Exploratory Data Analysis

This will shw us how we can do EDA using 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]:

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

In [2]:

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

In [3]:

```
kashti.to_csv('kashti.csv')
```

In [4]:

```
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 [5]:

```
ks = kashti
```

In [43]:

```
# dataset kis tarah ka hai
ks.head()
```

Out[43]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town

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

In [44]:

```
#rows and column #
ks.shape
```

Out[44]:

```
(891, 15)
```

In [45]:

```
ks.tail()
```

Out[45]:

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

In [46]:

```
ks.describe()
```

Out[46]:

	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 [47]:

```
# unique values
ks.nunique()
```

Out[47]:

```
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 [48]:

```
# column names
ks.columns
```

Out[48]:

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

In [49]:

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

Out[49]:

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

Assignment given in the video

In [102…

```
#first method
ks[["survived","sex","class"]].nunique()
```

Out[102…

```
survived    2
sex         2
class       3
dtype: int64
```

In [103…

```
#second method to get unique values
desc = ks[["sex","class"]].describe()
desc
```

Out[103…

	sex	class
count	891	891
unique	2	3
top	male	Third
freq	577	491

In [104…

```
#using loc/iloc i can access unique values
desc.loc["unique","sex"]
```

Out[104…

```
2
```

In [105…

```
ks[["sex", "class"]].values.ravel()
```

Out[105…

```
array(['male', 'Third', 'female', ..., 'First', 'male', 'Third'],
      dtype=object)
```

In [ ]:

3rd method and google based search on this task

In [106…

```
column_values = ks[["sex", "class","who"]].values
unique_values =  np.unique(column_values)
unique_values
```

Out[106…

```
array(['First', 'Second', 'Third', 'child', 'female', 'male', 'man',
       'woman'], dtype=object)
```

In [107…

```
# another method using Loop
for col in ks:
    print(ks[col].unique())
```

```
[0 1]
[3 1 2]
['male' 'female']
[22.   38.   26.   35.     nan 54.    2.   27.   14.    4.   58.   20.
 39.   55.   31.   34.   15.   28.    8.   19.   40.   66.   42.   21.
 18.    3.    7.   49.   29.   65.   28.5   5.   11.   45.   17.   32.
 16.   25.    0.83 30.   33.   23.   24.   46.   59.   71.   37.   47.
 14.5  70.5  32.5  12.    9.   36.5  51.   55.5  40.5  44.    1.   61.
 56.   50.   36.   45.5  20.5  62.   41.   52.   63.   23.5   0.92 43.
 60.   10.   64.   13.   48.    0.75 53.   57.   80.   70.   24.5   6.
  0.67 30.5   0.42 34.5  74.  ]
[1 0 3 4 2 5 8]
[0 1 2 5 3 4 6]
[  7.25    71.2833   7.925   53.1      8.05     8.4583  51.8625  21.075
  11.1333  30.0708  16.7     26.55    31.275    7.8542  16.      29.125
  13.      18.       7.225   26.       8.0292  35.5     31.3875 263.
   7.8792   7.8958  27.7208 146.5208   7.75    10.5     82.1708  52.
   7.2292  11.2417   9.475   21.      41.5792  15.5     21.6792  17.8
  39.6875   7.8     76.7292  61.9792  27.75    46.9     80.      83.475
  27.9     15.2458   8.1583   8.6625  73.5     14.4542  56.4958   7.65
  29.      12.475    9.       9.5      7.7875  47.1     15.85    34.375
  61.175   20.575   34.6542  63.3583  23.      77.2875   8.6542   7.775
  24.15     9.825   14.4583 247.5208   7.1417  22.3583   6.975    7.05
  14.5     15.0458  26.2833   9.2167  79.2      6.75    11.5     36.75
   7.7958  12.525   66.6      7.3125  61.3792   7.7333  69.55    16.1
  15.75    20.525   55.      25.925   33.5     30.6958  25.4667  28.7125
   0.      15.05    39.      22.025   50.       8.4042   6.4958  10.4625
  18.7875  31.     113.275   27.      76.2917  90.       9.35    13.5
   7.55    26.25    12.275    7.125   52.5542  20.2125  86.5    512.3292
  79.65   153.4625 135.6333  19.5     29.7     77.9583  20.25    78.85
  91.0792  12.875    8.85   151.55    30.5     23.25    12.35   110.8833
 108.9     24.      56.9292  83.1583 262.375   14.     164.8667 134.5
   6.2375  57.9792  28.5    133.65    15.9      9.225   35.      75.25
  69.3     55.4417 211.5      4.0125 227.525   15.7417   7.7292  12.
 120.      12.65    18.75     6.8583  32.5      7.875   14.4     55.9
   8.1125  81.8583  19.2583  19.9667  89.1042  38.5      7.725   13.7917
   9.8375   7.0458   7.5208  12.2875   9.5875  49.5042  78.2667  15.1
   7.6292  22.525   26.2875  59.4      7.4958  34.0208  93.5    221.7792
```

```
 106.425   49.5     71.       13.8625   7.8292  39.6      17.4      51.4792
  26.3875  30.      40.125     8.7125  15.       33.       42.4      15.55
  65.      32.3208   7.0542    8.4333  25.5875    9.8417   8.1375  10.1708
 211.3375  57.      13.4167    7.7417   9.4833    7.7375   8.3625  23.45
  25.9292   8.6833   8.5167    7.8875  37.0042    6.45     6.95      8.3
   6.4375  39.4     14.1083   13.8583  50.4958    5.       9.8458  10.5167]
['S' 'C' 'Q' nan]
['Third', 'First', 'Second']
Categories (3, object): ['First', 'Second', 'Third']
['man' 'woman' 'child']
[ True False]
[NaN, 'C', 'E', 'G', 'D', 'A', 'B', 'F']
Categories (7, object): ['A', 'B', 'C', 'D', 'E', 'F', 'G']
['Southampton' 'Cherbourg' 'Queenstown' nan]
['no' 'yes']
[False  True]
```

Cleaning and filtering the data

In [108…

```
# find missing values inside
ks.isnull().sum()
```

Out[108…

```
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 [109…

```
# removing missing value column (cleaning data)
ks_clean = ks.drop(['deck'], axis=1)
ks_clean.head()
```

Out[109…

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	embark_town	alive
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Southampton	no
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	Cherbourg	yes
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Southampton	yes
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	Southampton	yes
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Southampton	no

In [110…

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

Out[110…
```
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 [111…
```
ks_clean.shape
```

Out[111…
```
(891, 14)
```

In [113…
```
ks_clean = ks_clean.dropna()
```

In [114…
```
ks_clean.shape
```

Out[114…
```
(712, 14)
```

In [115…
```
ks_clean.isnull().sum()
```

Out[115…
```
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 [117…
```
ks_clean.shape
```

Out[117…
```
(712, 14)
```

In [118…
```
ks.shape
```

Out[118…
```
(891, 15)
```

In [119…
```
ks_clean['sex'].value_counts()
```

Out[119… male 453
female 259
Name: sex, dtype: int64

In [120…
```
ks.describe()
```

Out[120…

	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 [121…
```
ks_clean.describe()
```

Out[121…

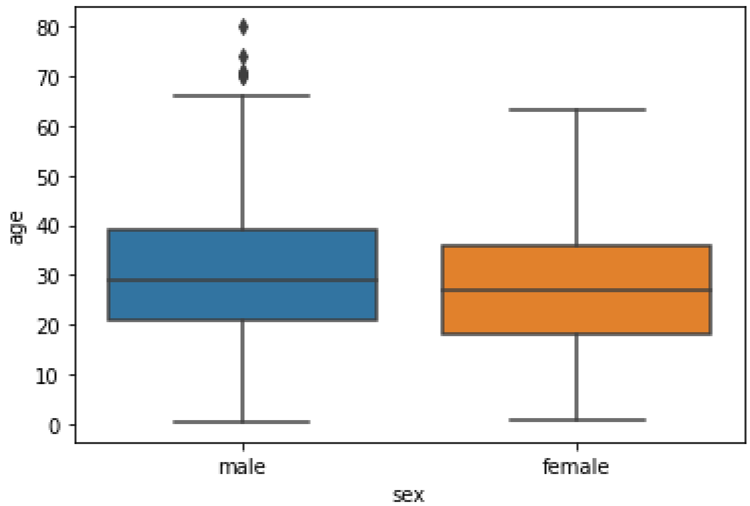
	survived	pclass	age	sibsp	parch	fare
count	712.000000	712.000000	712.000000	712.000000	712.000000	712.000000
mean	0.404494	2.240169	29.642093	0.514045	0.432584	34.567251
std	0.491139	0.836854	14.492933	0.930692	0.854181	52.938648
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	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 [122…
```
ks_clean.columns
```

Out[122…
```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'embark_town', 'alive',
       'alone'],
      dtype='object')
```

In [123…
```
sns.boxplot(x="sex",y='age', data=ks_clean)
```

Out[123… <AxesSubplot:xlabel='sex', ylabel='age'>

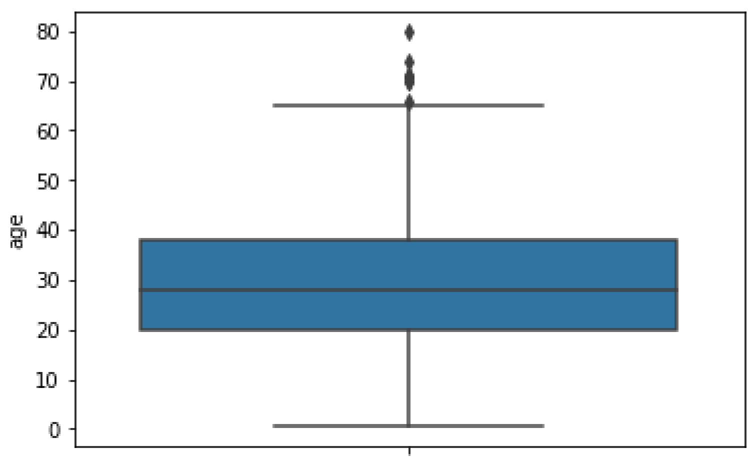


In [124…

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

Out[124…

```
<AxesSubplot:ylabel='age'>
```



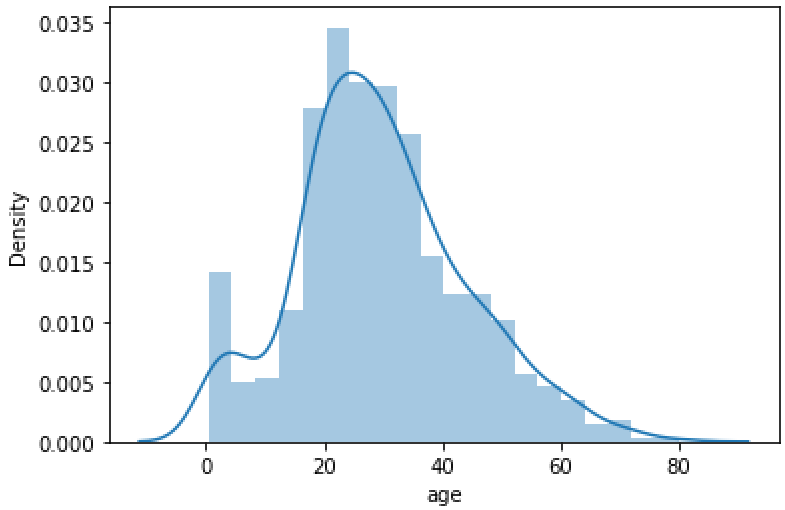
In [125…

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

```
H:\download\Anaconda\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 eithe
r `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level functi
on for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[125…

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



In [126…

```
# outliers removal
ks_clean['age'].mean()
ks_clean.head()
```

Out[126…

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	embark_town	alive
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Southampton	no
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	Cherbourg	yes
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Southampton	yes
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	Southampton	yes
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Southampton	no

In [127…

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

Out[127…

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	embark_town	alive
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Southampton	no
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	Cherbourg	yes
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Southampton	yes
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	Southampton	yes
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Southampton	no

In [128…

```
ks_clean.shape
```

Out[128…

```
(705, 14)
```

In [129…

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

Out[129…

```
29.21797163120567
```

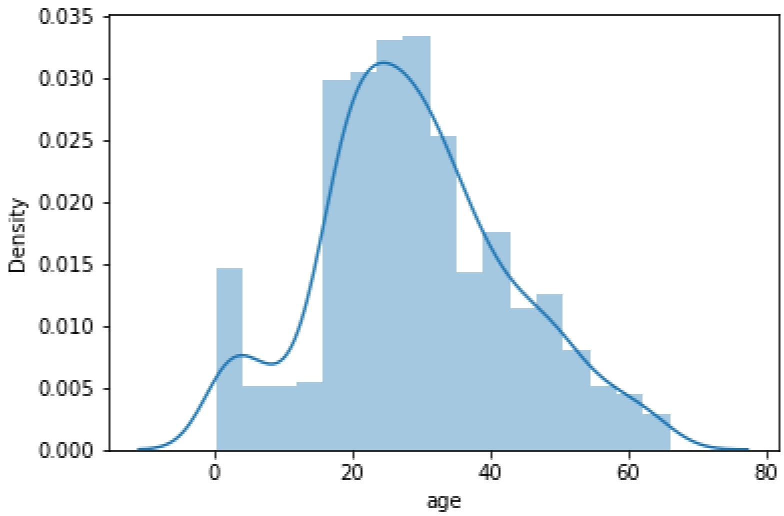
In [131…

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

```
H:\download\Anaconda\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[131…

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

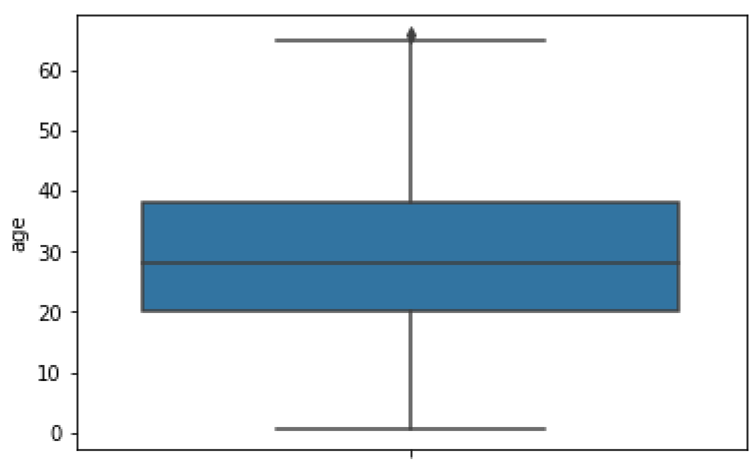


In [132…

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

Out[132…

```
<AxesSubplot:ylabel='age'>
```



In [133…

```
ks_clean.head()
```

Out[133…

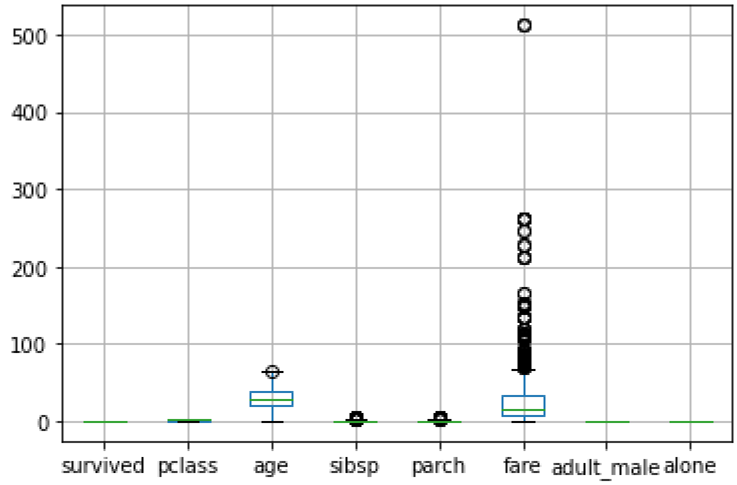
	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	embark_town	alive
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Southampton	no
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	Cherbourg	yes
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Southampton	yes
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	Southampton	yes
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Southampton	no

In [134…

```
ks_clean.boxplot()
```

Out[134…

```
<AxesSubplot:>
```

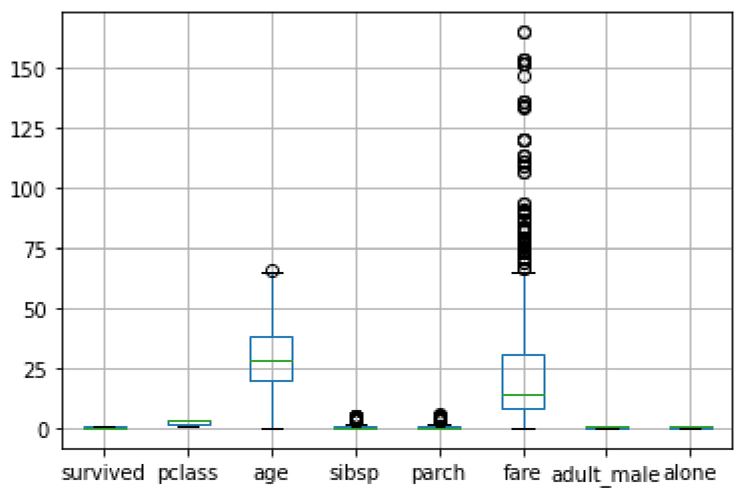


In [135…

```
ks_clean = ks_clean.loc[ks_clean['fare']<200]
ks_clean.boxplot()
```

Out[135…

```
<AxesSubplot:>
```



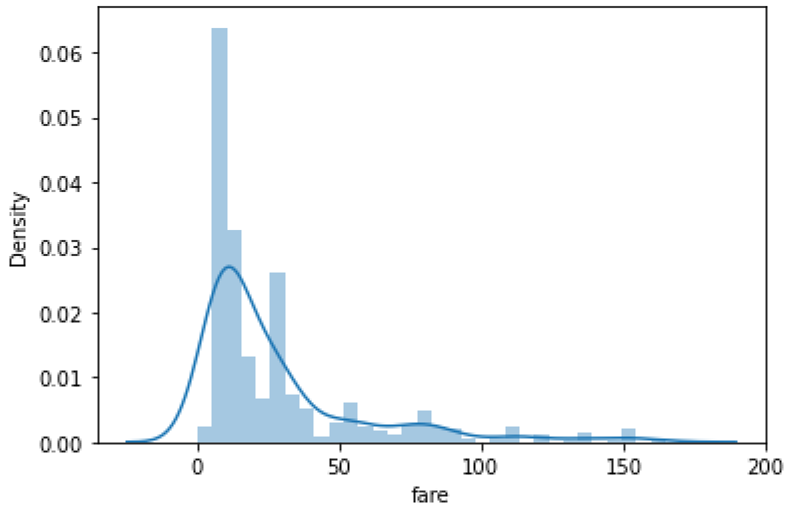
In [136…

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

```
H:\download\Anaconda\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 eithe
r `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level functi
on for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[136…

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



In [137…

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

```
H:\download\Anaconda\lib\site-packages\pandas\core\arraylike.py:364: RuntimeWarning: divide by zer
o encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

Out[137…

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	embark_town	alive
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Southampton	no
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	Cherbourg	yes
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Southampton	yes
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	Southampton	yes
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Southampton	no

In [140…

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

Out[140…

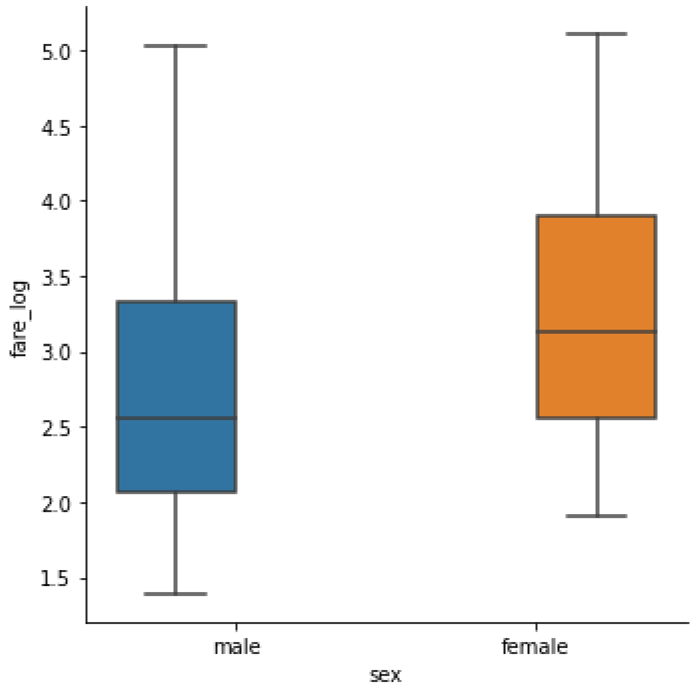
```
survived       0
pclass         0
sex            0
age            0
sibsp          0
parch          0
fare           0
embarked       0
class          0
who            0
adult_male     0
embark_town    0
alive          0
alone          0
fare_log       0
dtype: int64
```

In [141…

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

Out[141…

```
<seaborn.axisgrid.FacetGrid at 0x17257080400>
```

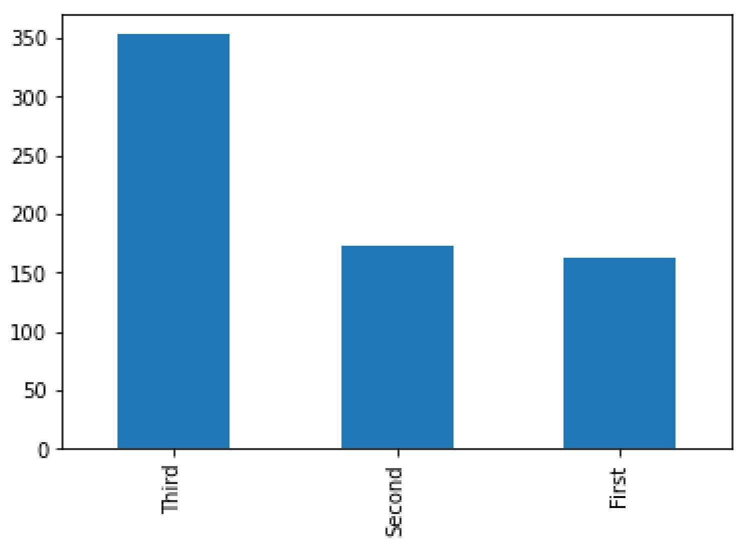


In [143…

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

Out[143…

<AxesSubplot:>



In [144…

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

Out[144…

		survived	pclass	age	sibsp	parch	fare	adult_male	alone	fare_log
sex	**class**									
female	**First**	0.957746	1.0	35.014085	0.492958	0.436620	82.933041	0.000000	0.366197	4.305164
	Second	0.918919	2.0	28.722973	0.500000	0.621622	21.951070	0.000000	0.405405	2.985791
	Third	0.460784	3.0	21.750000	0.823529	0.950980	15.875369	0.000000	0.372549	2.617667
male	**First**	0.406593	1.0	40.356264	0.362637	0.252747	54.841575	0.967033	0.549451	NaN
	Second	0.153061	2.0	30.340102	0.377551	0.244898	21.221429	0.908163	0.632653	2.894890
	Third	0.151394	3.0	26.143108	0.494024	0.258964	12.197757	0.888446	0.737052	NaN

In [145…

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

Out[145…

sex	class	who	survived	pclass	age	sibsp	parch	fare	adult_male	alone
female	First	child	0.666667	1.0	10.333333	0.666667	1.666667	160.962500	0.0	0.000000
		man	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
		woman	0.978022	1.0	35.500000	0.549451	0.417582	104.317995	0.0	0.373626
	Second	child	1.000000	2.0	6.600000	0.700000	1.300000	29.240000	0.0	0.000000
		man	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
		woman	0.909091	2.0	32.179688	0.454545	0.500000	20.868624	0.0	0.484848
	Third	child	0.533333	3.0	7.100000	1.533333	1.100000	19.023753	0.0	0.166667
		man	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
		woman	0.491228	3.0	27.854167	0.728070	0.719298	15.354351	0.0	0.482456
male	First	child	1.000000	1.0	5.306667	0.666667	2.000000	117.802767	0.0	0.000000
		man	0.352941	1.0	42.382653	0.302521	0.235294	65.951086	1.0	0.630252
		woman	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Second	child	1.000000	2.0	2.258889	0.888889	1.222222	27.306022	0.0	0.000000
		man	0.080808	2.0	33.588889	0.292929	0.131313	19.054124	1.0	0.727273
		woman	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Third	child	0.321429	3.0	6.515000	2.821429	1.321429	27.716371	0.0	0.035714
		man	0.119122	3.0	28.995556	0.294671	0.128527	11.340213	1.0	0.824451
		woman	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [146…

```
ks.head()
```

Out[146…

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

Relationship

In [147…

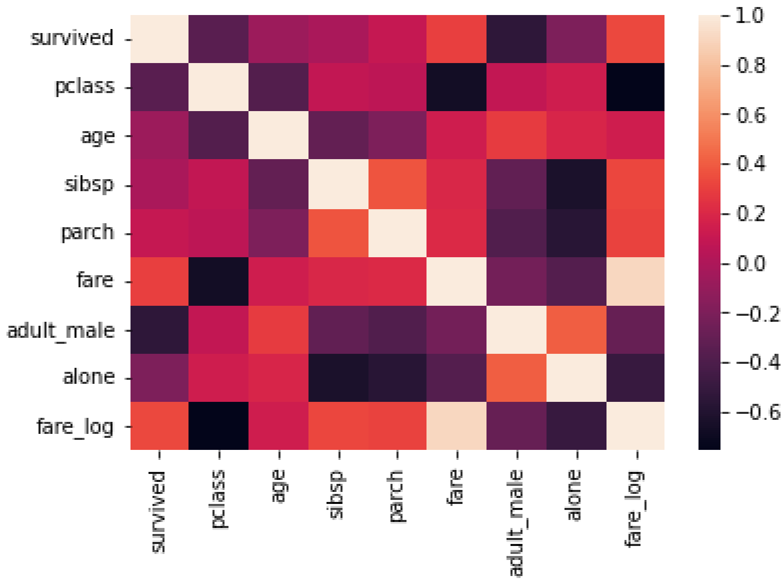
```
corr_ks_clean = ks_clean.corr()
```

In [148…

```
sns.heatmap(corr_ks_clean)
```

Out[148…

```
<AxesSubplot:>
```

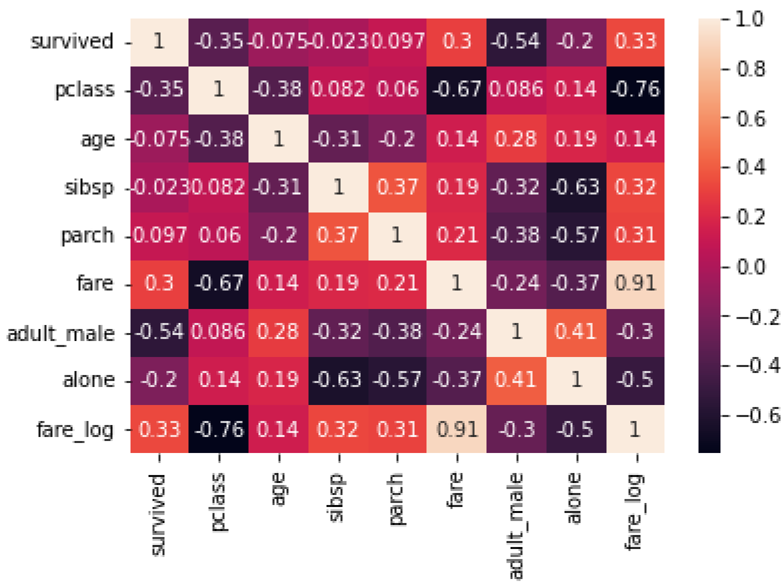


In [149…

```python
sns.heatmap(corr_ks_clean, annot=True)
```

Out[149…

```
<AxesSubplot:>
```

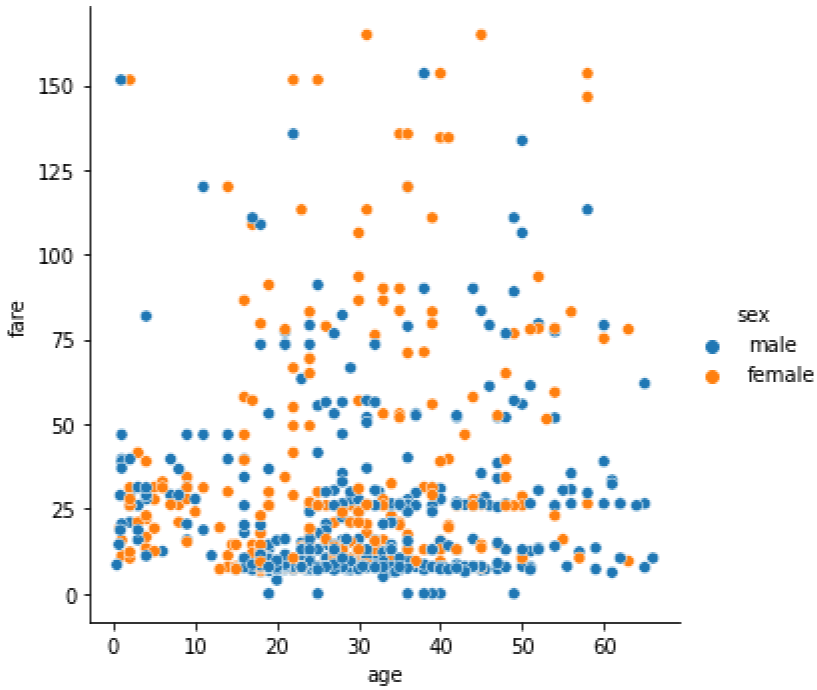


In [150…

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

Out[150…

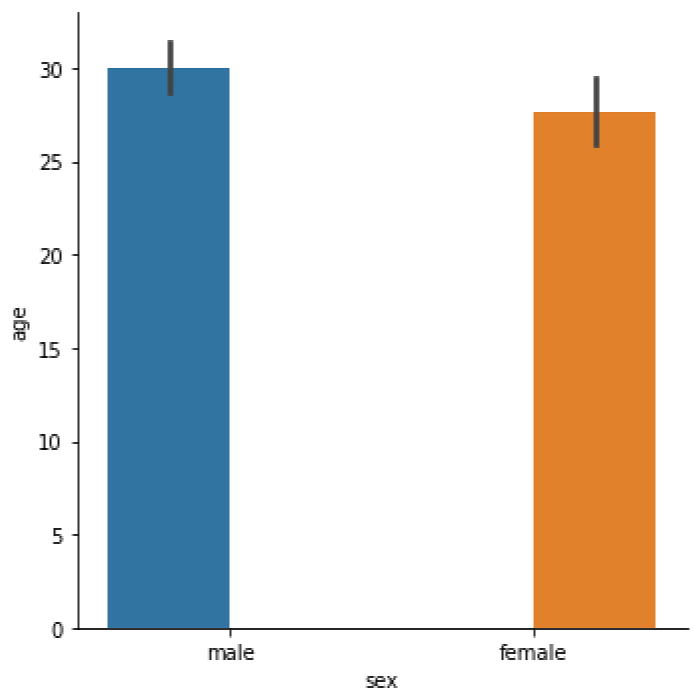
```
<seaborn.axisgrid.FacetGrid at 0x17258565b50>
```



In [157…

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

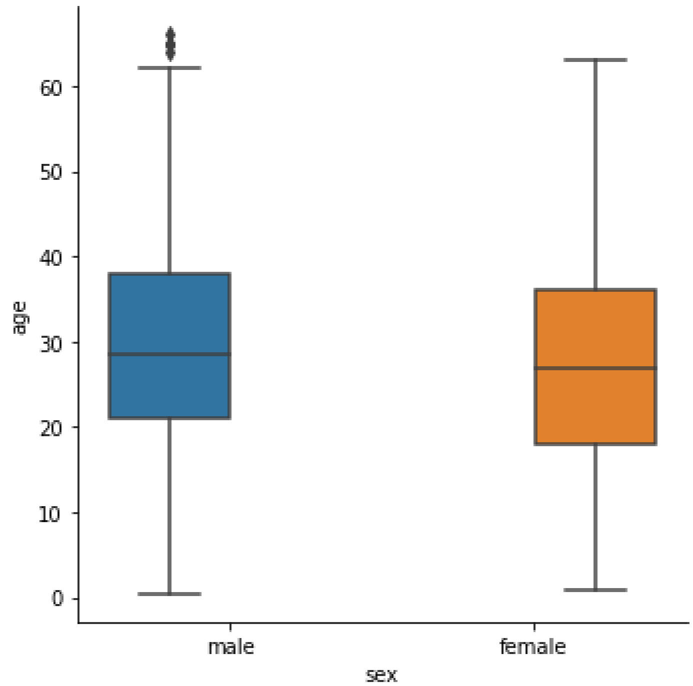
Out[157… <seaborn.axisgrid.FacetGrid at 0x17258790400>



In [158…

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

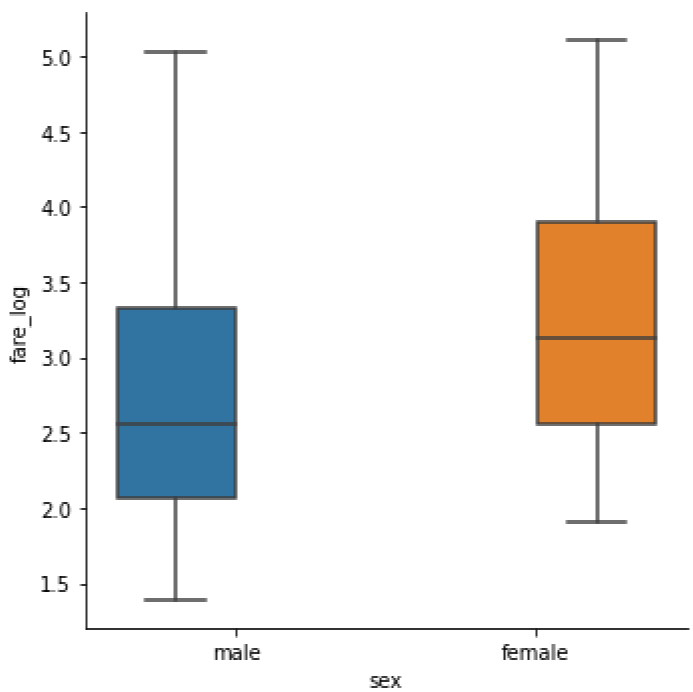
Out[158… <seaborn.axisgrid.FacetGrid at 0x17258800a00>



In [152…

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

Out[152… <seaborn.axisgrid.FacetGrid at 0x172553ca190>

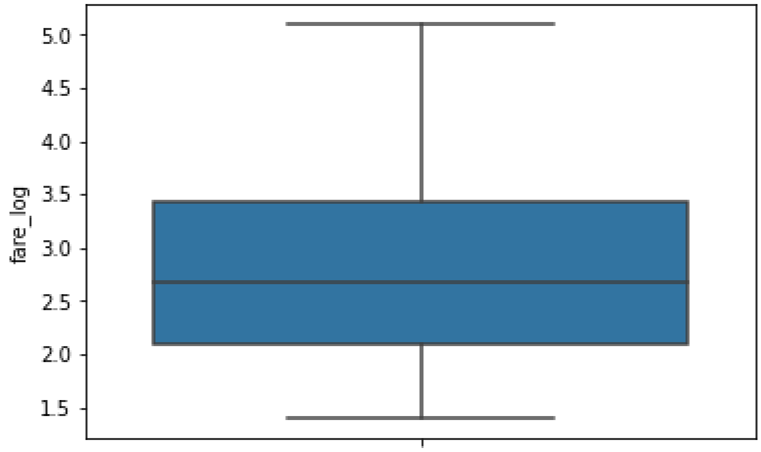


here ap ne video ma kaha ka log transformation ke bad boxplot check kro

In [153…

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

Out[153… <AxesSubplot:ylabel='fare_log'>



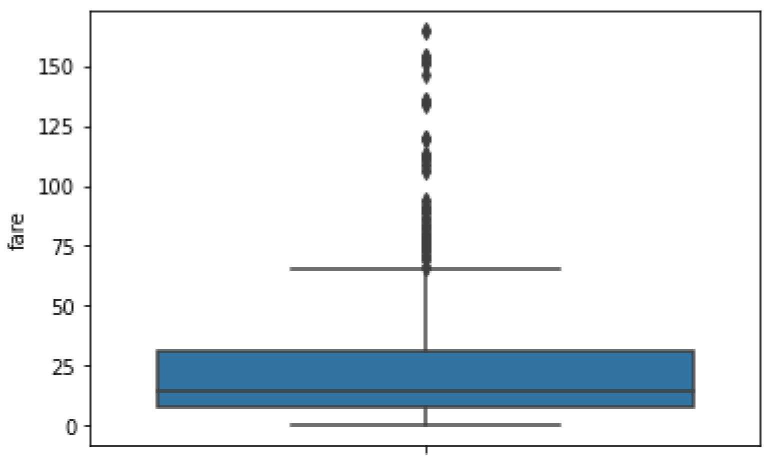
This one is simple and check the difference without log transformation

In [159…

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

Out[159…

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



In [ ]:

In [ ]: