PALLAV NAG

20BCE2023

Assignment 2

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

In [2]:
```
df=pd.read_csv('titanic.csv')
```

In [3]:
```
df.head()
```

Out[3]:

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

In [4]:
```
df.tail()
```

Out[4]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
886	0	2	male	27.0	0	0	13.00	S	Second	man	True
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
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

In [6]:
```
df.columns
```

Out[6]:
```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'deck', 'embark_town',
       'alive', 'alone'],
      dtype='object')
```

In [7]:
```
df.shape
```

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

In [8]:
```
df.describe()
```

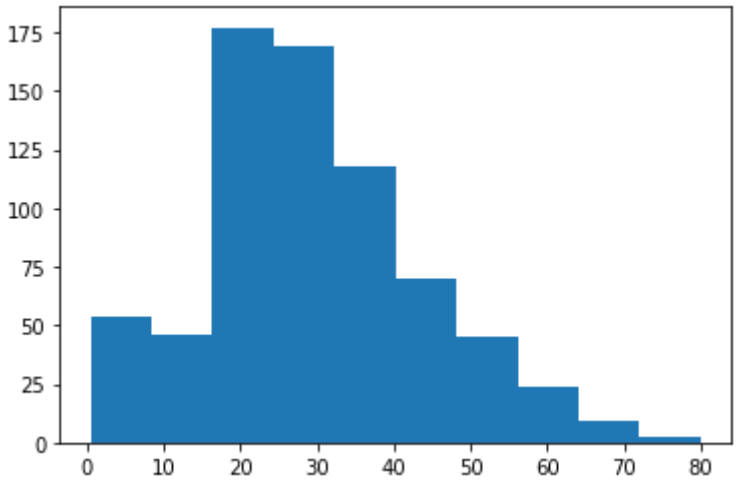
Out[8]:

	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

Univariate analysis

In [9]:
```
plt.hist(df['age'])
```

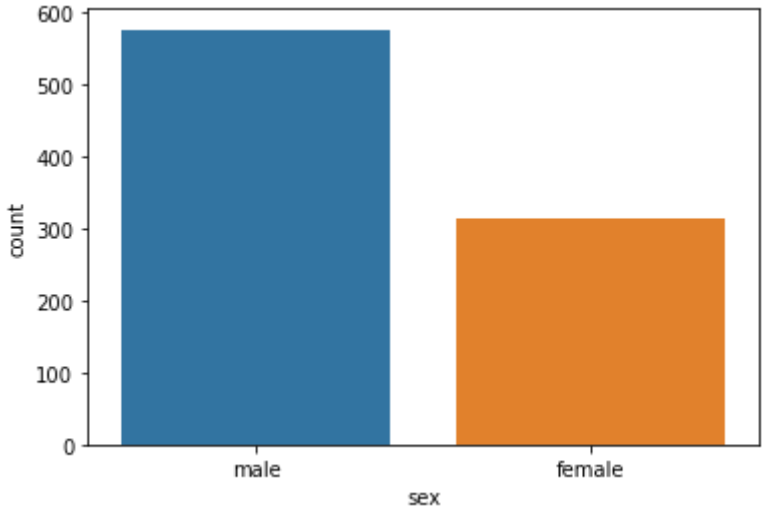
Out[9]:
```
(array([ 54.,  46., 177., 169., 118.,  70.,  45.,  24.,   9.,   2.]),
 array([ 0.42 ,  8.378, 16.336, 24.294, 32.252, 40.21 , 48.168, 56.126,
        64.084, 72.042, 80.   ]),
 <BarContainer object of 10 artists>)
```



In [20]:
```
sns.countplot(df.sex)
```

```
D:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the f
ollowing variable as a keyword arg: x. From version 0.12, the only valid positiona
l argument will be `data`, and passing other arguments without an explicit keyword
will result in an error or misinterpretation.
  warnings.warn(
```

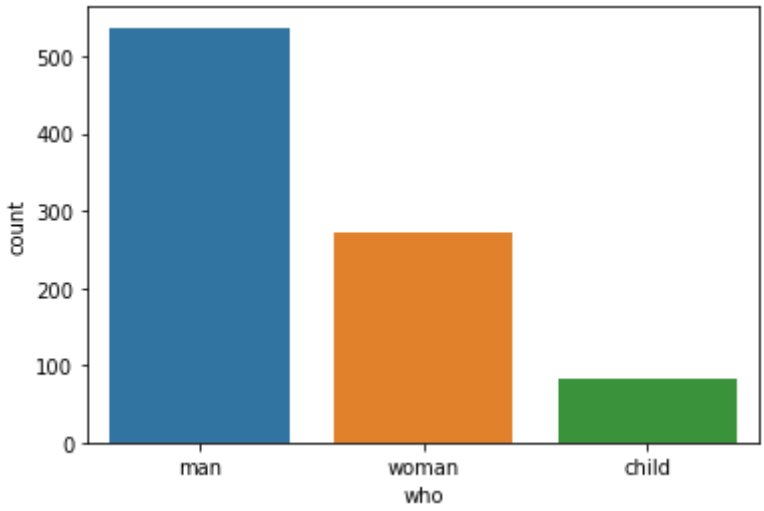
Out[20]:
```
<AxesSubplot:xlabel='sex', ylabel='count'>
```



In [40]:
```
sns.countplot(df.who)
```

```
D:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the f
ollowing variable as a keyword arg: x. From version 0.12, the only valid positiona
l argument will be `data`, and passing other arguments without an explicit keyword
will result in an error or misinterpretation.
  warnings.warn(
```

Out[40]:
```
<AxesSubplot:xlabel='who', ylabel='count'>
```

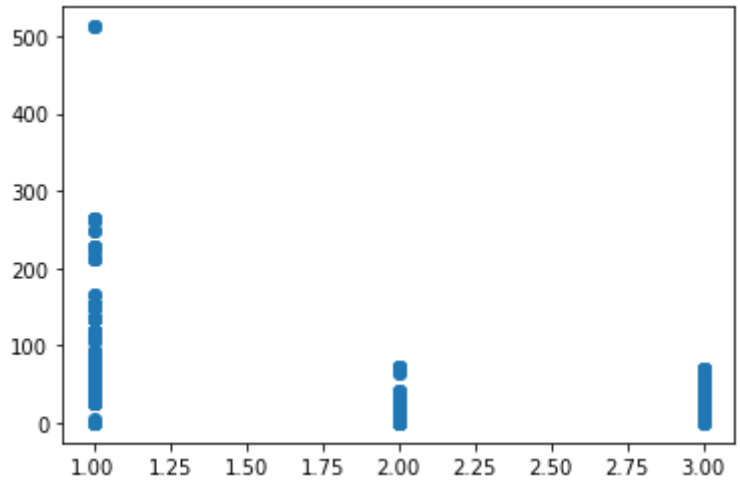


Bivariate analysis

Fares paid according to class booked

In [18]:
```
plt.scatter(df['pclass'],df['fare'])
```

Out[18]:
```
<matplotlib.collections.PathCollection at 0x17cd8fad430>
```



In [27]:
```
df.groupby(by='survived').agg('mean')[['pclass', 'age']]
```

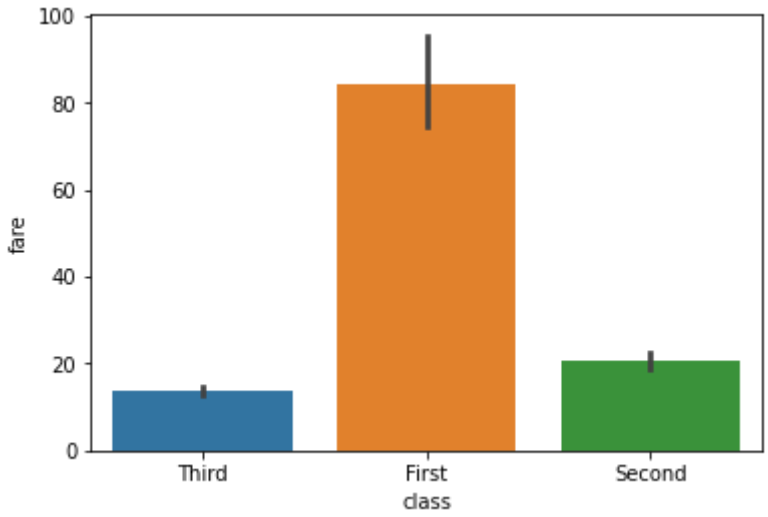
Out[27]:

survived	pclass	age
0	2.531876	30.626179
1	1.950292	28.343690

In [29]:
```
sns.barplot('class','fare',data=df)
```

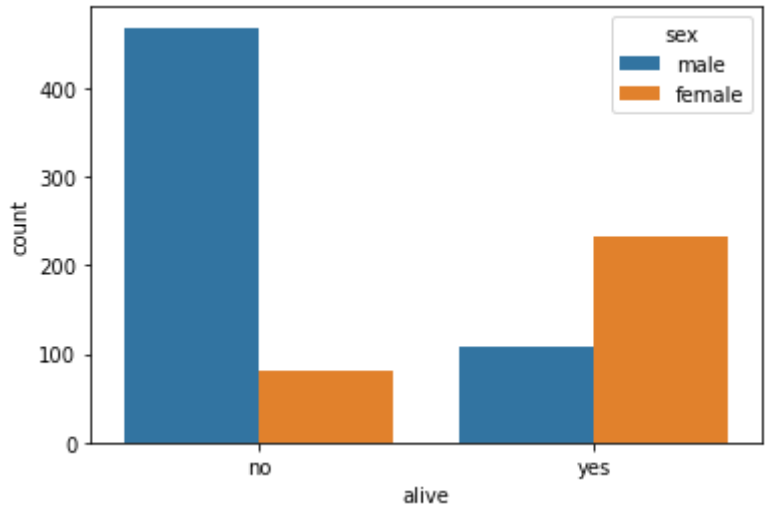
```
D:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the f
ollowing variables as keyword args: x, y. From version 0.12, the only valid positi
onal argument will be `data`, and passing other arguments without an explicit keyw
ord will result in an error or misinterpretation.
  warnings.warn(
```

Out[29]:
```
<AxesSubplot:xlabel='class', ylabel='fare'>
```



In [41]:
```
sns.countplot(data=df,x='alive',hue='sex')
```

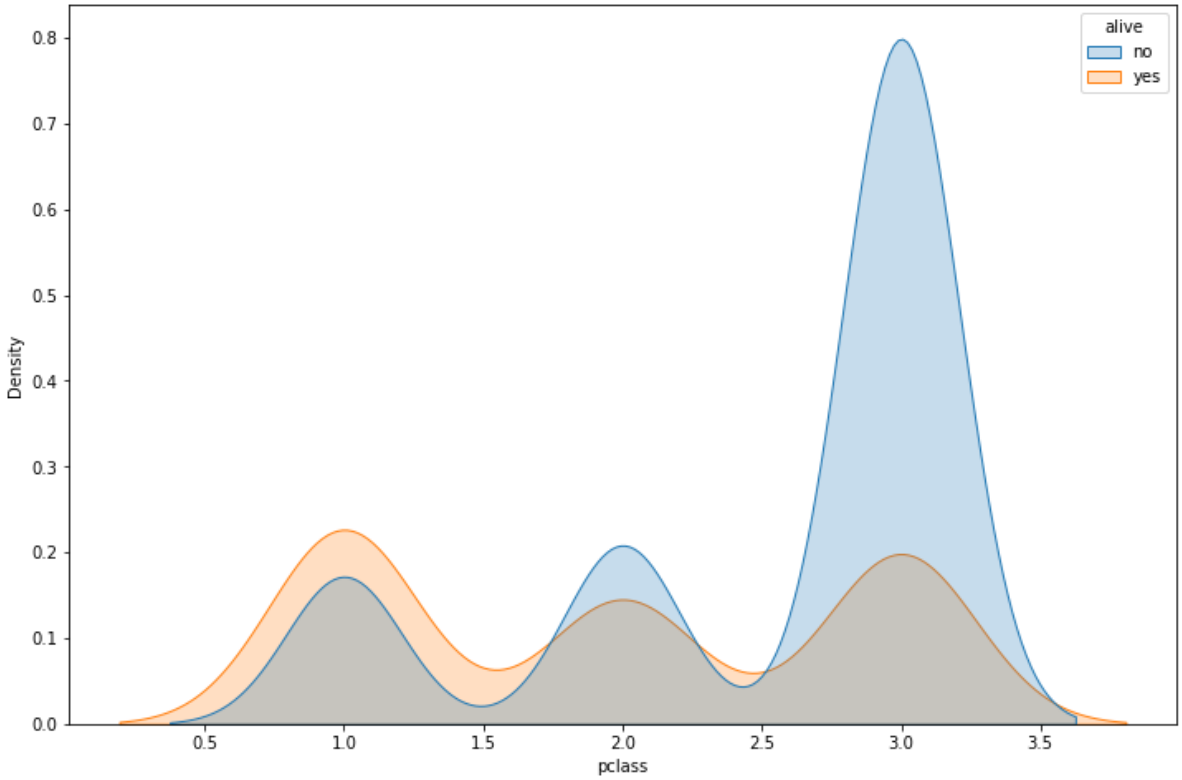
Out[41]:
```
<AxesSubplot:xlabel='alive', ylabel='count'>
```



In [42]:
```
plt.figure(figsize=(12,8))

sns.kdeplot(data=df, x= 'pclass',hue='alive',fill=True)
```

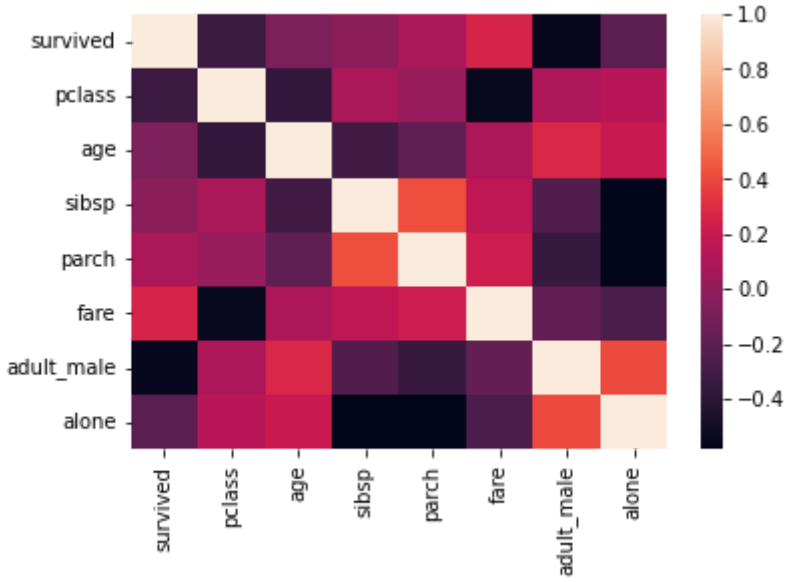
Out[42]:
```
<AxesSubplot:xlabel='pclass', ylabel='Density'>
```



multivariate

In [30]:
```
hm=df.corr()
sns.heatmap(hm)
```

Out[30]:
```
<AxesSubplot:>
```



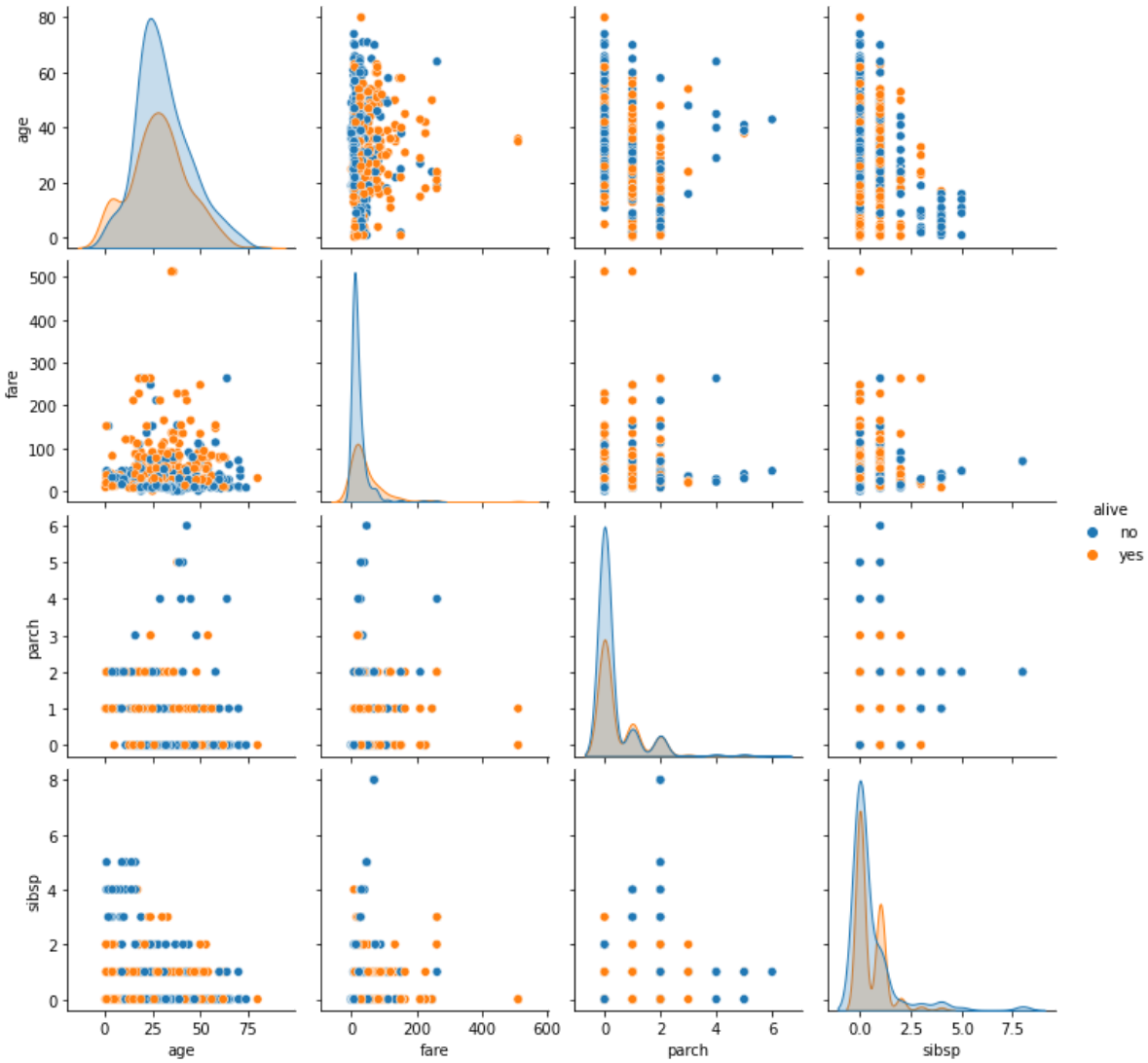
In [31]:
```
hm
```

Out[31]:

	survived	pclass	age	sibsp	parch	fare	adult_male	alone
survived	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307	-0.557080	-0.203367
pclass	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500	0.094035	0.135207
age	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067	0.280328	0.198270
sibsp	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651	-0.253586	-0.584471
parch	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225	-0.349943	-0.583398
fare	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000	-0.182024	-0.271832
adult_male	-0.557080	0.094035	0.280328	-0.253586	-0.349943	-0.182024	1.000000	0.404744
alone	-0.203367	0.135207	0.198270	-0.584471	-0.583398	-0.271832	0.404744	1.000000

In [43]:
```
sns.pairplot(data=df[['age', 'fare', 'parch','sibsp','alive']],hue='alive')
```

Out[43]:
```
<seaborn.axisgrid.PairGrid at 0x17ce2a26970>
```



descriptive statistics

In [32]:
```
df.mean()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_12960\3698961737.py:1: FutureWarning: Dro
pping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is de
precated; in a future version this will raise TypeError.  Select only valid column
s before calling the reduction.
  df.mean()
```

Out[32]:
```
survived       0.383838
pclass         2.308642
age           29.699118
sibsp          0.523008
parch          0.381594
fare          32.204208
adult_male     0.602694
alone          0.602694
dtype: float64
```

In [33]:
```
df.median()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_12960\530051474.py:1: FutureWarning: Drop
ping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is dep
recated; in a future version this will raise TypeError.  Select only valid columns
before calling the reduction.
  df.median()
```

Out[33]:
```
survived       0.0000
pclass         3.0000
age           28.0000
sibsp          0.0000
parch          0.0000
fare          14.4542
adult_male     1.0000
alone          1.0000
dtype: float64
```

In [34]:
```
df.mode()
```

Out[34]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	em
0	0	3	male	24.0	0	0	8.05	S	Third	man	True	C	So

In [35]:
```
df.skew()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_12960\1665899112.py:1: FutureWarning: Dro
pping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is de
precated; in a future version this will raise TypeError.  Select only valid column
s before calling the reduction.
  df.skew()
```

Out[35]:
```
survived      0.478523
pclass       -0.630548
age           0.389108
sibsp         3.695352
parch         2.749117
fare          4.787317
adult_male   -0.420431
alone        -0.420431
dtype: float64
```

In [36]:
```
df.kurt()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_12960\1257127604.py:1: FutureWarning: Dro
pping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is de
precated; in a future version this will raise TypeError.  Select only valid column
s before calling the reduction.
  df.kurt()
```

Out[36]:
```
survived      -1.775005
pclass        -1.280015
age            0.178274
sibsp         17.880420
parch          9.778125
fare          33.398141
adult_male    -1.827345
alone         -1.827345
dtype: float64
```

In [37]:
```
df.max()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_12960\1151452817.py:1: FutureWarning: Dro
pping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is de
precated; in a future version this will raise TypeError.  Select only valid column
s before calling the reduction.
  df.max()
```

Out[37]:
```
survived             1
pclass               3
sex               male
age               80.0
sibsp                8
parch                6
fare          512.3292
class            Third
who              woman
adult_male        True
alive              yes
alone             True
dtype: object
```

In [38]:
```
df.min()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_12960\3962516015.py:1: FutureWarning: Dro
pping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is de
precated; in a future version this will raise TypeError.  Select only valid column
s before calling the reduction.
  df.min()
```

Out[38]:
```
survived           0
pclass             1
sex           female
age             0.42
sibsp              0
parch              0
fare             0.0
class          First
who            child
adult_male     False
alive             no
alone          False
dtype: object
```

Handeling missing data

In [45]:
```
df.isnull().sum()
```

Out[45]:
```
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 [46]:
```
df1 = pd.read_csv('titanic.csv')
df2= pd.read_csv('titanic.csv')
df3 = pd.read_csv('titanic.csv')
df4= pd.read_csv('titanic.csv')
df5=pd.read_csv('titanic.csv')
```

In [47]:
```
#METHOD:1 DROP ALL ROWS WITH MISSING VALUE
df1.dropna(inplace= True)
df1.isnull().sum()
```

Out[47]:
```
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
```

In [48]:
```
df1
```

Out[48]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False
6	0	1	male	54.0	0	0	51.8625	S	First	man	True
10	1	3	female	4.0	1	1	16.7000	S	Third	child	False
11	1	1	female	58.0	0	0	26.5500	S	First	woman	False
...	...	...	...	...	...	...	...	...	...	...	...
871	1	1	female	47.0	1	1	52.5542	S	First	woman	False
872	0	1	male	33.0	0	0	5.0000	S	First	man	True
879	1	1	female	56.0	0	1	83.1583	C	First	woman	False
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

182 rows × 15 columns

In [49]:
```
df2['age'].replace(np.NaN,df2['age'].mean()).head(30)
```

Out[49]:
```
0     22.000000
1     38.000000
2     26.000000
3     35.000000
4     35.000000
5     29.699118
6     54.000000
7      2.000000
8     27.000000
9     14.000000
10     4.000000
11    58.000000
12    20.000000
13    39.000000
14    14.000000
15    55.000000
16     2.000000
17    29.699118
18    31.000000
19    29.699118
20    35.000000
21    34.000000
22    15.000000
23    28.000000
24     8.000000
25    38.000000
26    29.699118
27    19.000000
28    29.699118
29    29.699118
Name: age, dtype: float64
```

In [51]:
```
df3['age'].replace(np.NaN,df3['age'].median()).head(30)
```

Out[51]:
```
0     22.0
1     38.0
2     26.0
3     35.0
4     35.0
5     28.0
6     54.0
7      2.0
8     27.0
9     14.0
10     4.0
11    58.0
12    20.0
13    39.0
14    14.0
15    55.0
16     2.0
17    28.0
18    31.0
19    28.0
20    35.0
21    34.0
22    15.0
23    28.0
24     8.0
25    38.0
26    28.0
27    19.0
28    28.0
29    28.0
Name: age, dtype: float64
```

In [54]:
```
df4['age'].replace(np.NaN,24).head(30) #mode=24
```

Out[54]:
```
0     22.0
1     38.0
2     26.0
3     35.0
4     35.0
5     24.0
6     54.0
7      2.0
8     27.0
9     14.0
10     4.0
11    58.0
12    20.0
13    39.0
14    14.0
15    55.0
16     2.0
17    24.0
18    31.0
19    24.0
20    35.0
21    34.0
22    15.0
23    28.0
24     8.0
25    38.0
26    24.0
27    19.0
28    24.0
29    24.0
Name: age, dtype: float64
```

In [56]:
```
df['deck']=df3['deck'].fillna('NO')
df['age']=df3['age'].replace(np.NaN,df3['age'].median())
```

In [57]:
```
df
```

Out[57]:

	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
...	...	...	...	...	...	...	...	...	...	...	...
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
888	0	3	female	28.0	1	2	23.4500	S	Third	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

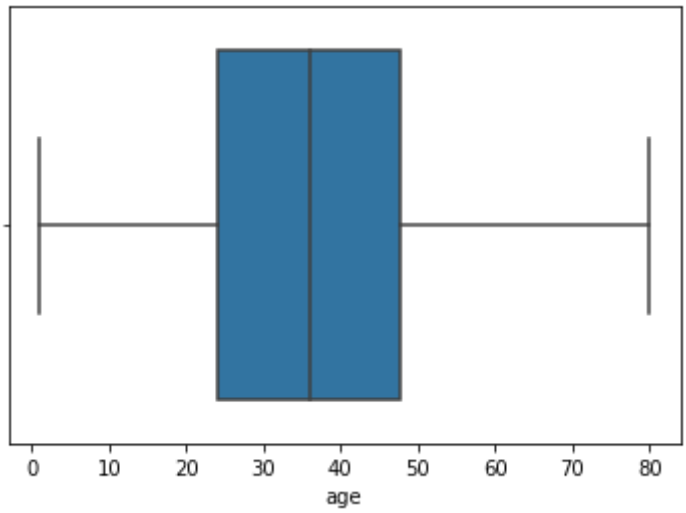
891 rows × 15 columns

Handling Outliers

In [58]:
```
sns.boxplot(df1['age'])
```

```
D:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the f
ollowing variable as a keyword arg: x. From version 0.12, the only valid positiona
l argument will be `data`, and passing other arguments without an explicit keyword
will result in an error or misinterpretation.
  warnings.warn(
```

Out[58]:
```
<AxesSubplot:xlabel='age'>
```



In [59]:
```
np.where(df1['age']>62)
```

Out[59]:
```
(array([  9,  15,  50,  85,  90, 128, 155], dtype=int64),)
```

In [60]:
```
from scipy import stats
z = np.abs(stats.zscore(df['age']))
z
```

Out[60]:
```
0      0.565736
1      0.663861
2      0.258337
3      0.433312
4      0.433312
         ...
886    0.181487
887    0.796286
888    0.104637
889    0.258337
890    0.202762
Name: age, Length: 891, dtype: float64
```

In [61]:
```
np.where(z > 2)
```

Out[61]:
```
(array([  7,  11,  16,  33,  43,  54,  78,  94,  96, 116, 119, 152, 164,
        170, 172, 174, 183, 193, 195, 205, 232, 252, 261, 268, 275, 280,
        297, 305, 326, 340, 348, 366, 374, 381, 386, 407, 438, 456, 467,
        469, 479, 483, 487, 493, 530, 545, 555, 570, 587, 625, 626, 630,
        642, 644, 647, 659, 672, 684, 694, 745, 755, 772, 788, 803, 824,
        827, 829, 831, 851, 879], dtype=int64),)
```

In [62]:
```
Q1 = np.percentile(df['age'], 25)
Q3 = np.percentile(df['age'], 75)
IQR = Q3 - Q1
```

In [63]:
```
upper=Q3+1.5*IQR
upper_array=np.array(df['age']>=upper)
print("Upper Bound:",upper)

lower=Q1-1.5*IQR
lower_array=np.array(df['age']<=lower)
print("Lower Bound:",lower)
```

```
Upper Bound: 54.5
Lower Bound: 2.5
```

Checking for Categorical columns and perform encoding.

In [64]:
```
categorical_columns = df.select_dtypes(include=['object']).columns
```

In [65]:
```
for column in categorical_columns:
    print(column)
```

```
sex
embarked
class
who
deck
embark_town
alive
```

In [66]:
```
encoded_df = pd.get_dummies(df, columns=categorical_columns)
```

In [67]:
```
print("\nEncoded dataframe:")
print(encoded_df)
```

```
Encoded dataframe:
     survived  pclass   age  sibsp  parch     fare  adult_male  alone  \
0           0       3  22.0      1      0   7.2500        True  False
1           1       1  38.0      1      0  71.2833       False  False
2           1       3  26.0      0      0   7.9250       False   True
3           1       1  35.0      1      0  53.1000       False  False
4           0       3  35.0      0      0   8.0500        True   True
..        ...     ...   ...    ...    ...      ...         ...    ...
886         0       2  27.0      0      0  13.0000        True   True
887         1       1  19.0      0      0  30.0000       False   True
888         0       3  28.0      1      2  23.4500       False  False
889         1       1  26.0      0      0  30.0000        True   True
890         0       3  32.0      0      0   7.7500        True   True

     sex_female  sex_male  ...  deck_D  deck_E  deck_F  deck_G  deck_NO  \
0             0         1  ...       0       0       0       0        1
1             1         0  ...       0       0       0       0        0
2             1         0  ...       0       0       0       0        1
3             1         0  ...       0       0       0       0        0
4             0         1  ...       0       0       0       0        1
..          ...       ...  ...     ...     ...     ...     ...      ...
886           0         1  ...       0       0       0       0        1
887           1         0  ...       0       0       0       0        0
888           1         0  ...       0       0       0       0        1
889           0         1  ...       0       0       0       0        0
890           0         1  ...       0       0       0       0        1

     embark_town_Cherbourg  embark_town_Queenstown  embark_town_Southampton  \
0                        0                       0                        1
1                        1                       0                        0
2                        0                       0                        1
3                        0                       0                        1
4                        0                       0                        1
..                     ...                     ...                      ...
886                      0                       0                        1
887                      0                       0                        1
888                      0                       0                        1
889                      1                       0                        0
890                      0                       1                        0

     alive_no  alive_yes
0           1          0
1           0          1
2           0          1
3           0          1
4           1          0
..        ...        ...
886         1          0
887         0          1
888         1          0
889         0          1
890         1          0

[891 rows x 32 columns]
```

Spliting the data into dependent and independent variables.

In [68]:
```
X = df.drop('survived', axis=1)  # Independent variables (all columns except 'surv
y = df['survived']  # Dependent variable ('survived' column)
```

In [69]:
```
print("Independent variables:")
print(X)
```

```
Independent variables:
     pclass     sex   age  sibsp  parch     fare embarked   class    who  \
0         3    male  22.0      1      0   7.2500        S   Third    man
1         1  female  38.0      1      0  71.2833        C   First  woman
2         3  female  26.0      0      0   7.9250        S   Third  woman
3         1  female  35.0      1      0  53.1000        S   First  woman
4         3    male  35.0      0      0   8.0500        S   Third    man
..      ...     ...   ...    ...    ...      ...      ...     ...    ...
886       2    male  27.0      0      0  13.0000        S  Second    man
887       1  female  19.0      0      0  30.0000        S   First  woman
888       3  female  28.0      1      2  23.4500        S   Third  woman
889       1    male  26.0      0      0  30.0000        C   First    man
890       3    male  32.0      0      0   7.7500        Q   Third    man

     adult_male deck  embark_town alive  alone
0          True   NO  Southampton    no  False
1         False    C    Cherbourg   yes  False
2         False   NO  Southampton   yes   True
3         False    C  Southampton   yes  False
4          True   NO  Southampton    no   True
..          ...  ...          ...   ...    ...
886        True   NO  Southampton    no   True
887       False    B  Southampton   yes   True
888       False   NO  Southampton    no  False
889        True    C    Cherbourg   yes   True
890        True   NO   Queenstown    no   True

[891 rows x 14 columns]
```

In [70]:
```
print("\nDependent variable:")
print(y)
```

```
Dependent variable:
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: survived, Length: 891, dtype: int64
```

Scaling the independent variables

In [71]:
```
dependent_variable = df['survived']  # Assuming 'survived' is the dependent variabl
independent_variables = df.drop('survived', axis=1)
```

In [72]:
```
scaled_independent_variables = (independent_variables - independent_variables.mean
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_12960\1149853797.py:1: FutureWarning: Dro
pping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is de
precated; in a future version this will raise TypeError.  Select only valid column
s before calling the reduction.
  scaled_independent_variables = (independent_variables - independent_variables.me
an()) / independent_variables.std()
C:\Users\HP\AppData\Local\Temp\ipykernel_12960\1149853797.py:1: FutureWarning: Dro
pping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is de
precated; in a future version this will raise TypeError.  Select only valid column
s before calling the reduction.
  scaled_independent_variables = (independent_variables - independent_variables.me
an()) / independent_variables.std()
```

In [73]:
```
print("Scaled independent variables:")
print(scaled_independent_variables)
```

```
Scaled independent variables:
     adult_male       age alive     alone class deck embark_town embarked  \
0      0.811467 -0.565419   NaN -1.230954   NaN  NaN         NaN      NaN
1     -1.230954  0.663488   NaN -1.230954   NaN  NaN         NaN      NaN
2     -1.230954 -0.258192   NaN  0.811467   NaN  NaN         NaN      NaN
3     -1.230954  0.433068   NaN -1.230954   NaN  NaN         NaN      NaN
4      0.811467  0.433068   NaN  0.811467   NaN  NaN         NaN      NaN
..          ...       ...   ...       ...   ...  ...         ...      ...
886    0.811467 -0.181385   NaN  0.811467   NaN  NaN         NaN      NaN
887   -1.230954 -0.795839   NaN  0.811467   NaN  NaN         NaN      NaN
888   -1.230954 -0.104579   NaN -1.230954   NaN  NaN         NaN      NaN
889    0.811467 -0.258192   NaN  0.811467   NaN  NaN         NaN      NaN
890    0.811467  0.202648   NaN  0.811467   NaN  NaN         NaN      NaN

         fare     parch    pclass  sex     sibsp  who
0   -0.502163 -0.473408  0.826913  NaN  0.432550  NaN
1    0.786404 -0.473408 -1.565228  NaN  0.432550  NaN
2   -0.488580 -0.473408  0.826913  NaN -0.474279  NaN
3    0.420494 -0.473408 -1.565228  NaN  0.432550  NaN
4   -0.486064 -0.473408  0.826913  NaN -0.474279  NaN
..        ...       ...       ...  ...       ...  ...
886 -0.386454 -0.473408 -0.369158  NaN -0.474279  NaN
887 -0.044356 -0.473408 -1.565228  NaN -0.474279  NaN
888 -0.176164  2.007806  0.826913  NaN  0.432550  NaN
889 -0.044356 -0.473408 -1.565228  NaN -0.474279  NaN
890 -0.492101 -0.473408  0.826913  NaN -0.474279  NaN

[891 rows x 14 columns]
```

Spliting the data into training and testing

In [75]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [76]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st
```

In [77]:
```
# Print the shapes of the training and testing sets
print("Training set shape:", X_train.shape, y_train.shape)
print("Testing set shape:", X_test.shape, y_test.shape)
```

```
Training set shape: (712, 14) (712,)
Testing set shape: (179, 14) (179,)
```

In [ ]:
```

```