Exploratory Data Analysis (EDA)

This will show us how to do EDA with Python.

Three important steps in EDA

1. Understand data
2. Clean data
3. Find a relationship between data

In [ ]:
```
# import libraries

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

In [ ]:
```
# step-1 Load any dataset into a variable

kashti = sns.load_dataset("titanic")
```

In [ ]:
```
# step-2 convert that dataset to csv file

kashti.to_csv("kashti.csv")
```

In [ ]:
```
# step-3 See complete information of dataset

kashti.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 [ ]:
```
# Step-4 save your dataset with a short name

ks = kashti
```

In [ ]:
```
# Step-5 check initial five rows of dataset

ks.head()
```

Out[ ]:

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

In [ ]:
```
# Step-6 see the shape of dataset to see number of rows and columns

ks.shape
```

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

In [ ]:
```
# Step-7 to see last five rows of dataset
ks.tail()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark
886	0	2	male	27.0	0	0	13.00	S	Second	man	True	NaN	Southar
887	1	1	female	19.0	0	0	30.00	S	First	woman	False	B	Southar
888	0	3	female	NaN	1	2	23.45	S	Third	woman	False	NaN	Southar
889	1	1	male	26.0	0	0	30.00	C	First	man	True	C	Cher
890	0	3	male	32.0	0	0	7.75	Q	Third	man	True	NaN	Queen

In [ ]:
```
# Step-8 get statistical information of numeric variables

ks.describe()
```

Out[ ]:

	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 [ ]:
```
# Step-9 Find unique values in rows
```

```
ks.nunique()
```

Out[ ]:
```
survived         2
pclass           3
sex              2
age             88
sibsp            7
parch            7
fare           248
embarked         3
class            3
who              3
adult_male       2
deck             7
embark_town      3
alive            2
alone            2
dtype: int64
```

- Now look at above dataset and find cells having most number of unique values and try to remove them. Cells should have less number of unique values.
- In this case, age has 88 different variables and fare has 248 different variables.

In [ ]:
```
# Step-10 get column names

ks.columns
```

Out[ ]:
```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'deck', 'embark_town',
       'alive', 'alone'],
      dtype='object')
```

In [ ]:
```
# Step-10 unique values in a particular column

ks['who'].unique()
```

Out[ ]:
```
array(['man', 'woman', 'child'], dtype=object)
```

Cleaning and Filtering the Data

In [ ]:
```
# Step-1 Find missing values in a dataset
# True means missing values

ks.isnull()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town
0	False	False	False	False	False	False	False	False	False	False	False	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	False	False	False	False	False	False	False	False	False	False	False	True	False
887	False	False	False	False	False	False	False	False	False	False	False	False	False

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town
888	False	False	False	True	False	False	False	False	False	False	False	True	False
889	False	False	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	False	False	True	False

891 rows × 15 columns

In [ ]:
```
# Step-1 Find missing values and then sum them for a complete column

ks.isnull().sum()
```

Out[ ]:
```
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 [ ]:
```
# Step-2 Remove or drop the column with the most missing values (deck = 688)

ks_clean = ks.drop(['deck'], axis=1)
ks_clean.head()
```

Out[ ]:

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

In [ ]:
```
# Step-3 Recheck null values and then sum them up
# still (age=177) has many empty cells

ks_clean.isnull().sum()
```

Out[ ]:
```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
```

```
who              0
adult_male       0
embark_town      2
alive            0
alone            0
dtype: int64
```

In [ ]:
```
ks_clean.shape
```

Out[ ]:
```
(891, 14)
```

In [ ]:
```
# Step-4 Drop all null values using dropna()

ks_clean.dropna()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	embark_town
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Southampton
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	Cherbourg
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Southampton
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	Southampton
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Southampton
...	...	...	...	...	...	...	...	...	...	...	...	...
885	0	3	female	39.0	0	5	29.1250	Q	Third	woman	False	Queenstown
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	Southampton
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	Southampton
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	Cherbourg
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	Queenstown

712 rows × 14 columns

In [ ]:
```
ks_clean.dropna().shape
```

Out[ ]:
```
(712, 14)
```

In [ ]:
```
# update our variable

ks_clean = ks_clean.dropna()
```

In [ ]:
```
ks_clean.shape
```

Out[ ]:
```
(712, 14)
```

In [ ]:
```
# Step-5 All missing values have been removed

ks_clean.isnull().sum()
```

Out[ ]:
```
survived       0
pclass         0
sex            0
```

```
age            0
sibsp          0
parch          0
fare           0
embarked       0
class          0
who            0
adult_male     0
embark_town    0
alive          0
alone          0
dtype: int64
```

In [ ]:
```
# Step-6 count the values in particular columns

ks_clean['sex'].value_counts()
```

Out[ ]:
```
male      453
female    259
Name: sex, dtype: int64
```

In [ ]:
```
# Step-7 Check uncleaned data

ks.describe()
```

Out[ ]:

	surviv	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 [ ]:
```
# Step-9 Check cleaned data, most of outliers have been removed and results are better

ks_clean.describe()
```

Out[ ]:

	survived	pclass	age	sibsp	parch	fare
count	712.000000	712.000000	712.000000	712.000000	712.000000	712.000000
mean	0.404494	2.240169	29.642093	0.514045	0.432584	34.567251
std	0.491139	0.836854	14.492933	0.930692	0.854181	52.938648
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	1.000000	20.000000	0.000000	0.000000	8.050000
50%	0.000000	2.000000	28.000000	0.000000	0.000000	15.645850
75%	1.000000	3.000000	38.000000	1.000000	1.000000	33.000000
max	1.000000	3.000000	80.000000	5.000000	6.000000	512.329200

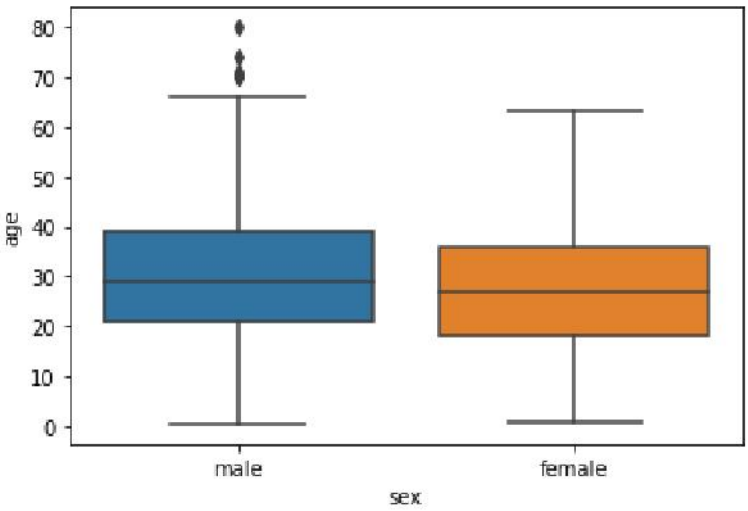
> Find and remove outliers

In [ ]:
```
# Step-1 Check list of columns

ks_clean.columns
```

Out[ ]:
```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'embark_town', 'alive',
       'alone'],
      dtype='object')
```

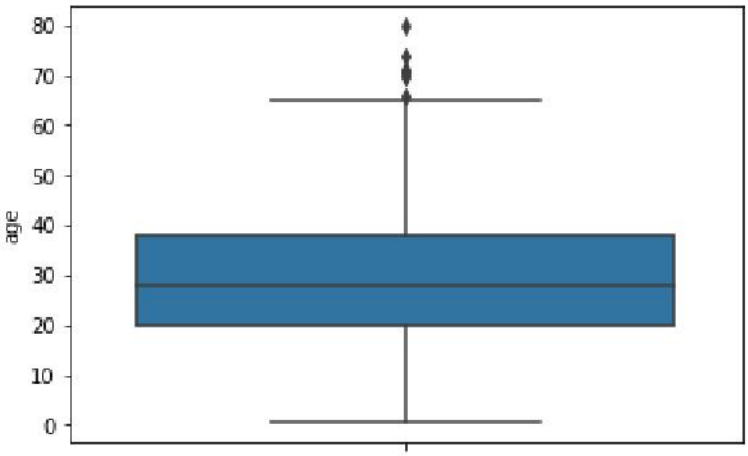
In [ ]:
```
# Step-2 Draw boxplot of any two variable, observe the outliers

sns.boxplot(x='sex', y='age', data=ks_clean)
```

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



In [ ]:
```
# Step-3 There are outliers in the agee of males so check them individually

sns.boxplot(y='age', data=ks_clean)
```

Out[ ]:
```
<AxesSubplot:ylabel='age'>
```

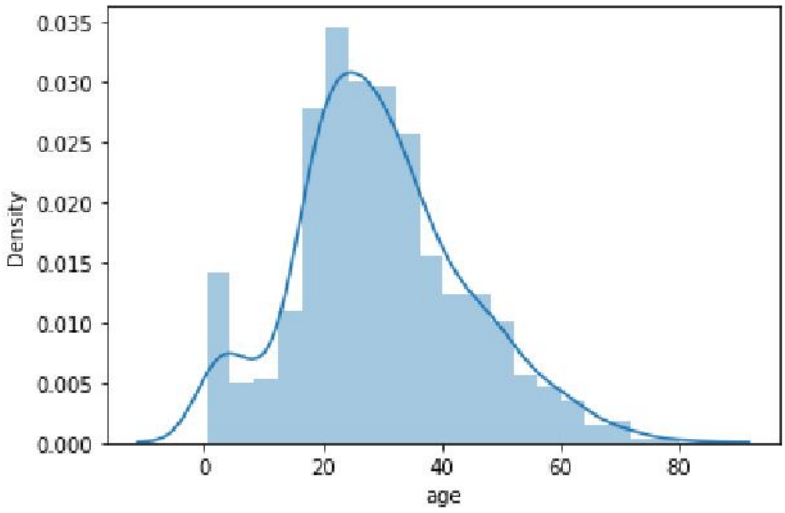


In [ ]:
```
# Step-4 Draw distance plot to see outliers better
# distplot should be in bell curve or across the mean, dispersion is higher
# this is called normality check

sns.distplot(ks_clean['age'])
```

```
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distp
lot` is a deprecated function and will be removed in a future version. Please adapt your code t
o use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axe
```

```
s-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

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



In [ ]:
```
# Step-5 Removing outliers

ks_clean['age'].mean()
ks_clean.head()
```

Out[ ]:

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

In [ ]:
```
ks_clean['age']<68
ks_clean.head()
```

Out[ ]:

	surviv	pclass	sex	age	sibsp	parch	fare	embemb	class	who	adult_male	embark_town	al
0	0	3	male	22.0	1	0	7.2500	S	Third	man	man	Southampton	
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	Cherbourg	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Southampton	
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	Southampton	
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Southampton	

In [ ]:
```
ks_clean = ks_clean[ks_clean['age']<68]
ks_clean.head()
```

Out[ ]:

	surviv	pclass	sex	age	sibsp	parch	fare	embemb	class	who	adult_male	embark_town	al
0	0	3	male	22.0	1	0	7.2500	S	Third	man	man	Southampton	
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	Cherbourg	

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	embark_town	al
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Southampton	
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	Southampton	
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Southampton	

In [ ]:
```
# Now check what happens after removing outliers

ks_clean.shape
```

Out[ ]:
```
(705, 14)
```

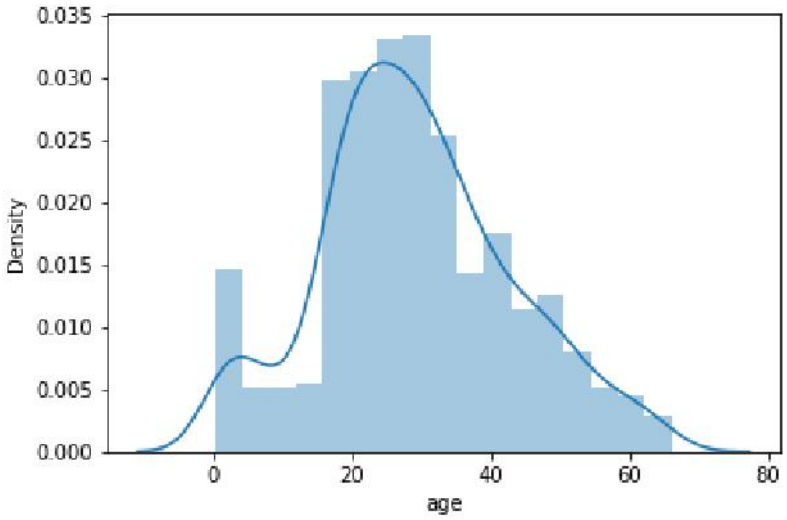
In [ ]:
```
ks_clean['age'].mean()
```

Out[ ]:
```
29.21797163120567
```

In [ ]:
```
# now draw distplot first

sns.distplot(ks_clean['age'])
```

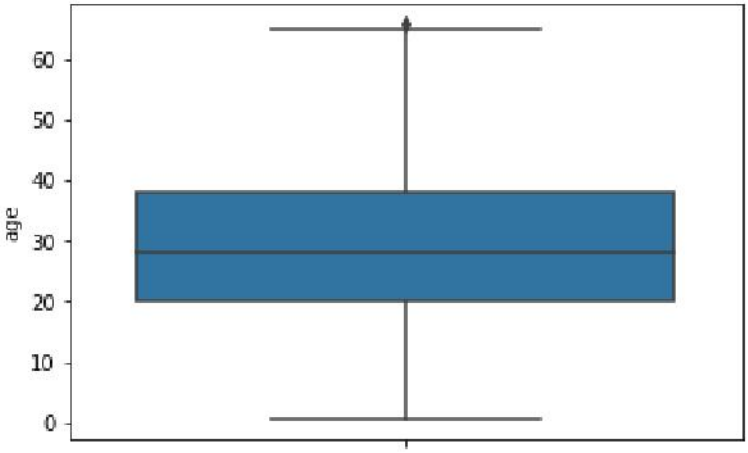
```
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distp
lot` is a deprecated function and will be removed in a future version. Please adapt your code t
o use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axe
s-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

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



In [ ]:
```
# Now draw the boxplot, outliers have been significantly removed

sns.boxplot(y='age', data=ks_clean)
```

Out[ ]:
```
<AxesSubplot:ylabel='age'>
```



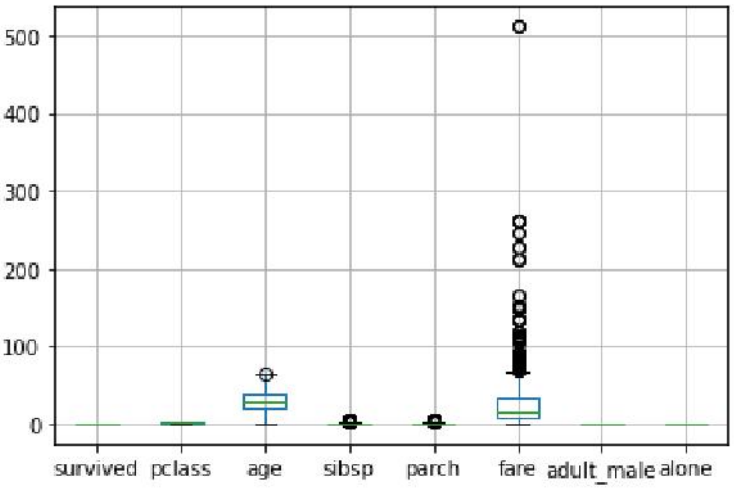
In [ ]:
```
ks_clean.head()
```

Out[ ]:

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

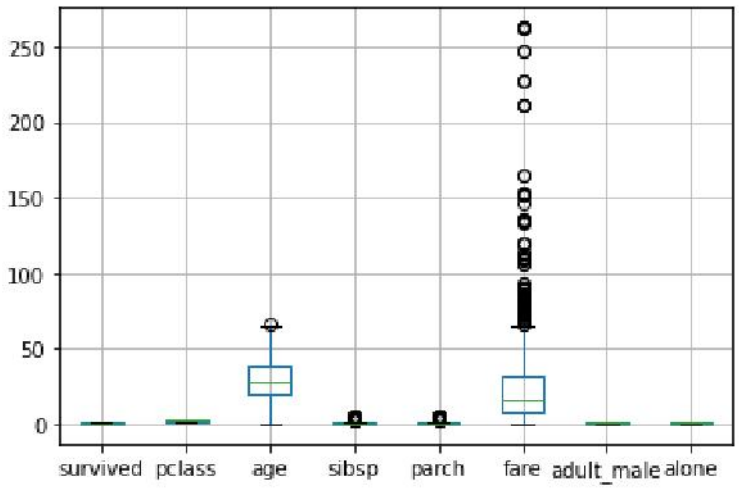
In [ ]:
```
# Draw a boxplot of whole dataset
# fare seems to have outliers, now we need to remove these

ks_clean.boxplot()
```

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



In [ ]:
```
# Clean the above data by simply having fare values less than 300

ks_clean = ks_clean[ks_clean['fare']<300]
ks_clean.boxplot()
```

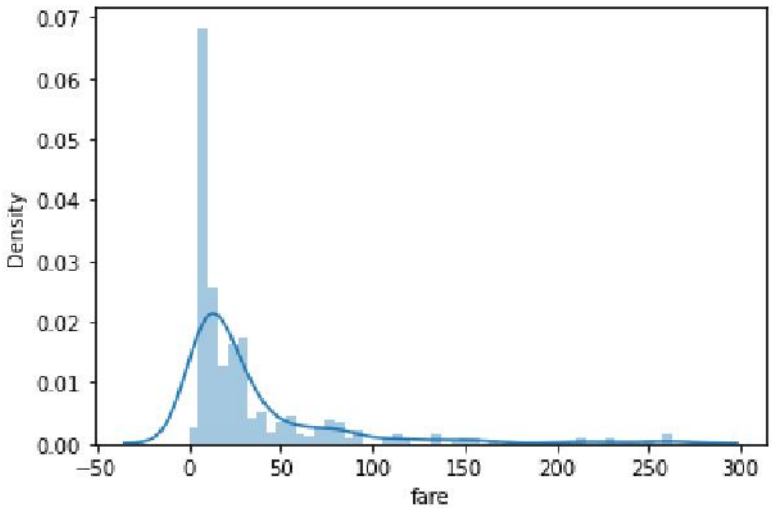
Out[ ]:
```
<AxesSubplot:>
```



In [ ]:
```
# Draw histogram / distplot of 'fare' to see its bell curse

sns.distplot(ks_clean['fare'])
```

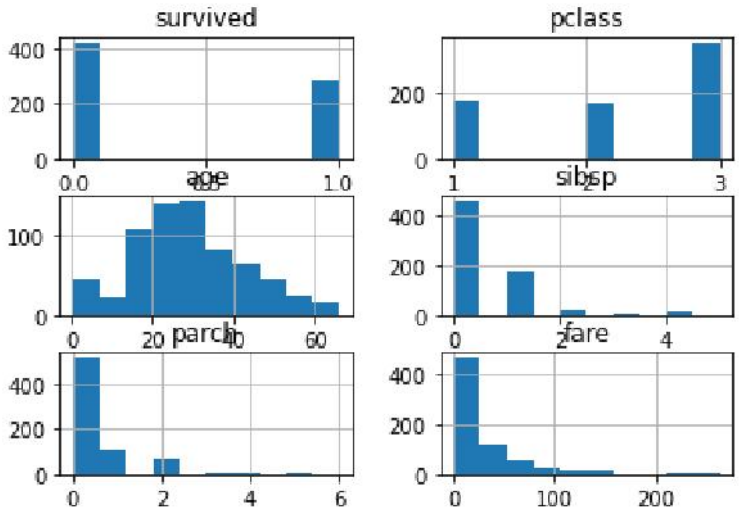
```
c:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distp
lot` is a deprecated function and will be removed in a future version. Please adapt your code t
o use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axe
s-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

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



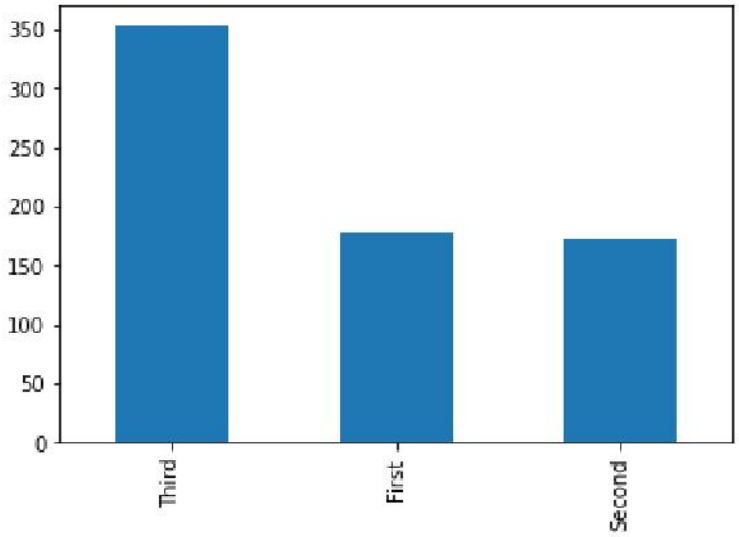
In [ ]:
```
ks_clean.hist()
```

Out[ ]:
```
array([[<AxesSubplot:title={'center':'survived'}>,
        <AxesSubplot:title={'center':'pclass'}>],
       [<AxesSubplot:title={'center':'age'}>,
        <AxesSubplot:title={'center':'sibsp'}>],
       [<AxesSubplot:title={'center':'parch'}>,
        <AxesSubplot:title={'center':'fare'}>]], dtype=object)
```



In [ ]:
```
# Draw barplots

pd.value_counts(ks_clean['class']).plot.bar()
```

Out[ ]: <AxesSubplot:>



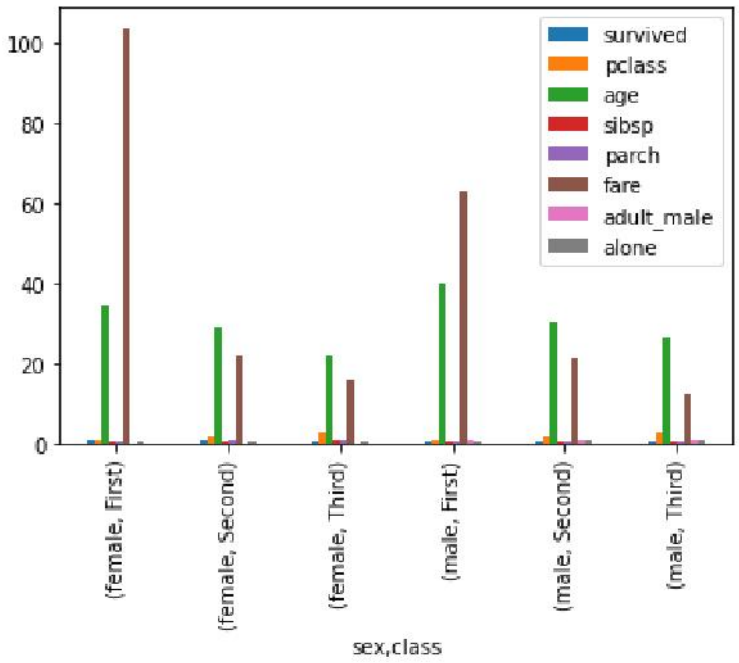
In [ ]:
```
ks_clean.groupby(['sex', 'class']).mean()
```

Out[ ]:

		survived	pclass	age	sibsp	parch	fare	adult_male	alone
sex	**class**								
female	**First**	0.963415	1.0	34.231707	0.560976	0.512195	103.696393	0.000000	0.353659
	Second	0.918919	2.0	28.722973	0.500000	0.621622	21.951070	0.000000	0.405405
	Third	0.460784	3.0	21.750000	0.823529	0.950980	15.875369	0.000000	0.372549
male	**First**	0.389474	1.0	40.067579	0.389474	0.336842	62.901096	0.968421	0.526316
	Second	0.153061	2.0	30.340102	0.377551	0.244898	21.221429	0.908163	0.632653
	Third	0.151394	3.0	26.143108	0.494024	0.258964	12.197757	0.888446	0.737052

In [ ]:
```
ks_clean.groupby(['sex', 'class']).mean().plot.bar()
```

Out[ ]: <AxesSubplot:xlabel='sex,class'>



In [ ]:
```
ks.groupby(['sex', 'class']).mean()
```

Out[ ]:

sex	class	survived	pclass	age	sibsp	parch	fare	adult_male	alone
female	First	0.968085	1.0	34.611765	0.553191	0.457447	106.125798	0.000000	0.361702
	Second	0.921053	2.0	28.722973	0.486842	0.605263	21.970121	0.000000	0.421053
	Third	0.500000	3.0	21.750000	0.895833	0.798611	16.118810	0.000000	0.416667
male	First	0.368852	1.0	41.281386	0.311475	0.278689	67.226127	0.975410	0.614754
	Second	0.157407	2.0	30.740707	0.342593	0.222222	19.741782	0.916667	0.666667
	Third	0.135447	3.0	26.507589	0.498559	0.224784	12.661633	0.919308	0.760807

Relationships

In [ ]:
```
# 1. Correlation
# if correlation between two varialbes is 1 they are directly proportional
# if correlation between two varialbes is -1 they are indirectly proportional
# if correlation between two varialbes is 0 they are not related

ks_clean.corr()
```

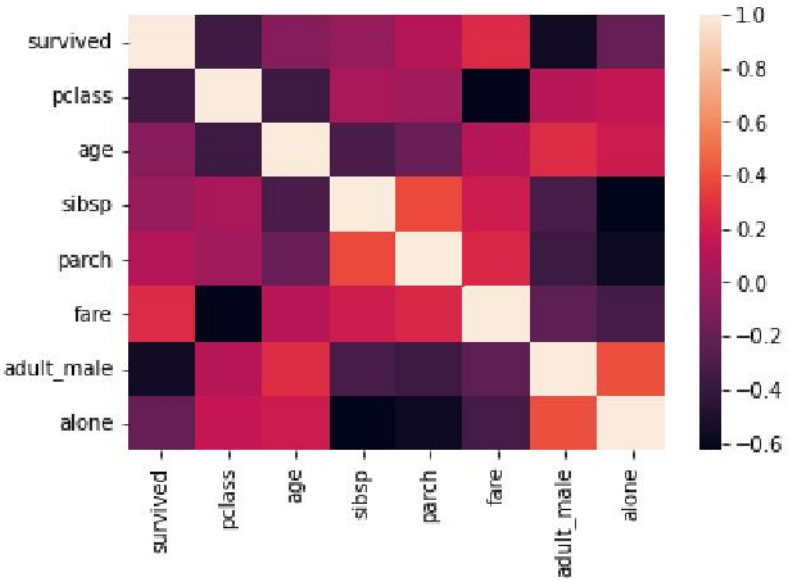
Out[ ]:

	survived	pclass	age	sibsp	parch	fare	adult_male	alone
survived	1.000000	-0.356549	-0.074335	-0.014483	0.095426	0.273531	-0.554567	-0.201175
pclass	-0.356549	1.000000	-0.365121	0.061354	0.022519	-0.617591	0.102930	0.156030
age	-0.074335	-0.365121	1.000000	-0.308906	-0.186271	0.103100	0.275035	0.187284
sibsp	-0.014483	0.061354	-0.308906	1.000000	0.381803	0.197954	-0.311622	-0.629200
parch	0.095426	0.022519	-0.186271	0.381803	1.000000	0.259948	-0.366540	-0.574701
fare	0.273531	-0.617591	0.103100	0.197954	0.259948	1.000000	-0.228675	-0.333949
adult_male	-0.554567	0.102930	0.275035	-0.311622	-0.366540	-0.228675	1.000000	0.402214
alone	-0.201175	0.156030	0.187284	-0.629200	-0.574701	-0.333949	0.402214	1.000000

In [ ]:
```
cor_ks_clean = ks_clean.corr()
```

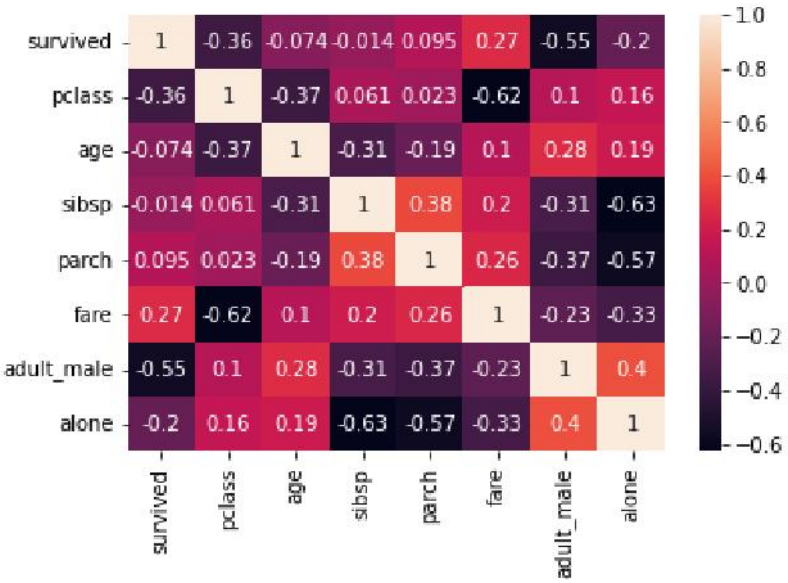
In [ ]:
```
# Draw heatmap of correlation

sns.heatmap(cor_ks_clean)
```

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



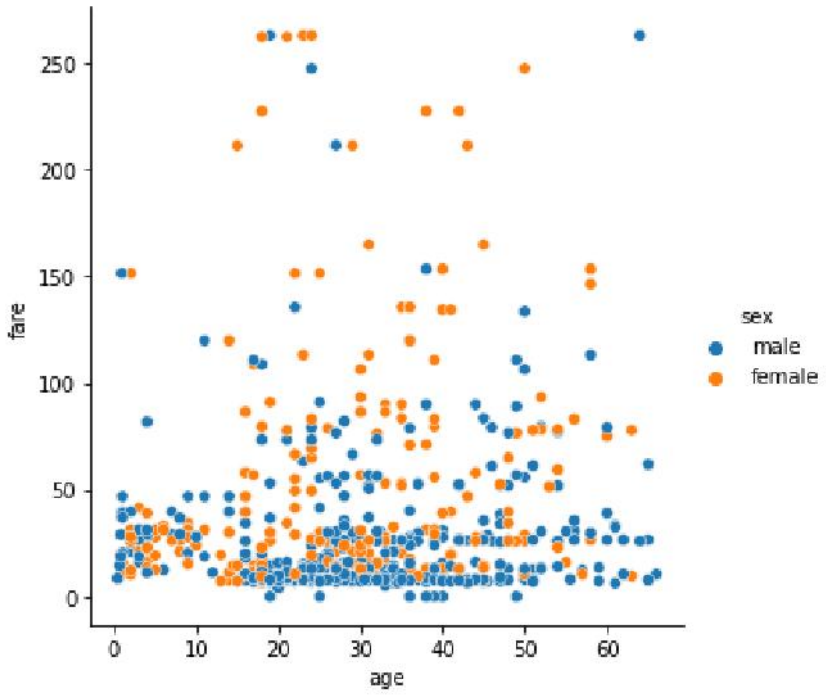
In [ ]:
```
sns.heatmap(cor_ks_clean, annot=True)
```

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



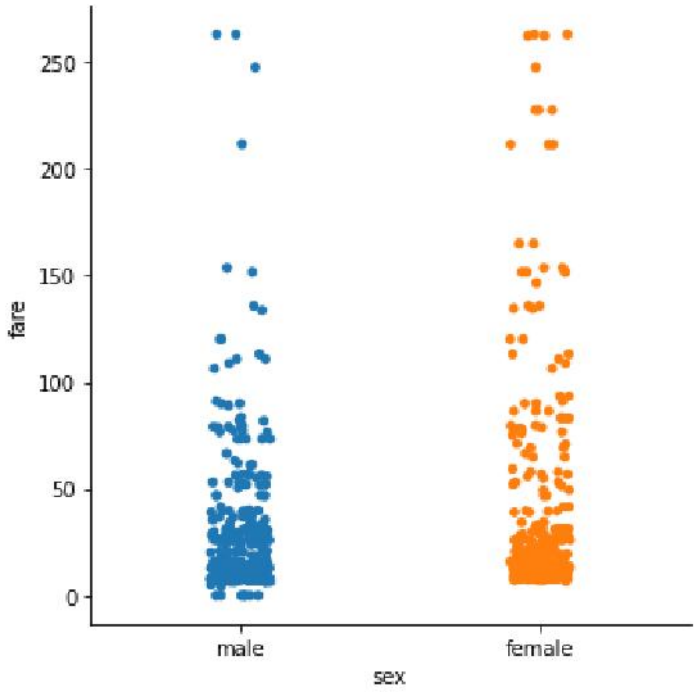
In [ ]:
```
# Relation plot of numeric variables

sns.relplot(x='age', y='fare', hue='sex', data=ks_clean)
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0xedec700>
```



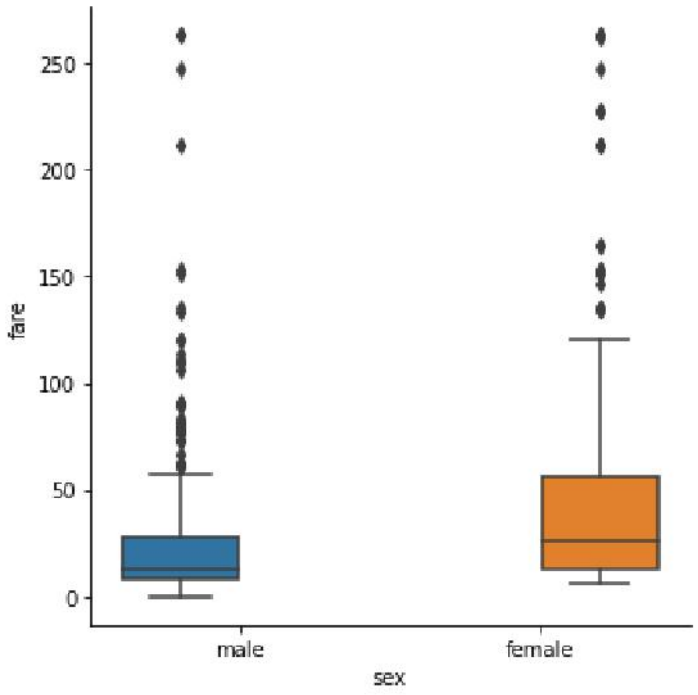
In [ ]:
```
# Draw catplot

sns.catplot(x='sex', y='fare', hue='sex', data=ks_clean)
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0xed52070>
```



In [ ]:
```
# Draw about catplot with box
# too many outliers in fare

sns.catplot(x='sex', y='fare', hue='sex', data=ks_clean, kind='box')
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0xeef0880>
```

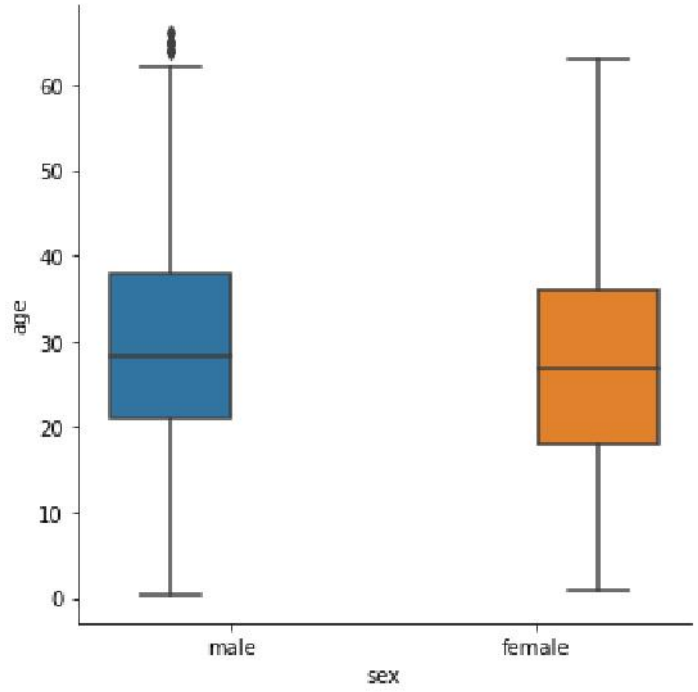


In [ ]:

```
# Draw about catplot with box
# Too many outliers in fare, so draw age
# Still few outliers in the age of males
# We can remove the outliers that are below 65 in the age of males

sns.catplot(x='sex', y='age', hue='sex', data=ks_clean, kind='box')
```

Out[ ]:

```
<seaborn.axisgrid.FacetGrid at 0xf0512e0>
```



- Removing outliers using log
- We'll add a new column fare_log in the dataset having the log of all fares
- And then we'll draw catplot of dataset with fare_log and see the difference

In [ ]:

```
# log transformation
# add a new column in ks_clean dataset

ks_clean['fare_log'] = np.log(ks_clean['fare'])
ks_clean.head()
```

```
c:\ProgramData\Anaconda3\lib\site-packages\pandas\core\arraylike.py:358: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

Out[ ]:

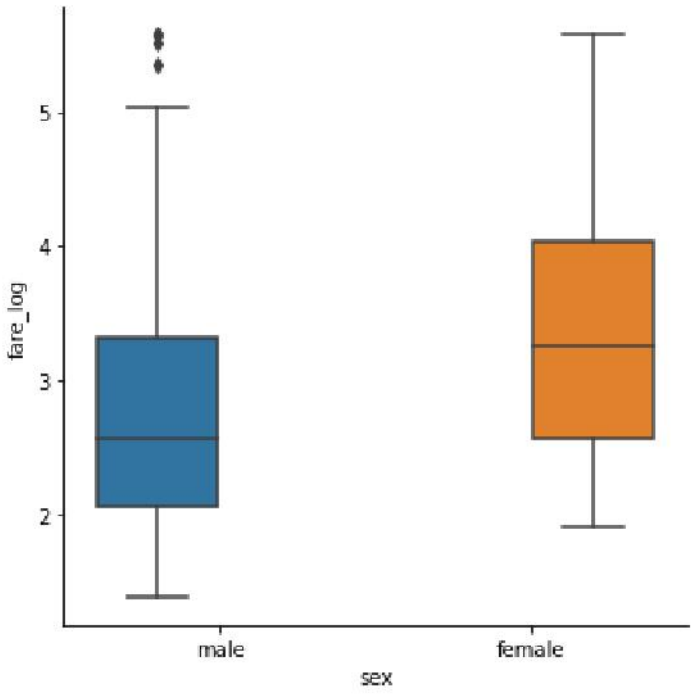
	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	embark_town	al
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Southampton	
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	Cherbourg	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Southampton	
3	1	1	female	35.0	1	0	53.1000	S	First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

In [ ]:

```
# Now draw the catplot with new column and compare with previous

sns.catplot(x='sex', y='fare_log', hue='sex', data=ks_clean, kind='box')
```

Out[ ]:

```
<seaborn.axisgrid.FacetGrid at 0xeef0790>
```



In [ ]: