YENDLURI SRUTHI

20BCE7396

Titanic Ship Case Study

Problem Description: On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. Translated 32% survival rate.

^ One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew.

^ Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper□class. The problem associated with the Titanic dataset is to predict whether a passenger survived the disaster or not. The dataset contains various features such as passenger class, age, gender, cabin, fare, and whether the passenger had any siblings or spouses on board. These features can be used to build a predictive model to determine the likelihood of a passenger surviving the disaster. The dataset offers opportunities for feature engineering, data visualization, and model selection, making it a valuable resource for developing and testing data analysis and machine learning skills.

1. Download the dataset

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
```

2. Load the dataset.

In [3]:

```
df = pd.read_csv("titanic.csv")
df.head()
```

Out[3]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_ma
0	0	3	male	22.0	1	0	7.2500	S	Third	man	Tr
1	1	1	female	38.0	1	0	71.2833	C	First	woman	Fal
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	Fal
3	1	1	female	35.0	1	0	53.1000	S	First	woman	Fal
4	0	3	male	35.0	0	0	8.0500	S	Third	man	Tr

In [5]:

```
df.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    object 
 9   who          891 non-null    object 
 10  adult_male   891 non-null    bool   
 11  deck         203 non-null    object 
 12  embark_town  889 non-null    object 
 13  alive        891 non-null    object 
 14  alone        891 non-null    bool   
dtypes: bool(2), float64(2), int64(4), object(7)
memory usage: 92.4+ KB
```

3. Perform Below Visualizations.

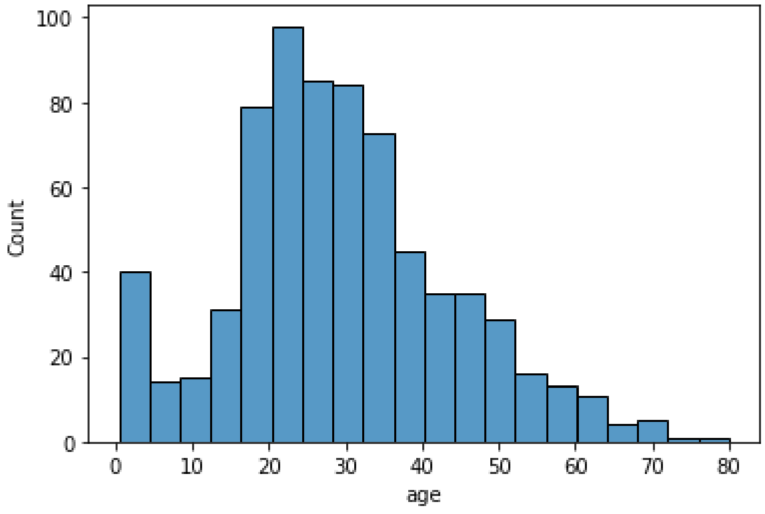
● Univariate Analysis

In [6]:

```
sns.histplot(df['age'])
```

Out[6]:

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

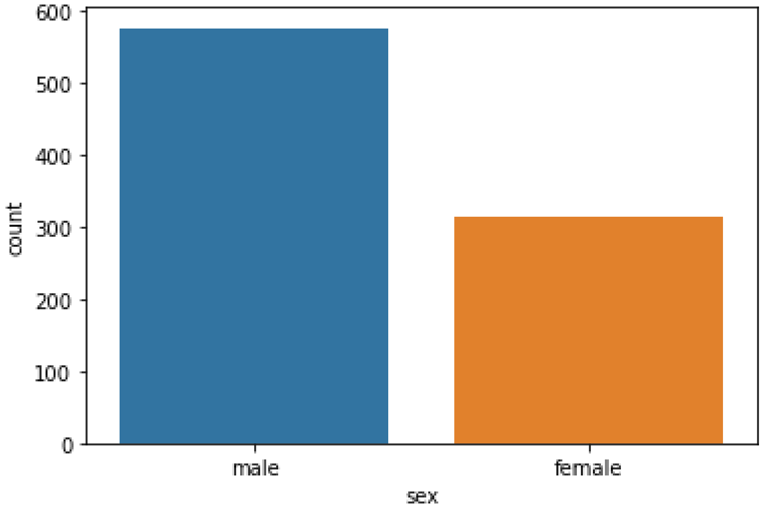


In [7]:

```
sns.countplot(x = df['sex'])
```

Out[7]:

```
<AxesSubplot:xlabel='sex', ylabel='count'>
```

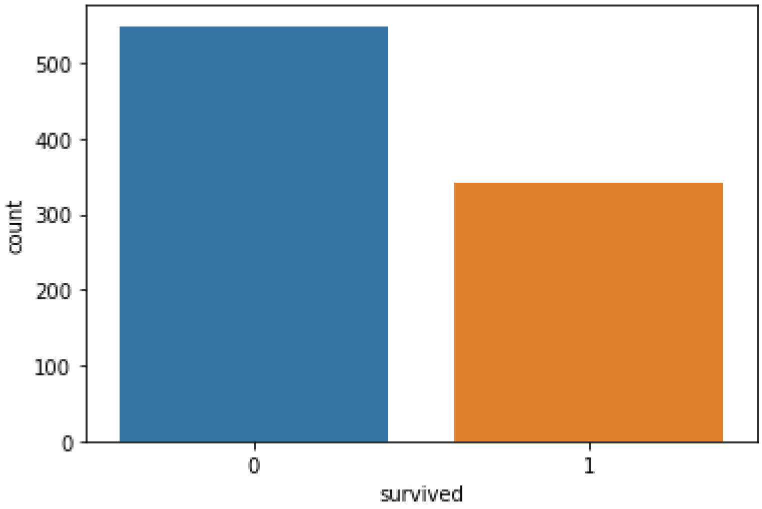


In [8]:

```
sns.countplot(x = df['survived'])
```

Out[8]:

```
<AxesSubplot:xlabel='survived', ylabel='count'>
```

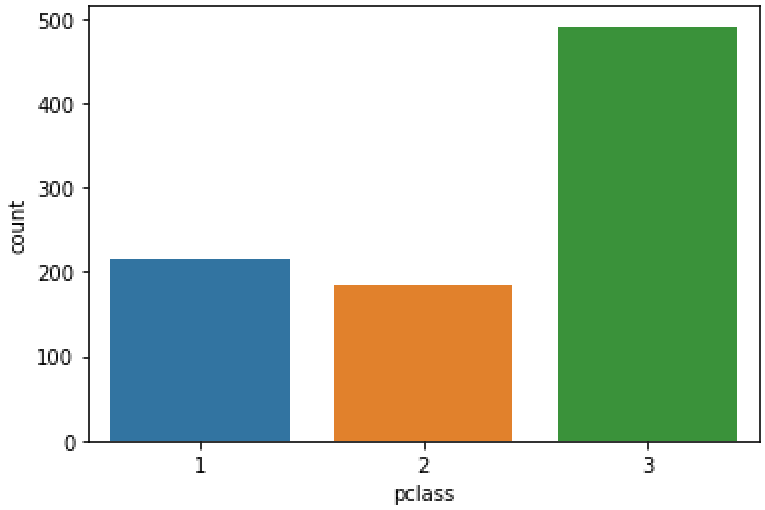


In [9]:

```
sns.countplot(x = df['pclass'])
```

Out[9]:

```
<AxesSubplot:xlabel='pclass', ylabel='count'>
```

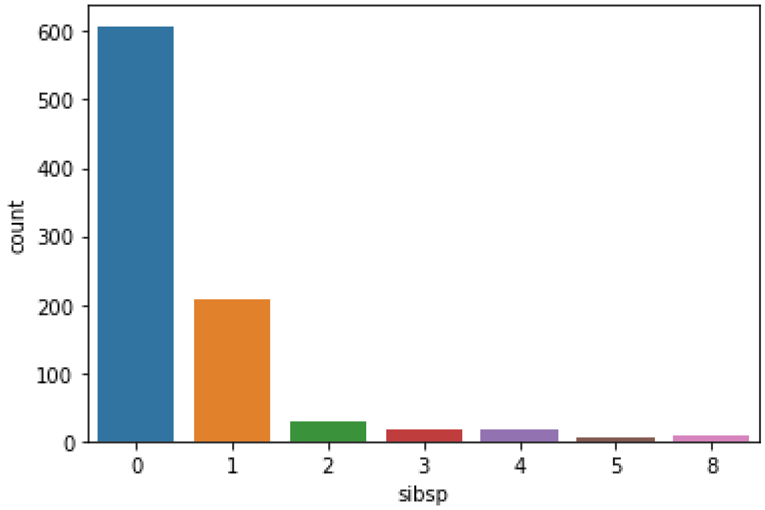


In [10]:

```
sns.countplot(x = df['sibsp'])
```

Out[10]:

```
<AxesSubplot:xlabel='sibsp', ylabel='count'>
```

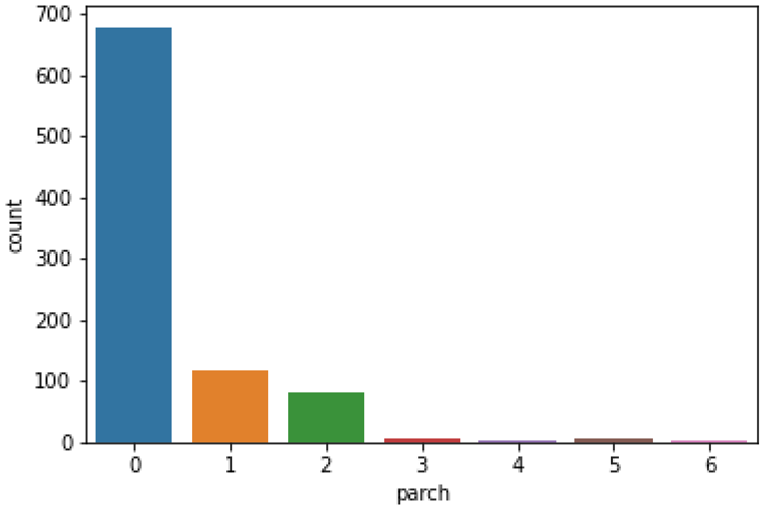


In [11]:

```
sns.countplot(x = df['parch'])
```

Out[11]:

```
<AxesSubplot:xlabel='parch', ylabel='count'>
```

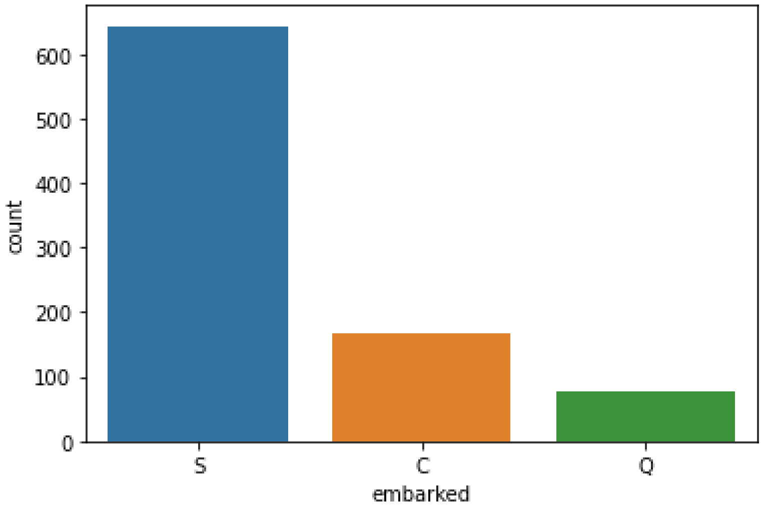


In [12]:

```
sns.countplot(x = df['embarked'])
```

Out[12]:

```
<AxesSubplot:xlabel='embarked', ylabel='count'>
```



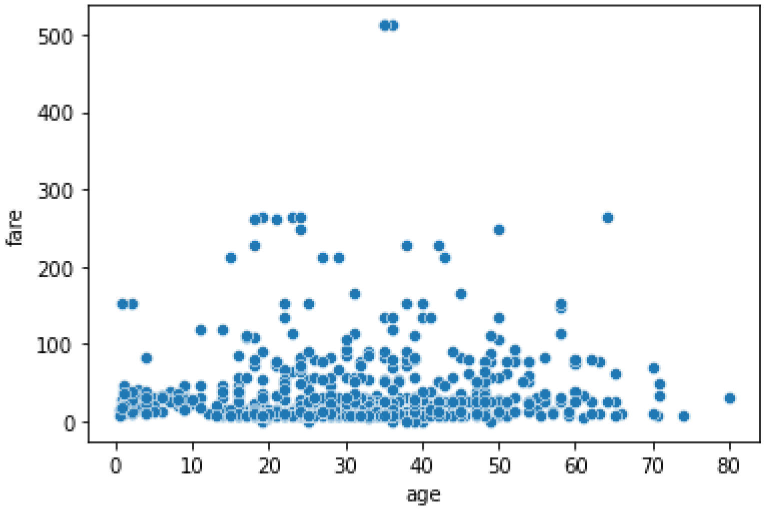
• Bi - Variate Analysis

In [13]:

```
sns.scatterplot(data = df, x='age', y = 'fare')
```

Out[13]:

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

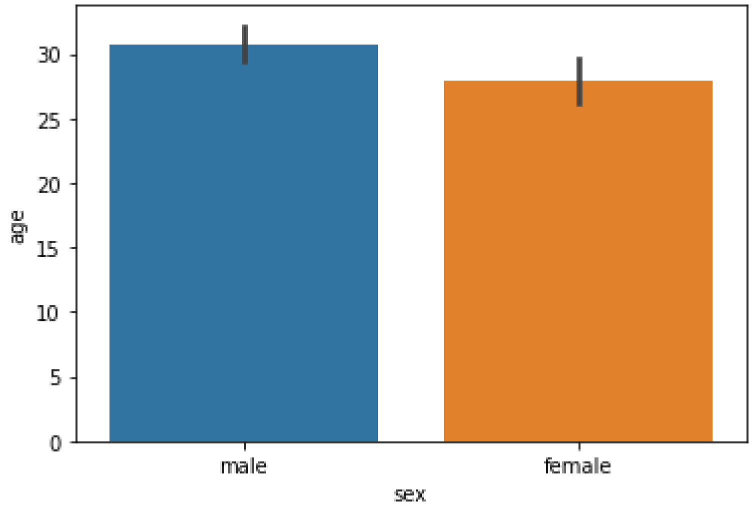


In [14]:

```
sns.barplot(data = df, x = 'sex', y = 'age')
```

Out[14]:

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

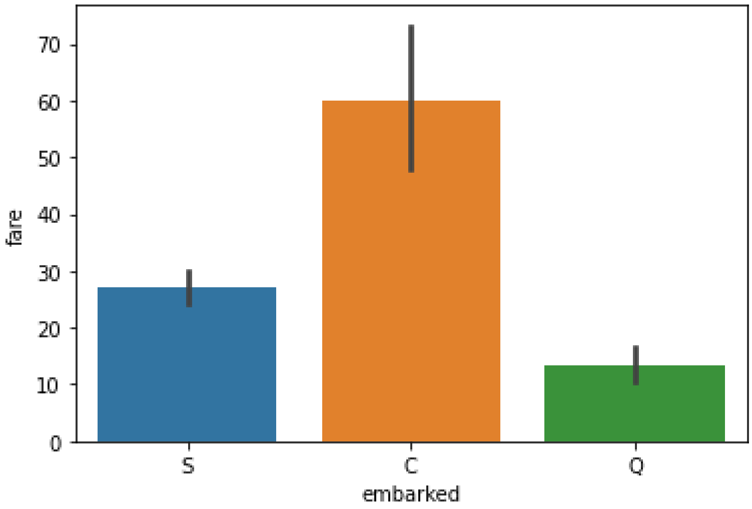


In [15]:

```
sns.barplot(data = df, x='embarked', y='fare')
```

Out[15]:

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

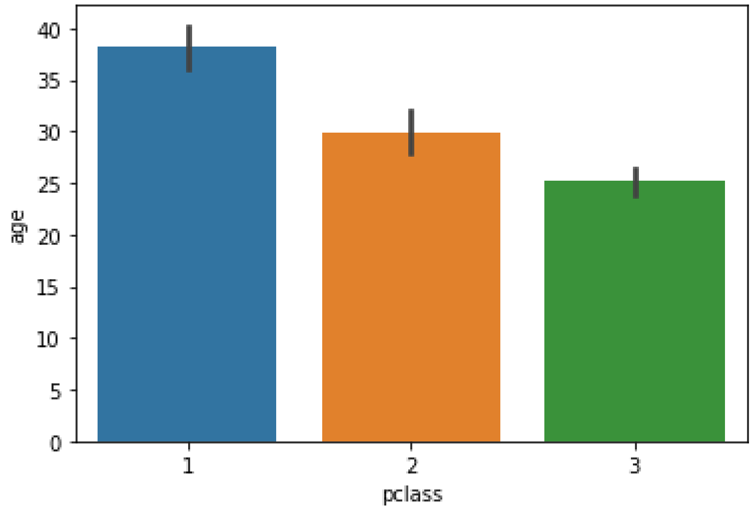


In [16]:

```
sns.barplot(data = df, x='pclass', y='age')
```

Out[16]:

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

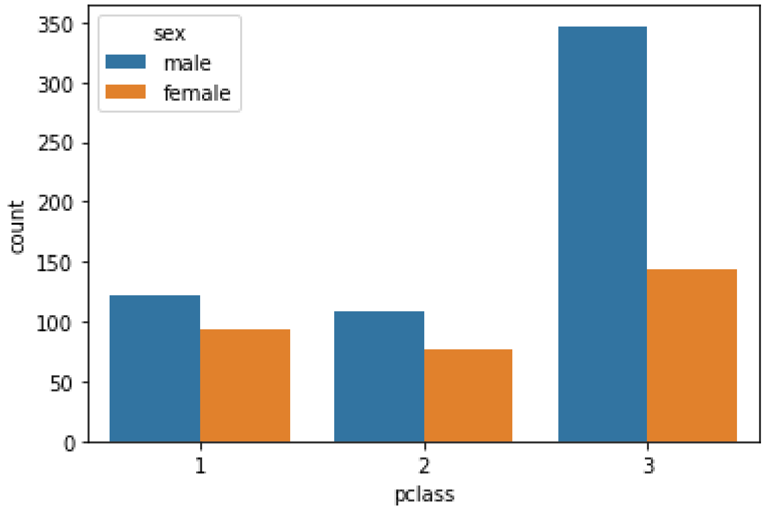


In [18]:

```
sns.countplot(x = df['pclass'], hue = df['sex'])
```

Out[18]:

```
<AxesSubplot:xlabel='pclass', ylabel='count'>
```

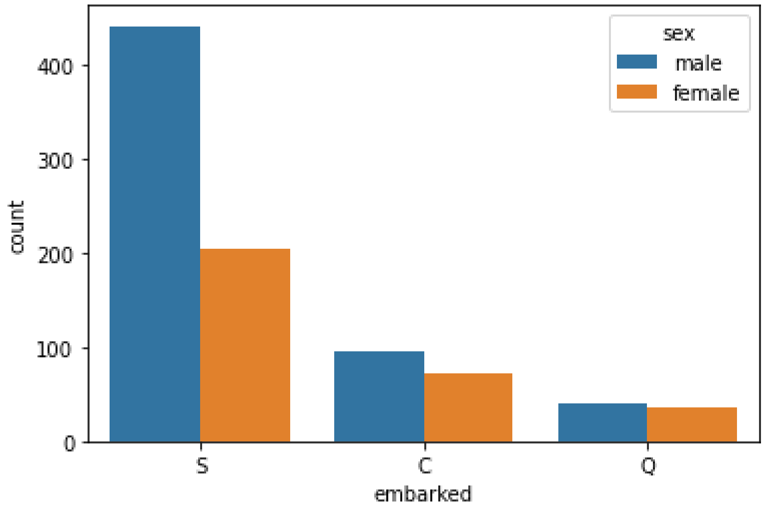


In [19]:

```
sns.countplot(x = df['embarked'], hue = df['sex'])
```

Out[19]:

```
<AxesSubplot:xlabel='embarked', ylabel='count'>
```

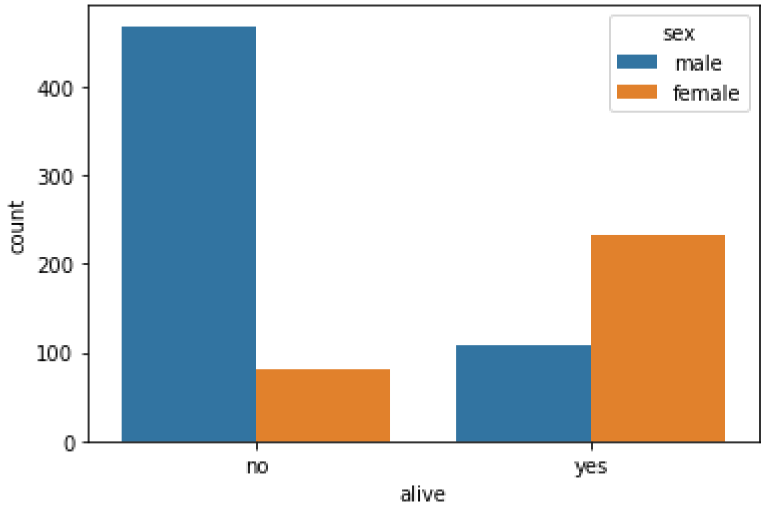


In [20]:

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

Out[20]:

```
<AxesSubplot:xlabel='alive', ylabel='count'>
```



● Multi - Variate Analysis

In [21]:

```
sns.heatmap(df.corr(numeric_only=True), annot = True)
```

```
---------------------------------------------------------------------------
--
TypeError                                 Traceback (most recent call las
t)
Input In [21], in <cell line: 1>()
----> 1 sns.heatmap(df.corr(numeric_only=True), annot = True)

TypeError: corr() got an unexpected keyword argument 'numeric_only'
```

4. Perform descriptive statistics on the dataset

In [22]:

```
df.describe()
```

Out[22]:

	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

5. Handle the Missing values

In [23]:

```
df.isnull().sum()
```

Out[23]:

```
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 [24]:

```
df.dropna(subset=['embark_town'], how='all', inplace = True)
df['age']=df['age'].fillna(df['age'].mean())
df.drop(['deck'], axis = 1,inplace = True)
df.isnull().sum()
```

Out[24]:

```
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
```

6. Find the outliers and replace the outliers

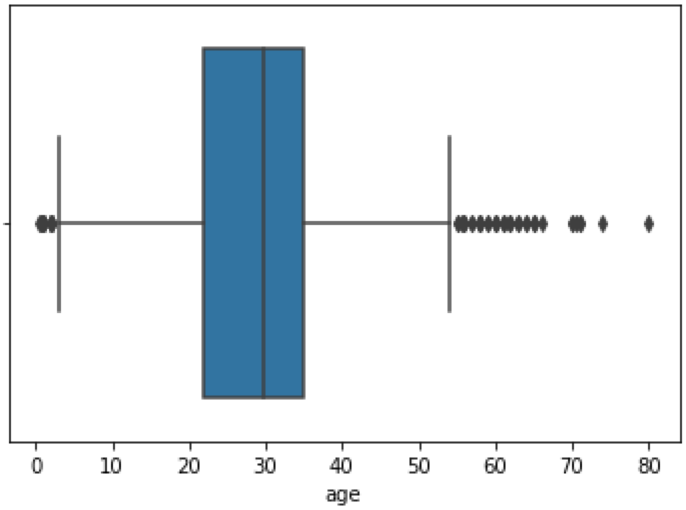
In [25]:

```
sns.boxplot(df['age'])
```

```
C:\Users\Sruthi Yendluri\anaconda3_1\lib\site-packages\seaborn\_decorator
s.py:36: FutureWarning: Pass the following variable as a keyword arg: x.
From version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an err
or or misinterpretation.
  warnings.warn(
```

Out[25]:

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



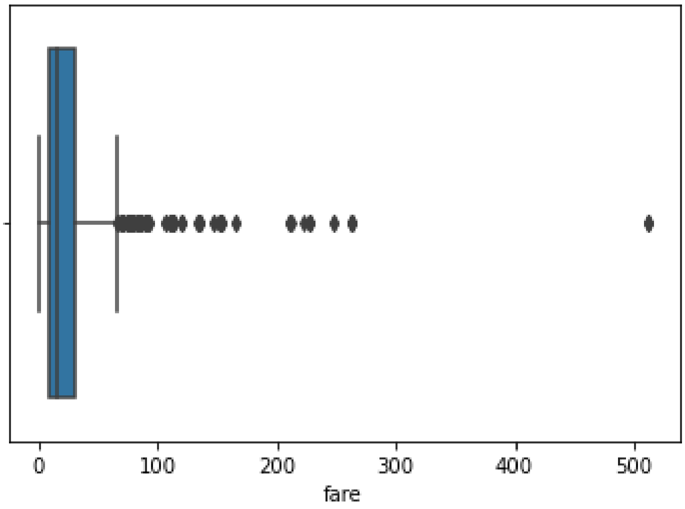
In [26]:

```
sns.boxplot(df['fare'])
```

```
C:\Users\Sruthi Yendluri\anaconda3_1\lib\site-packages\seaborn\_decorator
s.py:36: FutureWarning: Pass the following variable as a keyword arg: x.
From version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an err
or or misinterpretation.
  warnings.warn(
```

Out[26]:

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



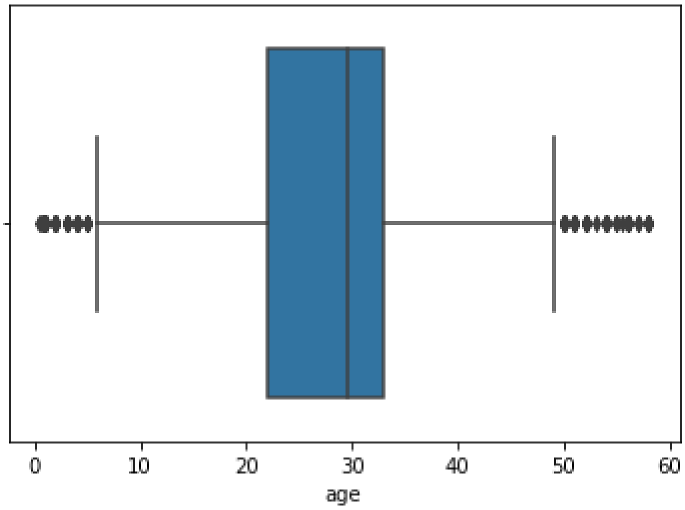
In [27]:

```
median_age = df['age'].median()
df["age"] = np.where(df["age"] > 58, median_age, df['age'])
sns.boxplot(df['age'])
```

```
C:\Users\Sruthi Yendluri\anaconda3_1\lib\site-packages\seaborn\_decorator
s.py:36: FutureWarning: Pass the following variable as a keyword arg: x.
From version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an err
or or misinterpretation.
  warnings.warn(
```

Out[27]:

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



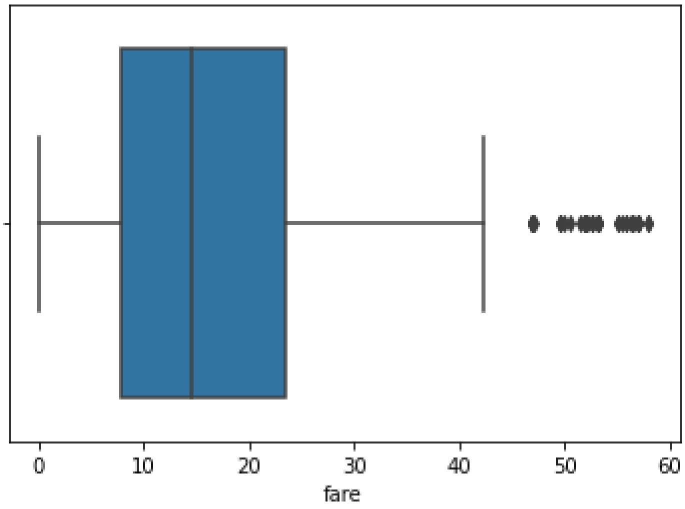
In [28]:

```
median_age = df['fare'].median()
df["fare"] = np.where(df["fare"] > 58, median_age, df['fare'])
sns.boxplot(df['fare'])
```

```
C:\Users\Sruthi Yendluri\anaconda3_1\lib\site-packages\seaborn\_decorator
s.py:36: FutureWarning: Pass the following variable as a keyword arg: x.
From version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an err
or or misinterpretation.
  warnings.warn(
```

Out[28]:

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



7. Check for Categorical columns and perform encoding

In [30]:

```
from sklearn.preprocessing import OneHotEncoder
encoding = pd.get_dummies(df, columns = ['sex','embarked','class','who','adult_male','em
encoding.head()
```

Out[30]:

	survived	pclass	age	sibsp	parch	fare	alive	sex_female	sex_male	embarked_C
0	0	3	22.0	1	0	7.2500	no	0	1	0
1	1	1	38.0	1	0	14.4542	yes	1	0	1
2	1	3	26.0	0	0	7.9250	yes	1	0	0
3	1	1	35.0	1	0	53.1000	yes	1	0	0
4	0	3	35.0	0	0	8.0500	no	0	1	0

5 rows × 25 columns

8. Split the data into dependent and independent variables.

In [31]:

```
df.columns
```

Out[31]:

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

In [34]:

```
x=encoding.drop(['survived','alive'],axis = 1)
x.head()
```

Out[34]:

	pclass	age	sibsp	parch	fare	sex_female	sex_male	embarked_C	embarked_Q	e
0	3	22.0	1	0	7.2500	0	1	0	0	
1	1	38.0	1	0	14.4542	1	0	1	0	
2	3	26.0	0	0	7.9250	1	0	0	0	
3	1	35.0	1	0	53.1000	1	0	0	0	
4	3	35.0	0	0	8.0500	0	1	0	0	

5 rows × 23 columns

9. Scale the independent variables

In [38]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_std = scaler.fit_transform(x)
x_std
```

Out[38]:

```
array([[ 0.82520863, -0.57985934,  0.43135024, ...,  0.61679395,
         1.22934919, -1.22934919],
       [-1.57221121,  0.83108889,  0.43135024, ..., -1.62128697,
         1.22934919, -1.22934919],
       [ 0.82520863, -0.22712228, -0.47519908, ...,  0.61679395,
        -0.81343853,  0.81343853],
       ...,
       [ 0.82520863,  0.09405298,  0.43135024, ...,  0.61679395,
         1.22934919, -1.22934919],
       [-1.57221121, -0.22712228, -0.47519908, ..., -1.62128697,
        -0.81343853,  0.81343853],
       [ 0.82520863,  0.3019833 , -0.47519908, ..., -1.62128697,
        -0.81343853,  0.81343853]])
```

10. Split the data into training and testing

In [43]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x, y['survived'], test_size=0.33)
```

```
--------------------------------------------------------------------------
--
NameError                                 Traceback (most recent call las
t)
Input In [43], in <cell line: 2>()
      1 from sklearn.model_selection import train_test_split
----> 2 x_train,x_test,y_train,y_test = train_test_split(x, y['survive
d'], test_size=0.33)

NameError: name 'y' is not defined
```

In [ ]:

```

```