Date: 22-02-2024

EXPERIMENT NO. 5

AIM:

- To study and analyse anova for iris dataset
- To study data visualization for given datasets in Python by using seaborn and matplotlib

SOFTWARE USED: Jupyter Notebook

THEORY:

ANOVA, or Analysis of Variance, is a statistical method used to analyze the differences among means of two or more groups. It assesses whether the means of different groups are statistically significantly different from each other by comparing the variation within groups to the variation between groups. ANOVA helps determine if there is a significant difference in means and if any observed differences are not due to random variation.

The formula for the one-way ANOVA (which is one of the most common forms of ANOVA) can be broken down into several components. Here's a simplified explanation of the formula:

1. Total Sum of Squares (SST):

$$SST = \sum_{i=1}^{k} \sum_{j=1}^{ni} (X_{ij} - \overline{X})^2$$

Where:

- k is the number of groups or treatments.
- n_i is the number of observations in group i
- X_{ij} is the j^{th} observation in group i
- $\overline{X}$ is the overall mean of all observations.

2. Between-Group Sum of Squares (SSB)

$$SSB = \sum_{i=1}^{k} n_i (\overline{X_i} - \overline{X})^2$$

Where:

- $\overline{X_i}$ is the mean of group i
- $\overline{X}$ is the overall mean.

3. Within-Group Sum of Squares (SSW)

$$SSW = \sum_{i=1}^{k} \sum_{j=1}^{ni} (X_{ij} - \overline{X_i})^2$$

Where:

- $\overline{X}_i$ is the mean of group i
- X_{ij} is the j[th] observation in group i

Then, the ANOVA F-statistic is calculated as:

$$F = \frac{SSB/(k-1)}{SSW/(N-k)}$$

Where:

- N is the total number of observations.

This F-statistic is then compared to a critical value from the F-distribution at a chosen significance level to determine whether to reject the null hypothesis that the means of all groups are equal. If the calculated F-value is greater than the critical value, it suggests that there are significant differences among the group means.

Data visualization is the graphical representation of data and information. It uses visual elements such as charts, graphs, and maps to help users understand complex datasets more easily and quickly. The primary goal of data visualization is to communicate information clearly and effectively, making it easier to identify patterns, trends, relationships, and outliers within the data.

Matplotlib is a comprehensive plotting library for Python widely used for creating static, animated, and interactive visualizations in various formats. It provides a wide range of plotting functions and tools for generating high-quality plots for data analysis, scientific computing, and visualization.
Some important functions and features of Matplotlib:

1. Creating Basic Plots:
   - `plt.plot()`: This function is used to create line plots. It takes two arrays of data, one for the x-axis values and another for the y-axis values.
   - `plt.scatter()`: Used to create scatter plots, which are useful for visualizing relationships between two variables.
   - `plt.bar()`: Creates vertical bar plots to represent categorical data or compare values across different categories.
   - `plt.hist()`: Generates histograms to visualize the distribution of numerical data.

2. Customizing Plot Appearance:
   - `plt.xlabel()`, `plt.ylabel()`: Set labels for the x-axis and y-axis, respectively.
   - `plt.title()`: Set the title of the plot.
   - `plt.legend()`: Add a legend to the plot to identify different data series.

- `plt.xlim()`, `plt.ylim()`: Set the limits for the x-axis and y-axis, respectively.
- `plt.xticks()`, `plt.yticks()`: Customize the tick marks and labels on the axes.

3. Subplots:
- `plt.subplots()`: Create a grid of subplots within a single figure, allowing multiple plots to be displayed together.
- `ax.set_title()`, `ax.set_xlabel()`, `ax.set_ylabel()`: Set titles, x-axis labels, and y-axis labels for subplots.

4. Annotations and Text:
- `plt.text()`: Add text at specified locations on the plot.
- `plt.annotate()`: Annotate points with arrows and text to highlight specific features of the data.

5. Saving and Displaying Plots:
- `plt.savefig()`: Save the current figure to a file in various formats such as PNG, PDF, SVG, etc.
- `plt.show()`: Display the current plot. This function is often used at the end of a script or Jupyter Notebook cell to show the plot.

6. Plotting Styles and Color Maps:
- Matplotlib provides various predefined styles and color maps to customize the appearance of plots. Styles can be set using `plt.style.use()`, and color maps are available through `plt.cm`.

These are just a few of the many functions provided by Matplotlib. It's a highly flexible library with extensive documentation and examples, making it suitable for both simple and complex plotting tasks.

Seaborn is a statistical data visualization library built on top of Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics.Seaborn simplifies the process of creating complex statistical plots by providing easy-to-use functions for common statistical visualization tasks. It offers support for visualizing relationships between variables, categorical data, distribution plots, and more.
Some important functions and features of Seaborn:

1. Data Visualization Functions:
- `sns.relplot()`: Creates relational plots such as scatter plots and line plots to visualize the relationship between two variables.
- `sns.catplot()`: Generates categorical plots like bar plots, box plots, and violin plots to visualize the distribution of categorical data.

- `sns.distplot()`, `sns.histplot()`: Creates distribution plots, including histograms and kernel density estimations (KDE), to visualize the distribution of numerical data.
- `sns.pairplot()`: Generates a grid of pairwise plots for all combinations of numerical variables in a dataset, allowing quick exploration of relationships.

2. Statistical Estimation and Summary:
- `sns.pointplot()`, `sns.lineplot()`: Creates line plots with optional estimation and confidence intervals to visualize trends across different groups.
- `sns.boxplot()`, `sns.violinplot()`: Generates box plots and violin plots to visualize the distribution of data within different categories, with optional overlay of individual data points.
- `sns.barplot()`: Creates bar plots to visualize the central tendency of numerical data within different categories, with optional error bars to represent uncertainty.

3. Matrix Plots:
- `sns.heatmap()`: Generates heatmaps to visualize matrix-like data, with colors representing the values of the data points.
- `sns.clustermap()`: Creates a hierarchically-clustered heatmap, allowing patterns and similarities in the data to be easily identified.

4. Customization and Styling:
- `sns.set_style()`: Sets the aesthetic style of the plots, including background color, grid lines, and font scale.
- `sns.set_palette()`: Sets the color palette used for the plots, allowing customization of colors for different elements.
- `sns.set_context()`: Sets the context for the plots, controlling the size of plot elements such as fonts and lines.

5. Regression and Model Visualization:
- `sns.regplot()`, `sns.lmplot()`: Creates scatter plots with fitted regression lines to visualize the relationship between two variables and estimate the regression model.
- `sns.residplot()`: Generates residual plots to visualize the residuals of a regression model, helping to assess the goodness of fit.

Seaborn simplifies the process of creating complex statistical visualizations by providing a concise and intuitive interface, making it a popular choice for data analysis, statistical modeling, and exploratory data visualization tasks.

OUTPUT:

In [1]:
```
#To study and analysis of ANOVA on iris dataset
#To visualize data using seaborn and matplotlib libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets
iris = datasets.load_iris()
```

In [2]:
```
iris_data = pd.DataFrame(iris.data)
```

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

Out[3]:

	0	1	2	3
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [4]:
```
iris['feature_names']
```

Out[4]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [5]:
```
import scipy.stats as stats
stats.f_oneway(iris_data.iloc[:,0],iris_data.iloc[:,1],iris_data.iloc[:,2],iris_data.iloc[:,3])
```

Out[5]:
```
F_onewayResult(statistic=482.91531656927964, pvalue=4.660592480454751e-159)
```

In [6]:
```
#p is almost 0 so we reject null hypothesis
#H0 = length is same of petal and sepal
#therefore length is not same
```

In [7]:
```
#train dataset
train = pd.read_csv('train.csv')
```

In [8]:
```
train.head()
```

Out[8]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [9]:
```
mtcars = pd.read_csv('mtcars.csv')
```

In [10]:
```
mtcars.head()
```

In [10]:
```
mtcars.head()
```

Out[10]:

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

In [11]:
```
mtcars.columns
```

Out[11]:
```
Index(['mpg', 'cyl', 'disp', 'hp', 'drat', 'wt', 'qsec', 'vs', 'am', 'gear',
       'carb'],
      dtype='object')
```

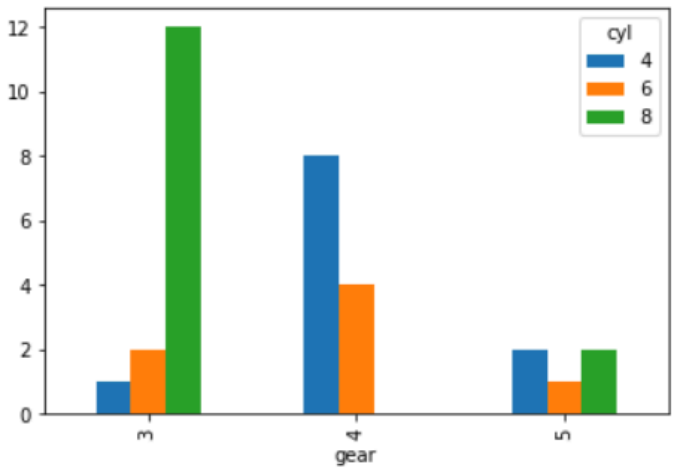
In [12]:
```
pd.crosstab(mtcars.gear,mtcars.cyl)
```

Out[12]:

cyl	4	6	8
gear			
3	1	2	12
4	8	4	0
5	2	1	2

In [13]:
```
pd.crosstab(mtcars.gear,mtcars.cyl).plot(kind ='bar')
```

Out[13]:
```
<Axes: xlabel='gear'>
```

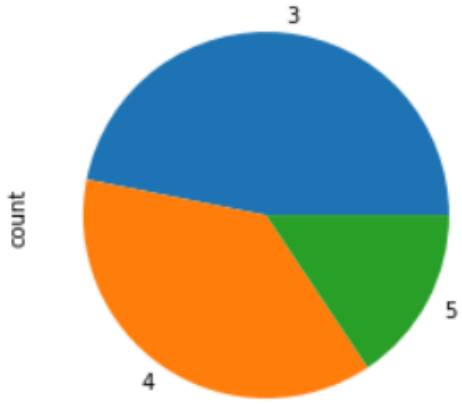


In [14]:
```
mtcars['gear'].value_counts()
mtcars.gear.value_counts().plot(kind ='pie')
```

Out[14]:
```
<Axes: ylabel='count'>
```

In [14]:
```
mtcars['gear'].value_counts()
mtcars.gear.value_counts().plot(kind ='pie')
```

Out[14]: <Axes: ylabel='count'>

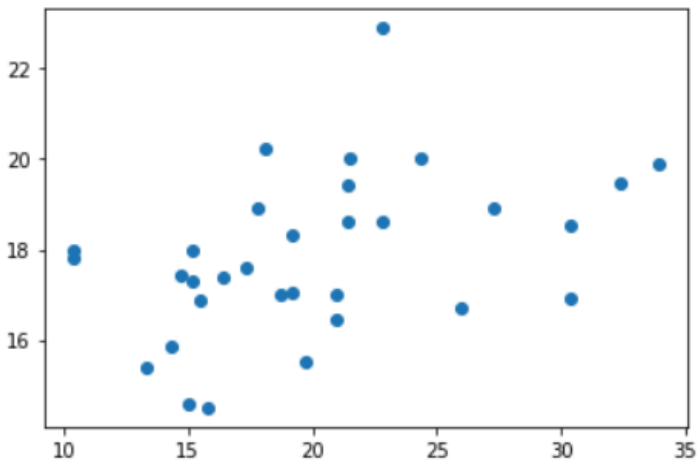


In [15]:
```
plt.scatter(mtcars.mpg,mtcars.qsec)
```

Out[15]: <matplotlib.collections.PathCollection at 0x1eca9176080>

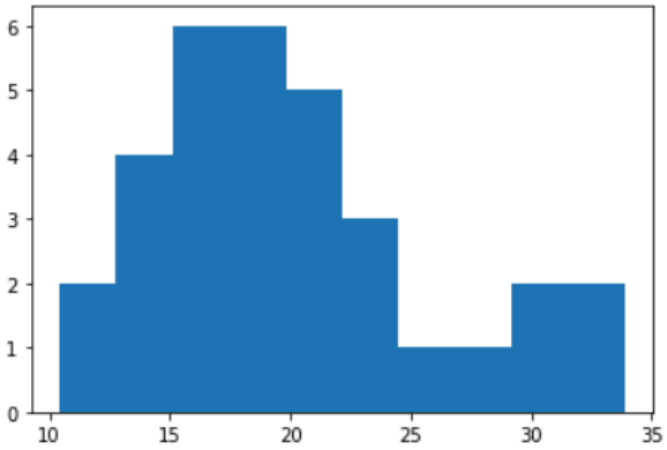
In [15]:
```
plt.scatter(mtcars.mpg,mtcars.qsec)
```

Out[15]: <matplotlib.collections.PathCollection at 0x1eca9176080>



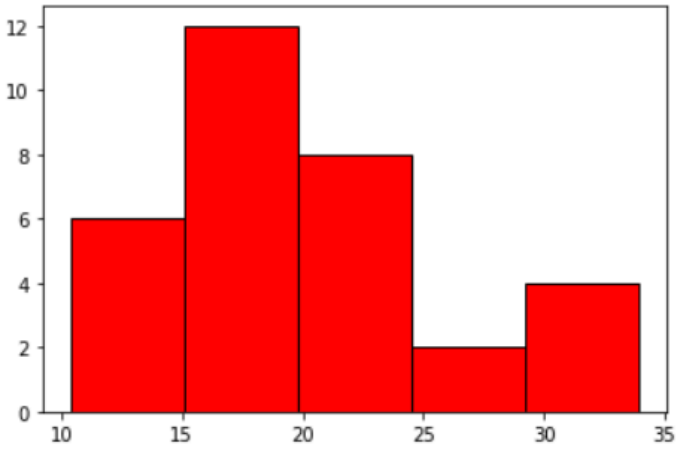
In [16]:
```
plt.hist(mtcars['mpg'])
```

Out[16]:
```
(array([2., 4., 6., 6., 5., 3., 1., 1., 2., 2.]),
 array([10.4 , 12.75, 15.1 , 17.45, 19.8 , 22.15, 24.5 , 26.85, 29.2 ,
        31.55, 33.9 ]),
 <BarContainer object of 10 artists>)
```



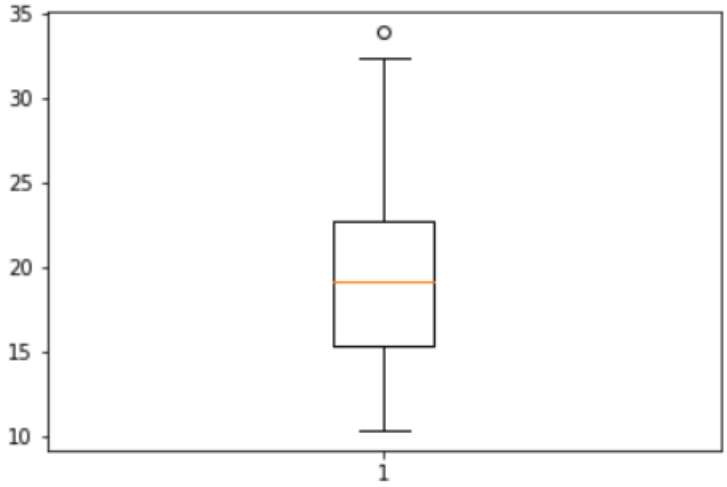
In [18]:
```
plt.hist(mtcars['mpg'],facecolor='red',edgecolor='black',bins=5)
```

Out[18]:
```
(array([ 6., 12.,  8.,  2.,  4.]),
 array([10.4, 15.1, 19.8, 24.5, 29.2, 33.9]),
 <BarContainer object of 5 artists>)
```



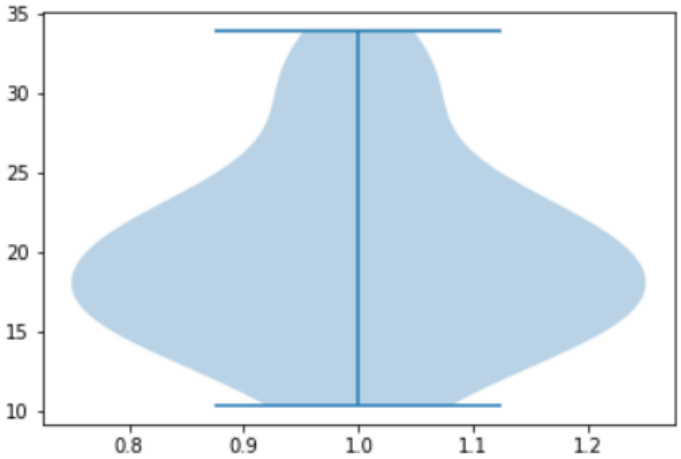
In [19]:
```
plt.boxplot(mtcars['mpg'],vert= True)
```

Out[19]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1eca9373c70>,
  <matplotlib.lines.Line2D at 0x1eca9373f10>],
 'caps': [<matplotlib.lines.Line2D at 0x1eca93a81f0>,
  <matplotlib.lines.Line2D at 0x1eca93a8490>],
 'boxes': [<matplotlib.lines.Line2D at 0x1eca93739d0>],
 'medians': [<matplotlib.lines.Line2D at 0x1eca93a8730>],
 'fliers': [<matplotlib.lines.Line2D at 0x1eca93a89d0>],
 'means': []}
```



In [20]:
```
plt.violinplot(mtcars['mpg'])
```

Out[20]:
```
{'bodies': [<matplotlib.collections.PolyCollection at 0x1eca90ce620>],
 'cmaxes': <matplotlib.collections.LineCollection at 0x1eca93f1ed0>,
 'cmins': <matplotlib.collections.LineCollection at 0x1eca93f2500>,
 'cbars': <matplotlib.collections.LineCollection at 0x1eca93f2860>}
```



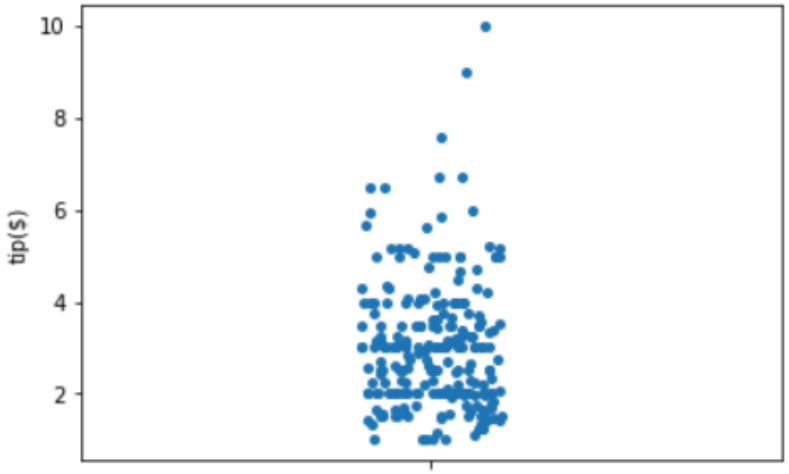
In [21]:
```
tips = sns.load_dataset('tips')
tips
```

Out[21]:

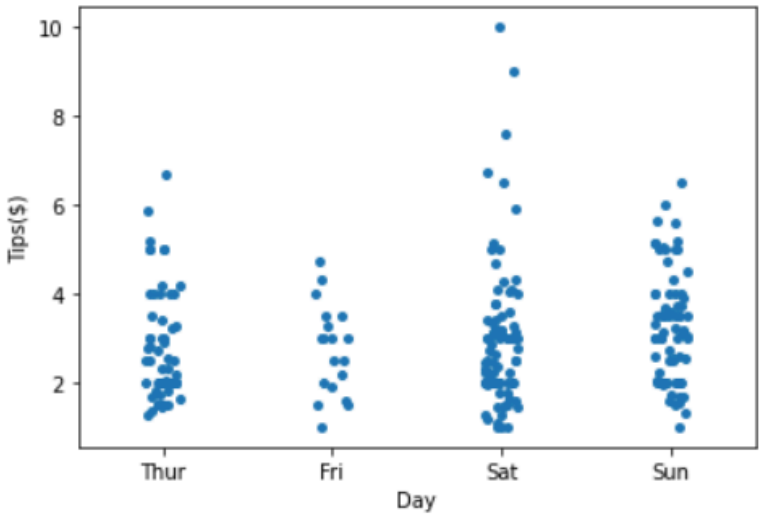
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

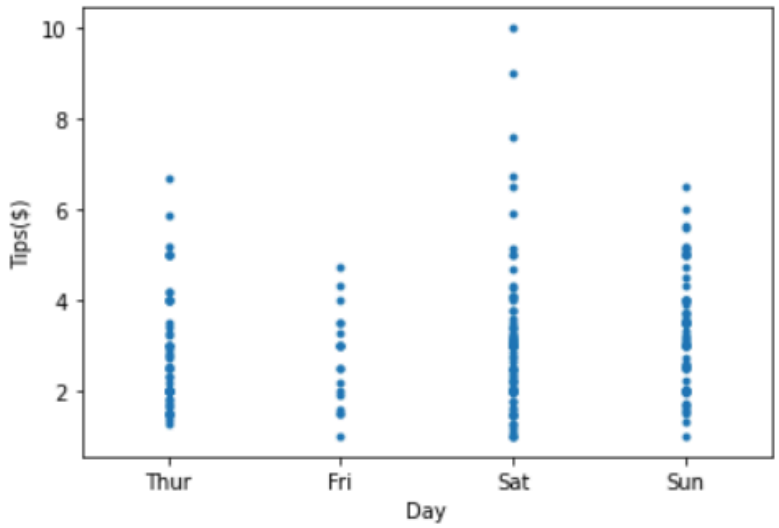
In [22]:
```
sns.stripplot(y='tip', data=tips, jitter = True)
plt.ylabel('tip($)')
plt.show()
```



In [24]:
```
sns.stripplot(x='day',y='tip',data=tips)
plt.ylabel('Tips($)')
plt.xlabel('Day')
plt.show()
```

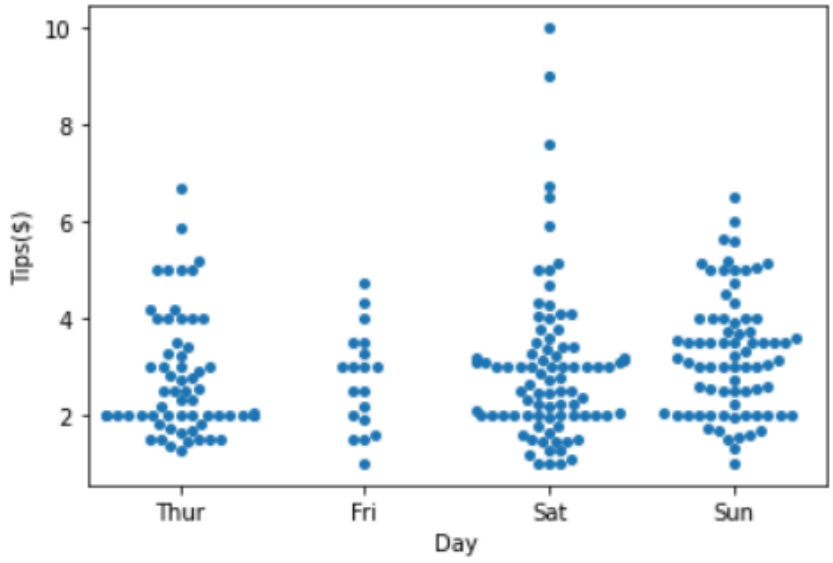


In [25]:
```
sns.stripplot(x='day',y='tip',data=tips, size=4, jitter = False)
plt.ylabel('Tips($)')
plt.xlabel('Day')
plt.show()
```



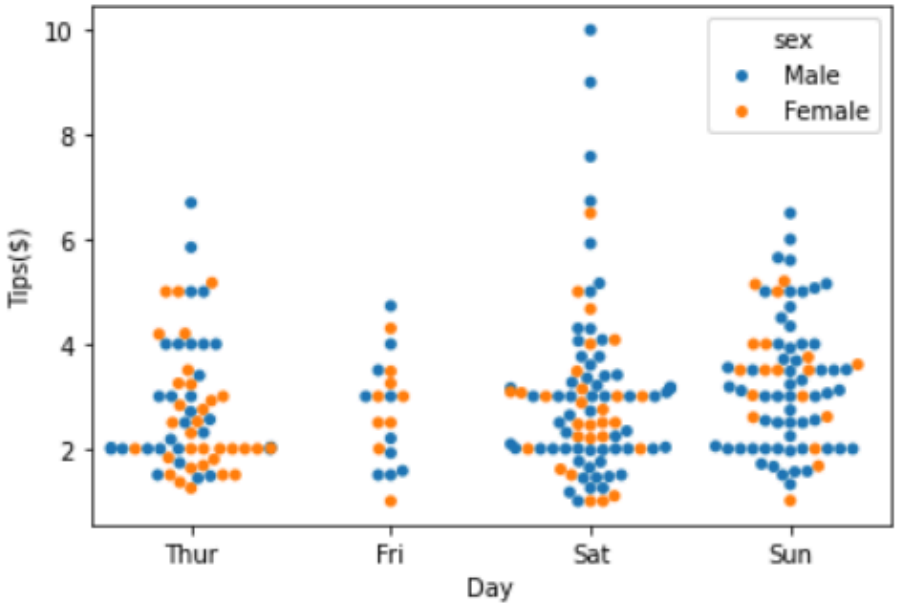
In [26]:
```
sns.swarmplot(x='day',y='tip',data=tips)
plt.ylabel('Tips($)')
plt.xlabel('Day')
plt.show()
```

```
C:\Users\anjal\anaconda3\envs\myenv\lib\site-packages\seaborn\c
aced; you may want to decrease the size of the markers or use s
  warnings.warn(msg, UserWarning)
C:\Users\anjal\anaconda3\envs\myenv\lib\site-packages\seaborn\c
aced; you may want to decrease the size of the markers or use s
  warnings.warn(msg, UserWarning)
```



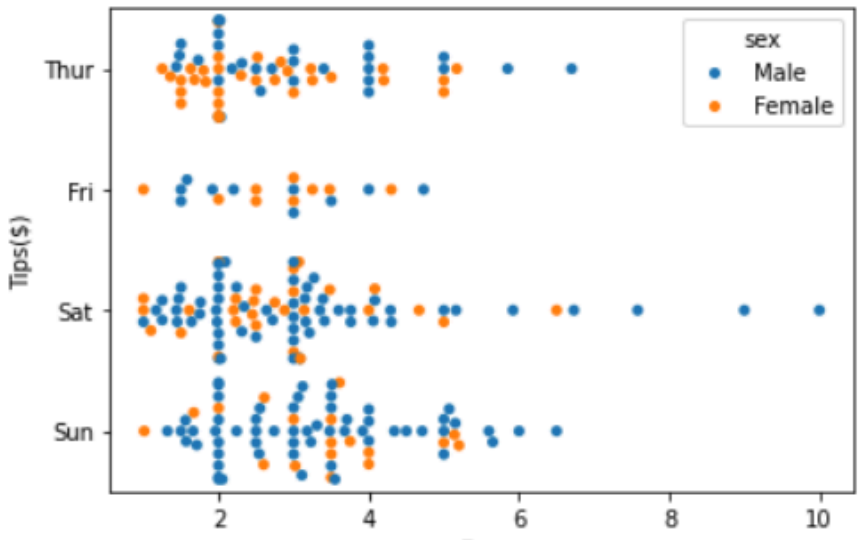
In [28]:
```
sns.swarmplot(x='day',y='tip',data=tips, hue='sex')
plt.ylabel('Tips($)')
plt.xlabel('Day')
plt.show()
```

```
C:\Users\anjal\anaconda3\envs\myenv\lib\site-packages\sea
aced; you may want to decrease the size of the markers or
  warnings.warn(msg, UserWarning)
C:\Users\anjal\anaconda3\envs\myenv\lib\site-packages\sea
aced; you may want to decrease the size of the markers or
  warnings.warn(msg, UserWarning)
```

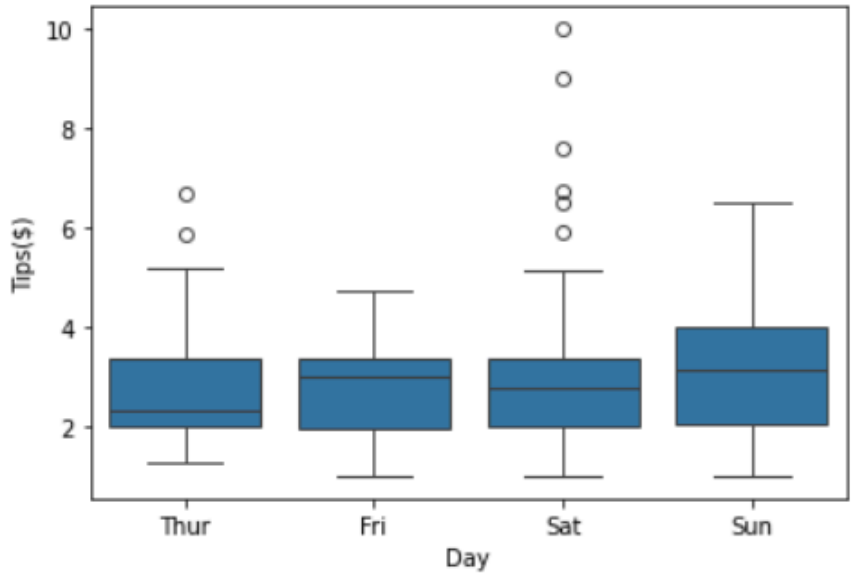


```
plt.ylabel('Tips($)')
plt.xlabel('Day')
plt.show()
```

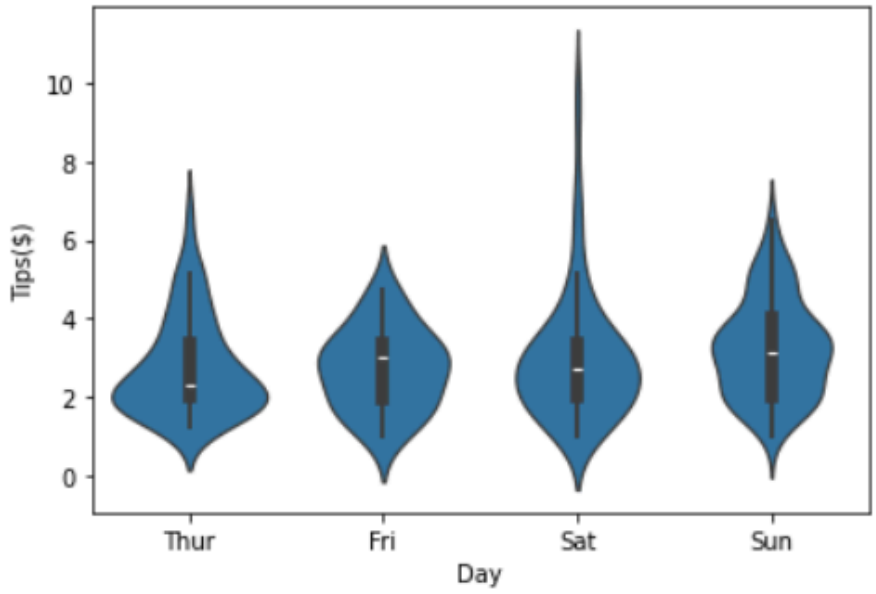
```
C:\Users\anjal\anaconda3\envs\myenv\lib\site-packages\seaborn\
aced; you may want to decrease the size of the markers or use
  warnings.warn(msg, UserWarning)
C:\Users\anjal\anaconda3\envs\myenv\lib\site-packages\seaborn\
aced; you may want to decrease the size of the markers or use
  warnings.warn(msg, UserWarning)
C:\Users\anjal\anaconda3\envs\myenv\lib\site-packages\seaborn\
laced; you may want to decrease the size of the markers or use
  warnings.warn(msg, UserWarning)
C:\Users\anjal\anaconda3\envs\myenv\lib\site-packages\seaborn\
laced; you may want to decrease the size of the markers or use
  warnings.warn(msg, UserWarning)
C:\Users\anjal\anaconda3\envs\myenv\lib\site-packages\seaborn\
aced; you may want to decrease the size of the markers or use
  warnings.warn(msg, UserWarning)
```



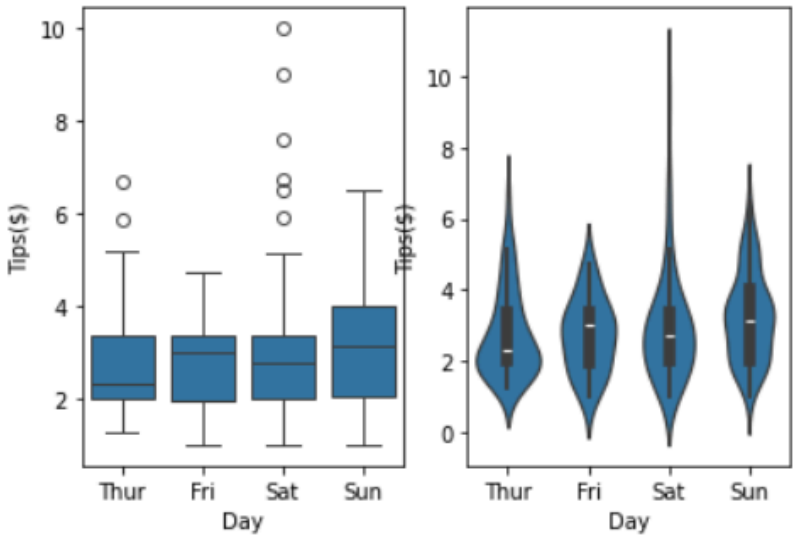
In [31]:
```
sns.boxplot(x='day',y='tip',data=tips)
plt.ylabel('Tips($)')
plt.xlabel('Day')
plt.show()
```



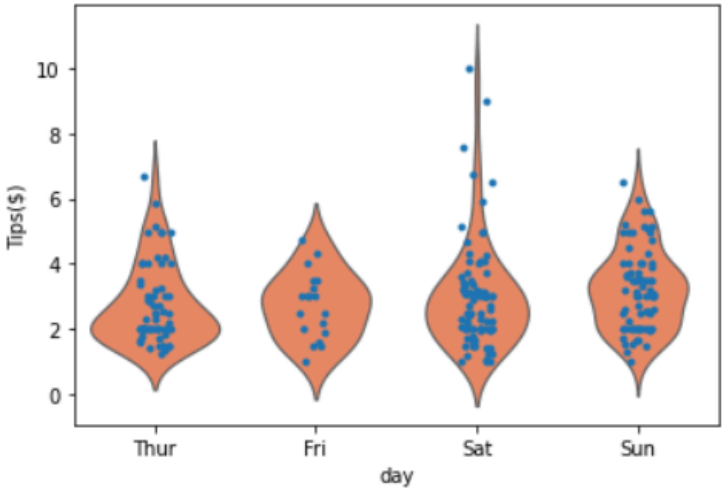
In [34]:
```
#plt.subplot(1,2,2)
sns.violinplot(x='day',y='tip',data=tips)
plt.ylabel('Tips($)')
plt.xlabel('Day')
plt.show()
```



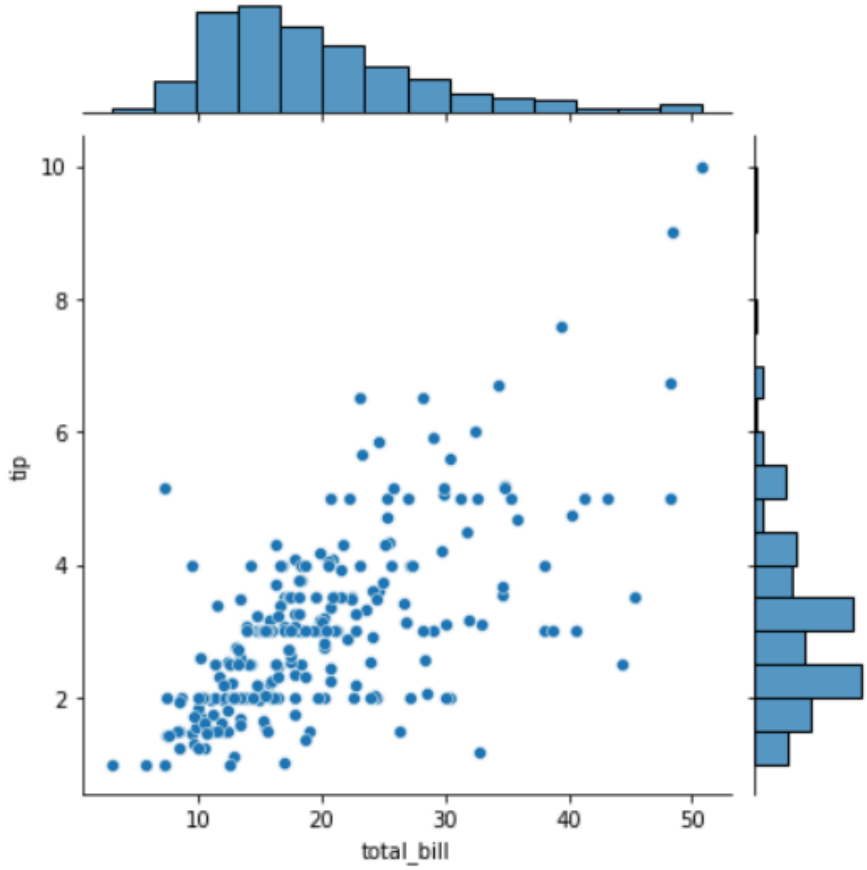
In [36]:
```
plt.subplot(1,2,1)
sns.boxplot(x='day',y='tip',data=tips)
plt.ylabel('Tips($)')
plt.xlabel('Day')

plt.subplot(1,2,2)
sns.violinplot(x='day',y='tip',data=tips)
plt.ylabel('Tips($)')
plt.xlabel('Day')
plt.show()
```



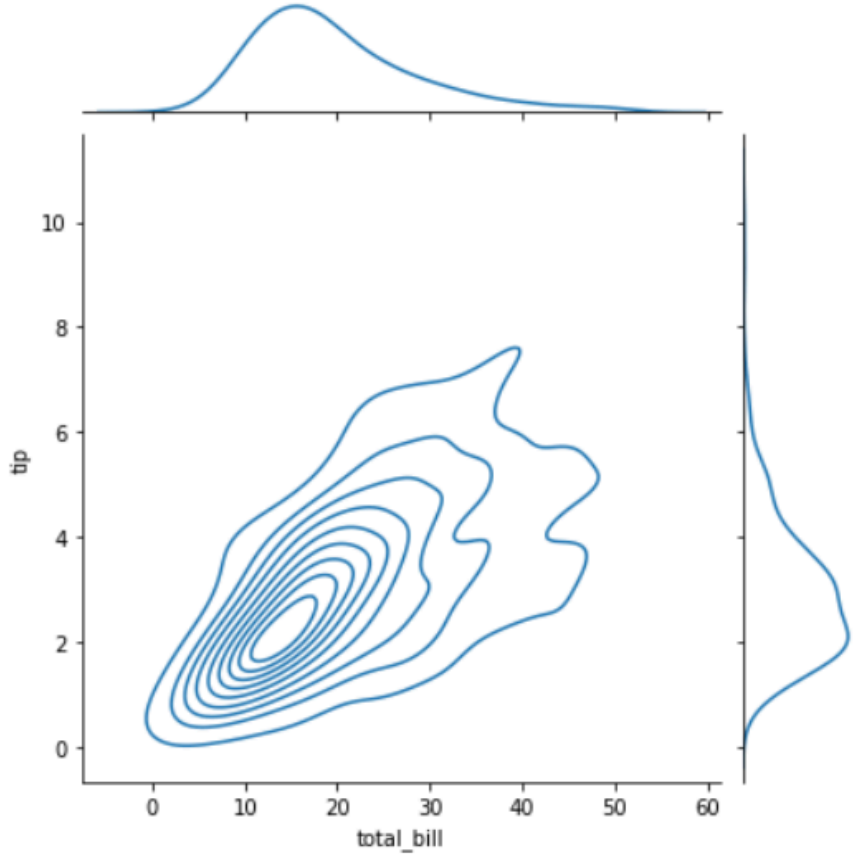
In [41]:
```
sns.violinplot(x='day',y='tip', data=tips, inner=None, color='coral')
sns.stripplot(x='day',y='tip', data=tips, size=4, jitter=True)
plt.ylabel('Tips($)')
plt.show()
```



In [43]:
```
sns.jointplot(x='total_bill', y='tip', data=tips)
plt.show()
```

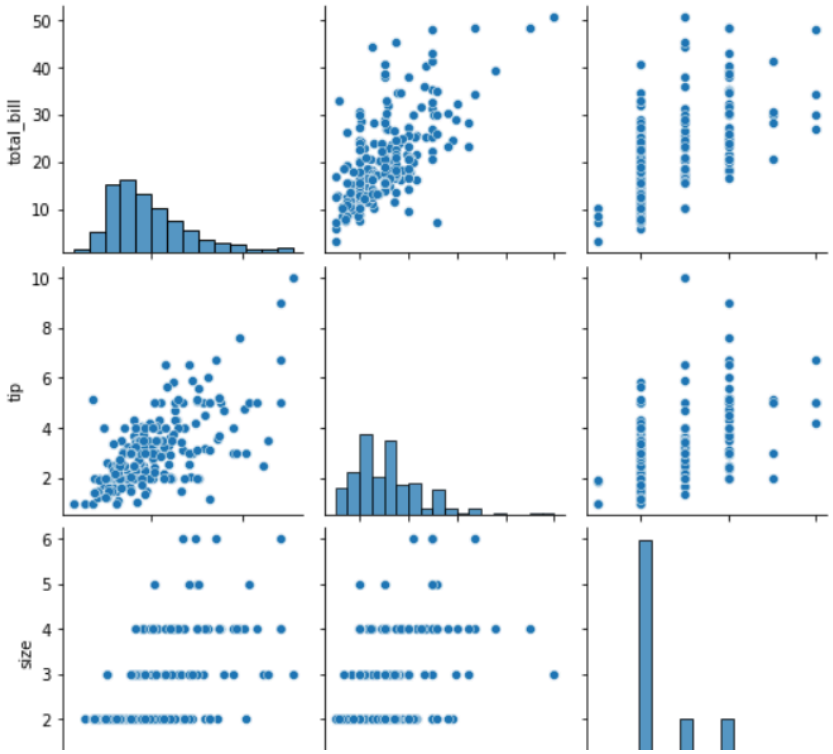


In [47]:
```
sns.jointplot(x='total_bill', y='tip', data=tips,kind='kde')
plt.show()
```



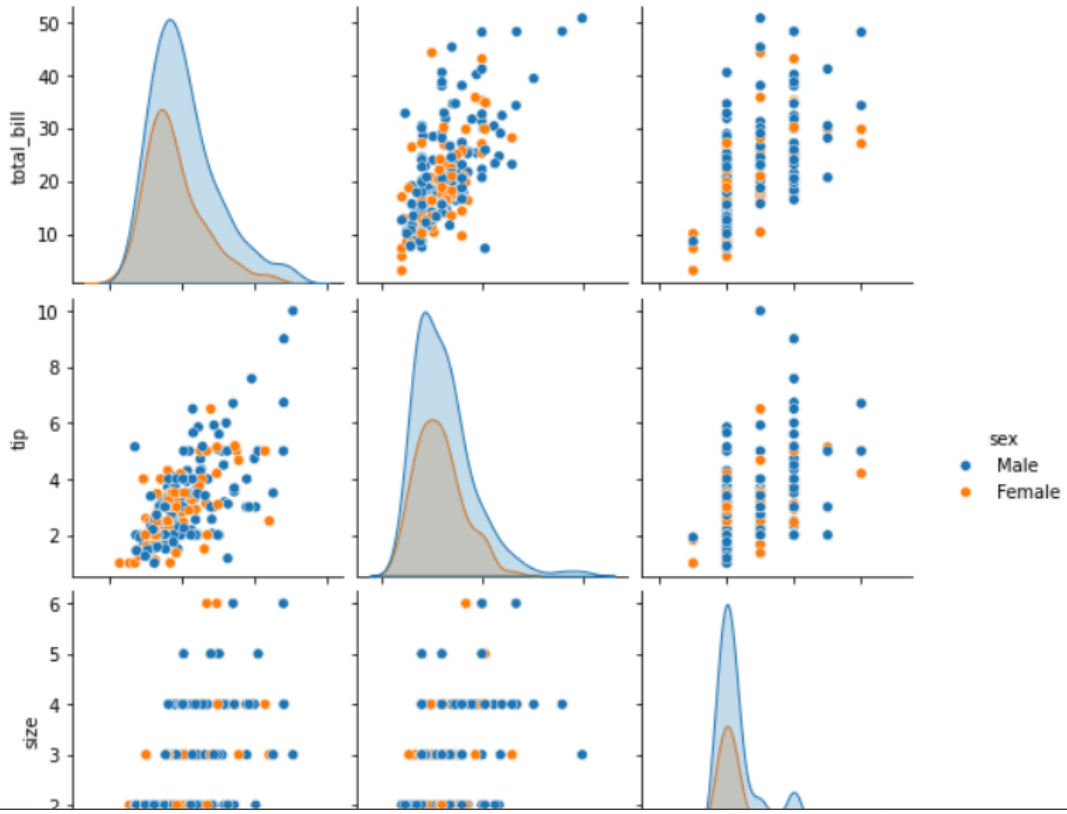
In [48]:
```
sns.pairplot(tips)
plt.show()
```

```
C:\Users\anjal\anaconda3\envs\myenv\lib\site-packages\seaborn\axisgrid.py:123: UserWarning: T
ght
  self._figure.tight_layout(*args, **kwargs)
```



In [49]:
```
sns.pairplot(tips, hue='sex')
plt.show()
```

```
C:\Users\anjal\anaconda3\envs\myenv\lib\site-packages\seaborn\axisgrid.py:123: Use
ght
  self._figure.tight_layout(*args, **kwargs)
```



In [50]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [51]:
```
train.head()
```

Out[51]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

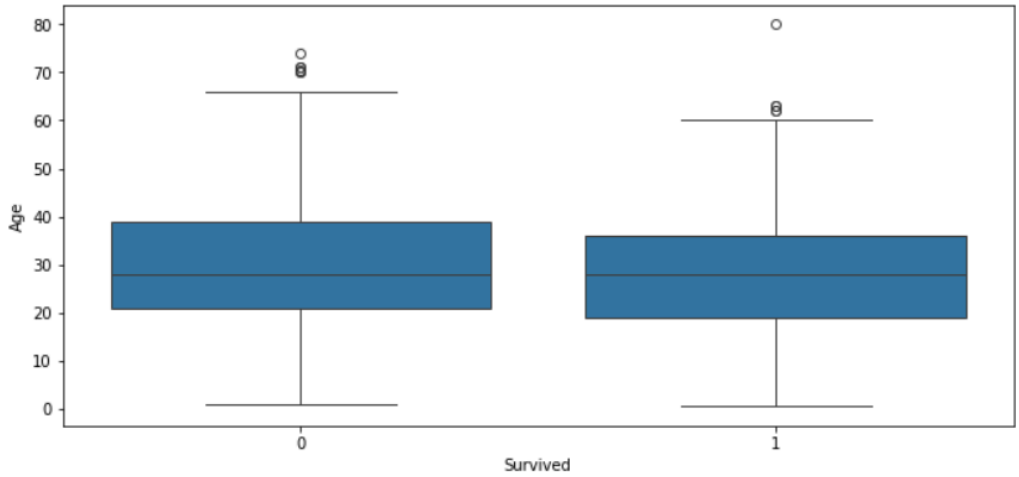
In [59]:
```
train.groupby('Survived')['PassengerId'].count()
```

Out[59]:
```
Survived
0    549
1    342
Name: PassengerId, dtype: int64
```

In [54]:
```
f, ax = plt.subplots(figsize=(11,5))
sns.boxplot(x='Survived',y='Age', data=train)
```

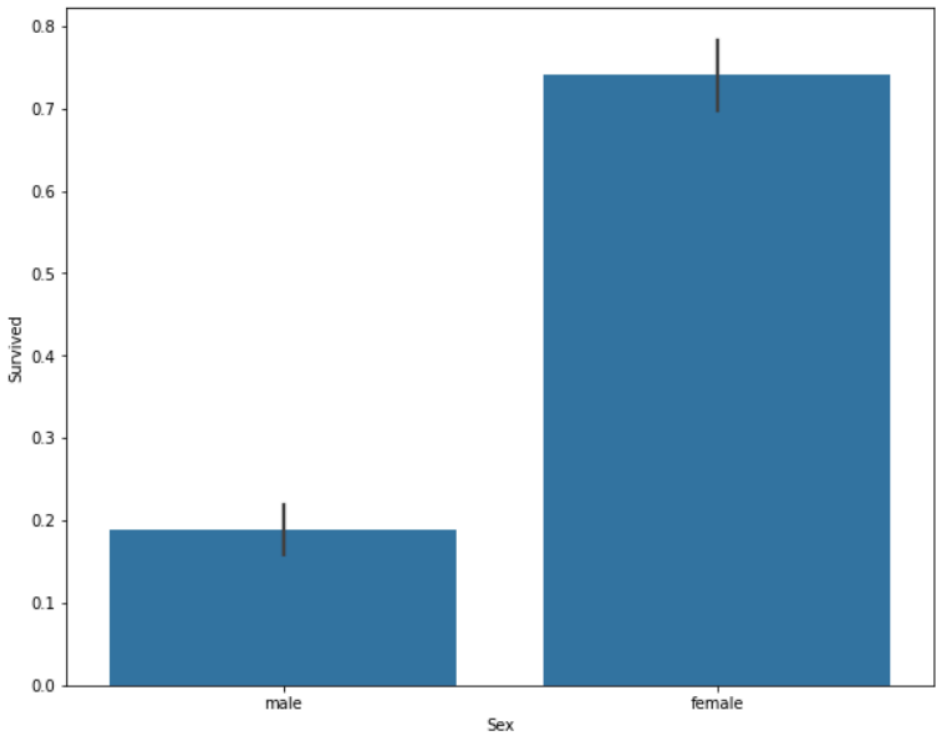
In [54]:
```
f, ax = plt.subplots(figsize=(11,5))
sns.boxplot(x='Survived',y='Age', data=train)
```

Out[54]: <Axes: xlabel='Survived', ylabel='Age'>



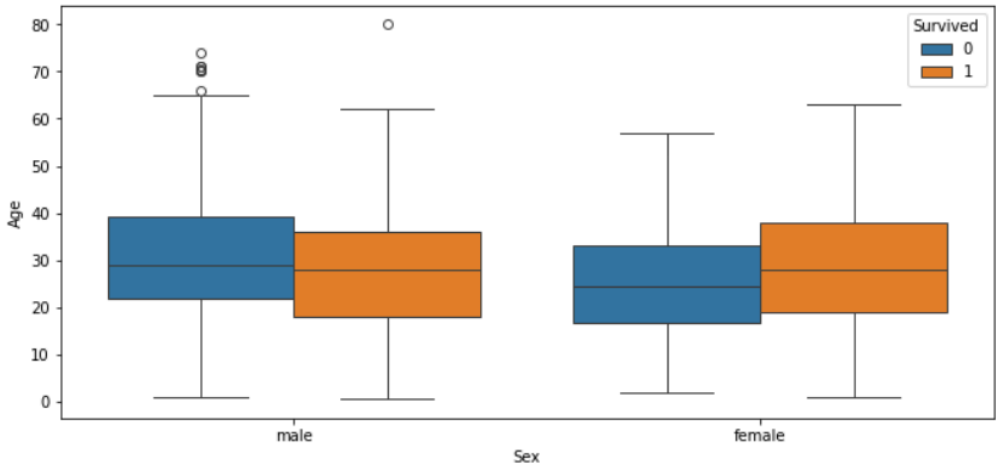
In [55]:
```
f, ax = plt.subplots(figsize=(10,8))
sns.barplot(x='Sex',y='Survived', data=train)
```

Out[55]: <Axes: xlabel='Sex', ylabel='Survived'>



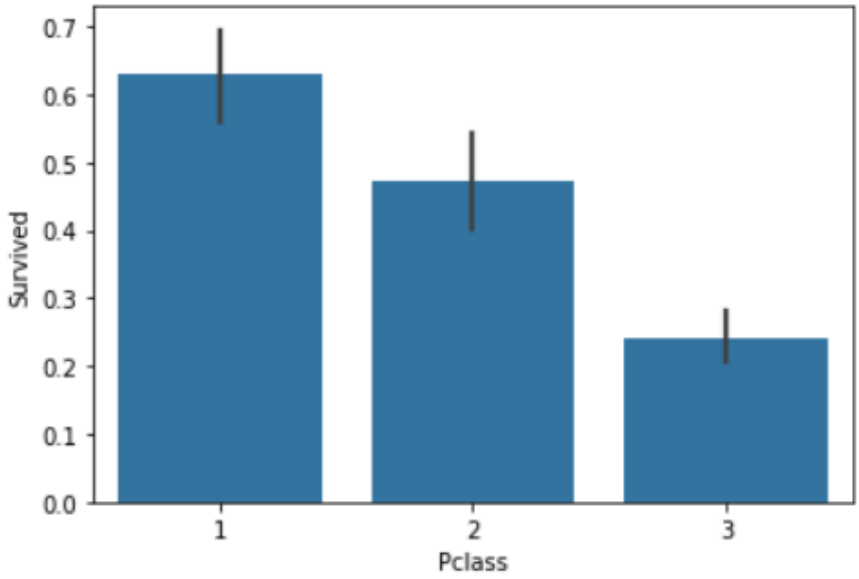
In [57]:
```
f, ax = plt.subplots(figsize=(11,5))
sns.boxplot(x='Sex',y='Age', hue='Survived', data=train)
```

Out[57]: <Axes: xlabel='Sex', ylabel='Age'>



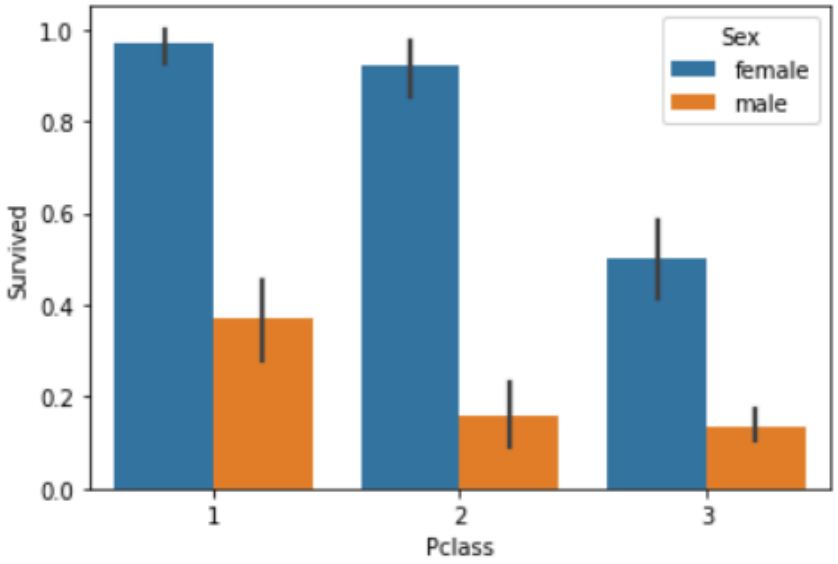
In [60]:
```
sns.barplot(x='Pclass',y='Survived',data=train)
```

Out[60]: <Axes: xlabel='Pclass', ylabel='Survived'>



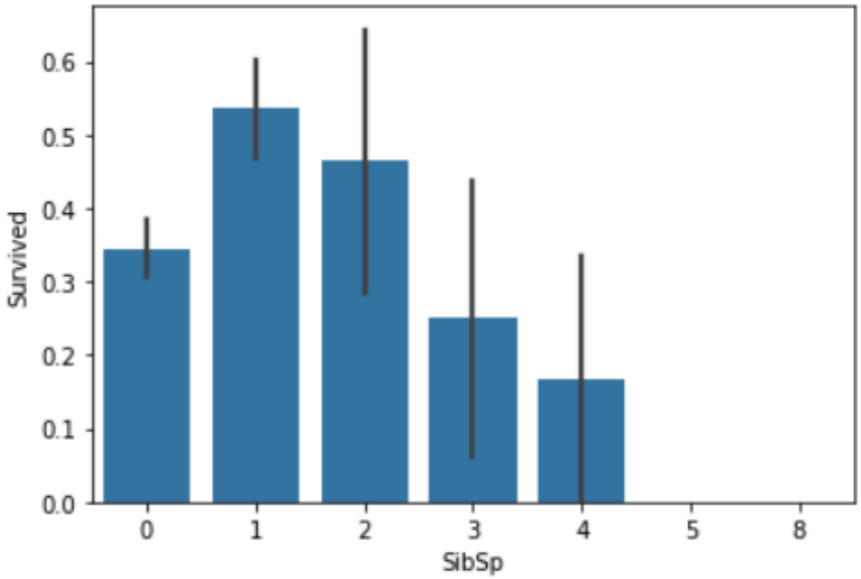
In [62]:
```
sns.barplot(x='Pclass',y='Survived', hue='Sex',data=train)
```

Out[62]: <Axes: xlabel='Pclass', ylabel='Survived'>



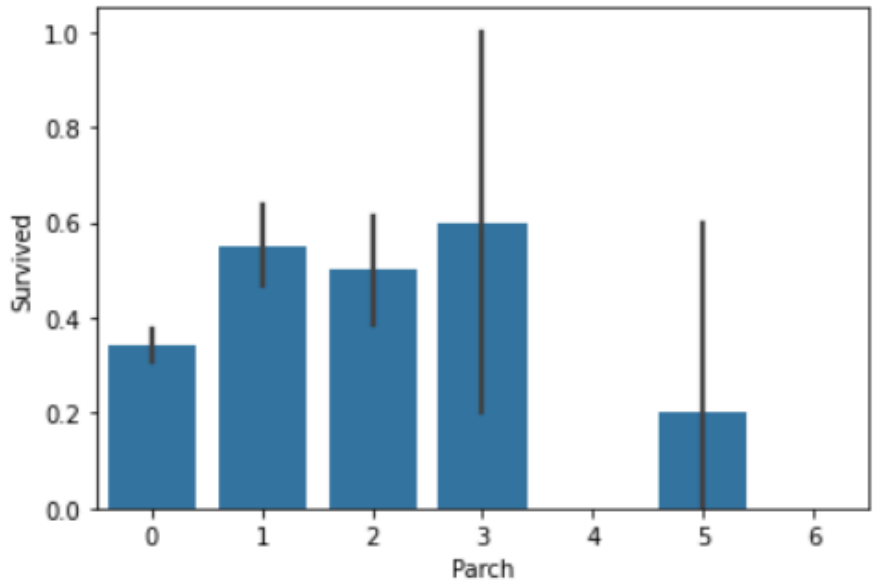
In [63]:
```
sns.barplot(x='SibSp',y='Survived',data=train)
```

Out[63]: <Axes: xlabel='SibSp', ylabel='Survived'>



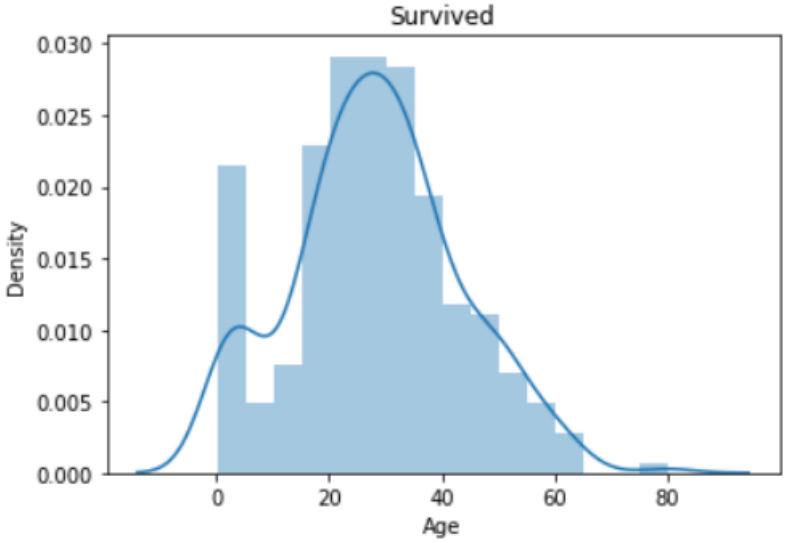
In [64]:
```
sns.barplot(x='Parch',y='Survived',data=train)
```

Out[64]:
```
<Axes: xlabel='Parch', ylabel='Survived'>
```



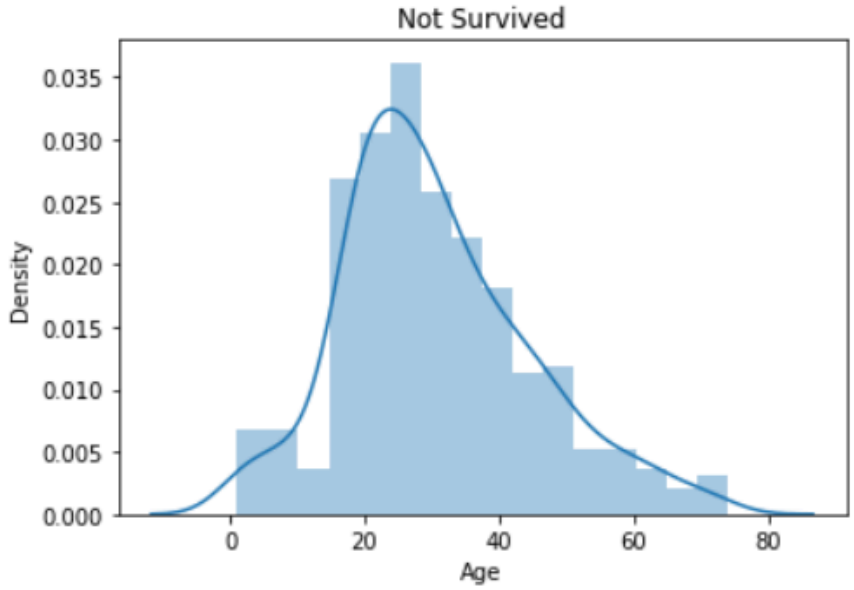
In [65]:
```
survived = train.loc[train['Survived']==1,'Age'].dropna()
sns.distplot(survived)
plt.title('Survived')
```

Out[65]:
```
Text(0.5, 1.0, 'Survived')
```



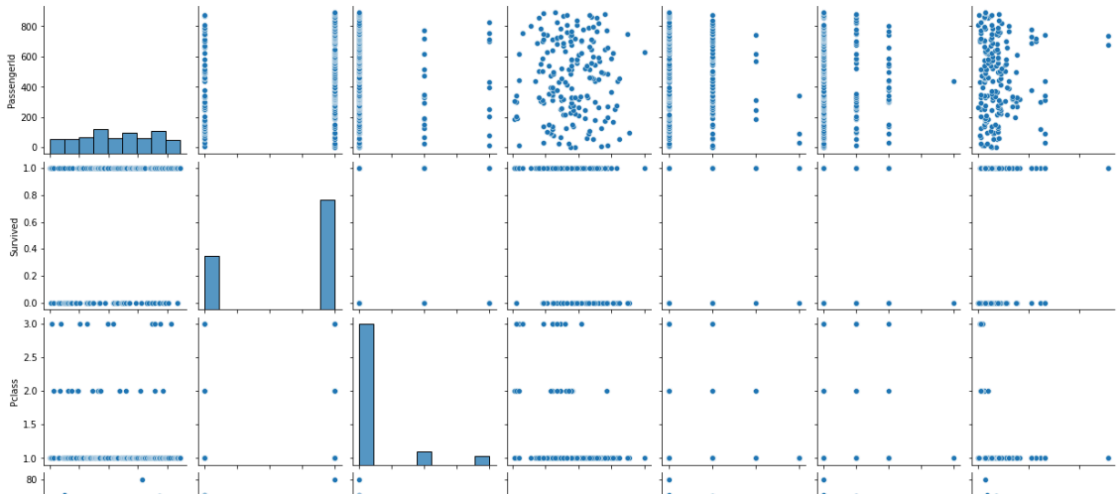
In [67]:
```
not_survived = train.loc[train['Survived']==0,'Age'].dropna()
sns.distplot(not_survived)
plt.title('Not Survived')
```

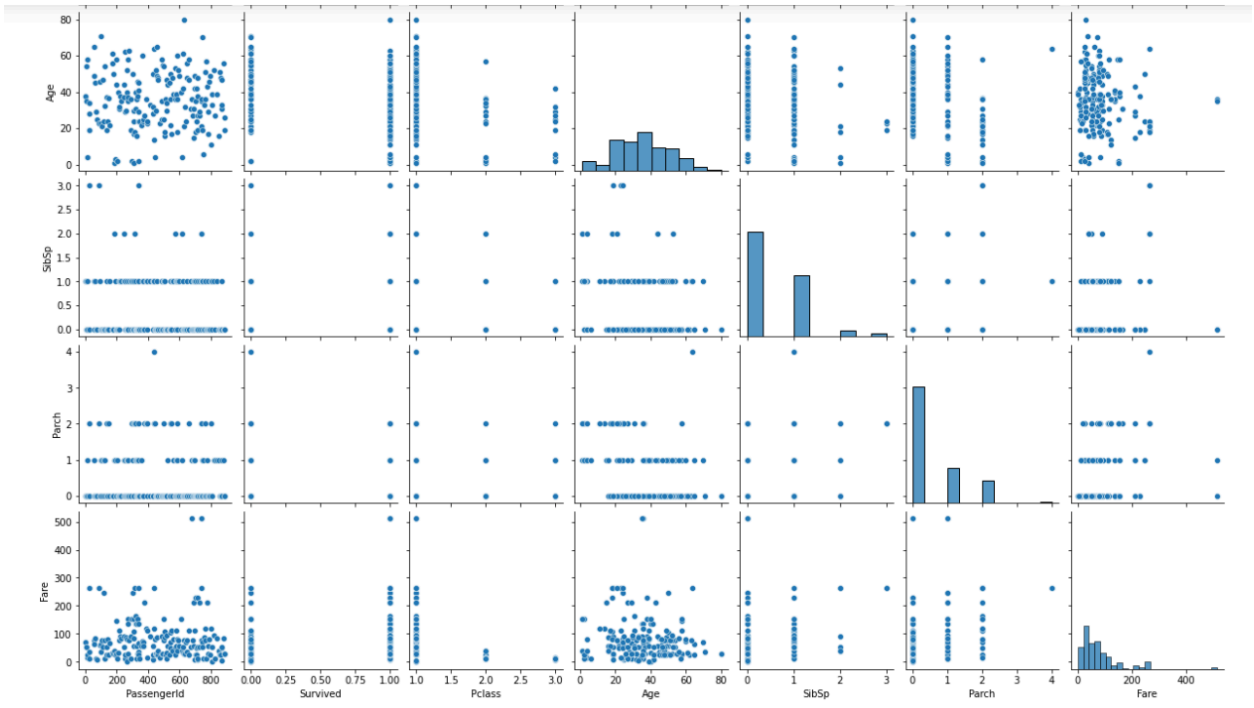
Out[67]:
```
Text(0.5, 1.0, 'Not Survived')
```



In [68]:
```
sns.pairplot(train.dropna())
```

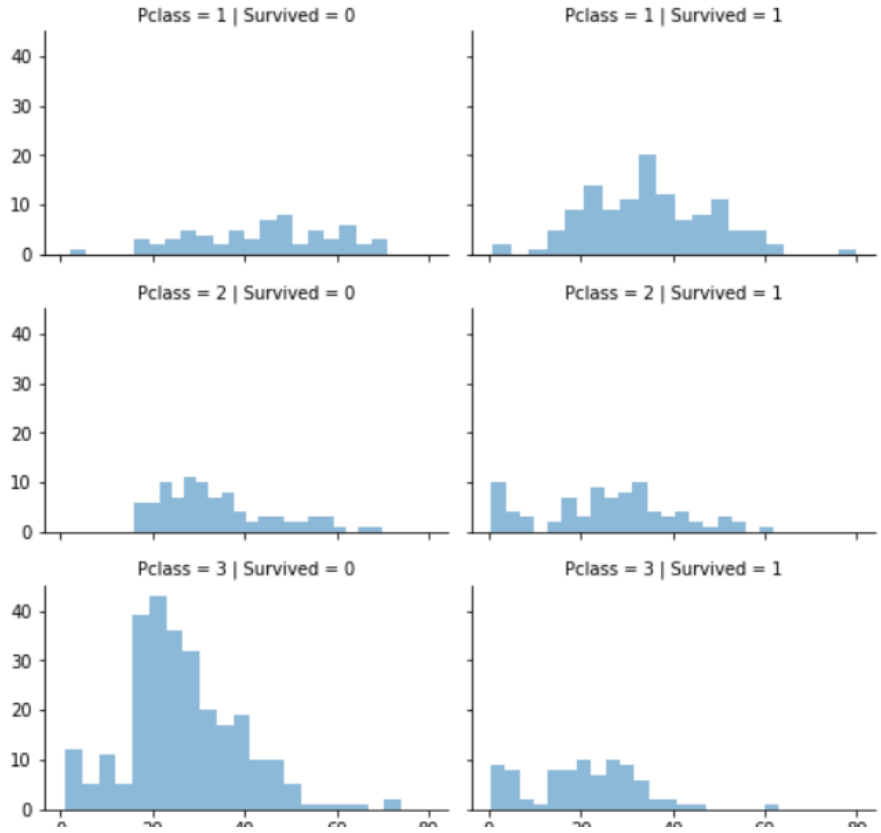
Out[68]:
```
<seaborn.axisgrid.PairGrid at 0x1ecb4712a10>
```



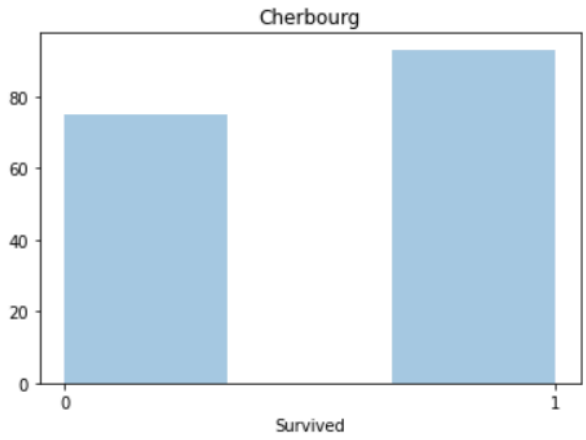

Age
SibSp
Parch
Fare
PassengerId
Survived
Pclass
Age
SibSp
Parch
Fare

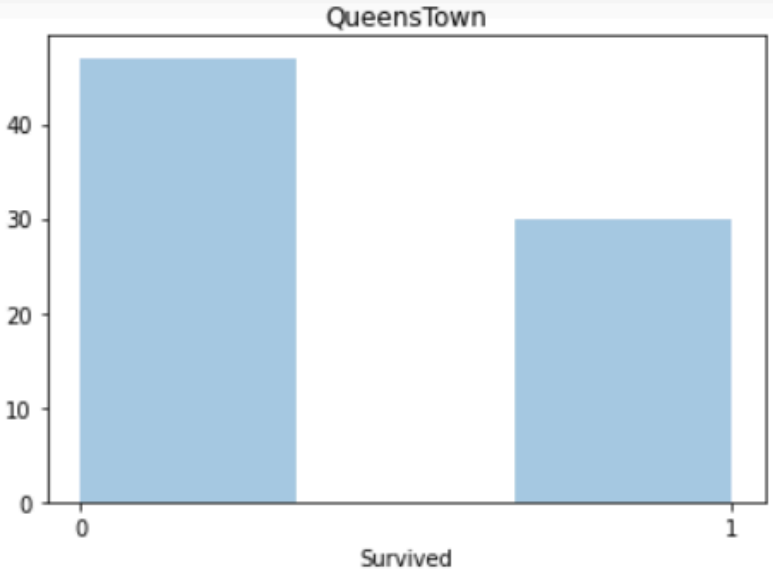
In [72]:
```
grid = sns.FacetGrid(train, col='Survived', row='Pclass', height=2.4, aspect=1.5)
grid.map(plt.hist, 'Age', alpha=0.5, bins=20)
grid.add_legend()
```

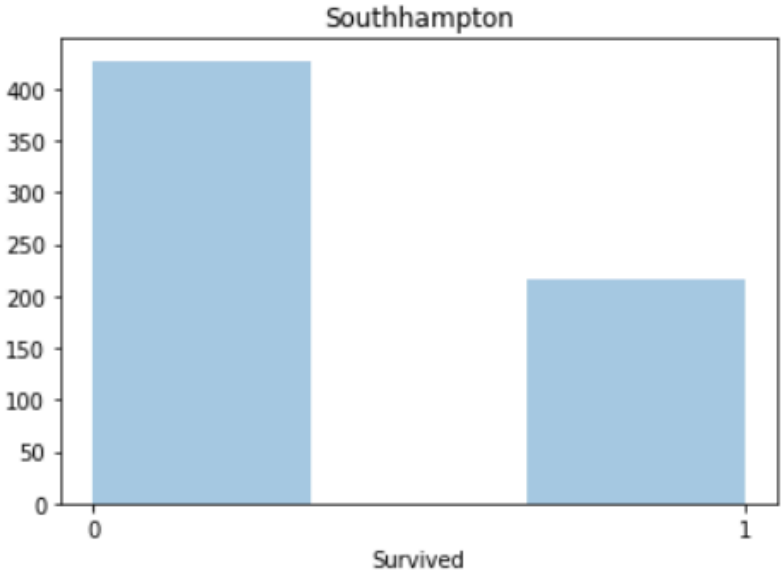
Out[72]: <seaborn.axisgrid.FacetGrid at 0x1ecb674e4d0>



In [71]:
```
sns.distplot(a=train[train['Embarked']=='C']['Survived'],bins=3,kde=False)
plt.title('Cherbourg')
plt.xticks([0,1])
plt.show()
plt.title('QueensTown')
sns.distplot(a=train[train['Embarked']=='Q']['Survived'],bins=3,kde=False)
plt.xticks([0,1])
plt.show()
plt.title('Southhampton')
sns.distplot(a=train[train['Embarked']=='S']['Survived'],bins=3,kde=False)
plt.xticks([0,1])
plt.show()
```

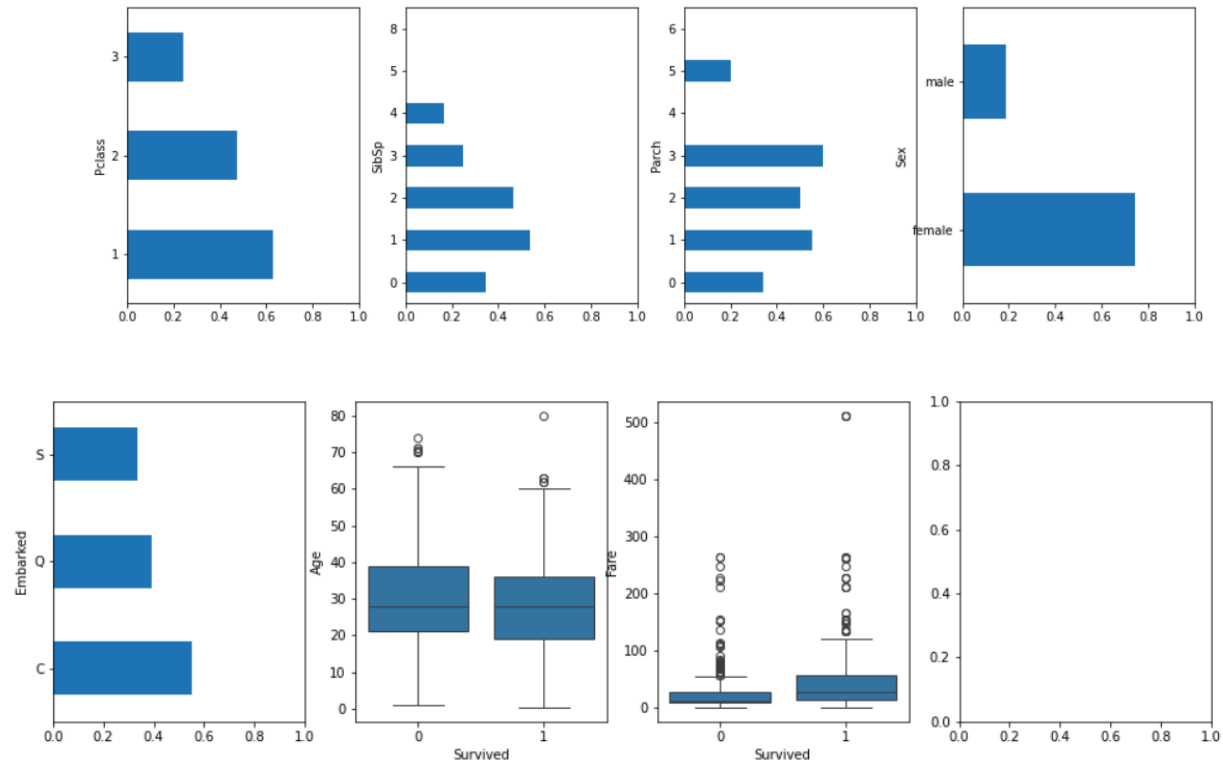



QueensTown
40
30
20
10
0
0
1
Survived


Southhampton
400
350
300
250
200
150
100
50
0
0
1
Survived

In [75]:
```
figbi, axesbi = plt.subplots(2,4, figsize=(16,10))
train.groupby('Pclass')['Survived'].mean().plot(kind='barh',ax=axesbi[0,0], xlim=[0,1])
train.groupby('SibSp')['Survived'].mean().plot(kind='barh',ax=axesbi[0,1], xlim=[0,1])
train.groupby('Parch')['Survived'].mean().plot(kind='barh',ax=axesbi[0,2], xlim=[0,1])
train.groupby('Sex')['Survived'].mean().plot(kind='barh',ax=axesbi[0,3], xlim=[0,1])
train.groupby('Embarked')['Survived'].mean().plot(kind='barh',ax=axesbi[1,0], xlim=[0,1])
sns.boxplot(x='Survived',y='Age',data=train,ax=axesbi[1,1])
sns.boxplot(x='Survived',y='Fare',data=train,ax=axesbi[1,2])
```

Out[75]: <Axes: xlabel='Survived', ylabel='Fare'>



CONCLUSION:

In conclusion, the ANOVA analysis on the Iris dataset revealed statistically significant differences in feature means among iris species, indicating distinct characteristics for each species. Additionally, data visualization techniques using Seaborn and Matplotlib facilitated the exploration of dataset distributions and relationships, uncovering valuable insights such as patterns, correlations, and outliers. Overall, both ANOVA and data visualization proved effective in analyzing and interpreting the datasets, providing valuable insights for further research or decision-making processes.