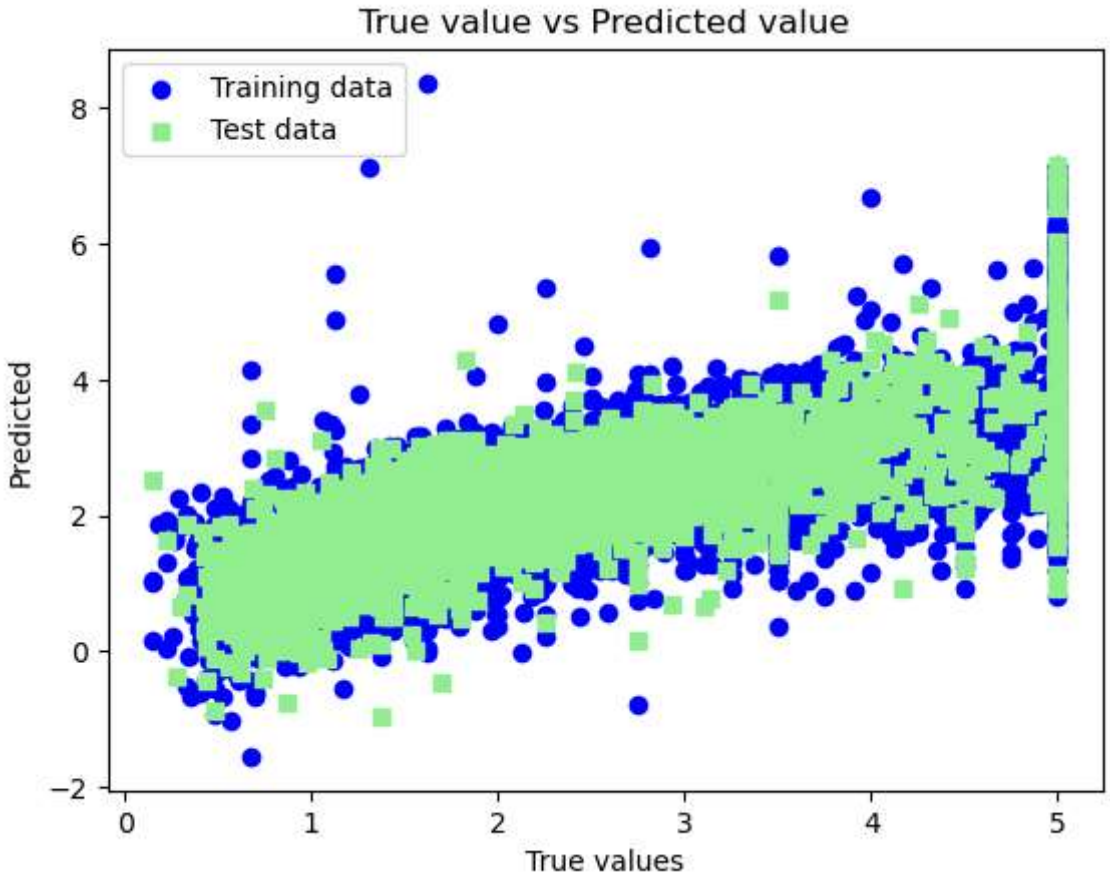
In [22]:
```
from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(ytest, ytest_pred)
print(mse)
mse = mean_squared_error(ytrain_pred,ytrain)
print(mse)
```

```
0.5289841670367192
0.5234413607125448
```

In [23]:
```
mse = mean_squared_error(ytest, ytest_pred)
print(mse)
```

```
0.5289841670367192
```

In [24]:
```
plt.scatter(ytrain ,ytrain_pred,c='blue',marker='o',label='Training data')
plt.scatter(ytest,ytest_pred ,c='lightgreen',marker='s',label='Test data')
plt.xlabel('True values')
plt.ylabel('Predicted')
plt.title("True value vs Predicted value")
plt.legend(loc= 'upper left')
#plt.hlines(y=0,xmin=0,xmax=50)
plt.plot()
plt.show()
```



In [25]:
```
#Pratical_5
```

In [26]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [32]:
```
data=pd.read_csv("C:\\Users\\System21\\Downloads\\diabetes.csv")
```

In [33]:
```
data
```

Out[33]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Ag
0	6	148	72	35	0	33.6	0.627	5
1	1	85	66	29	0	26.6	0.351	3
2	8	183	64	0	0	23.3	0.672	3
3	1	89	66	23	94	28.1	0.167	2
4	0	137	40	35	168	43.1	2.288	3
...	...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.171	6
764	2	122	70	27	0	36.8	0.340	2
765	5	121	72	23	112	26.2	0.245	3
766	1	126	60	0	0	30.1	0.349	4
767	1	93	70	31	0	30.4	0.315	2

768 rows × 9 columns

In [34]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [35]:
```
data.describe()
```

Out[35]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigr
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In [36]:
```
d
```

Out[36]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

In [37]:
```
data.tail()
```

Out[37]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Ag
763	10	101	76	48	180	32.9	0.171	6
764	2	122	70	27	0	36.8	0.340	2
765	5	121	72	23	112	26.2	0.245	3
766	1	126	60	0	0	30.1	0.349	4
767	1	93	70	31	0	30.4	0.315	2

In [38]:
```
data["Outcome"].value_counts(normalize=True)
```

Out[38]:
```
0    0.651042
1    0.348958
Name: Outcome, dtype: float64
```

In [39]:
```
x=data.drop(["Outcome"],axis=1)
```

In [40]:
```
y=data["Outcome"]
```

In [41]:
```
x
```

Out[41]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Ag
0	6	148	72	35	0	33.6	0.627	5
1	1	85	66	29	0	26.6	0.351	3
2	8	183	64	0	0	23.3	0.672	3
3	1	89	66	23	94	28.1	0.167	2
4	0	137	40	35	168	43.1	2.288	3
...	...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.171	6
764	2	122	70	27	0	36.8	0.340	2
765	5	121	72	23	112	26.2	0.245	3
766	1	126	60	0	0	30.1	0.349	4
767	1	93	70	31	0	30.4	0.315	2

768 rows × 8 columns

In [42]:
```
y
```

Out[42]:
```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

In [43]:
```
from sklearn.model_selection import train_test_split
```

In [44]:
```
train_x,test_x,train_y,test_y=train_test_split(x,y,random_state=56)
```

In [45]:
```
train_x
```

Out[45]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Ag
536	0	105	90	0	0	29.6	0.197	
547	4	131	68	21	166	33.1	0.160	
307	0	137	68	14	148	24.8	0.143	
45	0	180	66	39	0	42.0	1.893	
196	1	105	58	0	0	24.3	0.187	
...	...	...	...	...	...	...	...	
235	4	171	72	0	0	43.6	0.479	
418	1	83	68	0	0	18.2	0.624	
192	7	159	66	0	0	30.4	0.383	
399	3	193	70	31	0	34.9	0.241	
484	0	145	0	0	0	44.2	0.630	

576 rows × 8 columns

In [46]:
```
test_x
```

Out[46]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Ag
123	5	132	80	0	0	26.8	0.186	
295	6	151	62	31	120	35.5	0.692	
370	3	173	82	48	465	38.4	2.137	
300	0	167	0	0	0	32.3	0.839	
155	7	152	88	44	0	50.0	0.337	
...	...	...	...	...	...	...	...	
443	8	108	70	0	0	30.5	0.955	
134	2	96	68	13	49	21.1	0.647	
181	0	119	64	18	92	34.9	0.725	
588	3	176	86	27	156	33.3	1.154	
737	8	65	72	23	0	32.0	0.600	

192 rows × 8 columns

In [47]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [48]:
```
scaler=MinMaxScaler()
scaler
```

Out[48]:
```
MinMaxScaler()
```

In [49]:
```
cols=train_x.columns
cols
```

Out[49]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age'],
      dtype='object')
```

In [50]:
```
train_x_scaled=scaler.fit_transform(train_x)
train_x_scaled
```

Out[50]:
```
array([[0.        , 0.53030303, 0.73770492, ..., 0.44113264, 0.05081127,
        0.41666667],
       [0.23529412, 0.66161616, 0.55737705, ..., 0.49329359, 0.03501281,
        0.11666667],
       [0.        , 0.69191919, 0.55737705, ..., 0.36959762, 0.02775406,
        0.        ],
       ...,
       [0.41176471, 0.8030303 , 0.54098361, ..., 0.45305514, 0.13023057,
        0.25      ],
       [0.17647059, 0.97474747, 0.57377049, ..., 0.52011923, 0.06959863,
        0.06666667],
       [0.        , 0.73232323, 0.        , ..., 0.65871833, 0.23569599,
        0.16666667]])
```

In [51]:
```
train_x_scaled=pd.DataFrame(train_x_scaled,columns=cols)
```

In [52]:
```
train_x_scaled
```

Out[52]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFuncti
0	0.000000	0.530303	0.737705	0.000000	0.000000	0.441133	0.0508
1	0.235294	0.661616	0.557377	0.212121	0.196217	0.493294	0.0350
2	0.000000	0.691919	0.557377	0.141414	0.174941	0.369598	0.0277
3	0.000000	0.909091	0.540984	0.393939	0.000000	0.625931	0.7749
4	0.058824	0.530303	0.475410	0.000000	0.000000	0.362146	0.0465
...	...	...	...	...	...	...	
571	0.235294	0.863636	0.590164	0.000000	0.000000	0.649776	0.1712
572	0.058824	0.419192	0.557377	0.000000	0.000000	0.271237	0.2331
573	0.411765	0.803030	0.540984	0.000000	0.000000	0.453055	0.1302
574	0.176471	0.974747	0.573770	0.313131	0.000000	0.520119	0.0695
575	0.000000	0.732323	0.000000	0.000000	0.000000	0.658718	0.2356

576 rows × 8 columns

In [53]:
```
from sklearn.linear_model import LogisticRegression as LogReg
```

In [54]:
```
logreg=LogReg()
```

In [55]:
```
logreg.fit(train_x,train_y)
```

Out[55]:
```
LogisticRegression()
```

In [56]:
```
train_predict=logreg.predict(train_x)
test_predict=logreg.predict(test_x)
```

In [57]:
```
train_predict
```

Out[57]:
```
array([0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0,
       1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 1, 1, 1], dtype=int64)
```

In [58]:
```
test_predict
```

Out[58]:
```
array([0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,
       0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0],
      dtype=int64)
```

In [59]:
```
from sklearn.metrics import f1_score, confusion_matrix,roc_auc_score,roc_curve
```

In [60]:
```
f1_score(train_predict,train_y)
```

Out[60]:
```
0.6304347826086956
```

In [61]:
```
f1_score(test_predict,test_y)
```

Out[61]:
```
0.7008547008547009
```

In [62]:
```
conf1=confusion_matrix(train_y,train_predict)
```

In [63]:
```
conf1
```

Out[63]:
```
array([[324,  42],
       [ 94, 116]], dtype=int64)
```

In [64]:
```
conf=confusion_matrix(test_y,test_predict)
```

In [65]:
```
conf
```

Out[65]:
```
array([[116,  18],
       [ 17,  41]], dtype=int64)
```

In [66]:
```
true_negative=conf[0][0]
false_negative=conf[1][0]
false_positive=conf[0][1]
true_positive=conf[1][1]
```

In [69]:
```
Accuracy = (true_positive + true_negative) / (true_positive +false_positive +false_neg
Accuracy
Precision=true_positive/(true_positive + false_positive)
Precision

Recall= true_positive/ (true_positive + false_negative)
Recall

F1_score= 2*(Recall * Precision)/(Recall + Precision)
F1_score
```

Out[69]:
```
0.7008547008547009
```

In [70]:
```
Accuracy
```

Out[70]:
```
2.0657894736842106
```

In [71]:
```
Precision
```

Out[71]:
```
0.6949152542372882
```

In [72]:
```
Recall
```

Out[72]:
```
0.7068965517241379
```

In [73]:
```
F1_score
```

Out[73]:
```
0.7008547008547009
```

In [74]:
```
auc_score= roc_auc_score(test_y,test_predict)
```

In [76]:
```
fpr,tpr,threasholds= roc_curve(test_y,test_predict)
```

In [77]:
```
threasholds
```

Out[77]:
```
array([2, 1, 0], dtype=int64)
```

In [78]:
```
plt.plot(fpr, tpr, color='orange', label='ROC')
plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--',label='ROC curve (a
```

```
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()
plt.show()
```

```
  File "C:\Users\System21\AppData\Local\Temp\ipykernel_8520\1031383496.py", line 2
    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--',label='ROC curve (a
                                                                                 ^
SyntaxError: EOL while scanning string literal
```

In [1]:
```
#Pratical_6
```

In [76]:
```
import pandas as pd
import seaborn as sns
```

In [77]:
```
df=sns.load_dataset("titanic")
df
```

Out[77]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True
...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True

891 rows × 15 columns

In [78]:
```
df = df.dropna()
df
```

Out[78]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	de
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	
6	0	1	male	54.0	0	0	51.8625	S	First	man	True	
10	1	3	female	4.0	1	1	16.7000	S	Third	child	False	
11	1	1	female	58.0	0	0	26.5500	S	First	woman	False	
...	...	...	...	...	...	...	...	...	...	...	...	
871	1	1	female	47.0	1	1	52.5542	S	First	woman	False	
872	0	1	male	33.0	0	0	5.0000	S	First	man	True	
879	1	1	female	56.0	0	1	83.1583	C	First	woman	False	
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	

182 rows × 15 columns

In [80]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
label = le.fit_transform(df['sex'])
label
```

Out[80]:
```
array([0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1,
       0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 1, 0, 0, 1])
```

In [81]:
```
df.isnull()
```

Out[81]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	en
1	False	False	False	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	False	False	
6	False	False	False	False	False	False	False	False	False	False	False	False	
10	False	False	False	False	False	False	False	False	False	False	False	False	
11	False	False	False	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	...	
871	False	False	False	False	False	False	False	False	False	False	False	False	
872	False	False	False	False	False	False	False	False	False	False	False	False	
879	False	False	False	False	False	False	False	False	False	False	False	False	
887	False	False	False	False	False	False	False	False	False	False	False	False	
889	False	False	False	False	False	False	False	False	False	False	False	False	

182 rows × 15 columns

In [85]:
```
a=df.drop(columns=["sex","alive","class","embarked","who","adult_male","deck","embark_
a
```

Out[85]:

	survived	pclass	age	sibsp	parch	fare
1	1	1	38.0	1	0	71.2833
3	1	1	35.0	1	0	53.1000
6	0	1	54.0	0	0	51.8625
10	1	3	4.0	1	1	16.7000
11	1	1	58.0	0	0	26.5500
...	...	...	...	...	...	...
871	1	1	47.0	1	1	52.5542
872	0	1	33.0	0	0	5.0000
879	1	1	56.0	0	1	83.1583
887	1	1	19.0	0	0	30.0000
889	1	1	26.0	0	0	30.0000

182 rows × 6 columns

In [86]:
```
x = a.iloc[:, [2, 3]].values
y = a.iloc[:, 4].values
```

In [89]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25,random_stat
```

In [90]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [91]:
```
from sklearn.naive_bayes import GaussianNB
gaussian = GaussianNB()
gaussian.fit(x_train, y_train)
```

Out[91]:
```
GaussianNB()
```

In [92]:
```
y_pred = gaussian.predict(x_test)
y_pred
```

Out[92]:
```
array([0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2,
       0, 0], dtype=int64)
```

In [93]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

Out[93]:
```
array([[31,  0,  0],
       [10,  0,  1],
       [ 2,  1,  1]], dtype=int64)
```

In [94]:
```
true_negative =cm[0][0]
false_negative =cm[1][0]
false_positive =cm[0][1]
true_positive =cm[1][1]
```

In [95]:
```
Accuracy = (true_positive + true_negative) / (true_positive +false_positive)
Accuracy
```

Out[95]:
```
inf
```

In [96]:
```
Precision = true_positive/(true_positive+false_positive)
Precision
```

Out[96]:
```
nan
```

In [97]:
```
Recall = true_positive/(true_positive+false_negative)
Recall
```

Out[97]:
```
0.0
```

In [ ]:
```

```

In [8]:
```
#Pratical_7
```

In [9]:
```
import nltk
```

In [10]:
```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\System21\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\System21\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\System21\AppData\Roaming\nltk_data...
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\System21\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping taggers\averaged_perceptron_tagger.zip.
```

Out[10]:
```
True
```

In [11]:
```
from nltk.tokenize import sent_tokenize
tokenized_text= sent_tokenize(text)
print(tokenized_text)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_7236\306484814.py in <module>
      1 from nltk.tokenize import sent_tokenize
----> 2 tokenized_text= sent_tokenize(text)
      3 print(tokenized_text)

NameError: name 'text' is not defined
```

In [12]:
```
True
```

Out[12]:
```
True
```

In [14]:
```
text="Tokenization is the first step in text analytics.The process of breaking down a
```

In [15]:
```
from nltk.tokenize import sent_tokenize
tokenized_text= sent_tokenize(text)
print(tokenized_text)
```

```
['Tokenization is the first step in text analytics.The process of breaking down a tex
t paragraph into smaller chunks such as words or sentences is called Tokenization.']
```

In [16]:
```
from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)
```

```
['Tokenization', 'is', 'the', 'first', 'step', 'in', 'text', 'analytics.The', 'proces
s', 'of', 'breaking', 'down', 'a', 'text', 'paragraph', 'into', 'smaller', 'chunks',
'such', 'as', 'words', 'or', 'sentences', 'is', 'called', 'Tokenization', '.']
```

In [17]:
```
from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))
print(stop_words)
```

```
{'yours', 'were', 'than', 'and', 'me', 'further', 'before', 'the', 'that', 'his', 'mo
re", "hadn't", "mustn't", "haven't", 'shouldn', 'we', 'was', 'or', 'down', 'our', 'ou
t', 'from', 'hers', 's', 'because', 'during', 'all', 'most', 'am', 'themselves', 'aft
er', 'myself', 'are', 'on', 'having', 'isn', 'should', "won't", 'been', 'you', 'won',
"you'll", 'my', 'didn', 've', 'no', 'if', "you'd", 'into', 'to', 'can', "that'll", 'f
or', 'yourselves', 'haven', 'y', "isn't", 'doing', 'who', 'm', 'off', 'ourselves', 'i
n', 'being', 'again', 'any', 'it', 'so', 'is', "mightn't", 'mightn', "doesn't", 'suc
h', 'between', 'with', "don't", 'will', 'now', 'your', 'about', 'over', 'ain', "were
n't", 'as', 'its', 'does', 'same', 'their', 'under', 'how', 'these', 'them', "shoul
d've", 'him', 'some', "shan't", 'couldn', 'she', "she's", 'hadn', 'those', 'what', 'a
n', 'wouldn', 'he', 'itself', 'there', 'o', 'doesn', 'wasn', 'only', 'both', 'up', 'l
l', 'when', 'do', 'own', 'here', 'of', 'himself', 'nor', 'just', "didn't", 'have',
'i', 'd', 'herself', 'not', 'a', 'her', 'above', 'shan', 'be', 'needn', "wasn't", 'mu
stn', 'ma', 'once', "needn't", 'has', 'at', 'other', "aren't", 'did', 'against', "i
t's", 't', 'aren', 'they', 'where', "couldn't", 'through', 'then', 'this', 'very', 'f
ew', 'while', 're', 'by', "you're", 'whom', 'weren', "shouldn't", "wouldn't", 'too',
'yourself', 'but', 'below', 'ours', 'until', 'each', "you've", 'theirs', 'had', 'do
n', 'which', 'hasn', "hasn't", 'why'}
```

In [18]:
```
import re
```

In [20]:
```
text= "How to remove stop words with NLTK library in Python?"
text= re.sub('[^a-zA-Z]', ' ',text)
tokens = word_tokenize(text.lower())
filtered_text=[]
for w in tokens:
 if w not in stop_words:
     filtered_text.append(w)
 print("Tokenized Sentence:",tokens)
 print("Filterd Sentence:",filtered_text)
```

```
Tokenized Sentence: ['how', 'to', 'remove', 'stop', 'words', 'with', 'nltk', 'librar
y', 'in', 'python']
Filterd Sentence: []
Tokenized Sentence: ['how', 'to', 'remove', 'stop', 'words', 'with', 'nltk', 'librar
y', 'in', 'python']
Filterd Sentence: []
Tokenized Sentence: ['how', 'to', 'remove', 'stop', 'words', 'with', 'nltk', 'librar
y', 'in', 'python']
Filterd Sentence: ['remove']
Tokenized Sentence: ['how', 'to', 'remove', 'stop', 'words', 'with', 'nltk', 'librar
y', 'in', 'python']
Filterd Sentence: ['remove', 'stop']
Tokenized Sentence: ['how', 'to', 'remove', 'stop', 'words', 'with', 'nltk', 'librar
y', 'in', 'python']
Filterd Sentence: ['remove', 'stop', 'words']
Tokenized Sentence: ['how', 'to', 'remove', 'stop', 'words', 'with', 'nltk', 'librar
y', 'in', 'python']
Filterd Sentence: ['remove', 'stop', 'words']
Tokenized Sentence: ['how', 'to', 'remove', 'stop', 'words', 'with', 'nltk', 'librar
y', 'in', 'python']
Filterd Sentence: ['remove', 'stop', 'words', 'nltk']
Tokenized Sentence: ['how', 'to', 'remove', 'stop', 'words', 'with', 'nltk', 'librar
y', 'in', 'python']
Filterd Sentence: ['remove', 'stop', 'words', 'nltk', 'library']
Tokenized Sentence: ['how', 'to', 'remove', 'stop', 'words', 'with', 'nltk', 'librar
y', 'in', 'python']
Filterd Sentence: ['remove', 'stop', 'words', 'nltk', 'library']
Tokenized Sentence: ['how', 'to', 'remove', 'stop', 'words', 'with', 'nltk', 'librar
y', 'in', 'python']
Filterd Sentence: ['remove', 'stop', 'words', 'nltk', 'library', 'python']
```

In [21]:
```
from nltk.stem import PorterStemmer
e_words= ["wait", "waiting", "waited", "waits"]
ps =PorterStemmer()
for w in e_words:
 rootWord=ps.stem(w)
 print(rootWord)
```

```
wait
wait
wait
wait
```

In [23]:
```
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
text = "studies studying cries cry"
tokenization = nltk.word_tokenize(text)
for w in tokenization:
     print("Lemma for {} is {}".format(w, wordnet_lemmatizer.lemmatize(w)))
```

```
---------------------------------------------------------------------------
LookupError                               Traceback (most recent call last)
~\anaconda3\lib\site-packages\nltk\corpus\util.py in __load(self)
     83                 try:
---> 84                     root = nltk.data.find(f"{self.subdir}/{zip_name}")
     85                 except LookupError:

~\anaconda3\lib\site-packages\nltk\data.py in find(resource_name, paths)
    582     resource_not_found = f"\n{sep}\n{msg}\n{sep}\n"
--> 583     raise LookupError(resource_not_found)
    584

LookupError:
**********************************************************************
  Resource omw-1.4 not found.
  Please use the NLTK Downloader to obtain the resource:

  >>> import nltk
  >>> nltk.download('omw-1.4')

  For more information see: https://www.nltk.org/data.html

  Attempted to load corpora/omw-1.4.zip/omw-1.4/

  Searched in:
    - 'C:\\Users\\System21/nltk_data'
    - 'C:\\Users\\System21\\anaconda3\\nltk_data'
    - 'C:\\Users\\System21\\anaconda3\\share\\nltk_data'
    - 'C:\\Users\\System21\\anaconda3\\lib\\nltk_data'
    - 'C:\\Users\\System21\\AppData\\Roaming\\nltk_data'
    - 'C:\\nltk_data'
    - 'D:\\nltk_data'
    - 'E:\\nltk_data'
**********************************************************************


During handling of the above exception, another exception occurred:

LookupError                               Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_7236\3682176561.py in <module>
      4 tokenization = nltk.word_tokenize(text)
      5 for w in tokenization:
----> 6       print("Lemma for {} is {}".format(w, wordnet_lemmatizer.lemmatize(w)))

~\anaconda3\lib\site-packages\nltk\stem\wordnet.py in lemmatize(self, word, pos)
     43         :return: The lemma of `word`, for the given `pos`.
     44         """
---> 45         lemmas = wn._morphy(word, pos)
     46         return min(lemmas, key=len) if lemmas else word
     47

~\anaconda3\lib\site-packages\nltk\corpus\util.py in __getattr__(self, attr)
    119             raise AttributeError("LazyCorpusLoader object has no attribute '_
_bases__'")
    120
--> 121         self.__load()
    122         # This looks circular, but its not, since __load() changes our
    123         # __class__ to something new:

~\anaconda3\lib\site-packages\nltk\corpus\util.py in __load(self)
```

```
     87 
     88         # Load the corpus.
---> 89         corpus = self.__reader_cls(root, *self.__args, **self.__kwargs)
     90 
     91         # This is where the magic happens!  Transform ourselves into

~\anaconda3\lib\site-packages\nltk\corpus\reader\wordnet.py in __init__(self, root, o
mw_reader)
   1174             )
   1175         else:
-> 1176             self.provenances = self.omw_prov()
   1177 
   1178         # A cache to store the wordnet data of multiple languages

~\anaconda3\lib\site-packages\nltk\corpus\reader\wordnet.py in omw_prov(self)
   1283         provdict = {}
   1284         provdict["eng"] = ""
-> 1285         fileids = self._omw_reader.fileids()
   1286         for fileid in fileids:
   1287             prov, langfile = os.path.split(fileid)

~\anaconda3\lib\site-packages\nltk\corpus\util.py in __getattr__(self, attr)
    119             raise AttributeError("LazyCorpusLoader object has no attribute '_
_bases__'")
    120 
--> 121         self.__load()
    122         # This looks circular, but its not, since __load() changes our
    123         # __class__ to something new:

~\anaconda3\lib\site-packages\nltk\corpus\util.py in __load(self)
     84                     root = nltk.data.find(f"{self.subdir}/{zip_name}")
     85                 except LookupError:
---> 86                     raise e
     87 
     88         # Load the corpus.

~\anaconda3\lib\site-packages\nltk\corpus\util.py in __load(self)
     79         else:
     80             try:
---> 81                 root = nltk.data.find(f"{self.subdir}/{self.__name}")
     82             except LookupError as e:
     83                 try:

~\anaconda3\lib\site-packages\nltk\data.py in find(resource_name, paths)
    581     sep = "*" * 70
    582     resource_not_found = f"\n{sep}\n{msg}\n{sep}\n"
--> 583     raise LookupError(resource_not_found)
    584 
    585 

LookupError: 
**********************************************************************
  Resource omw-1.4 not found.
  Please use the NLTK Downloader to obtain the resource:

  >>> import nltk
  >>> nltk.download('omw-1.4')

  For more information see: https://www.nltk.org/data.html

  Attempted to load corpora/omw-1.4
```

```
  Searched in:
    - 'C:\\Users\\System21/nltk_data'
    - 'C:\\Users\\System21\\anaconda3\\nltk_data'
    - 'C:\\Users\\System21\\anaconda3\\share\\nltk_data'
    - 'C:\\Users\\System21\\anaconda3\\lib\\nltk_data'
    - 'C:\\Users\\System21\\AppData\\Roaming\\nltk_data'
    - 'C:\\nltk_data'
    - 'D:\\nltk_data'
    - 'E:\\nltk_data'
**********************************************************************
```

In [24]:
```
import nltk
from nltk.tokenize import word_tokenize
data="The pink sweater fit her perfectly"
words=word_tokenize(data)
for word in words:
 print(nltk.pos_tag([word]))
```

```
[('The', 'DT')]
[('pink', 'NN')]
[('sweater', 'NN')]
[('fit', 'NN')]
[('her', 'PRP$')]
[('perfectly', 'RB')]
```

In [25]:
```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [26]:
```
documentA = 'Jupiter is the largest Planet'
documentB = 'Mars is the fourth planet from the Sun'
```

In [27]:
```
bagOfWordsA = documentA.split(' ')
bagOfWordsB = documentB.split(' ')
```

In [28]:
```
uniqueWords = set(bagOfWordsA).union(set(bagOfWordsB))
```

In [29]:
```
numOfWordsA = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsA:
 numOfWordsA[word] += 1
 numOfWordsB = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsB:
 numOfWordsB[word] += 1
```

In [31]:
```
def computeTF(wordDict, bagOfWords):
 tfDict = {}
 bagOfWordsCount = len(bagOfWords)
 for word, count in wordDict.items():
     tfDict[word] = count / float(bagOfWordsCount)
 return tfDict
tfA = computeTF(numOfWordsA, bagOfWordsA)
tfB = computeTF(numOfWordsB, bagOfWordsB)
```

In [35]:
```
def computeIDF(documents):
 import math
 N = len(documents)
 idfDict = dict.fromkeys(documents[0].keys(), 0)
 for document in documents:
         for word, val in document.items():
```

```
                if val > 0:
                    idfDict[word] += 1
 for word, val in idfDict.items():
     idfDict[word] = math.log(N / float(val))
 return idfDict
idfs = computeIDF([numOfWordsA, numOfWordsB])
idfs
```

Out[35]:
```
{'largest': 0.6931471805599453,
 'is': 0.0,
 'fourth': 0.6931471805599453,
 'Mars': 0.6931471805599453,
 'the': 0.0,
 'Planet': 0.6931471805599453,
 'planet': 0.6931471805599453,
 'Jupiter': 0.6931471805599453,
 'Sun': 0.6931471805599453,
 'from': 0.6931471805599453}
```

In [39]:
```
def computeTFIDF(tfBagOfWords, idfs):
 tfidf = {}
 for word, val in tfBagOfWords.items():
         tfidf[word] = val * idfs[word]
         return tfidf
tfidfA = computeTFIDF(tfA, idfs)
tfidfB = computeTFIDF(tfB, idfs)
df = pd.DataFrame([tfidfA, tfidfB])
df
```

Out[39]:

	largest
0	0.138629
1	0.000000

In [40]:
```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
import math
```

In [41]:
```
documentA = 'Jupiter is the largest Planet'
documentB = 'Mars is the fourth planet from the sun'
```

In [42]:
```
bagOfWordsA = documentA.split(' ')
bagOfWordsB = documentB.split(' ')
```

In [43]:
```
uniqueWords = set(bagOfWordsA).union(set(bagOfWordsB))
```

In [44]:
```
numOfWordsA = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsA:
 numOfWordsA[word] += 1
numOfWordsB = dict.fromkeys(uniqueWords, 0)
for word in bagOfWordsB:
 numOfWordsB[word] += 1
```

In [46]:
```
def computeTF(wordDict, bagOfWords):
 tfDict = {}
 bagOfWordsCount = len(bagOfWords)
 for word, count in wordDict.items():
```

```
    tfDict[word] = count/float(bagOfWordsCount)
 return tfDict
```

In [47]:
```
tfA = computeTF(numOfWordsA, bagOfWordsA)
tfB = computeTF(numOfWordsB, bagOfWordsB)
tfA
```

Out[47]:
```
{'largest': 0.2,
 'is': 0.2,
 'fourth': 0.0,
 'Mars': 0.0,
 'the': 0.2,
 'Planet': 0.2,
 'sun': 0.0,
 'planet': 0.0,
 'Jupiter': 0.2,
 'from': 0.0}
```

In [59]:
```
def computeIDF(documents):
 N = len(documents)
 idfDict = dict.fromkeys(documents[0].keys(),0)
 for document in documents:
     for word, val in document.items():
             if val > 0:
                     idfDict[word] += 1
 for word, val in idfDict.items():
     idfDict[word] = math.log(N/float(val))
     return idfDict
```

In [60]:
```
idfs = computeIDF([numOfWordsA, numOfWordsB])
idfs
```

Out[60]:
```
{'largest': 0.6931471805599453,
 'is': 2,
 'fourth': 1,
 'Mars': 1,
 'the': 2,
 'Planet': 1,
 'sun': 1,
 'planet': 1,
 'Jupiter': 1,
 'from': 1}
```

In [61]:
```
 #Pratical_8
```

In [62]:
```
pip install seaborn
```

```
Requirement already satisfied: seaborn in c:\users\system21\anaconda3\lib\site-packag
es (0.11.2)Note: you may need to restart the kernel to use updated packages.

Requirement already satisfied: scipy>=1.0 in c:\users\system21\anaconda3\lib\site-pac
kages (from seaborn) (1.9.1)
Requirement already satisfied: numpy>=1.15 in c:\users\system21\anaconda3\lib\site-pa
ckages (from seaborn) (1.21.5)
Requirement already satisfied: matplotlib>=2.2 in c:\users\system21\anaconda3\lib\sit
e-packages (from seaborn) (3.5.2)
Requirement already satisfied: pandas>=0.23 in c:\users\system21\anaconda3\lib\site-p
ackages (from seaborn) (1.4.4)
Requirement already satisfied: pillow>=6.2.0 in c:\users\system21\anaconda3\lib\site-
packages (from matplotlib>=2.2->seaborn) (9.2.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\system21\anaconda3\lib\s
ite-packages (from matplotlib>=2.2->seaborn) (1.4.2)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\system21\anaconda3\lib\s
ite-packages (from matplotlib>=2.2->seaborn) (4.25.0)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\system21\anaconda3\lib\si
te-packages (from matplotlib>=2.2->seaborn) (3.0.9)
Requirement already satisfied: cycler>=0.10 in c:\users\system21\anaconda3\lib\site-p
ackages (from matplotlib>=2.2->seaborn) (0.11.0)
Requirement already satisfied: packaging>=20.0 in c:\users\system21\anaconda3\lib\sit
e-packages (from matplotlib>=2.2->seaborn) (21.3)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\system21\anaconda3\li
b\site-packages (from matplotlib>=2.2->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\system21\anaconda3\lib\site-p
ackages (from pandas>=0.23->seaborn) (2022.1)
Requirement already satisfied: six>=1.5 in c:\users\system21\anaconda3\lib\site-packa
ges (from python-dateutil>=2.7->matplotlib>=2.2->seaborn) (1.16.0)
```

In [64]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
dataset = sns.load_dataset('titanic')
dataset.head()
```

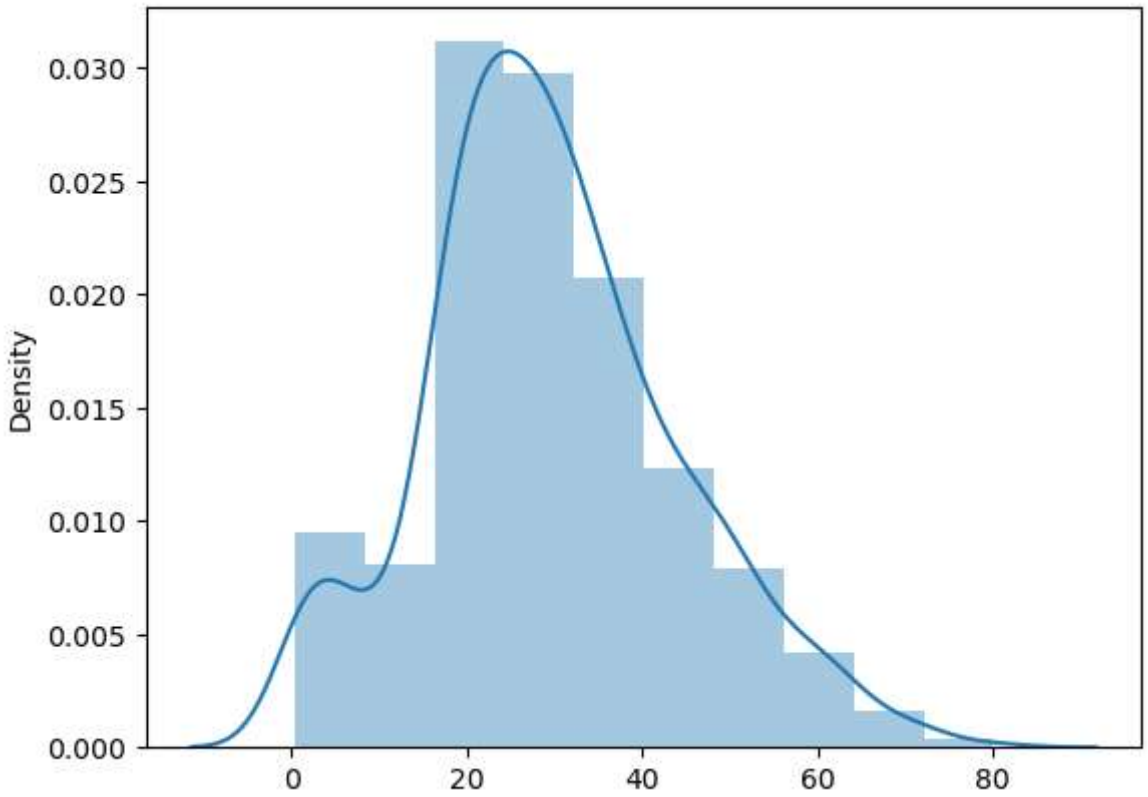
Out[64]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN

In [65]:
```
import seaborn as sns
sns.distplot(x = dataset['age'], bins = 10)
```

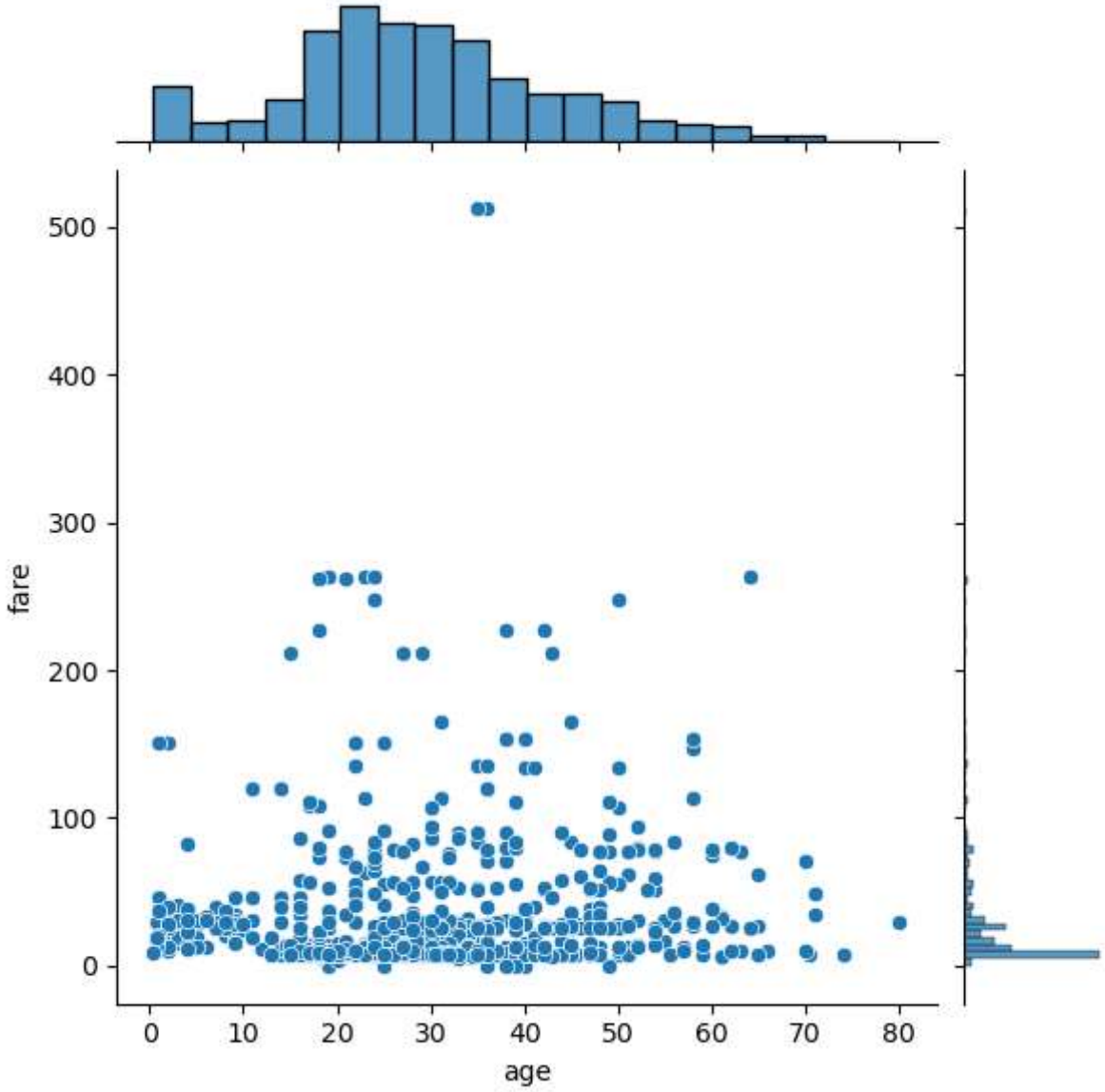
```
C:\Users\System21\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWa
rning: `distplot` is a deprecated function and will be removed in a future version. P
lease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[65]:
```
<AxesSubplot:ylabel='Density'>
```



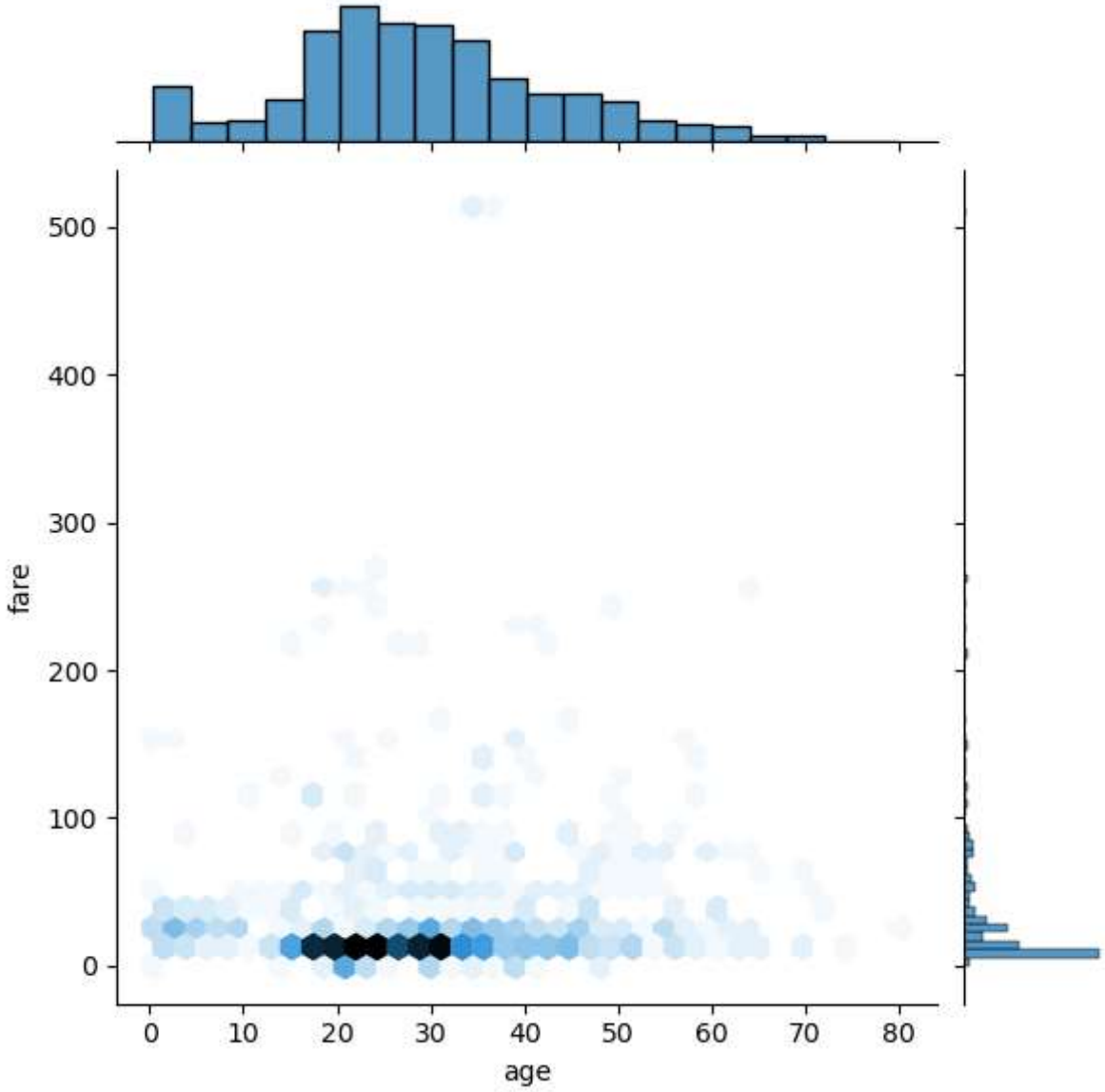
In [66]:
```
import seaborn as sns
sns.jointplot(x = dataset['age'], y = dataset['fare'], kind = 'scatter')
```

Out[66]:
```
<seaborn.axisgrid.JointGrid at 0x20f62fa7580>
```



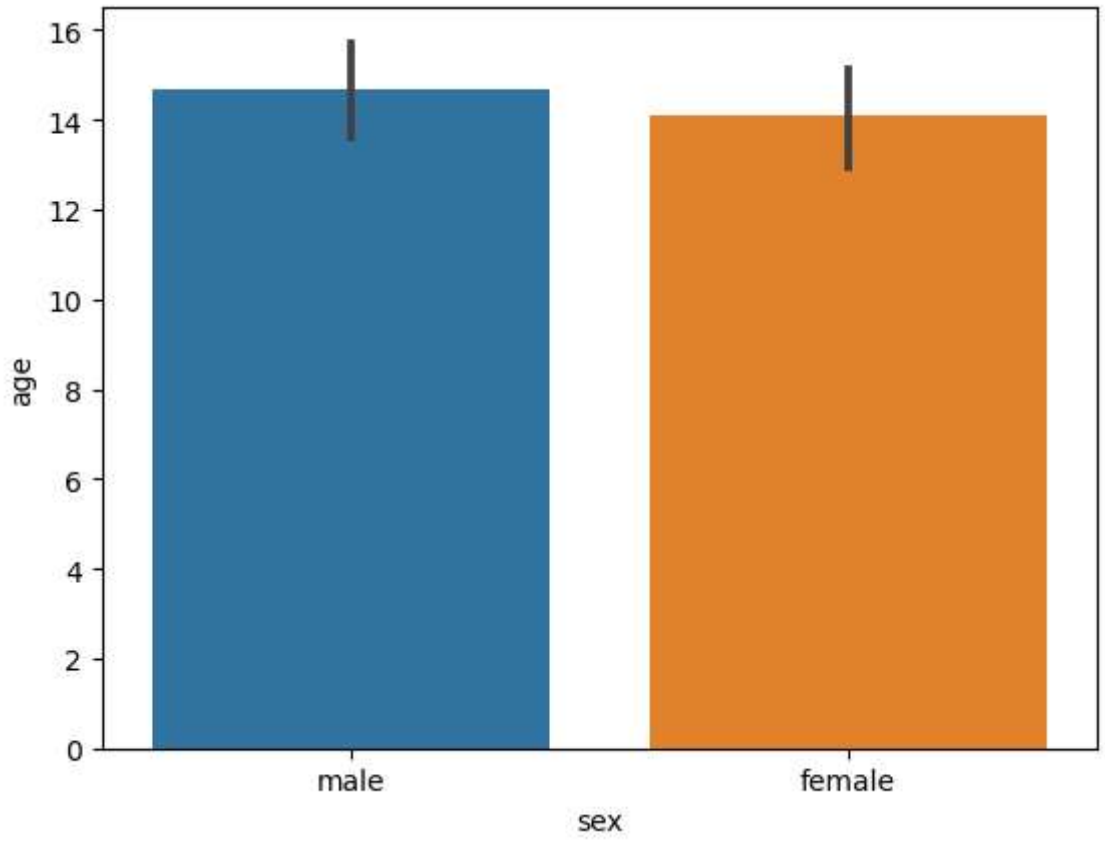
In [67]:
```
sns.jointplot(x = dataset['age'], y = dataset['fare'], kind = 'hex')
```

Out[67]:
```
<seaborn.axisgrid.JointGrid at 0x20f62fb5ee0>
```



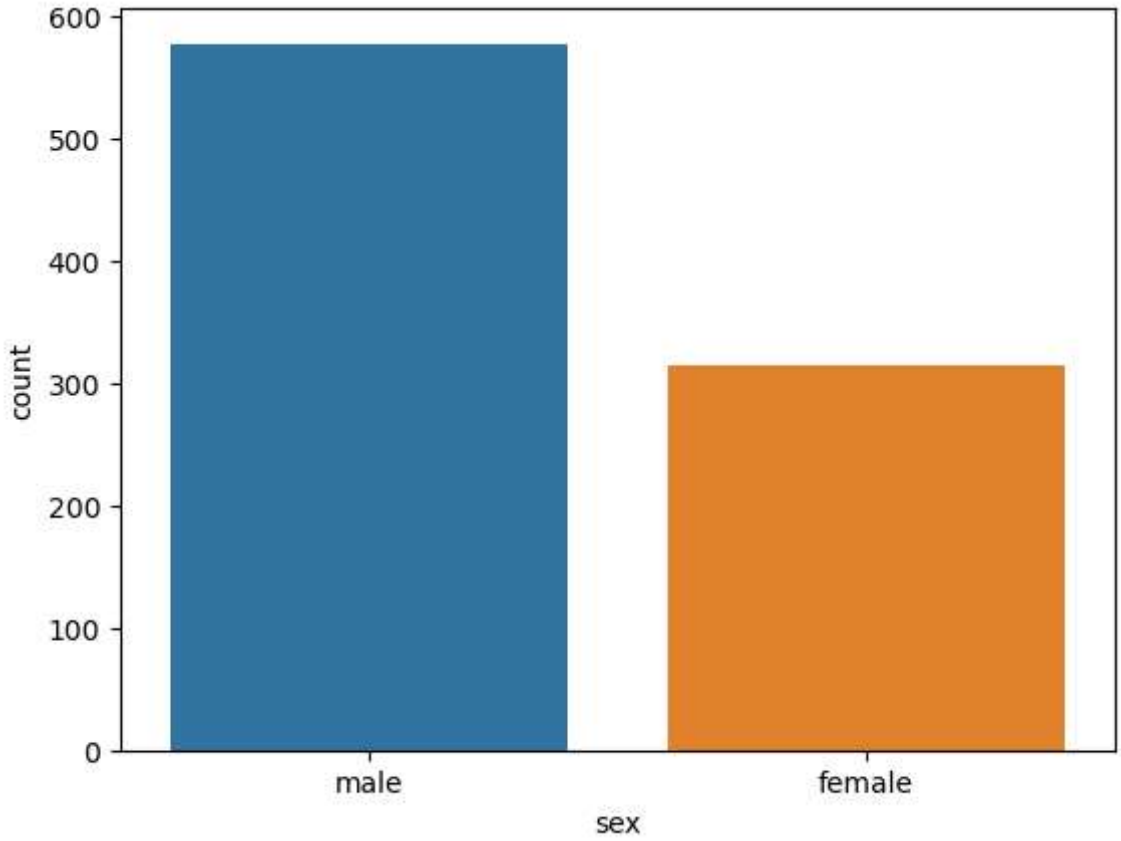
In [68]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.barplot(x='sex', y='age', data=dataset, estimator=np.std)
```

Out[68]:
```
<AxesSubplot:xlabel='sex', ylabel='age'>
```



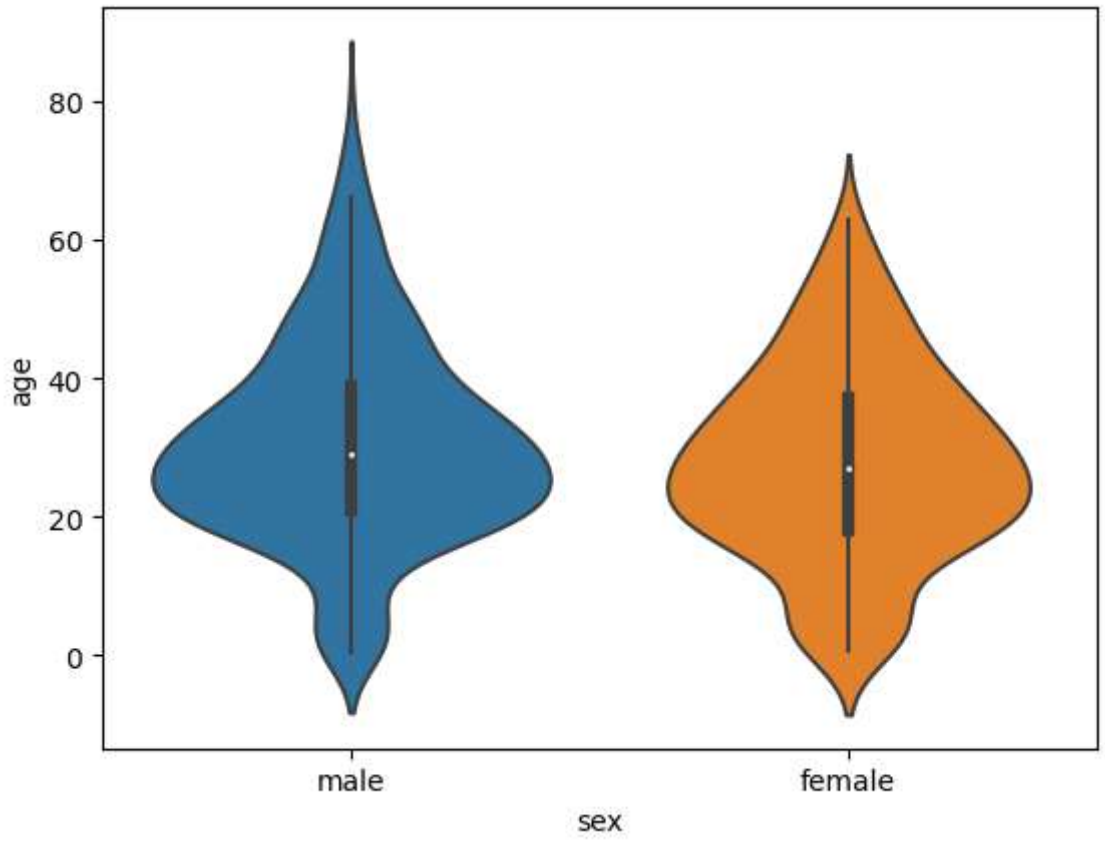
In [69]:
```
sns.countplot(x='sex', data=dataset)
```

Out[69]:
```
<AxesSubplot:xlabel='sex', ylabel='count'>
```



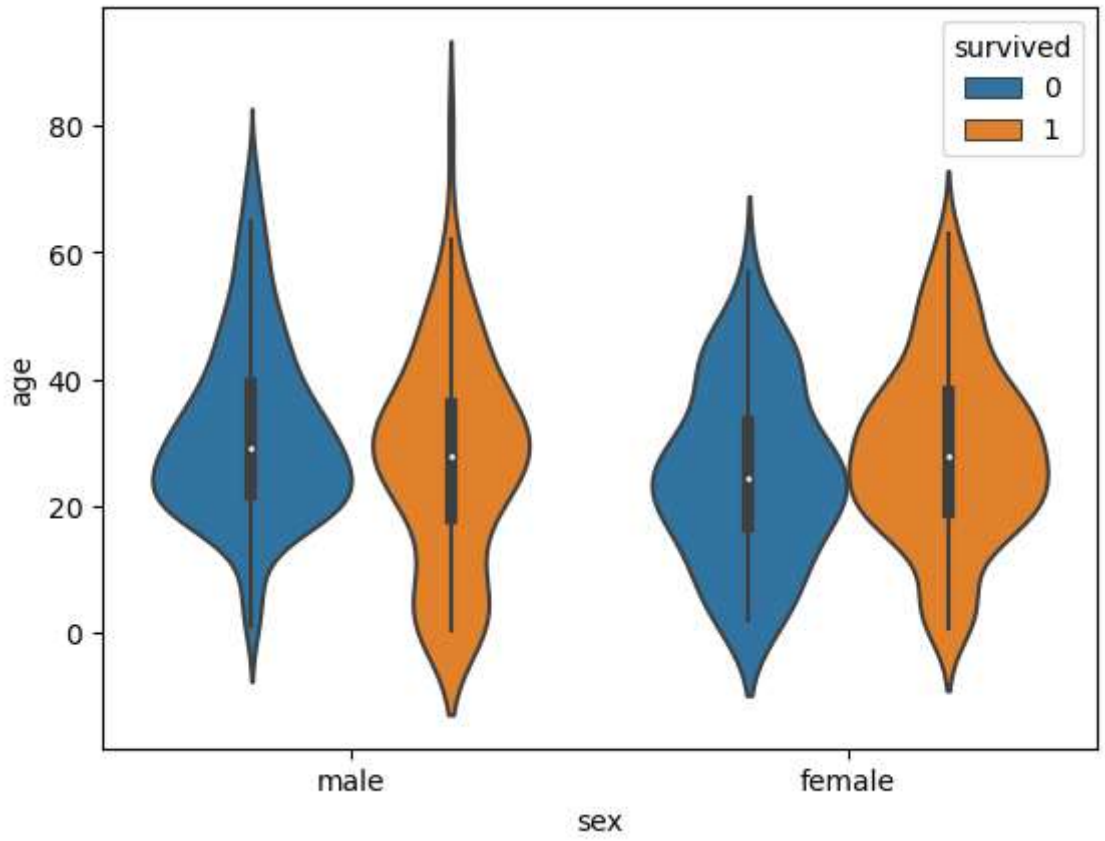
In [70]:
```
sns.violinplot(x='sex', y='age', data=dataset)
```

Out[70]: <AxesSubplot:xlabel='sex', ylabel='age'>



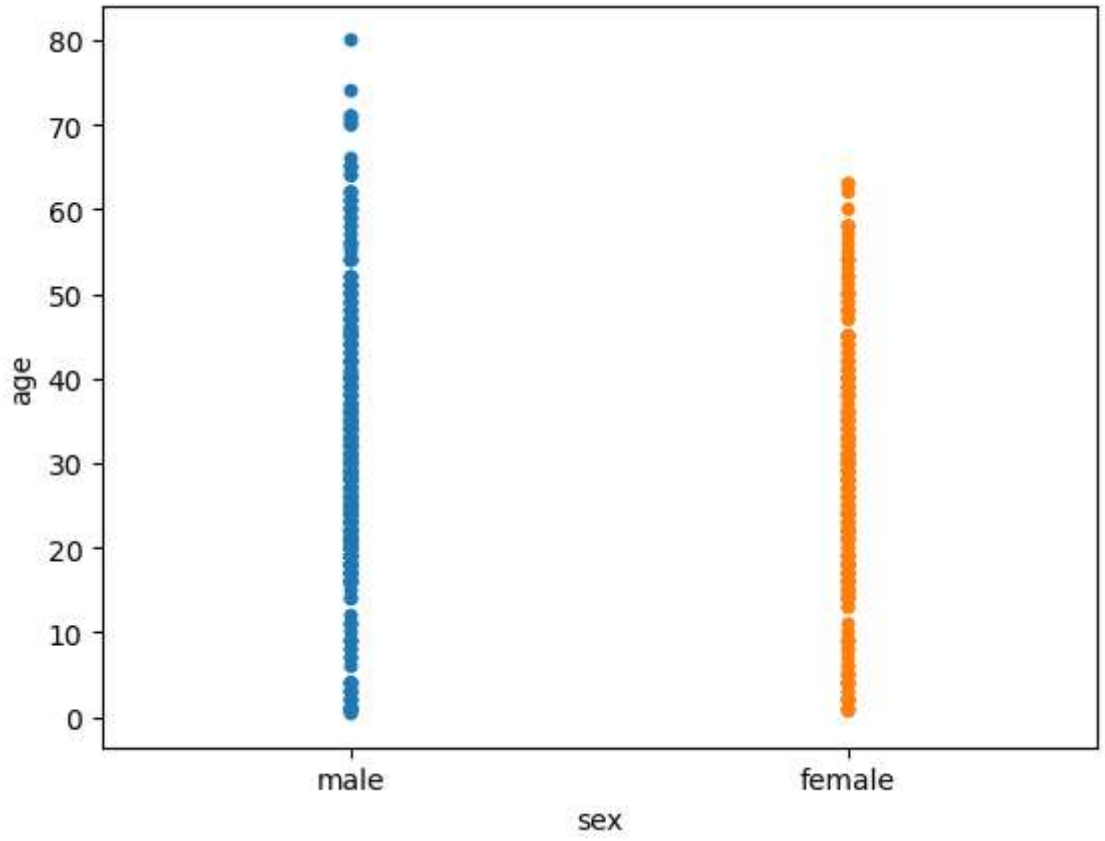
In [71]:
```
sns.violinplot(x='sex', y='age', data=dataset, hue='survived')
```

Out[71]: <AxesSubplot:xlabel='sex', ylabel='age'>



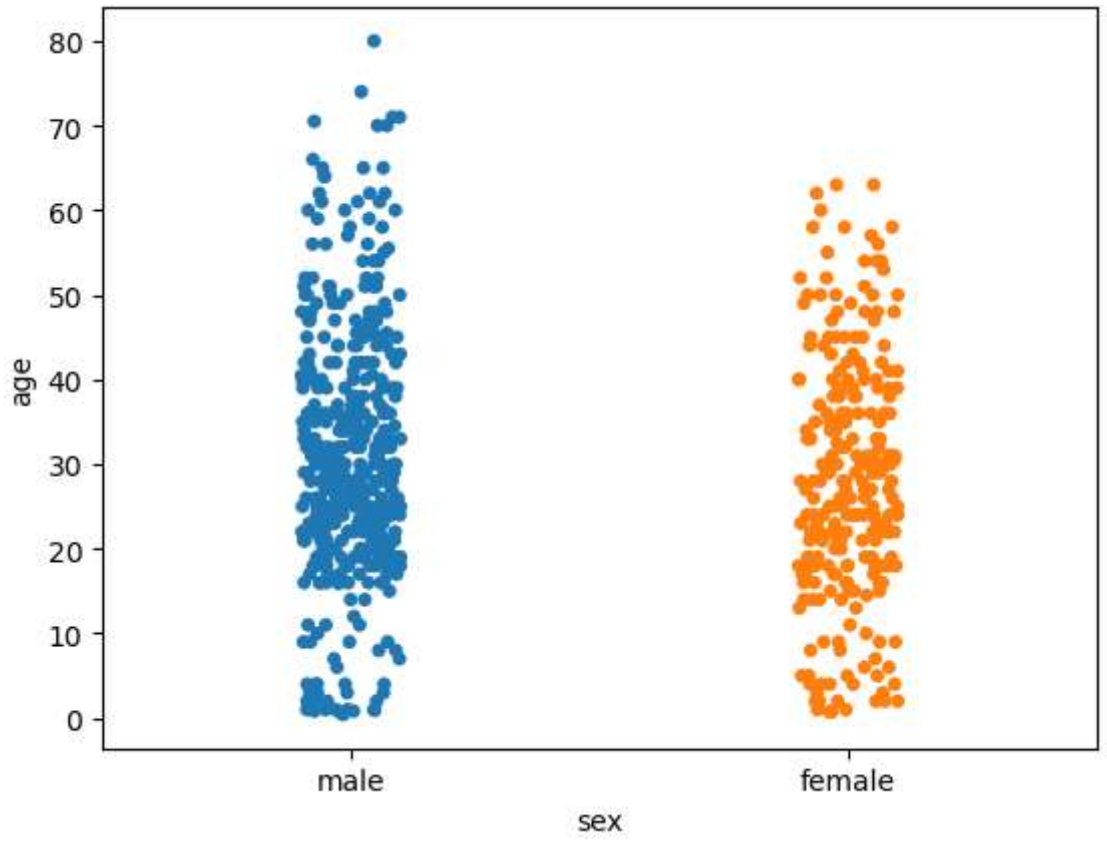
In [72]:
```
sns.stripplot(x='sex', y='age', data=dataset, jitter=False)
```

Out[72]:
```
<AxesSubplot:xlabel='sex', ylabel='age'>
```



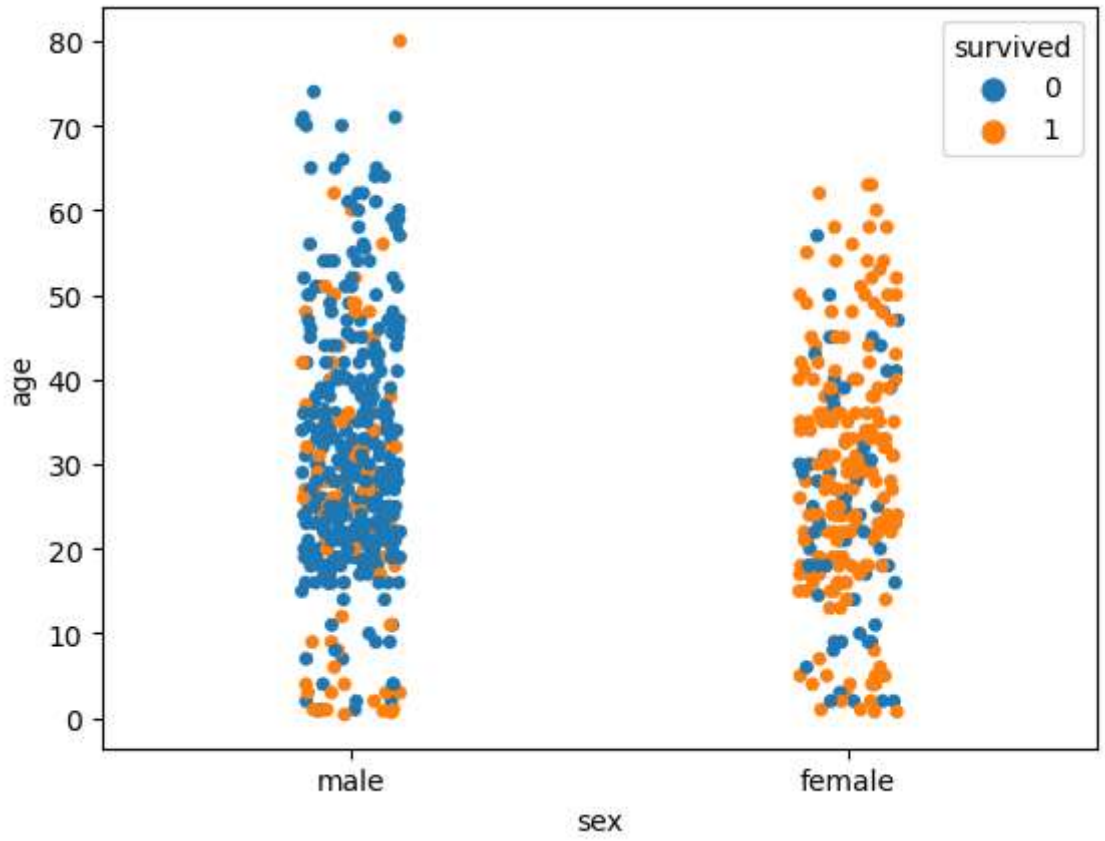
In [73]:
```
sns.stripplot(x='sex', y='age', data=dataset, jitter=True)
```

Out[73]: <AxesSubplot:xlabel='sex', ylabel='age'>



In [74]:
```
sns.stripplot(x='sex', y='age', data=dataset, jitter=True, hue='survived')
```

Out[74]: <AxesSubplot:xlabel='sex', ylabel='age'>



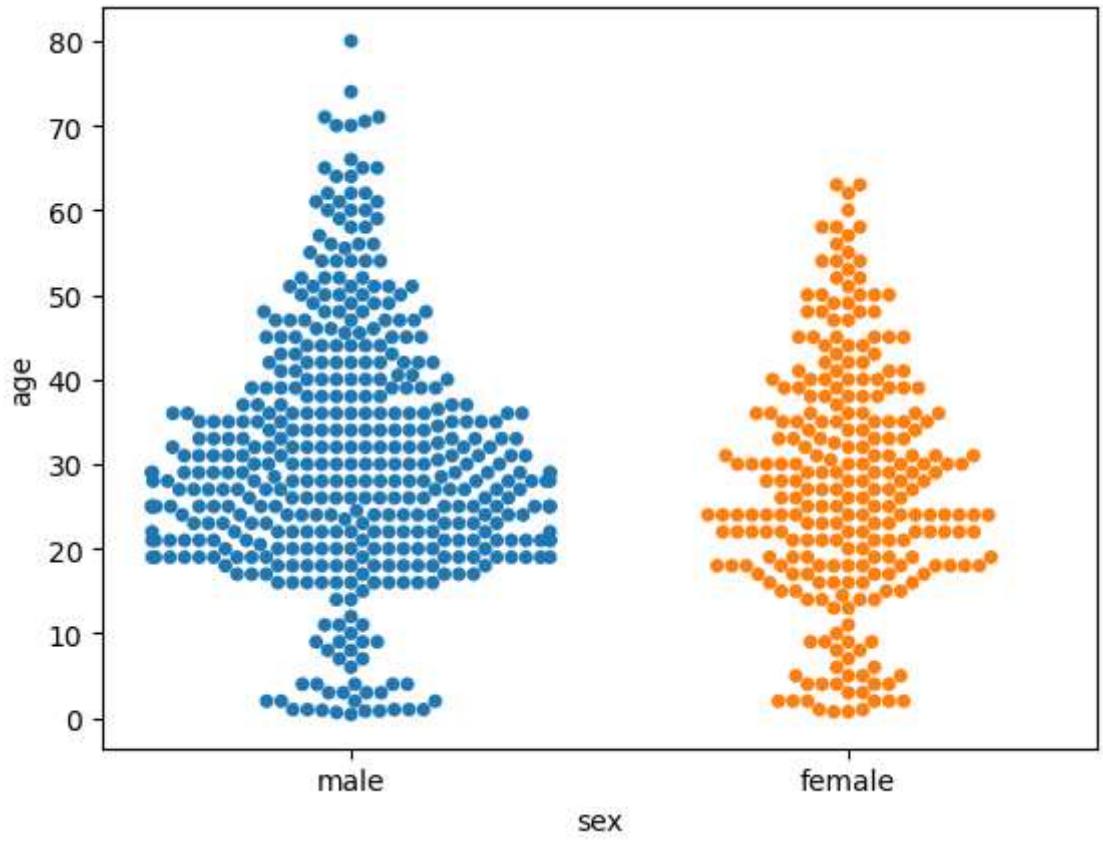
In [75]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
dataset = sns.load_dataset('titanic')
dataset.head()
```

Out[75]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN

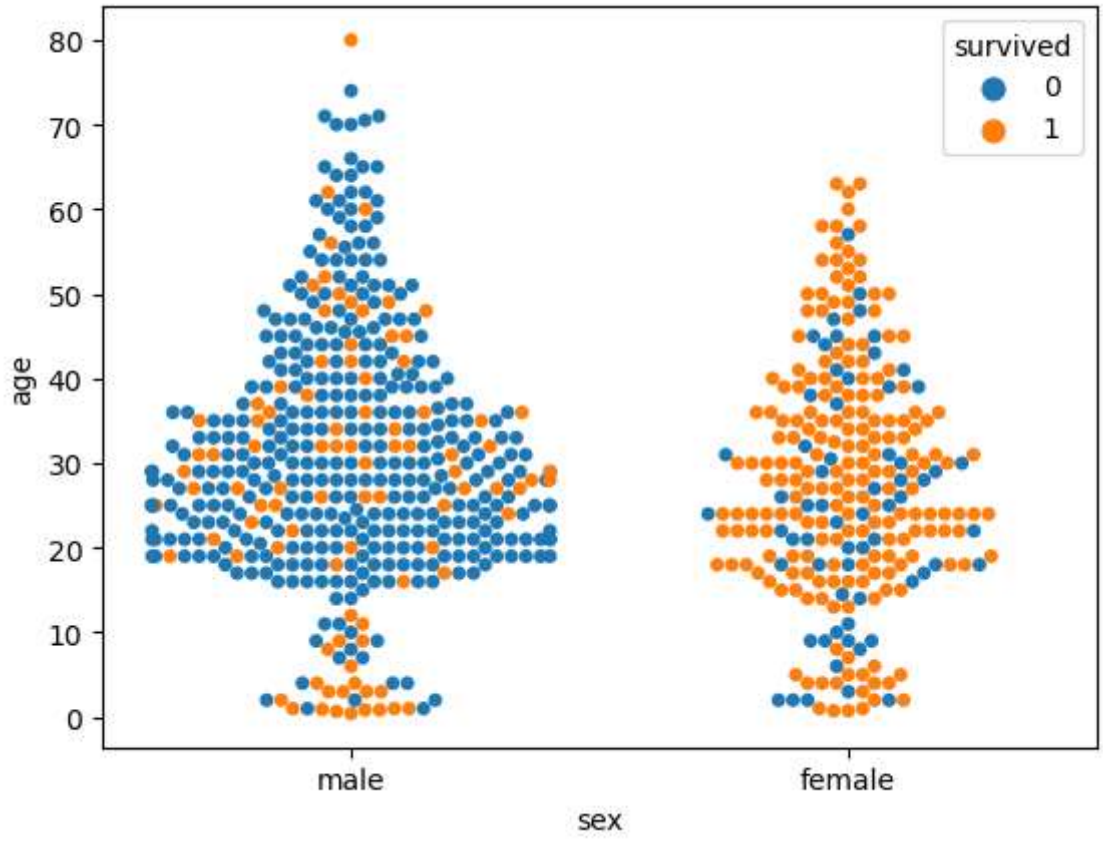
In [76]:
```
sns.swarmplot(x='sex', y='age', data=dataset)
```

Out[76]:
```
<AxesSubplot:xlabel='sex', ylabel='age'>
```



In [77]:
```
sns.swarmplot(x='sex', y='age', data=dataset, hue='survived')
```

Out[77]:
```
<AxesSubplot:xlabel='sex', ylabel='age'>
```



In [78]:
```
dataset
```

Out[78]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True
...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True

891 rows × 15 columns

In [84]:
```
dataset.drop(['pclass', 'sibsp', 'parch', 'embarked', 'class', 'deck']axis.items())
```

```
  File "C:\Users\System21\AppData\Local\Temp\ipykernel_7236\1403516129.py", line 1
    dataset.drop(['pclass', 'sibsp', 'parch', 'embarked', 'class', 'deck']axis.items
())
                                                                           ^
SyntaxError: invalid syntax
```

In [85]:
```
dataset
```

Out[85]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True
...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True

891 rows × 15 columns

In [87]:
```
dataset.sex = dataset.sex.map({'female' : 0, 'male' : 1})
```

In [88]:
```
dataset.who = dataset.who.map({'woman' : 0, 'man' : 1})
```

In [89]:
```
dataset.adult_male = dataset.adult_male.map({'False' : 0, 'True' : 1})
```

In [90]:
```
dataset.alive = dataset.alive.map({'yes' : 0, 'no' : 1})
```

In [91]:
```
dataset.drop(['embark_town', 'adult_male', 'alone'], axis=1 ,inplace=True)
```

In [92]:
```
dataset
```

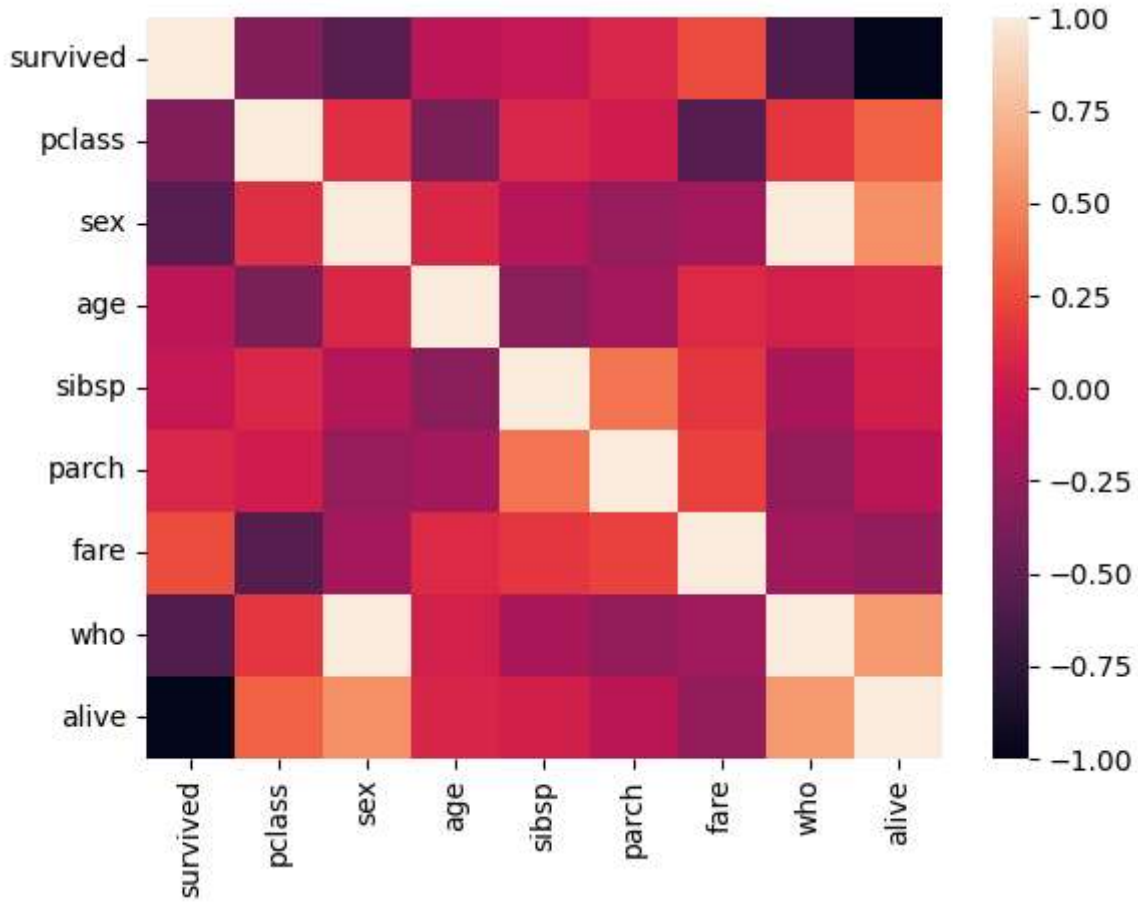
Out[92]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	deck	alive
0	0	3	1	22.0	1	0	7.2500	S	Third	1.0	NaN	1
1	1	1	0	38.0	1	0	71.2833	C	First	0.0	C	0
2	1	3	0	26.0	0	0	7.9250	S	Third	0.0	NaN	0
3	1	1	0	35.0	1	0	53.1000	S	First	0.0	C	0
4	0	3	1	35.0	0	0	8.0500	S	Third	1.0	NaN	1
...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	1	27.0	0	0	13.0000	S	Second	1.0	NaN	1
887	1	1	0	19.0	0	0	30.0000	S	First	0.0	B	0
888	0	3	0	NaN	1	2	23.4500	S	Third	0.0	NaN	1
889	1	1	1	26.0	0	0	30.0000	C	First	1.0	C	0
890	0	3	1	32.0	0	0	7.7500	Q	Third	1.0	NaN	1

891 rows × 12 columns

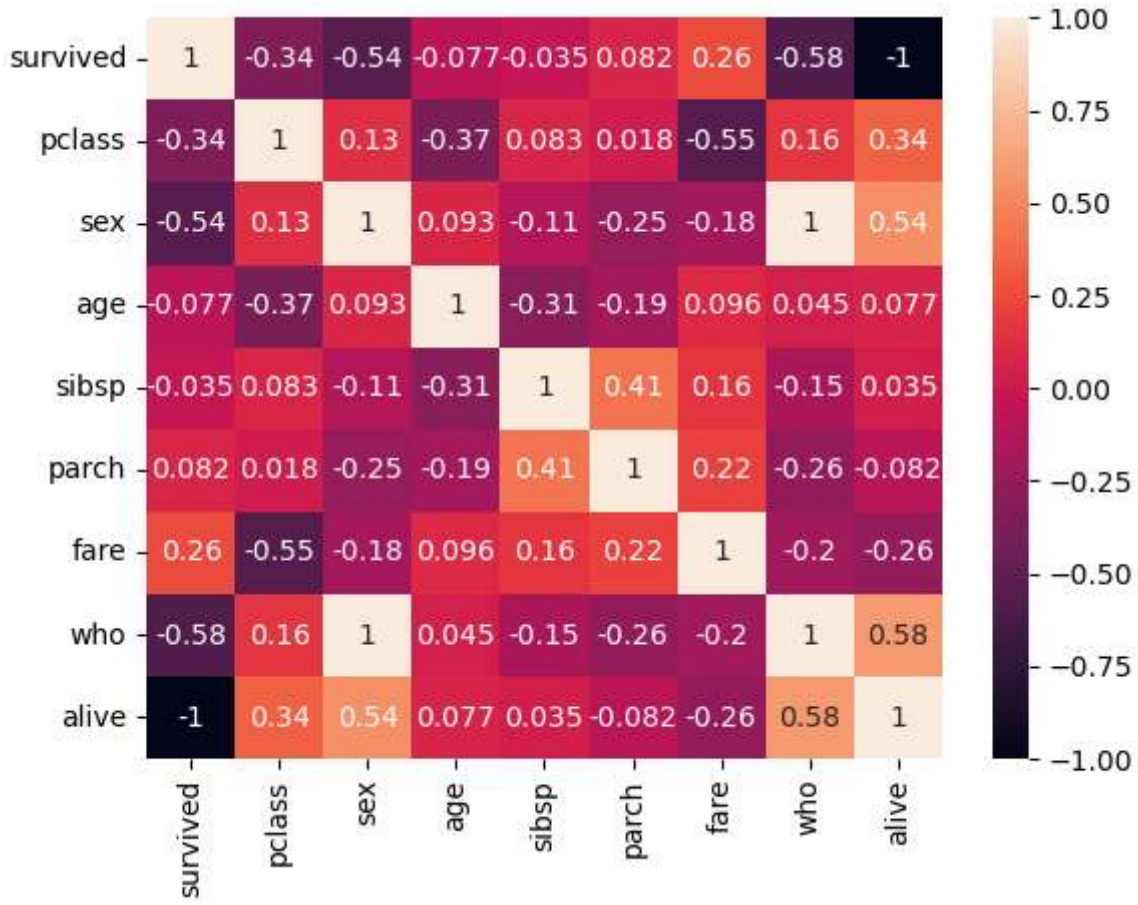
In [93]:
```
corr = dataset.corr()
sns.heatmap(corr)
```

Out[93]:
```
<AxesSubplot:>
```



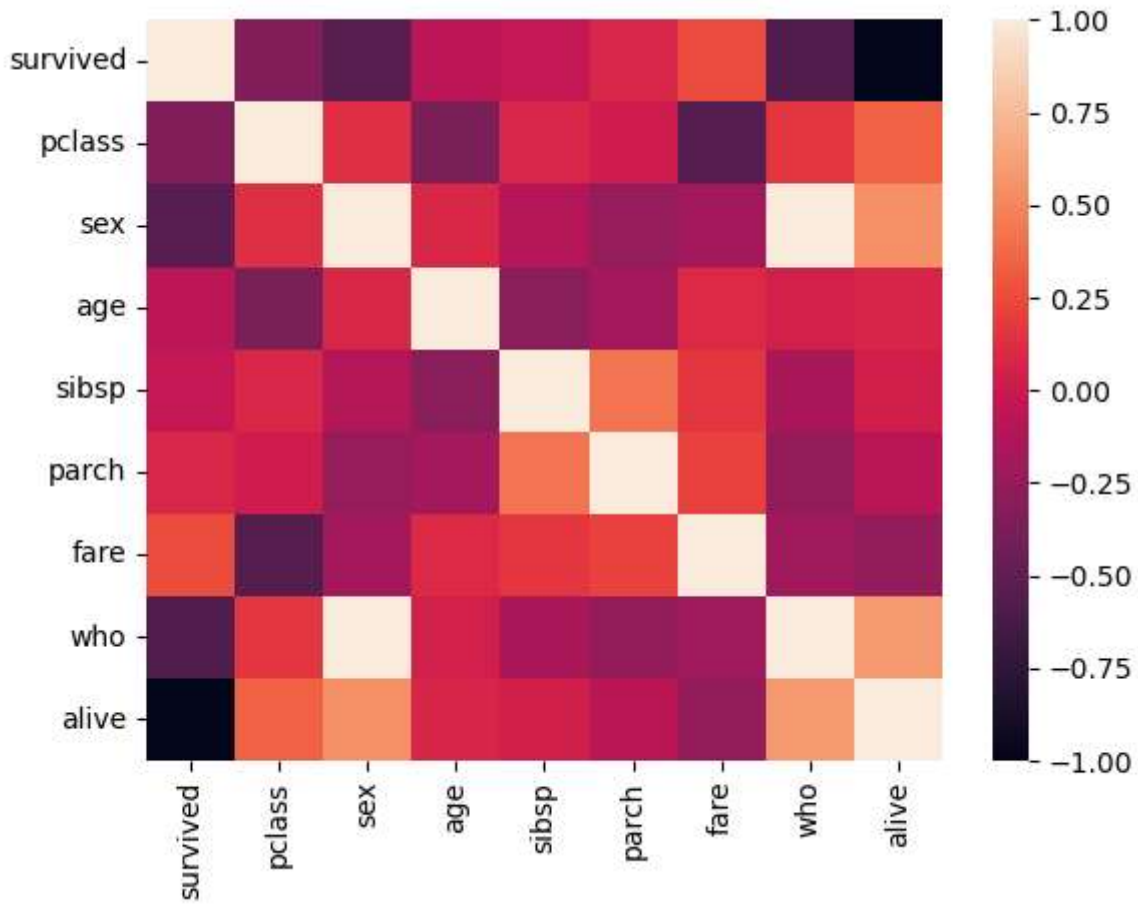
In [94]:
```
corr = dataset.corr()
sns.heatmap(corr, annot=True)
```

Out[94]:
```
<AxesSubplot:>
```



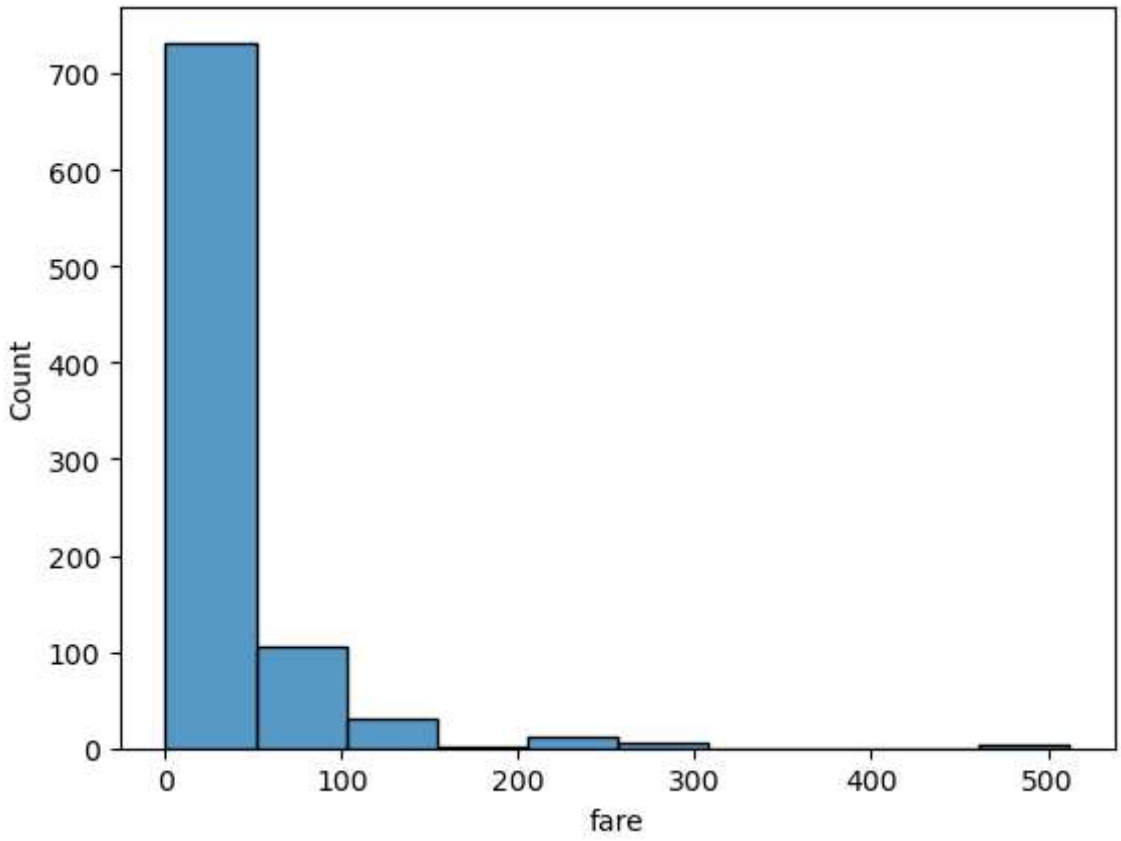
In [95]:
```
corr = dataset.corr()
sns.heatmap(corr)
```

Out[95]:
```
<AxesSubplot:>
```



In [96]:
```
import seaborn as sns
dataset = sns.load_dataset('titanic')
sns.histplot(dataset['fare'], kde=False, bins=10)
```

Out[96]:
```
<AxesSubplot:xlabel='fare', ylabel='Count'>
```



In [1]:
```
#Pratical_9
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

In [3]:
```
df=sns.get_dataset_names()
print(df)
```

```
['anagrams', 'anscombe', 'attention', 'brain_networks', 'car_crashes', 'diamonds', 'd
ots', 'dowjones', 'exercise', 'flights', 'fmri', 'geyser', 'glue', 'healthexp', 'iri
s', 'mpg', 'penguins', 'planets', 'seaice', 'taxis', 'tips', 'titanic', 'anagrams',
'anagrams', 'anscombe', 'anscombe', 'attention', 'attention', 'brain_networks', 'brai
n_networks', 'car_crashes', 'car_crashes', 'diamonds', 'diamonds', 'dots', 'dots', 'd
owjones', 'dowjones', 'exercise', 'exercise', 'flights', 'flights', 'fmri', 'fmri',
'geyser', 'geyser', 'glue', 'glue', 'healthexp', 'healthexp', 'iris', 'iris', 'mpg',
'mpg', 'penguins', 'penguins', 'planets', 'planets', 'seaice', 'seaice', 'taxis', 'ta
xis', 'tips', 'tips', 'titanic', 'titanic', 'anagrams', 'anscombe', 'attention', 'bra
in_networks', 'car_crashes', 'diamonds', 'dots', 'dowjones', 'exercise', 'flights',
'fmri', 'geyser', 'glue', 'healthexp', 'iris', 'mpg', 'penguins', 'planets', 'seaic
e', 'taxis', 'tips', 'titanic']
```

In [4]:
```
df1=sns.load_dataset("titanic")
(df1)
```

Out[4]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True
...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True

891 rows × 15 columns

In [5]:
```
df1.shape
```

Out[5]:
```
(891, 15)
```

In [6]:
```
df1.head()
```

Out[6]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN

In [7]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype   
---  ------       --------------  -----   
 0   survived     891 non-null    int64   
 1   pclass       891 non-null    int64   
 2   sex          891 non-null    object  
 3   age          714 non-null    float64 
 4   sibsp        891 non-null    int64   
 5   parch        891 non-null    int64   
 6   fare         891 non-null    float64 
 7   embarked     889 non-null    object  
 8   class        891 non-null    category
 9   who          891 non-null    object  
 10  adult_male   891 non-null    bool    
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object  
 13  alive        891 non-null    object  
 14  alone        891 non-null    bool    
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

In [8]:
```
df1.describe()
```

Out[8]:

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [9]:
```
df1.isna().sum()
```

Out[9]:
```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

In [10]:
```
df1['age'] = df1['age'].fillna(df1['age'].mean())
```

In [25]:
```
def fun1(value):
     if(value == "male"):
             return 1
     else:
         return 0
```

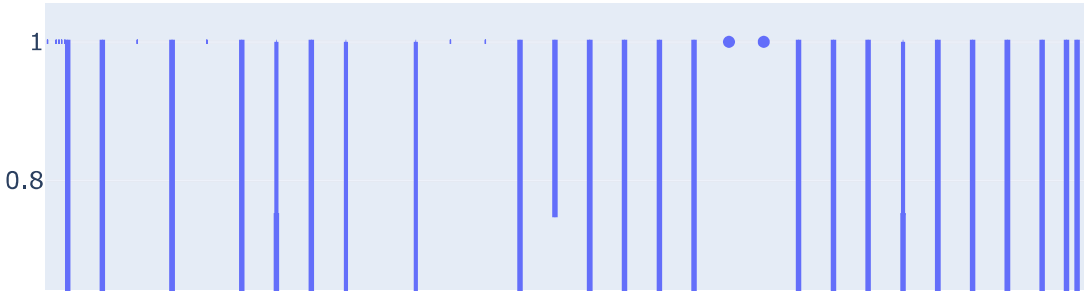
In [48]:
```
def fun2(value):
 if(value == 's'):
     return 0
 elif (value == 'c'):
     return 1
 elif(value == 'q'):
     return 2

 else:
     return 0
```

In [49]:
```
df1['sex'] = df1['sex'].apply(fun1)
```

In [50]:
```
df1['embarked']=df1['embarked'].apply(fun2)
```

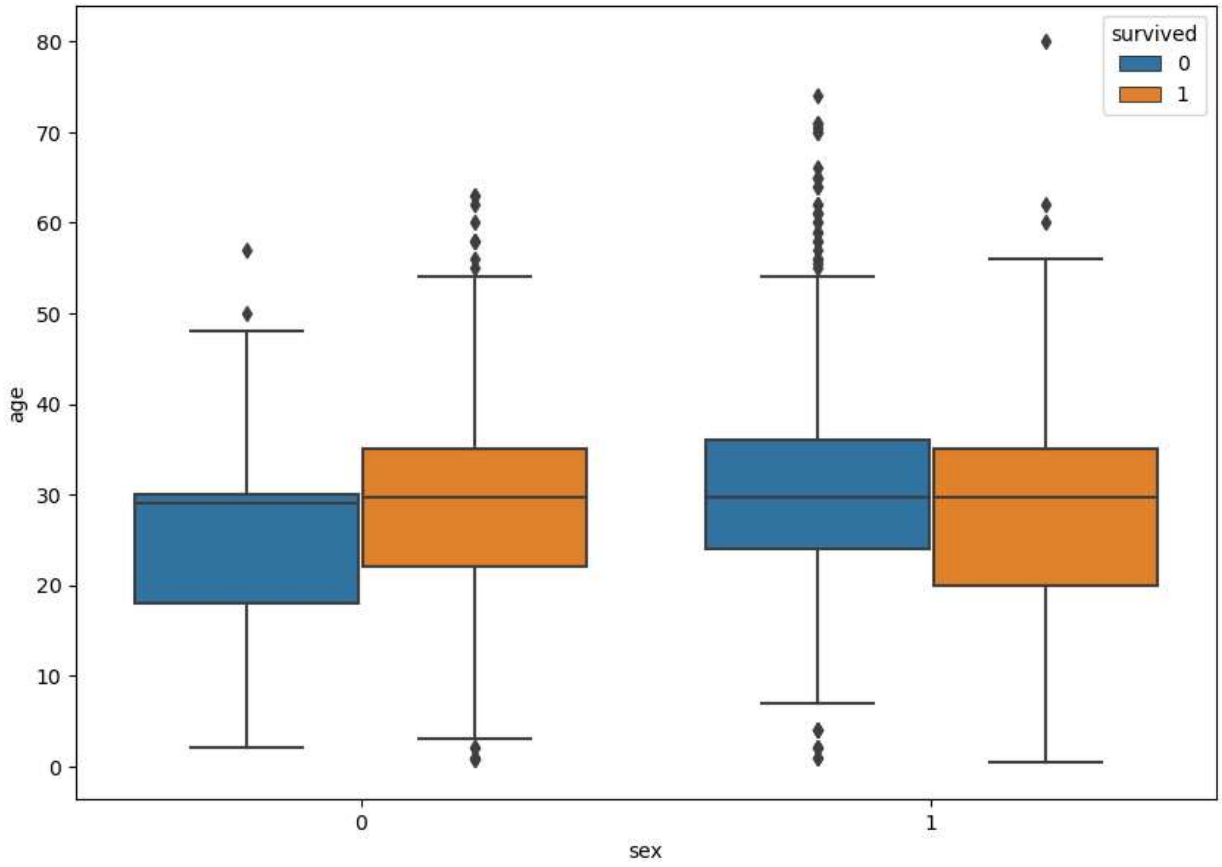
In [51]:
```
df1 = df1.drop('deck',axis =1)
```

In [52]:
```
px.box(df1['sex'],df1['age'],df1['survived'])
```



In [53]:
```
plt.figure(figsize=(10,7))
sns.boxplot(x ='sex',y='age',data=df1,hue="survived")
plt.show
```

Out[53]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [56]:
```
function Matplotlib.pyplot.show(close=None, block=None)>
```

```
  File "C:\Users\System21\AppData\Local\Temp\ipykernel_4428\1894050862.py", line 1
    function Matplotlib.pyplot.show(close=None, block=None)>
             ^
SyntaxError: invalid syntax
```

In [57]:
```
#Pratical_10
```

In [59]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.datasets import load_iris
import warnings
warnings.filterwarnings("ignore")
```

In [60]:
```
data = load_iris()
```

In [61]:
```
data
```

Out[61]:
```
{'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, 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, 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, 2, 2, 2, 2]),
 'frame': None,
 'target_names': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'),
 'DESCR': '.. _iris_dataset:\n\nIris plants dataset\n--------------------\n\n**Data S
et Characteristics:**\n\n    :Number of Instances: 150 (50 in each of three classes)
\n    :Number of Attributes: 4 numeric, predictive attributes and the class\n    :Att
ribute Information:\n        - sepal length in cm\n        - sepal width in cm\n
- petal length in cm\n        - petal 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    sepal 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    peta
l 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 classes.\n    :Creator: R.A. Fisher\n    :Donor: Michael Marshall (MARSHAL
L%PLU@io.arc.nasa.gov)\n    :Date: July, 1988\n\nThe famous Iris database, first used
by Sir R.A. Fisher. The dataset is taken\nfrom Fisher\'s paper. Note that it\'s the s
ame as in R, but not as in the UCI\nMachine Learning Repository, which has two wrong
data points.\n\nThis is perhaps the best known database to be found in the\npattern r
ecognition literature.  Fisher\'s paper is a classic in the field and\nis referenced
frequently to this day.  (See Duda & Hart, for example.)  The\ndata set contains 3 cl
asses of 50 instances each, where each class refers to a\ntype of iris plant.  One cl
ass is linearly separable from the other 2; the\nlatter are NOT linearly separable fr
om each other.\n\n.. topic:: References\n\n   - Fisher, R.A. "The use of multiple mea
surements in taxonomic problems"\n     Annual Eugenics, 7, Part II, 179-188 (1936); a
```

```
lso in "Contributions to\n     Mathematical Statistics" (John Wiley, NY, 1950).\n   -
Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.\n     (Q32
7.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.\n   - Dasarathy, B.V.
(1980) "Nosing Around the Neighborhood: A New System\n     Structure and Classificati
on Rule for Recognition in Partially Exposed\n     Environments".  IEEE Transactions
on Pattern Analysis and Machine\n     Intelligence, Vol. PAMI-2, No. 1, 67-71.\n   -
Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions\n     on I
nformation Theory, May 1972, 431-433.\n   - See also: 1988 MLC Proceedings, 54-64.  C
heeseman et al"s AUTOCLASS II\n     conceptual clustering system finds 3 classes in t
he 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 [62]:
```
df = pd.DataFrame()
df[data['feature_names']] = data['data']
df['label'] = data['target']
```

In [63]:
```
df.head()
```

Out[63]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	label
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [64]:
```
df.shape
```

Out[64]:
```
(150, 5)
```

In [65]:
```
df.info()
```

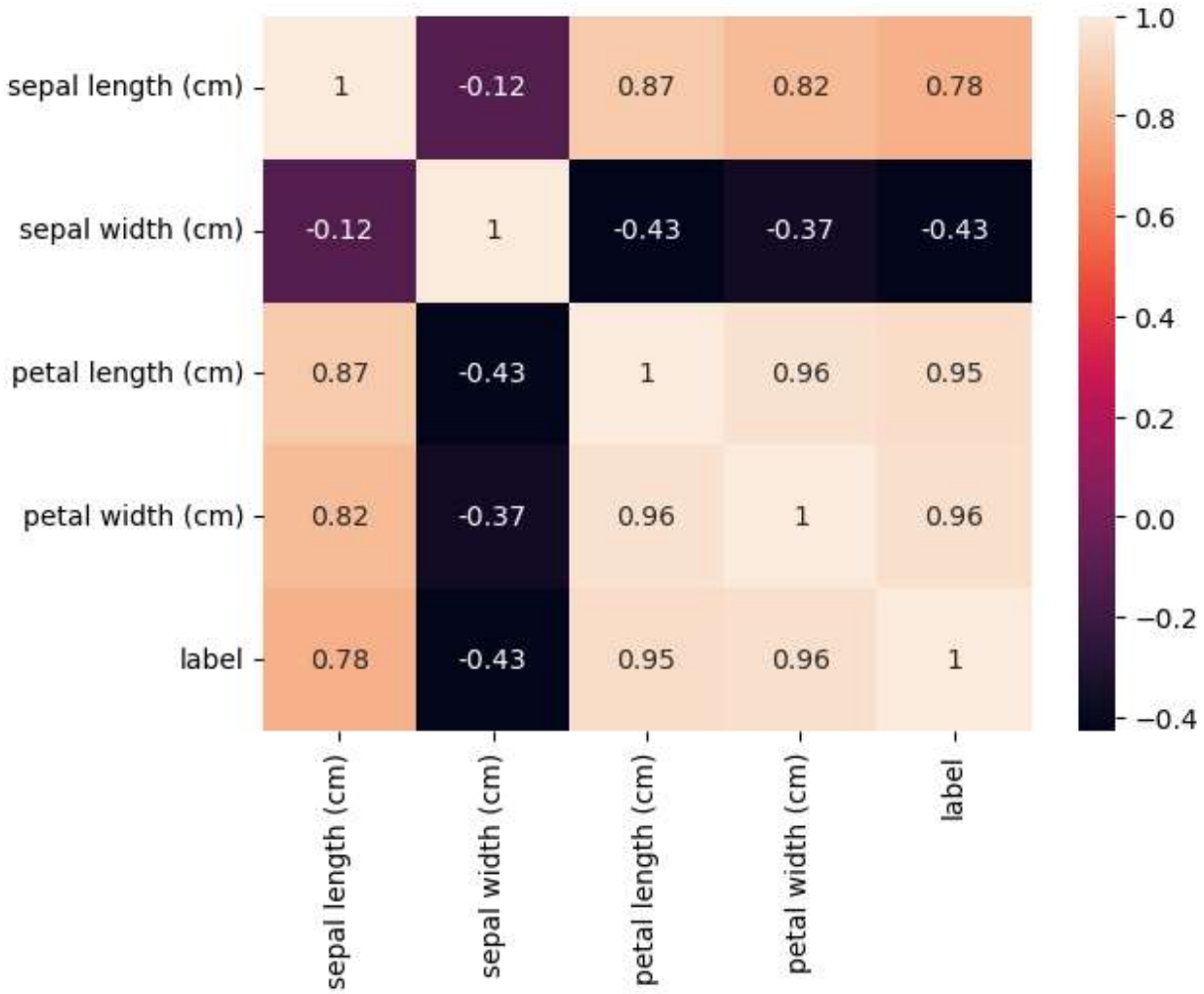
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   label              150 non-null    int32
dtypes: float64(4), int32(1)
memory usage: 5.4 KB
```

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

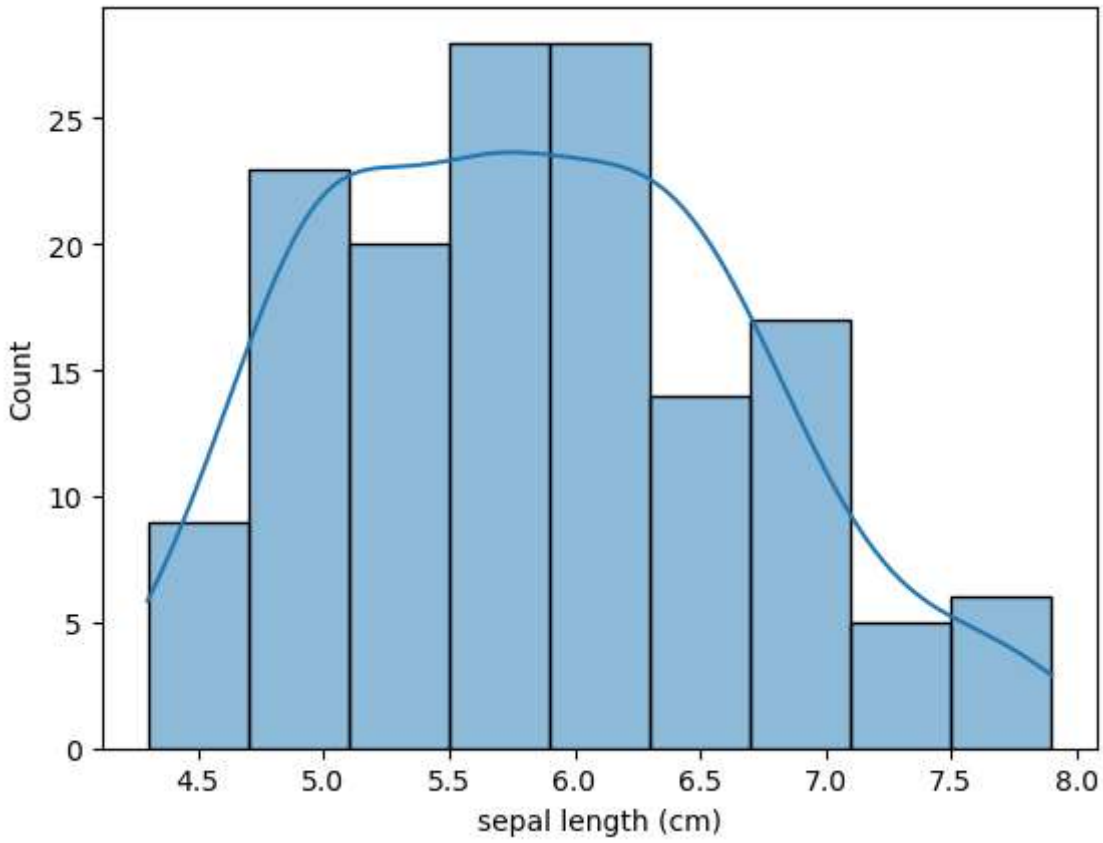
Out[66]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	label
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

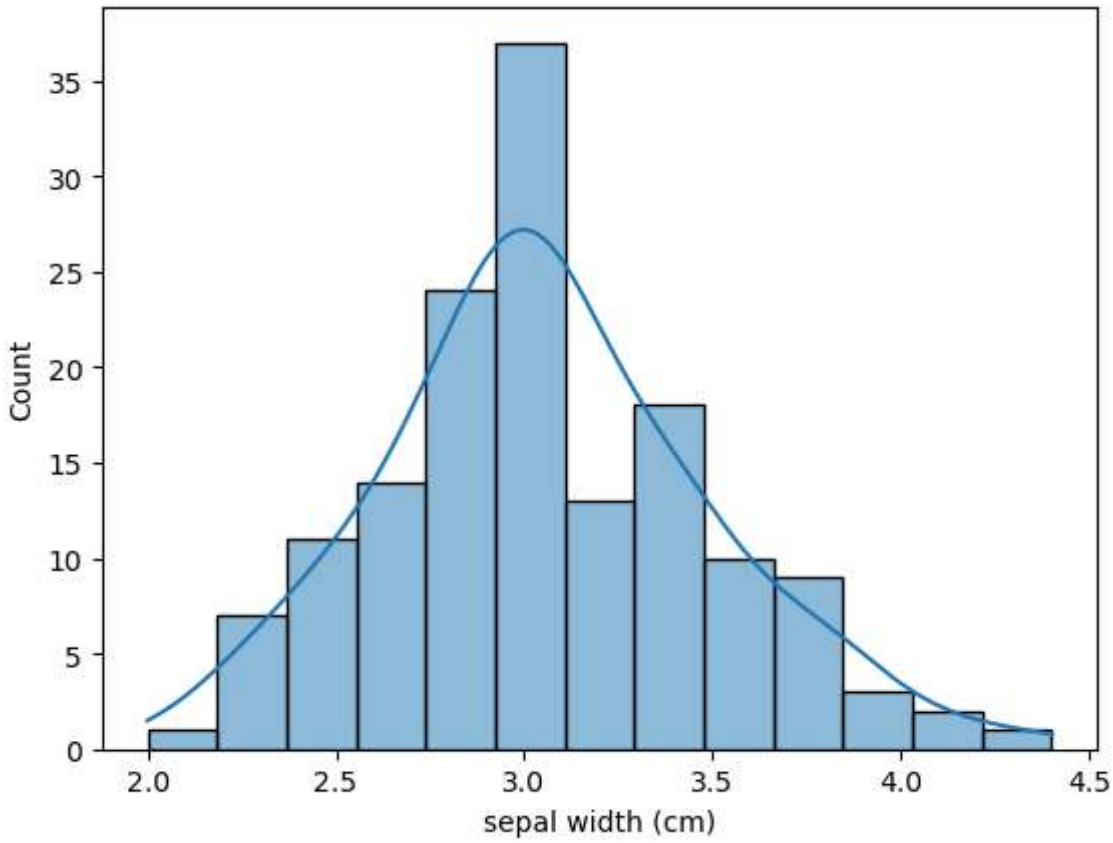
In [67]:
```
sns.heatmap(df.corr(), annot=True)
plt.show()
```



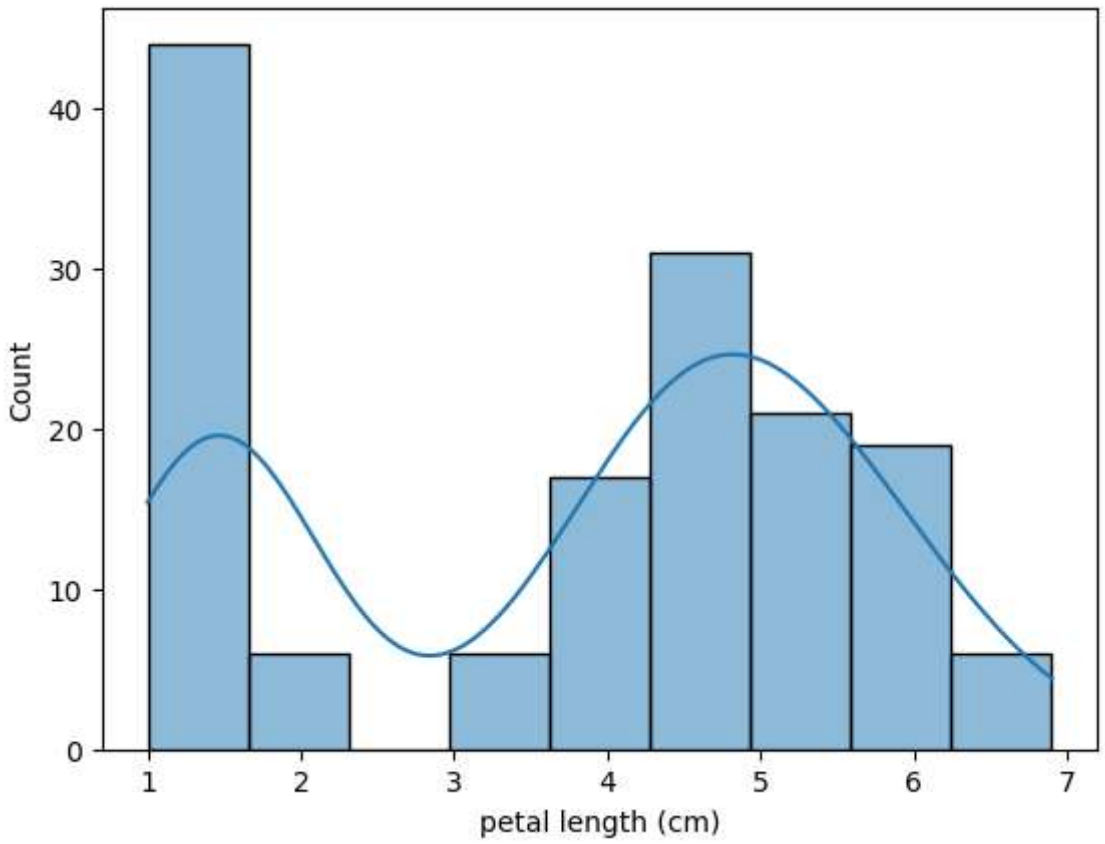
In [68]:
```
sns.histplot(df["sepal length (cm)"], kde=True)
plt.show()
```



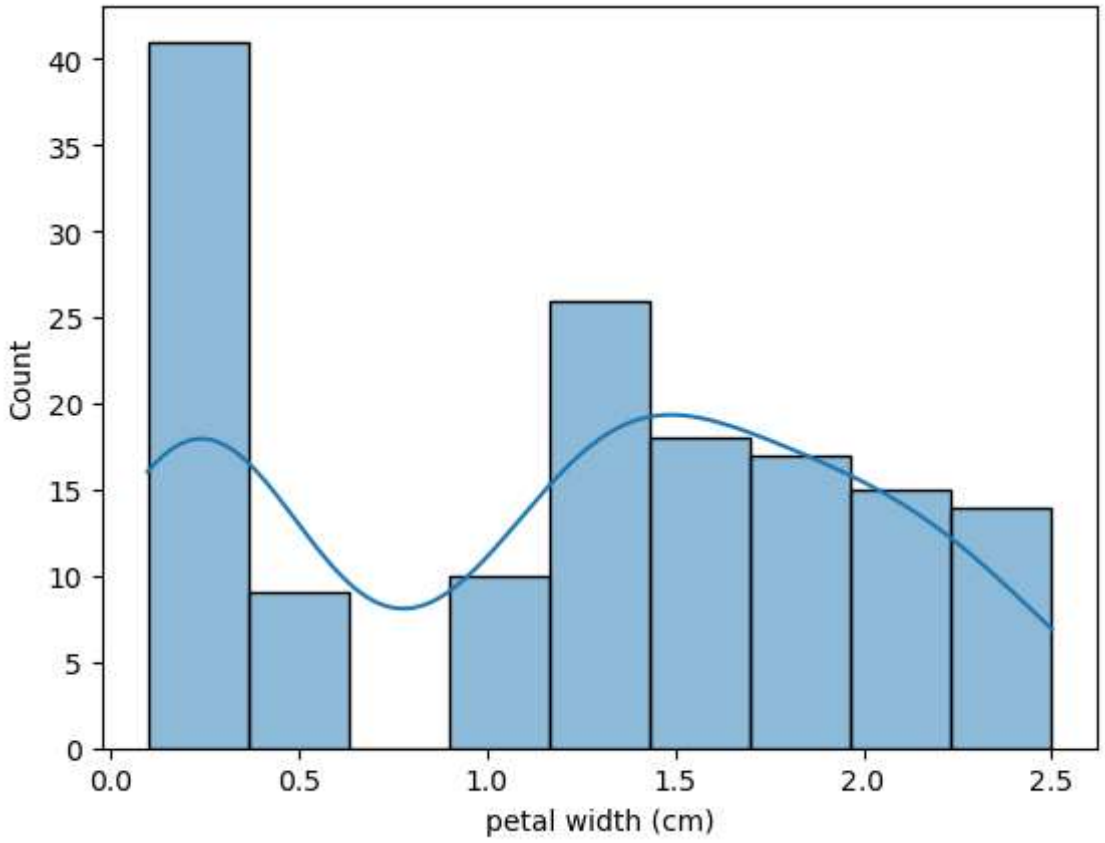
In [69]:
```
sns.histplot(df["sepal width (cm)"], kde=True)
plt.show()
```



In [70]:
```
sns.histplot(df["petal length (cm)"], kde=True)
plt.show()
```

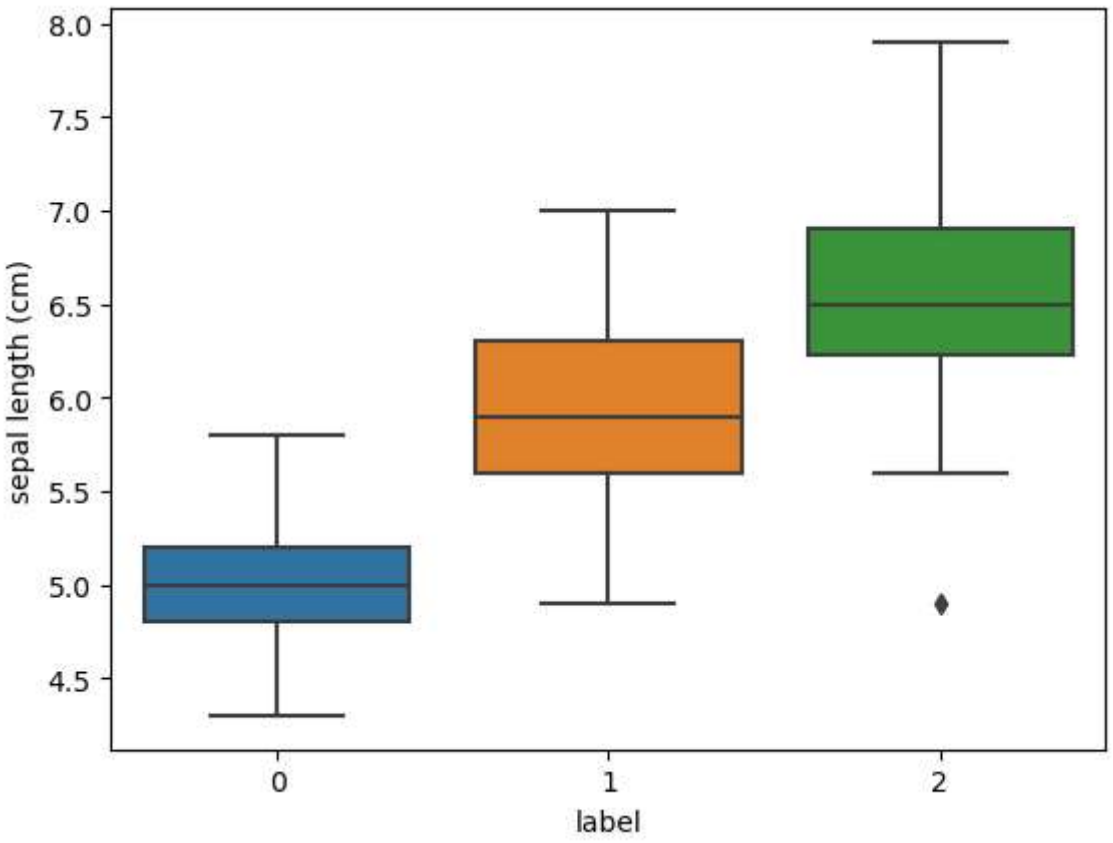


In [71]:
```
sns.histplot(df["petal width (cm)"], kde=True)
plt.show()
```

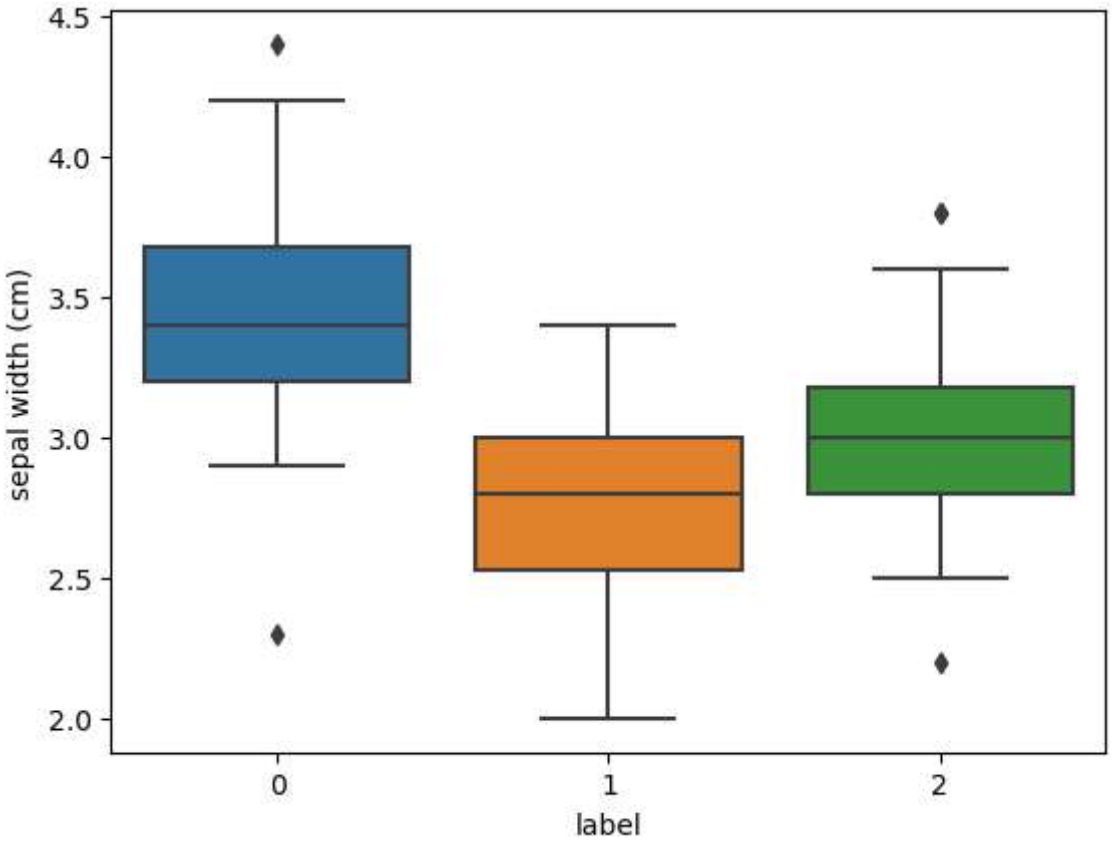


In [72]:
```
sns.boxplot(x=df['label'], y=df["sepal length (cm)"])
```

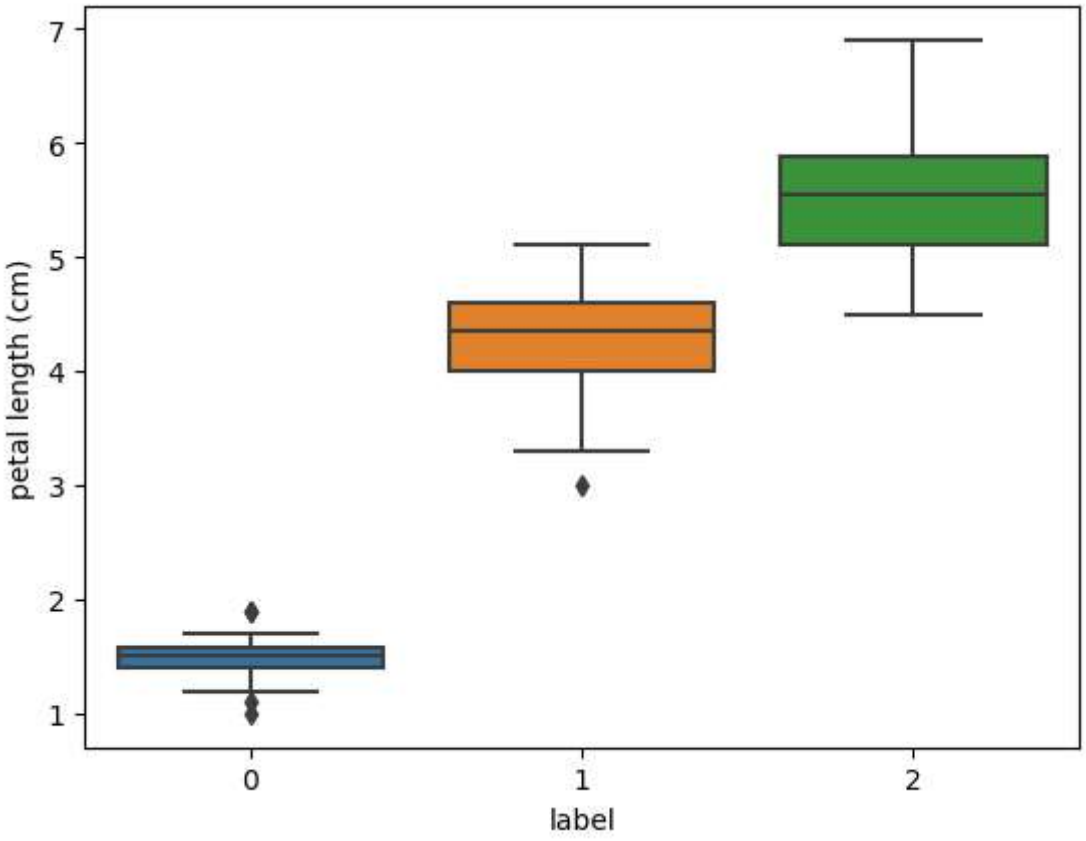
```
plt.show()
```



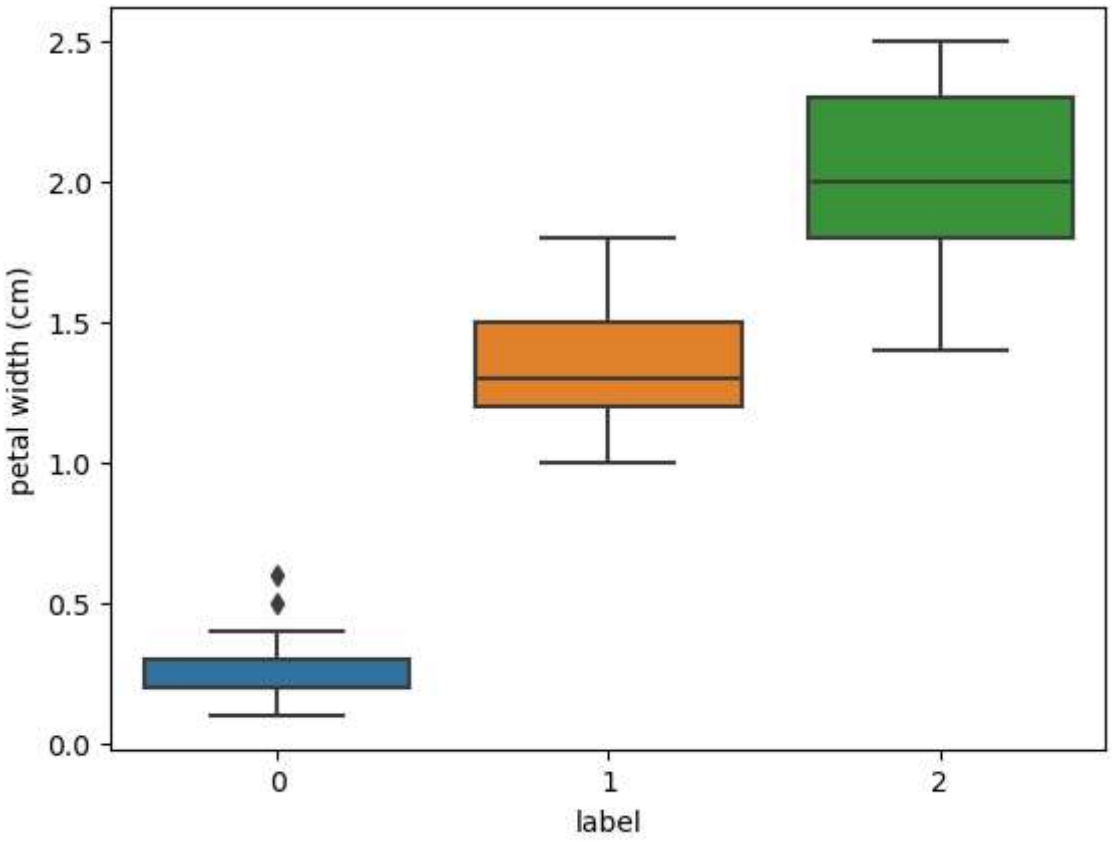
In [73]:
```
sns.boxplot(x=df['label'] ,y=df["sepal width (cm)"])
plt.show()
```



In [74]:
```
sns.boxplot(x=df["label"] ,y=df["petal length (cm)"])
plt.show()
```



In [75]:
```
sns.boxplot(x=df['label'] ,y=df["petal width (cm)"])
plt.show()
```



In [ ]: