As of now , we have a dataset named as titanic , & now we will perform EDA(Exploretory Data Analysis)

Lets import some libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Now we will read the datasets

```
df = pd.read_csv('/content/titanic_datasets.csv')
```

```
df.shape
```

```
(418, 12)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Survived     418 non-null    int64  
 2   Pclass       418 non-null    int64  
 3   Name         418 non-null    object 
 4   Sex          418 non-null    object 
 5   Age          332 non-null    float64
 6   SibSp        418 non-null    int64  
 7   Parch        418 non-null    int64  
 8   Ticket       418 non-null    object 
 9   Fare         417 non-null    float64
 10  Cabin        91 non-null     object 
 11  Embarked     418 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 39.3+ KB
```

```
df.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch
count	418.000000	418.000000	418.000000	332.000000	418.000000	418.000000
mean	1100.500000	0.363636	2.265550	30.272590	0.447368	0.392344
std	120.810458	0.481622	0.841838	14.181209	0.896760	0.981429
min	892.000000	0.000000	1.000000	0.170000	0.000000	0.000000
25%	996.250000	0.000000	1.000000	21.000000	0.000000	0.000000
50%	1100.500000	0.000000	3.000000	27.000000	0.000000	0.000000
75%	1204.750000	1.000000	3.000000	39.000000	1.000000	0.000000
max	1309.000000	1.000000	3.000000	76.000000	8.000000	9.000000

Now we will try to find out missing values , duplicates values & try to fill it out

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

```
               0
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age           86
SibSp          0
Parch          0
Ticket         0
Fare           1
Cabin        327
Embarked       0

dtype: int64
```

```
df['Age'].fillna(df['Age'].mean(),inplace=True)
```

```
5544653.py:1: FutureWarning: A value is trying to be set on a copy of a DataFr
ge in pandas 3.0. This inplace method will never work because the intermediate

g 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, ir
'Age'].mean(),inplace=True)
```

```
df['Fare'].fillna(df['Fare'].mean(), inplace=True)
```

```
/tmp/ipython-input-2379335405.py:1: FutureWarning: A value is trying to be se
The behavior will change in pandas 3.0. This inplace method will never work b

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.


  df['Fare'].fillna(df['Fare'].mean(), inplace=True)
```

We can see that 'Age' column are in float64 , so we will change it in 'int64' ,so that we can reduce the memory.

```
df['Age'] = df['Age'].astype('int64')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Survived     418 non-null    int64  
 2   Pclass       418 non-null    int64  
 3   Name         418 non-null    object 
 4   Sex          418 non-null    object 
 5   Age          418 non-null    int64  
 6   SibSp        418 non-null    int64  
 7   Parch        418 non-null    int64  
 8   Ticket       418 non-null    object 
 9   Fare         418 non-null    float64
 10  Cabin        91 non-null     object 
 11  Embarked     418 non-null    object 
dtypes: float64(1), int64(6), object(5)
memory usage: 39.3+ KB
```

```
df.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch
count	418.000000	418.000000	418.000000	418.000000	418.000000	418.000000
mean	1100.500000	0.363636	2.265550	30.191388	0.447368	0.392344
std	120.810458	0.481622	0.841838	12.654104	0.896760	0.981429
min	892.000000	0.000000	1.000000	0.000000	0.000000	0.000000
25%	996.250000	0.000000	1.000000	23.000000	0.000000	0.000000
50%	1100.500000	0.000000	3.000000	30.000000	0.000000	0.000000
75%	1204.750000	1.000000	3.000000	35.750000	1.000000	0.000000
max	1309.000000	1.000000	3.000000	76.000000	8.000000	9.000000

```
df.duplicated().sum()
```

```
np.int64(0)
```

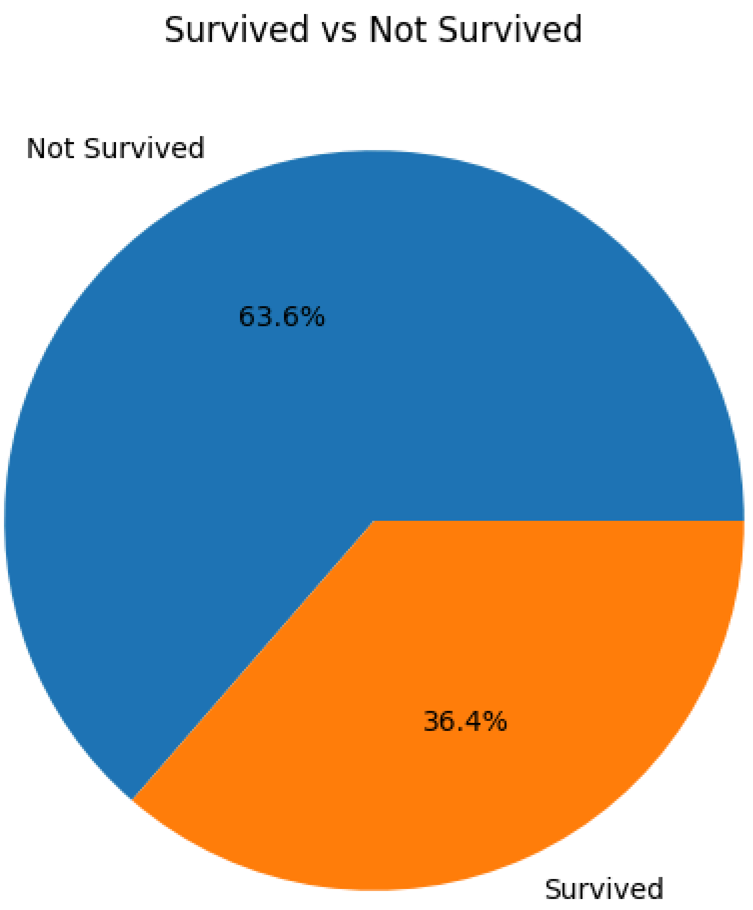
```
df.head(7)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
0	892	0	3	Kelly, Mr. James	male	34	0	0	330911
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47	1	0	363272
2	894	0	2	Myles, Mr. Thomas Francis	male	62	0	0	240276

Next steps: Generate code with df New interactive sheet

As of now , we are done with data cleaning , now we will focus on some graphs that can make sense

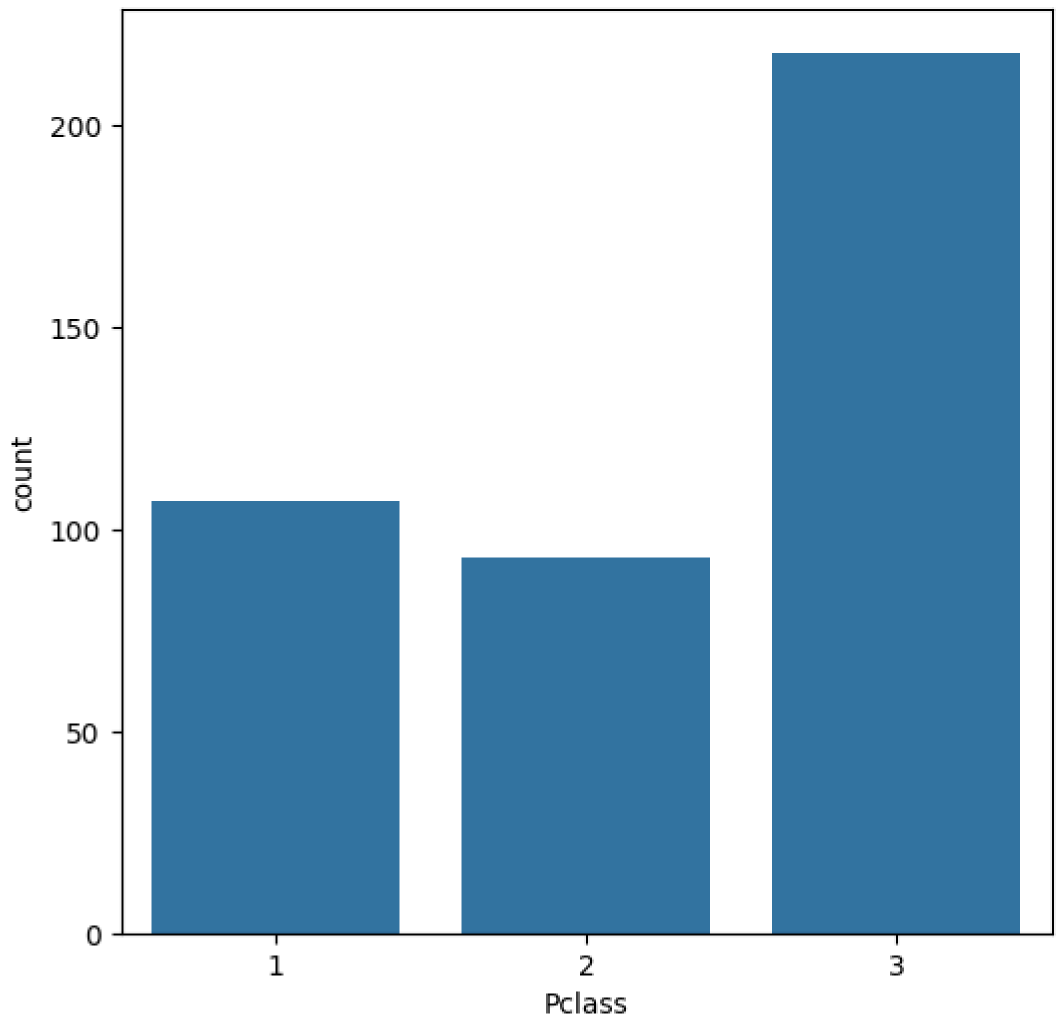
```
plt.figure(figsize=(6,6))
plt.pie(df['Survived'].value_counts(),labels=['Not Survived','Survived'],aut
plt.title('Survived vs Not Survived')
plt.show()
```



- 36.4% of the total population had been survived
- 63.6% of total population hadn't survived

```
plt.figure(figsize = (6,6))
sns.countplot(data = df , x = 'Pclass')
```

```
<Axes: xlabel='Pclass', ylabel='count'>
```



- Most of passengers(around 200+) travelled in Pclass3.
- Pclass 1 have around 100+ passengers.
- Pclass 2 have the less passengers(under 100) compare to other Pclass

```
df['Sex'].value_counts()
```

	count
Sex	
male	266
female	152

dtype: int64

- The total Number of Male pupulations is 266
- The total Number of Female pupulations is 152

```
family = df['SibSp'] + df['Parch']
family.value_counts()
```

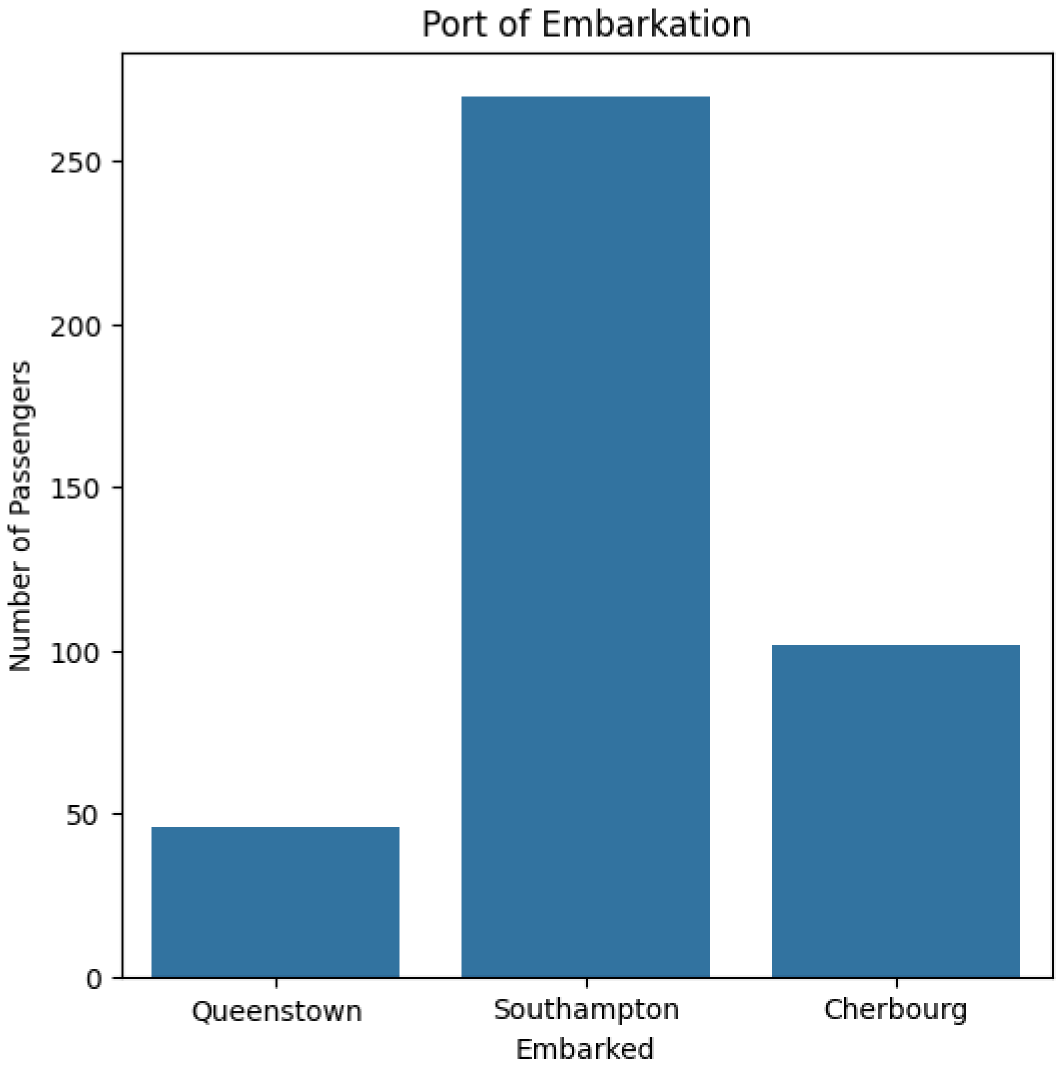
```
      count
 0      253
 1       74
 2       57
 3       14
 4        7
10        4
 6        4
 5        3
 7        2

dtype: int64
```

Most of the people were travelling alone

```
plt.figure(figsize = (6,6))
ax =sns.countplot(data = df , x = 'Embarked')
ax.set_xticklabels(['Queenstown' , 'Southampton' , 'Cherbourg'])
ax.set_xlabel('Embarked')
ax.set_ylabel('Number of Passengers')
ax.set_title('Port of Embarkation')
```

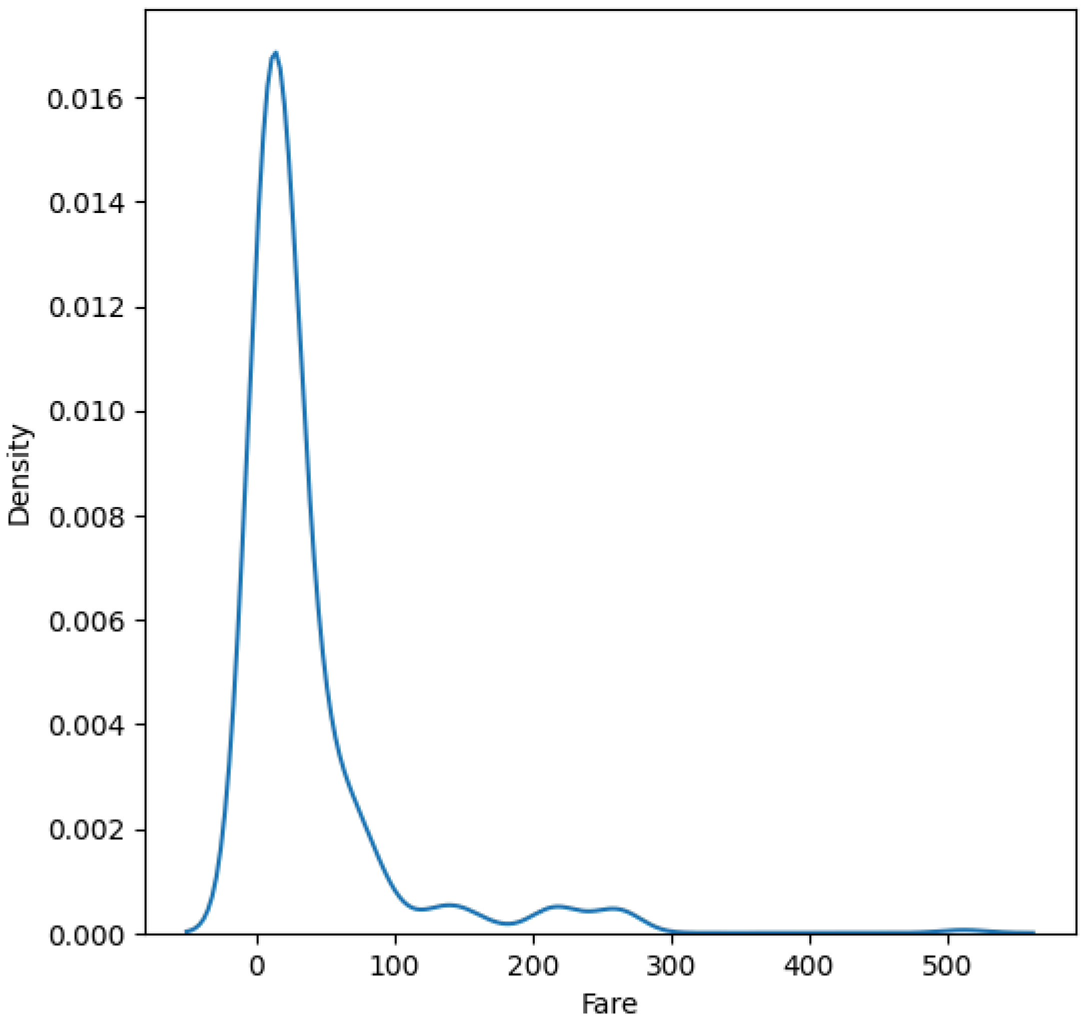
```
/tmp/ipython-input-3998979463.py:3: UserWarning: set_ticklabels() should only
  ax.set_xticklabels(['Queenstown' , 'Southampton' , 'Cherbourg'])
Text(0.5, 1.0, 'Port of Embarkation')
```



Most of the people Embarked from Southampton & then Cherbourg & very less people embarked from Queenstown

```
plt.figure(figsize=(6,6))
sns.kdeplot(data=df,x='Fare')
```

```
<Axes: xlabel='Fare', ylabel='Density'>
```

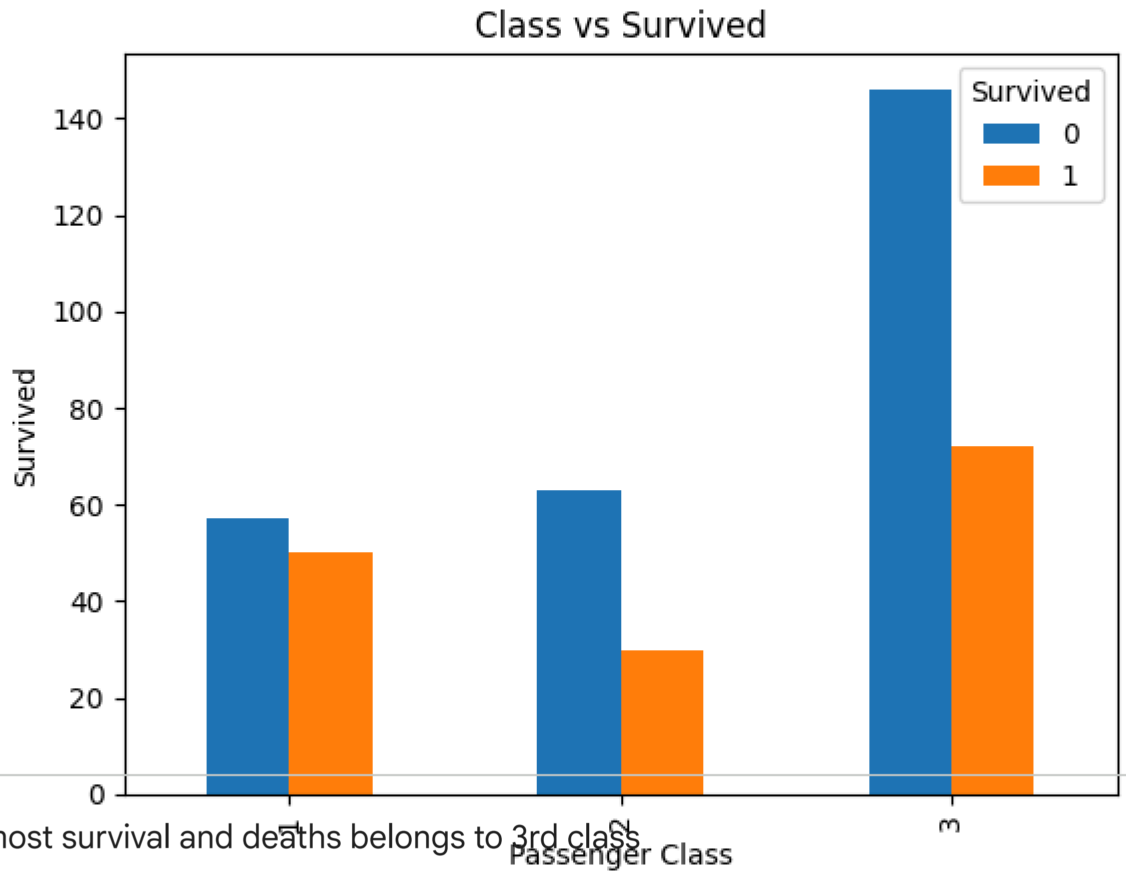


This graphs show the distributions of the Fare , there are some people who paid really high price & mostly people had fare of under 100

```
pd.crosstab(df['Sex'],df['Survived'])
```

Survived	0	1
Sex		
female	0	152
male	266	0

```
table = pd.crosstab(df['Pclass'],df['Survived'])
table.plot(kind='bar')
plt.title('Class vs Survived')
plt.xlabel('Passenger Class')
plt.ylabel('Survived')
plt.show()
```



The most survival and deaths belongs to 3rd class

```
table = pd.crosstab(df['Embarked'],df['Survived'])
table.plot(kind='bar')
plt.title('Embraked vs Survived')
plt.xlabel('Embarkation port')
plt.ylabel('Survived')
plt.xticks(
    ticks=range(3),
    labels=['Cherbourg','Queenstown','Southampton'],
    rotation=0
)
plt.show()
```