

```
In [1]: import numpy as np
import pandas as pd
import seaborn as sns
```

```
In [2]: #load the dataset

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

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

```
Out[3]:
```

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

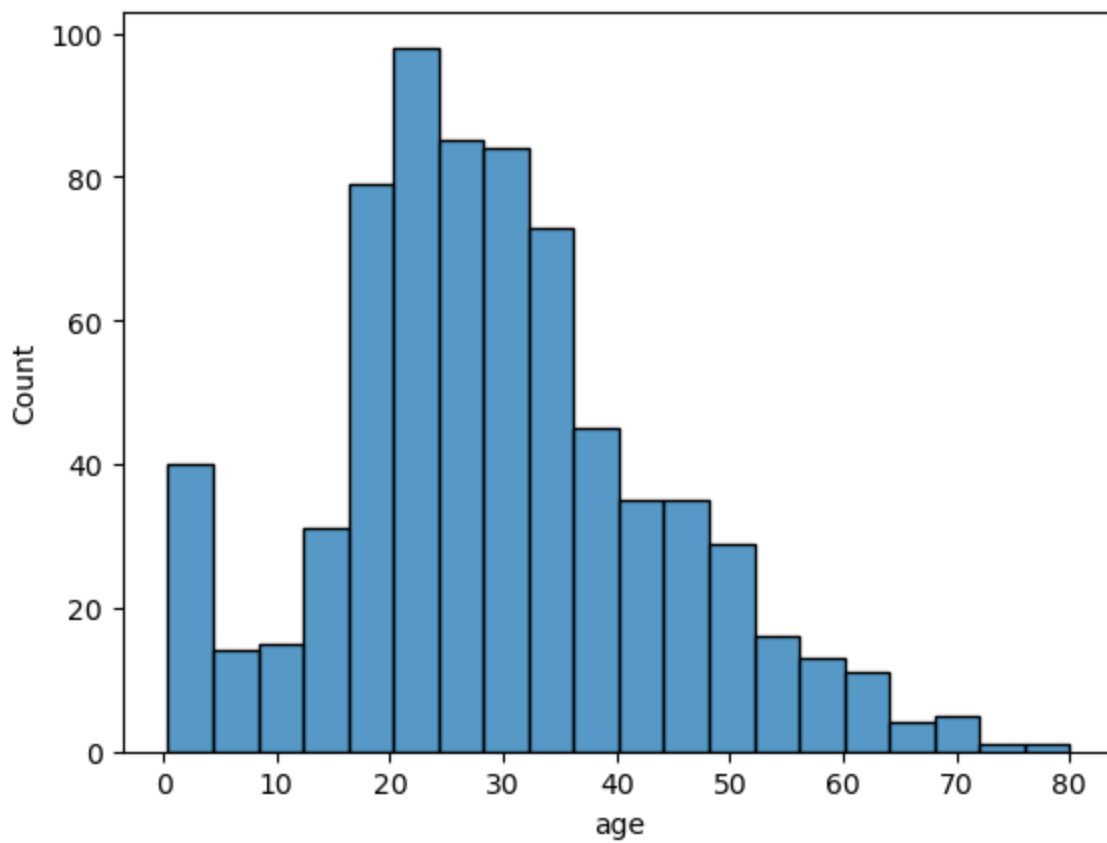
```
In [4]: df.info()

<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
```

```
In [5]: #univariate analysis

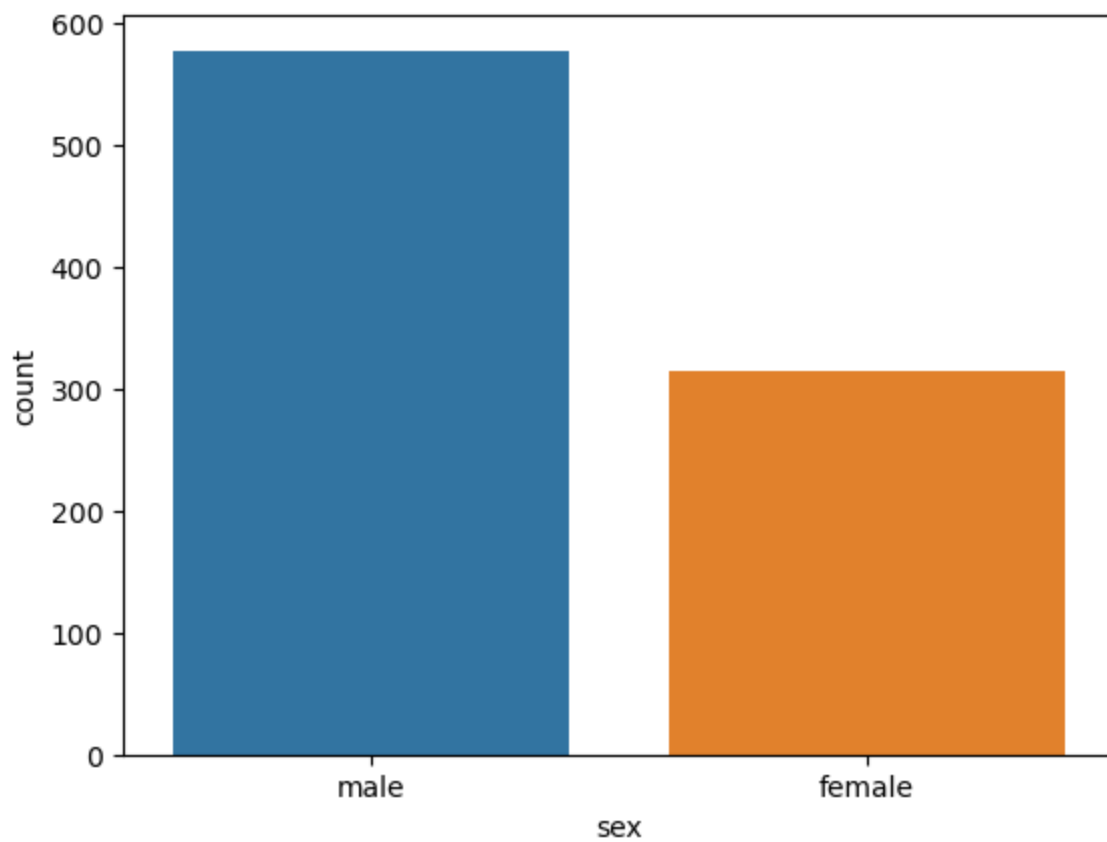
sns.histplot(df['age'])
```

```
Out[5]: <Axes: xlabel='age', ylabel='Count'>
```



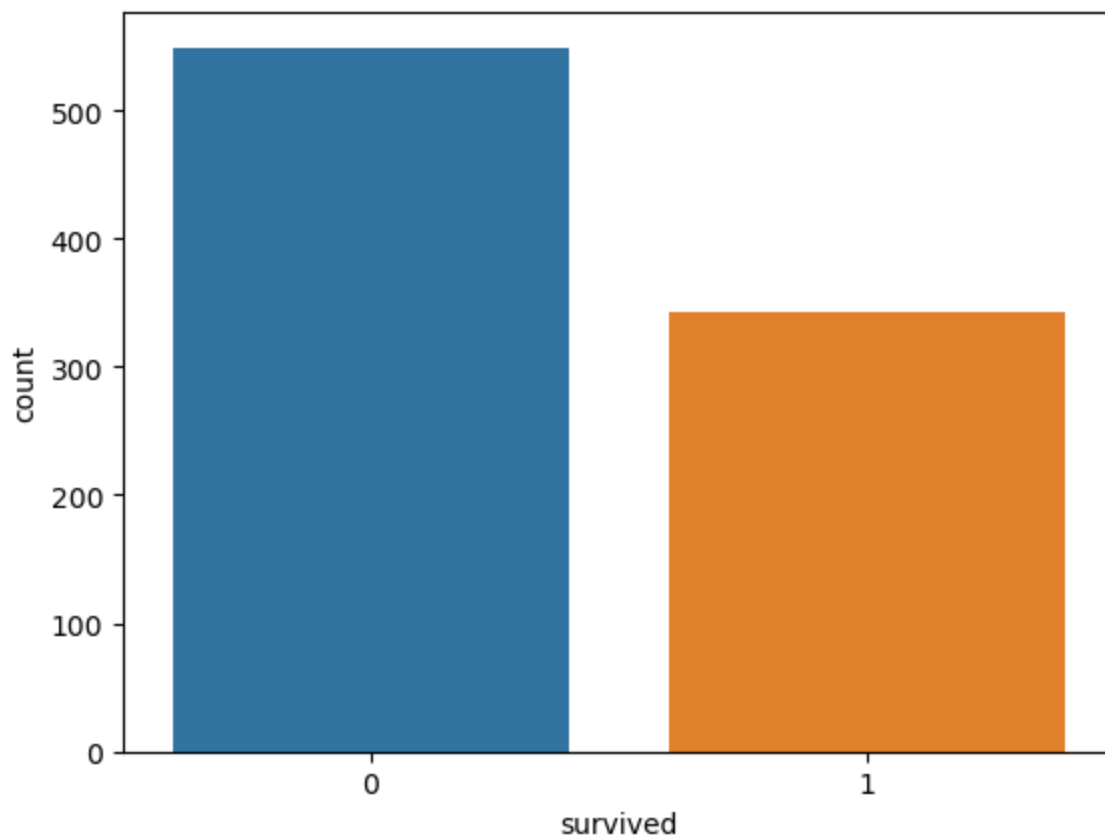
```
In [6]: sns.countplot(x = df['sex'])
```

```
Out[6]: <Axes: xlabel='sex', ylabel='count'>
```



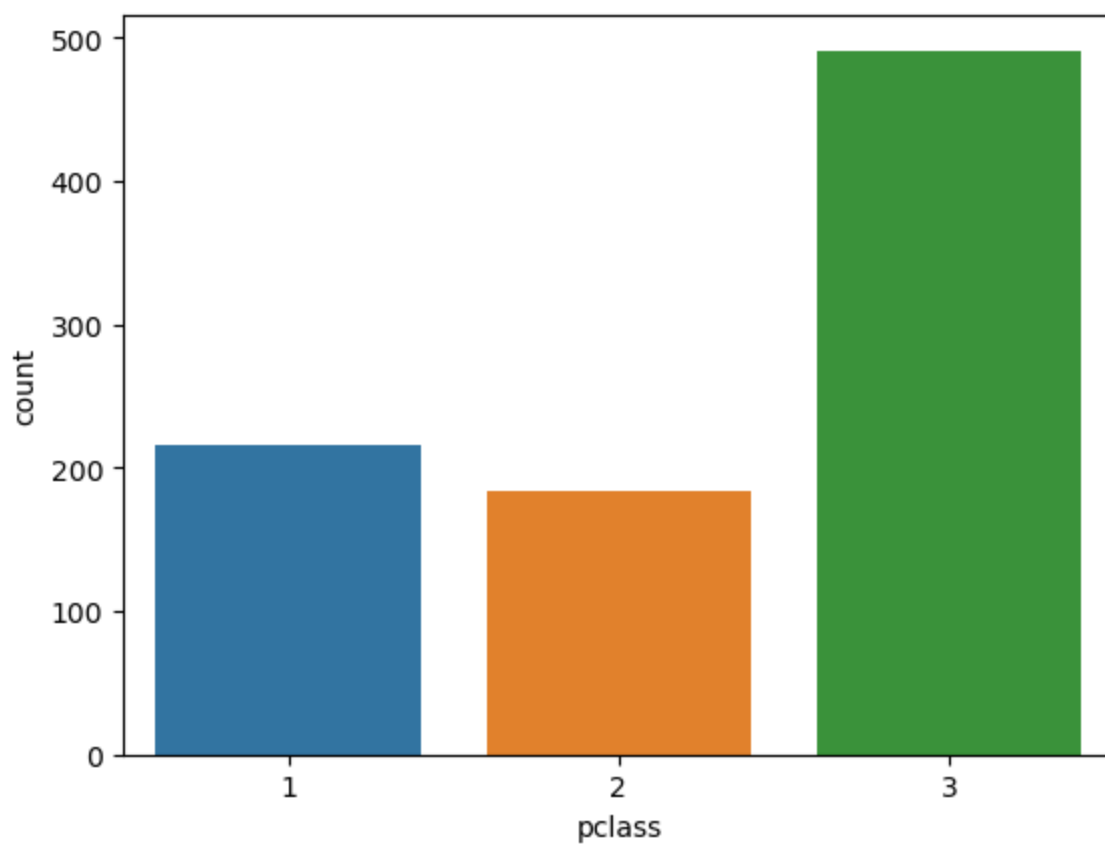
```
In [7]: sns.countplot(x = df['survived'])
```

```
Out[7]: <Axes: xlabel='survived', ylabel='count'>
```



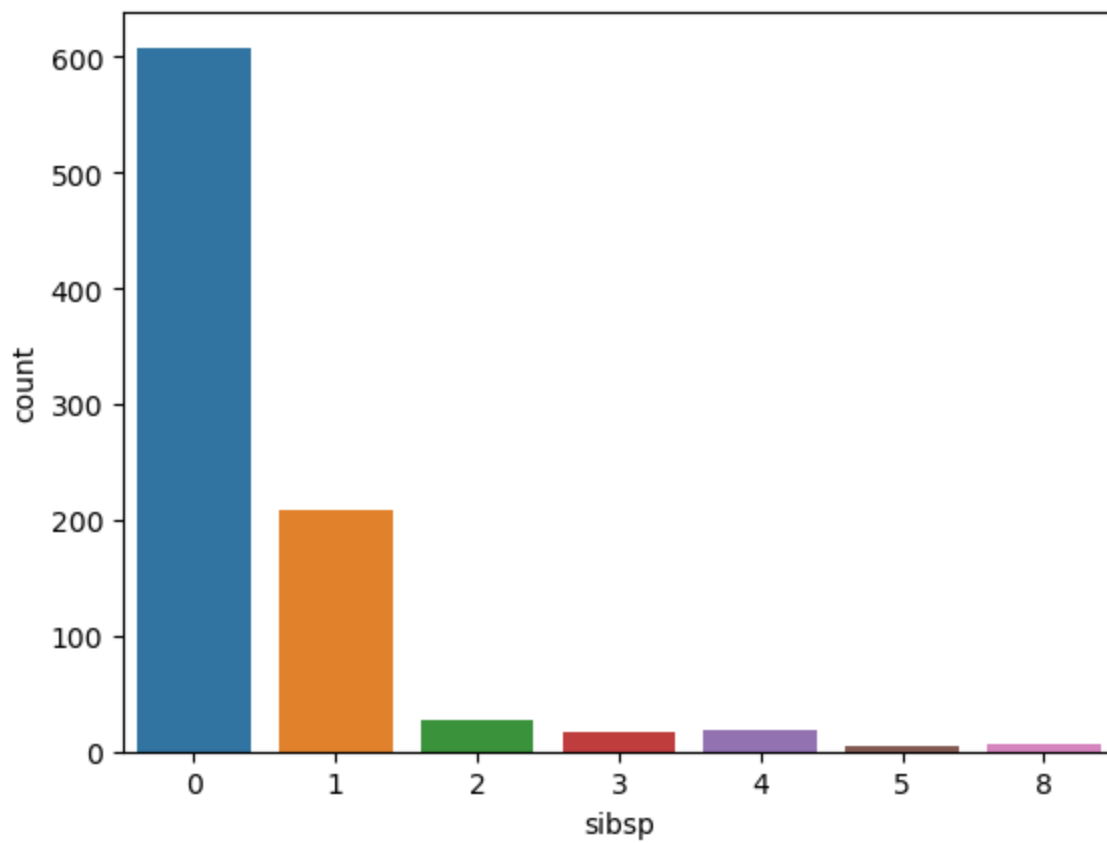
```
In [8]: sns.countplot(x = df['pclass'])
```

```
Out[8]: <Axes: xlabel='pclass', ylabel='count'>
```



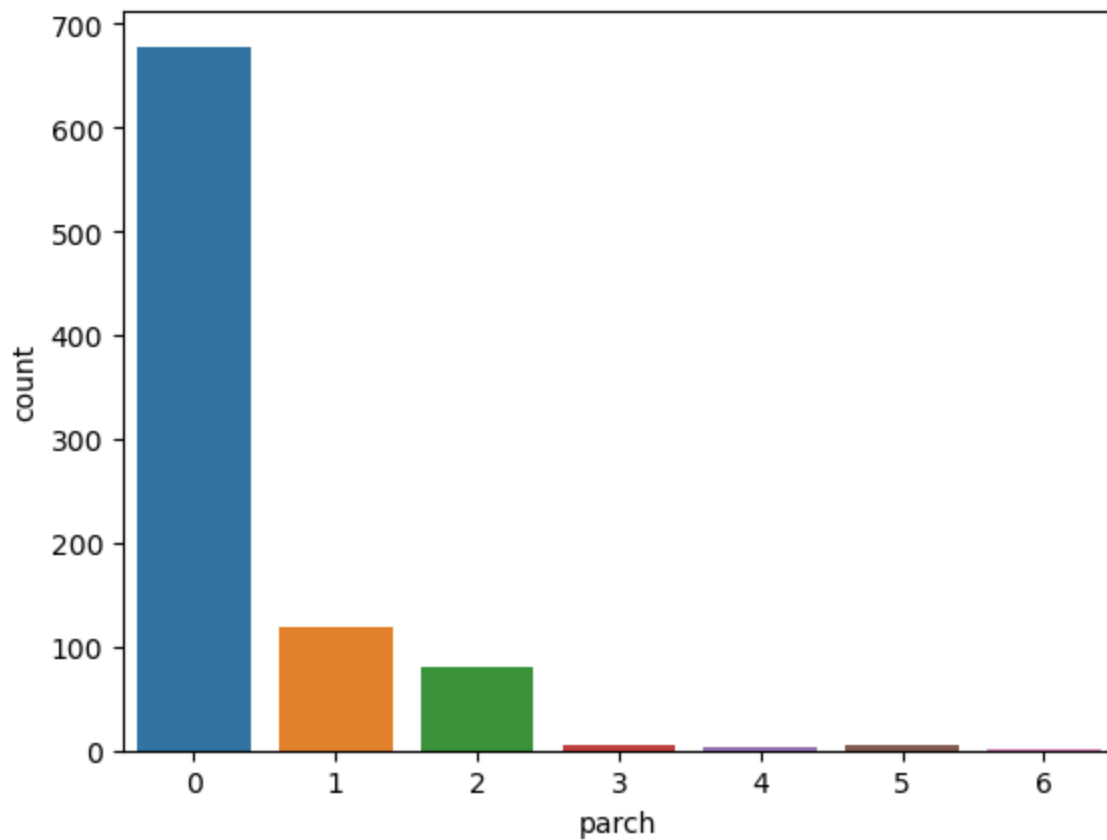
```
In [9]: sns.countplot(x = df['sibsp'])
```

```
Out[9]: <Axes: xlabel='sibsp', ylabel='count'>
```



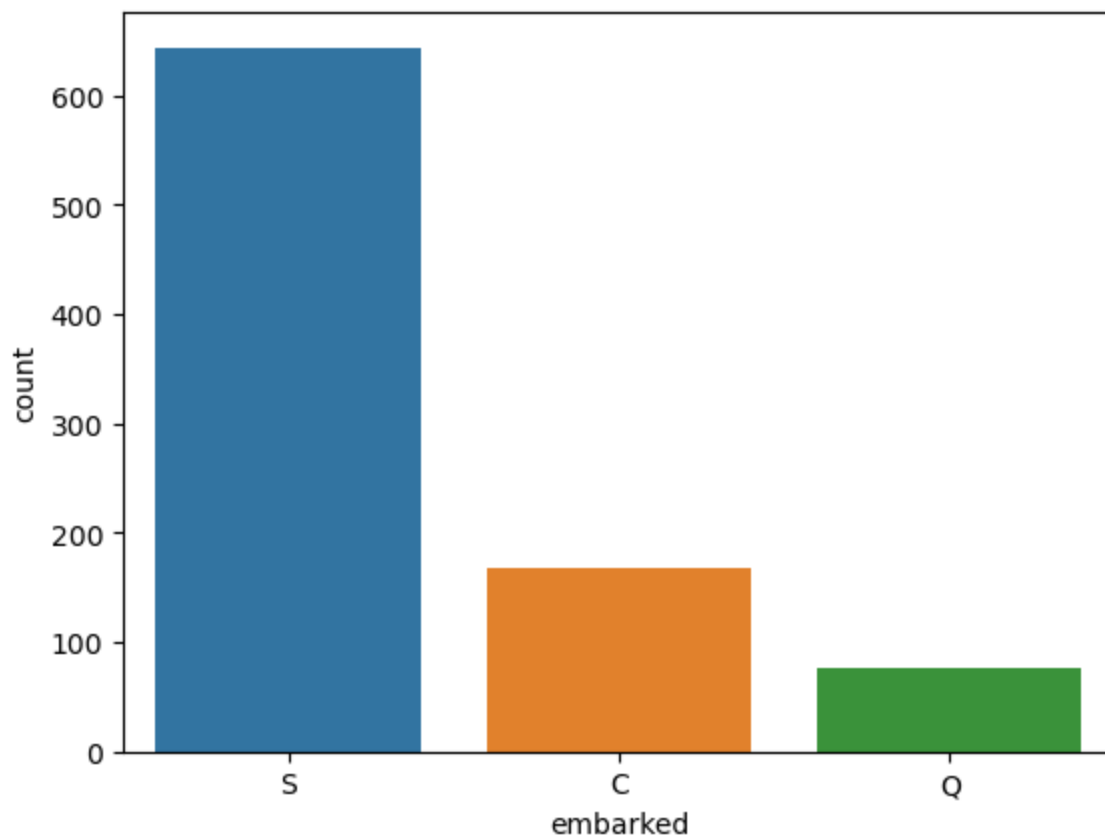
```
In [10]: sns.countplot(x = df['parch'])
```

```
Out[10]: <Axes: xlabel='parch', ylabel='count'>
```



```
In [11]: sns.countplot(x = df['embarked'])
```

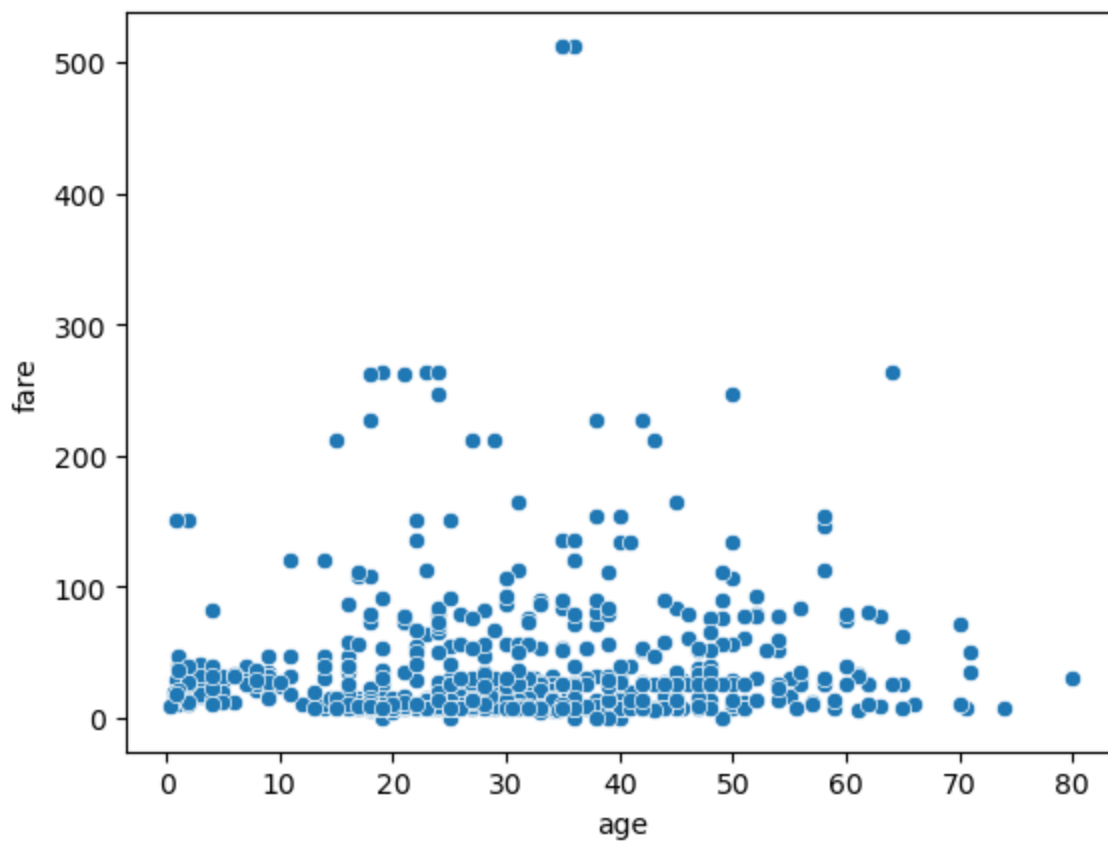
```
Out[11]: <Axes: xlabel='embarked', ylabel='count'>
```



```
In [12]: # bivariate analysis
```

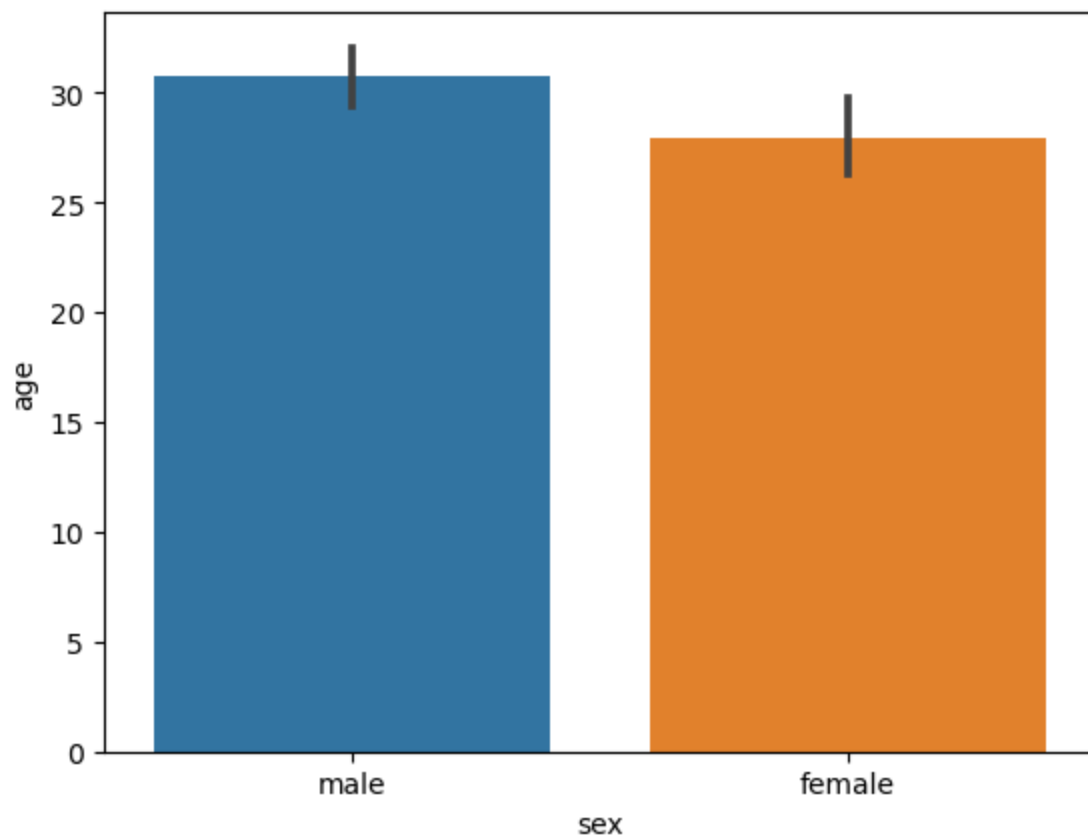
```
In [13]: sns.scatterplot(data = df, x = 'age', y = 'fare')
```

```
Out[13]: <Axes: xlabel='age', ylabel='fare'>
```



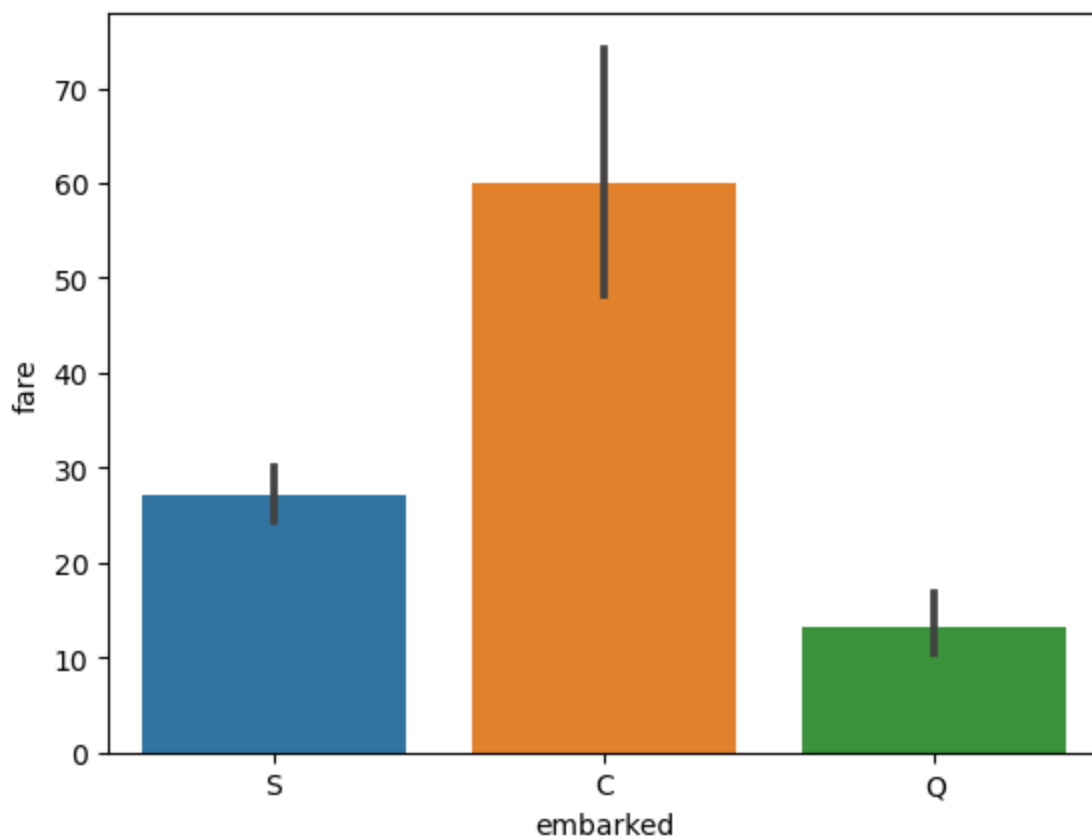
```
In [14]: sns.barplot(data = df, x = 'sex', y = 'age')
```

```
Out[14]: <Axes: xlabel='sex', ylabel='age'>
```



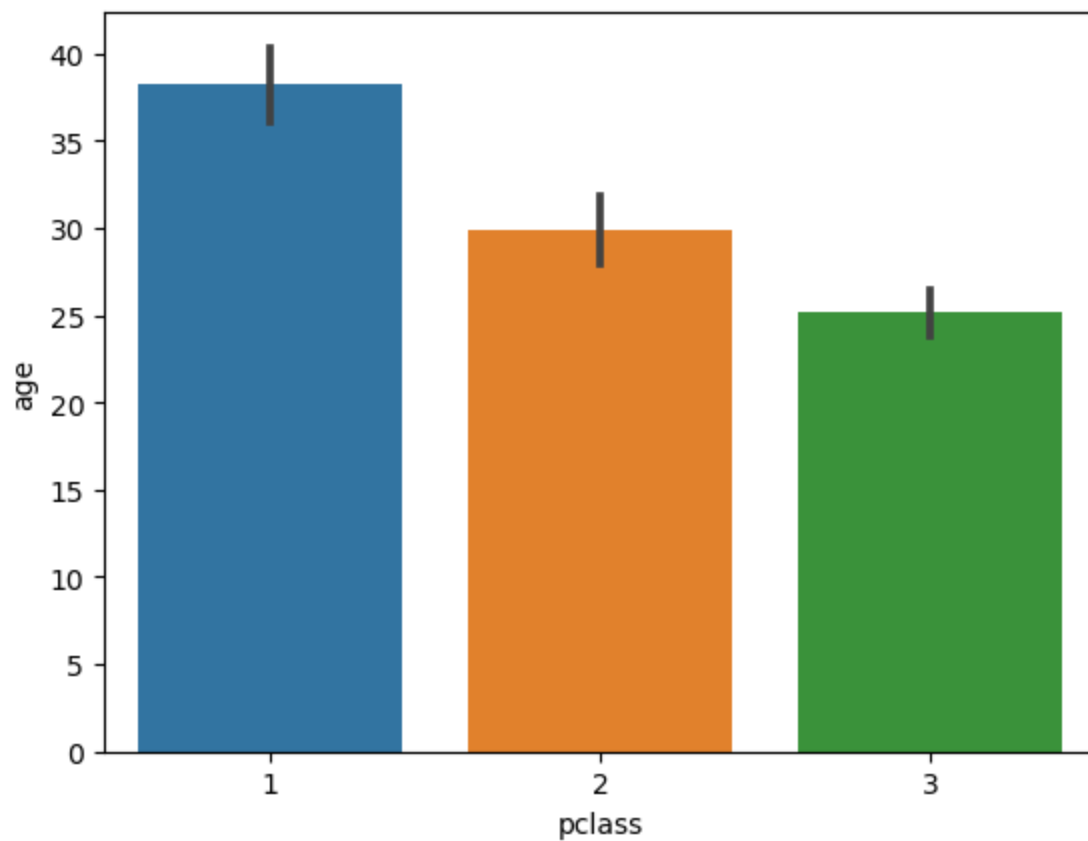
```
In [15]: sns.barplot(data = df, x = 'embarked', y = 'fare')
```

```
Out[15]: <Axes: xlabel='embarked', ylabel='fare'>
```



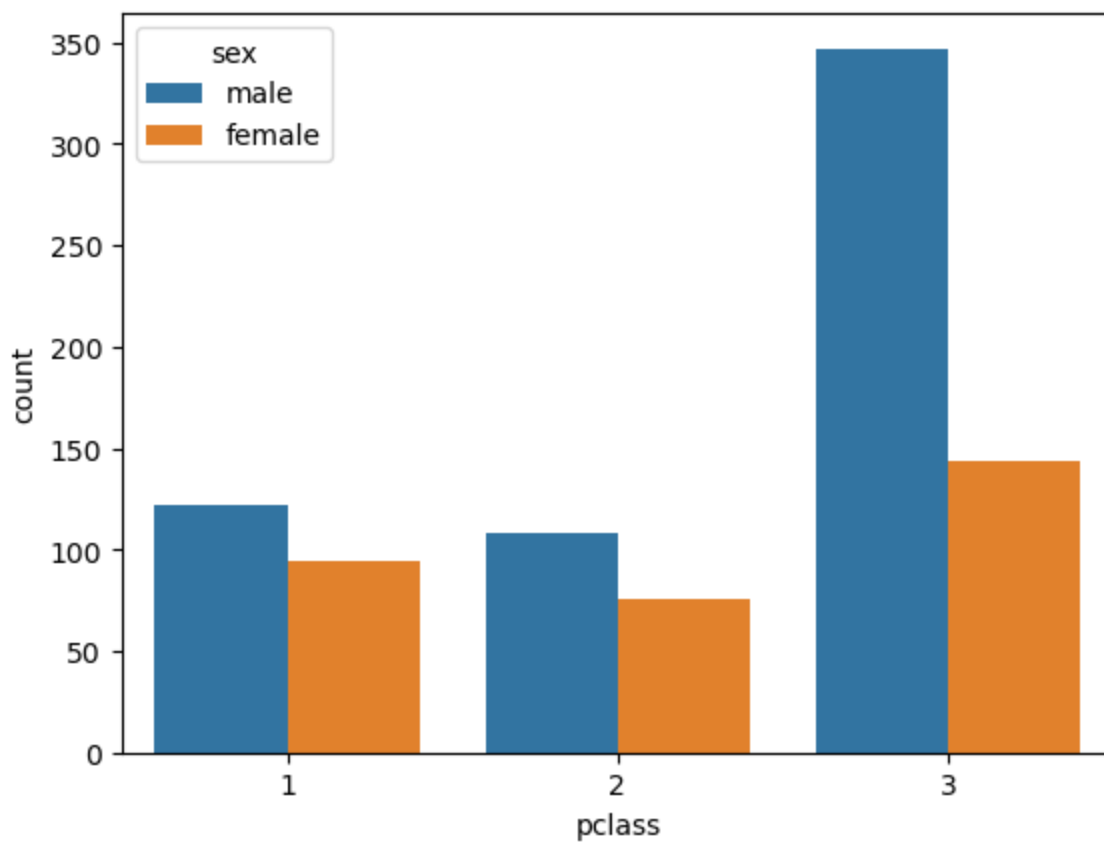
```
In [16]: sns.barplot(data = df, x = 'pclass', y = 'age')
```

```
Out[16]: <Axes: xlabel='pclass', ylabel='age'>
```



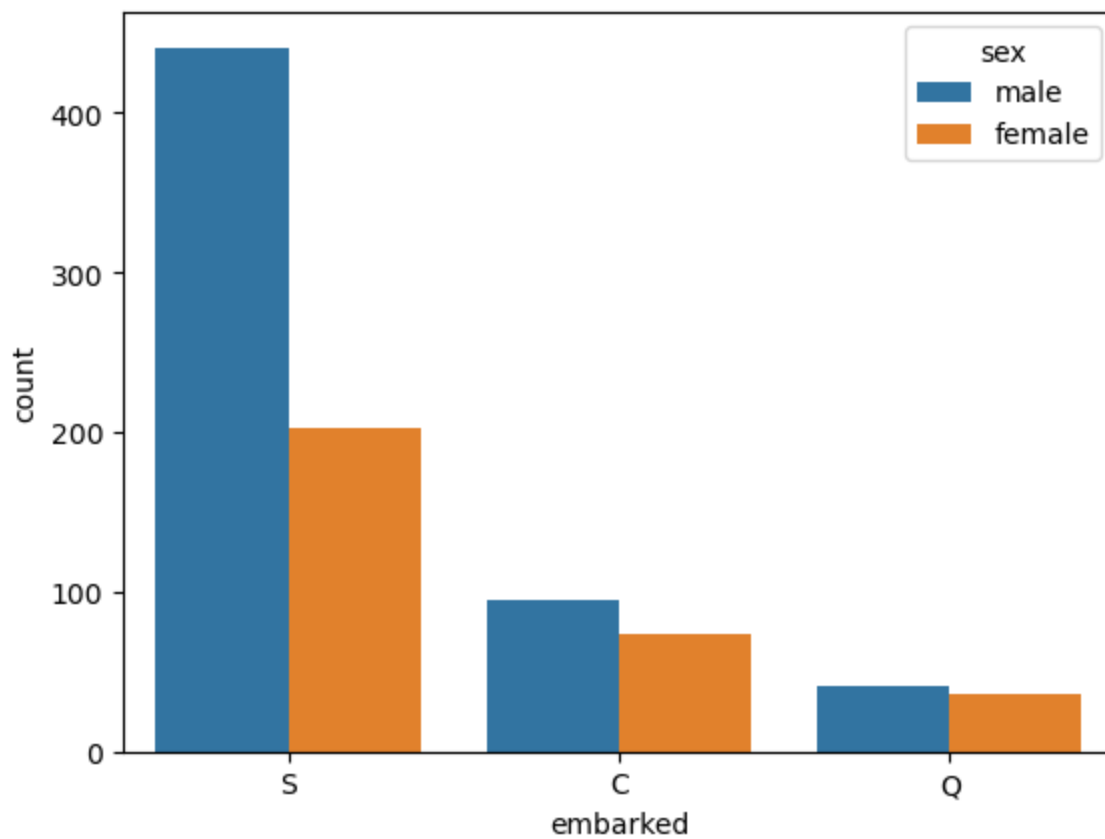
```
In [17]: sns.countplot(x = df['pclass'], hue = df['sex'])
```

```
Out[17]: <Axes: xlabel='pclass', ylabel='count'>
```



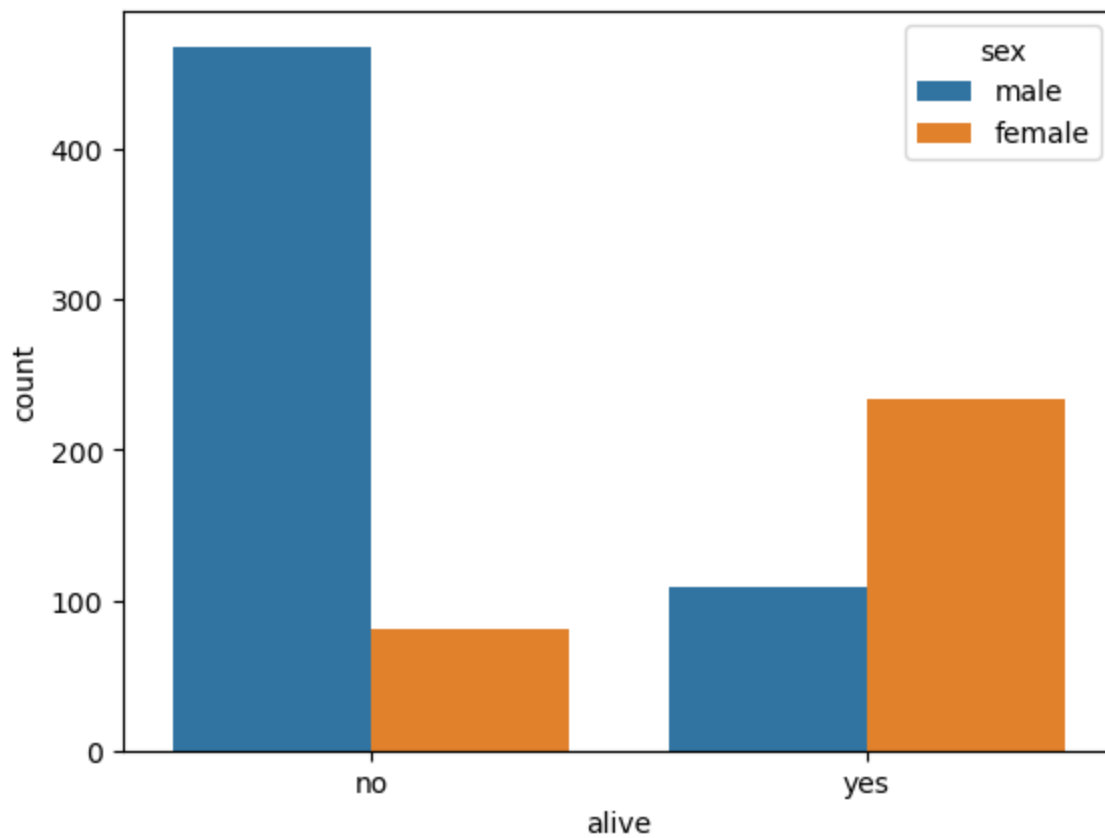
```
In [18]: sns.countplot(x = df['embarked'], hue = df['sex'])
```

```
Out[18]: <Axes: xlabel='embarked', ylabel='count'>
```



```
In [19]: sns.countplot(x = df['alive'], hue = df['sex'])
```

```
Out[19]: <Axes: xlabel='alive', ylabel='count'>
```

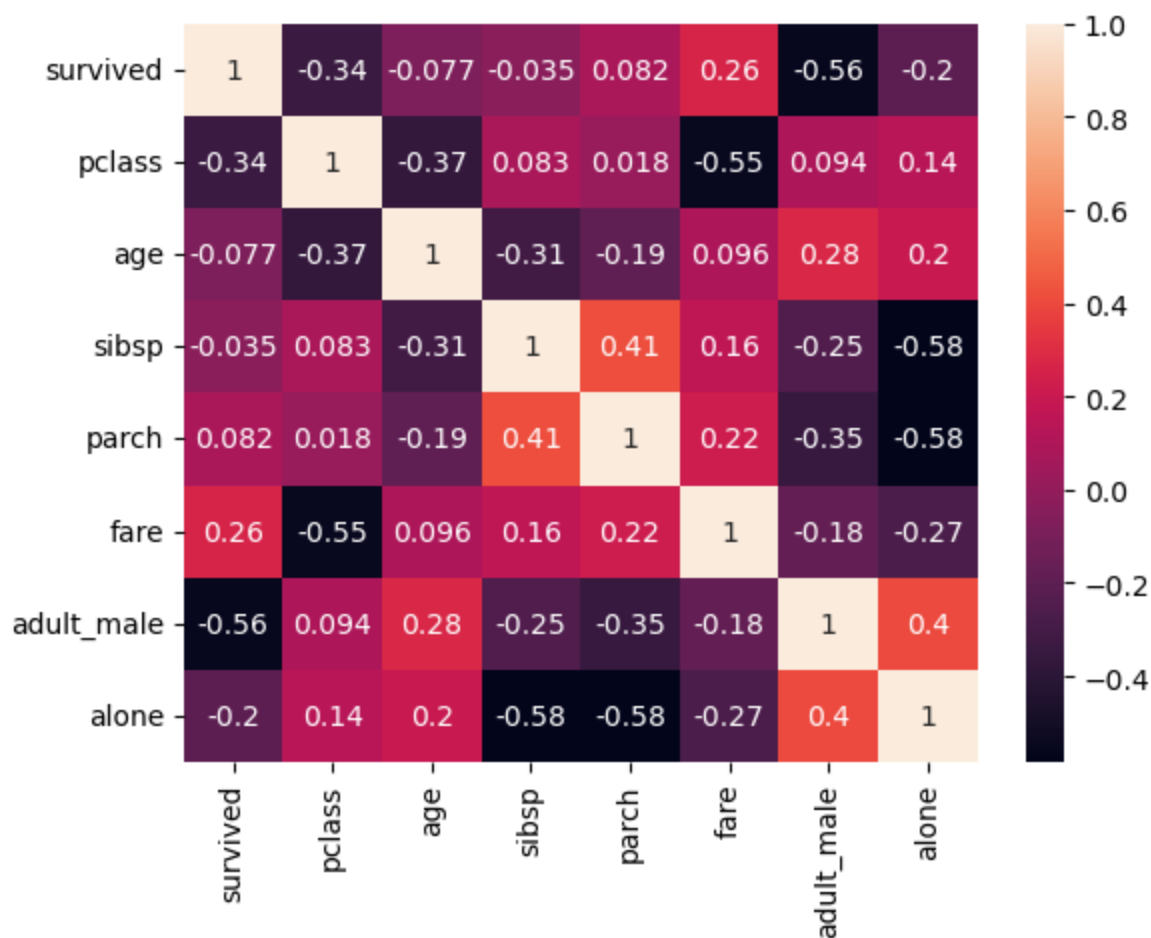


```
In [20]: # multivariate analysis
```

```
In [21]: sns.heatmap(df.corr(numeric_only=True), annot = True)
```

```
Out[21]: <Axes: >
```





```
In [22]: # descriptive stats
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

```
In [23]: # handling missing values
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)
```

```
In [25]: #for age column we will fill with the average
df['age'] = df['age'].fillna(df['age'].mean())
```

```
In [26]: #only 203 records have valid values for deck column so we will drop that

df.drop(['deck'], axis = 1,inplace = True)
```

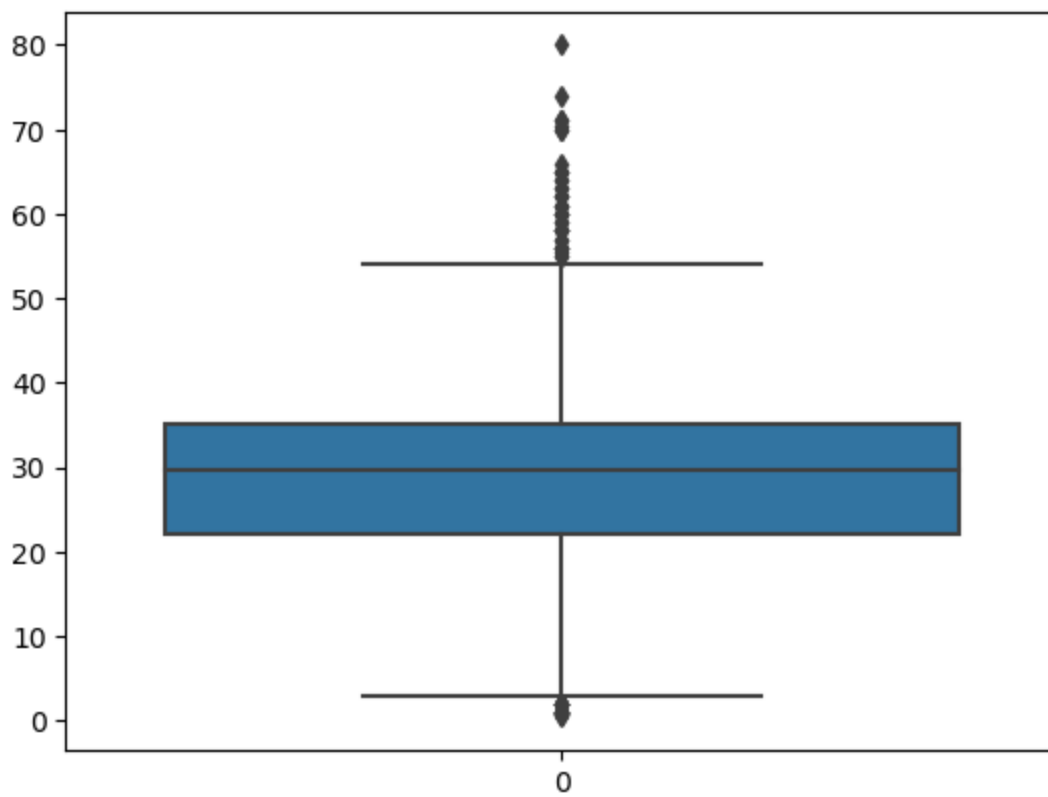
```
In [27]: df.isnull().sum()
```

```
Out[27]: 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 [28]: #checking for outliers and replacing them
```

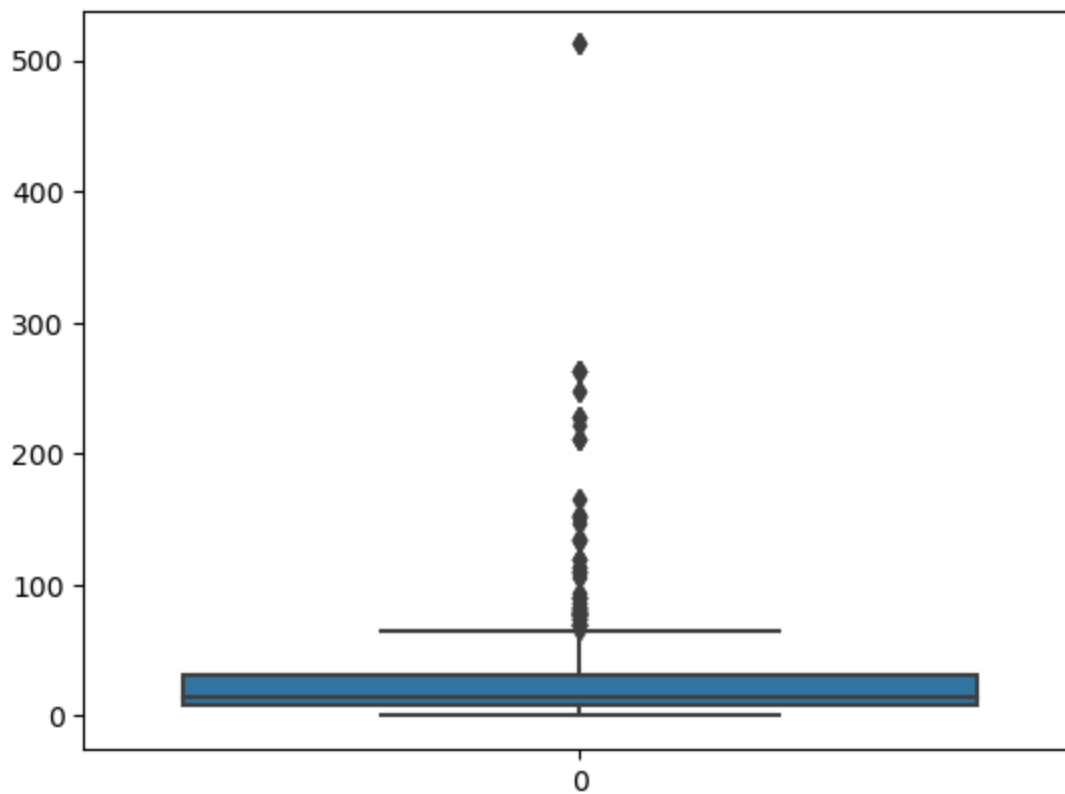
```
In [29]: sns.boxplot(df['age'])
```

```
Out[29]: <Axes: >
```



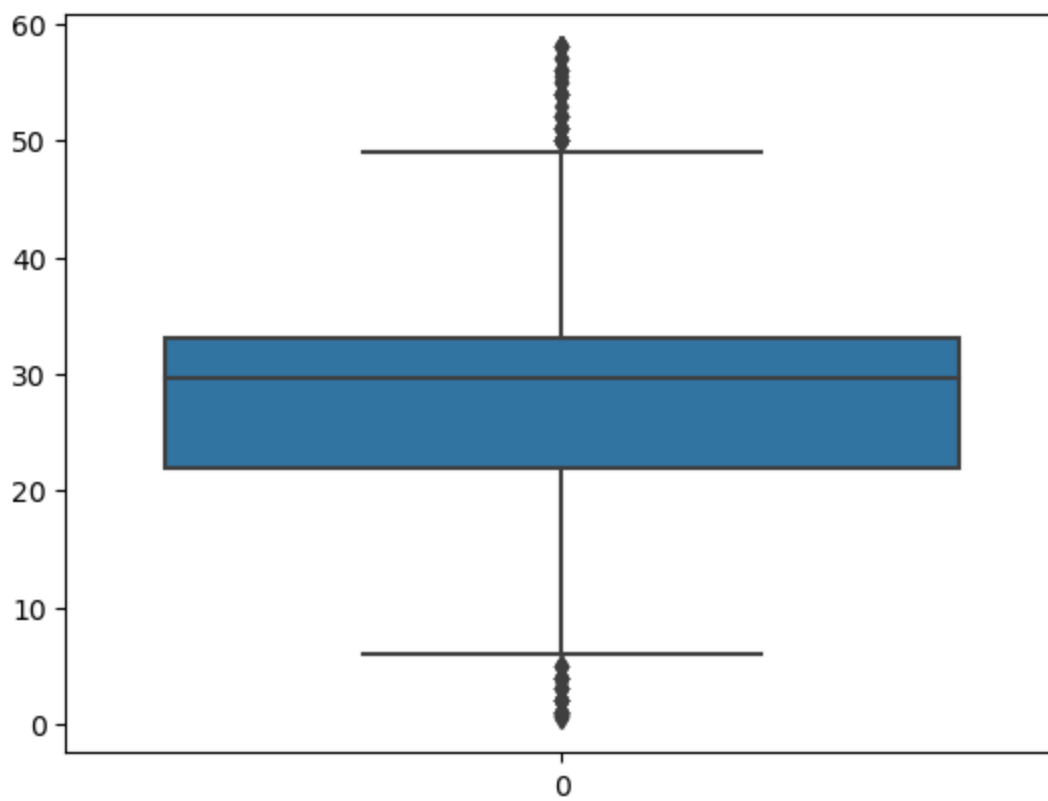
```
In [30]: sns.boxplot(df['fare'])
```

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



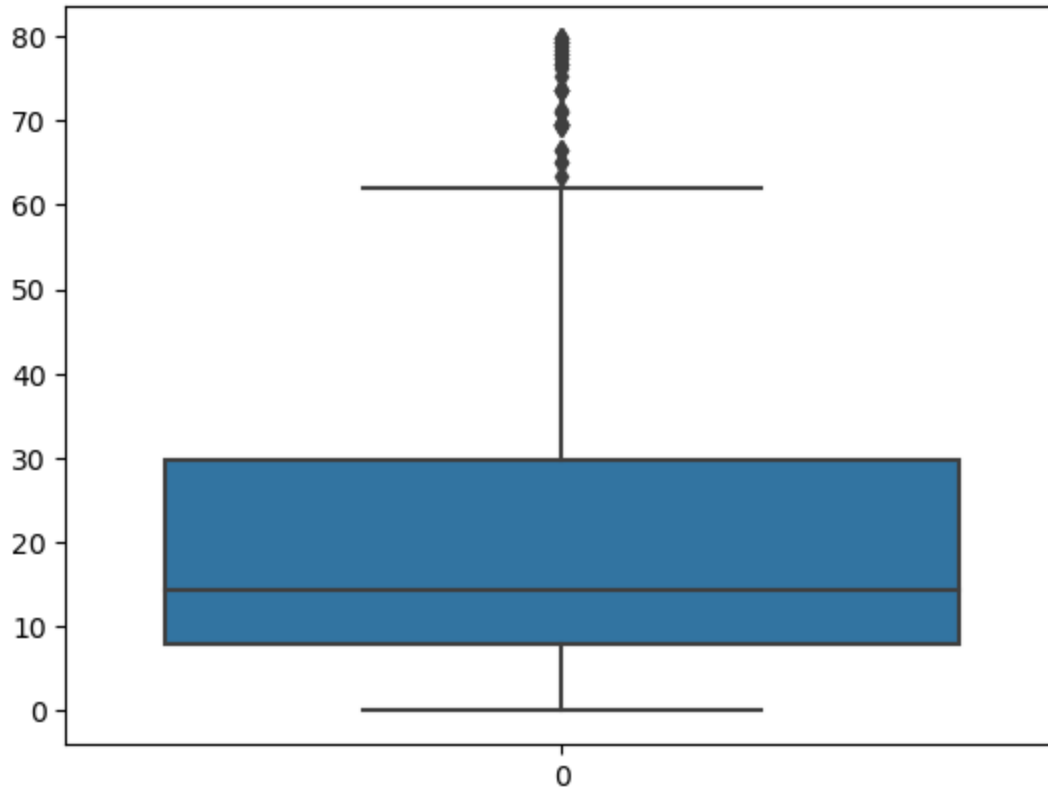
```
In [31]: median_age = df['age'].median()
df["age"] = np.where(df["age"] > 58, median_age, df['age'])
sns.boxplot(df['age'])
```

```
Out[31]: <Axes: >
```



```
In [32]: median_fare = df['fare'].median()
df["fare"] = np.where(df["fare"] > 80, median_age, df['fare'])
sns.boxplot(df['fare'])
```

Out[32]: <Axes: >



```
In [33]: #Check for Categorical columns and perform encoding.
from sklearn.preprocessing import OneHotEncoder
```

```
In [34]: encoding = pd.get_dummies(df, columns = ['sex', 'embarked', 'class', 'who', 'adult_male', 'a
```

```
In [35]: encoding.head()
```

```
Out[35]:
```

	survived	pclass	age	sibsp	parch	fare	alive	sex_female	sex_male	embarked_C	...	who_child	who_m
0	0	3	22.0	1	0	7.2500	no	0	1	0	...	0	
1	1	1	38.0	1	0	71.2833	yes	1	0	1	...	0	
2	1	3	26.0	0	0	7.9250	yes	1	0	0	...	0	
3	1	1	35.0	1	0	53.1000	yes	1	0	0	...	0	
4	0	3	35.0	0	0	8.0500	no	0	1	0	...	0	

5 rows × 25 columns

```
In [36]: #Split the data into dependent and independent variables.
df.columns
```

```
Out[36]: Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
              'embarked', 'class', 'who', 'adult_male', 'embark_town', 'alive',
              'alone'],
              dtype='object')
```

```
In [37]: # independent variables
X = encoding.drop(['survived', 'alive'], axis = 1)
X.head()
```

```
Out[37]:
```

	pclass	age	sibsp	parch	fare	sex_female	sex_male	embarked_C	embarked_Q	embarked_S	...	who_chi
0	3	22.0	1	0	7.2500	0	1	0	0	1	...	
1	1	38.0	1	0	71.2833	1	0	1	0	0	...	
2	3	26.0	0	0	7.9250	1	0	0	0	1	...	
3	1	35.0	1	0	53.1000	1	0	0	0	1	...	
4	3	35.0	0	0	8.0500	0	1	0	0	1	...	

5 rows × 23 columns

```
In [38]: y = df[['survived', 'alive']]
y.head()
```

```
Out[38]:
```

	survived	alive
0	0	no
1	1	yes
2	1	yes
3	1	yes
4	0	no

```
In [39]: # Scaling the independent variables
```

```
In [40]: from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_std = scaler.fit_transform(X)
```

```
In [41]: x_std
```

```
Out[41]: array([[ 0.82520863, -0.57985934,  0.43135024, ..., -0.48271079,
                -0.30794088,  0.61679395],
                [-1.57221121,  0.83108889,  0.43135024, ...,  2.07163382,
                -0.30794088, -1.62128697],
                [ 0.82520863, -0.22712228, -0.47519908, ..., -0.48271079,
                -0.30794088,  0.61679395],
                ...,
                [ 0.82520863,  0.09405298,  0.43135024, ..., -0.48271079,
                -0.30794088,  0.61679395],
                [-1.57221121, -0.22712228, -0.47519908, ...,  2.07163382,
                -0.30794088, -1.62128697],
                [ 0.82520863,  0.3019833 , -0.47519908, ..., -0.48271079,
                3.24737656, -1.62128697]])
```

```
In [42]: #train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=0
```

```
In [ ]:
```