# Participant

Title= "Matric Student"

Name= "Muhammad Bin Saqib Ali"

email = "muhammad.saqib8761@gmail.com"

whatsapp = "00923470159155"

# EXPLORATORY DATA ANALYSIS

## Three important steps to keep in mind are:

1. Understanding the Data
2. Clean the Data
3. Find Relationship between the Data

### Importing Libraries

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

## Importing Data

In [ ]:
```
data = sns.load_dataset ("titanic")
data.head()
```

Out[ ]:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_m |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | T |
| **1** | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | Fa |
| **2** | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | Fa |
| **3** | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | Fa |
| **4** | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | T |

### Checking data from bottom

In [ ]:
```
data.tail (5)
```

Out[ ]:

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

## To save Data in our Folder

In [ ]:
```
data.to_csv ('data.csv')
```

## It will give Information of our of dataframe. e.g: Column, Non-Null Count, Dtype

In [ ]:
```
data.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
```

## To check data from **Top**

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

Out[ ]:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_m |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | T |
| **1** | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | Fa |
| **2** | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | Fa |
| **3** | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | Fa |
| **4** | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | T |

## To number of Rows & Columns

In [ ]:
```
data.shape
```

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

## This will give Summary of our data

In [ ]:
```
data.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 |

## It will give the **Unique** value number with Column name

In [ ]:
```
data.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
```

## To check Columns name

In [ ]:
```
data.columns
```

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

## To check specific Column unique value

In [ ]:
```
data ['who'].unique ()
```

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

## By this we can check the numbers of null value of Column

In [ ]:
```
data.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
```

## To remove Column

In [ ]:
```
new_data=data.drop (["deck"], axis=1)
new_data.head ()
```

Out[ ]:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_m |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | T |
| **1** | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | Fa |
| **2** | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | Fa |
| **3** | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | Fa |
| **4** | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | T |

## By this we can check the numbers of null value of Column

In [ ]:
```
new_data.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
```

## To Check number of Rows and Columns

In [ ]:
```
new_data.shape
```

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

## By this we can remove **All** null value from our Dataframe

In [ ]:
```
new_data = new_data.dropna()
```

## By this we can check the numbers of null value of Column

In [ ]:
```
new_data.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
```

## Checking new data Rows and Columns

In [ ]:
```
new_data.shape
```

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

## Checking old data Rows and Columns

In [ ]:
```
data.shape
```

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

## By this we can check the number of values repeated in Columns

In [ ]:
```
new_data["who"].value_counts()
```

Out[ ]:
```
man      413
woman    216
child     83
Name: who, dtype: int64
```

## Old data Summary of our Dataframe

In [ ]:
```
data.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 |

## New data Summary of our Dataframe

In [ ]:
```
new_data.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 |

## To check name of columns

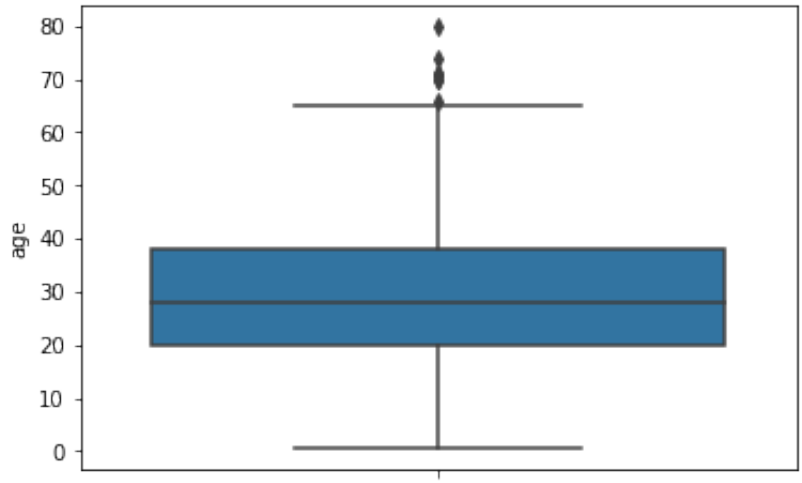
In [ ]:
```
new_data.columns
```

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

## Creating Box Plot

In [ ]:
```
sns.boxplot(y="age", data=new_data)
```

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

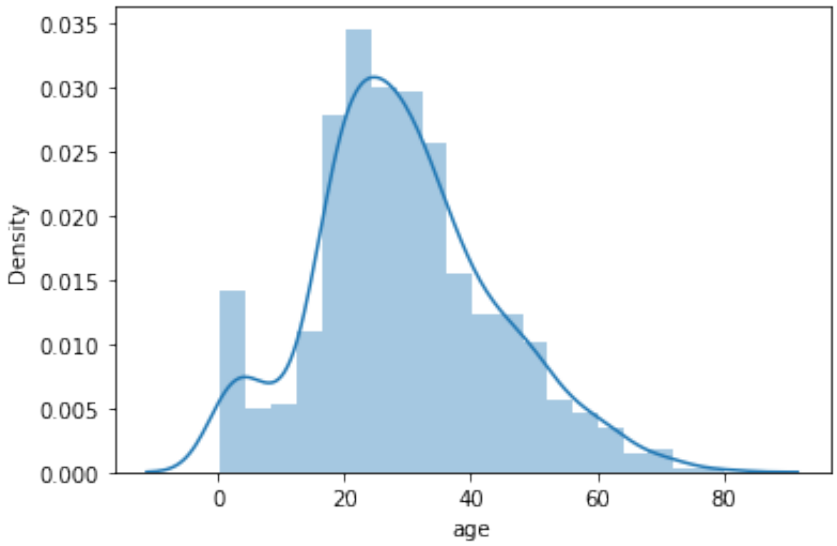



## Creating DisPlot

In [ ]:
```
sns.distplot (new_data["age"])
```

```
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/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[ ]:
```
<AxesSubplot:xlabel='age', ylabel='Density'>
```




## Checking mean of our Specific Column

In [ ]:
```
new_data["age"].mean ()
```

Out[ ]:
```
29.64209269662921
```

## Removing values above than 62 in specific Column

In [ ]:
```
new_data=new_data[new_data['age']<62]
new_data.head ()
```

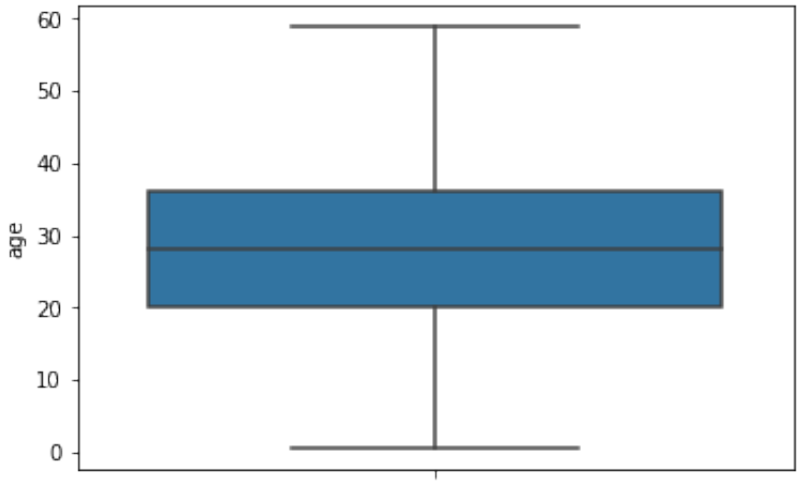
Out[ ]:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_m |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | T |
| **1** | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | Fa |
| **2** | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | Fa |
| **3** | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | Fa |
| **4** | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | T |

## Creating Box Plot

In [ ]:
```
sns.boxplot(y="age", data=new_data)
```

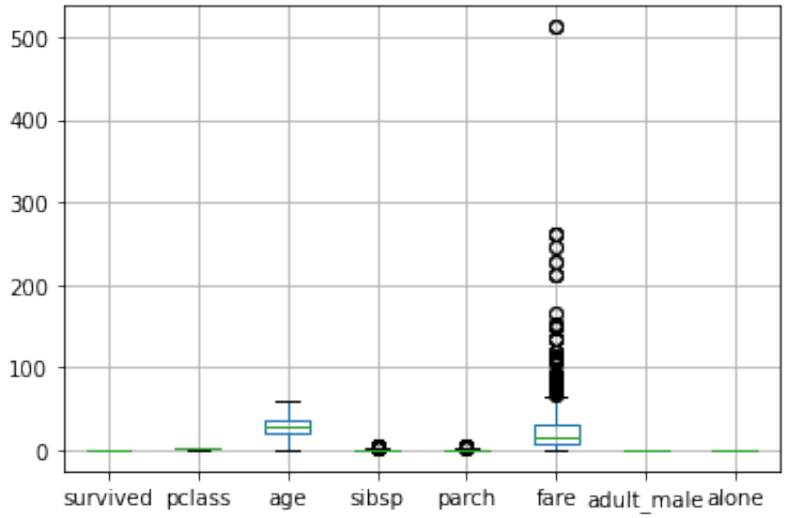
Out[ ]:
```
<AxesSubplot:ylabel='age'>
```




## Creating Box Plot

In [ ]:
```
new_data.boxplot ()
```

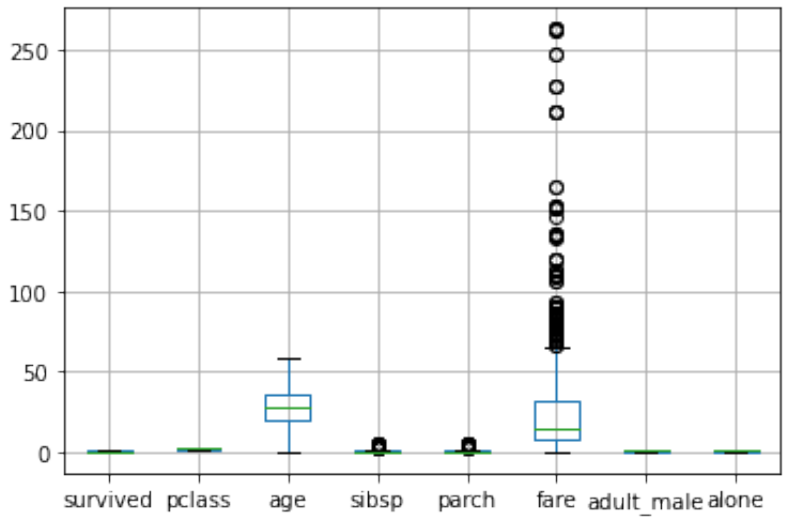
Out[ ]: <AxesSubplot:>




## Removing values above than 270 in specific Column

In [ ]:
```
new_data=new_data[new_data['fare']<270]
new_data.boxplot()
```

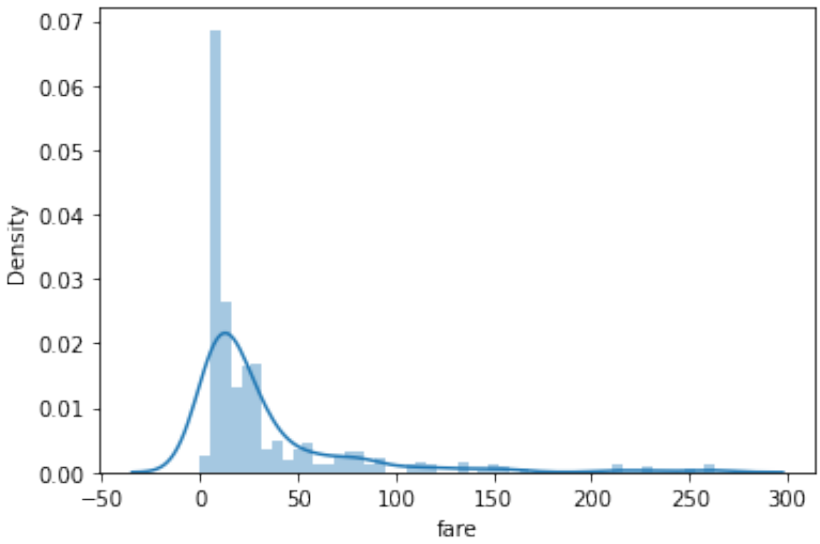
Out[ ]: <AxesSubplot:>




## Creating Displot and Adding New column of fare_log and taking Log of Specific Column in Dataframe

In [ ]:
```
sns.distplot (new_data["fare"])
new_data["fare_log"]= np.log(new_data["fare"])
```

```
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pack
ages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecat
ed 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)
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pack
ages/pandas/core/arraylike.py:364: RuntimeWarning: divide by zero encounter
ed in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```




## To check Top rows of dataframe

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

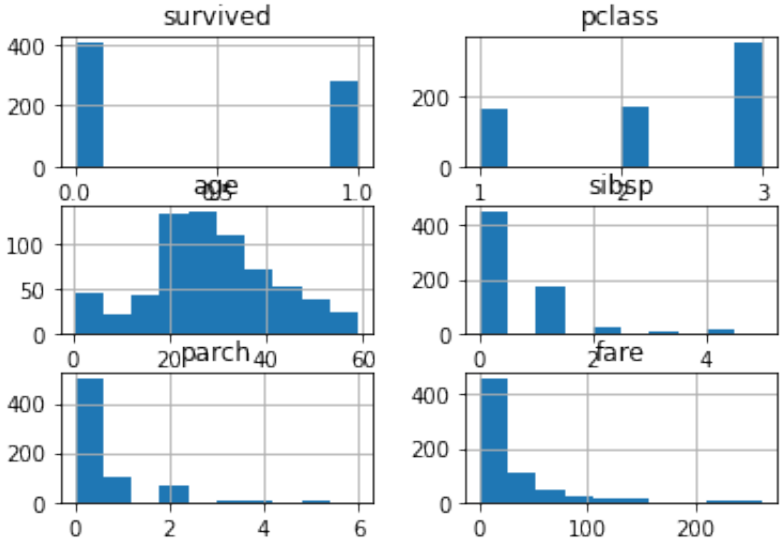
Out[ ]:

| | survived | pclass | sex | age | sibsp | parch | fare | embarked | class | who | adult_m |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | male | 22.0 | 1 | 0 | 7.2500 | S | Third | man | T |
| **1** | 1 | 1 | female | 38.0 | 1 | 0 | 71.2833 | C | First | woman | Fa |
| **2** | 1 | 3 | female | 26.0 | 0 | 0 | 7.9250 | S | Third | woman | Fa |
| **3** | 1 | 1 | female | 35.0 | 1 | 0 | 53.1000 | S | First | woman | Fa |
| **4** | 0 | 3 | male | 35.0 | 0 | 0 | 8.0500 | S | Third | man | T |

## creating histplot

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

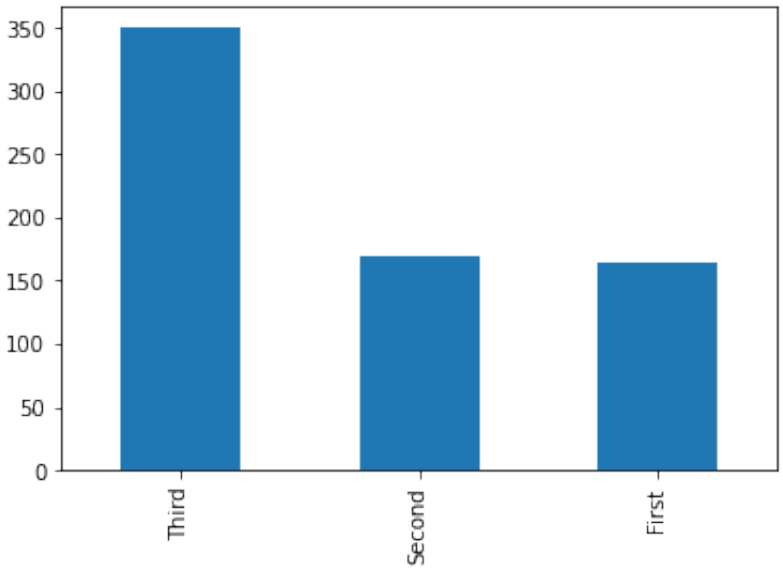
Out[ ]:
```
array([[<AxesSubplot:title={'center':'survived'}>,
        <AxesSubplot:title={'center':'pclass'}>],
       [<AxesSubplot:title={'center':'age'}>,
        <AxesSubplot:title={'center':'sibsp'}>],
       [<AxesSubplot:title={'center':'parch'}>,
        <AxesSubplot:title={'center':'fare'}>]],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




## Counting a value of Specific Column and making Bar plot In 1 Statement

In [ ]:
```
pd.value_counts(new_data["class"]).plot.bar ()
```

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




## Grouping 2 Columns of new data and taking mean of them

In [ ]:
```
new_data.groupby(["sex", "class"]).mean ()
```

Out[ ]:

| sex | class | survived | pclass | age | sibsp | parch | fare | adult_male |
|---|---|---|---|---|---|---|---|---|
| female | First | 0.962500 | 1.0 | 33.550000 | 0.550000 | 0.525000 | 104.373699 | 0.000000 |
| | Second | 0.918919 | 2.0 | 28.722973 | 0.500000 | 0.621622 | 21.951070 | 0.000000 |
| | Third | 0.455446 | 3.0 | 21.341584 | 0.831683 | 0.960396 | 15.937625 | 0.000000 |
| male | First | 0.423529 | 1.0 | 37.440235 | 0.411765 | 0.305882 | 63.216519 | 0.964706 |
| | Second | 0.147368 | 2.0 | 29.319263 | 0.378947 | 0.242105 | 21.260000 | 0.905263 |
| | Third | 0.152610 | 3.0 | 25.847068 | 0.497992 | 0.261044 | 12.239556 | 0.887550 |

## Grouping 2 Columns of old data and taking mean of them

In [ ]:
```
data.groupby(["sex", "class"]).mean ()
```

Out[ ]:

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

## CO Relation of Dataframe ( If it's value is 1 It's means that it is directly proportional and if it's in - soo it will inversly proportional)

In [ ]:
```
cor_new_data= new_data.corr ()
cor_new_data.head ()
```

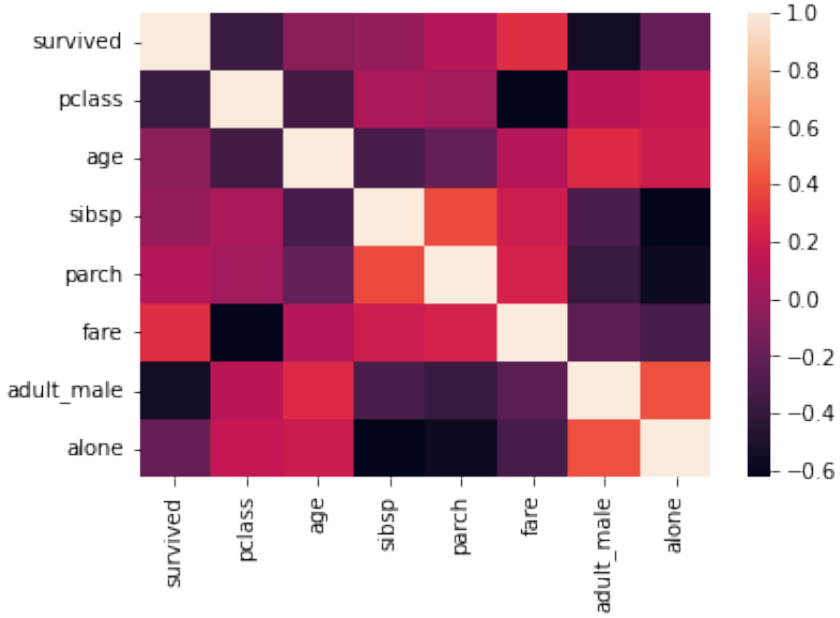
Out[ ]:

| | survived | pclass | age | sibsp | parch | fare | adult_male | |
|---|---|---|---|---|---|---|---|---|
| survived | 1.000000 | -0.376913 | -0.062820 | -0.021580 | 0.101012 | 0.284657 | -0.550647 | -( |
| pclass | -0.376913 | 1.000000 | -0.342623 | 0.059466 | 0.027224 | -0.626093 | 0.120975 | ( |
| age | -0.062820 | -0.342623 | 1.000000 | -0.318082 | -0.202076 | 0.091596 | 0.265445 | ( |
| sibsp | -0.021580 | 0.059466 | -0.318082 | 1.000000 | 0.381742 | 0.195031 | -0.309017 | - |
| parch | 0.101012 | 0.027224 | -0.202076 | 0.381742 | 1.000000 | 0.234899 | -0.379839 | -( |

## Heatmap use for Checking CO Relation

In [ ]:
```
sns.heatmap (cor_new_data)
```

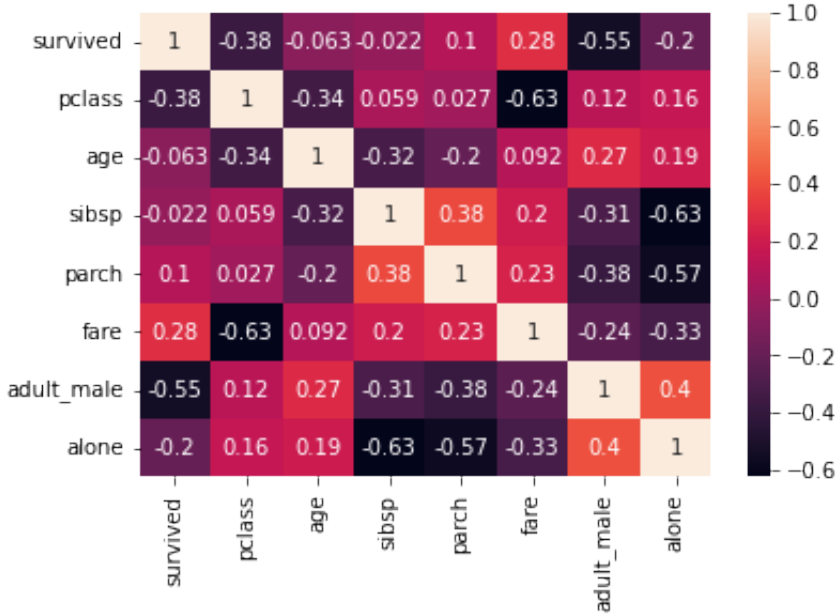
Out[ ]:
```
<AxesSubplot:>
```




## Heatmap use for Checking CO Relation with number

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

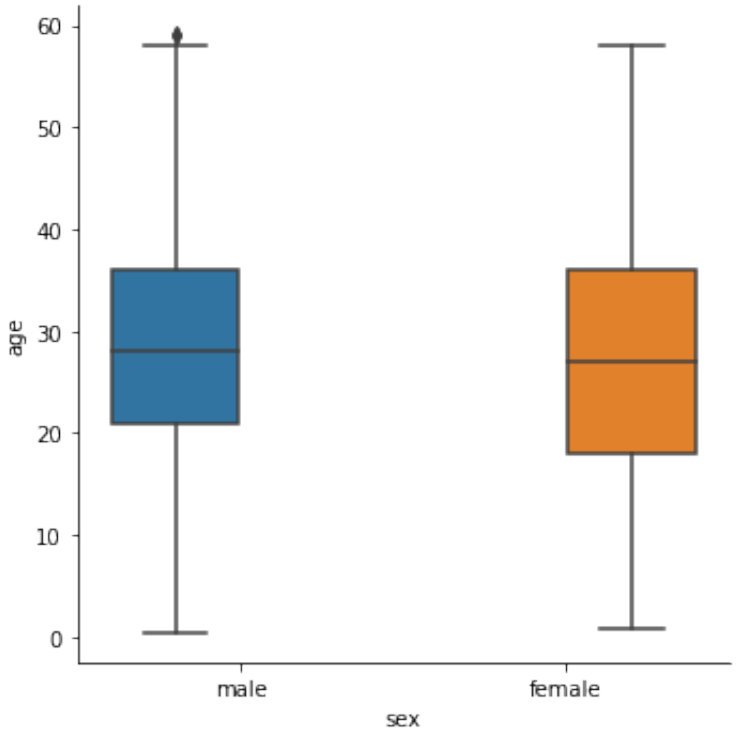
Out[ ]:
```
<AxesSubplot:>
```




## Creating catplot

In [ ]:
```
sns.catplot (x="sex",y="age",hue="sex",data=new_data, kind="box")
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x12b77c040>
```



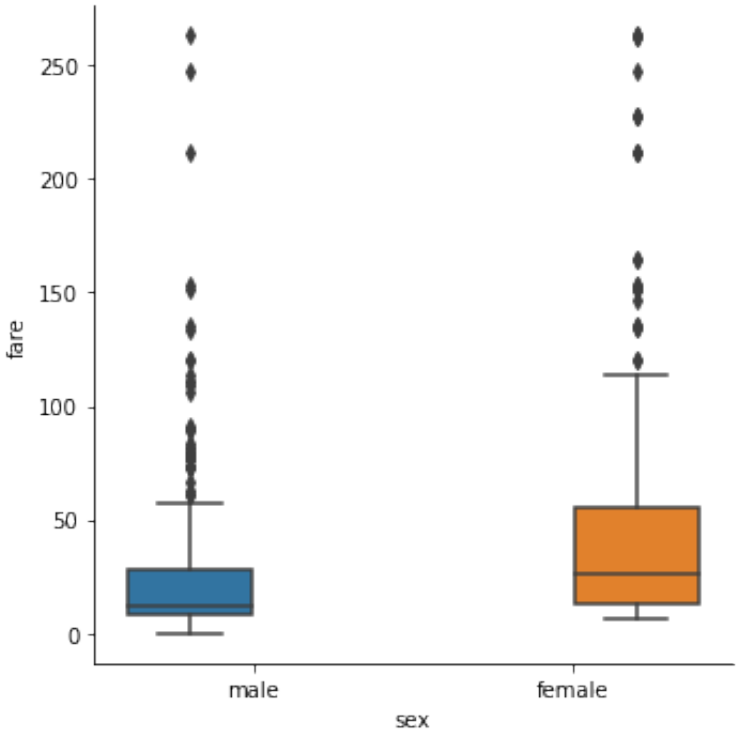

In [ ]:
```
new_data.columns
```

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

## Creating catplot

In [ ]:
```
sns.catplot (x="sex",y="fare",hue="sex",data=new_data, kind="box")
```

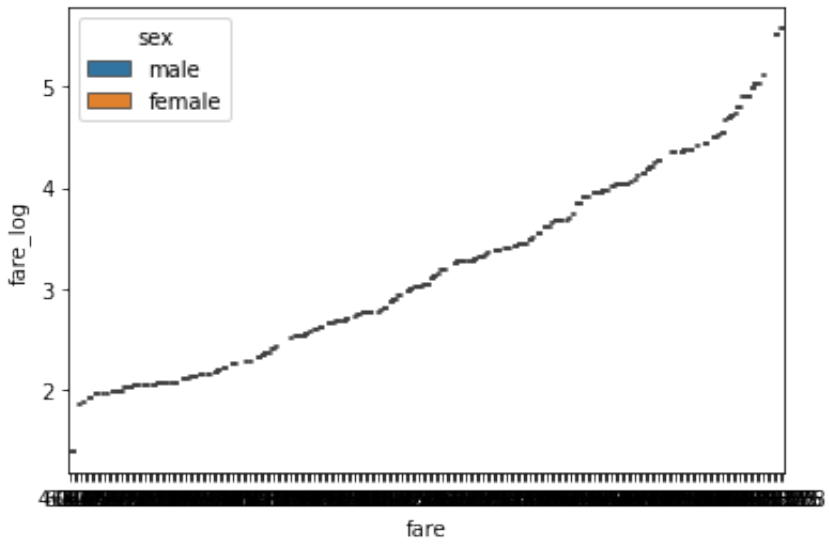
Out[ ]: <seaborn.axisgrid.FacetGrid at 0x13130d300>




## Box Plot with Fare_log Column

In [ ]:
```
g =sns.boxplot (x="fare",y="fare_log",hue="sex",data=new_data)
```

```
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-pack
ages/numpy/lib/function_base.py:4486: RuntimeWarning: invalid value encount
ered in subtract
  diff_b_a = subtract(b, a)
```




In [ ]: