8 - Data Visualization

In [1]:

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

In [2]:

```
df=sns.load_dataset("titanic")
```

In [3]:

```
df.head()
```

Out[3]:

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

In [4]:

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

Out[4]:

```
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 [5]:

```
df.shape
```

Out[5]:

```
(891, 15)
```

In [6]:

```
df.info()
```

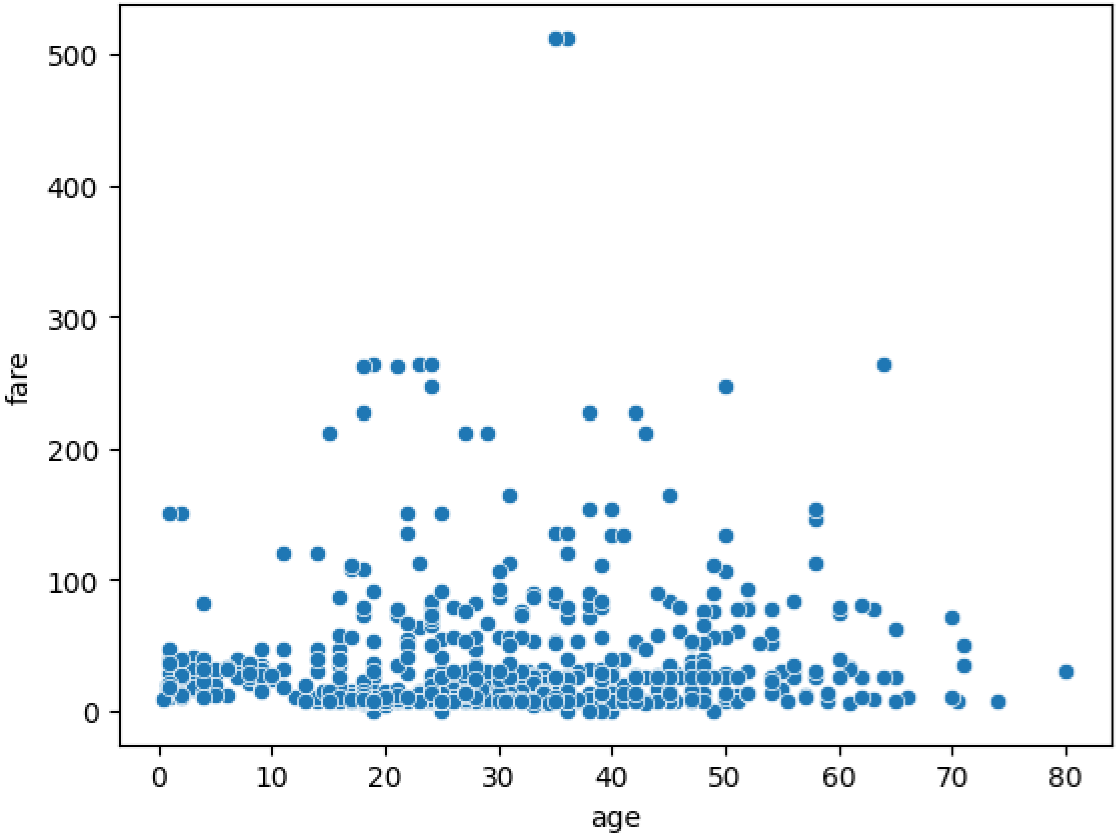
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    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    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
```

Sactter Plot

To check Outliers in the data

In [7]:
```
sns.scatterplot(data=df,x='age',y='fare')
```

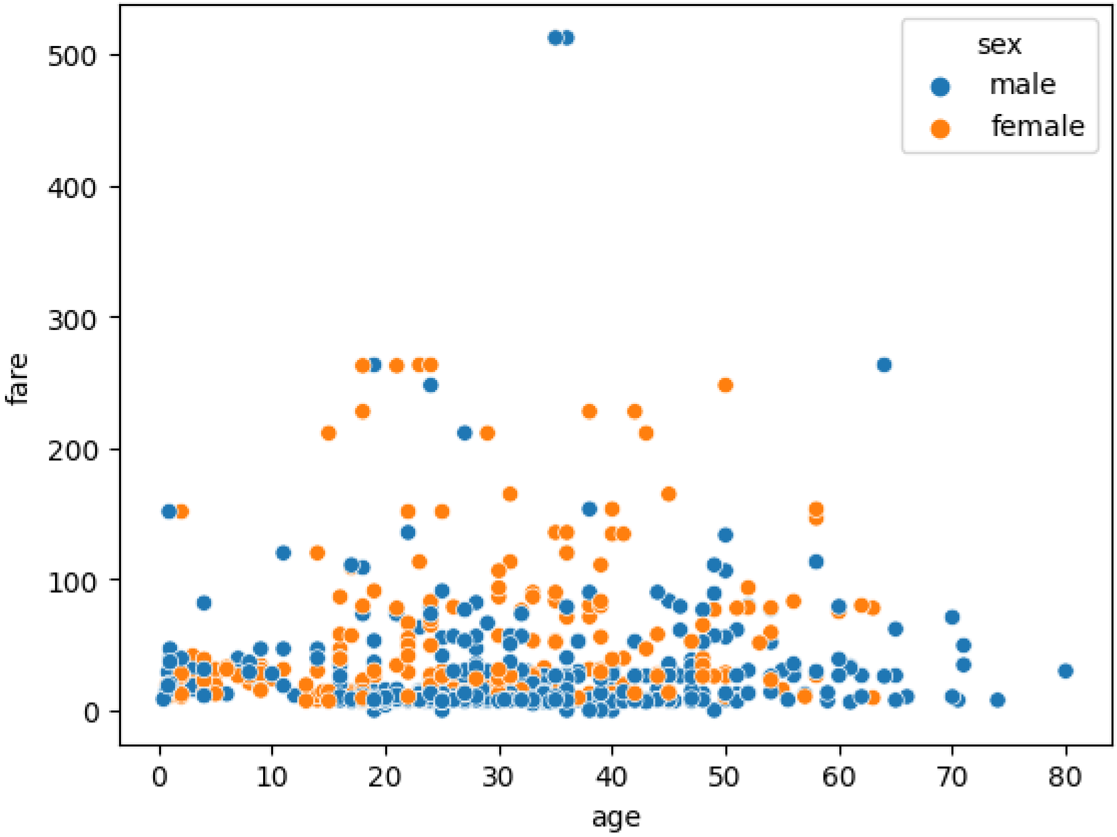
Out[7]:
```
<AxesSubplot: xlabel='age', ylabel='fare'>
```



In above graph, we can observe the outliers of column 'Fare' with respect to the column 'Age'.

In [8]:
```
sns.scatterplot(data=df,x='age',y='fare',hue='sex')
```

Out[8]:
```
<AxesSubplot: xlabel='age', ylabel='fare'>
```



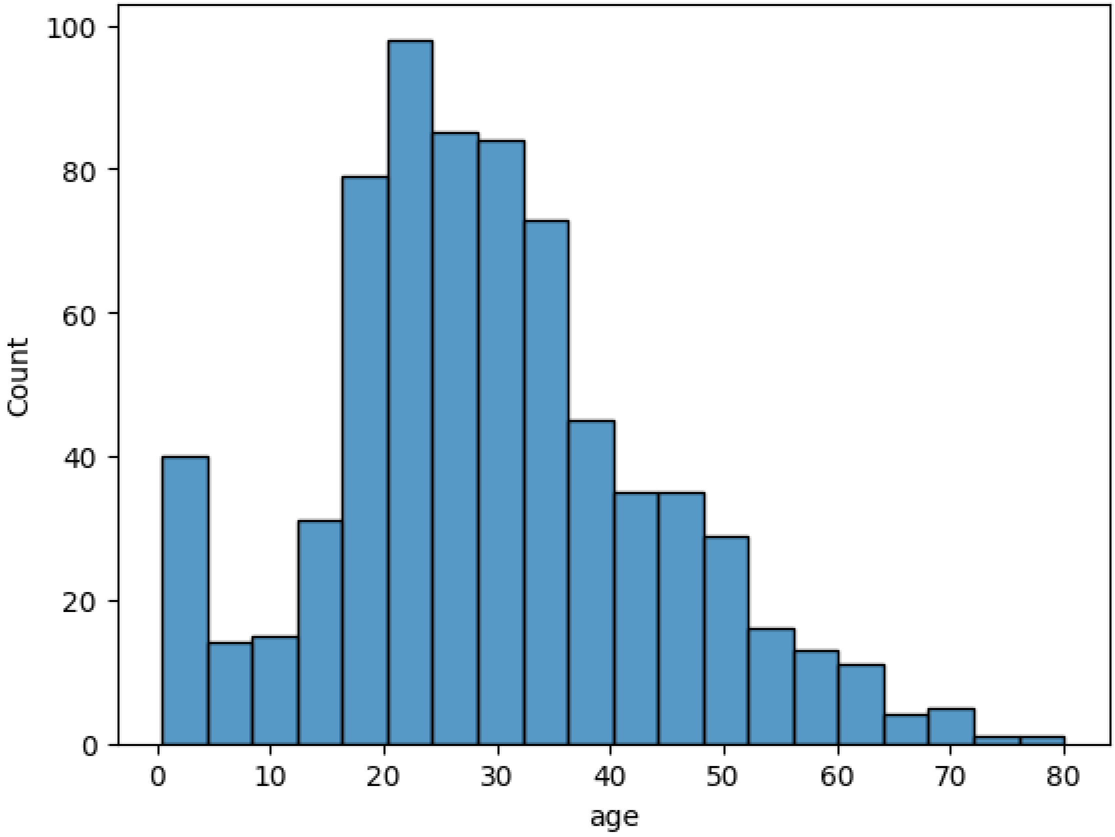
In above graph, we can observe the outliers of column 'Fare' with respect to the column 'Age'. As hue is 'Gender' it is also segregated w.r.t to gender.

Histplot

Histograms are visualization tools that represent the distribution of a set of continuous data. In a histogram, the data is divided into a set of intervals or bins (usually on the x-axis) and the count of data points that fall into each bin corresponding to the height of the bar above that bin. These bins may or may not be equal in width but are adjacent (with no gaps).

```
In [9]: sns.histplot(data=df,x='age')
```

```
Out[9]: <AxesSubplot: xlabel='age', ylabel='Count'>
```



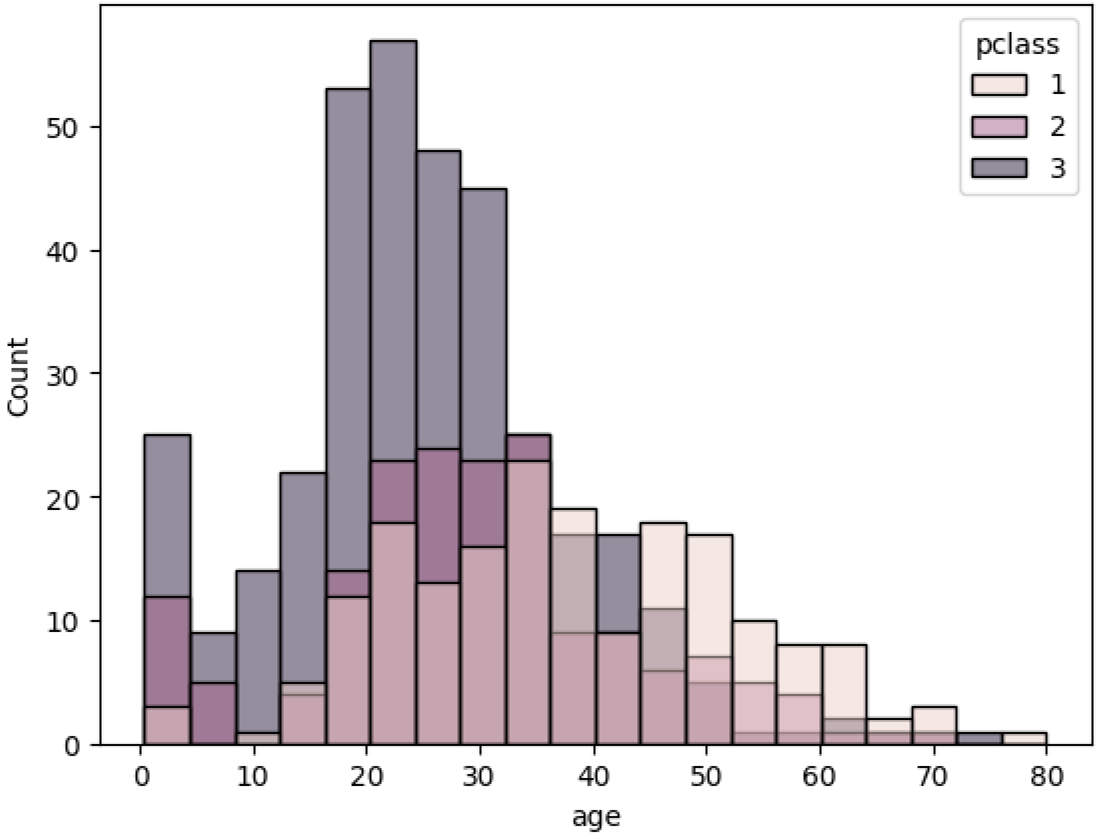
Inference: Histplot is used for continuos data. Above graph gives count per age group.

In [10]:

```
sns.histplot(data=df,x='age',hue='pclass')
```

Out[10]:

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



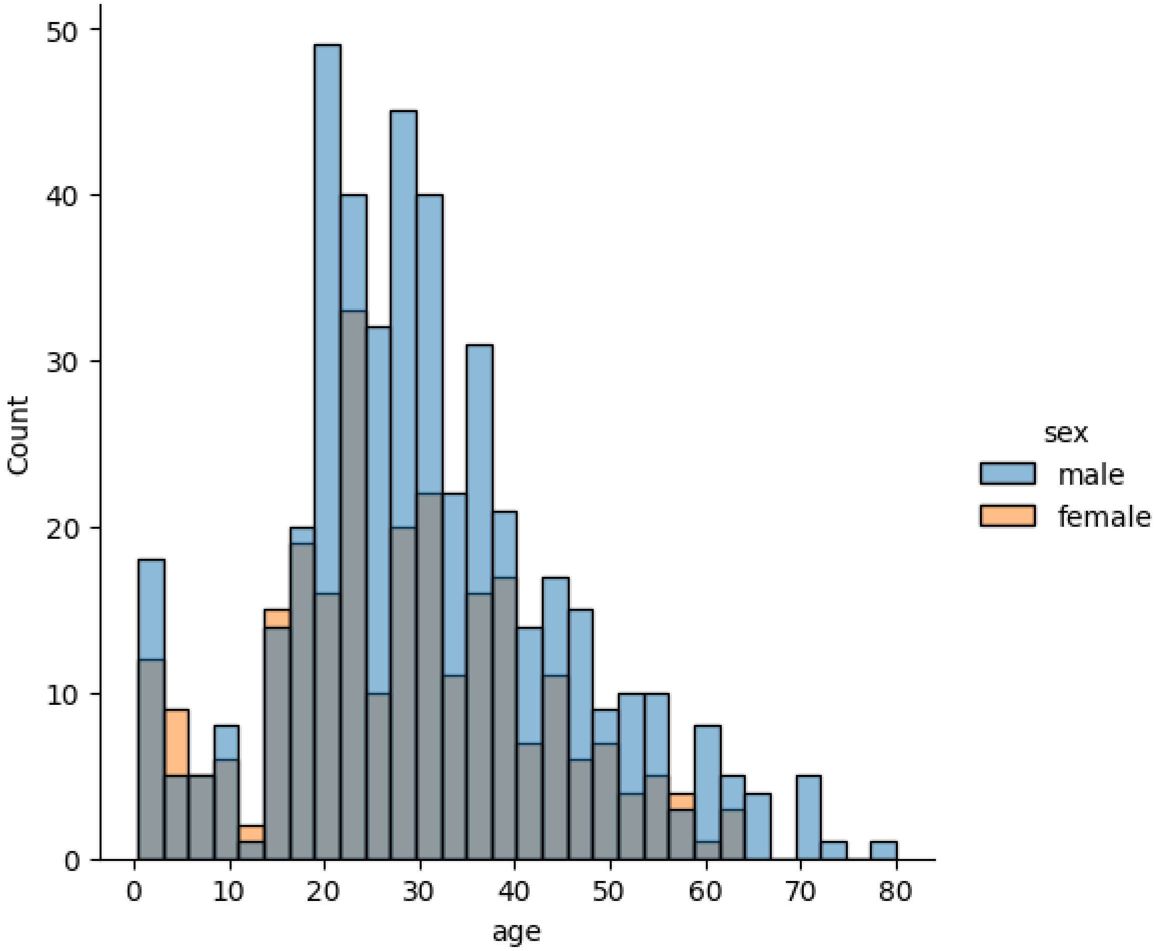
Above graph gives count of Age and w.r.t pclass column

In [11]:

```
sns.displot(data=df,x='age',hue='sex',bins=30)
```

Out[11]:

```
<seaborn.axisgrid.FacetGrid at 0x1fd7174a380>
```



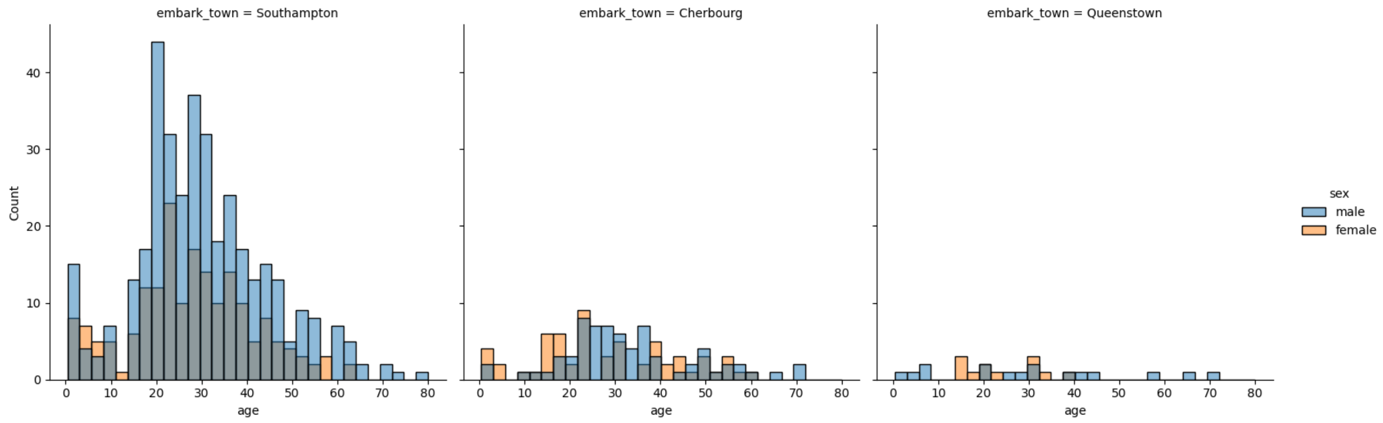
In above graph,Age wise count w.r.t gender

In [12]:

```
sns.displot(data=df,x='age',hue='sex',bins=30,col='embark_town')
```

Out[12]:

```
<seaborn.axisgrid.FacetGrid at 0x1fd71720760>
```



In above graph, we can see Age wise count w.r.t gender for each town.

With this count we can have clear idea of the people from the different town

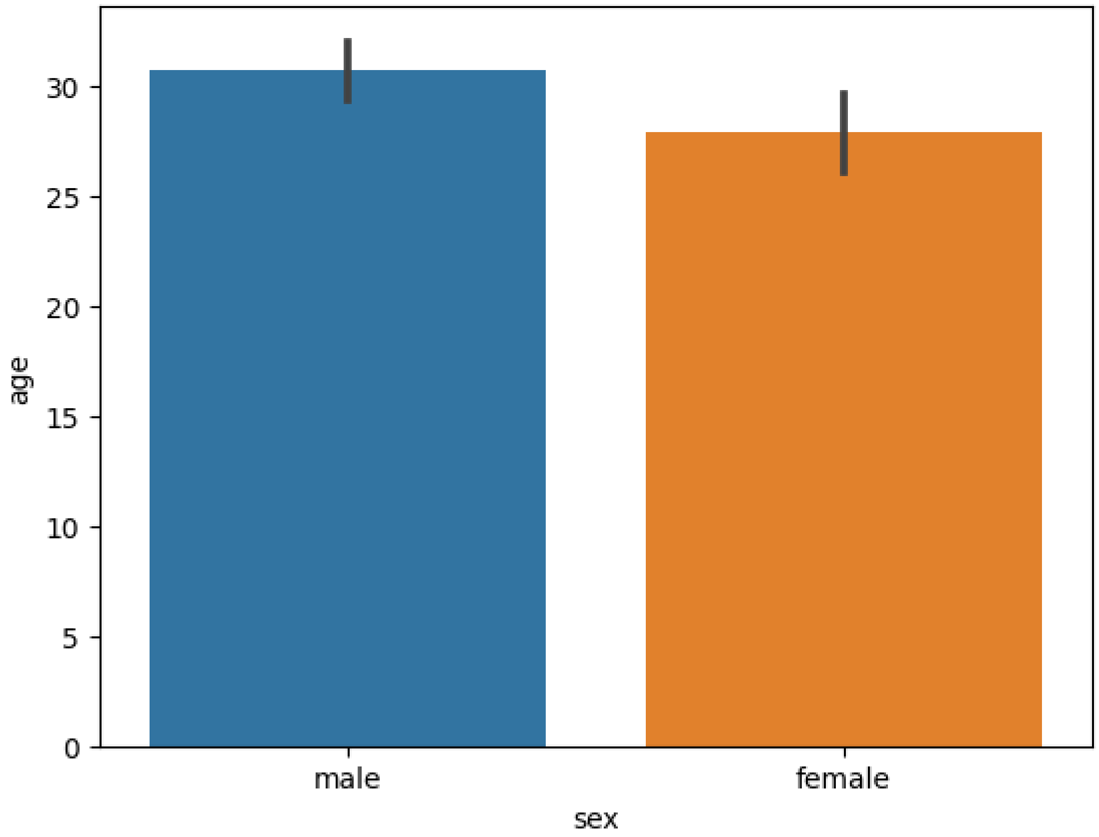
Bar Plot

In [13]:

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

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

Out[13]:



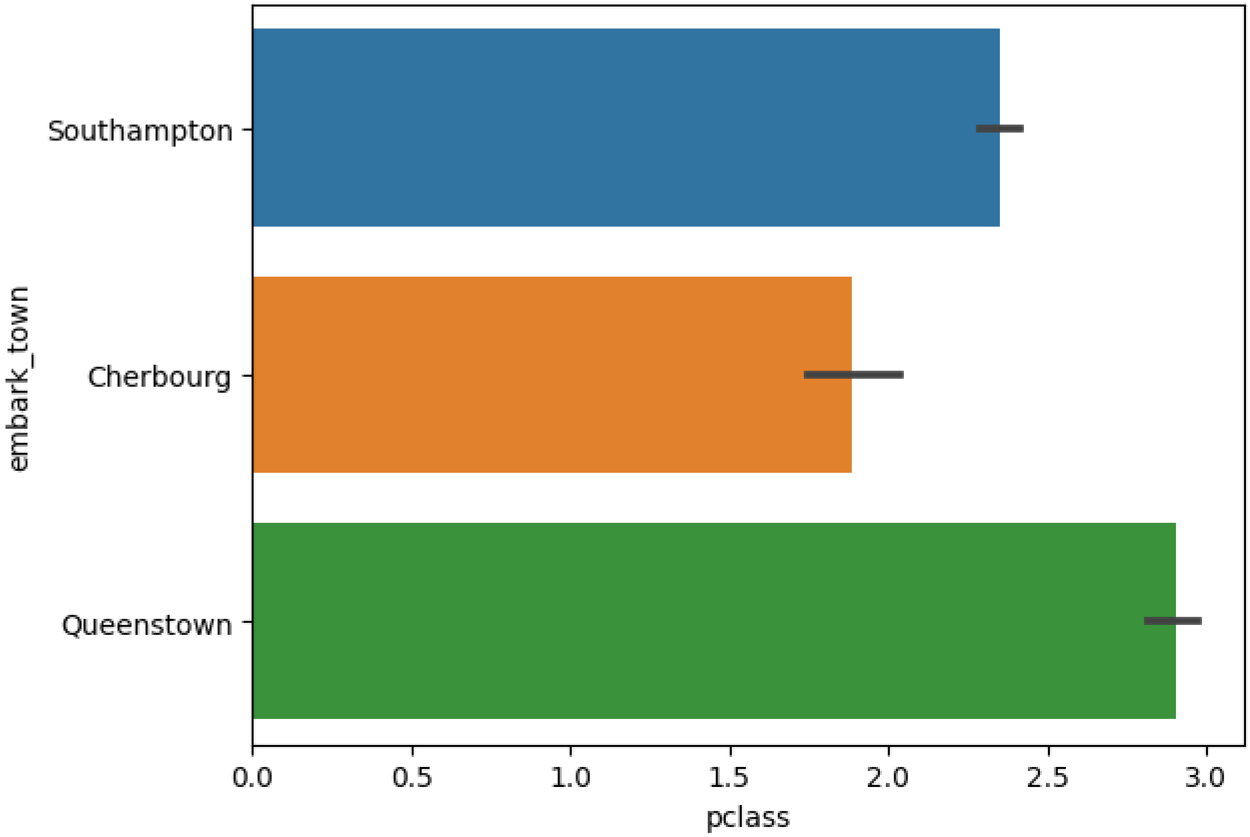
Inference: Above graph gives the age with respect to gender.

In [14]:

```
sns.barplot(x='pclass',y='embark_town',data=df,orient='h')
```

Out[14]:

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



Above graph gives pclass and town wise relation

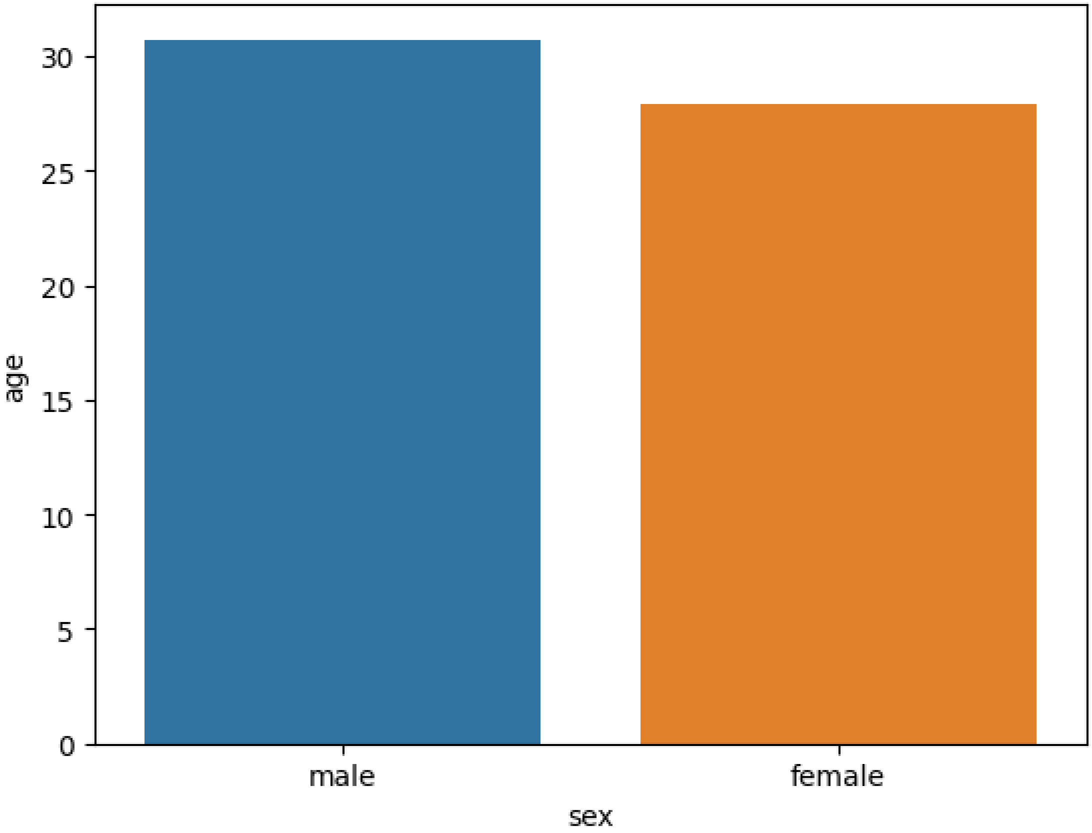
```
In [15]: sns.barplot(x='sex',y='age',data=df,ci=None)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_1588\3046591102.py:1: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x='sex',y='age',data=df,ci=None)
```

```
Out[15]: <AxesSubplot: xlabel='sex', ylabel='age'>
```

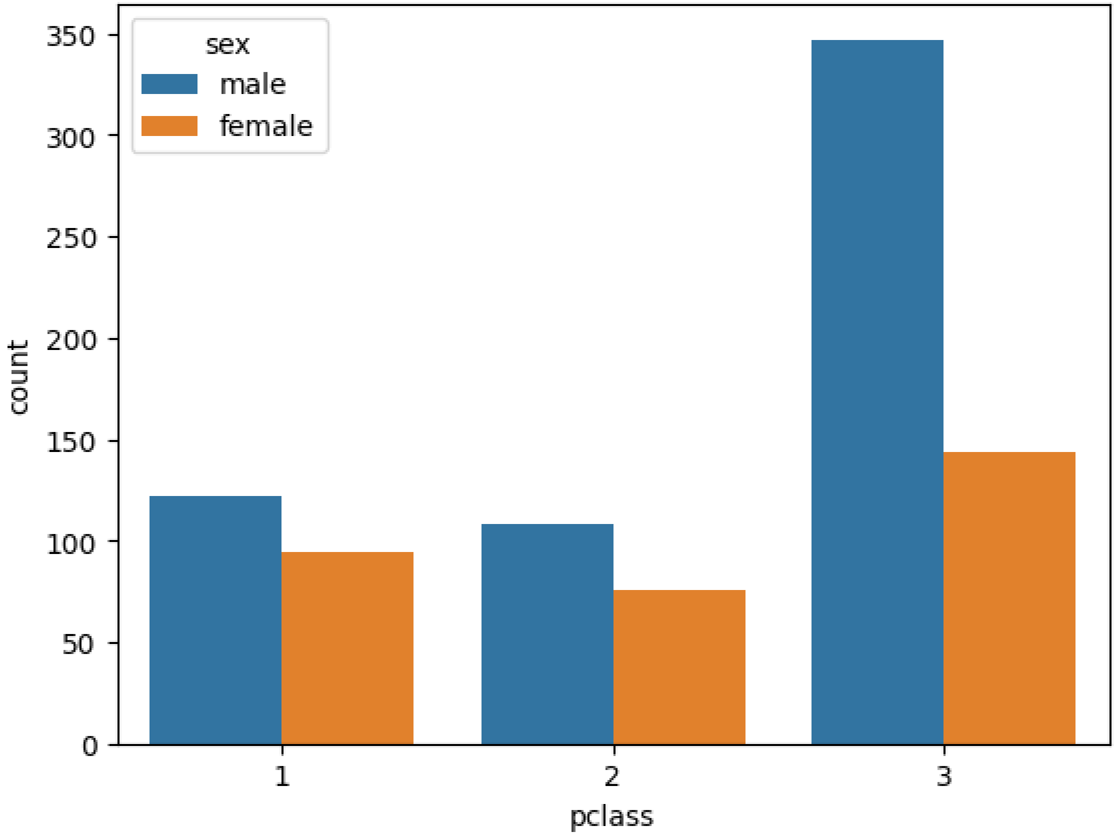


Count Plot

The count plot is similar to the bar plot, however it displays the count of the categories in a specific column. For instance, if we want to count the number of males and women passenger we can do so using count plot as follows:

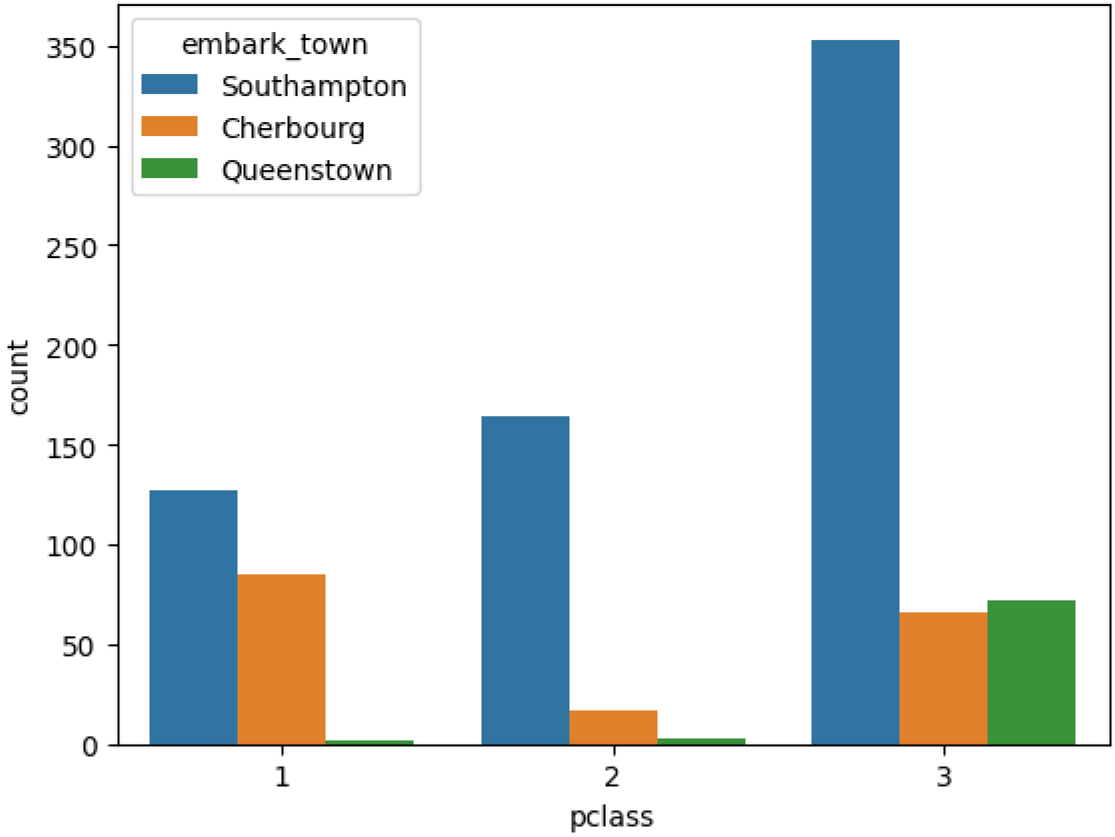
```
In [16]: sns.countplot(x='pclass',hue='sex',data=df)
```

```
Out[16]: <AxesSubplot: xlabel='pclass', ylabel='count'>
```



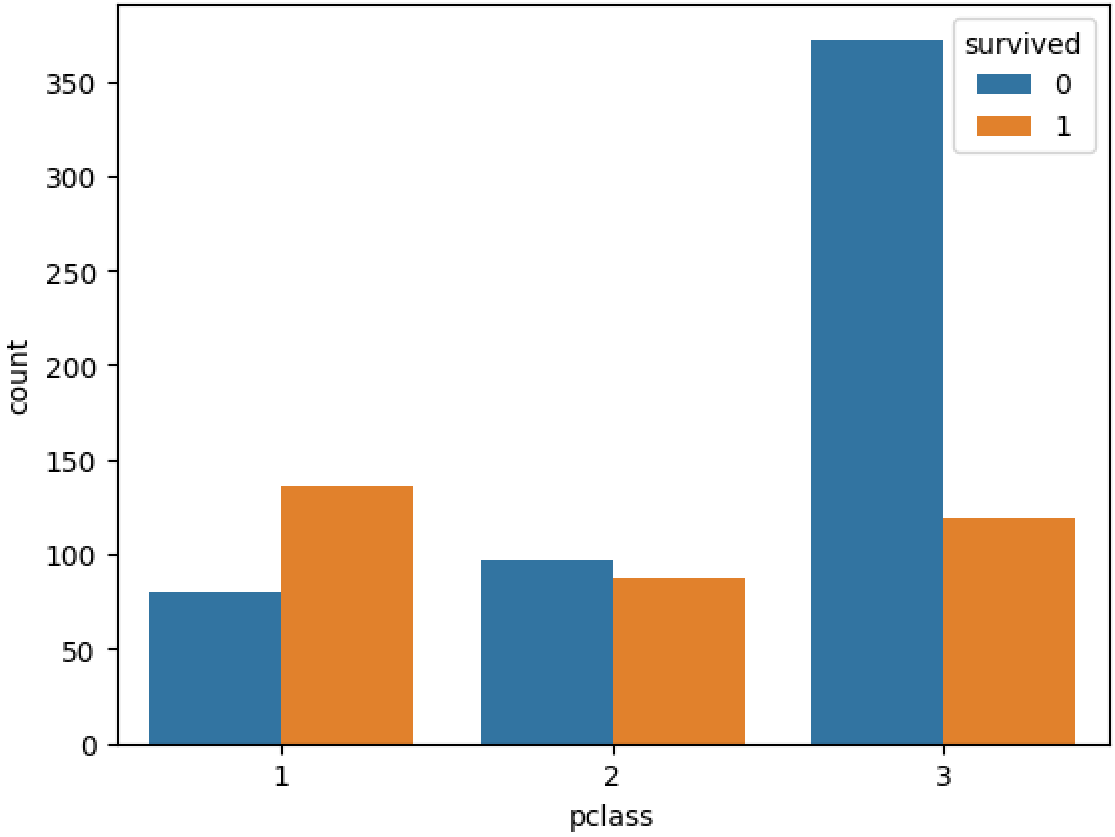
In [17]:
```
sns.countplot(x='pclass',hue='embark_town',data=df)
```

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



In [18]:
```
sns.countplot(x='pclass',hue='survived',data=df)
```

Out[18]:
```
<AxesSubplot: xlabel='pclass', ylabel='count'>
```



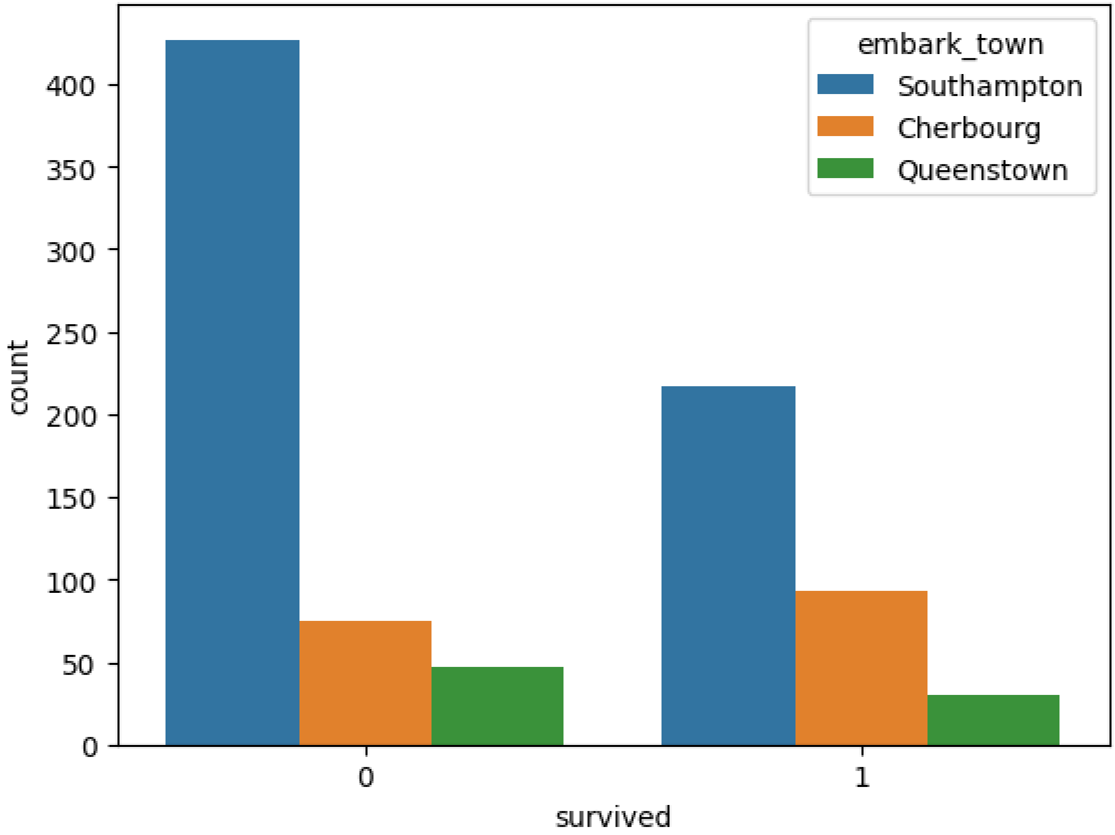
- We can observe that people from pclass 1 have survived more then pclass 3
- Death of pclass 3 people is more

In [19]:

```
sns.countplot(x='survived',hue='embark_town',data=df)
```

Out[19]:

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



- For above graph, survival of people town wise.
- It is town wise count.

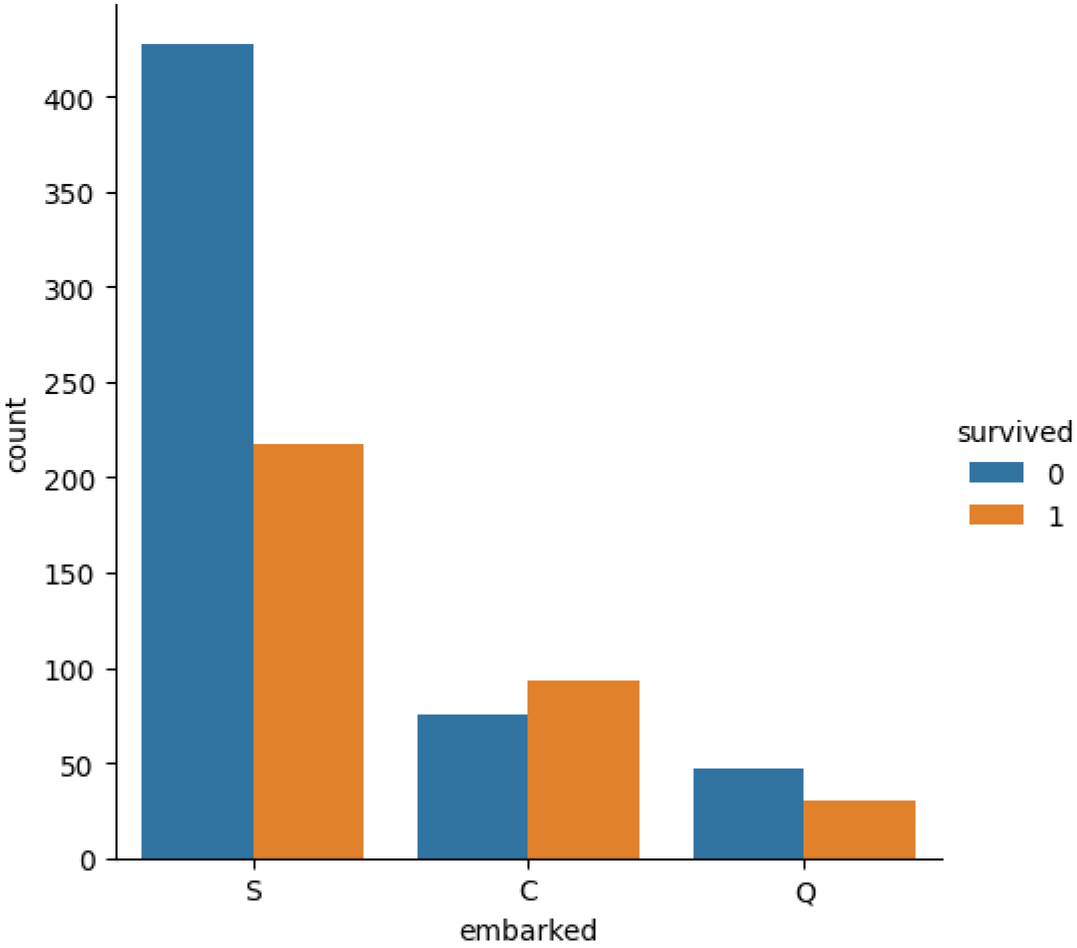
Catplot

In [20]:

```
sns.catplot(x='embarked',hue='survived',data=df,kind='count')
```

Out[20]:

```
<seaborn.axisgrid.FacetGrid at 0x1fd73aa7400>
```

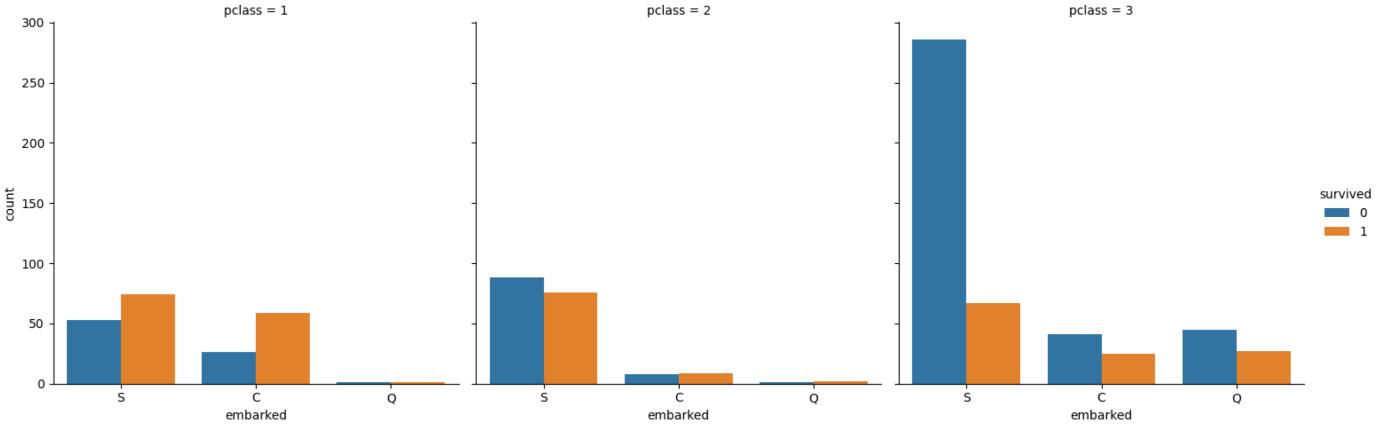


In [21]:

```
sns.catplot(x='embarked',hue='survived',data=df,kind='count',col='pclass')
```

Out[21]:

```
<seaborn.axisgrid.FacetGrid at 0x1fd7163a230>
```

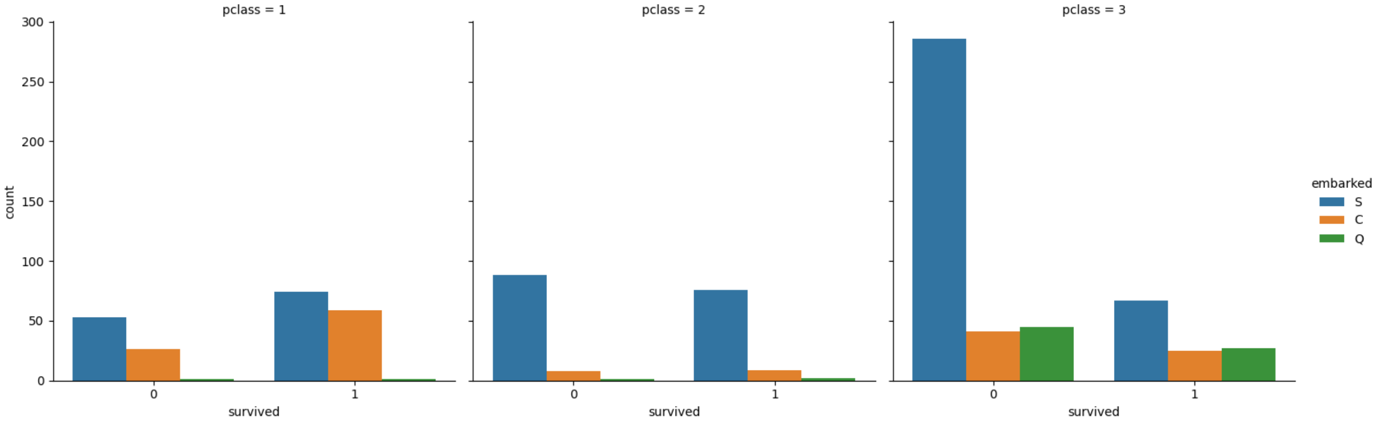


In [22]:

```
sns.catplot(x='survived',hue='embarked',data=df,kind='count',col='pclass')
```

Out[22]:

```
<seaborn.axisgrid.FacetGrid at 0x1fd73488eb0>
```



Cat plot is used to plot different graphs. The kind parameter can be changed according to the requirement

In [ ]: