In [4]:

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

In [5]:

```
col_names  =  ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width','Species']
```

In [6]:

```
iris  =  pd.read_csv('C:/Users/Lenovo/Downloads/iris.csv',names=col_names)
```

In [7]:

```
iris
```

Out[7]:

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

In [8]:

```
len(list(iris))
```

Out[8]:

```
5
```

In [9]:

```
iris.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  150 non-null    float64
 1   Sepal_Width   150 non-null    float64
 2   Petal_Length  150 non-null    float64
 3   Petal_Width   150 non-null    float64
 4   Species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [10]:

```
np.unique(iris["Species"])
```

Out[10]:

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [11]:

```
iris.describe()
```

Out[11]:

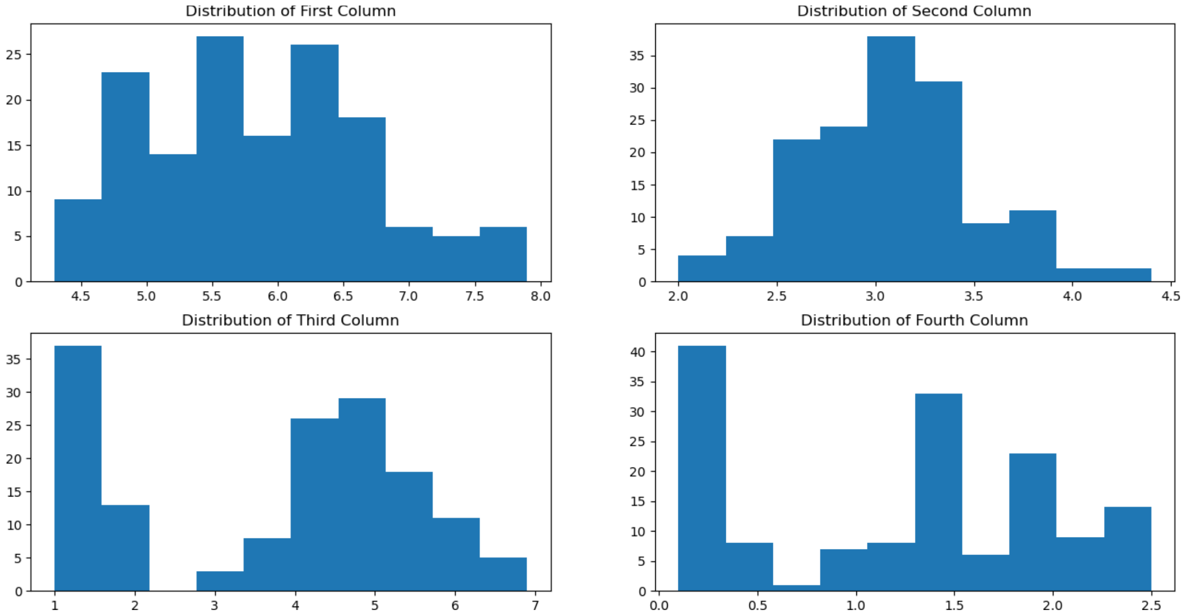
	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000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

In [12]:

```
import matplotlib.pyplot as plt
%matplotlib inline
```

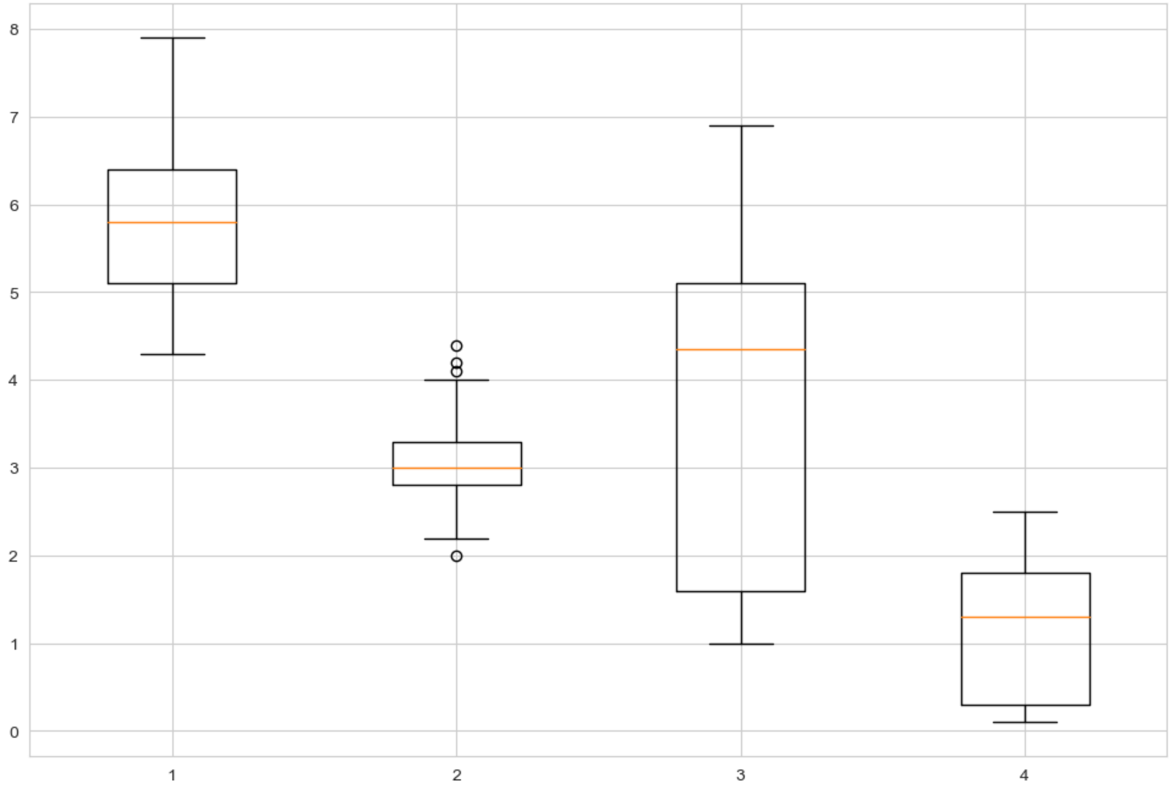
In [13]:

```
fig, axes = plt.subplots(2, 2, figsize=(16, 8))
axes[0,0].set_title("Distribution of First Column")
axes[0,0].hist(iris["Sepal_Length"]);
axes[0,1].set_title("Distribution of Second Column")
axes[0,1].hist(iris["Sepal_Width"]);
axes[1,0].set_title("Distribution of Third Column")
axes[1,0].hist(iris["Petal_Length"]);
axes[1,1].set_title("Distribution of Fourth Column")
axes[1,1].hist(iris["Petal_Width"]);
```



In [14]:

```
data_to_plot = [iris["Sepal_Length"],iris["Sepal_Width"],iris["Petal_Length"],iris
sns.set_style("whitegrid")

fig = plt.figure(1,

figsize=(12,8)) ax =

fig.add_subplot(111)
```



```
In [15]: from scipy import stats
```

```
In [16]: z = np.abs(stats.zscore(iris['Sepal_Length']))
```

```
In [17]: print(z)
```

```
0      0.900681
1      1.143017
2      1.385353
3      1.506521
4      1.021849
         ...
145    1.038005
146    0.553333
147    0.795669
148    0.432165
149    0.068662
Name: Sepal_Length, Length: 150, dtype:
      float64
```

```
In [18]: threshold = 0.5
```

```
In [19]: sample_outliers = np.where(z <threshold)
         sample_outliers
```

```
Out[19]: (array([ 14,  15,  18,  33,  36,  53,  55,  61,  62,  63,  64,  66,  67,
                  68,  69,  70,  71,  73,  78,  79,  80,  81,  82,  83,  85,  88,
                  89,  90,  91,  92,  94,  95,  96,  97,  99, 101, 113, 114, 119,
                 121, 126, 127, 134, 138, 142, 148, 149],
               dtype=int64),)
```

```
In [20]: sorted_rscore=  sorted(iris['Sepal_Length'])
```

```
In [21]: sorted_rscore
```

```
Out[21]: [4.3,
          4.4,
          4.4,
          4.4,
          4.5,
          4.6,
          4.6,
          4.6,
          4.6,
          4.7,
          4.7,
          4.8,
          4.8,
          4.8,
          4.8,
          4.8,
          4.9,
          4.9,
          4.9,
          4.9,
          4.9,
          4.9,
          5.5,
          5.5,
          5.6,
          5.6,
          5.6,
          5.6,
          5.6,
```

```
In [22]: q1 = np.percentile(sorted_rscore, 25)
         q3 = np.percentile(sorted_rscore, 75)
         print(q1,q3)

         5.1 6.4
```

```
In [23]: IQR = q3-q1
```

```
In [24]: IQR

Out[24]: 1.3000000000000007
```

```
In [25]: lwr_bound = q1-
         (1.5*IQR) upr_bound =
         q3+(1.5*IQR)

         3.1499999999999986  8.350000000000001
```

```
In [26]: r_outliers = []
         for i in sorted_rscore:
                 if (i<lwr_bound or
                         i>upr_bound):
                         r_outliers.append(i)
```

```
In [27]: print(r_outliers)

         []
```

```
In [28]: z = np.abs(stats.zscore(iris['Sepal_Width']))
```

```
In [29]: print(z)
```

```
0      1.032057
1      0.124958
2      0.337848
3      0.106445
4      1.263460
         ...
145    0.124958
146    1.281972
147    0.124958
148    0.800654
149    0.124958
Name: Sepal_Width, Length: 150, dtype:
      float64
```

In [30]:
```
threshold = 0.5
```

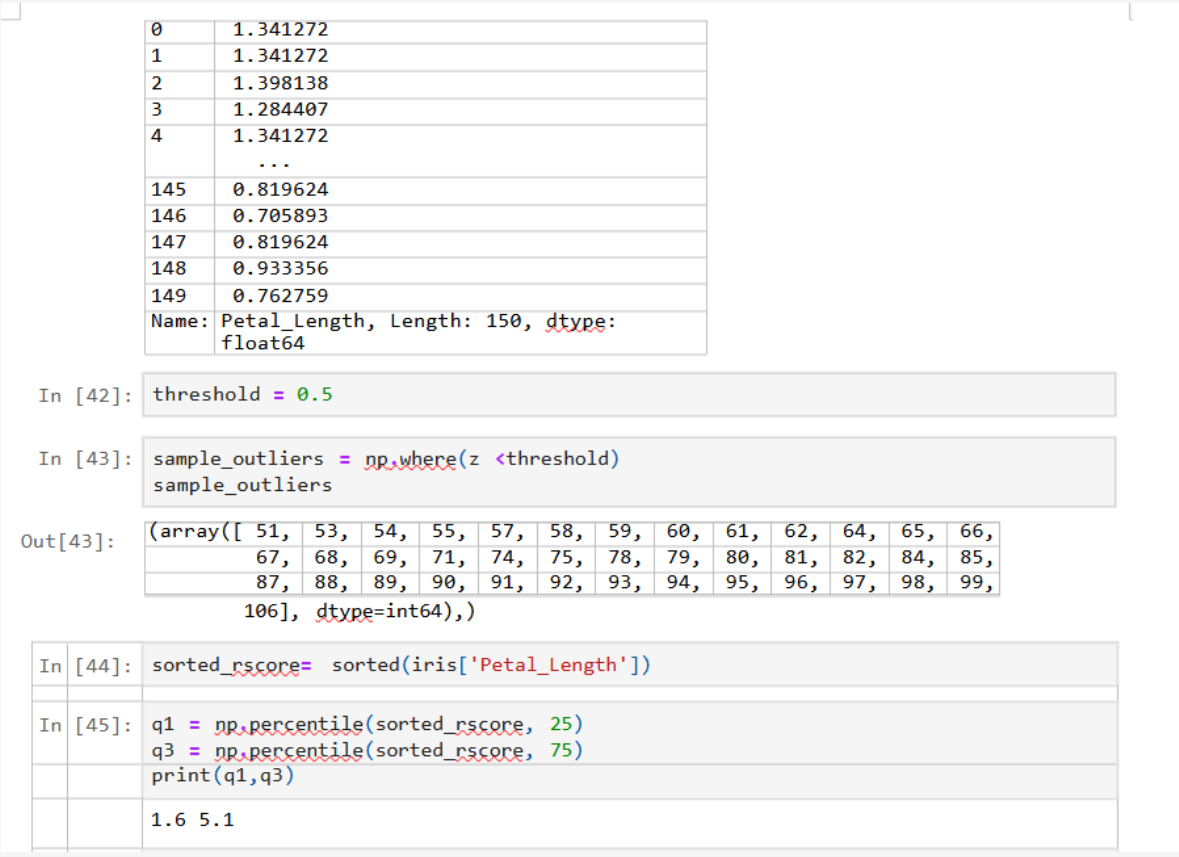
In [31]:
```
sample_outliers = np.where(z <threshold)
sample_outliers
```

Out[31]:
```
(array([  1,   2,   3,   8,   9,  12,  13,  25,  29,  30,  34,  35,  37,
         38,  42,  45,  47,  50,  51,  52,  58,  61,  63,  64,  65,  66,
         70,  74,  75,  77,  78,  84,  86,  88,  91,  95,  96,  97, 102,
        103, 104, 105, 107, 110, 112, 115, 116, 120, 125, 127, 129, 135,
               137, 138, 139, 140, 141, 143, 145, 147, 149],
                               dtype=int64),)
```

In [32]:
```
sorted_rscore=  sorted(iris['Sepal_Width'])
```

In [33]:
```
sorted_rscore
```

Out[33]:
```
[2.0,
 2.2,
 2.2,
 2.2,
 2.3,
 2.3,
 2.3,
 2.3,
 2.4,
 2.4,
 2.4,
 2.5,
 2.5,
 2.5,
 2.5,
 2.5,
 2.5,
 2.5,
 2.5,
 2.6,
 2.6,
 2.6,
 2.6,
 2.6,
 2.7,
 2.7,
 2.7,
 2.7,
 2.7,
 2.7,
 2.7,
 2.7,
 2.7,
 2.8,
 2.8,
In [34]: q1  = np.percentile(sorted_rscore,  25)
         q3  = np.percentile(sorted_rscore,  75)
         print(q1,q3)

         2.8 3.3

In [35]: IQR = q3-q1

In [36]: IQR

Out[36]: 0.5

In [37]: lwr_bound = q1-
         (1.5*IQR) upr_bound =
         q3+(1.5*IQR)

         2.05 4.05

In [38]: r_outliers = []
         for i in sorted_rscore:
                 if (i<lwr_bound or
                         i>upr_bound):
                         r_outliers.append(i)
         print(r_outliers)

         [2.0, 4.1, 4.2, 4.4]

In [39]: from scipy import stats

In [40]: z  = np.abs(stats.zscore(iris['Petal_Length']))

In [41]: print(z)
```



```
0       1.341272
1       1.341272
2       1.398138
3       1.284407
4       1.341272
          ...
145     0.819624
146     0.705893
147     0.819624
148     0.933356
149     0.762759
Name: Petal_Length, Length: 150, dtype:
float64

In [42]: threshold = 0.5

In [43]: sample_outliers = np.where(z <threshold)
         sample_outliers

Out[43]: (array([ 51,  53,  54,  55,  57,  58,  59,  60,  61,  62,  64,  65,  66,
                  67,  68,  69,  71,  74,  75,  78,  79,  80,  81,  82,  84,  85,
                  87,  88,  89,  90,  91,  92,  93,  94,  95,  96,  97,  98,  99,
                 106], dtype=int64),)

In [44]: sorted_rscore=  sorted(iris['Petal_Length'])

In [45]: q1 = np.percentile(sorted_rscore, 25)
         q3 = np.percentile(sorted_rscore, 75)
         print(q1,q3)

         1.6 5.1
```

In [46]:
```
IQR = q3-q1
```

In [47]:
```
lwr_bound = q1-(1.5*IQR)
upr_bound = q3+(1.5*IQR)
print(lwr_bound,  upr_bound)
```

```
-3.6499999999999999  10.349999999999998
```

In [60]:
```
r_outliers = []
for i in sorted_rscore:
        if (i<lwr_bound or i>upr_bound):
                r_outliers.append(i)
print(r_outliers)
```

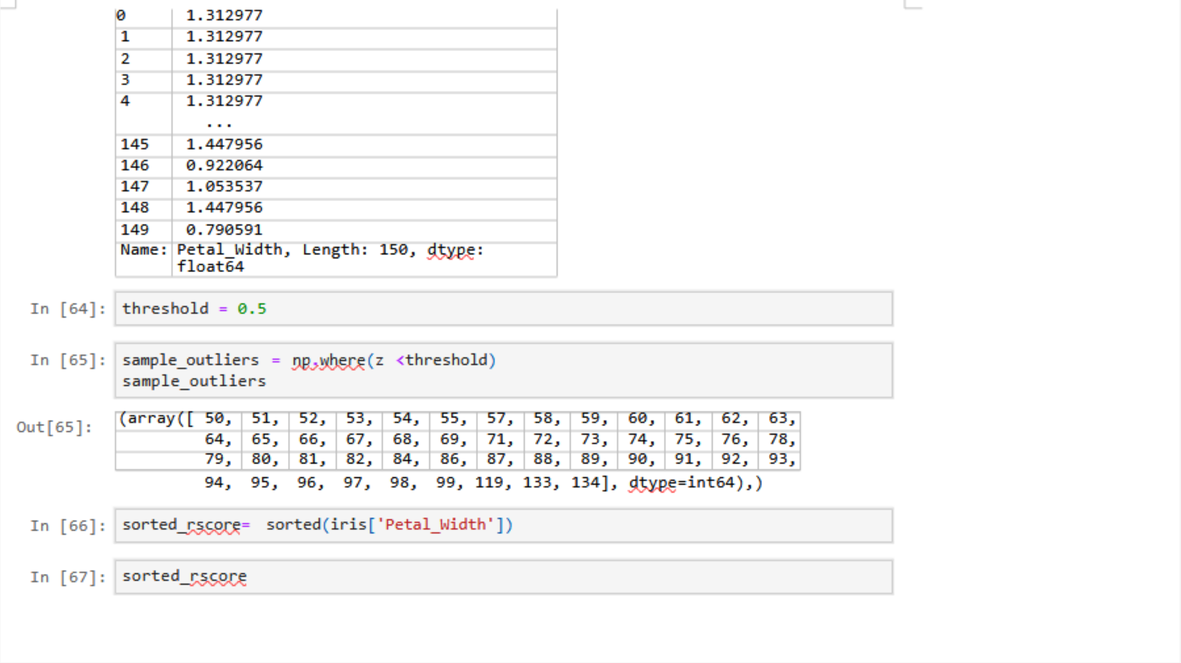
```
[]
```

In [61]:
```
print(r_outliers)
```

```
[]
```

In [62]:
```
z  =  np.abs(stats.zscore(iris['Petal_Width']))
```

In [63]:
```
print(z)
```



```
0       1.312977
1       1.312977
2       1.312977
3       1.312977
4       1.312977
          ...
145     1.447956
146     0.922064
147     1.053537
148     1.447956
149     0.790591
Name: Petal_Width, Length: 150, dtype:
        float64
```

In [64]:
```
threshold = 0.5
```

In [65]:
```
sample_outliers  =  np.where(z  <threshold)
sample_outliers
```

Out[65]:
```
(array([ 50,  51,  52,  53,  54,  55,  57,  58,  59,  60,  61,  62,  63,
         64,  65,  66,  67,  68,  69,  71,  72,  73,  74,  75,  76,  78,
         79,  80,  81,  82,  84,  86,  87,  88,  89,  90,  91,  92,  93,
         94,  95,  96,  97,  98,  99, 119, 133, 134], dtype=int64),)
```

In [66]:
```
sorted_rscore=  sorted(iris['Petal_Width'])
```

In [67]:
```
sorted_rscore
```

```
Out[67]: [0.1,
          0.1,
          0.1,
          0.1,
          0.1,
          0.1,
          0.2,
          0.2,
          1.1,
          1.2,
          1.2,
          1.2,
          1.2,
          1.8,
          1.8,
          1.8,
          1.8,
          1.8,
          1.8,
          1.9,
          1.9,
          1.9,
          2.0,
          2.0,
          2.0,
          2.0,
          2.0,
          2.0,
          2.1,
          2.3,
          2.3,
          2.3,
          2.3,
          2.3,
          2.3,
          2.3,
          2.4,
          2.4,
          2.4,
          2.5,
          2.5,
          2.5]
```

```
In [68]: q1  =  np.percentile(sorted_rscore,  25)
         q3  =  np.percentile(sorted_rscore,  75)
         print(q1,q3)
```

```
0.3 1.8
```

```
In [69]: IQR  =  q3-q1
```

```
In [70]: lwr_bound  =  q1-
         (1.5*IQR)  upr_bound  =
         q3+(1.5*IQR)
         print(lwr_bound,  upr_bound)
```

```
-1.95 4.05
```

```
In [71]: r_outliers  =  []
         for  i  in  sorted_rscore:
                 if  (i<lwr_bound  or
                         i>upr_bound):
                         r_outliers.append(i)
         print(r_outliers)
```

```
[]
```

In [3]:

```python
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[3]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	de
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Na
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	Na
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	Na

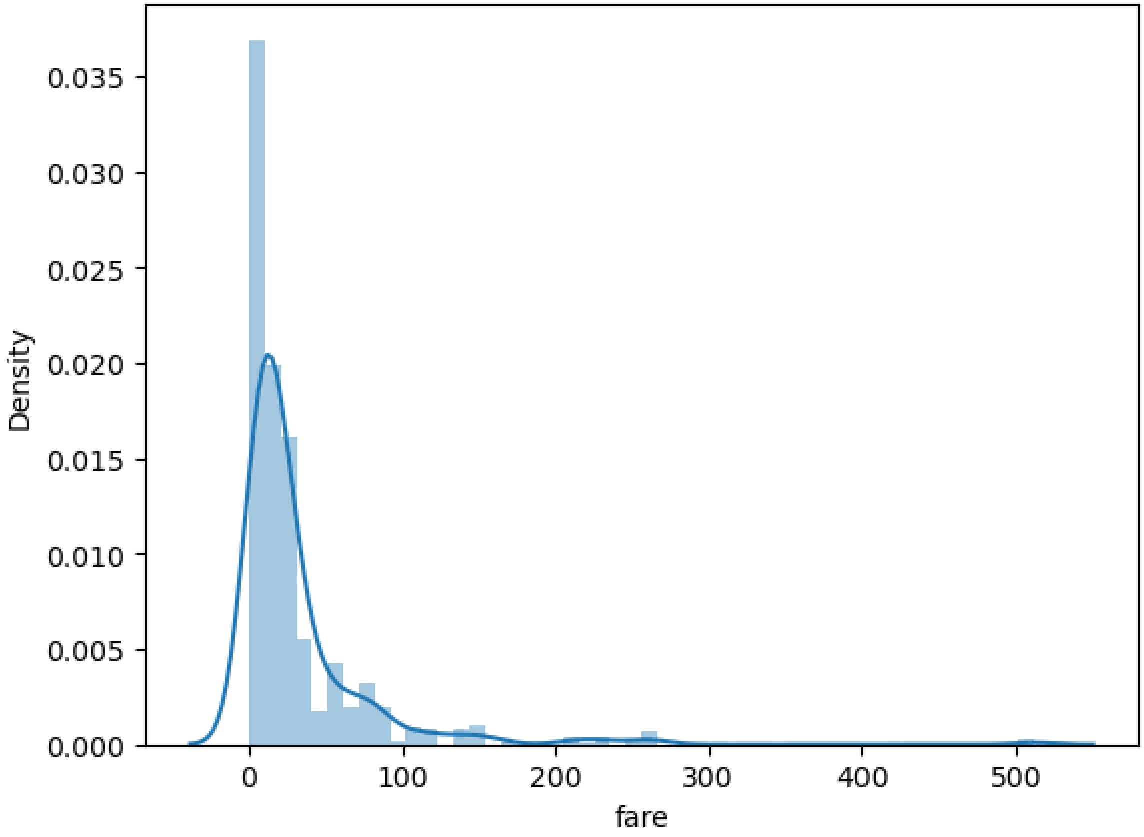
In [4]:

```python
sns.distplot(dataset['fare'])
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureW
arning: `distplot` is a deprecated function and will be removed in a future versio
n. Please adapt your code to use either `displot` (a figure-level function with si
milar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[4]:

```
<AxesSubplot:xlabel='fare', ylabel='Density'>
```



In [5]:

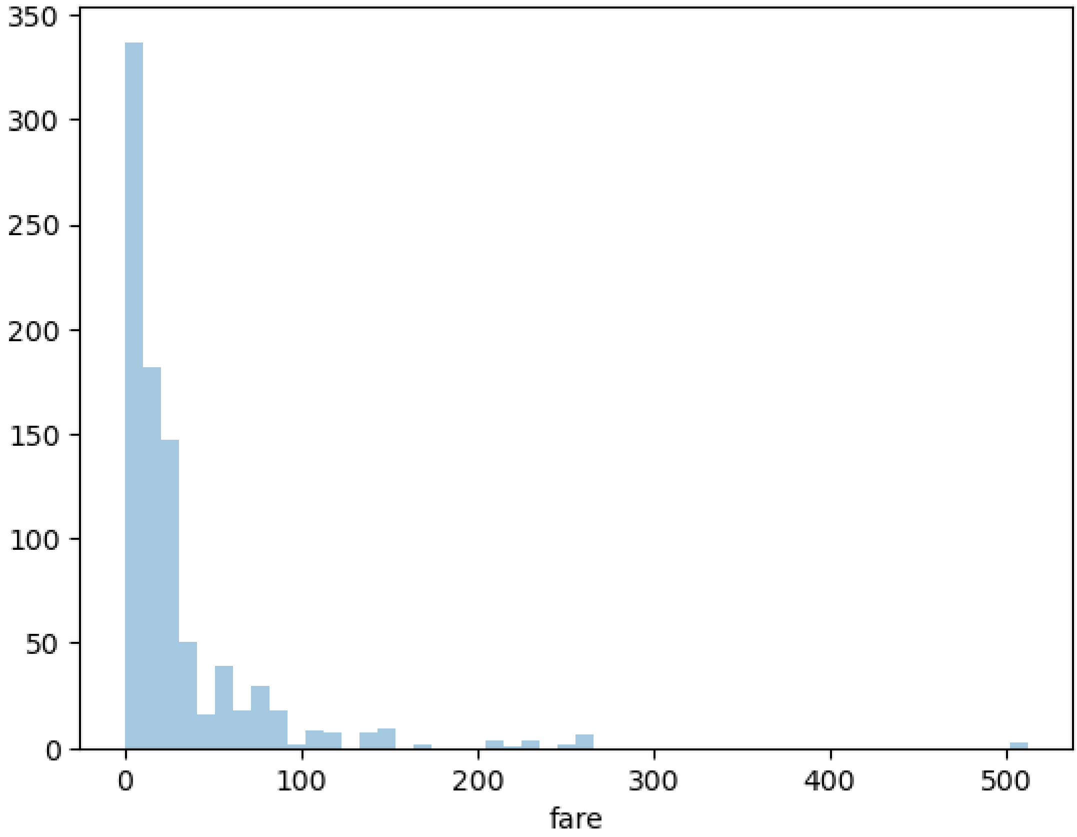
```python
sns.distplot(dataset['fare'], kde=False)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureW
arning: `distplot` is a deprecated function and will be removed in a future versio
n. Please adapt your code to use either `displot` (a figure-level function with si
milar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[5]:

```
<AxesSubplot:xlabel='fare'>
```



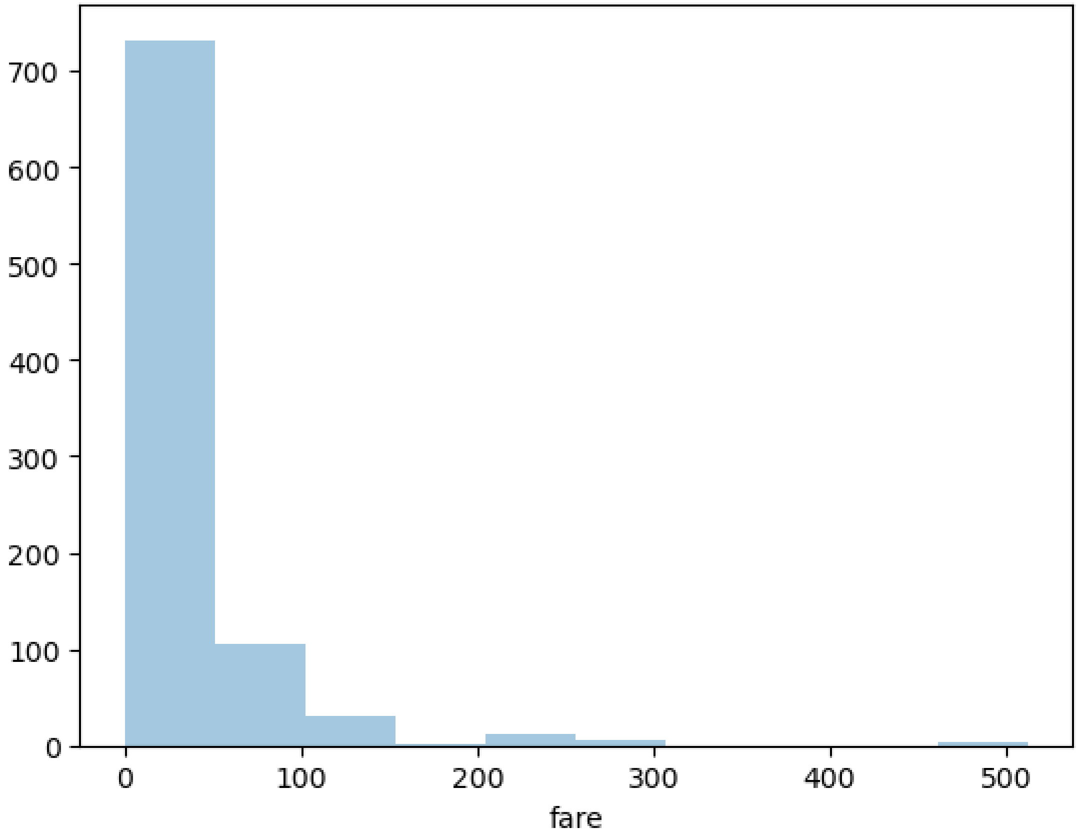
In [6]:

```python
sns.distplot(dataset['fare'], kde=False, bins=10)
```

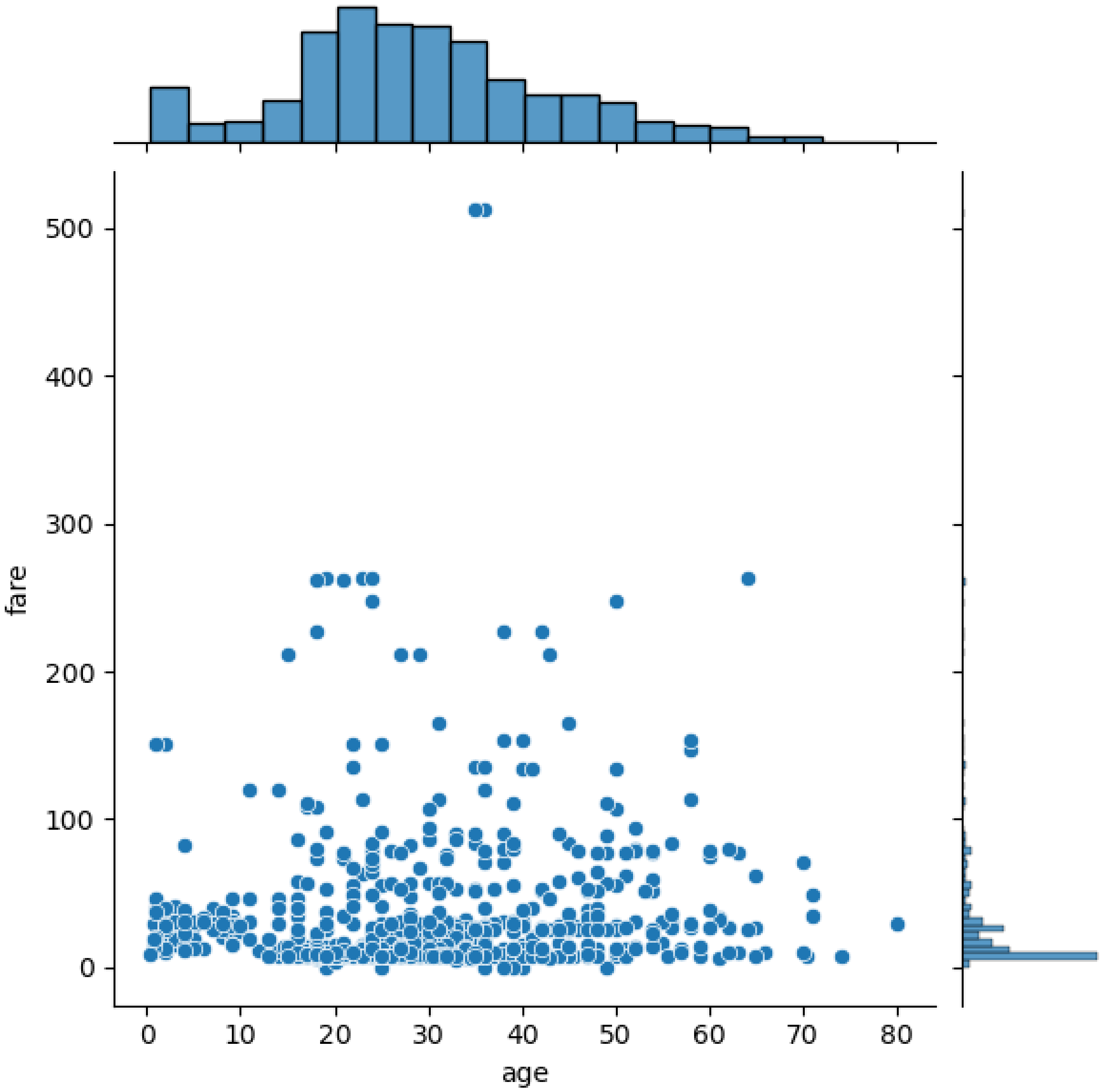
Out[6]:

```
<AxesSubplot:xlabel='fare'>
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



```
In [7]: sns.jointplot(x='age', y='fare', data=dataset)

Out[7]: <seaborn.axisgrid.JointGrid at 0x27a9bc08ac0>
```



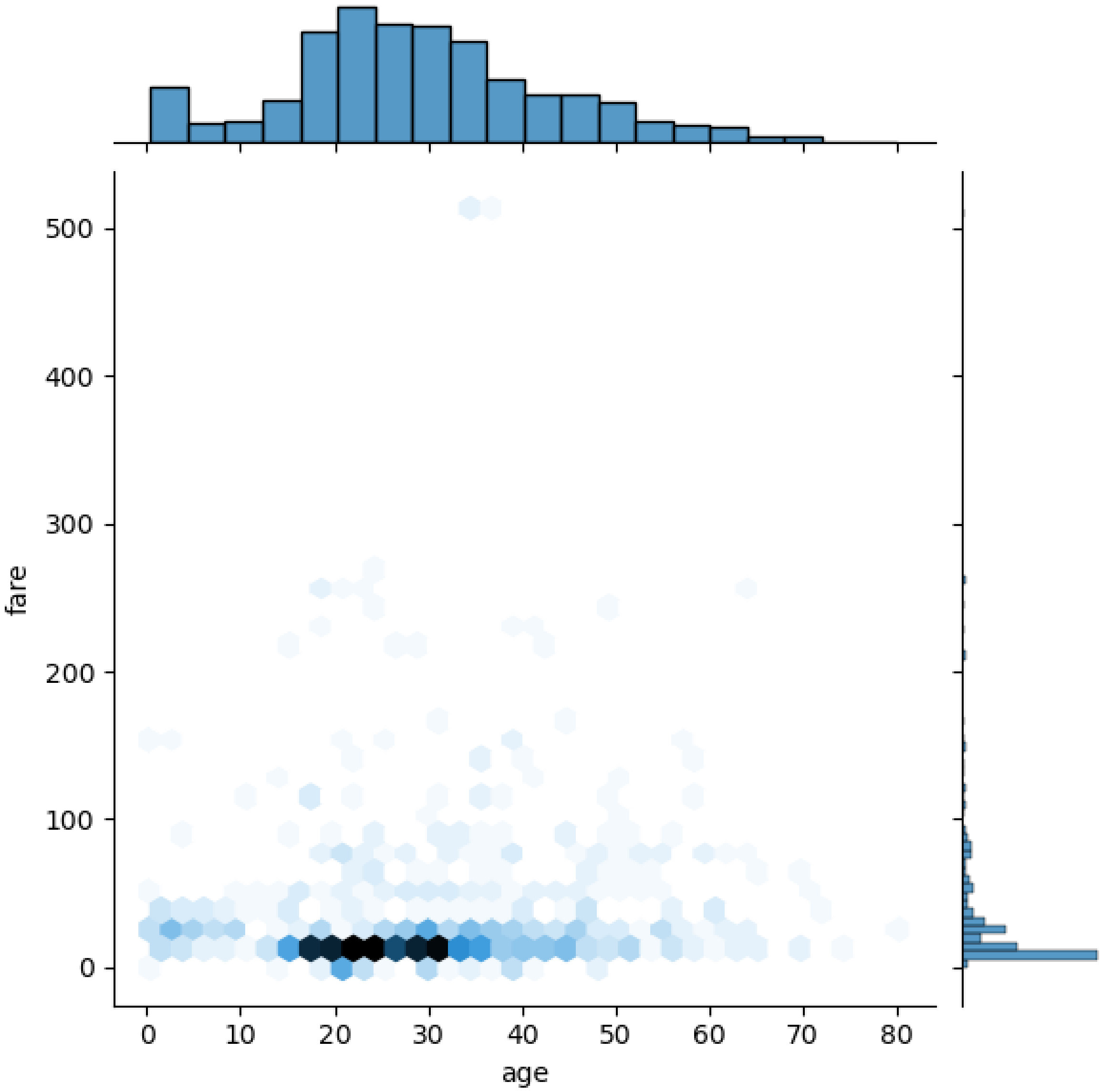
In [8]:

```
sns.jointplot(x='age', y='fare', data=dataset, kind='hex')
```

Out[8]:

```
<seaborn.axisgrid.JointGrid at 0x27a9c0adeb0>
```

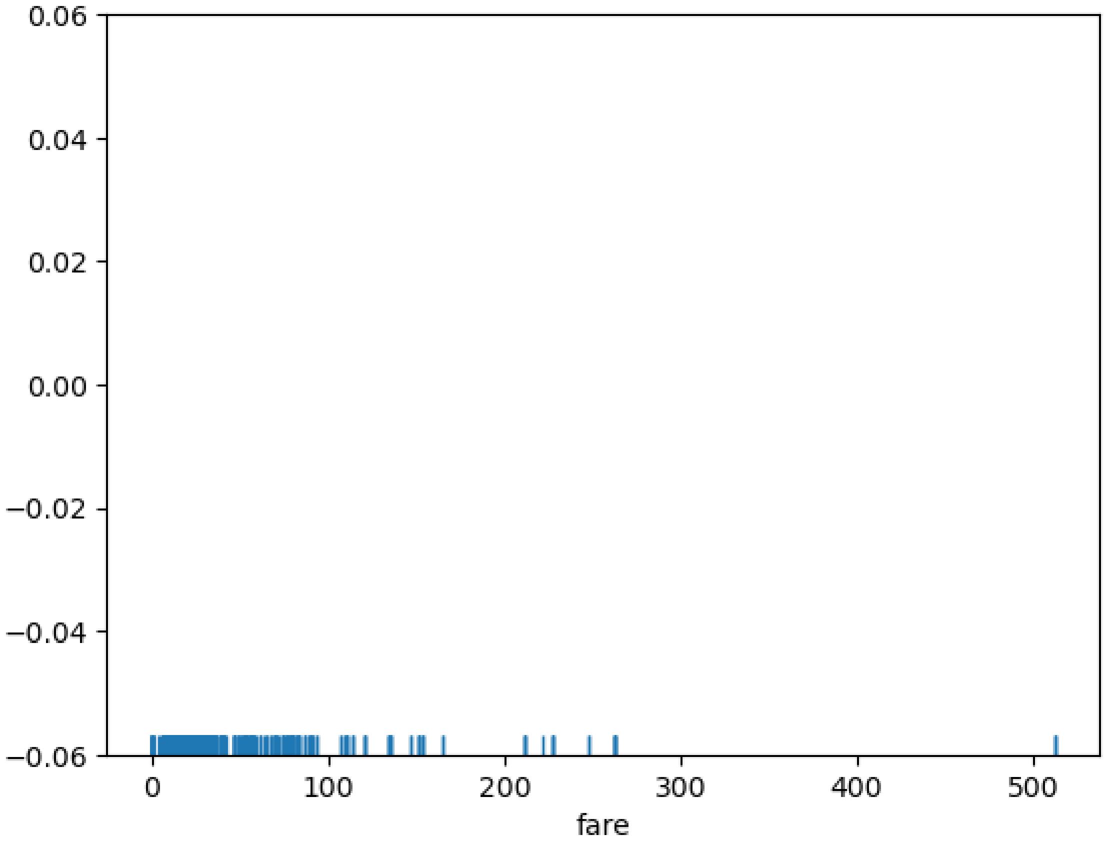
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



In [9]:
```
sns.rugplot(dataset['fare'])
```

Out[9]:
```
<AxesSubplot:xlabel='fare'>
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

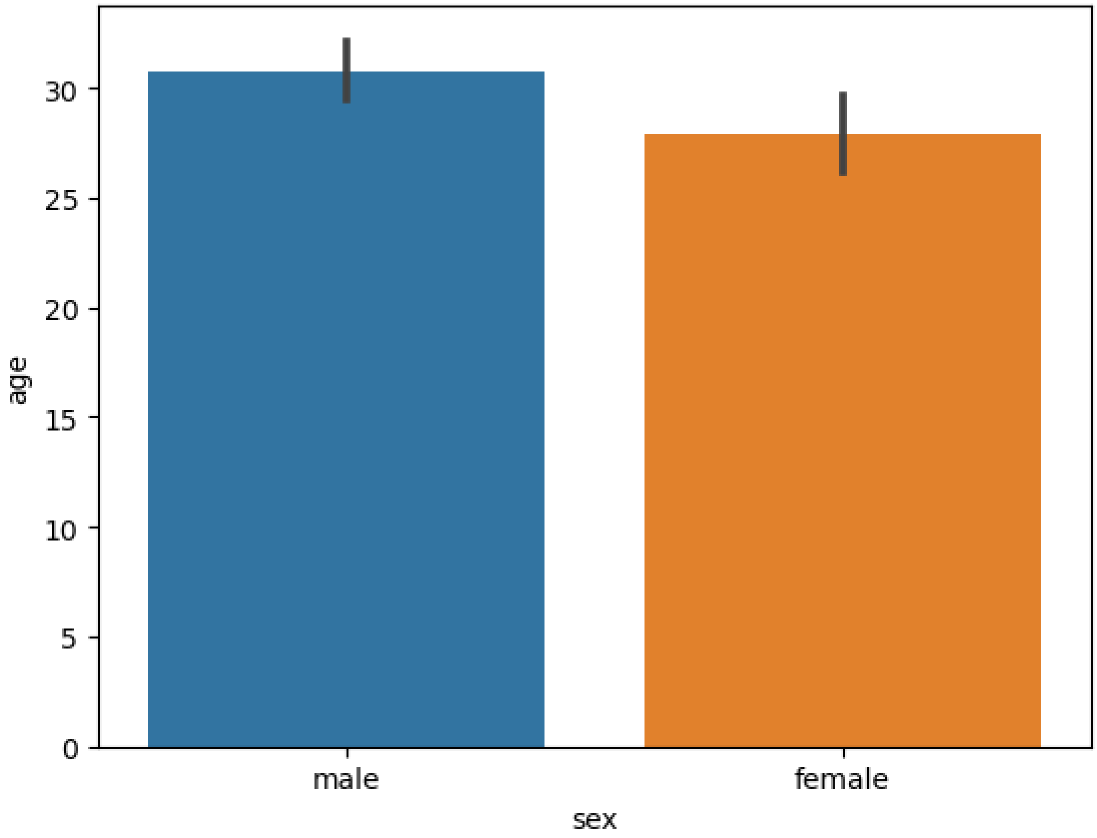


In [10]:

```
sns.barplot(x='sex', y='age', data=dataset)
```

Out[10]:

```
<AxesSubplot:xlabel='sex', ylabel='age'>
```



In [11]:

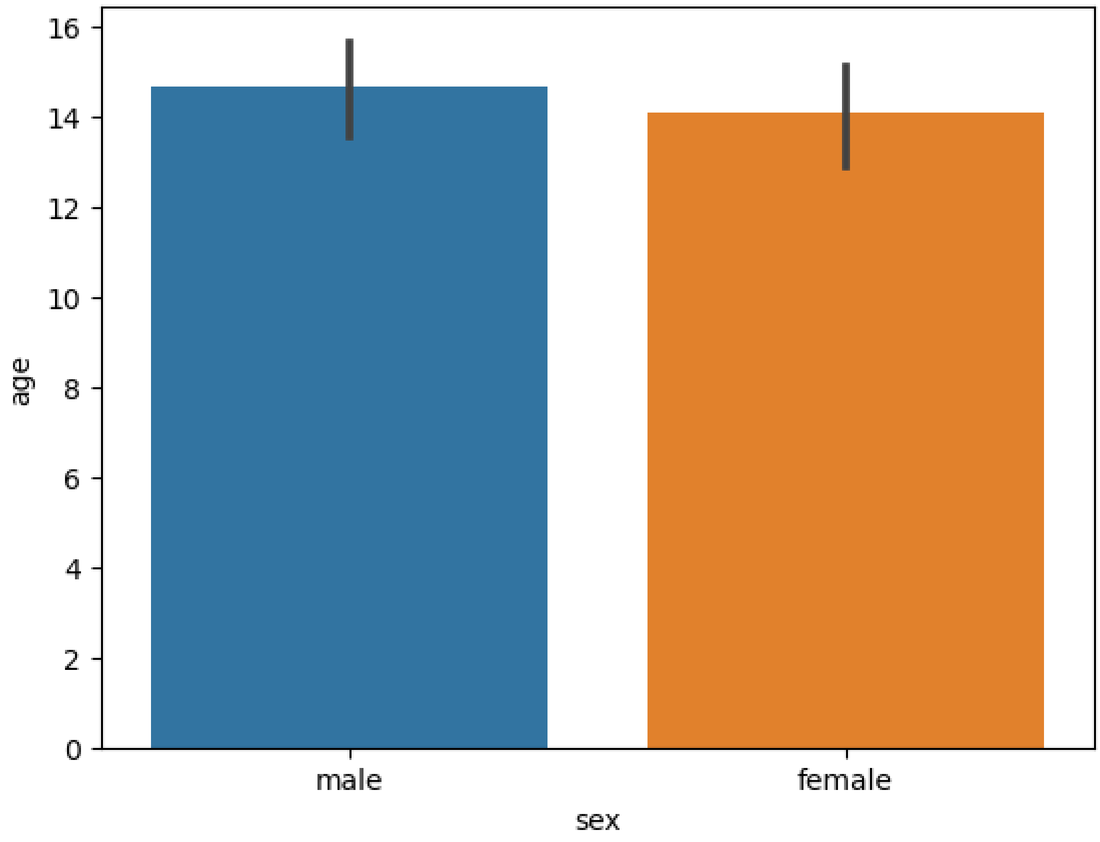
```
import numpy as np

import matplotlib.pyplot as plt
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
sns.barplot(x='sex', y='age', data=dataset, estimator=np.std)
```

Out[11]:

```
<AxesSubplot:xlabel='sex', ylabel='age'>
```

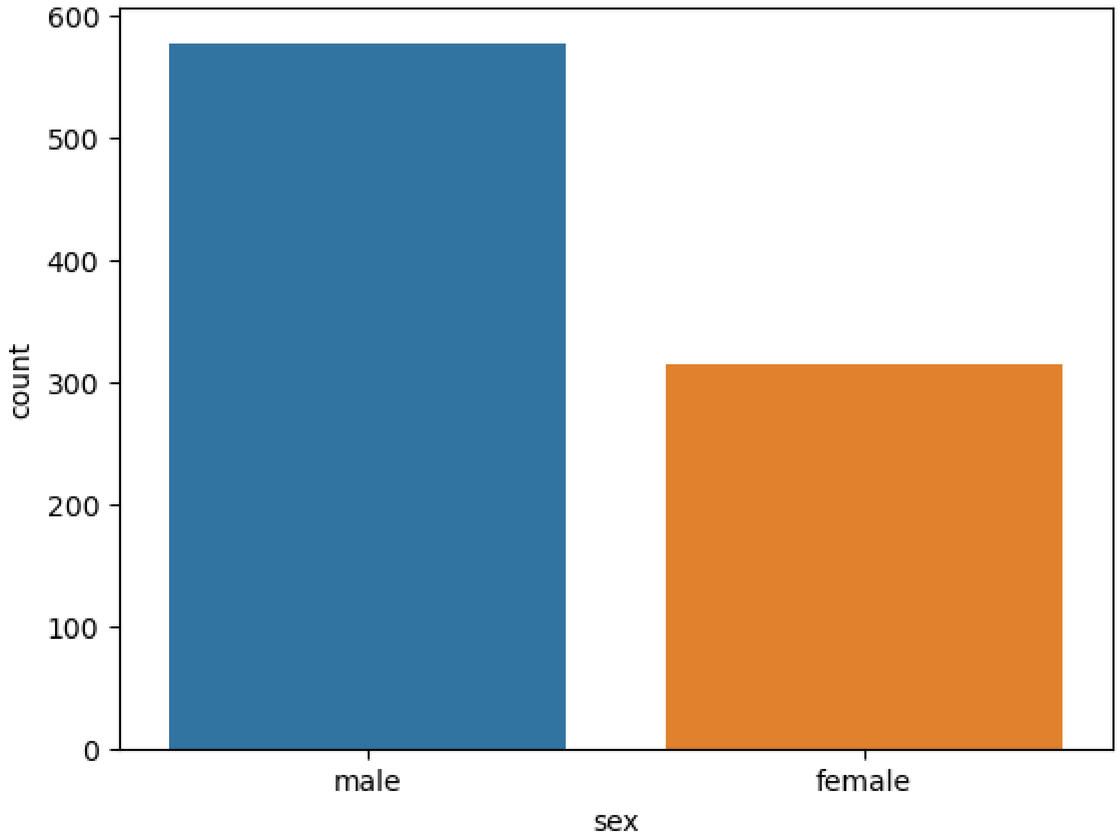


In [12]:

```
sns.countplot(x='sex', data=dataset)
```

Out[12]:

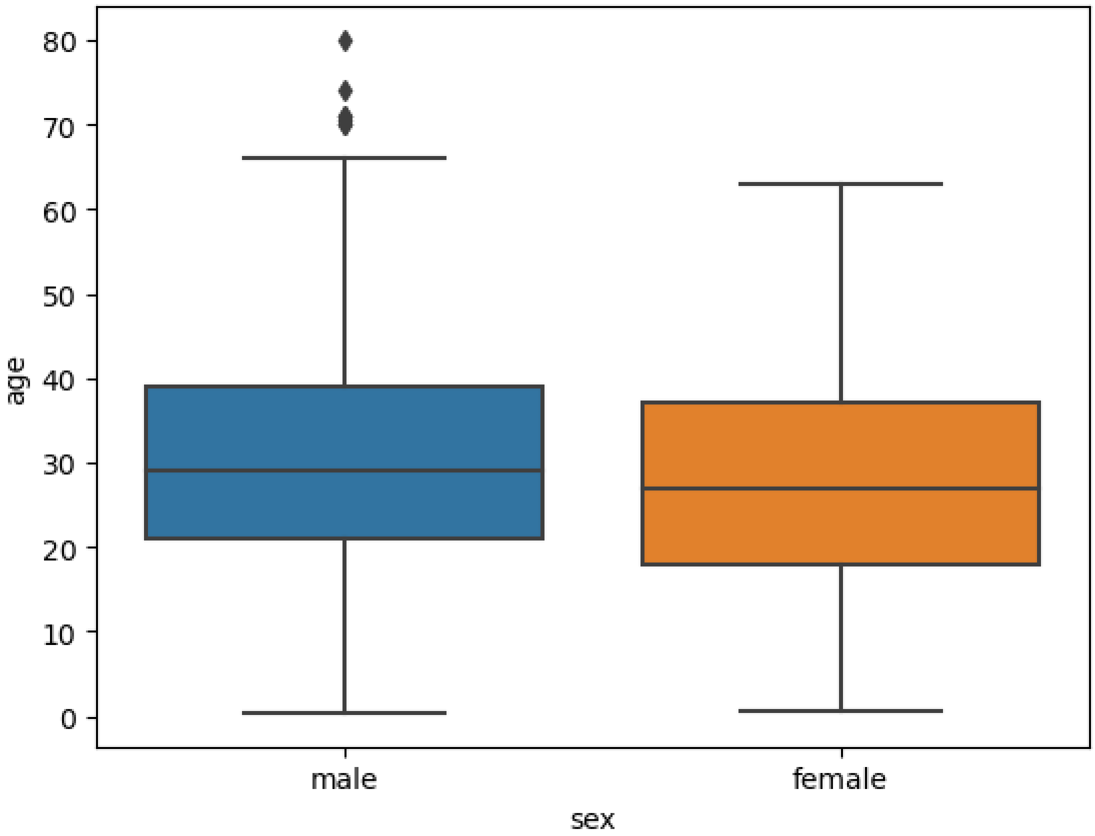
```
<AxesSubplot:xlabel='sex', ylabel='count'>
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

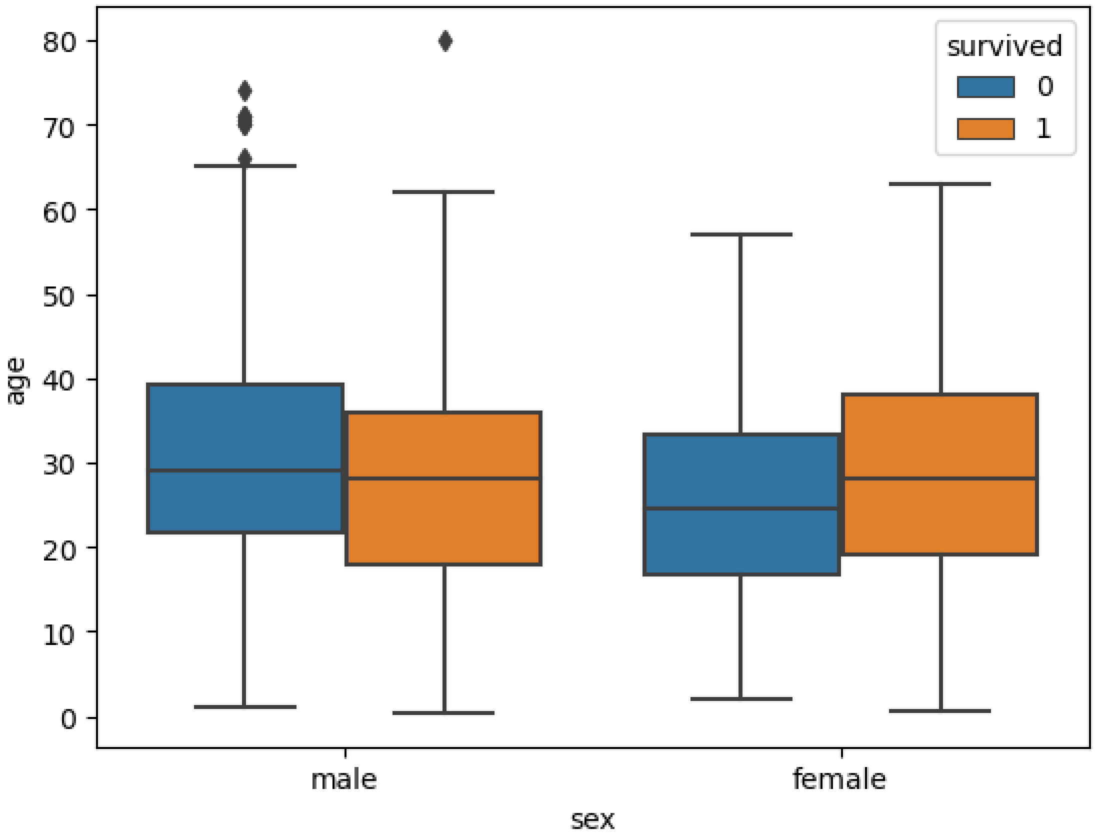
```
In [13]: sns.boxplot(x='sex', y='age', data=dataset)
```

```
Out[13]: <AxesSubplot:xlabel='sex', ylabel='age'>
```



```
In [14]: sns.boxplot(x='sex', y='age', data=dataset, hue="survived")
```

```
Out[14]: <AxesSubplot:xlabel='sex', ylabel='age'>
```



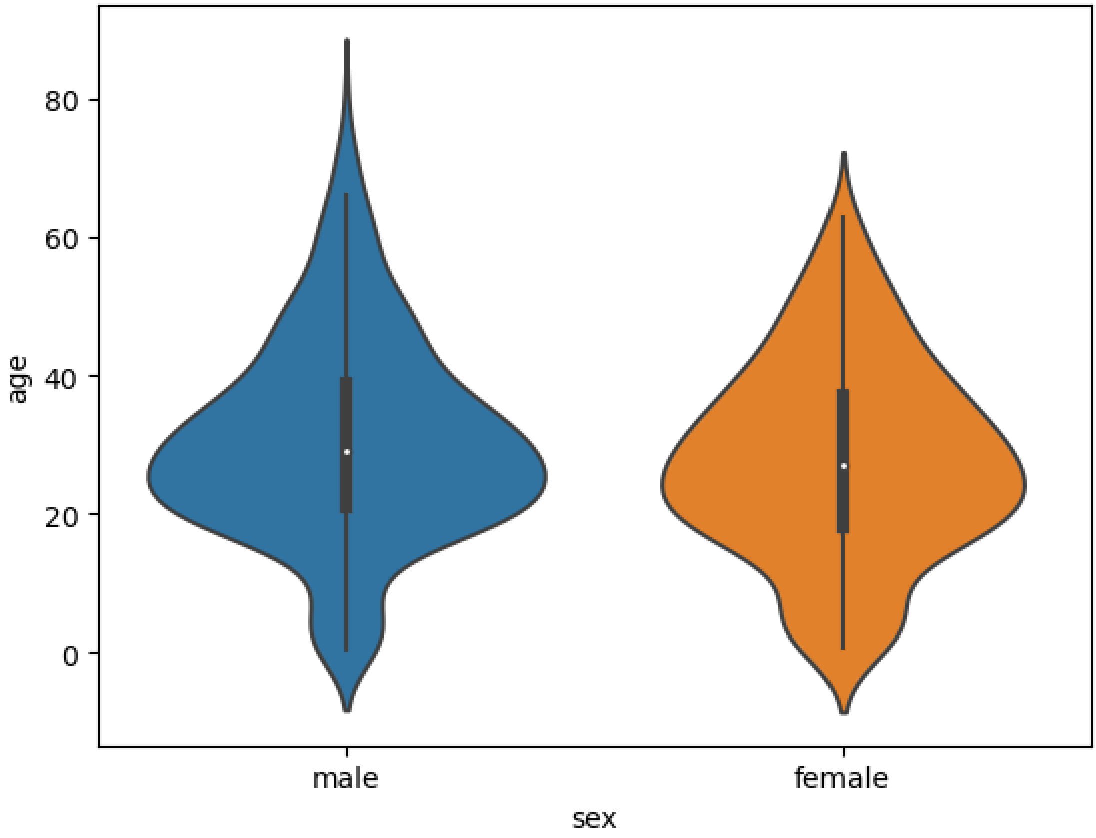
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [15]:

```
sns.violinplot(x='sex', y='age', data=dataset)
```

Out[15]:

```
<AxesSubplot:xlabel='sex', ylabel='age'>
```

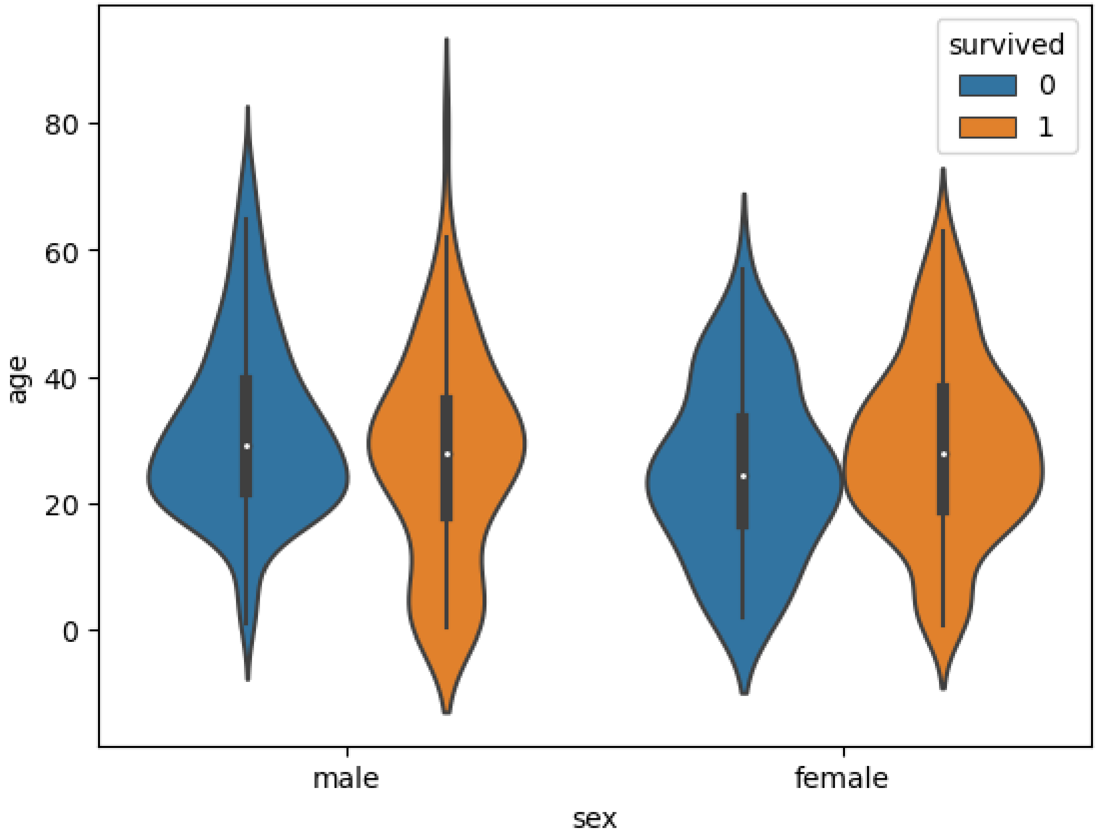


In [16]:

```
sns.violinplot(x='sex', y='age', data=dataset, hue='survived')
```

Out[16]:

```
<AxesSubplot:xlabel='sex', ylabel='age'>
```



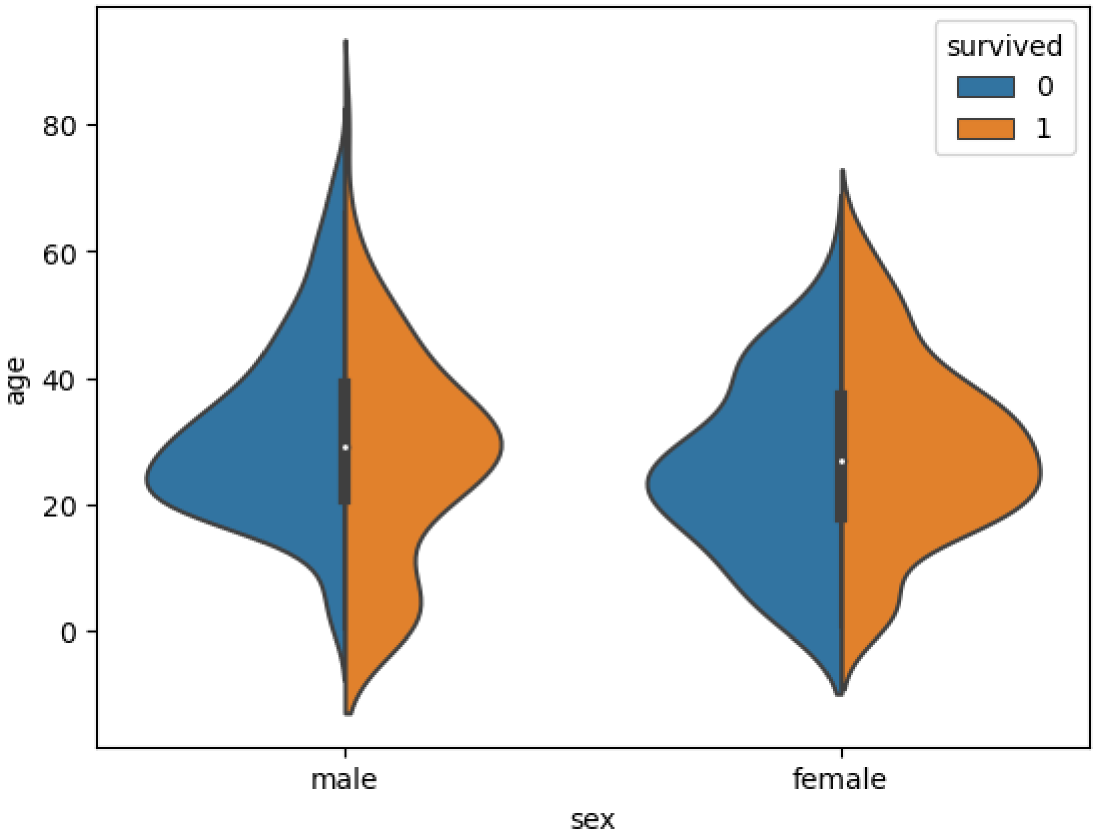
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [17]:

```
sns.violinplot(x='sex', y='age', data=dataset, hue='survived', split=True)
```

Out[17]:

```
<AxesSubplot:xlabel='sex', ylabel='age'>
```

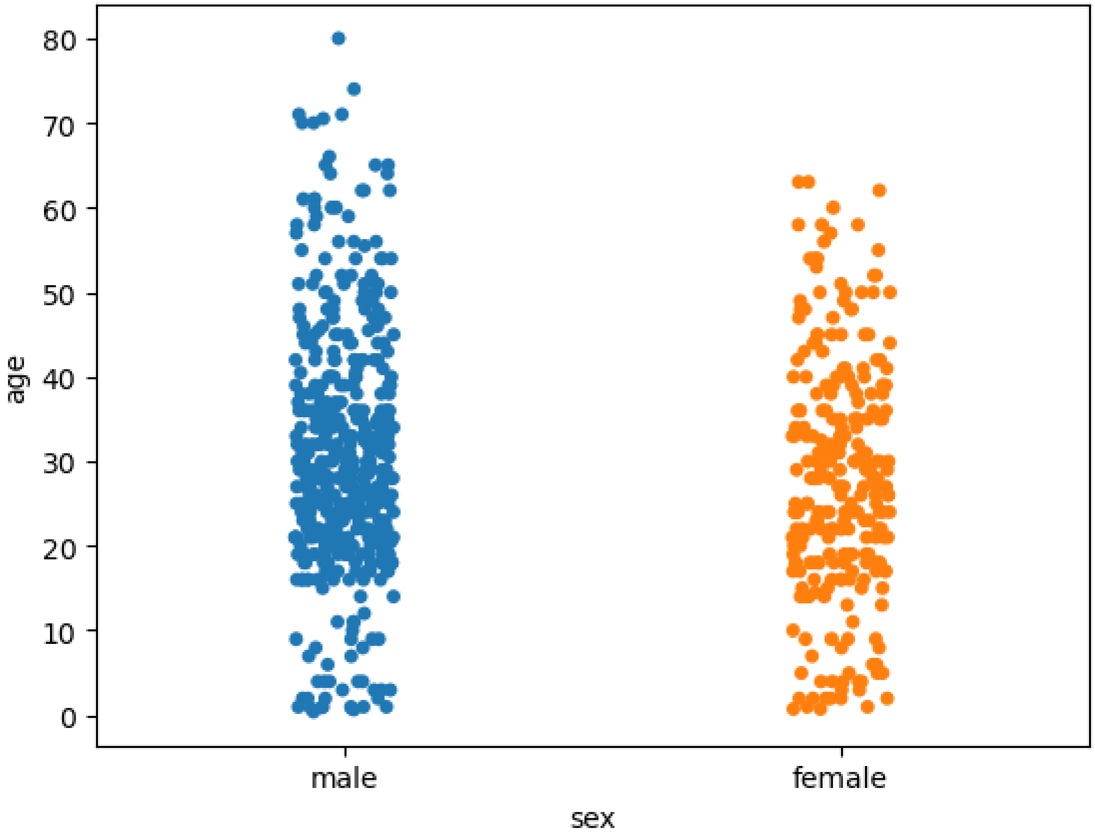


In [18]:

```
sns.stripplot(x='sex', y='age', data=dataset)
```

Out[18]:

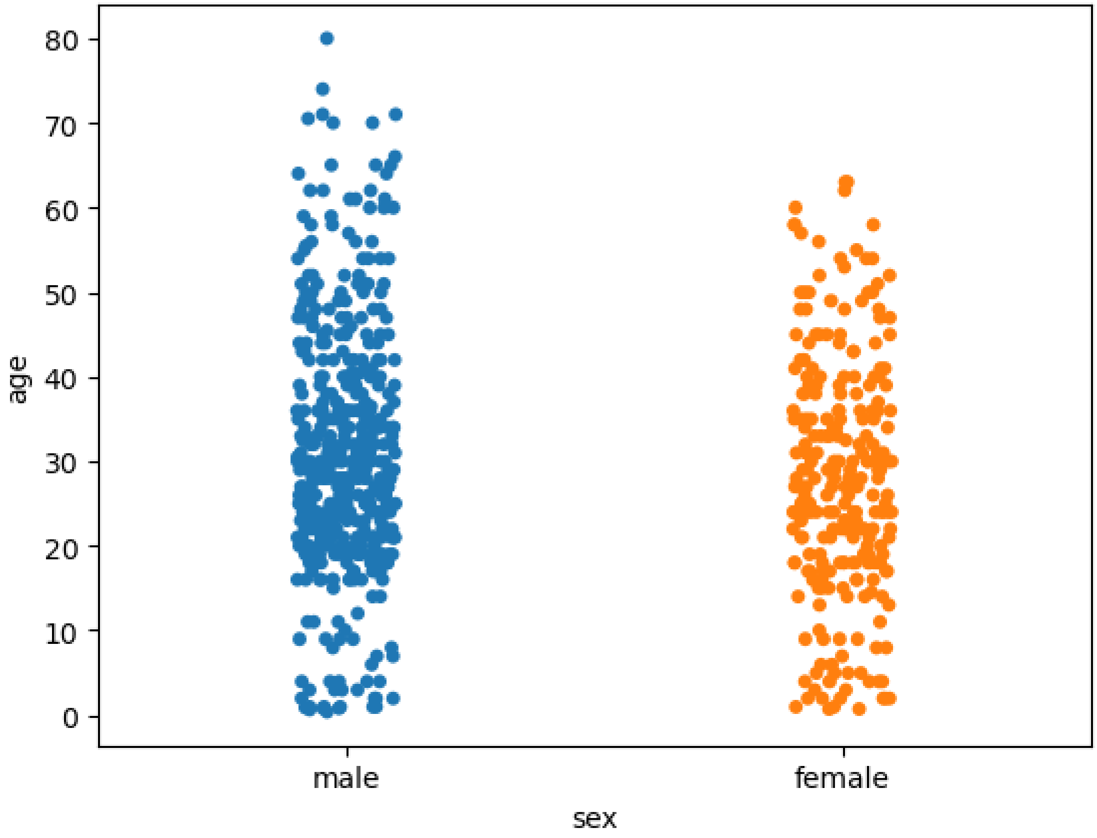
```
<AxesSubplot:xlabel='sex', ylabel='age'>
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

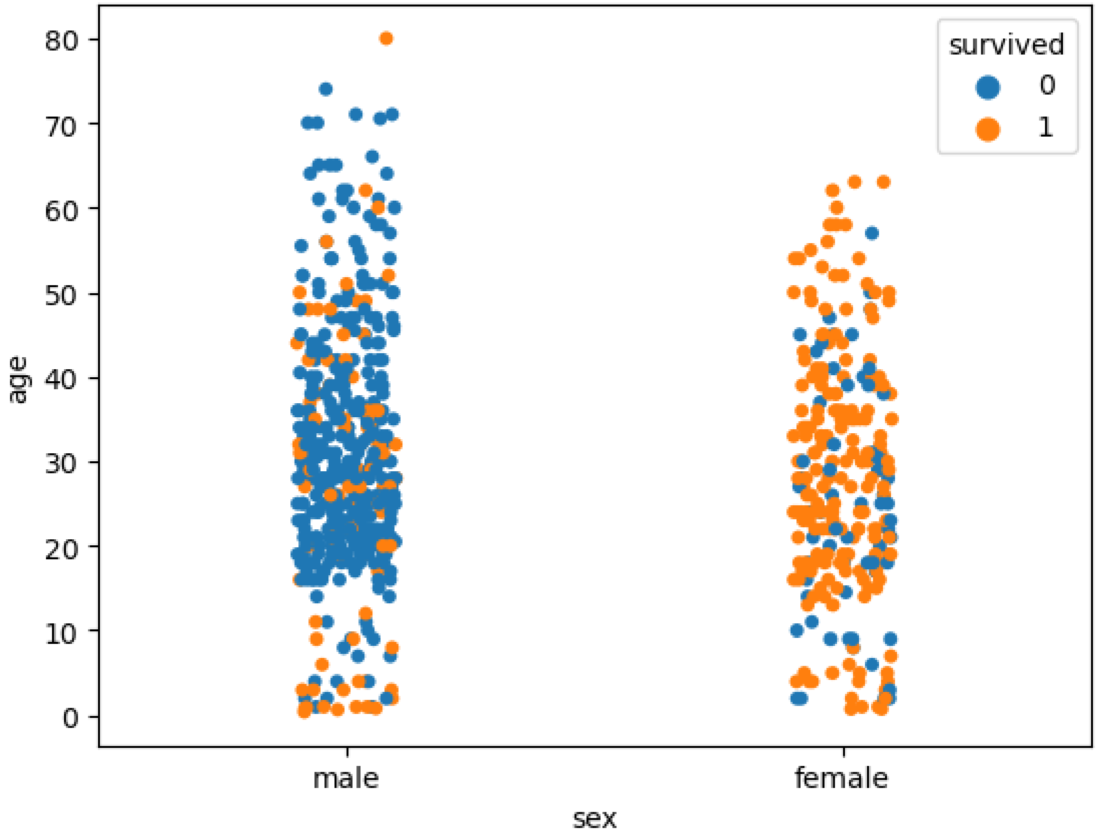
In [19]:
```
sns.stripplot(x='sex', y='age', data=dataset, jitter=True)
```

Out[19]:
```
<AxesSubplot:xlabel='sex', ylabel='age'>
```



In [20]:
```
sns.stripplot(x='sex', y='age', data=dataset, jitter=True, hue='survived')
```

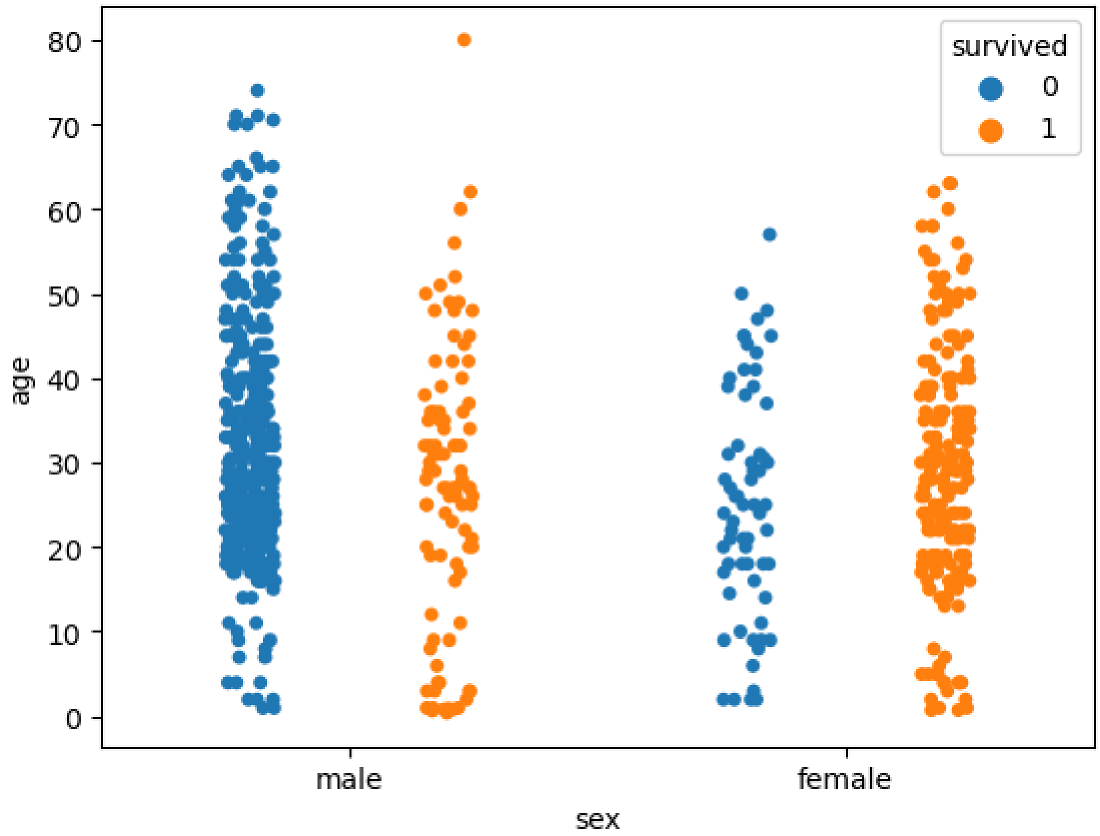
Out[20]:
```
<AxesSubplot:xlabel='sex', ylabel='age'>
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

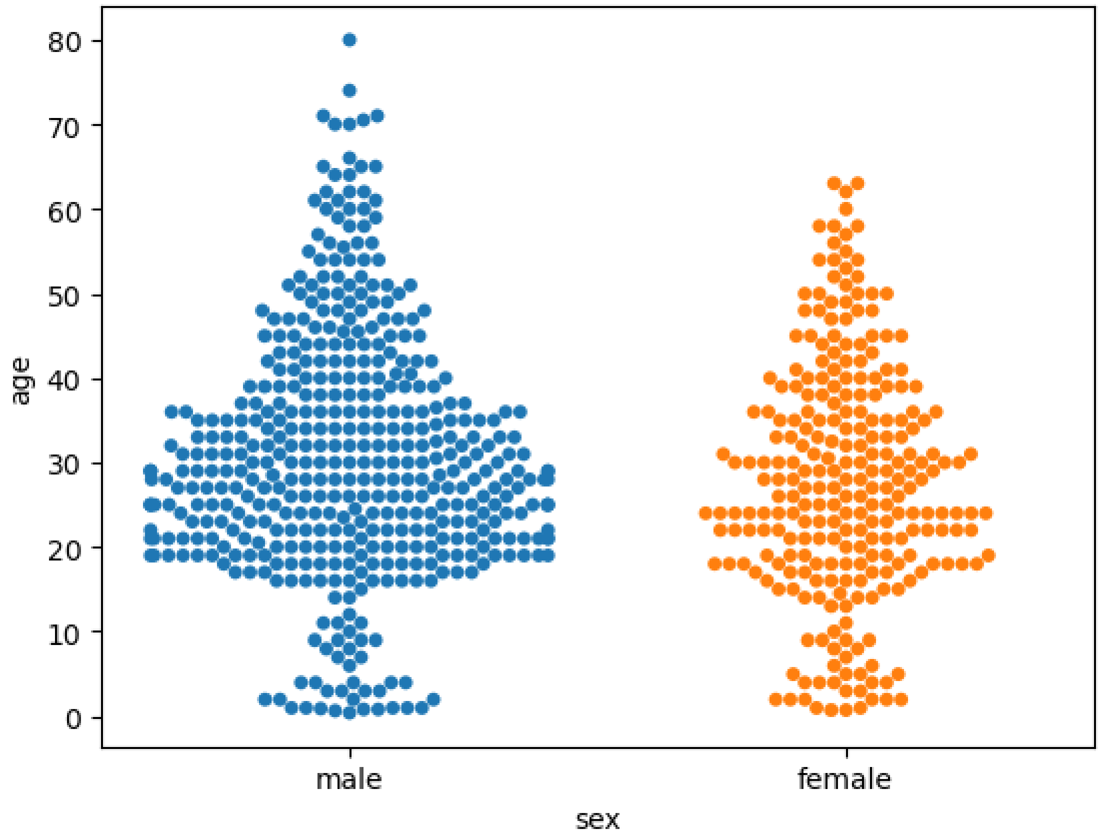
```
In [21]: sns.stripplot(x='sex', y='age', data=dataset, jitter=True, hue='survived', dodge=Tr
```

```
Out[21]: <AxesSubplot:xlabel='sex', ylabel='age'>
```



```
In [25]: sns.swarmplot(x='sex', y='age', data=dataset)
```

```
Out[25]: <AxesSubplot:xlabel='sex', ylabel='age'>
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [26]:
```
sns.matrixplot(x='sex', y='age', data=dataset)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_14272\2259532084.py in <module>
----> 1 sns.matrixplot(x='sex', y='age', data=dataset)

AttributeError: module 'seaborn' has no attribute 'matrixplot'
```

In [27]:
```
dataset.corr()
```

Out[27]:

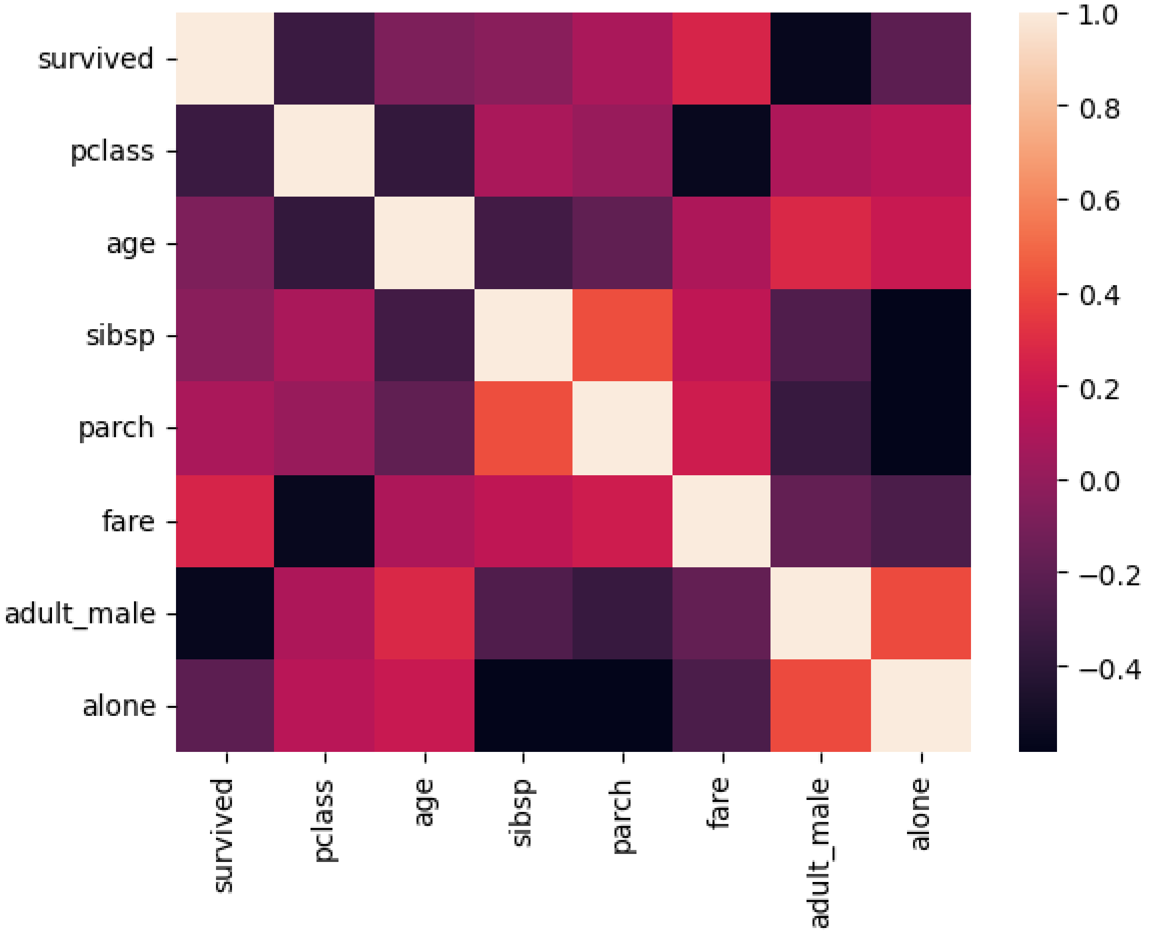
	survived	pclass	age	sibsp	parch	fare	adult_male	alone
survived	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307	-0.557080	-0.203367
pclass	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500	0.094035	0.135207
age	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067	0.280328	0.198270
sibsp	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651	-0.253586	-0.584471
parch	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225	-0.349943	-0.583398
fare	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000	-0.182024	-0.271832
adult_male	-0.557080	0.094035	0.280328	-0.253586	-0.349943	-0.182024	1.000000	0.404744
alone	-0.203367	0.135207	0.198270	-0.584471	-0.583398	-0.271832	0.404744	1.000000

In [28]:
```
corr=dataset.corr()
```

In [31]:
```
sns.heatmap(corr)
```

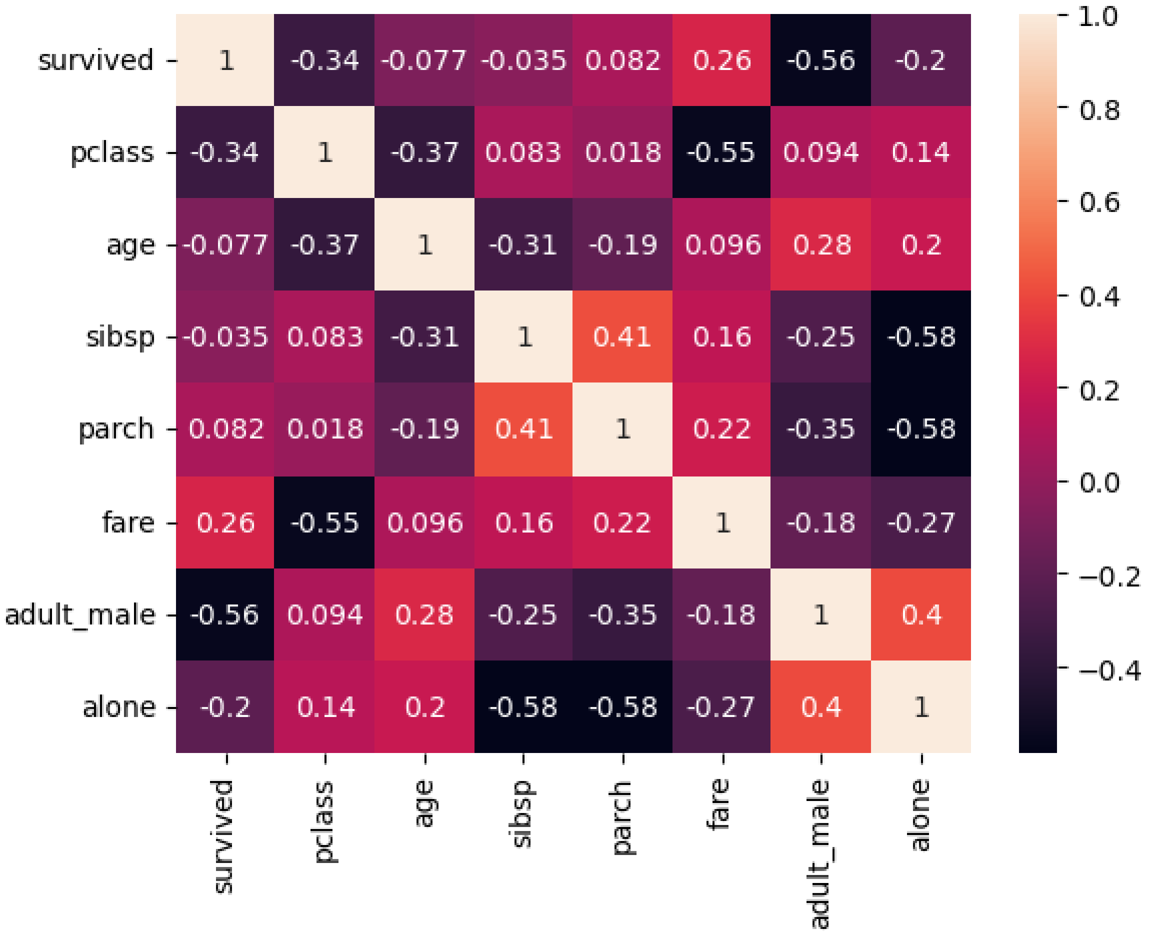
Out[31]:
```
<AxesSubplot:>
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



```
In [33]: sns.heatmap(corr, annot=True)

Out[33]: <AxesSubplot:>
```

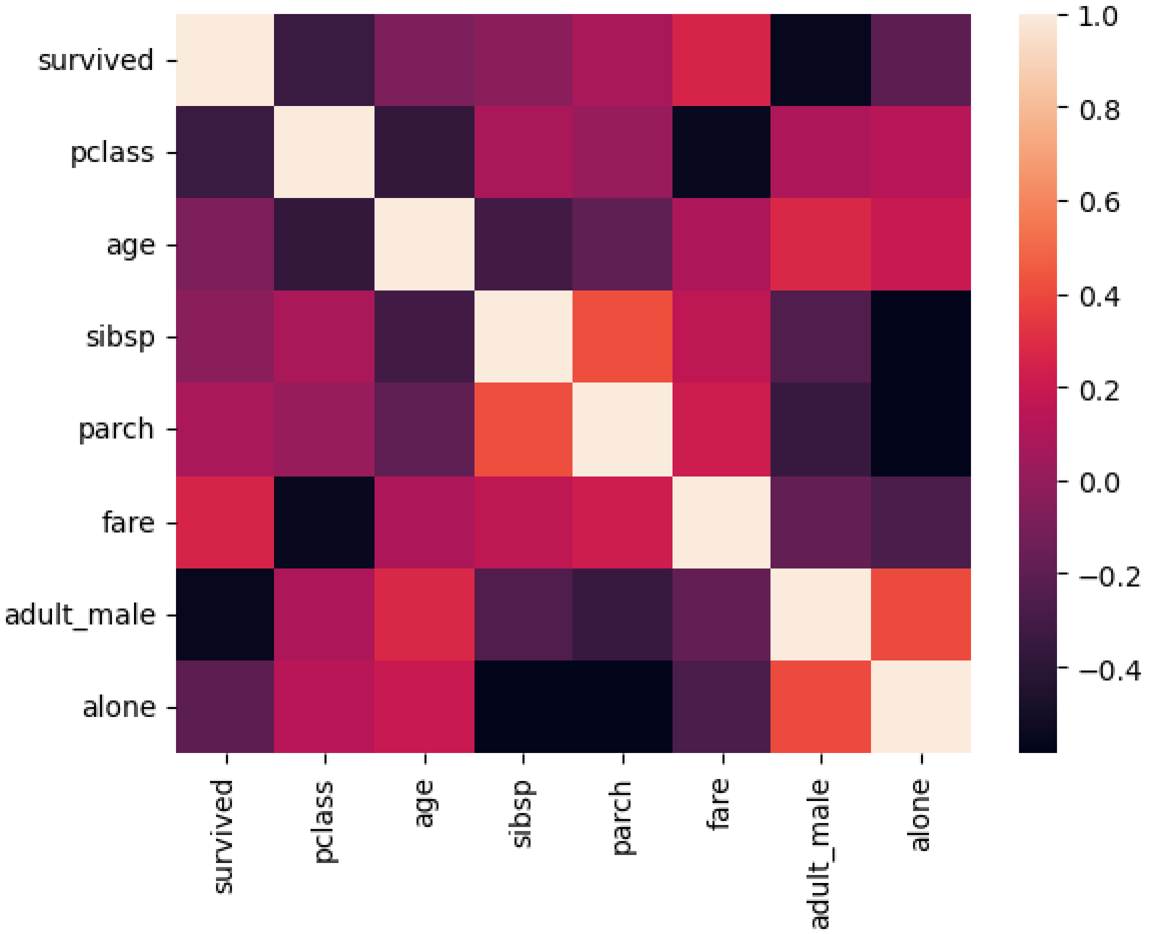
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



```
In [34]: sns.heatmap(corr)
```

```
Out[34]: <AxesSubplot:>
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



```
In [ ]:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js