DSBDAL-8

April 1, 2024

#**Data Visualization I** (without preprocessing of data)

1. Use the inbuilt dataset ‘titanic’. The dataset contains 891 rows and contains information about the passengers who boarded the unfortunate Titanic ship. Use the Seaborn library to see if we can find any patterns in the data.

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

Loading the Dataset

[2]:
```
df = sns.load_dataset('titanic')
df.head()
```

[2]:
```
   survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third
3         1       1  female  35.0      1      0  53.1000        S  First
4         0       3    male  35.0      0      0   8.0500        S  Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

[3]:
```
df.shape
```

[3]:
```
(891, 15)
```

[4]:
```
df.notnull()
```

[4]:
```
     survived  pclass   sex   age  sibsp  parch  fare  embarked  class   who  \
0        True    True  True  True   True   True  True      True   True  True
1        True    True  True  True   True   True  True      True   True  True
```

```
2        True    True   True   True   True   True   True      True   True  True
3        True    True   True   True   True   True   True      True   True  True
4        True    True   True   True   True   True   True      True   True  True
..        …      …   …    …    …    …   …       …    …   …
886      True    True   True   True   True   True   True      True   True  True
887      True    True   True   True   True   True   True      True   True  True
888      True    True   True  False   True   True   True      True   True  True
889      True    True   True   True   True   True   True      True   True  True
890      True    True   True   True   True   True   True      True   True  True

     adult_male   deck  embark_town  alive  alone
0          True  False         True   True   True
1          True   True         True   True   True
2          True  False         True   True   True
3          True   True         True   True   True
4          True  False         True   True   True
..           …    …          …    …    …
886        True  False         True   True   True
887        True   True         True   True   True
888        True  False         True   True   True
889        True   True         True   True   True
890        True  False         True   True   True

[891 rows x 15 columns]
```

[5]:
```
df.isnull().sum()
```

[5]:
```
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
```

Finding patterns of data

1. Distribution plots :These plots help us to visualise the distribution of data. We can use these plots to understand the mean, median, range, variance, deviation, etc of the data.

a. Distplot : gives us the histogram of the selected continuous variable.

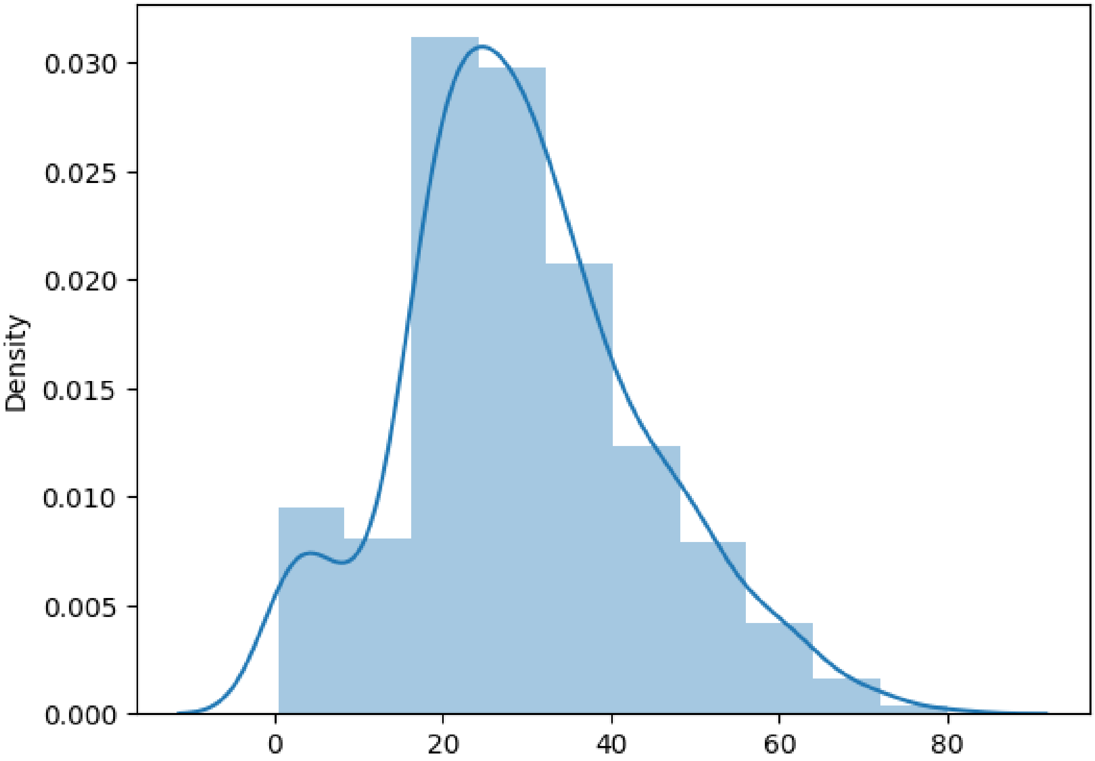
[6]:

```
sns.distplot (x = df['age'], bins = 10)
```

```
<ipython-input-6-23974792078a>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

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

  sns.distplot (x = df['age'], bins = 10)
```

[6]:

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



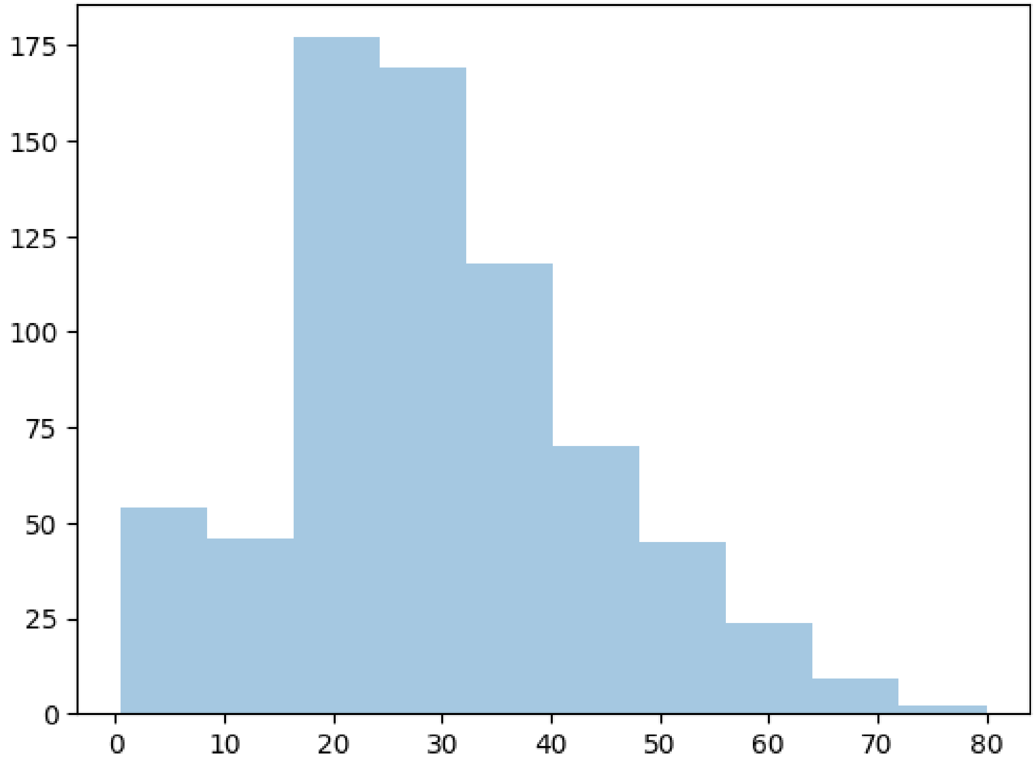
[7]:

```
sns.distplot (x = df['age'], bins = 10, kde = False)
```

```
<ipython-input-7-70918041e863>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.
```

```
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

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

  sns.distplot (x = df['age'], bins = 10, kde = False)
```

[7]:
```
<Axes: >
```

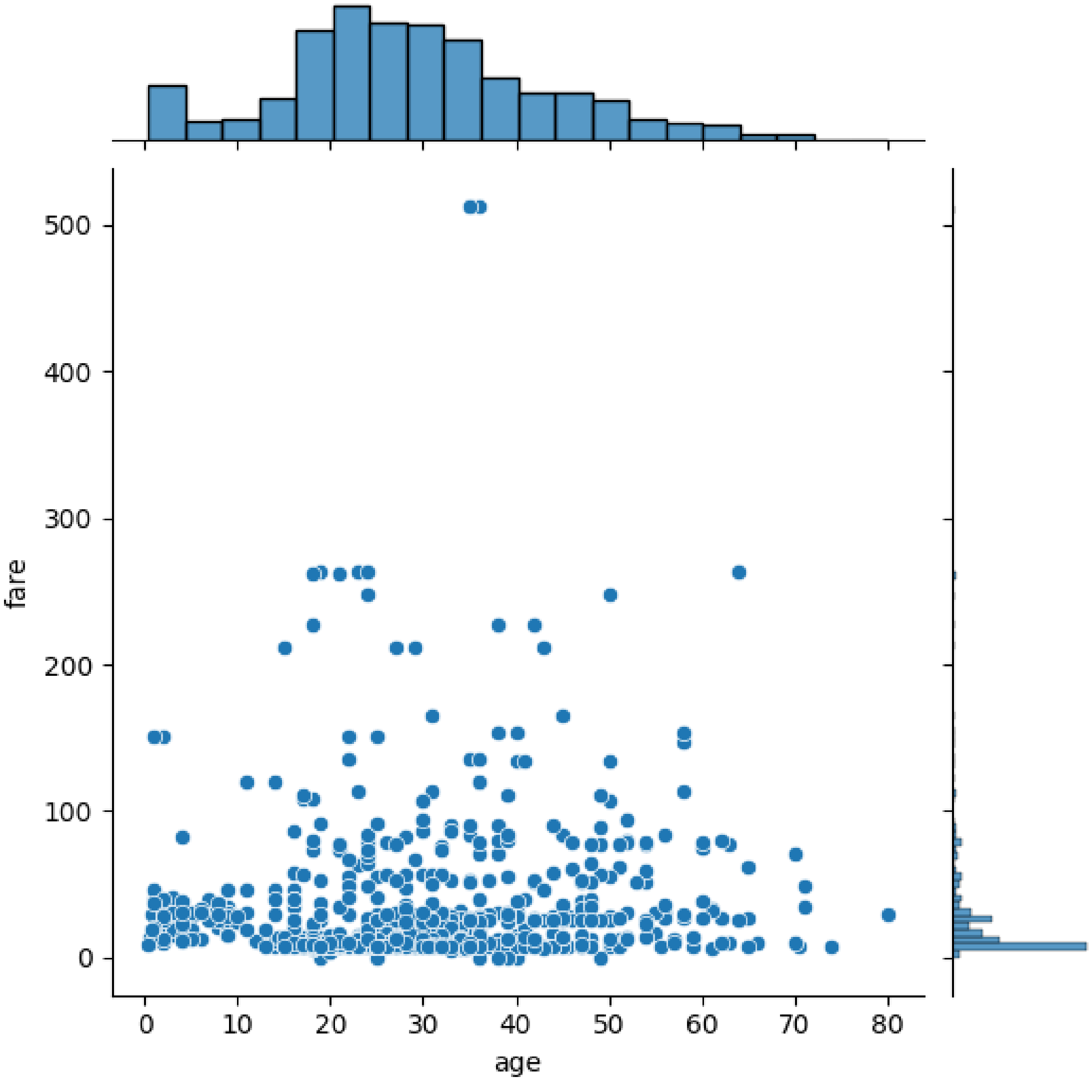


Here the x-axis is the age and the y-axis displays frequency. For example, for bins = 10, there are around 50 people having age 0 to 10

b. Joint plot: It is the combination of the distplot of two variables. It is an example of bivariate analysis.

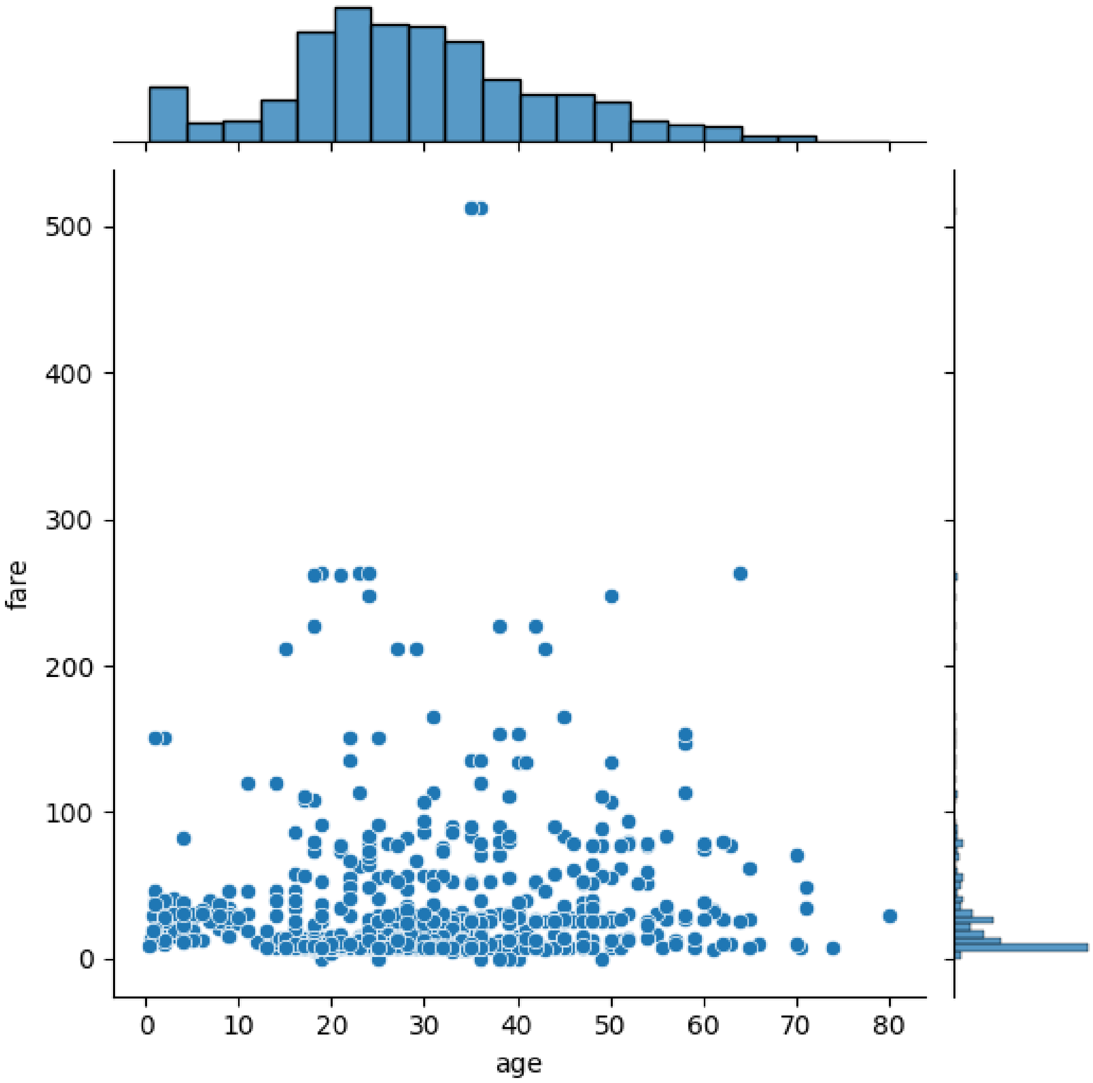
[8]:
```
sns.jointplot (x = df['age'], y = df['fare'])
```

[8]:
```
<seaborn.axisgrid.JointGrid at 0x7e09f4421180>
```



```
[9]: sns.jointplot (x = df['age'], y = df['fare'], kind = 'scatter')
```

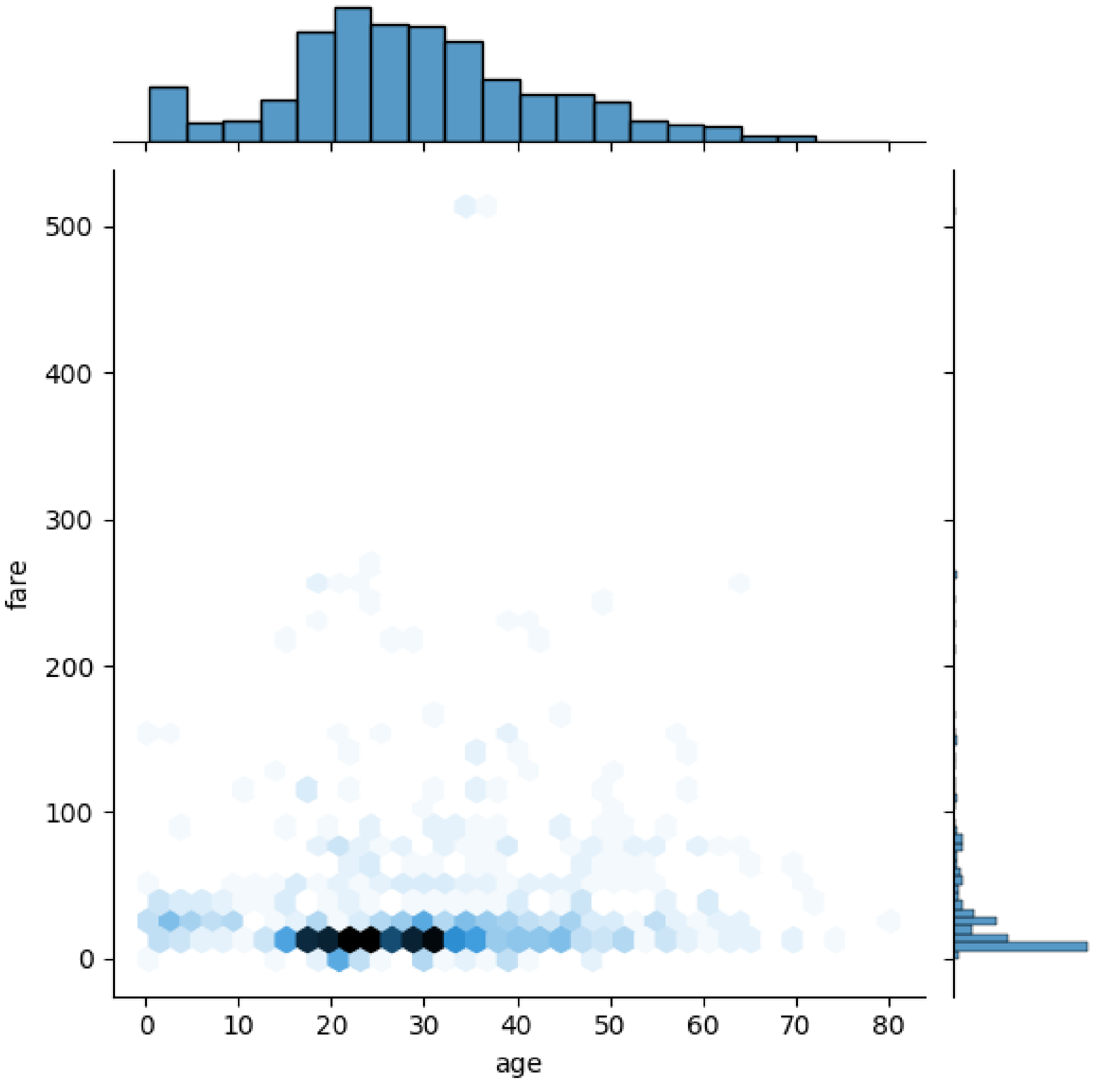
```
[9]: <seaborn.axisgrid.JointGrid at 0x7e09b1c0a1d0>
```



there is no correlation observed between prices and the fares.

[10]:
```
sns.jointplot (x = df['age'], y = df['fare'], kind = 'hex')
```

[10]:
```
<seaborn.axisgrid.JointGrid at 0x7e09b17bd360>
```

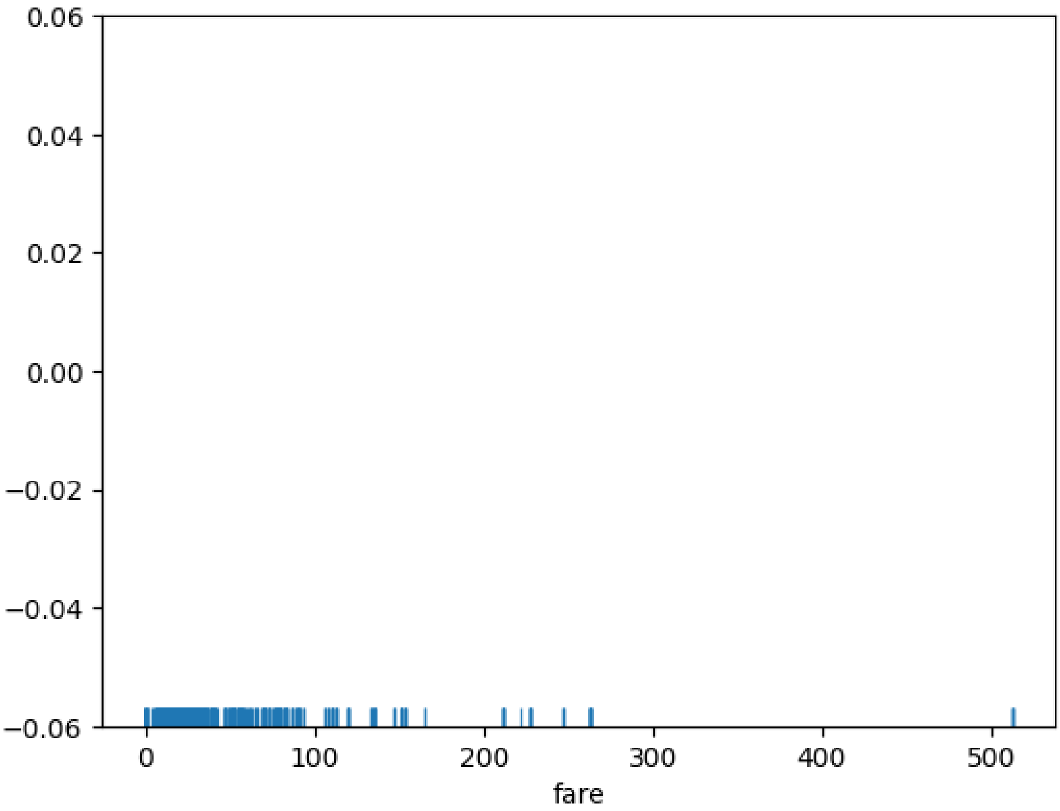


So if you look at the above plot, you can see that most of the passengers are between the ages of 20 and 30 and most of them paid between 10-50 for the tickets.

c. Rug plot: The rugplot() is used to draw small bars along the x-axis for each point in the dataset.

```
[11]: sns.rugplot(df['fare'])
```

```
[11]: <Axes: xlabel='fare'>
```



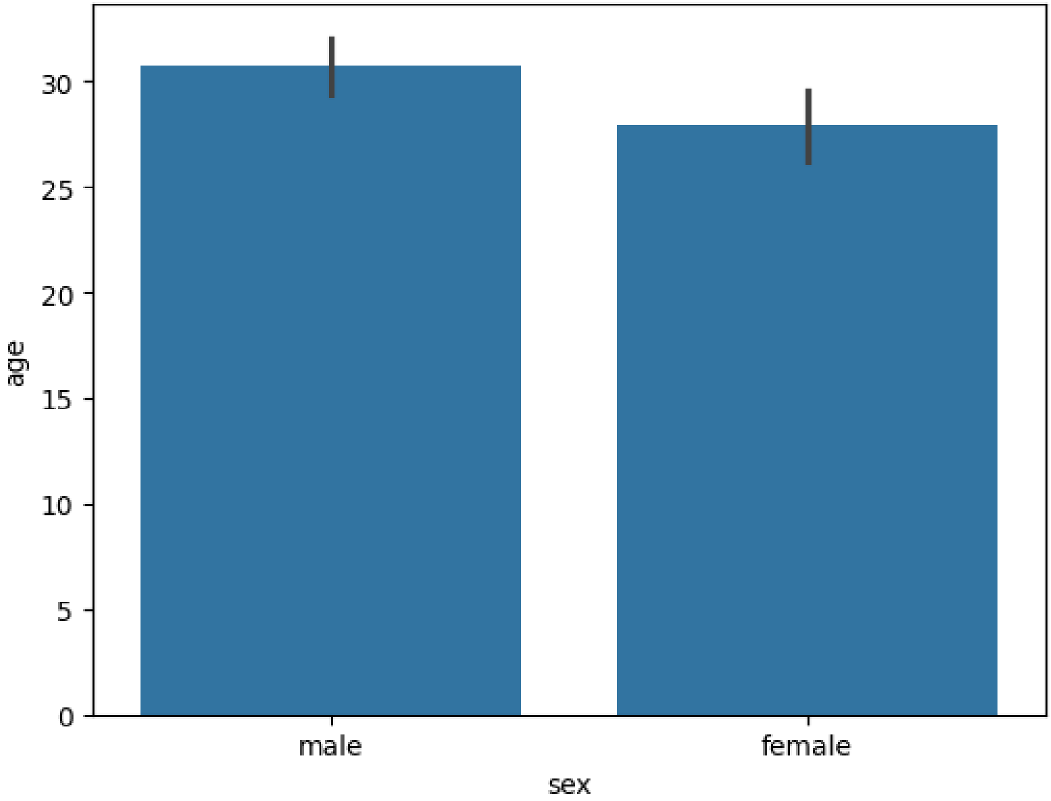
From the output, you can see that most of the instances for the fares have values between 0 and 100.

2. Categorical plots: used to plot categorical data. The categorical plots plot the values in the categorical column against another categorical column or a numeric column.

i. Bar Plots: The barplot() is used to display the mean value for each value in a categorical column, against a numeric column.

```
[12]: sns.barplot (x = 'sex', y = 'age', data = df)
```

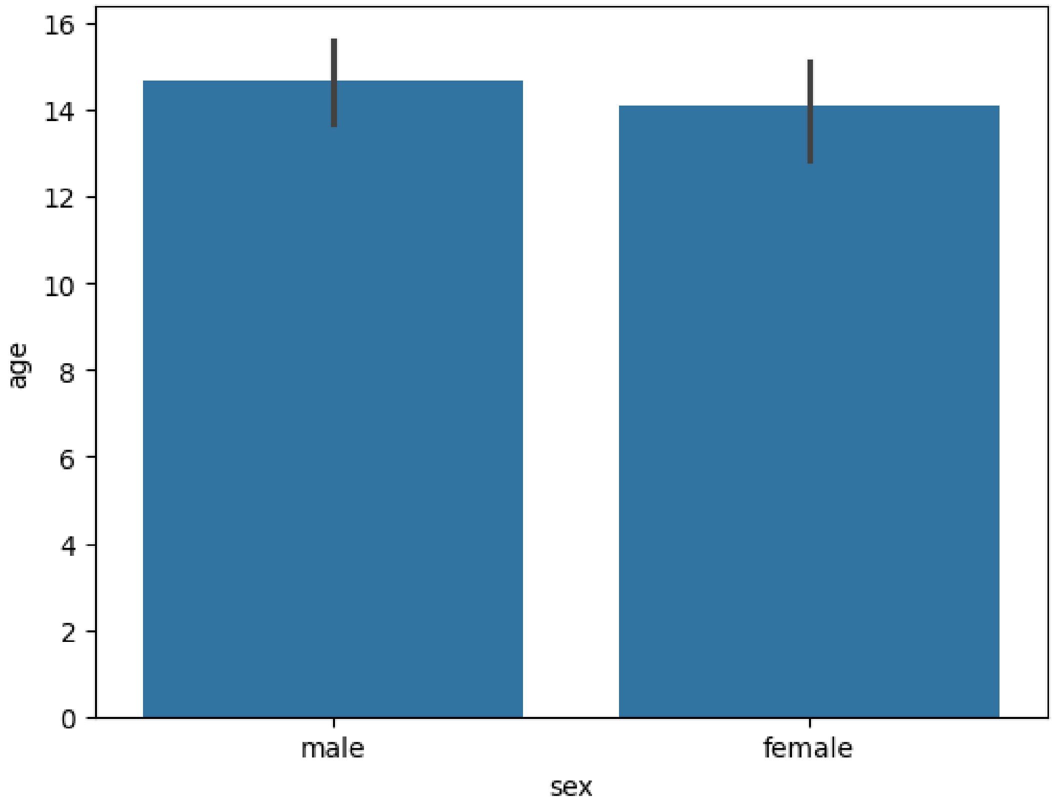
```
[12]: <Axes: xlabel='sex', ylabel='age'>
```



the average age of male passengers is just less than 40 while the average age of female passengers is around 33.

```
[13]: sns.barplot (x = 'sex', y = 'age', data = df, estimator = np.std)
```

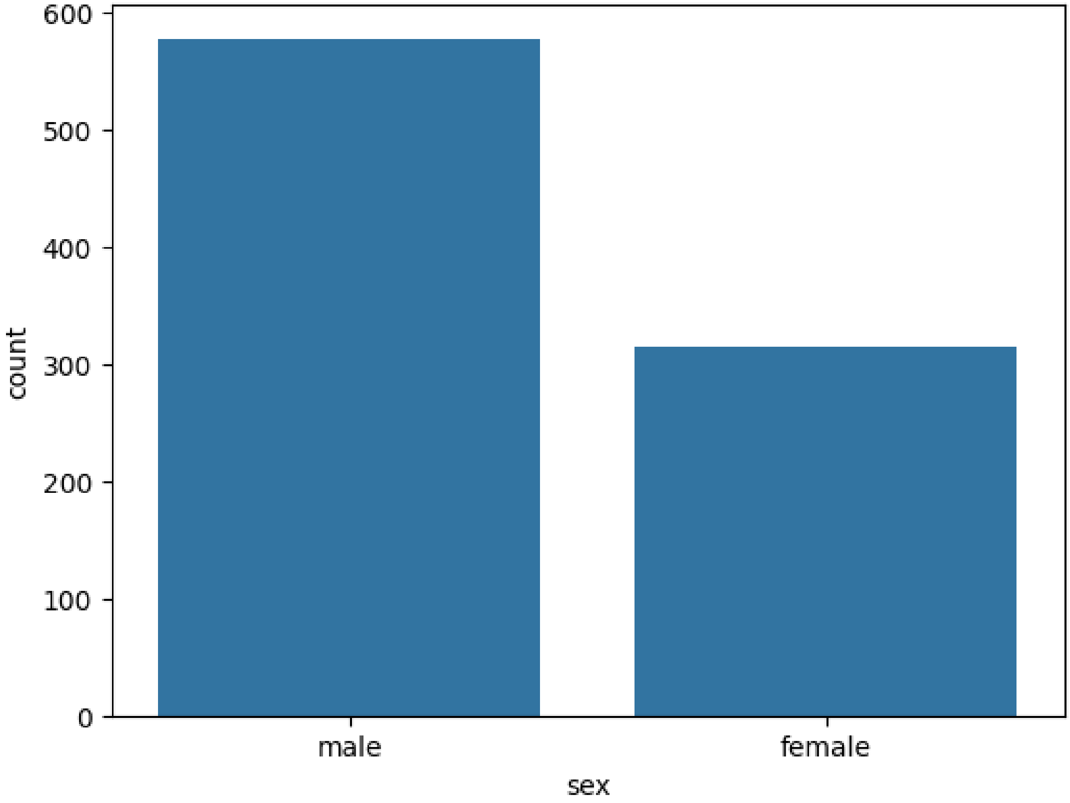
```
[13]: <Axes: xlabel='sex', ylabel='age'>
```



ii. Count Plot: The count plot is similar to the bar plot, however it displays the count of the categories in a specific column.

[14]:
```
sns.countplot(x = 'sex', data = df)
```

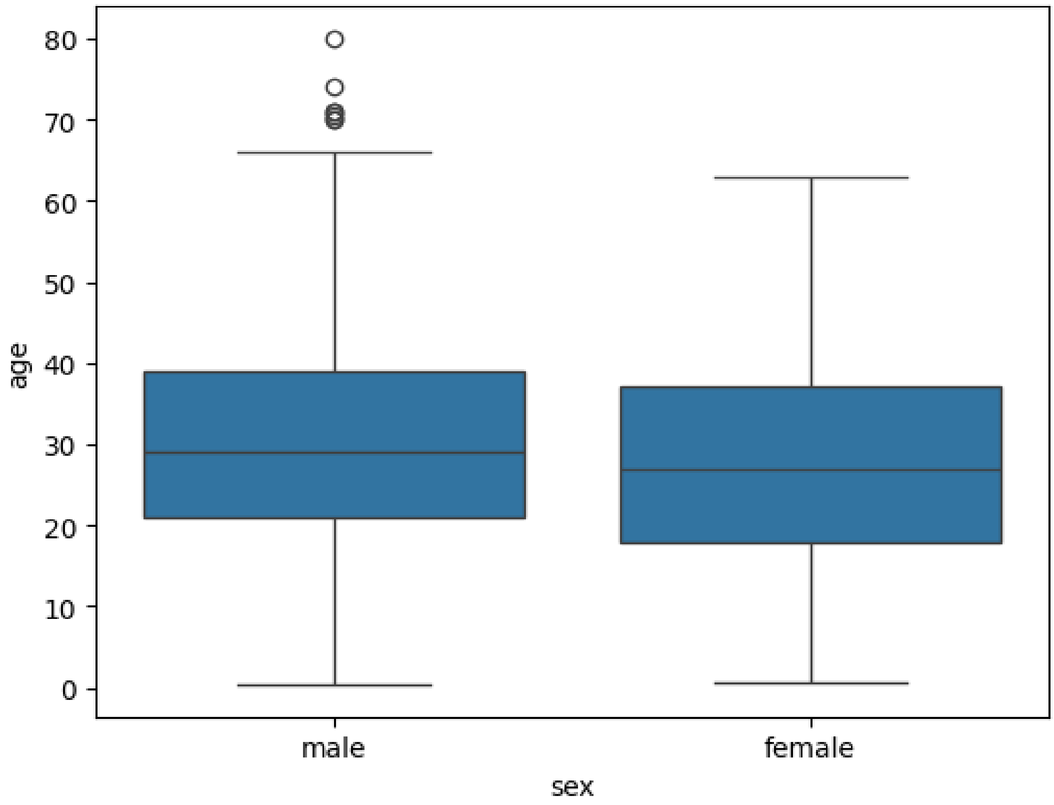
[14]:
```
<Axes: xlabel='sex', ylabel='count'>
```



iii. Box Plot: The box plot is used to display the distribution of the categorical data in the form of quartiles. The centre of the box shows the median value.

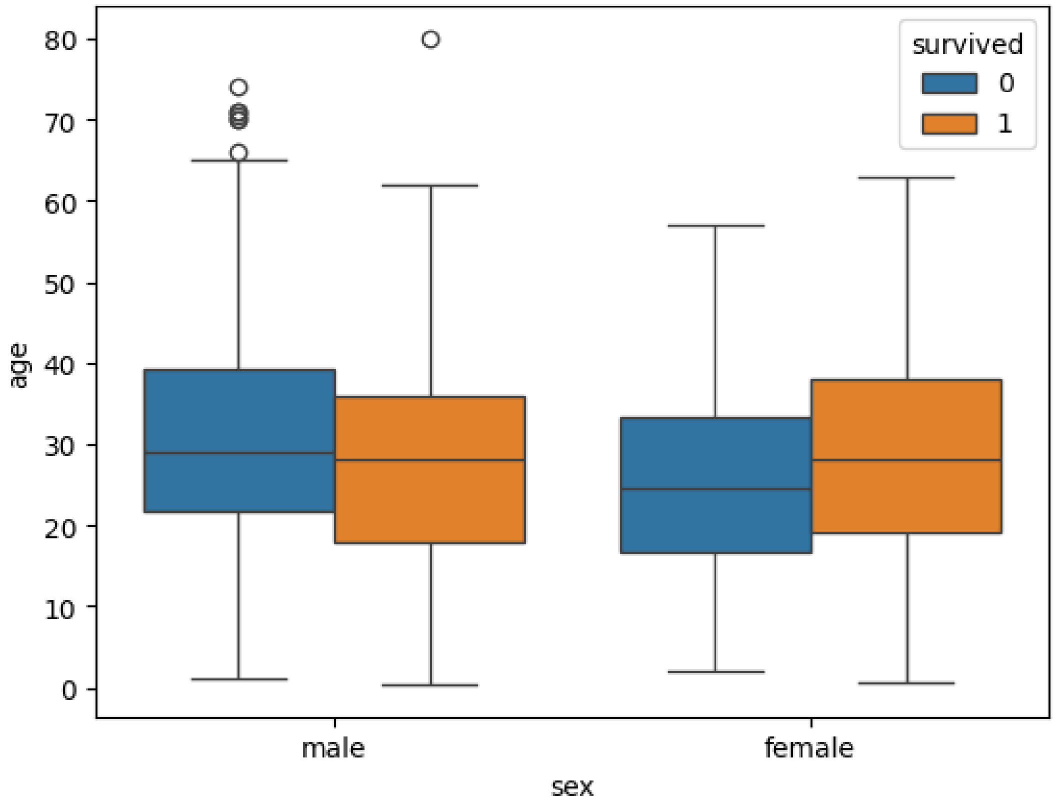
```
[15]: sns.boxplot (x = 'sex', y = 'age', data = df)
```

```
[15]: <Axes: xlabel='sex', ylabel='age'>
```



```
[16]: sns.boxplot (x = 'sex', y = 'age', data = df, hue = 'survived')
```

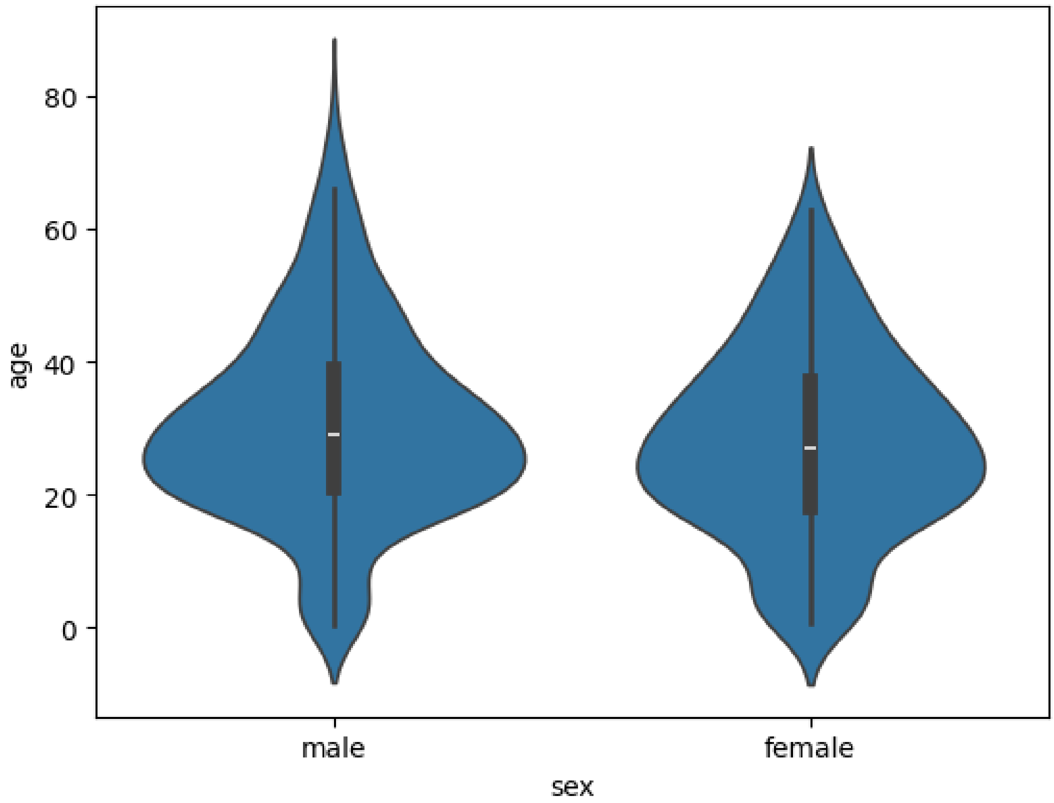
```
[16]: <Axes: xlabel='sex', ylabel='age'>
```



iv. Violin Plot: allows us to display all the components that actually correspond to the data point.

```
[17]: sns.violinplot (x = 'sex', y = 'age', data = df)
```

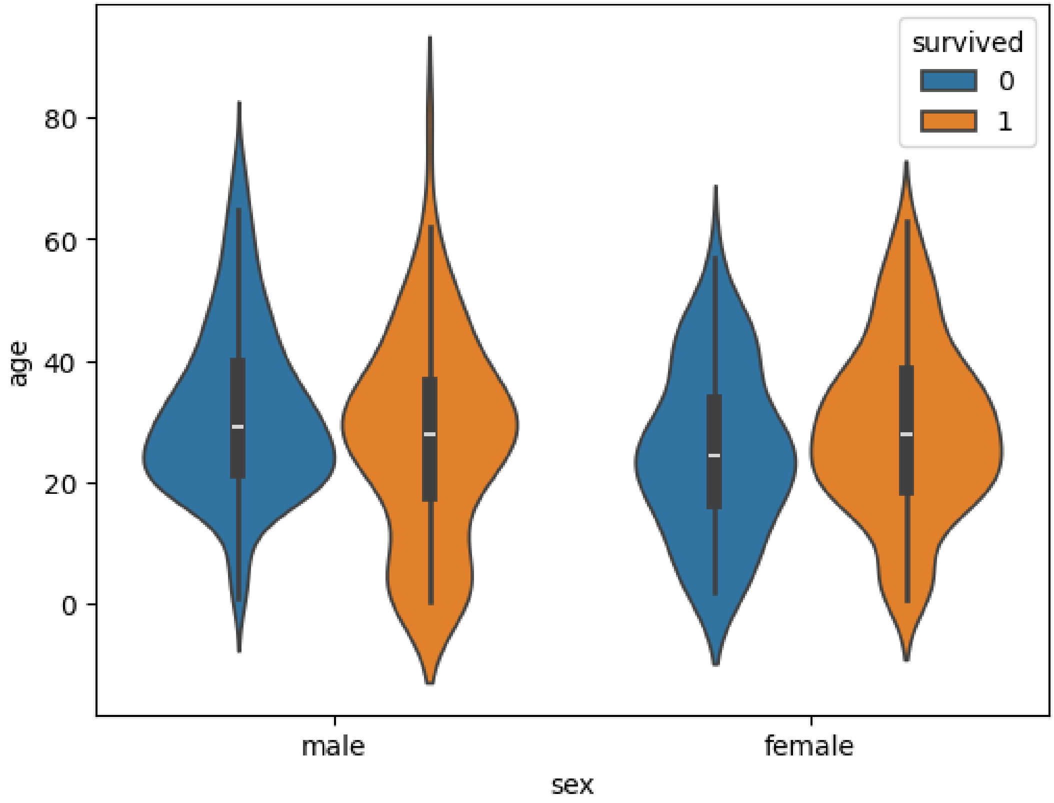
```
[17]: <Axes: xlabel='sex', ylabel='age'>
```



from the violin plot for males, it is clearly evident that the number of passengers with age between 20 and 40 is higher than all the rest of the age brackets.

```
[18]: sns.violinplot (x = 'sex', y = 'age', data = df, hue = 'survived')
```

```
[18]: <Axes: xlabel='sex', ylabel='age'>
```

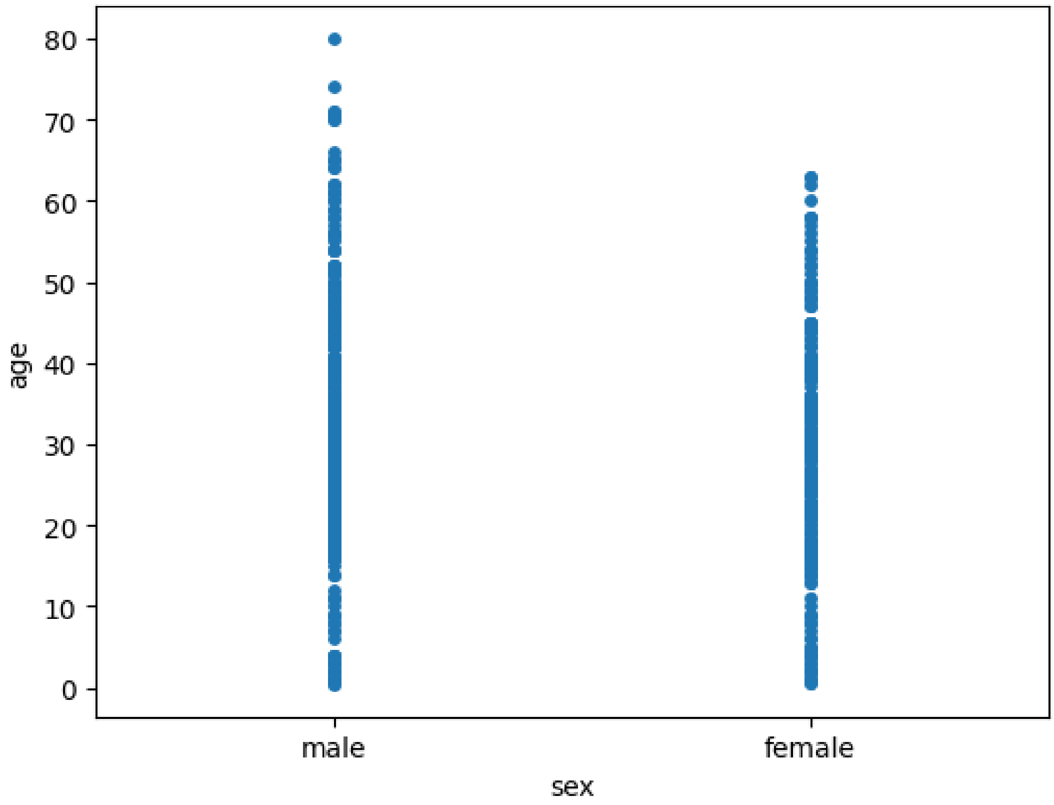


3. Advanced plots

i. Strip Plot: draws a scatter plot where one of the variables is categorical.

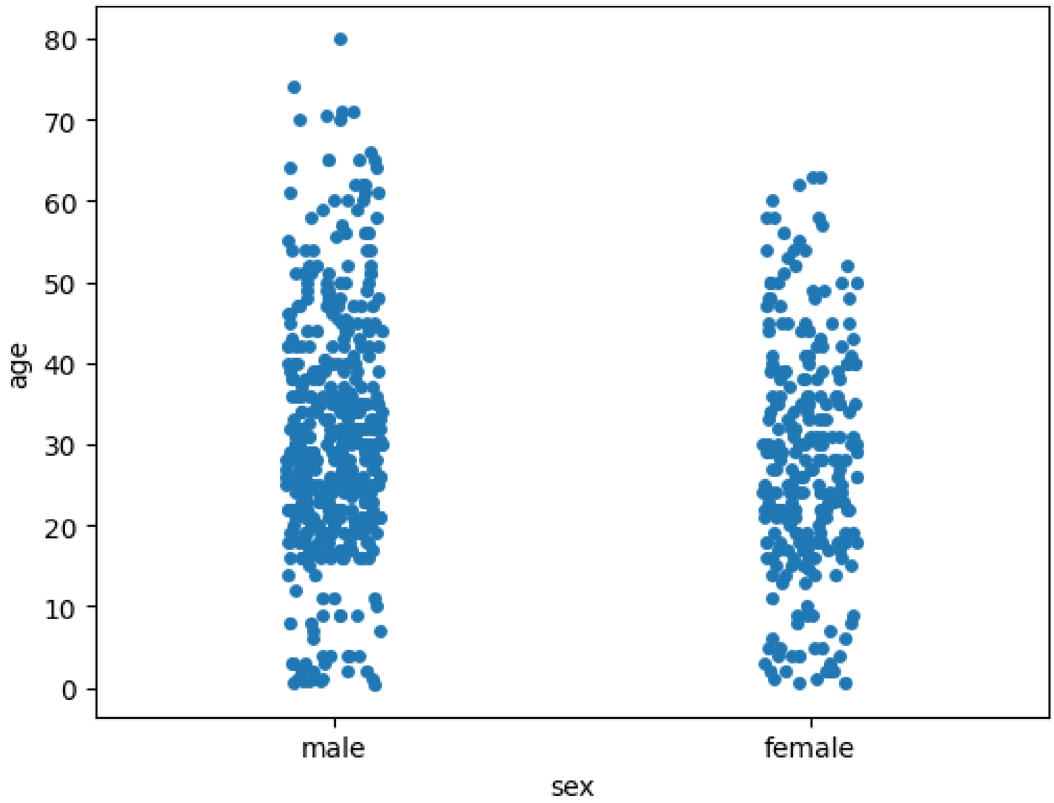
```
[19]: sns.stripplot(x = 'sex', y = 'age', data = df, jitter = False)
```

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



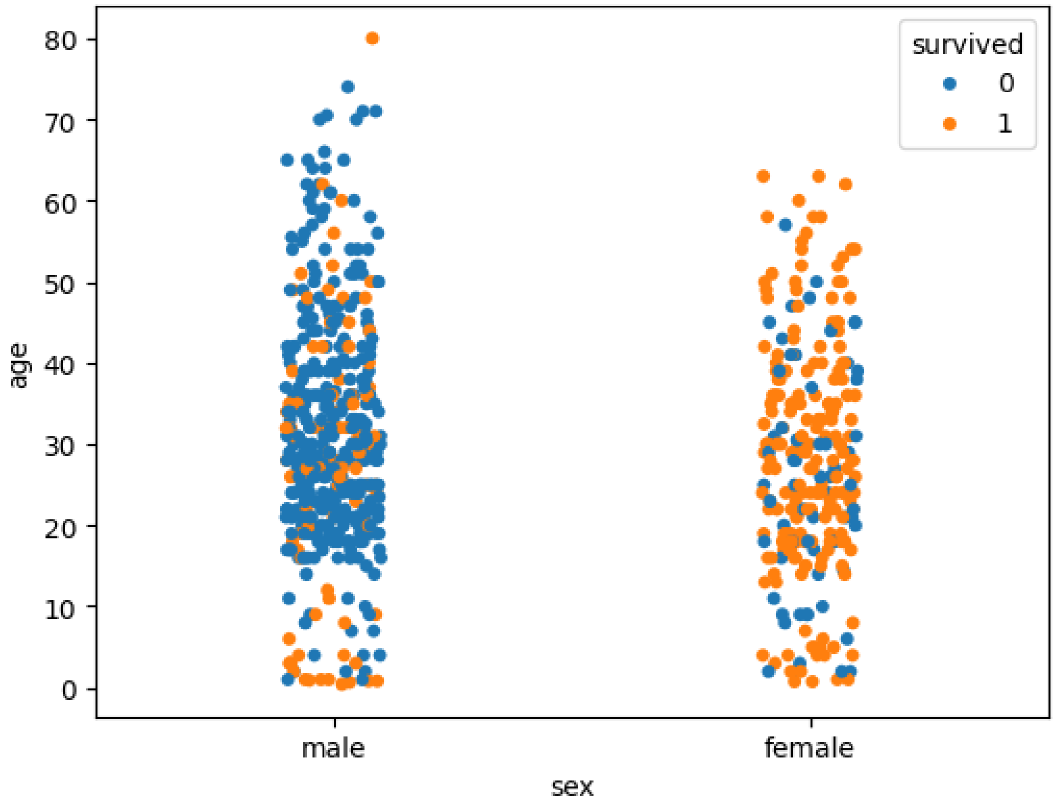
```
[20]: sns.stripplot(x = 'sex', y = 'age', data = df, jitter = True)
```

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



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

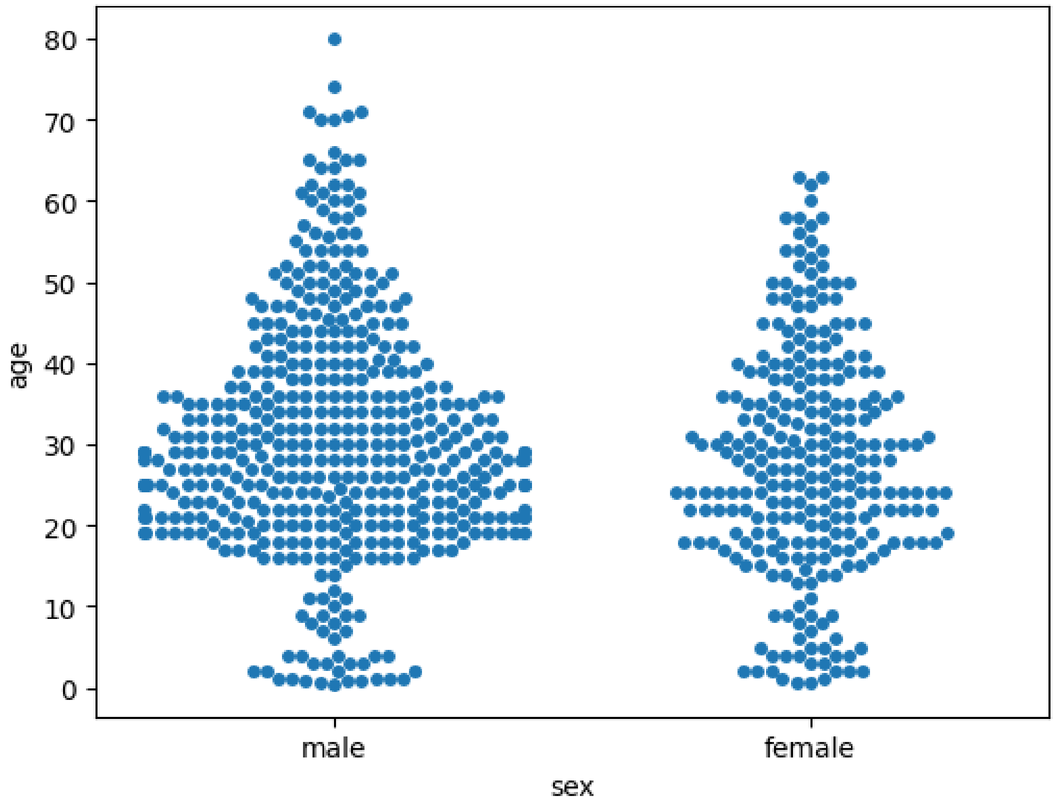
```
[21]: <Axes: xlabel='sex', ylabel='age'>
```



ii. Swarm Plot: is a combination of the strip and the violin plots. In the swarm plots, the points are adjusted in such a way that they don't overlap.

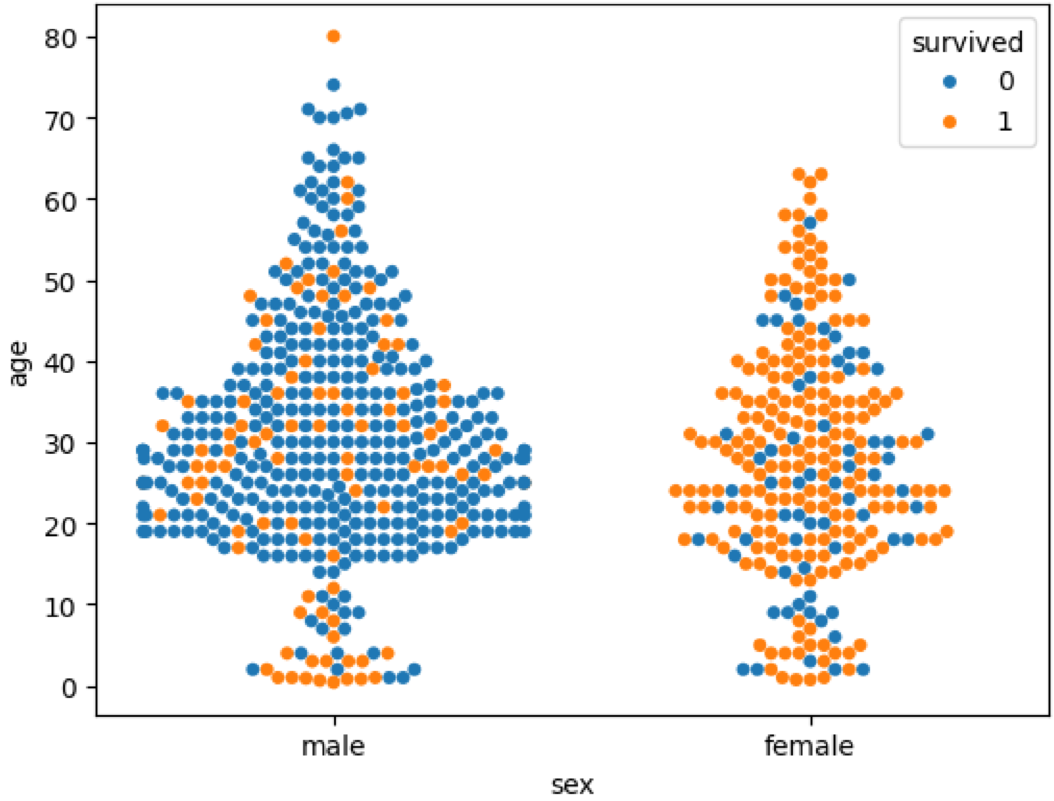
```
[22]: sns.swarmplot (x = 'sex', y = 'age', data = df)
```

```
[22]: <Axes: xlabel='sex', ylabel='age'>
```



```
[23]: sns.swarmplot (x = 'sex', y = 'age', data = df, hue = 'survived')
```

```
[23]: <Axes: xlabel='sex', ylabel='age'>
```



that the ratio of surviving males is less than the ratio of surviving females. Since for the male plot, there are more blue points and less orange points. On the other hand, for females, there are more orange points (surviving) than the blue points (not surviving). Another observation is that amongst males of age less than 10, more passengers survived as compared to those who didn't

4. Matrix plots: are the type of plots that show data in the form of rows and columns. Heat maps are the prime examples of matrix plots.

i. Heat Maps: used to plot correlation between numeric columns in the form of a matrix.

[24]:
```
df.corr()
```

```
<ipython-input-24-2f6f6606aa2c>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  df.corr()
```

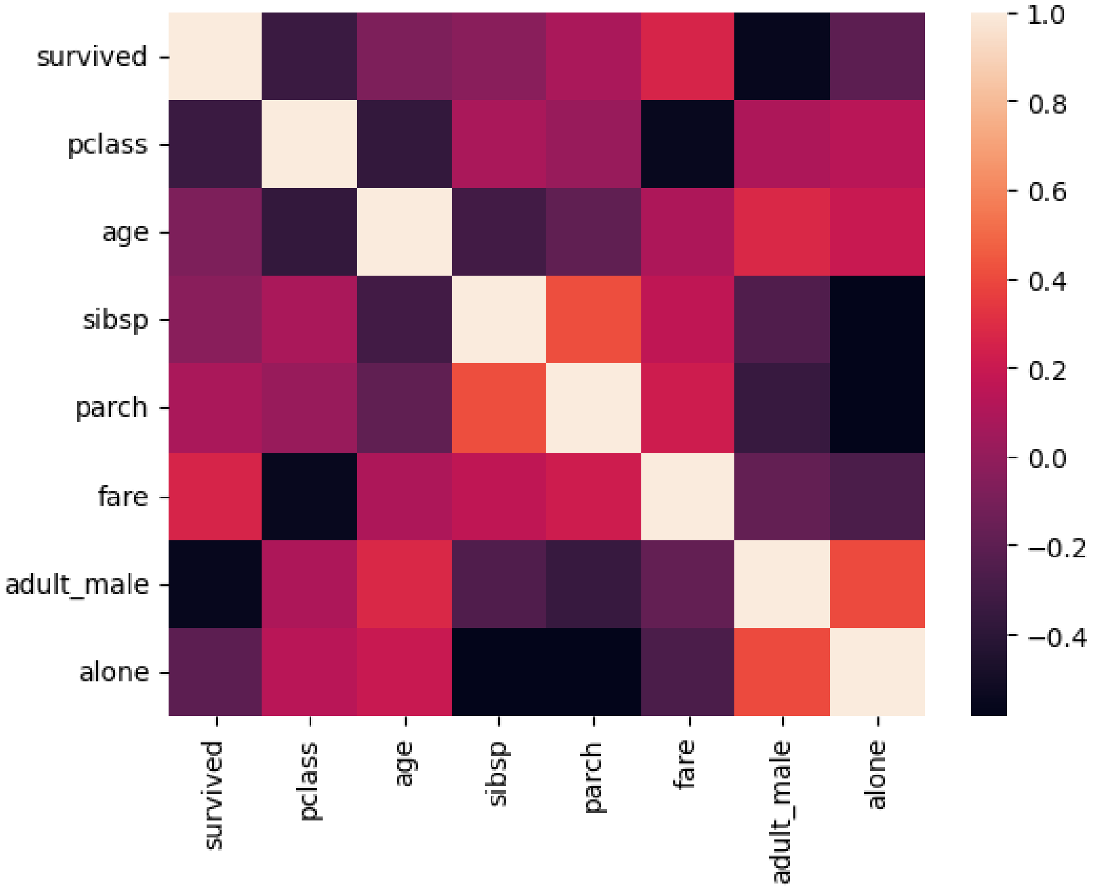
[24]:
```
            survived    pclass       age     sibsp     parch      fare  \
survived    1.000000 -0.338481 -0.077221 -0.035322  0.081629  0.257307
pclass     -0.338481  1.000000 -0.369226  0.083081  0.018443 -0.549500
age        -0.077221 -0.369226  1.000000 -0.308247 -0.189119  0.096067
```

```
sibsp      -0.035322  0.083081 -0.308247  1.000000  0.414838  0.159651
parch       0.081629  0.018443 -0.189119  0.414838  1.000000  0.216225
fare        0.257307 -0.549500  0.096067  0.159651  0.216225  1.000000
adult_male -0.557080  0.094035  0.280328 -0.253586 -0.349943 -0.182024
alone      -0.203367  0.135207  0.198270 -0.584471 -0.583398 -0.271832

            adult_male     alone
survived     -0.557080 -0.203367
pclass        0.094035  0.135207
age           0.280328  0.198270
sibsp        -0.253586 -0.584471
parch        -0.349943 -0.583398
fare         -0.182024 -0.271832
adult_male    1.000000  0.404744
alone         0.404744  1.000000
```

[25]:
```
corr = df.corr()
sns.heatmap(corr)
```

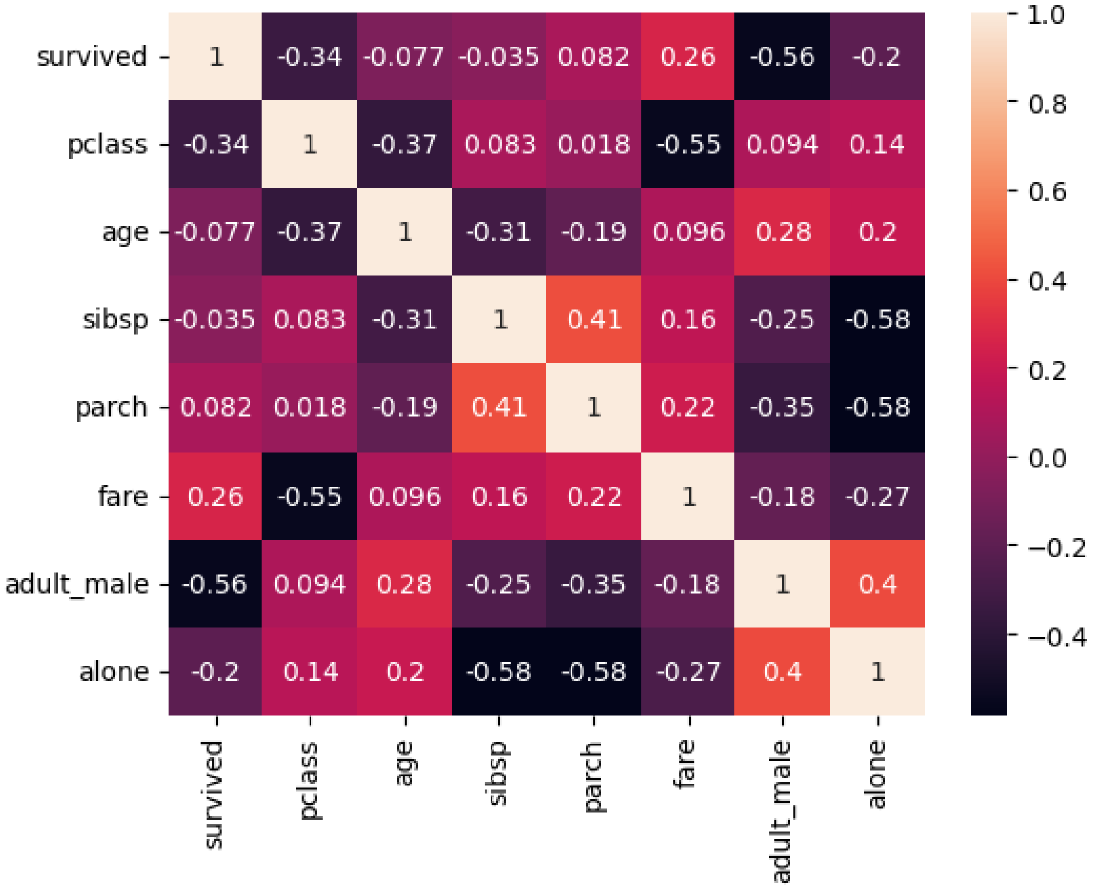
```
<ipython-input-25-753ca5bff919>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  corr = df.corr()
```

[25]:
```
<Axes: >
```



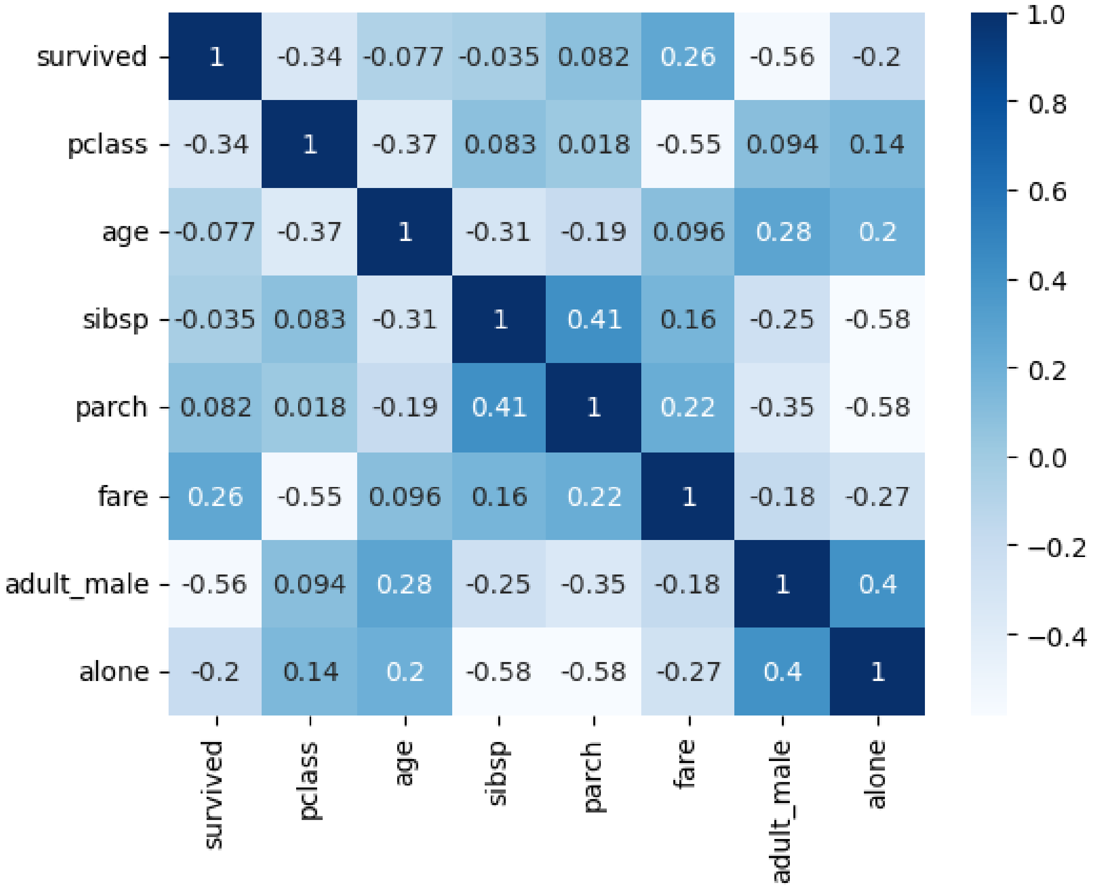
```
[26]: sns.heatmap(corr, annot = True)
```

```
[26]: <Axes: >
```



```
[27]: sns.heatmap(corr, cmap = 'Blues', annot = True)
```

```
[27]: <Axes: >
```



#**Data Visualization I** (with preprocessing of data)

I. Loading the Dataset, checking for null values and preprocessing data

[28]:
```
df1 = sns.load_dataset('titanic')
df1.head()
```

[28]:
```
   survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third
3         1       1  female  35.0      1      0  53.1000        S  First
4         0       3    male  35.0      0      0   8.0500        S  Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
```

```
4    man        True  NaN  Southampton    no   True
```

```
[29]: df1.shape
```

```
[29]: (891, 15)
```

```
[30]: df1.head()
```

```
[30]:    survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
      0         0       3    male  22.0      1      0   7.2500        S  Third
      1         1       1  female  38.0      1      0  71.2833        C  First
      2         1       3  female  26.0      0      0   7.9250        S  Third
      3         1       1  female  35.0      1      0  53.1000        S  First
      4         0       3    male  35.0      0      0   8.0500        S  Third

           who  adult_male deck  embark_town alive  alone
      0    man        True  NaN  Southampton    no  False
      1  woman       False    C    Cherbourg   yes  False
      2  woman       False  NaN  Southampton   yes   True
      3  woman       False    C  Southampton   yes  False
      4    man        True  NaN  Southampton    no   True
```

```
[31]: df1.describe()
```

```
[31]:          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
```

```
[32]: df1.describe(include = 'object')
```

```
[32]:          sex embarked  who  embark_town alive
      count    891      889  891          889   891
      unique     2        3    3            3     2
      top     male        S  man  Southampton    no
      freq     577      644  537          644   549
```

```
[33]: df1.isnull().sum()
```

```
[33]: 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
```

II. Filling up the NULL values in the dataset.

```
[34]: df1['age'] = df1['age'].fillna(np.mean(df1['age']))
```

```
[35]: df1['deck'] = df1['deck'].fillna(df1['deck'].mode()[0])
```

```
[36]: df1['embark_town'] = df1['embark_town'].fillna(df1['embark_town'].mode()[0])
```

```
[37]: df1['embarked'] = df1['embarked'].fillna(df1['embarked'].mode()[0])
```

```
[38]: df1.isnull().sum()
```

```
[38]: survived       0
      pclass         0
      sex            0
      age            0
      sibsp          0
      parch          0
      fare           0
      embarked       0
      class          0
      who            0
      adult_male     0
      deck           0
      embark_town    0
      alive          0
      alone          0
      dtype: int64
```

```
[39]: df1.head(n = 10)
```

```
[39]:    survived  pclass     sex        age  sibsp  parch     fare embarked  \
      0         0       3    male  22.000000      1      0   7.2500        S
```

```
1         1       1  female  38.000000      1      0  71.2833        C
2         1       3  female  26.000000      0      0   7.9250        S
3         1       1  female  35.000000      1      0  53.1000        S
4         0       3    male  35.000000      0      0   8.0500        S
5         0       3    male  29.699118      0      0   8.4583        Q
6         0       1    male  54.000000      0      0  51.8625        S
7         0       3    male   2.000000      3      1  21.0750        S
8         1       3  female  27.000000      0      2  11.1333        S
9         1       2  female  14.000000      1      0  30.0708        C

    class    who  adult_male deck  embark_town alive  alone
0   Third    man        True    C  Southampton    no  False
1   First  woman       False    C    Cherbourg   yes  False
2   Third  woman       False    C  Southampton   yes   True
3   First  woman       False    C  Southampton   yes  False
4   Third    man        True    C  Southampton    no   True
5   Third    man        True    C   Queenstown    no   True
6   First    man        True    E  Southampton    no   True
7   Third  child       False    C  Southampton    no  False
8   Third  woman       False    C  Southampton   yes  False
9  Second  child       False    C    Cherbourg   yes  False
```

III. Finding patterns of data

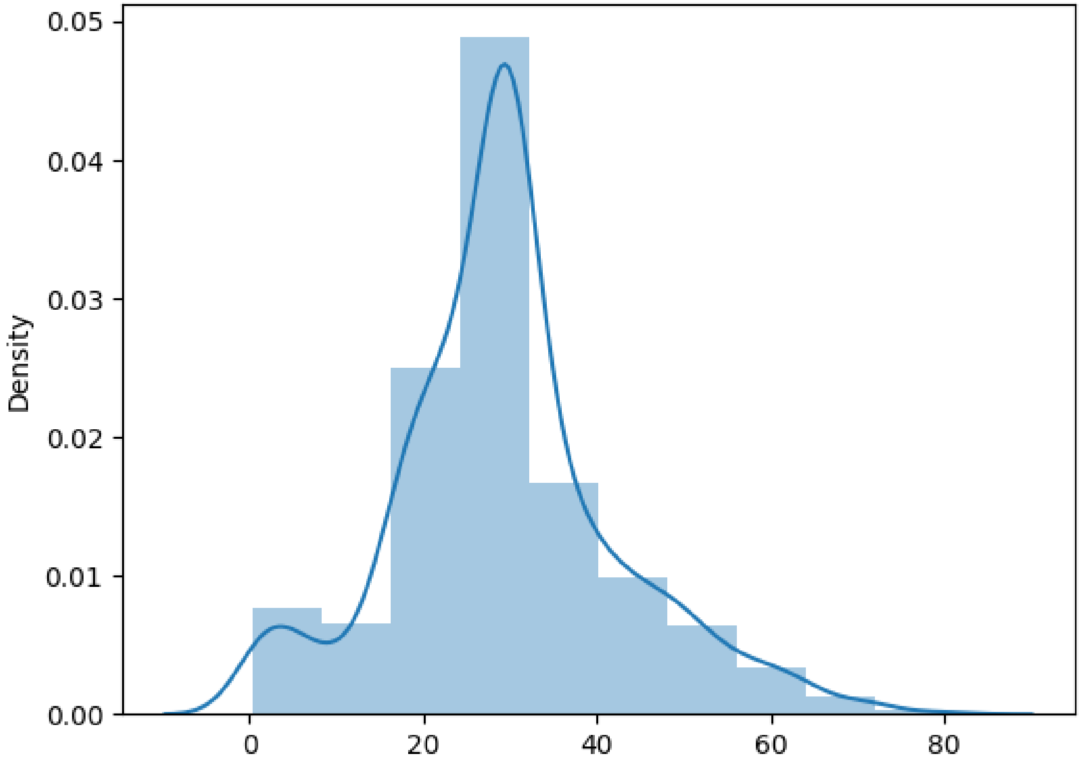
1. Distribution plots

a. Dist plot

[40]:
```
sns.distplot(x = df1['age'], bins = 10)
```

```
<ipython-input-40-2bc27e173dad>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

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

  sns.distplot(x = df1['age'], bins = 10)
```

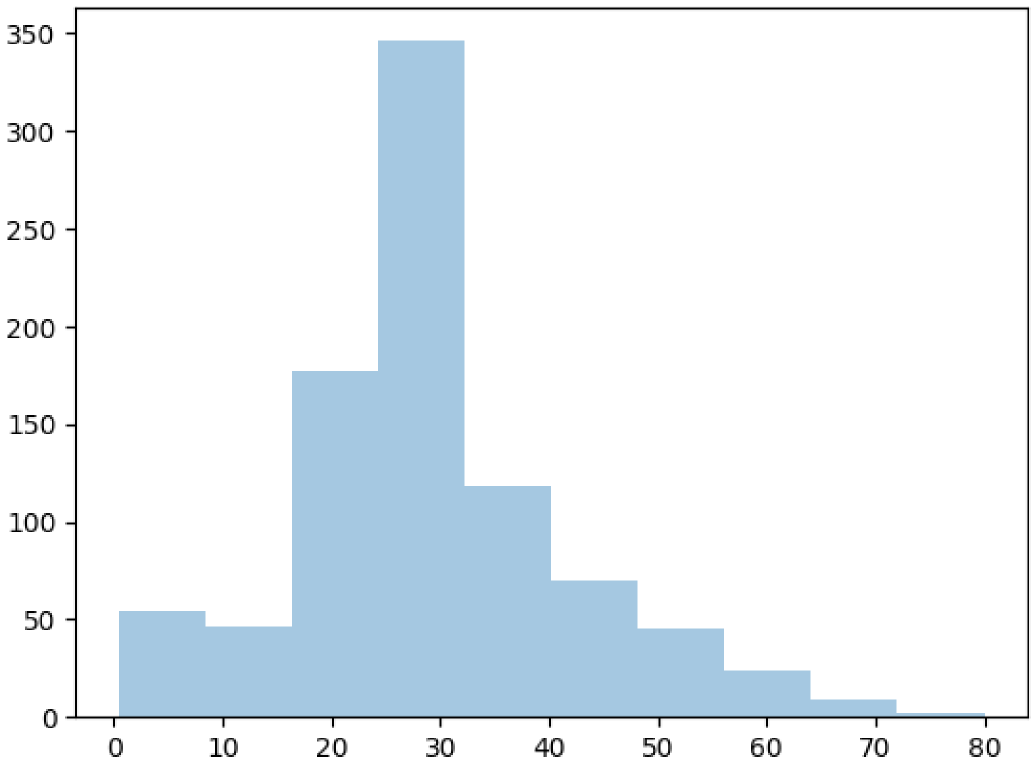
[40]:
```
<Axes: ylabel='Density'>
```



```
[41]: sns.distplot(x = df1['age'], bins = 10, kde = False)
```

```
<ipython-input-41-88bc91aaa657>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

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

  sns.distplot(x = df1['age'], bins = 10, kde = False)
```

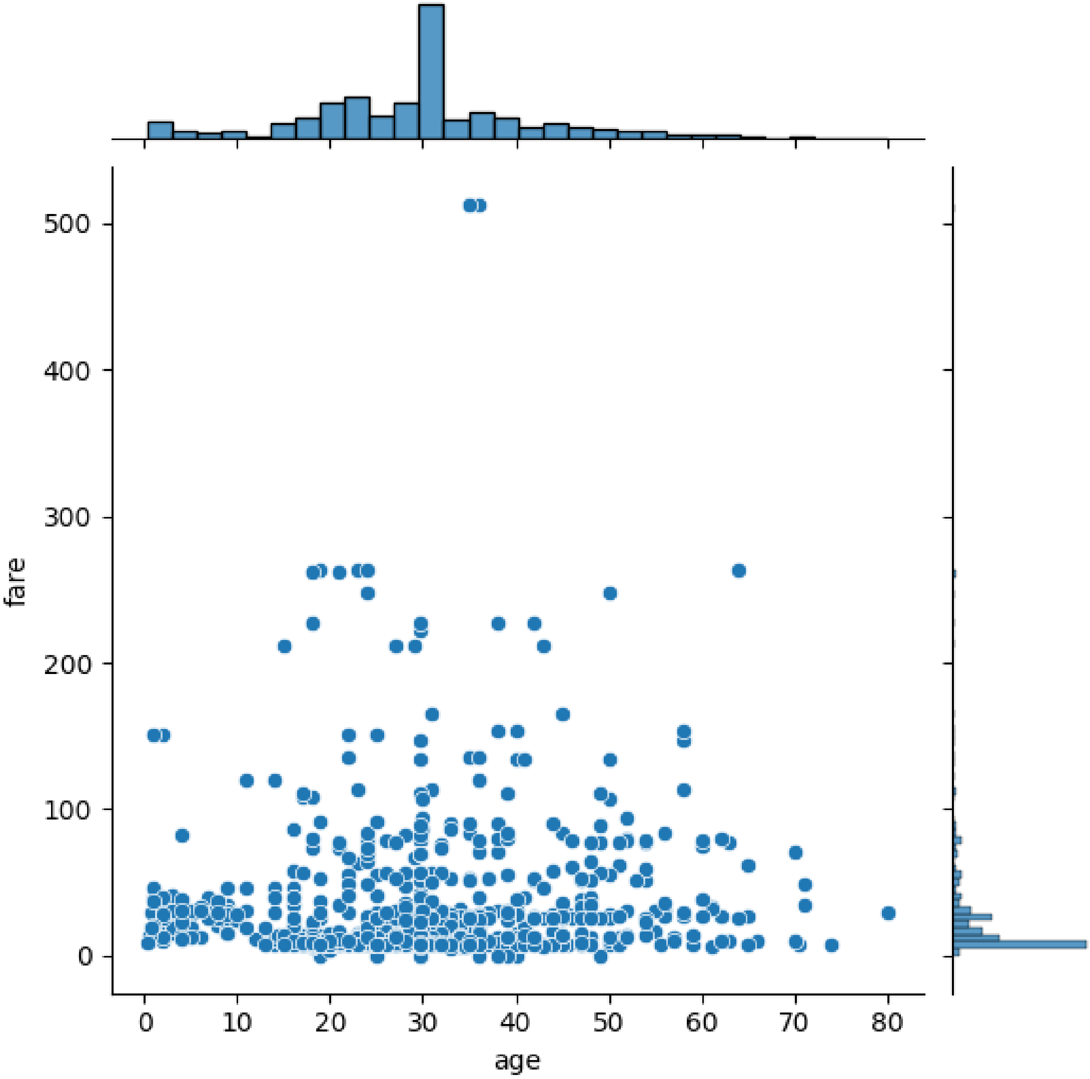
```
[41]: <Axes: >
```



b. Joint plot

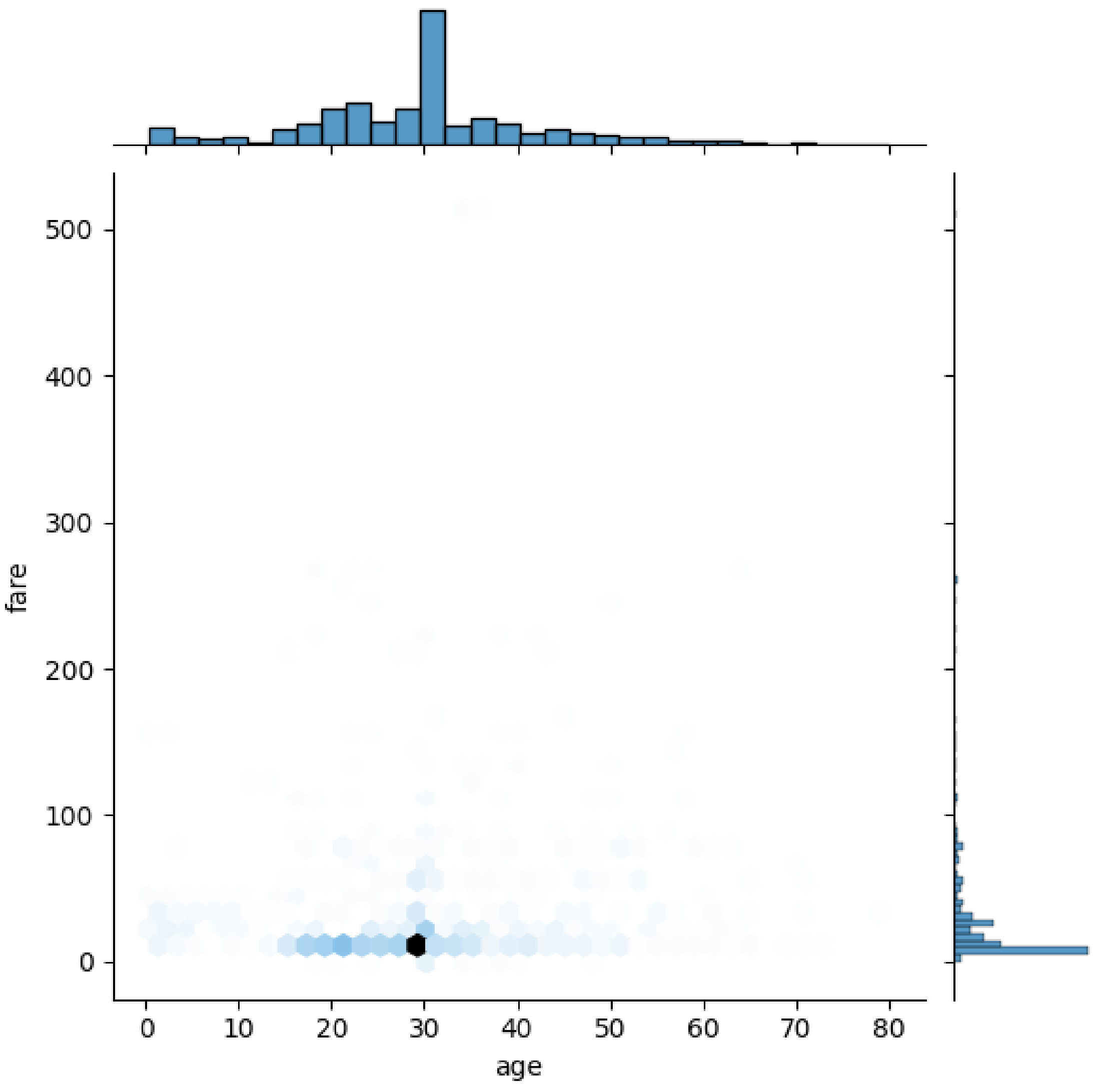
```
[42]: sns.jointplot(x = df1['age'], y = df1['fare'], kind = 'scatter')
```

```
[42]: <seaborn.axisgrid.JointGrid at 0x7e09b0e7cd60>
```



[43]:
```
sns.jointplot(x = df1['age'], y = df1['fare'], kind = 'hex')
```

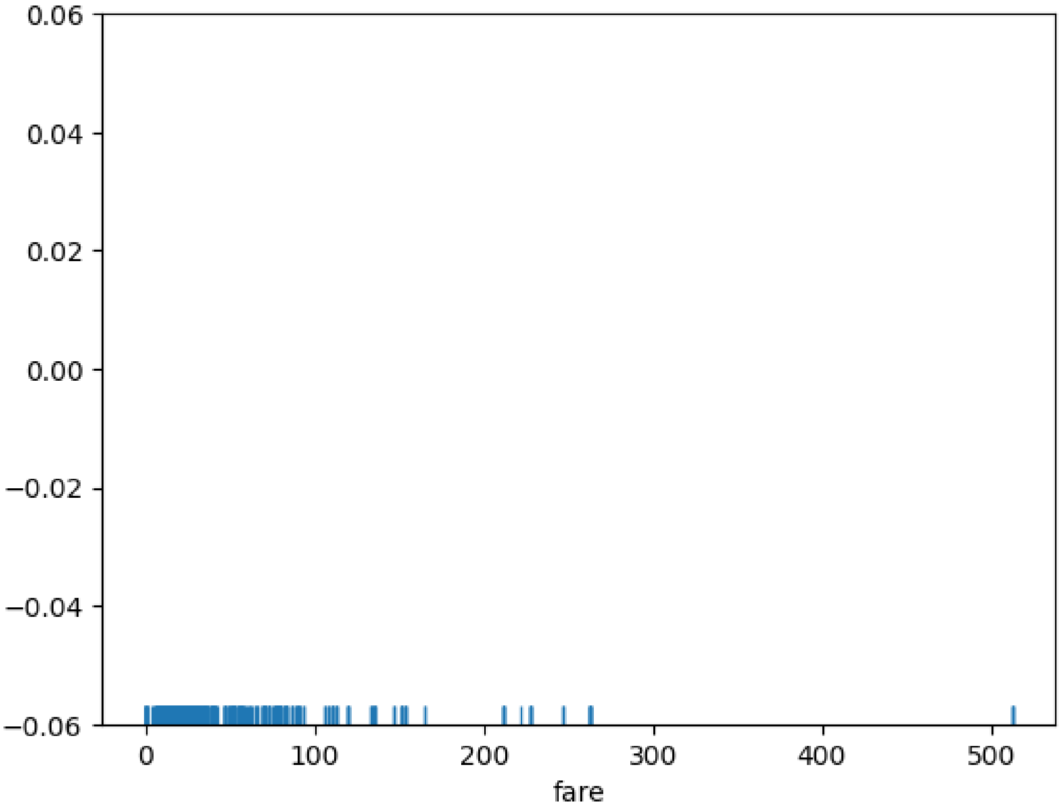
[43]:
```
<seaborn.axisgrid.JointGrid at 0x7e09b0e7e320>
```



c. Rug plot

```
[44]: sns.rugplot(df1['fare'])
```

```
[44]: <Axes: xlabel='fare'>
```

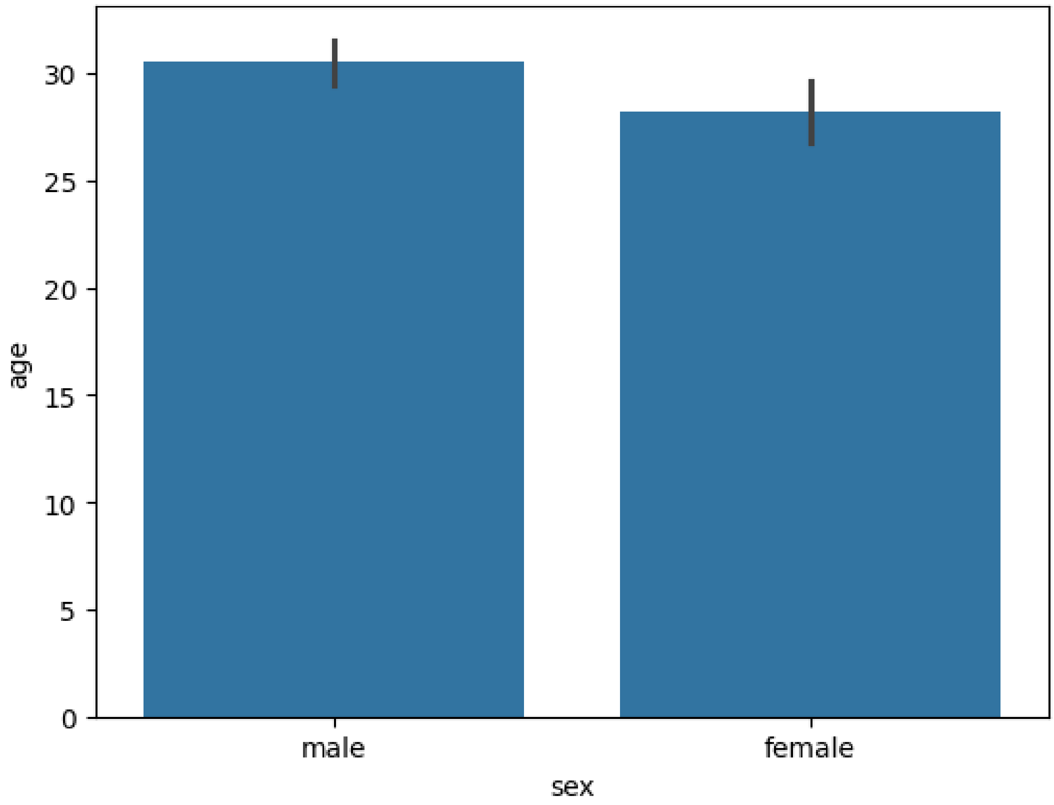


2. Categorical Plots

a. Bar plot

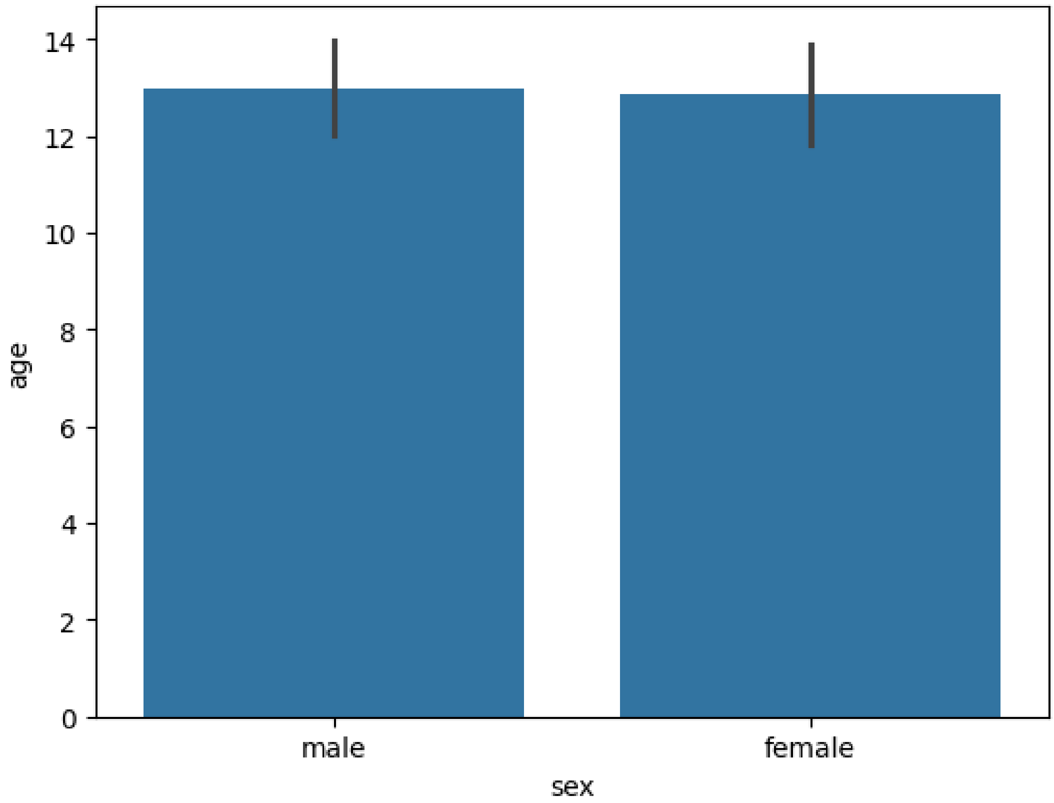
[45]:
```
sns.barplot (x = 'sex', y = 'age', data = df1)
```

[45]:
```
<Axes: xlabel='sex', ylabel='age'>
```



```
[46]: sns.barplot (x = 'sex', y = 'age', data = df1, estimator = np.std)
```

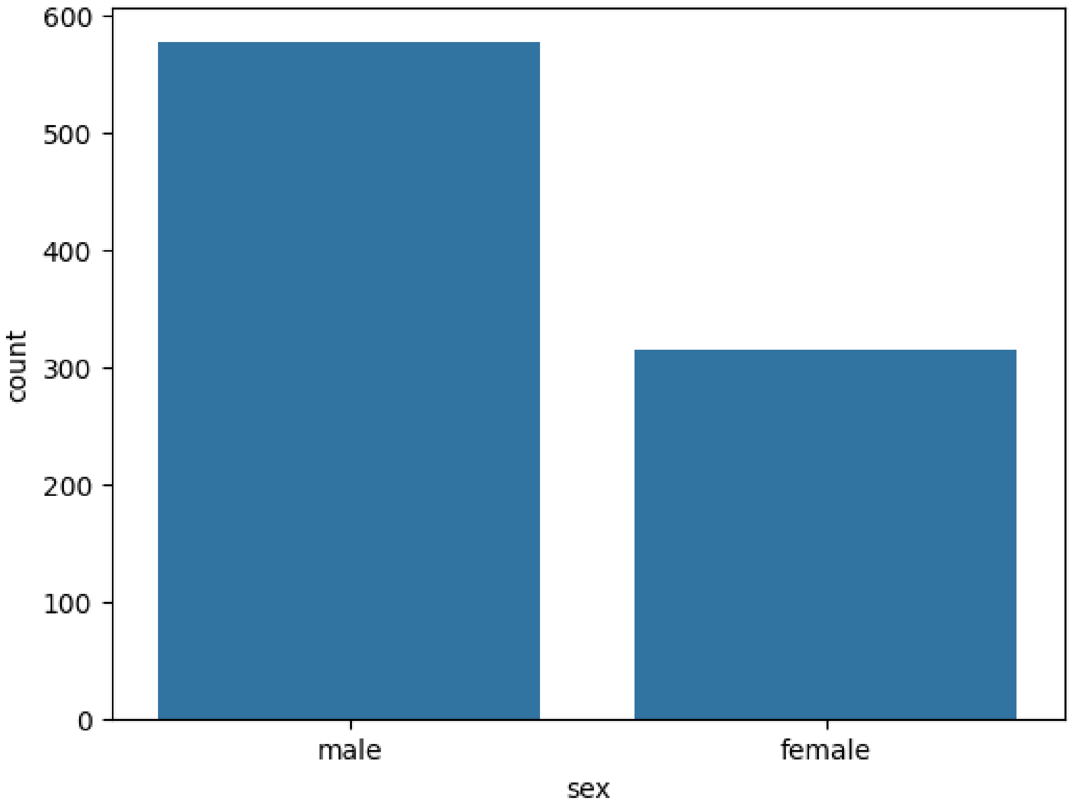
```
[46]: <Axes: xlabel='sex', ylabel='age'>
```



b. Count plot

```
[47]: sns.countplot(x = 'sex', data = df1)
```

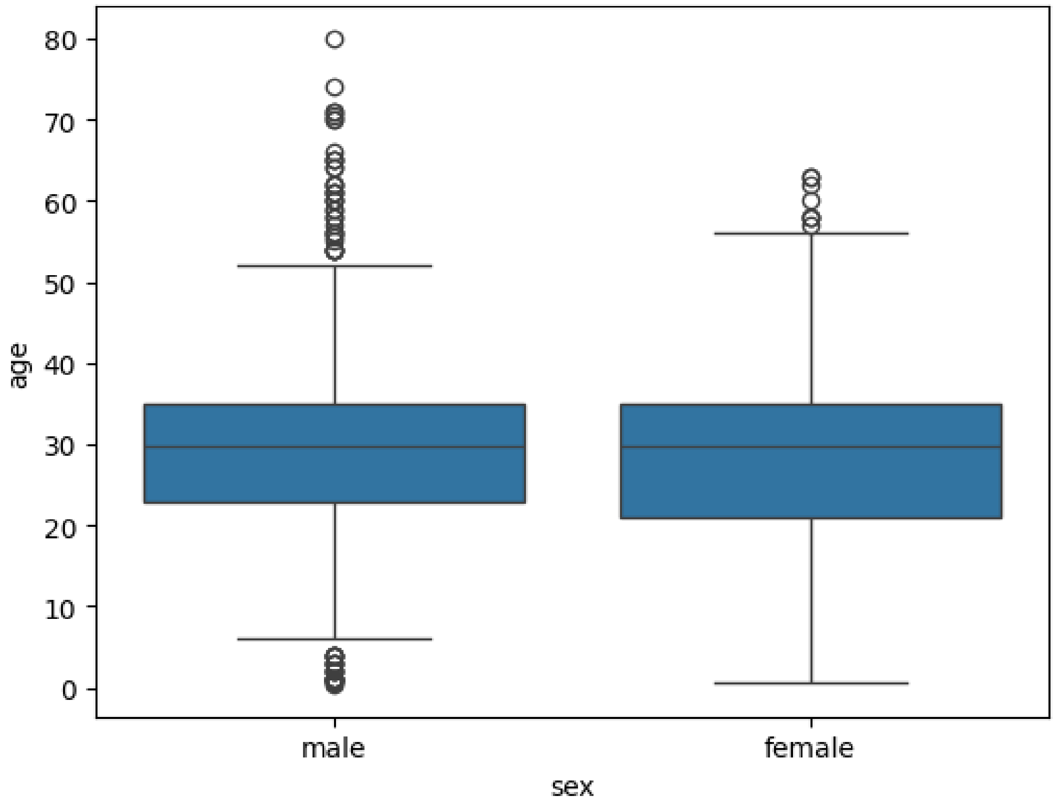
```
[47]: <Axes: xlabel='sex', ylabel='count'>
```



c. Box plot

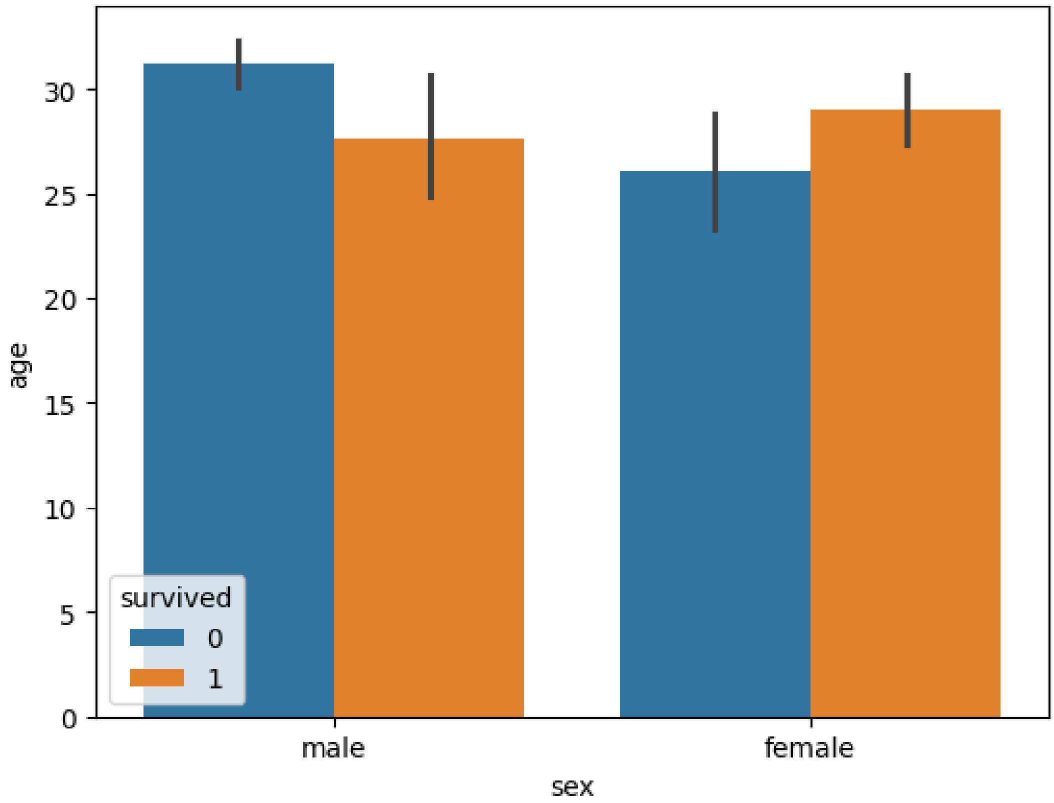
```
[48]: sns.boxplot (x = 'sex', y = 'age', data = df1)
```

```
[48]: <Axes: xlabel='sex', ylabel='age'>
```



```
[49]: sns.barplot (x = 'sex', y = 'age', data = df1, hue = 'survived')
```

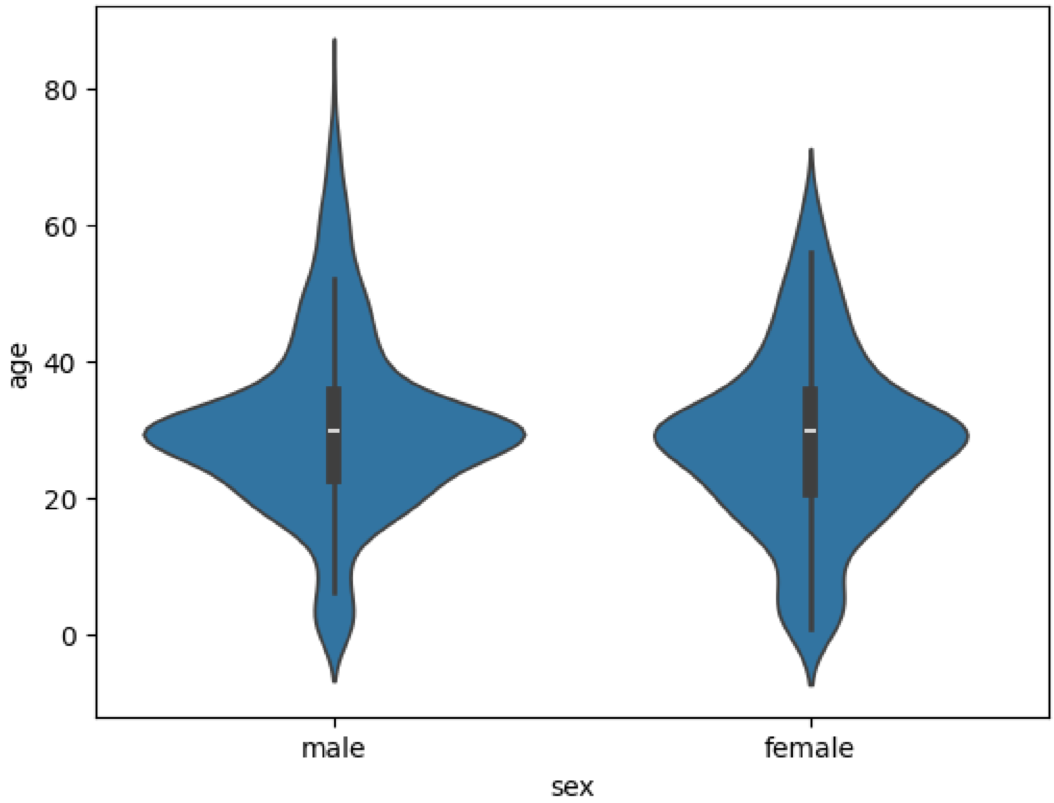
```
[49]: <Axes: xlabel='sex', ylabel='age'>
```



d. Violin plot

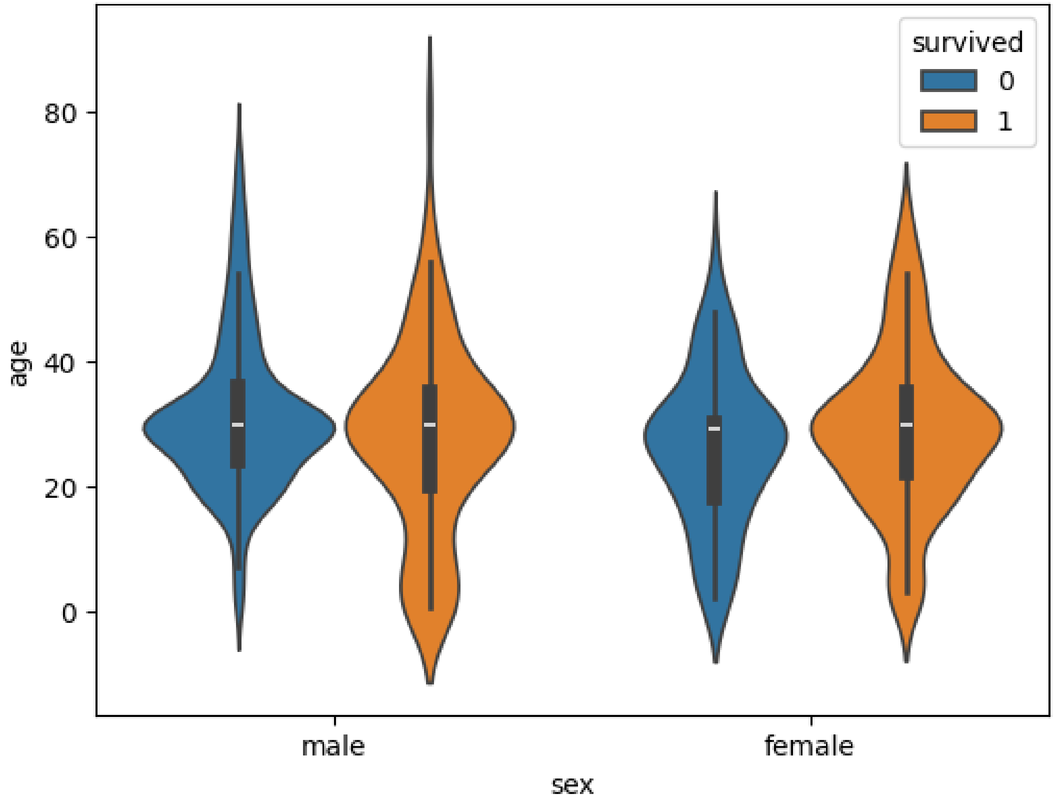
```
[50]: sns.violinplot (x = 'sex', y = 'age', data = df1)
```

```
[50]: <Axes: xlabel='sex', ylabel='age'>
```



```
[51]: sns.violinplot (x = 'sex', y = 'age', data = df1, hue = 'survived')
```

```
[51]: <Axes: xlabel='sex', ylabel='age'>
```

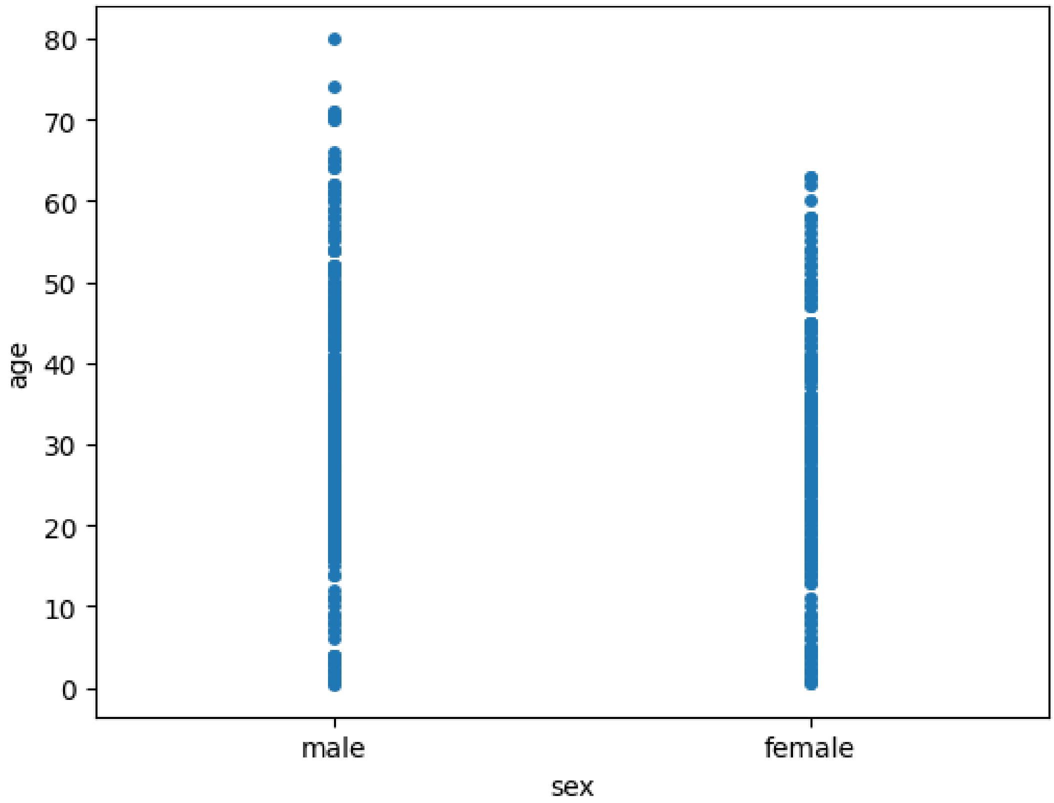


3. Advanced Plots

a. Strip plot

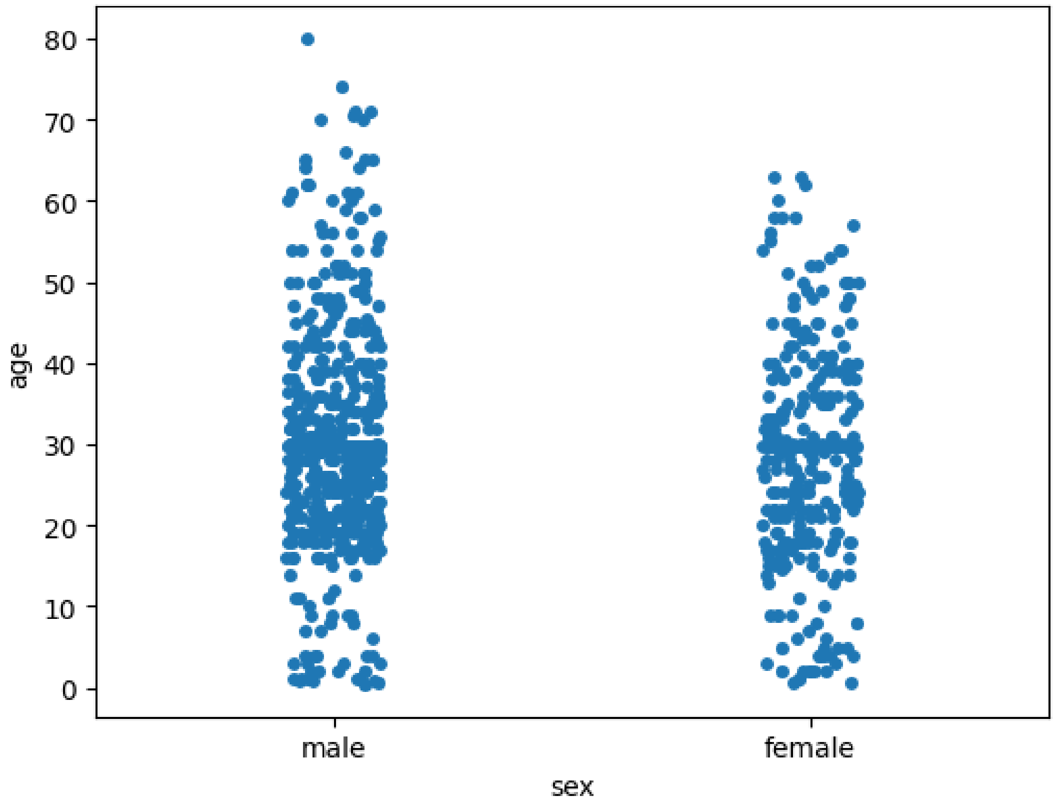
```
[52]: sns.stripplot (x = 'sex', y = 'age', data = df1, jitter = False)
```

```
[52]: <Axes: xlabel='sex', ylabel='age'>
```



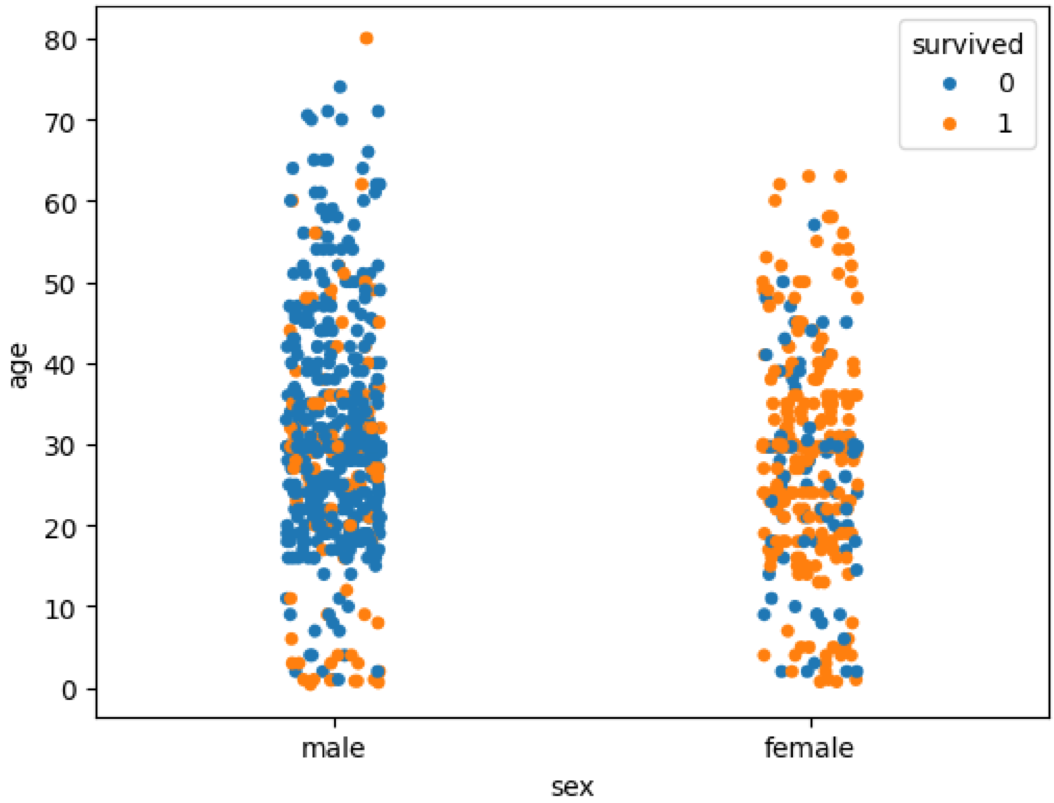
```
[53]: sns.stripplot (x = 'sex', y = 'age', data = df1, jitter = True)
```

```
[53]: <Axes: xlabel='sex', ylabel='age'>
```



```
[54]: sns.stripplot (x = 'sex', y = 'age', data = df1, jitter = True, hue =␣
        ↪'survived')
```

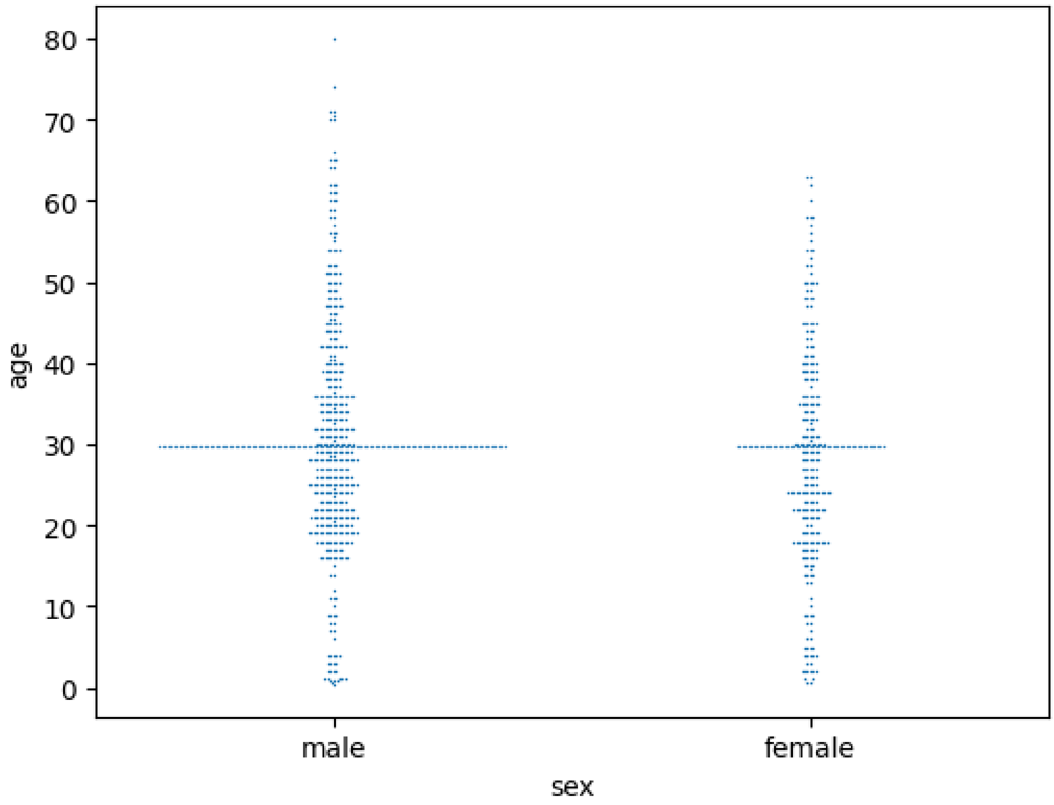
```
[54]: <Axes: xlabel='sex', ylabel='age'>
```



b. Swarm plot

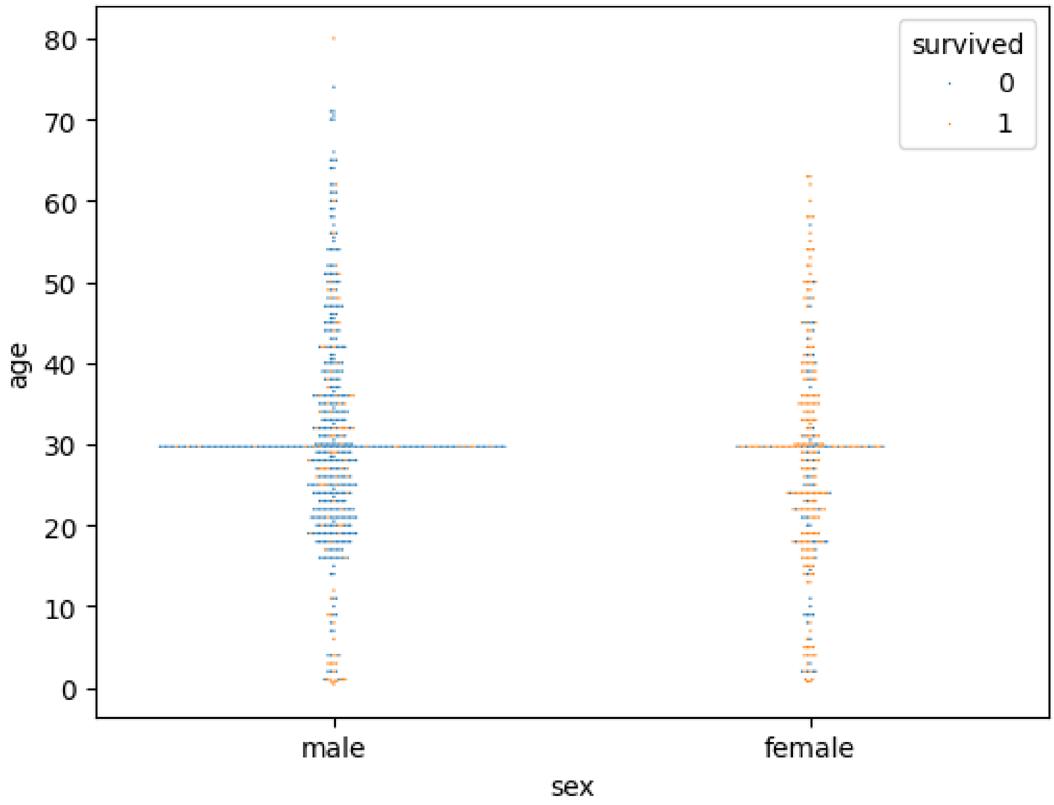
```
[55]: sns.swarmplot (x = 'sex', y = 'age', data = df1, size = 1)
```

```
[55]: <Axes: xlabel='sex', ylabel='age'>
```



```
[56]: sns.swarmplot (x = 'sex', y = 'age', data = df1, size = 1, hue = 'survived')
```

```
[56]: <Axes: xlabel='sex', ylabel='age'>
```



4. Matrix plots

a. Heat maps

```
[57]: df1.corr()
```

```
<ipython-input-57-49b3fcfeb4d1>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  df1.corr()
```

```
[57]:             survived    pclass       age     sibsp     parch      fare  \
      survived    1.000000 -0.338481 -0.069809 -0.035322  0.081629  0.257307
      pclass     -0.338481  1.000000 -0.331339  0.083081  0.018443 -0.549500
      age        -0.069809 -0.331339  1.000000 -0.232625 -0.179191  0.091566
      sibsp      -0.035322  0.083081 -0.232625  1.000000  0.414838  0.159651
      parch       0.081629  0.018443 -0.179191  0.414838  1.000000  0.216225
      fare        0.257307 -0.549500  0.091566  0.159651  0.216225  1.000000
      adult_male -0.557080  0.094035  0.253236 -0.253586 -0.349943 -0.182024
      alone      -0.203367  0.135207  0.179775 -0.584471 -0.583398 -0.271832
```

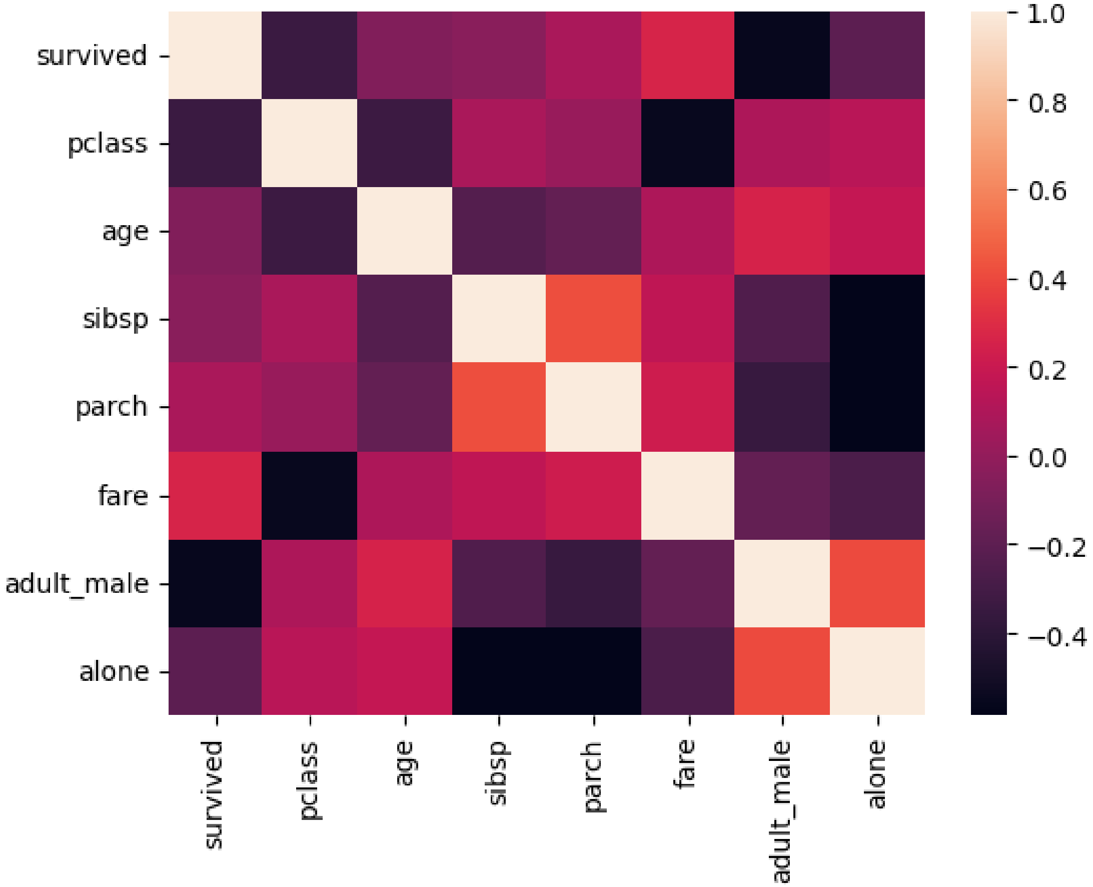
```
            adult_male     alone
survived     -0.557080 -0.203367
pclass        0.094035  0.135207
age           0.253236  0.179775
sibsp        -0.253586 -0.584471
parch        -0.349943 -0.583398
fare         -0.182024 -0.271832
adult_male    1.000000  0.404744
alone         0.404744  1.000000
```

[58]:

```
corr = df1.corr()
sns.heatmap(corr)
```

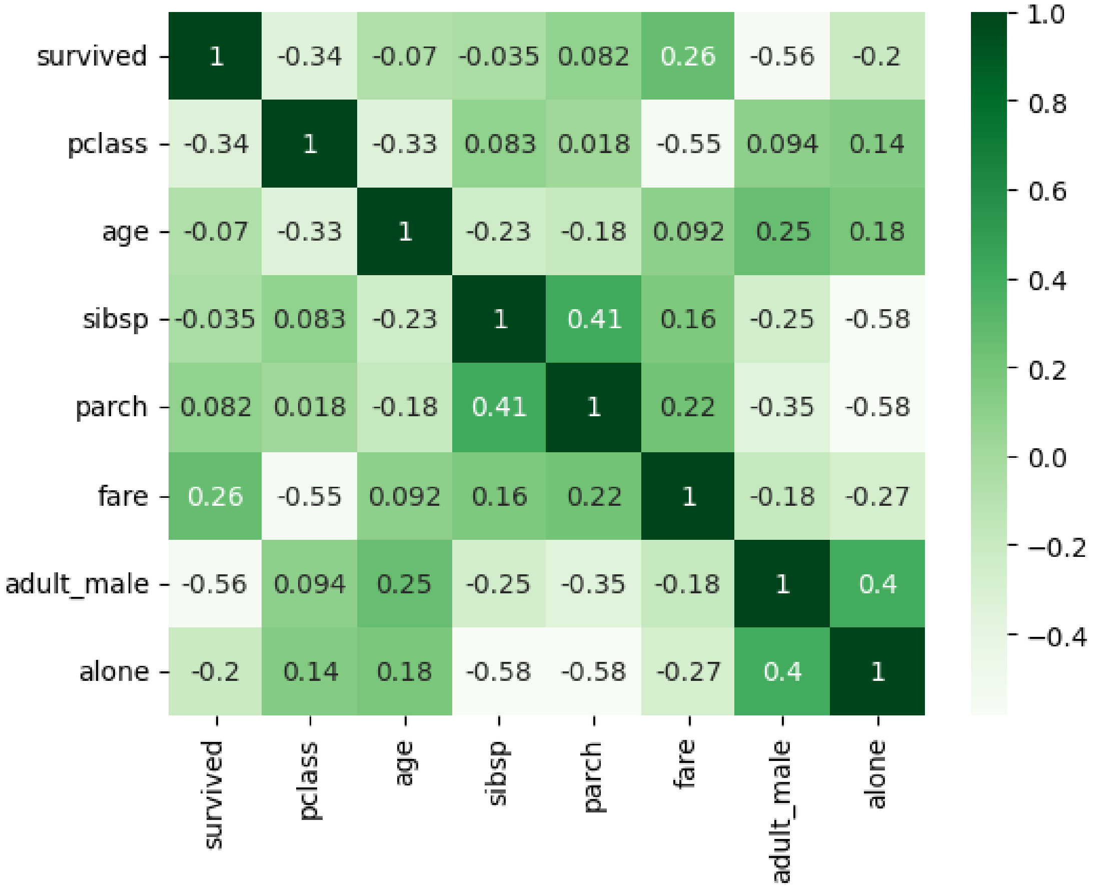
```
<ipython-input-58-7884c29f6e71>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  corr = df1.corr()
```

[58]: <Axes: >



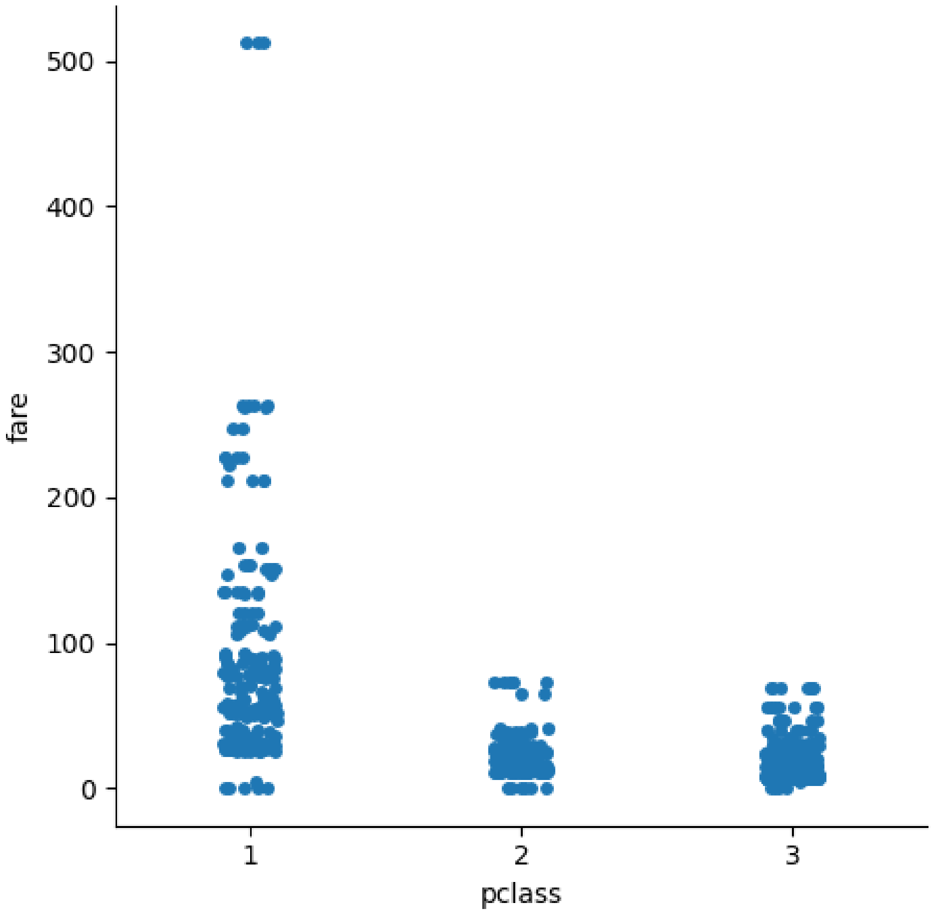
```
[59]: sns.heatmap(corr, annot = True, cmap = 'Greens')
```

```
[59]: <Axes: >
```



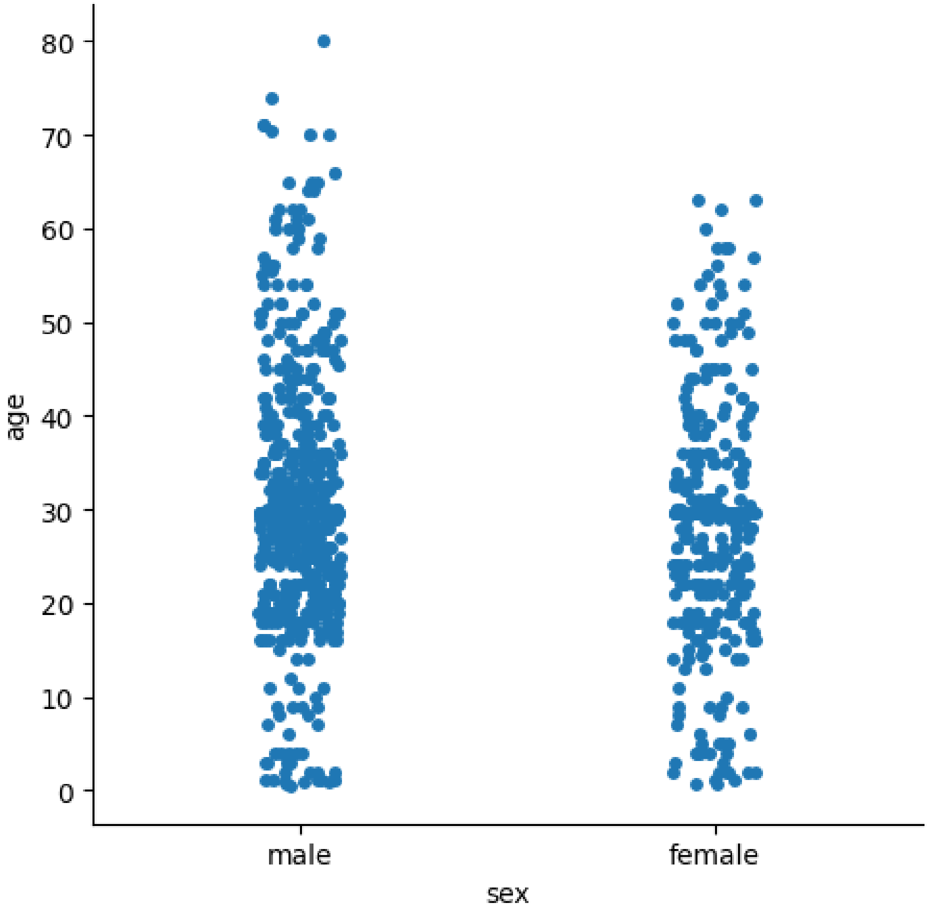
```
[63]: sns.catplot(x= 'pclass', y = 'fare', data=df, kind = 'strip')
```

```
[63]: <seaborn.axisgrid.FacetGrid at 0x7e09abd659c0>
```



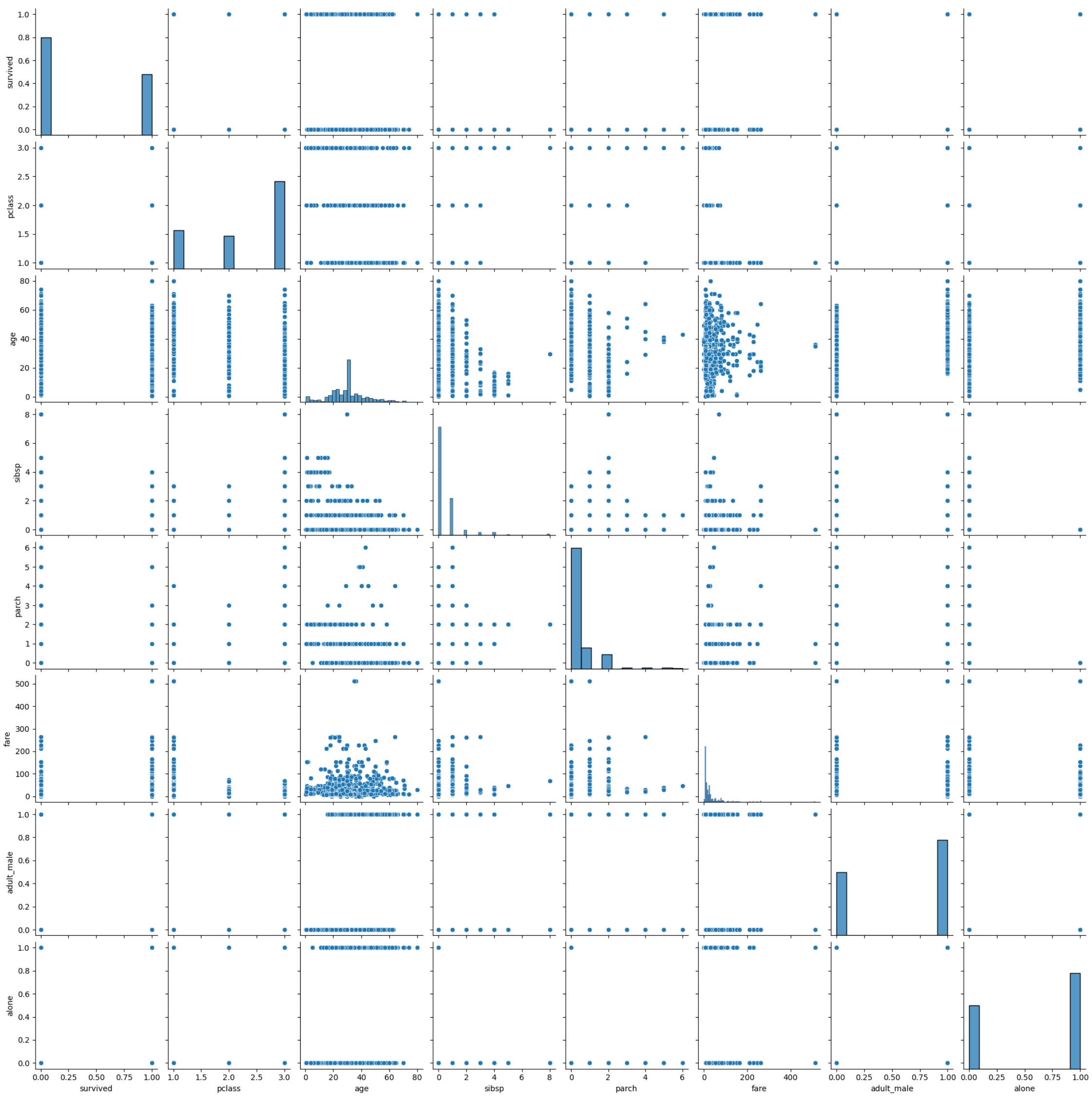
```
[65]: sns.catplot(x= 'sex', y = 'age', data=df1, kind = 'strip')
```

```
[65]: <seaborn.axisgrid.FacetGrid at 0x7e09b0eeea40>
```



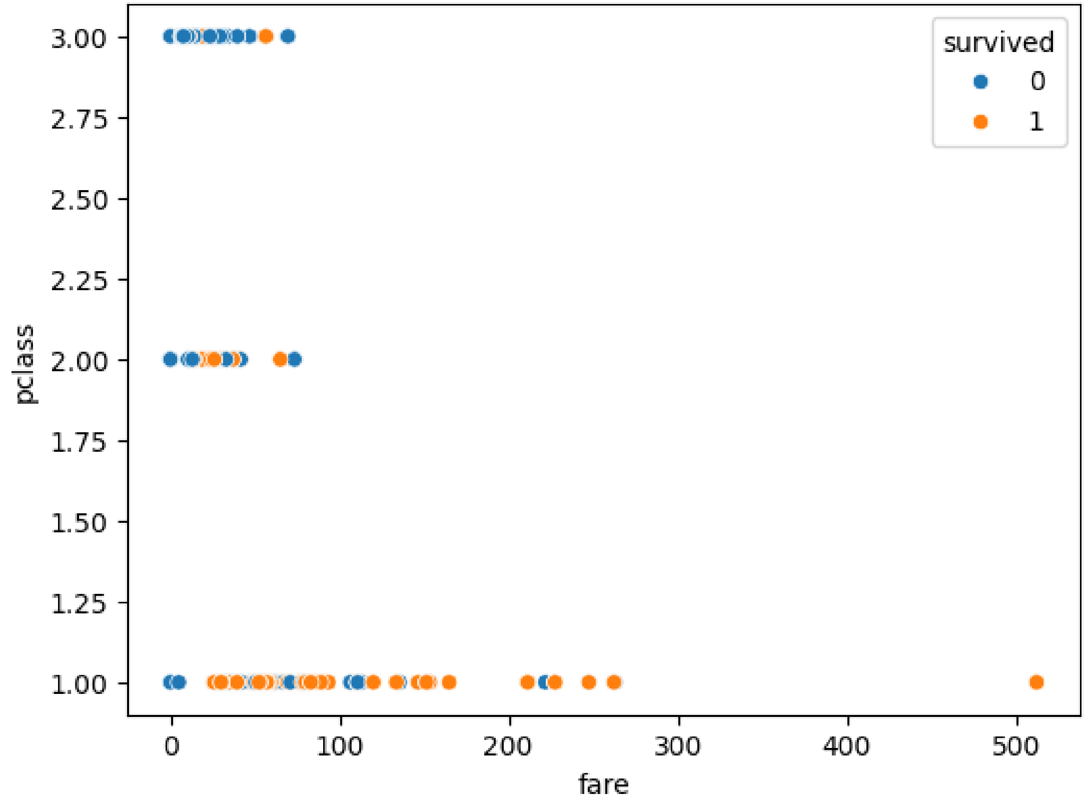
[66]:
```
sns.pairplot(df1)
```

[66]:
```
<seaborn.axisgrid.PairGrid at 0x7e09aa60f700>
```



```
[67]: sns.scatterplot(x = 'fare', y = 'pclass', hue = 'survived', data = df1)
```

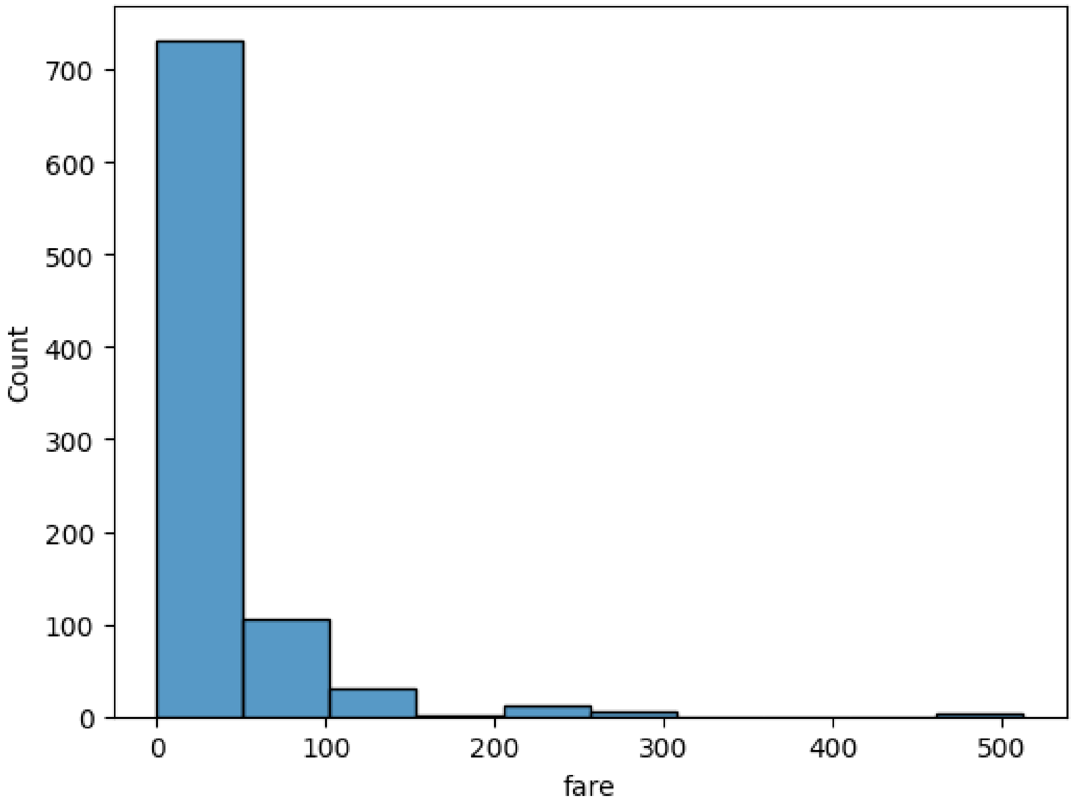
```
[67]: <Axes: xlabel='fare', ylabel='pclass'>
```



2. Write a code to check how the price of the ticket (column name: 'fare') for each passenger is distributed by plotting a histogram.

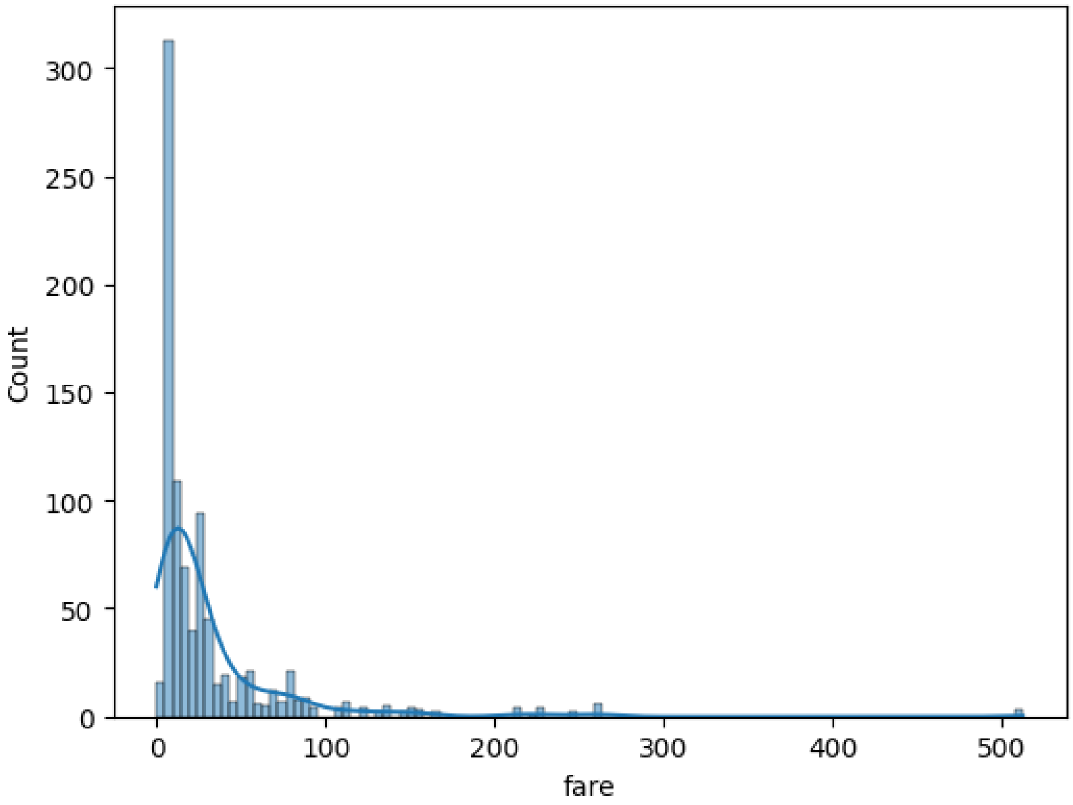
[68]:
```
sns.histplot(df['fare'], kde=False, bins=10)
```

[68]:
```
<Axes: xlabel='fare', ylabel='Count'>
```



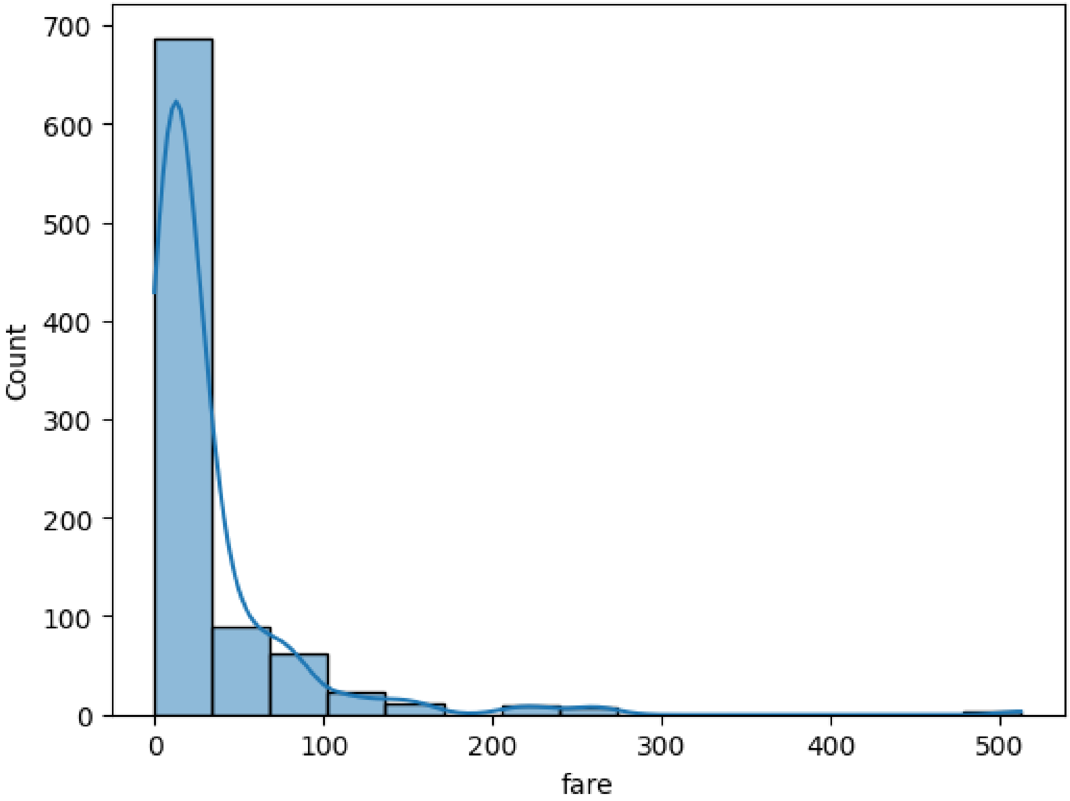
[72]:
```
sns.histplot(df['fare'], kde= True)
```

[72]:
```
<Axes: xlabel='fare', ylabel='Count'>
```



```
[74]: sns.histplot(df['fare'], kde= True, bins = 15)
```

```
[74]: <Axes: xlabel='fare', ylabel='Count'>
```



Conclusion- Seaborn is an advanced data visualisation library built on top of Matplotlib library. In this assignment, we looked at how we can draw distributional and categorical plots using the Seaborn library. We have seen how to plot matrix plots in Seaborn. We also saw how to change plot styles and use grid functions to manipulate subplots.