EXPLORATORY DATA ANALYSIS

This will help us how we can do EDA in python

Three important steps to keep in mind are:

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

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

In [2]:
```
kashti=sns.load_dataset("titanic")
kashti.to_csv("kashti.csv")
```

In [3]:
```
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 [4]:
```
ks= kashti
```

In [5]:
```
ks.head()
```

Out[5]:

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

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	decl
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]:
```
#rows amd column
ks.shape
```

Out[7]:
```
(891, 15)
```

In [8]:
```
ks.tail()
```

Out[8]:

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

In [9]:
```
ks.describe()
```

Out[9]:

	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 [11]:
```
#unique values
ks.nunique()
```

Out[11]:
```
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
```

In [12]:
```
ks.columns
```

Out[12]:
```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'deck', 'embark_town',
       'alive', 'alone'],
      dtype='object')
```

In [14]:
```
#finding unique value in a specific column
ks["adult_male"].unique()
```

Out[14]:
```
array([ True, False])
```

In [15]:
```
 ks["sex"].unique()
```

Out[15]:
```
array(['male', 'female'], dtype=object)
```

Assignment work

In [16]:
```
#finding unique value in two columns
(ks['sex'].append(ks['class'])).unique()
```

Out[16]:
```
C:\Users\Azka\AppData\Local\Temp\ipykernel_9544\2810069508.py:1: FutureWarning: The
series.append method is deprecated and will be removed from pandas in a future versi
on. Use pandas.concat instead.
  (ks['sex'].append(ks['class'])).unique()
array(['male', 'female', 'Third', 'First', 'Second'], dtype=object)
```

Cleaning and filtering the data

In [19]:
```
ks.isnull().sum()
```

Out[19]:
```
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 [47]:
```
#removing missing values
ks_clean= ks.drop(["deck"],axis=1)
ks_clean.head()
```

Out[47]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	emb
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Sou
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	Sou
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	Sou
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Sou

In [48]:
```
ks_clean.isnull().sum()
```

Out[48]:
```
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 [49]:
```
ks_clean.shape
```

Out[49]:
```
(891, 14)
```

In [44]:
```
ks.shape
```

Out[44]:
```
(891, 15)
```

In [50]:
```
ks_clean.dropna()
```

Out[50]:

	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
...	...	...	...	...	...	...	...	...	...	...	...

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
885	0	3	female	39.0	0	5	29.1250	Q	Third	woman	False
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
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

712 rows × 14 columns

In [30]:
```
891-177-2-2
```

Out[30]:
```
714
```

In [51]:
```
ks_clean = ks_clean.dropna()
```

In [52]:
```
ks_clean.shape
```

Out[52]:
```
(712, 14)
```

In [53]:
```
# the data is clean now
ks_clean.isnull().sum()
```

Out[53]:
```
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 [54]:
```
ks_clean["sex"].value_counts()
```

Out[54]:
```
male      453
female    259
Name: sex, dtype: int64
```

In [55]:
```
ks_clean["age"].value_counts()
```

Out[55]:
```
24.00    30
22.00    27
18.00    26
19.00    25
28.00    25
```

```
          ..
36.50     1
55.50     1
0.92      1
23.50     1
74.00     1
Name: age, Length: 88, dtype: int64
```

In [56]:
```
ks.describe()
```

Out[56]:

	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 [57]:
```
ks_clean.describe()
```

Out[57]:

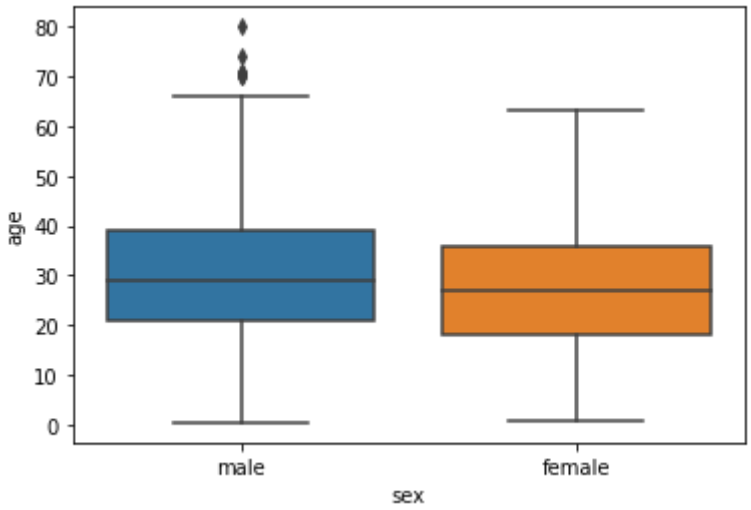
	survived	pclass	age	sibsp	parch	fare
count	712.000000	712.000000	712.000000	712.000000	712.000000	712.000000
mean	0.404494	2.240169	29.642093	0.514045	0.432584	34.567251
std	0.491139	0.836854	14.492933	0.930692	0.854181	52.938648
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	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

In [58]:
```
ks_clean.columns
```

Out[58]:
```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'embark_town', 'alive',
       'alone'],
      dtype='object')
```

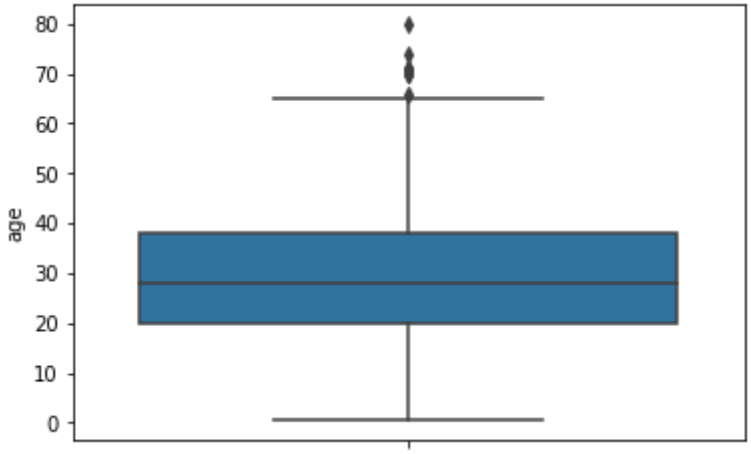
In [59]:
```
sns.boxplot(x="sex",y="age",data= ks_clean)
```

Out[59]:
```
<AxesSubplot:xlabel='sex', ylabel='age'>
```



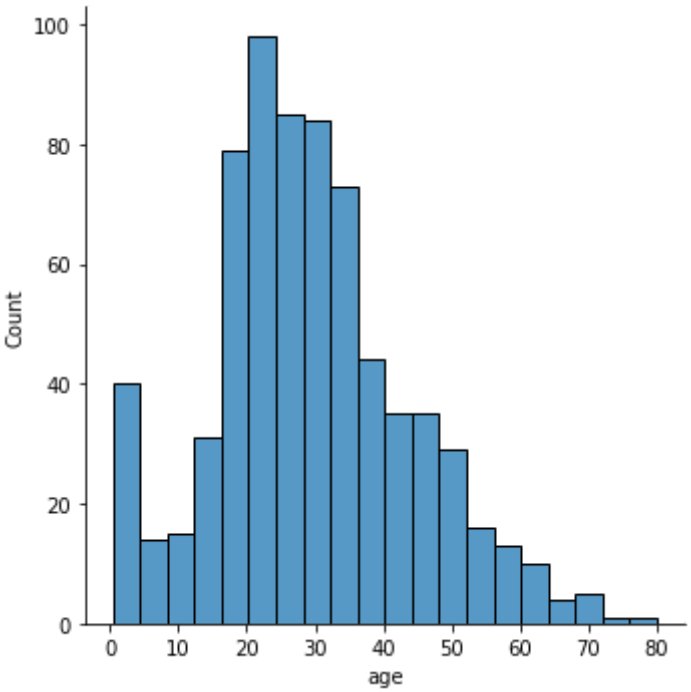
In [61]:
```
sns.boxplot(y='age',data= ks_clean)
```

Out[61]:
```
<AxesSubplot:ylabel='age'>
```



In [67]:
```
sns.displot(ks_clean['age'])
```

Out[67]:
```
<seaborn.axisgrid.FacetGrid at 0x238b3d0f790>
```



In [68]:
```
#removing outliers
ks_clean['age'].mean()
```

Out[68]:
```
29.64209269662921
```

In [77]:
```
ks_clean= ks_clean[ks_clean['age']<60 ]
ks_clean.head()
ks_clean.shape
```

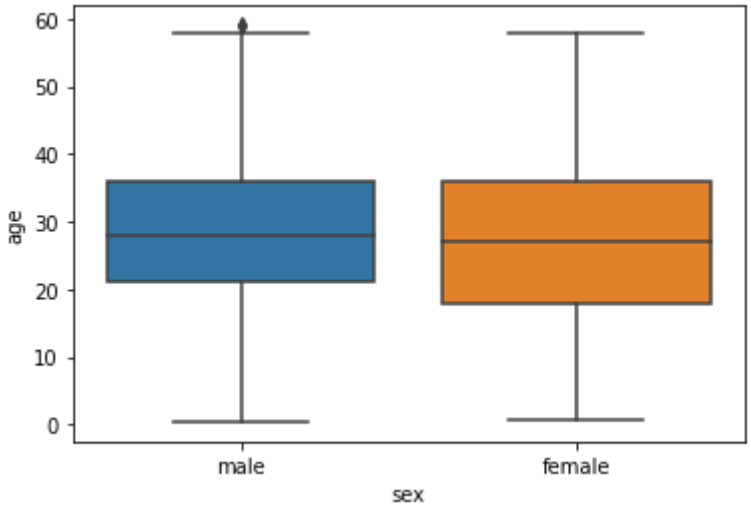
Out[77]:
```
(687, 14)
```

In [78]:
```
ks_clean['age'].mean()
```

Out[78]:
```
28.347409024745268
```

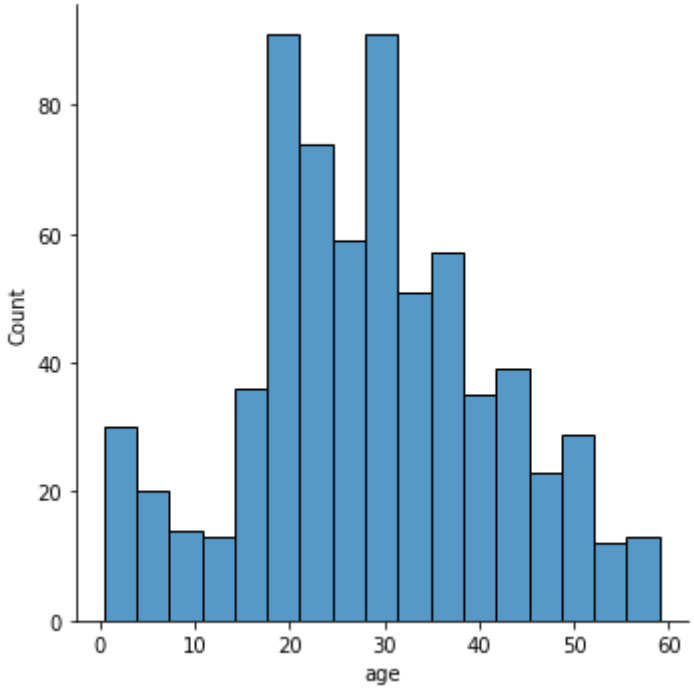
In [79]:
```
sns.boxplot(x="sex",y="age",data= ks_clean)
```

Out[79]:
```
<AxesSubplot:xlabel='sex', ylabel='age'>
```

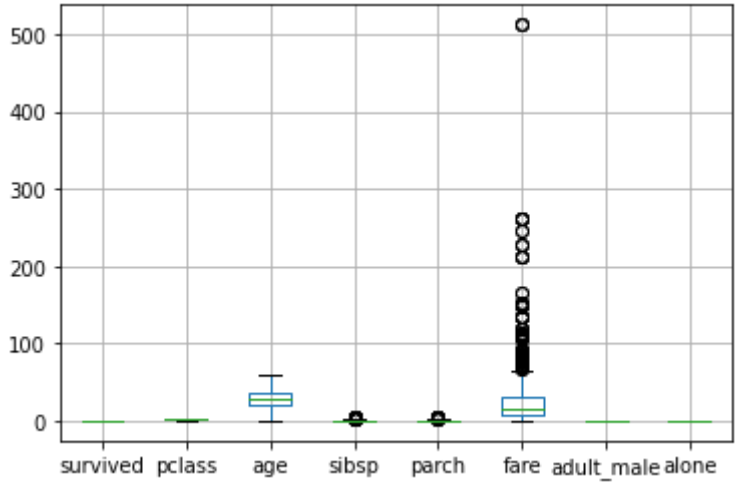


In [80]:
```
sns.displot(ks_clean['age'])
```

Out[80]:
```
<seaborn.axisgrid.FacetGrid at 0x238b3d0e530>
```

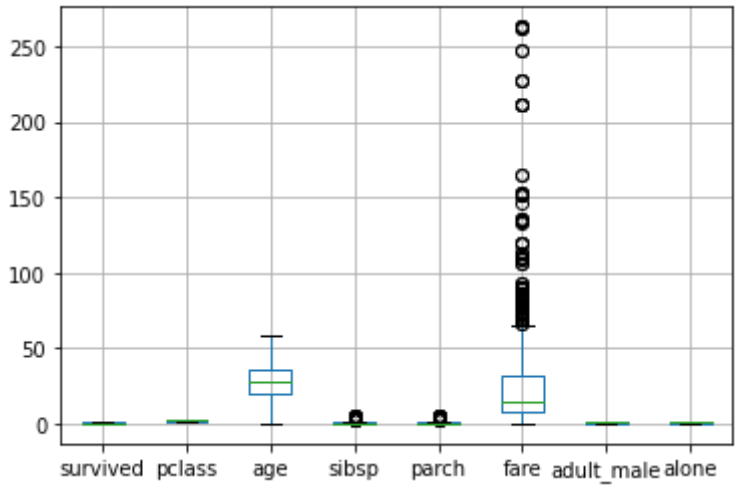


In [86]:
```
ks_clean.boxplot()
plt.show()
```



In [88]:
```
ks_clean= ks_clean[ks_clean["fare"]<300]
ks_clean.boxplot()
```

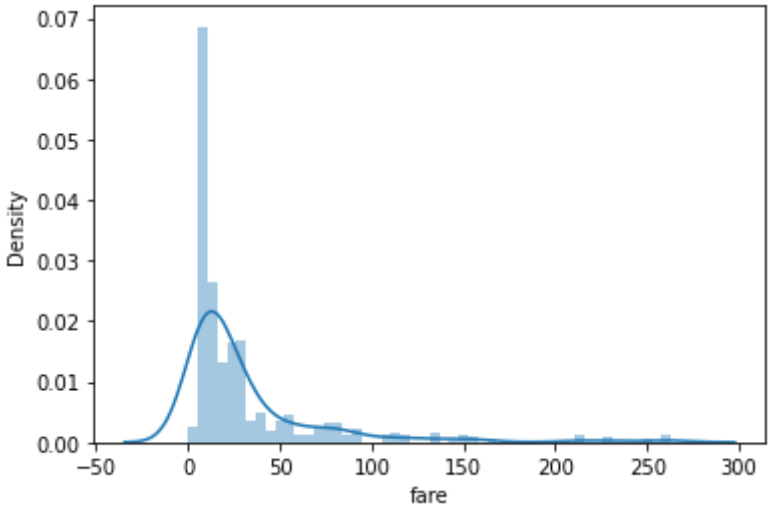
Out[88]: <AxesSubplot:>



In [89]:
```
sns.distplot(ks_clean["fare"])
```

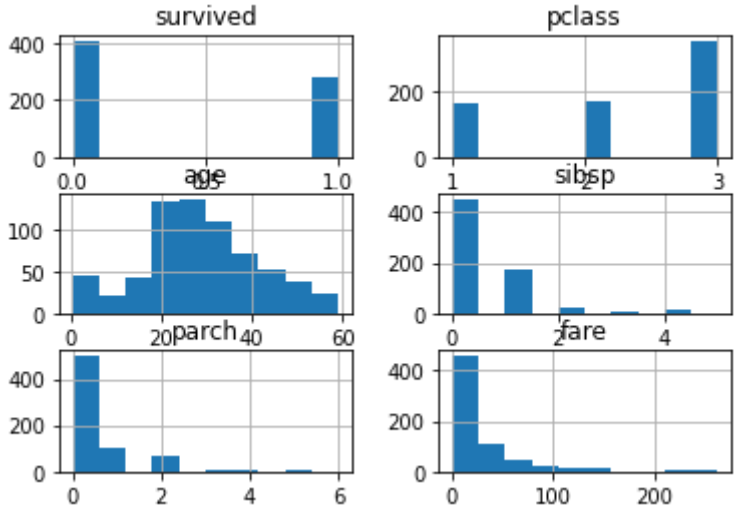
```
C:\Users\Azka\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[89]:
```
<AxesSubplot:xlabel='fare', ylabel='Density'>
```



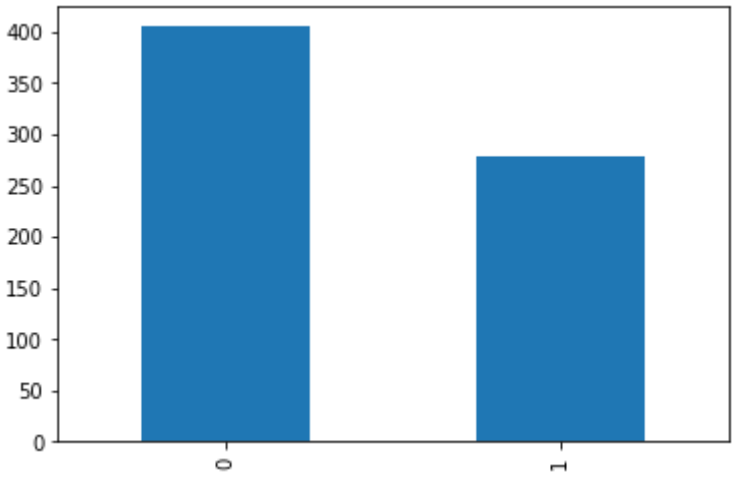
In [90]:
```
ks_clean.hist()
```

Out[90]:
```
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)
```



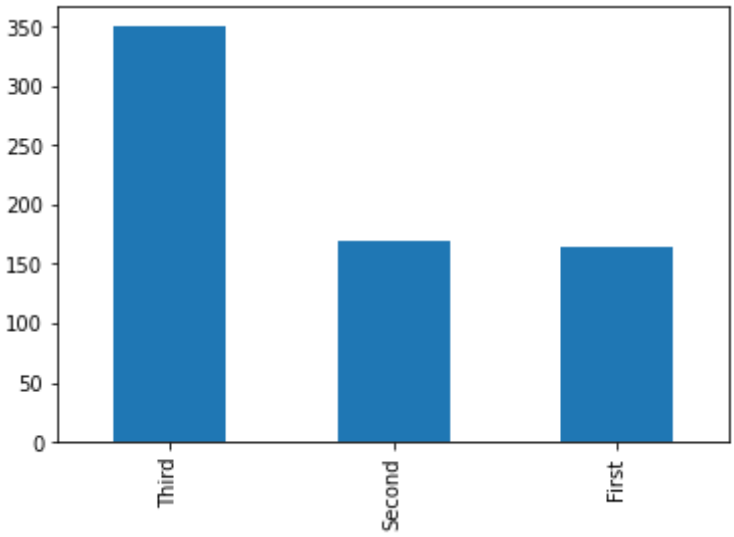
In [94]:
```
pd.value_counts(ks_clean['survived']).plot.bar()
```

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



In [95]:
```
pd.value_counts(ks_clean['class']).plot.bar()
```

Out[95]: <AxesSubplot:>



In [98]:
```
ks_clean.groupby(['sex']).mean()
```

Out[98]:

sex	survived	pclass	age	sibsp	parch	fare	adult_male	alone
female	0.749020	2.082353	27.313725	0.647059	0.725490	45.427354	0.00000	0.376471
male	0.205128	2.382284	28.912984	0.454545	0.265734	24.337421	0.90676	0.666667

In [97]:
```
ks_clean.groupby(['sex','class']).mean()
```

Out[97]:

sex	class	survived	pclass	age	sibsp	parch	fare	adult_male	alone
female	**First**	0.962500	1.0	33.550000	0.550000	0.525000	104.373699	0.000000	0.362500
	Second	0.918919	2.0	28.722973	0.500000	0.621622	21.951070	0.000000	0.405405
	Third	0.455446	3.0	21.341584	0.831683	0.960396	15.937625	0.000000	0.366337
male	**First**	0.423529	1.0	37.440235	0.411765	0.305882	63.216519	0.964706	0.505882

		survived	pclass	age	sibsp	parch	fare	adult_male	alone
sex	class								
	Second	0.147368	2.0	29.319263	0.378947	0.242105	21.260000	0.905263	0.631579
	Third	0.152610	3.0	25.847068	0.497992	0.261044	12.239556	0.887550	0.734940

In [99]:
```
ks.groupby(['sex','class']).mean()
```

Out[99]:

		survived	pclass	age	sibsp	parch	fare	adult_male	alone
sex	class								
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

In [100…
```
#relationship
ks_clean.corr()
```

Out[100…

	survived	pclass	age	sibsp	parch	fare	adult_male	alone
survived	1.000000	-0.376913	-0.062820	-0.021580	0.101012	0.284657	-0.550647	-0.196596
pclass	-0.376913	1.000000	-0.342623	0.059466	0.027224	-0.626093	0.120975	0.159555
age	-0.062820	-0.342623	1.000000	-0.318082	-0.202076	0.091596	0.265445	0.190447
sibsp	-0.021580	0.059466	-0.318082	1.000000	0.381742	0.195031	-0.309017	-0.627571
parch	0.101012	0.027224	-0.202076	0.381742	1.000000	0.234899	-0.379839	-0.574854
fare	0.284657	-0.626093	0.091596	0.195031	0.234899	1.000000	-0.240071	-0.326577
adult_male	-0.550647	0.120975	0.265445	-0.309017	-0.379839	-0.240071	1.000000	0.402767
alone	-0.196596	0.159555	0.190447	-0.627571	-0.574854	-0.326577	0.402767	1.000000

In [102…
```
corr_ks_clean=ks_clean.corr()
corr_ks_clean
```

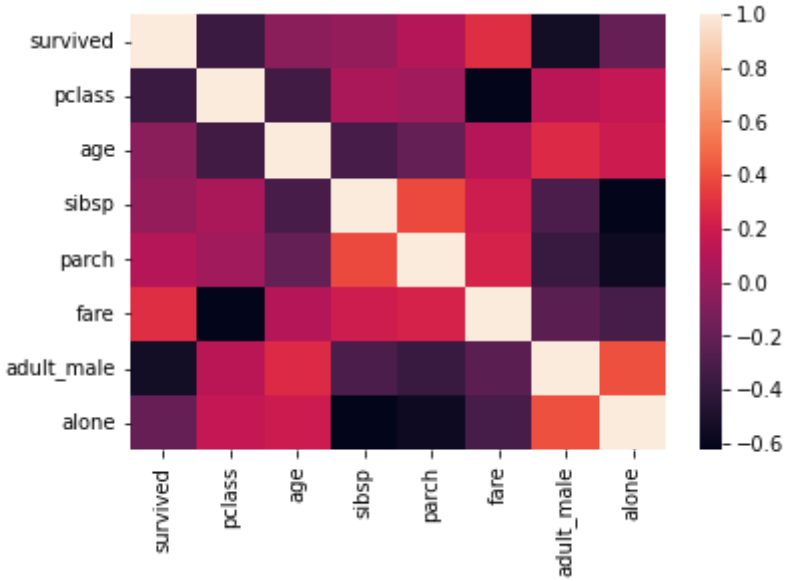
Out[102…

	survived	pclass	age	sibsp	parch	fare	adult_male	alone
survived	1.000000	-0.376913	-0.062820	-0.021580	0.101012	0.284657	-0.550647	-0.196596
pclass	-0.376913	1.000000	-0.342623	0.059466	0.027224	-0.626093	0.120975	0.159555
age	-0.062820	-0.342623	1.000000	-0.318082	-0.202076	0.091596	0.265445	0.190447
sibsp	-0.021580	0.059466	-0.318082	1.000000	0.381742	0.195031	-0.309017	-0.627571
parch	0.101012	0.027224	-0.202076	0.381742	1.000000	0.234899	-0.379839	-0.574854
fare	0.284657	-0.626093	0.091596	0.195031	0.234899	1.000000	-0.240071	-0.326577

	survived	pclass	age	sibsp	parch	fare	adult_male	alone
adult_male	-0.550647	0.120975	0.265445	-0.309017	-0.379839	-0.240071	1.000000	0.402767
alone	-0.196596	0.159555	0.190447	-0.627571	-0.574854	-0.326577	0.402767	1.000000

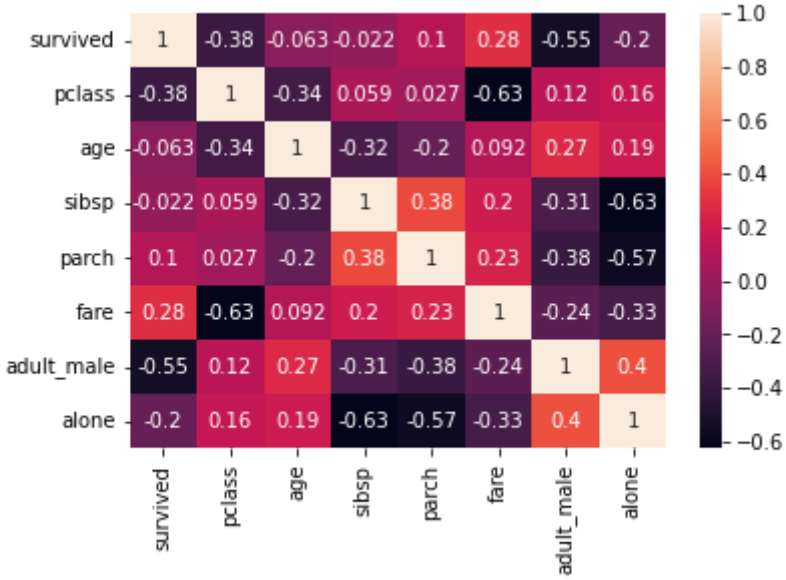
In [103…
```
sns.heatmap(corr_ks_clean)
```

Out[103…
```
<AxesSubplot:>
```



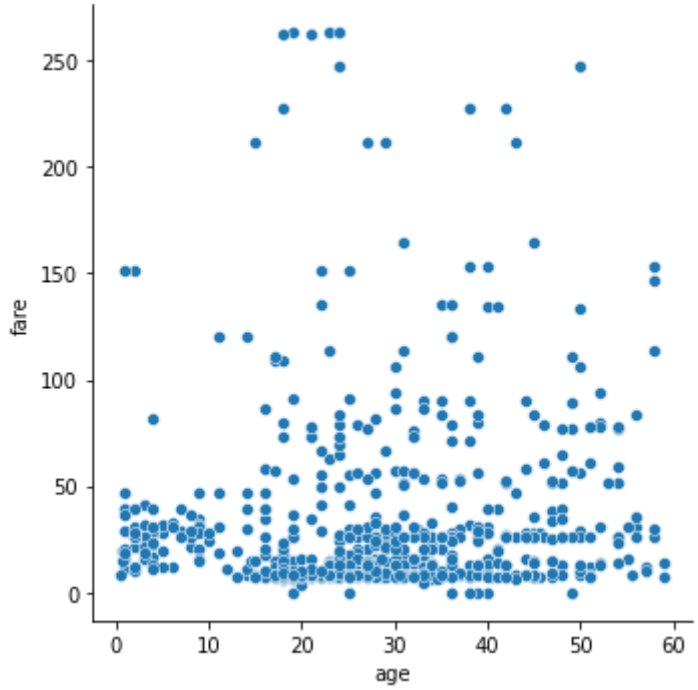
In [104…
```
sns.heatmap(corr_ks_clean,annot=True)
```

Out[104…
```
<AxesSubplot:>
```



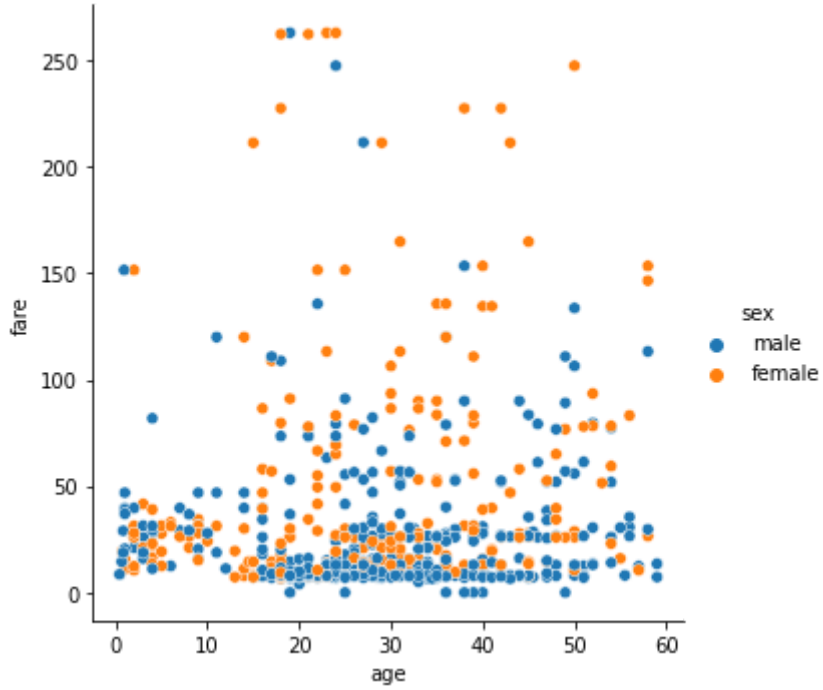
In [105…
```
sns.relplot(x='age',y='fare',data=ks_clean)
```

Out[105…
```
<seaborn.axisgrid.FacetGrid at 0x238bb3d3370>
```



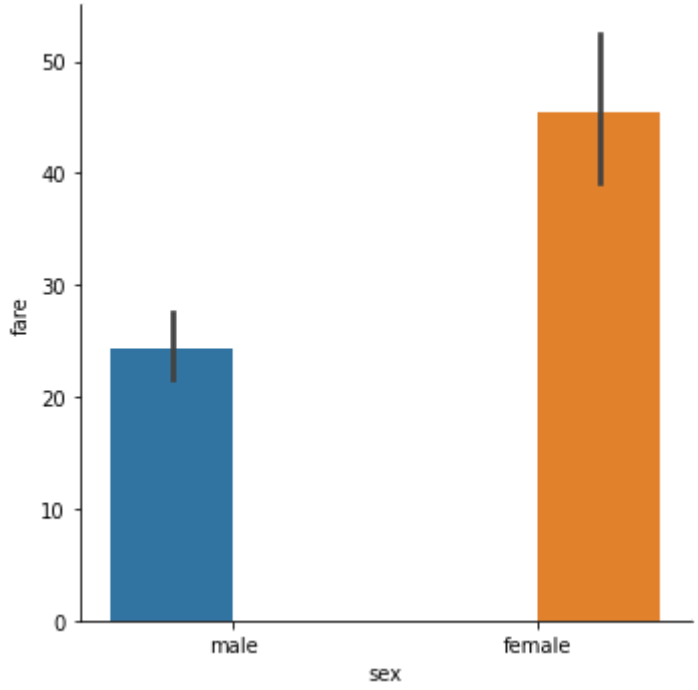
In [106…
```
sns.relplot(x='age',y='fare',hue='sex', data=ks_clean)
```

Out[106…
```
<seaborn.axisgrid.FacetGrid at 0x238bb338430>
```



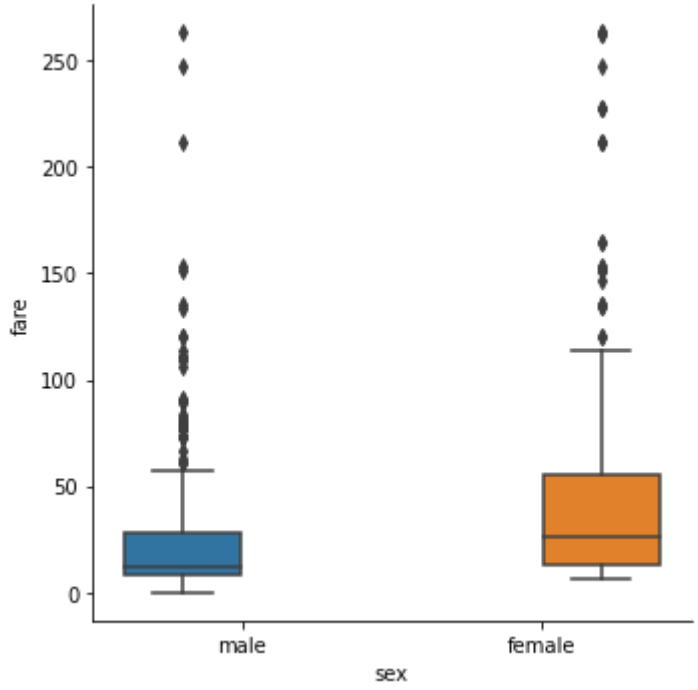
In [109…
```
sns.catplot(x='sex',y='fare',hue='sex', data=ks_clean,kind="bar")
```

Out[109…
```
<seaborn.axisgrid.FacetGrid at 0x238bb62eb00>
```



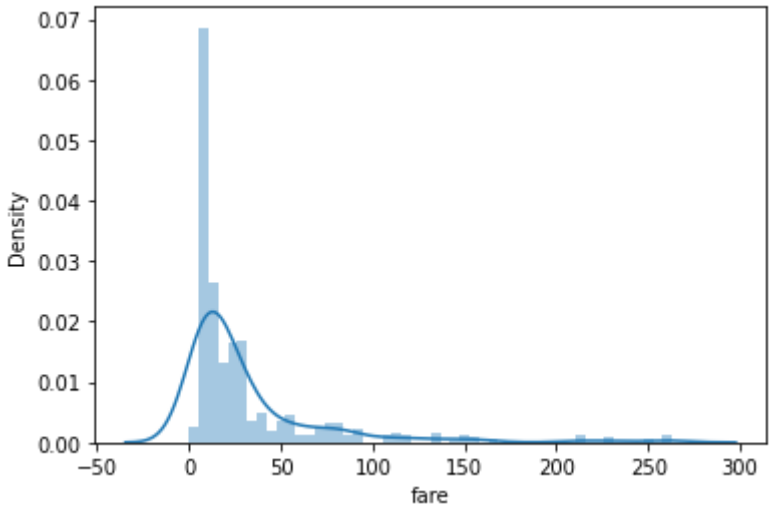
In [110…
```
sns.catplot(x='sex',y='fare',hue='sex', data=ks_clean,kind="box")
```

Out[110…
```
<seaborn.axisgrid.FacetGrid at 0x238bb6cbfa0>
```



In [112…
```
sns.distplot(ks_clean['fare'])
ks_clean['fare_log']=np.log(ks_clean['fare'])
```

```
C:\Users\Azka\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\dist
ributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be re
moved in a future version. Please adapt your code to use either `displot` (a figure-
level function with similar flexibility) or `histplot` (an axes-level function for h
istograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Azka\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core
\arraylike.py:397: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```



In [113…
```
ks_clean.head()
```

Out[113…

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	emb
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Sou
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	Sou
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	Sou
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Sou

In [115…
```
ks_clean.hist
```

Out[115…
```
<bound method hist_frame of      survived  pclass     sex   age  sibsp  parch     fa
re 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
..        ...     ...     ...   ...    ...    ...      ...      ...     ...
885         0       3  female  39.0      0      5  29.1250        Q   Third
886         0       2    male  27.0      0      0  13.0000        S  Second
887         1       1  female  19.0      0      0  30.0000        S   First
889         1       1    male  26.0      0      0  30.0000        C   First
890         0       3    male  32.0      0      0   7.7500        Q   Third

       who  adult_male  embark_town alive  alone  fare_log
0      man        True  Southampton    no  False  1.981001
1    woman       False    Cherbourg   yes  False  4.266662
2    woman       False  Southampton   yes   True  2.070022
3    woman       False  Southampton   yes  False  3.972177
4      man        True  Southampton    no   True  2.085672
..     ...         ...          ...   ...    ...       ...
885  woman       False   Queenstown    no  False  3.371597
886    man        True  Southampton    no   True  2.564949
887  woman       False  Southampton   yes   True  3.401197
889    man        True    Cherbourg   yes   True  3.401197
890    man        True   Queenstown    no   True  2.047693

[684 rows x 15 columns]>
```

In [116…
```
sns.catplot(x='sex',y='fare_log',hue='sex', data=ks_clean,kind="box")
```

Out[116…
```
<seaborn.axisgrid.FacetGrid at 0x238b3ad0160>
```