EXPLORATORY DATA ANALYSIS ON TITANIC DATASET

Introduction:

I will be undertaking exploratory data analysis on the famous Titanic dataset for the sake of this honours project. In the parts that follow, I will describe the overall structure of this dataset, as well as some of the attempts at cleaning the data and designing certain characteristics. This dataset has provided me with some critical results and insights. I will then form some hypotheses about the data and run tests to determine its importance. Finally, I shall share specifics ideas for more data analysis and a discussion of the quality of this data collection

Description on the dataset:

The Titanic dataset will be used in this analysis. It contains information on the passengers of the Titanic. I picked this data set since the characteristics are simple to understand and it is an excellent dataset for machine learning. There are 891 rows in the dataset and 11 columns in total.

In [48]:
```
# Load the required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from scipy import stats as st

# Load the data
df = pd.read_csv('titanic_train.csv')
```

In [7]:
```
# View the data
df.head()
```

Out[7]:

	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 [8]:
```
# Basic information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [10]:
```
# Describe the data
df.describe()
```

Out[10]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [11]:
```
# Find the duplicates
df.duplicated().sum()
```

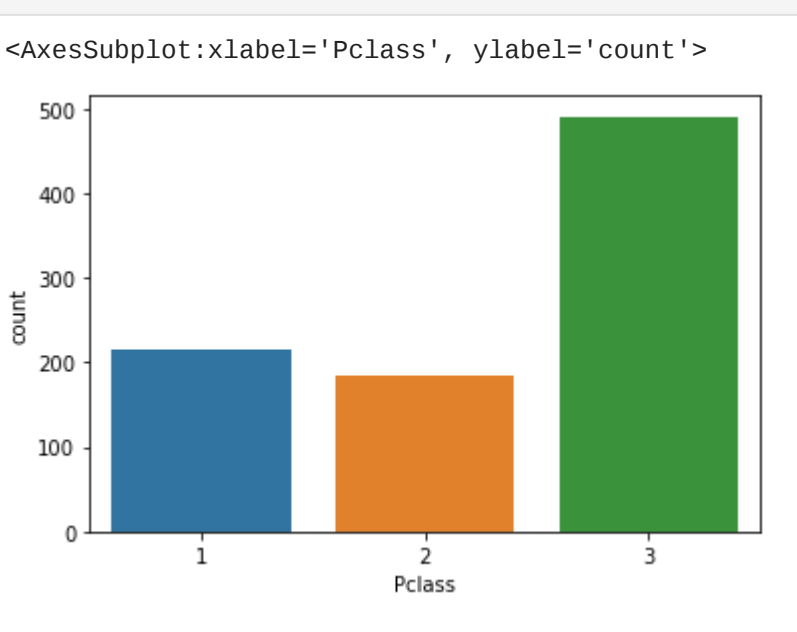
Out[11]: 0

In [16]:
```
# unique values
df['Pclass'].unique()
df['Survived'].unique()
df['Sex'].unique()
```

Out[16]: array(['male', 'female'], dtype=object)

In [24]:
```
# Plot the unique values
sns.countplot(df['Pclass'])
```

Out[24]: <AxesSubplot:xlabel='Pclass', ylabel='count'>



In [25]:
```
# Find null values
df.isnull().sum()
```

Out[25]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [26]:
```
# Replace null values
df.replace(np.nan,'0',inplace = True)

# Check the changes now
df.isnull().sum()
```

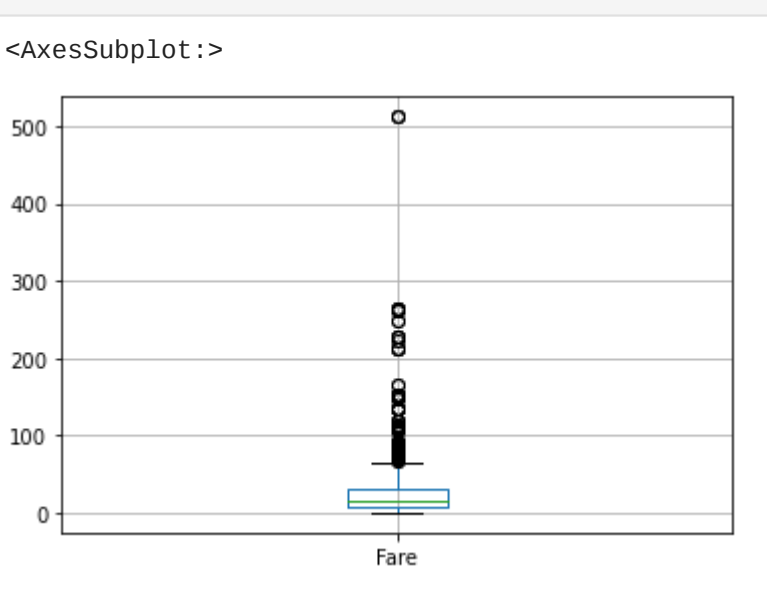
Out[26]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

In [27]:
```
# Datatypes
df.dtypes
```

Out[27]:
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age             object
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

In [29]:
```
# Boxplot
df[['Fare']].boxplot()
```

Out[29]: <AxesSubplot:>



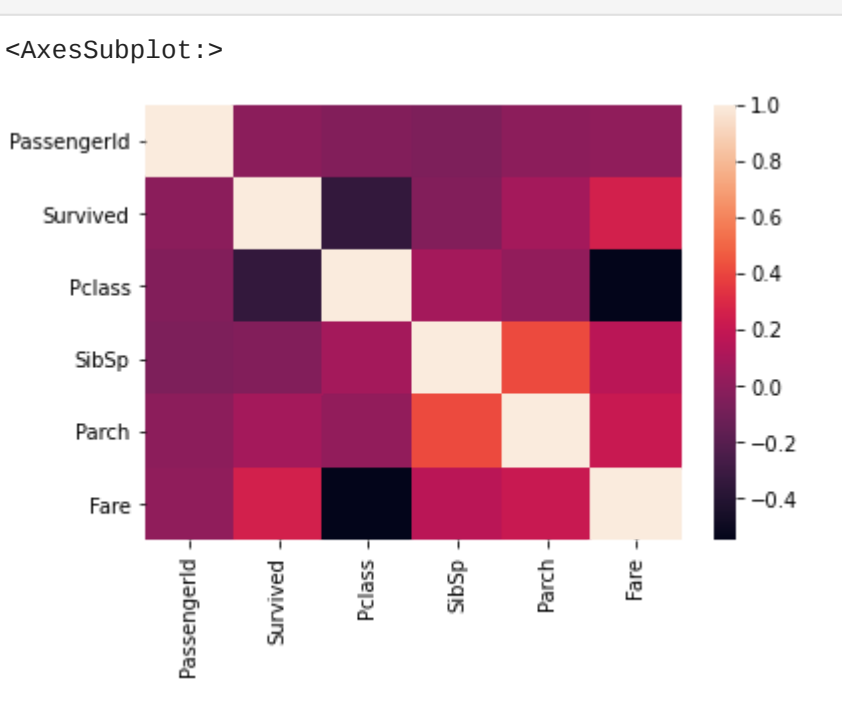
In [30]:
```
# Correlation
df.corr()
```

Out[30]:

	PassengerId	Survived	Pclass	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	0.083081	0.018443	-0.549500
SibSp	-0.057527	-0.035322	0.083081	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.159651	0.216225	1.000000

In [31]:
```
# Correlation plot
sns.heatmap(df.corr())
```

Out[31]: <AxesSubplot:>

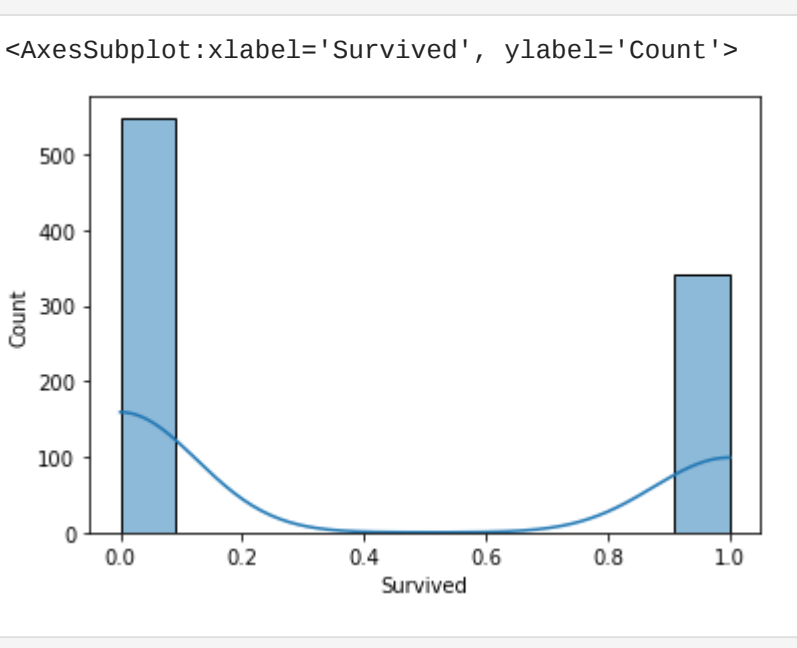


The two main variables we are concenred with are "Survived"and "Pclass".

- "Survived" is a binary variable that refers to whether that person survivied
- "Pclass": is a categorical variable that refers to the class of the person in the ship (first, second, or third)

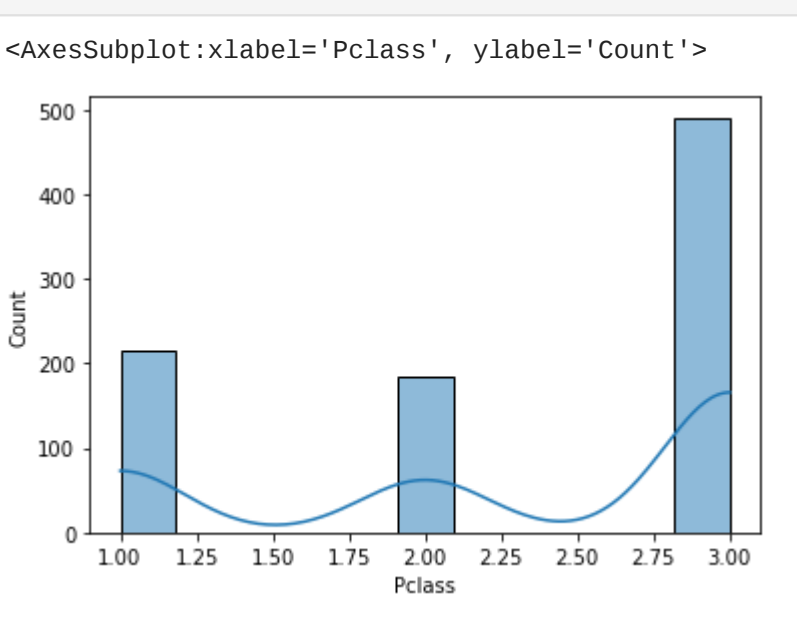
In [64]:
```
# histplot of the Survived people
sns.histplot(data=df, x="Survived", kde=True)
```

Out[64]: <AxesSubplot:xlabel='Survived', ylabel='Count'>



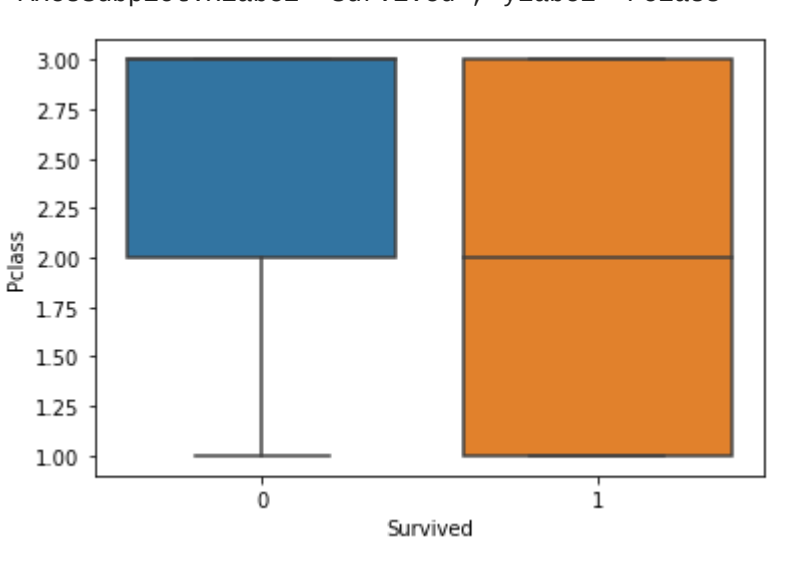
In [65]:
```
# histplot of the Pclass
sns.histplot(data=df, x="Pclass", kde=True)
```

Out[65]: <AxesSubplot:xlabel='Pclass', ylabel='Count'>



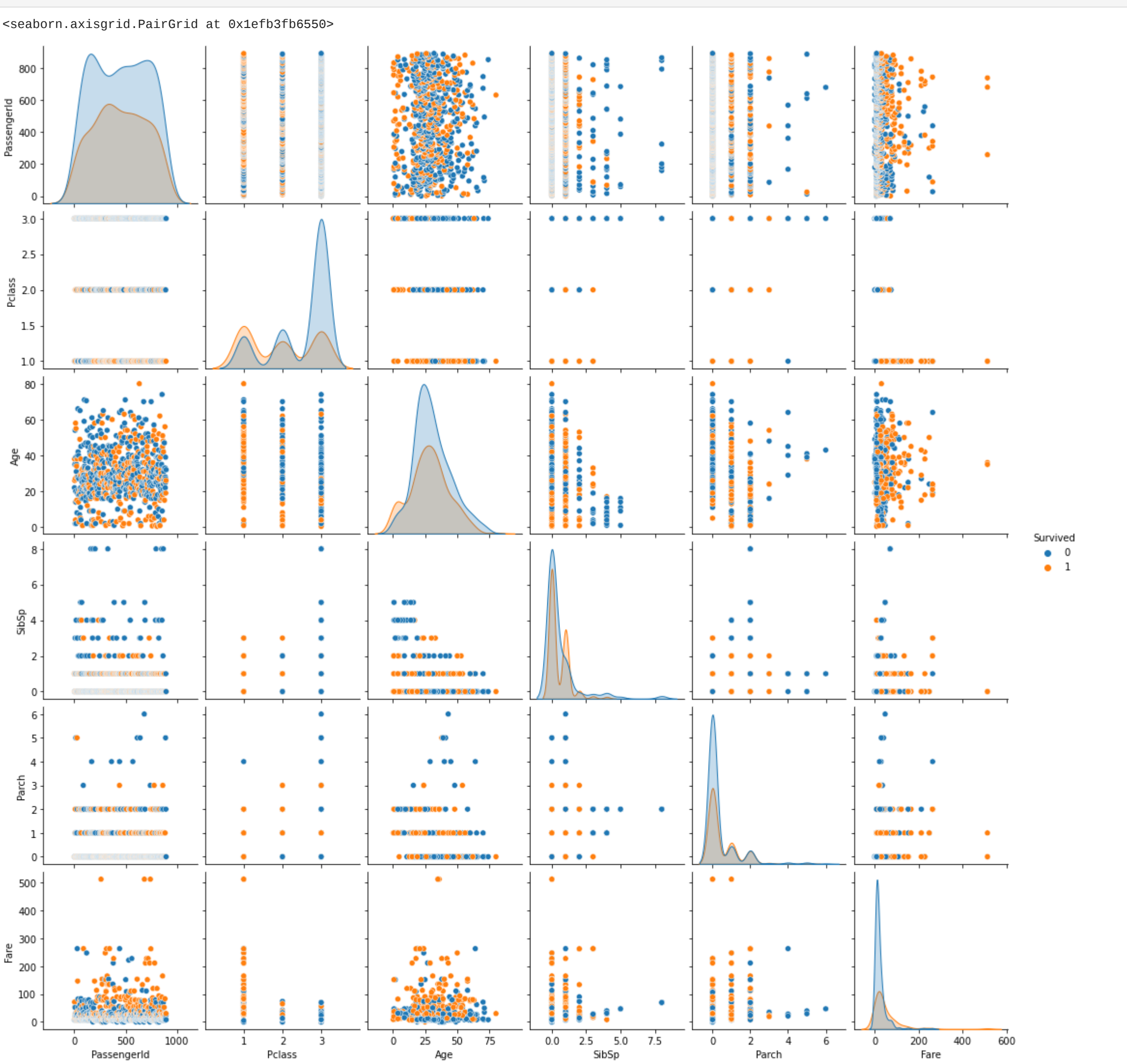
In [63]:
```
# boxplot of Survived against Pclass
sns.boxplot(x="Survived", y="Pclass", data=df)
```

Out[63]: <AxesSubplot:xlabel='Survived', ylabel='Pclass'>



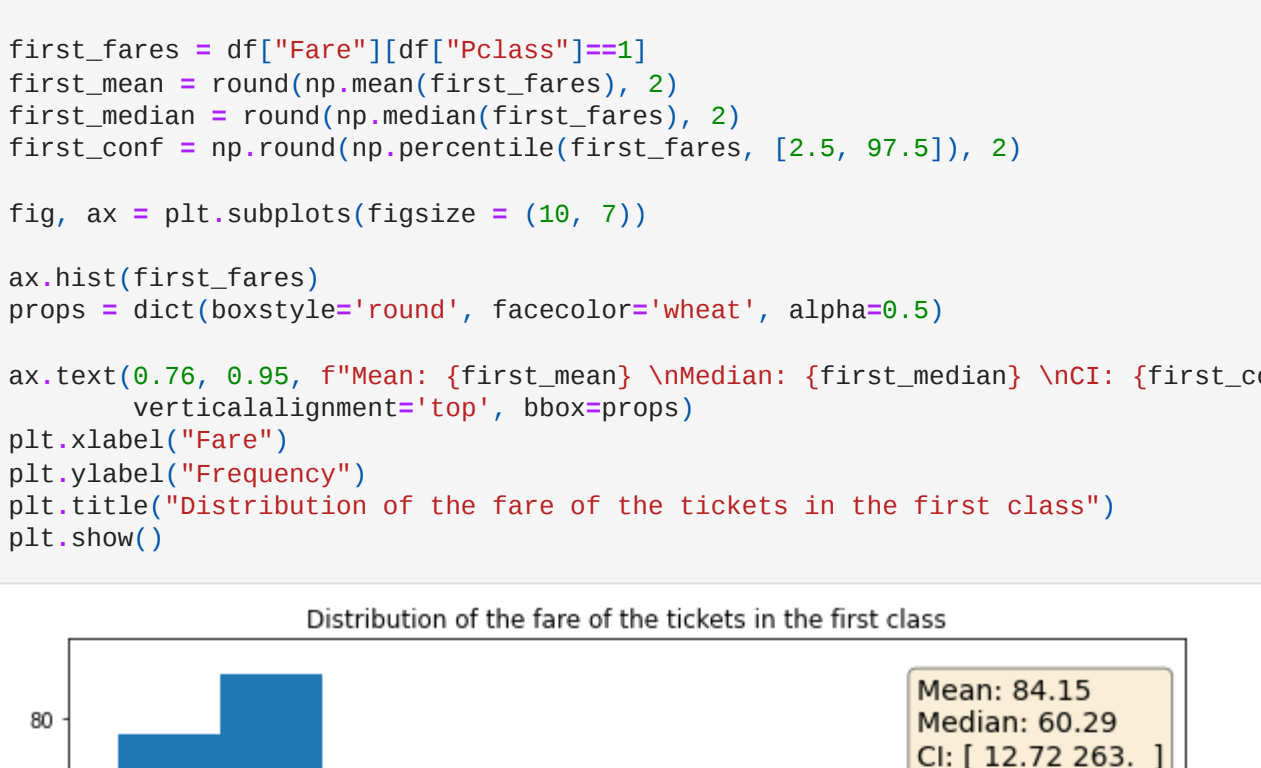
In [62]:
```
# pairplot with hue Survived
sns.pairplot(df, hue ='Survived')
```

Out[62]: <seaborn.axisgrid.PairGrid at 0x1efb3fb6550>

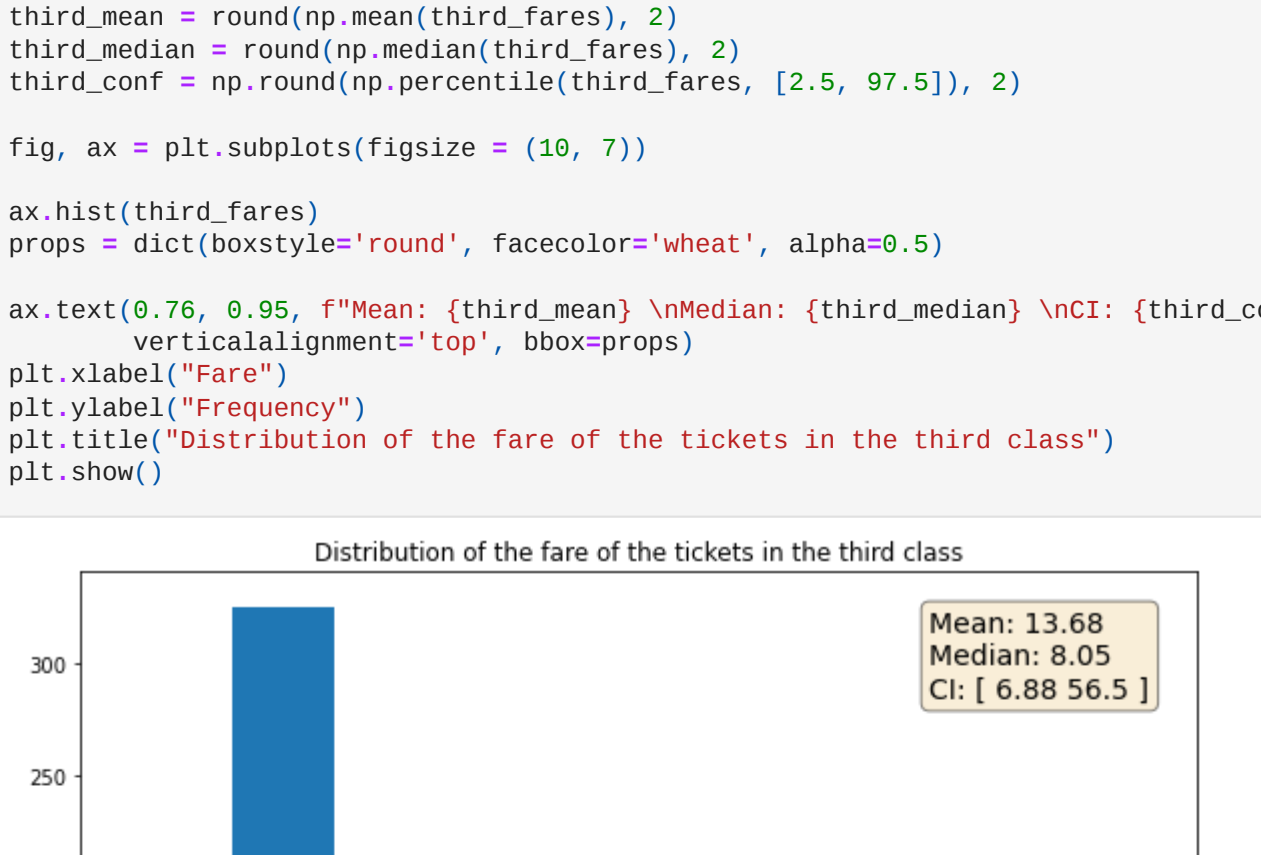


Note: It seems safe to assume that rich people were on the first class while poor were on the third but let's check that using the "Fare" variable which refers to how much people paid.

In [38]:
```
# make histograms of fares over the two classes

first_fares = df["Fare"][df["Pclass"]==1]
first_mean = round(np.mean(first_fares), 2)
first_median = round(np.median(first_fares), 2)
first_conf = np.round(np.percentile(first_fares, [2.5, 97.5]), 2)

fig, ax = plt.subplots(figsize = (10, 7))

ax.hist(first_fares)
props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)

ax.text(0.76, 0.95, f"Mean: {first_mean} \nMedian: {first_median} \nCI: {first_conf}", transform=ax.transAxes, fontsize=14,
        verticalalignment='top', bbox=props)
plt.xlabel("Fare")
plt.ylabel("Frequency")
plt.title("Distribution of the fare of the tickets in the first class")
plt.show()
```

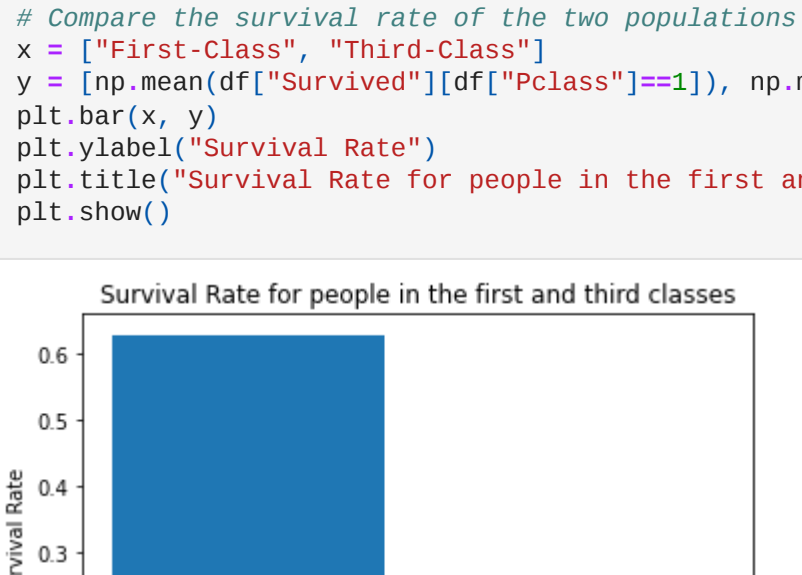


In [40]:
```
third_fares = df["Fare"][df["Pclass"]==3]
third_mean = round(np.mean(third_fares), 2)
third_median = round(np.median(third_fares), 2)
third_conf = np.round(np.percentile(third_fares, [2.5, 97.5]), 2)

fig, ax = plt.subplots(figsize = (10, 7))

ax.hist(third_fares)
props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)

ax.text(0.76, 0.95, f"Mean: {third_mean} \nMedian: {third_median} \nCI: {third_conf}", transform=ax.transAxes, fontsize=14,
        verticalalignment='top', bbox=props)
plt.xlabel("Fare")
plt.ylabel("Frequency")
plt.title("Distribution of the fare of the tickets in the third class")
plt.show()
```



Our assumption seems fair, First class had rich people who paid more and third class had people who paid less.

In [42]:
```
# Compare the survival rate of the two populations
x = ["First-Class", "Third-Class"]
y = [np.mean(df["Survived"][df["Pclass"]==1]), np.mean(df["Survived"][df["Pclass"]==3])]
plt.bar(x, y)
plt.ylabel("Survival Rate")
plt.title("Survival Rate for people in the first and third classes")
plt.show()
```

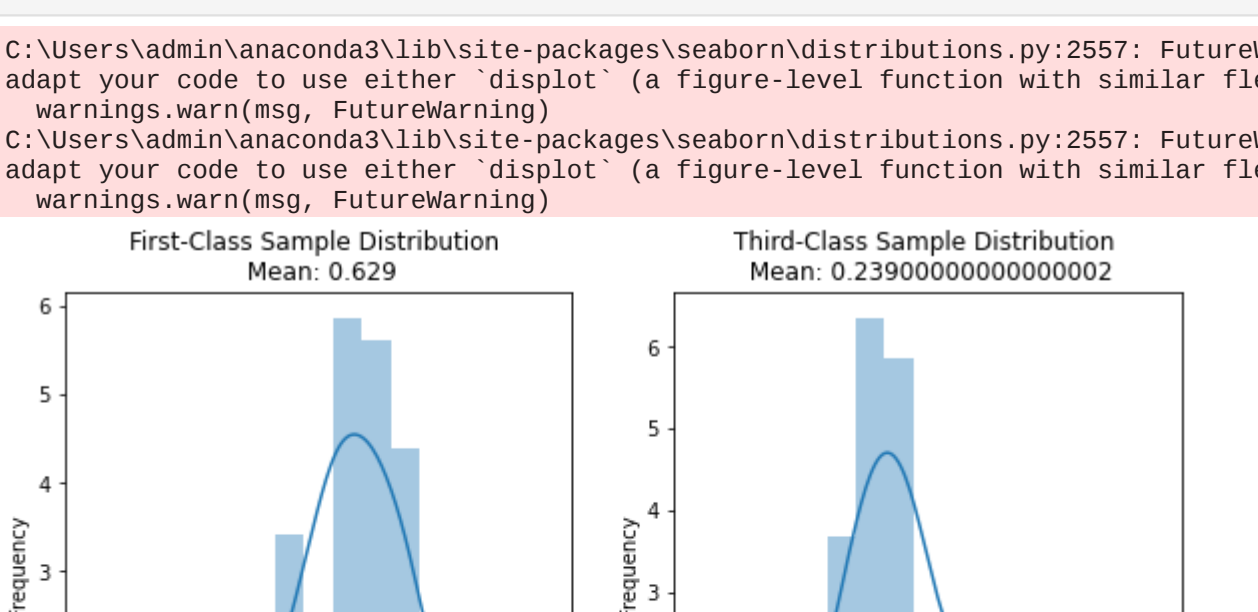


To make sure the two sample population distributions are normally distributed, let's use the central limit theorem and get a large enough sample to use the Z-test

In [49]:
```
# get samples using Central Limit Theorem
First_Class_Sample = np.array([np.mean(df[df["Pclass"]==1].sample(20)["Survived"].values) for i in range(100)])
third_Class_Sample = np.array([np.mean(df[df["Pclass"]==3].sample(20)["Survived"].values) for i in range(100)])
```

In [53]:
```
# make plots to check if the distributions are normal.
plt.subplots(1, 2, figsize = (10, 5))
plt.subplot(1,2, 1)
sns.distplot(First_Class_Sample)
plt.title(f"First-Class Sample Distribution \n Mean: {np.mean(First_Class_Sample)}")
plt.xlabel("Survival Rate")
plt.ylabel("Frequency")
plt.subplot(1, 2, 2)
sns.distplot(third_Class_Sample)
plt.title(f"Third-Class Sample Distribution \n Mean: {np.mean(third_Class_Sample)}")
plt.xlabel("Survival Rate")
plt.ylabel("Frequency")
plt.show()
```

```
C:\Users\admin\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\admin\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



Finally, calculating the Z-score and the P-value while assuming a significance level of 0.05.

In [56]:
```
# calculate the z-score and the p-value
effect = np.mean(First_Class_Sample) - np.mean(third_Class_Sample)
sigma_first = np.std(First_Class_Sample)
sigma_third = np.std(third_Class_Sample)
sigma_difference = np.sqrt((sigma_first**2)/len(First_Class_Sample)  +  (sigma_third**2)/len(third_Class_Sample))
z_score = effect / sigma_difference
```

In [57]:
```
# P-value for two-tailed test
st.norm.sf(abs(z_score))*2
```

Out[57]: 5.863246227865918e-222

Conclusion:

- The P-value is 5.86e-222. This is definitely much smaller than the significance level of 0.05 that we set. That P-value means that if we assume that the null hypothesis is true, then the probability of observing that effect by random is 5.86e-222% which is an extremely small probability that makes us comfortable in rejecting the Null hypothesis.

Quality of this data set:

Because of the significant number of null values in the data, this dataset is rather unclean. This was fixed by replacing the null values with zeros or by adding a new category. Otherwise, the attributes presented in this dataset were useful, and numerous insights on how people survived the Titanic may be gained.