Within Matplotlib and Seaborn, we will be covering a few of the most commonly used plots in the data science world for easy visualization. Useful packages for visualizations in python **Matplotlib** Matplotlib is a visualization library in Python for 2D plots of arrays. Matplotlib is written in Python and makes use of the NumPy library. It can be used in Python and IPython shells, Jupyter notebook, and web application servers. Matplotlib comes with a wide variety of plots like line, bar, scatter, histogram, etc. which can help us, deep-dive, into understanding trends, patterns, correlations. It was introduced by John Hunter in 2002.

Seaborn Seaborn is a dataset-oriented library for making statistical representations in Python. It is developed atop matplotlib and to create different visualizations. It is integrated with pandas data structures. The library internally performs the required mapping and aggregation to create informative visuals It is recommended to use a Jupyter/IPython interface in matplotlib mode.

Bokeh Bokeh is an interactive visualization library for modern web browsers. It is suitable for large or streaming data assets and can be used to develop interactive plots and dashboards. There is a wide array of intuitive graphs in the library which can be leveraged to develop solutions. It works closely with PyData tools. The library is well-suited for creating customized visuals according to required use-cases. The visuals can also be made interactive to serve a what-if scenario model. All the codes are open source and available on GitHub.

plotly plotly.py is an interactive, open-source, high-level, declarative, and browser-based visualization library for Python. It holds an array of useful visualization which includes scientific charts, 3D graphs, statistical charts, financial charts among others. Plotly graphs can be viewed in Jupyter notebooks, standalone HTML files, or hosted online. Plotly library provides options for interaction and editing. The robust API works perfectly in both local and web browser mode.

ggplot **bold text** ggplot is a Python implementation of the grammar of graphics. The Grammar of Graphics refers to the mapping of data to aesthetic attributes (colour, shape, size) and geometric objects (points, lines, bars). The basic building blocks according to the grammar of graphics are data, geom (geometric objects), stats (statistical transformations), scale, coordinate system, and facet.

Using ggplot in Python allows you to develop informative visualizations incrementally, understanding the nuances of the data first, and then tuning the components to improve the visual representations. titatic data set sibsp Number of Siblings/Spouses Aboard

parch Number of Parents/Children Aboard

In [ ]:

```
import seaborn as sns
#Creating the dataset
import matplotlib.pyplot as plt

df = sns.load_dataset('titanic')
df.head(10)
```

Out[ ]:

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

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	d
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	N
5	0	3	male	NaN	0	0	8.4583	Q	Third	man	True	N
6	0	1	male	54.0	0	0	51.8625	S	First	man	True	
7	0	3	male	2.0	3	1	21.0750	S	Third	child	False	N
8	1	3	female	27.0	0	2	11.1333	S	Third	woman	False	N
9	1	2	female	14.0	1	0	30.0708	C	Second	child	False	N

Grouping on who column to find which group paid most fare

In [ ]:

```
df1=df.groupby('who')['fare'].sum().to_frame().reset_index()
df1.head(10)
```

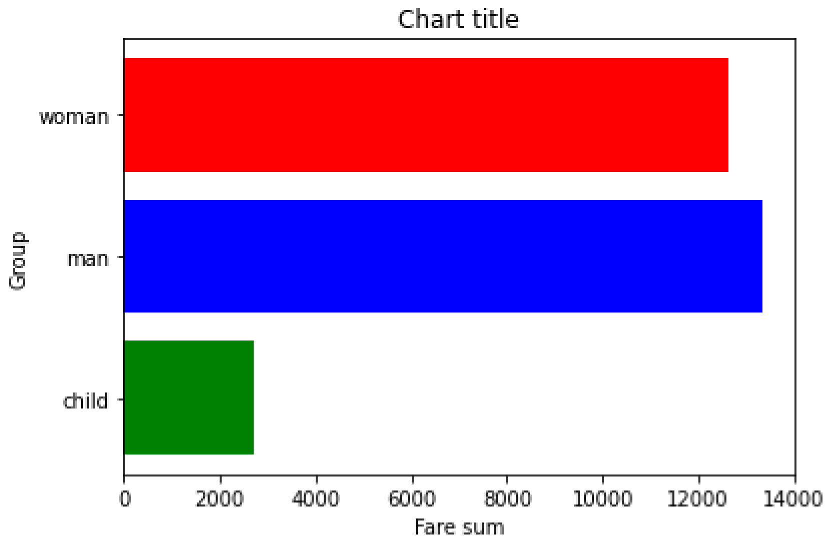
Out[ ]:

	who	fare
0	child	2721.2210
1	man	13352.0656
2	woman	12620.6627

displaying the result in barh using matlpot

In [ ]:

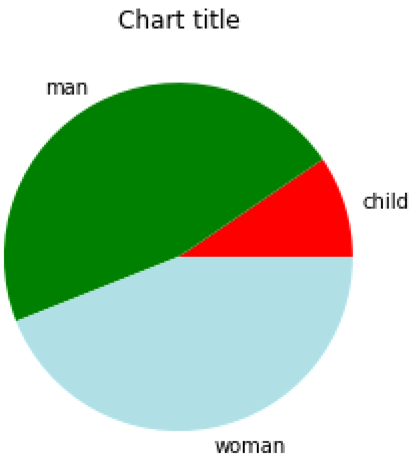
```
#Creating the bar chart
plt.barh(df1['who'],df1['fare'],color = ['Green','Blue','Red'])

#Adding the aesthetics
plt.title('Chart title')
plt.xlabel('Fare sum')
plt.ylabel('Group')

#Show the plot
plt.show()
```



Pie chart Pie charts can be used to identify proportions of the different components in a given whole.

In [ ]:

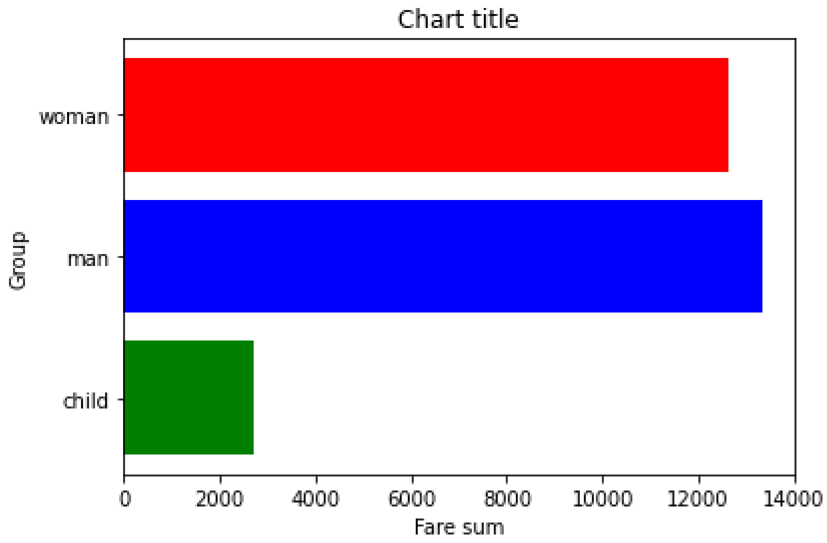
```
plt.pie(df1['fare'], labels = df1['who'],colors = ['Red','green','#B0E0E6'])
#Adding the aesthetics
plt.title('Chart title')
#Show the plot
plt.show()
```



Bar chart using Seaborn

In [ ]:

```
sns.barplot(x = 'fare',y = 'who',data = df1,palette = "Blues")
#Adding the aesthetics

plt.title('Chart title')
plt.xlabel('X axis title')
plt.ylabel('Y axis title')
# Show the plot
plt.show()
```



See how many of child, man and woman from each town

In [ ]:

```
df1=df.groupby(['embark_town','who'])['fare'].sum().to_frame().reset_index()
df1.head(10)
```

Out[ ]:

	embark_town	who	fare
0	Cherbourg	child	311.2127
1	Cherbourg	man	4502.1211
2	Cherbourg	woman	5258.9624
3	Queenstown	child	124.5292
4	Queenstown	man	450.8958
5	Queenstown	woman	446.8293
6	Southampton	child	2285.4791
7	Southampton	man	8399.0487
8	Southampton	woman	6754.8710

In [ ]:

```
#Creating the bar chart
plt.barh(df1['embark_town','who'],df1['fare'],color = ['Green','Blue','Red'])

#Adding the aesthetics
plt.title('Chart title')
plt.xlabel('Fare sum')
plt.ylabel('Group')

#Show the plot
plt.show()
```

In [ ]:

Grouped bar chart

A grouped bar chart is used when we want to compare the values in certain groups and sub-groups

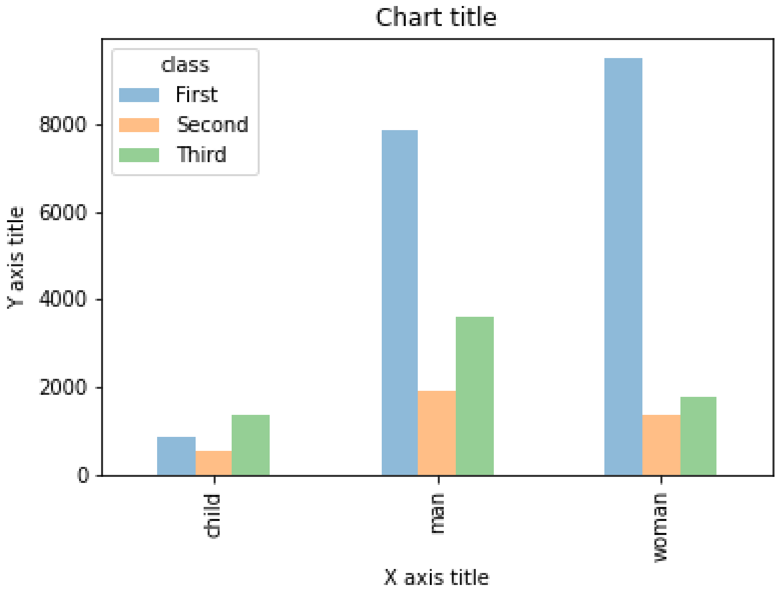
Stacked bar chart

A stacked bar chart is used when we want to compare the total sizes across the available groups and the composition of the different sub-groups

Show woman child and man for each town

In [ ]:

```
#Creating the dataset
import pandas as pd
import numpy as np
df = sns.load_dataset('titanic')
df_pivot = pd.pivot_table(df, values="fare",index="who",columns="class", aggfunc=np.
#Creating a grouped bar chart
ax = df_pivot.plot(kind="bar",alpha=0.5)
#Adding the aesthetics
plt.title('Chart title')
plt.xlabel('X axis title')
plt.ylabel('Y axis title')
# Show the plot
plt.show()
```



how many male and female

In [ ]:

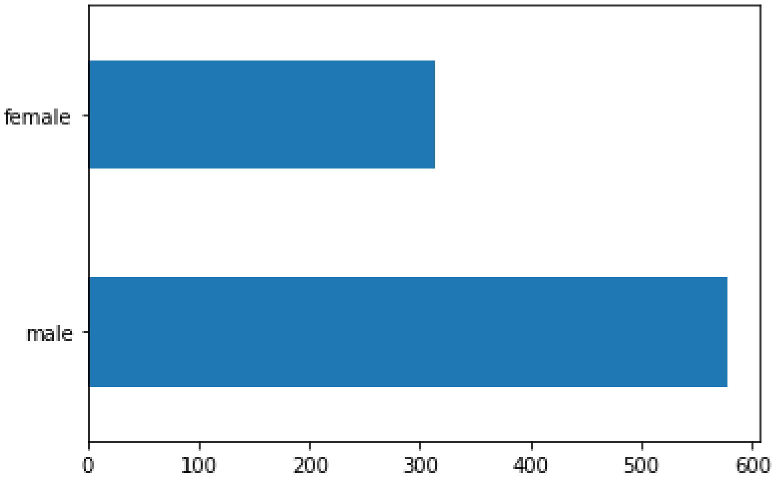
```
df = sns.load_dataset('titanic')
#Creating the bar chart
print(df['sex'].value_counts(ascending=True))
df.sex.value_counts().plot(kind = 'barh')
```

```
female     314
```

```
male      577
Name: sex, dtype: int64
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3dae7bfed0>
```



how many passengers in each town

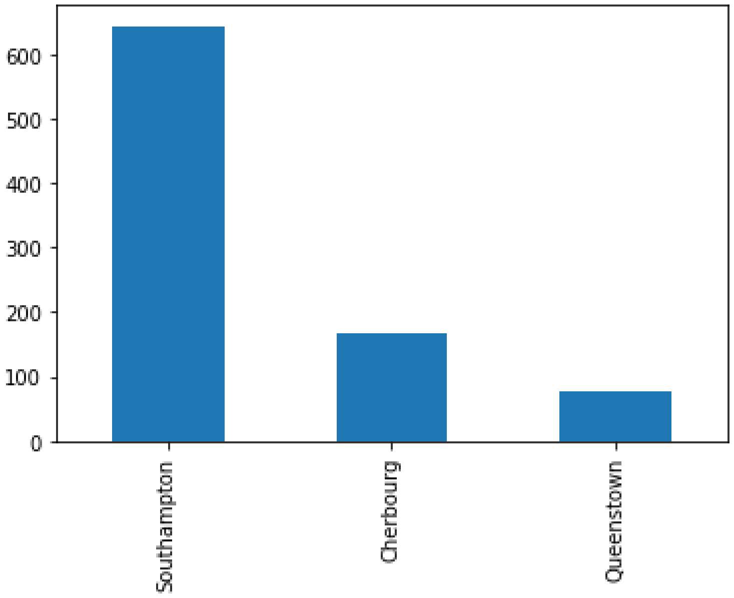
In [ ]:

```
df = sns.load_dataset('titanic')
#Creating the bar chart
print(df['embark_town'].value_counts(ascending=True))
df.embark_town.value_counts().plot(kind = 'bar')
```

```
Queenstown      77
Cherbourg      168
Southampton    644
Name: embark_town, dtype: int64
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3db072f950>
```



Looking at gender survival rates using seaborn. A quick look at gender survival rates shows that less males survived. It should be noted that there are more males in the data set, but you can see by comparing the heights of the bars that only about 1/6 of the males survived, while 2/3 of the females survived.

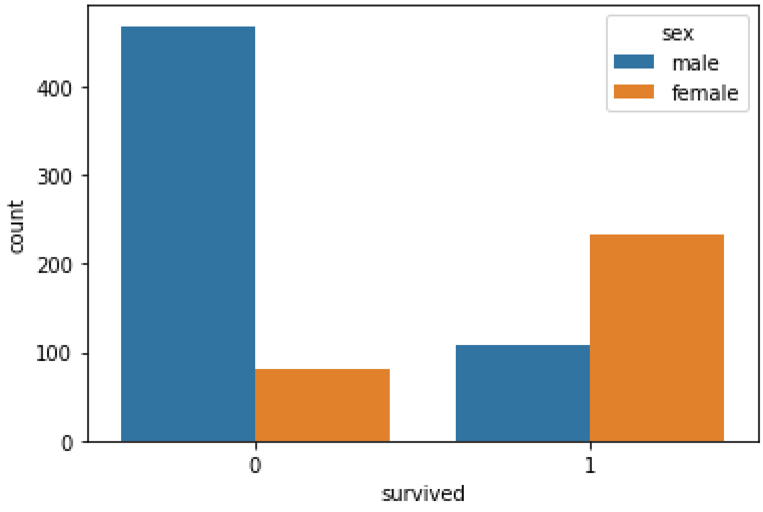
In [ ]:

```
import seaborn as sns
df = sns.load_dataset('titanic')
```

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

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3dad59b3d0>
```



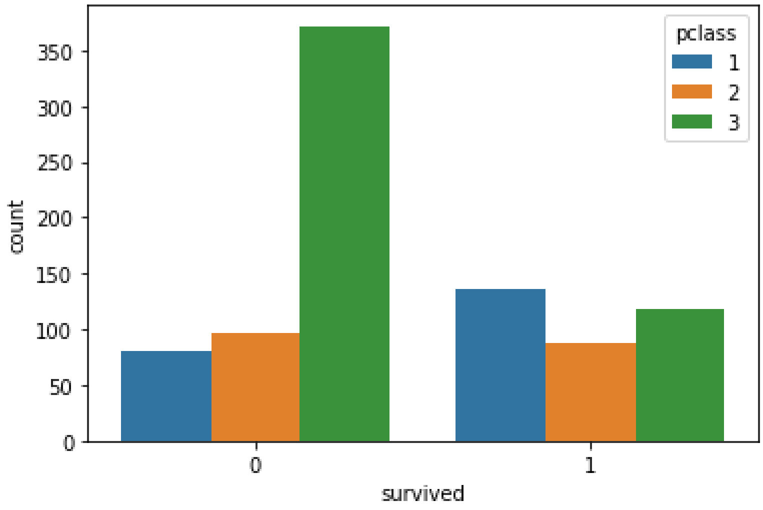
how many passengers survived in each class 1,2,3

In [ ]:

```
import seaborn as sns
df = sns.load_dataset('titanic')
sns.countplot(x='survived',data=df, hue='pclass')
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3dad722a50>
```



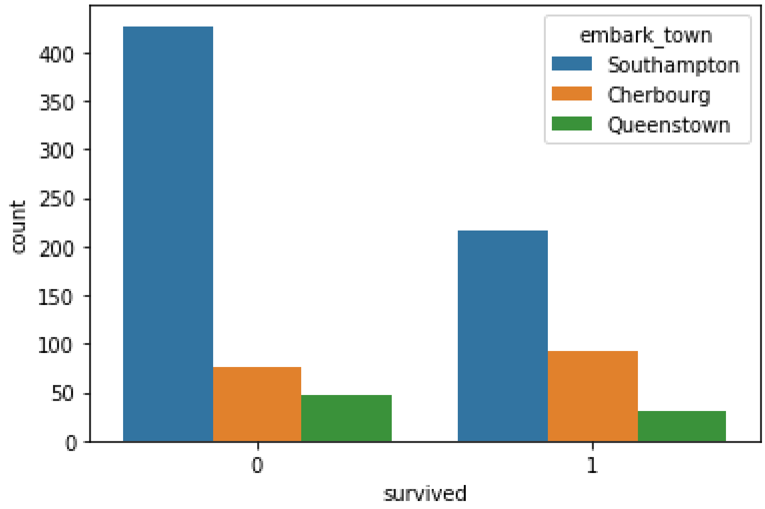
how many passengers survived in each town

In [ ]:

```
import seaborn as sns
df = sns.load_dataset('titanic')
sns.countplot(x='survived',data=df, hue='embark_town')
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3dae6c3b10>
```

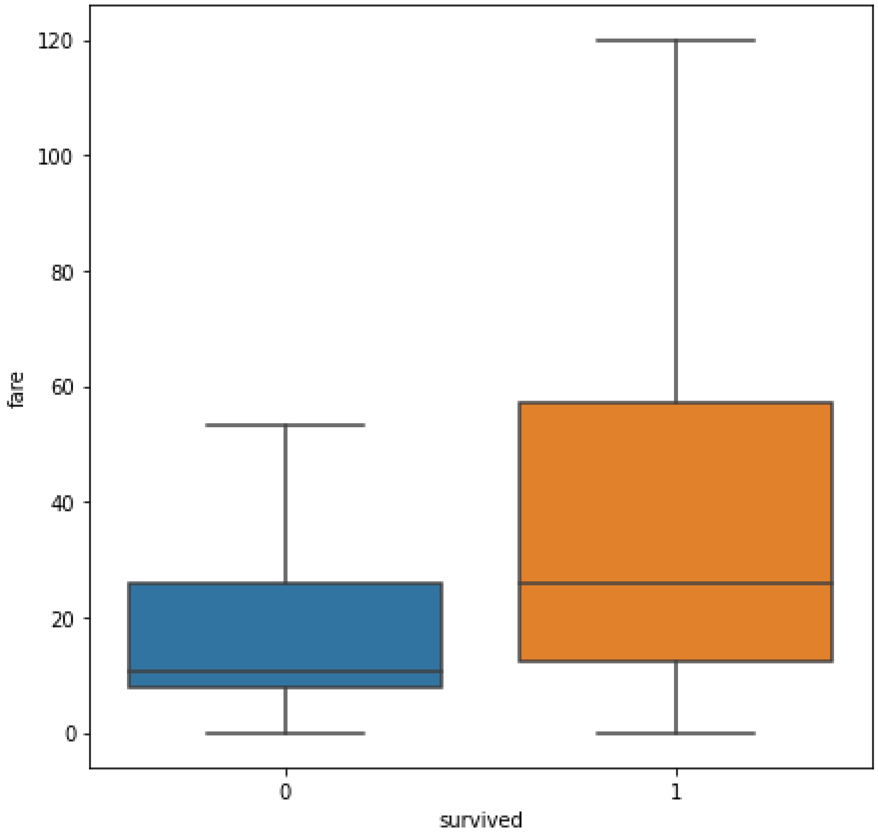


In [ ]:

```
plt.figure(figsize=(7,7))
sns.boxplot(x='survived',y='fare',data=df, showfliers = False)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3dac580810>
```



But what about the children? do the people having more children died?

Pandas Index.value_counts() function returns object containing counts of unique values. The resulting object will be in descending order so that the first element is the most frequently-occurring element. Excludes NA values by default.

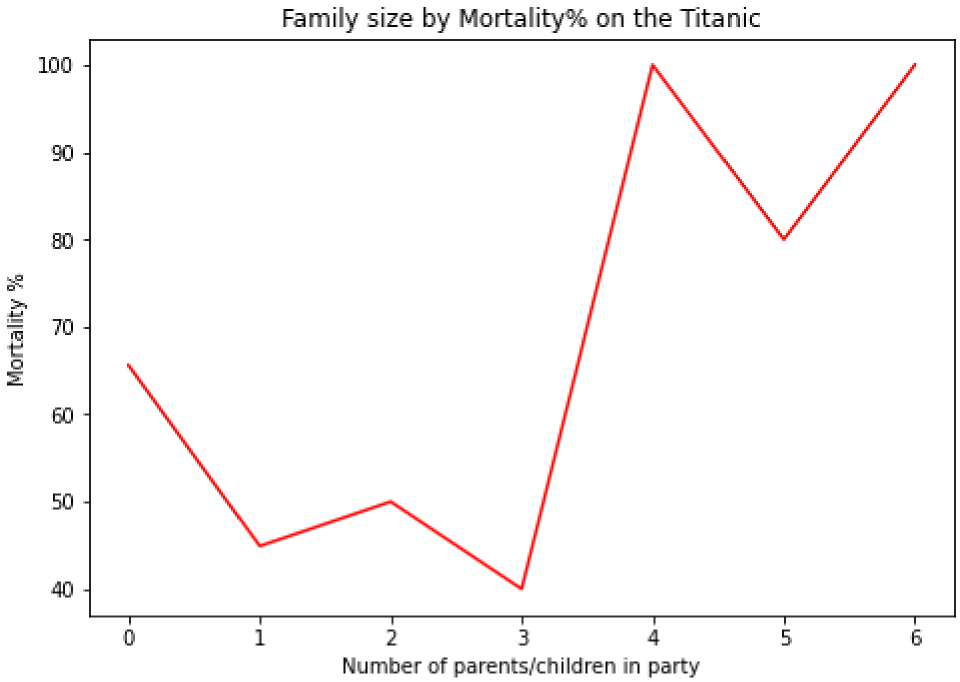
In [156…

```
titanic = sns.load_dataset('titanic')
print(titanic.isnull().sum())
titanic.head()
died = titanic[titanic['survived']==False]['parch'].value_counts().sort_index()
totals = titanic['parch'].value_counts().sort_index()
print(died)
dead_pct = (100*died/totals).fillna(0)
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.plot(dead_pct, color='red')
ax.set_xlabel('Number of parents/children in party')
ax.set_ylabel('Mortality %')
ax.set_title('Family size by Mortality% on the Titanic')
```

```
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
0    445
1     53
2     40
3      2
4      4
5      4
6      1
Name: parch, dtype: int64
```

Out[156…

```
Text(0.5, 1.0, 'Family size by Mortality% on the Titanic')
```



which passenger class survived more

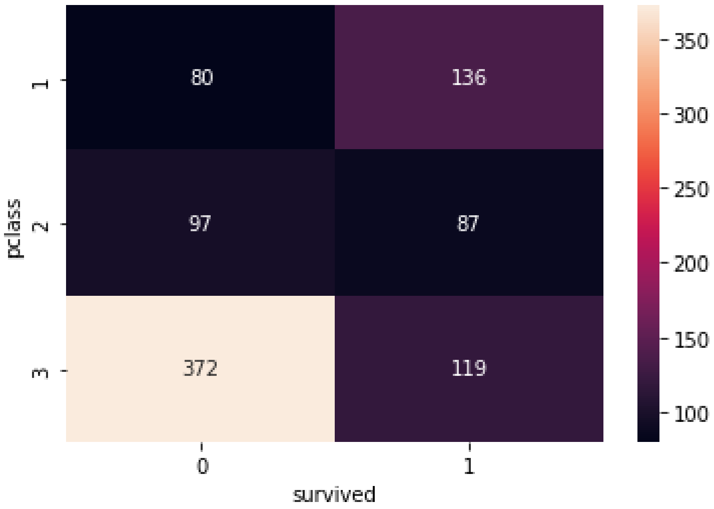
Pclass (Ordinal Feature) vs Survived

In [161…

```
# Group the dataset by Pclass and Survived and then unstack them
group = titanic.groupby(['pclass', 'survived'])

pclass_survived = group.size().unstack()

# Heatmap - Color encoded 2D representation of data.
sns.heatmap(pclass_survived, annot = True, fmt ="d")
```

Out[161…

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3d9e33acd0>
```


pclass
survived