```
# importing the required lybraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Importing the dataset
df=pd.read_csv('/content/drive/MyDrive/datasets/Titanic-Dataset.csv')
```

```
# Shape
df.shape
```

```
(891, 12)
```

```
# Previewing the data
df.head()
```

	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

Next steps: Generate code with df View recommended plots New interactive sheet

```
# Listing down the columns
df.columns.values
```

```
array(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'], dtype=object)
```

```
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
```

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

	0
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

Few conclusions

1. Missing values in Age, Cabin and Embarked columns
2. More than 70 percent values are missing in cabin columns, will have to drop
3. Few columns have inappropriate data types

```
# Dropping cabin column

df.drop(columns=['Cabin'],inplace=True)
```

```
# Imputing missing values for age
# Strategy - mean

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

```
<ipython-input-11-469df478e40e>:4: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained ass
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

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


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

```
# Imputing missing values for embarked

# finding the most appeared value in embarked column

df['Embarked'].value_counts()

# S it is

df['Embarked'].fillna('S', inplace=True)
```

```
<ipython-input-12-647ac518be7e>:9: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained ass
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

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


  df['Embarked'].fillna('S', inplace=True)
```

```
# Want to check one more thing...

# Should I change the SibSp and Parch to categories

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

SibSp	count
0	608
1	209
2	28
4	18
3	16
8	7
5	5

dtype: int64

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

```
df['Survived']=df['Survived'].astype('category')
df['Pclass']=df['Pclass'].astype('category')
df['Sex']=df['Sex'].astype('category')
df['Age']=df['Age'].astype('int')
df['Embarked']=df['Embarked'].astype('category')
```

```
df.info()
```

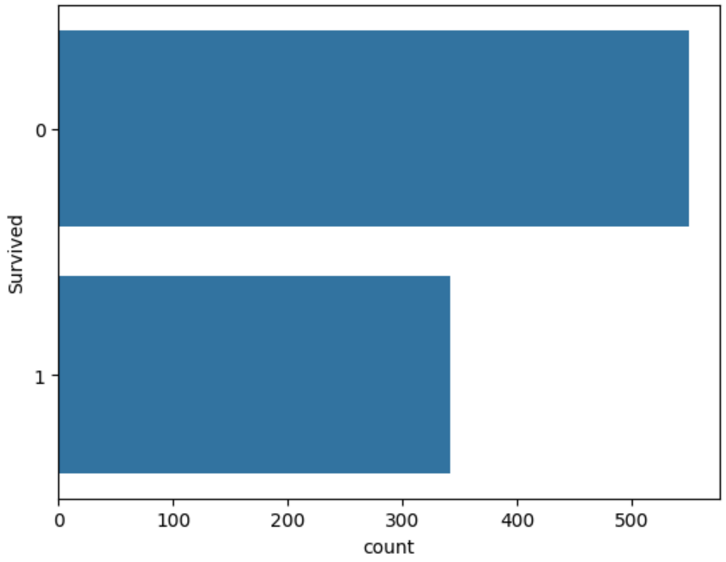
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype   
---  ------       --------------  -----   
 0   PassengerId  891 non-null    int64   
 1   Survived     891 non-null    category
 2   Pclass       891 non-null    category
 3   Name         891 non-null    object  
 4   Sex          891 non-null    category
 5   Age          891 non-null    int64   
 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  Embarked     891 non-null    category
dtypes: category(4), float64(1), int64(4), object(2)
memory usage: 52.8+ KB
```

```
# Five point summary
df.describe()
```

	PassengerId	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	891.000000	891.000000
mean	446.000000	29.544332	0.523008	0.381594	32.204208
std	257.353842	13.013778	1.102743	0.806057	49.693429
min	1.000000	0.000000	0.000000	0.000000	0.000000
25%	223.500000	22.000000	0.000000	0.000000	7.910400
50%	446.000000	29.000000	0.000000	0.000000	14.454200
75%	668.500000	35.000000	1.000000	0.000000	31.000000
max	891.000000	80.000000	8.000000	6.000000	512.329200

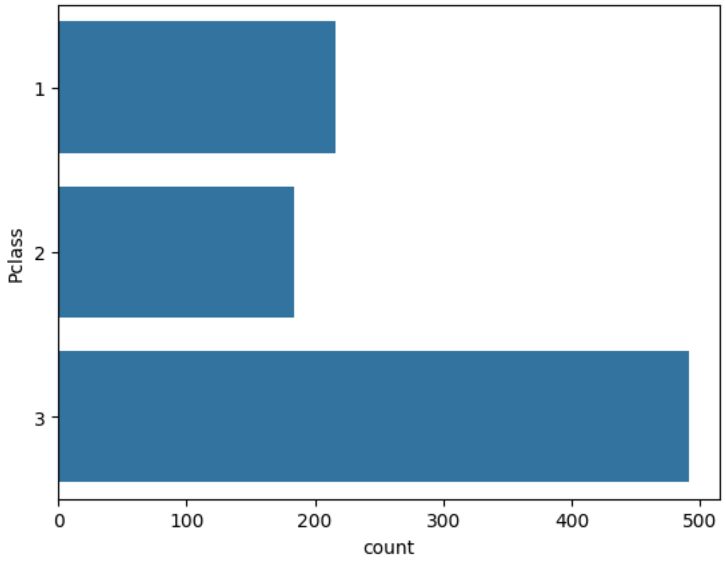
```
# Univariate Analysis

# Let's start with the Survived col

sns.countplot(df['Survived'])

death_percent=round((df['Survived'].value_counts().values[0]/891)*100)

print("Out of 891 {} people died in the accident".format(death_percent))
```

```
Out of 891 62 people died in the accident
```



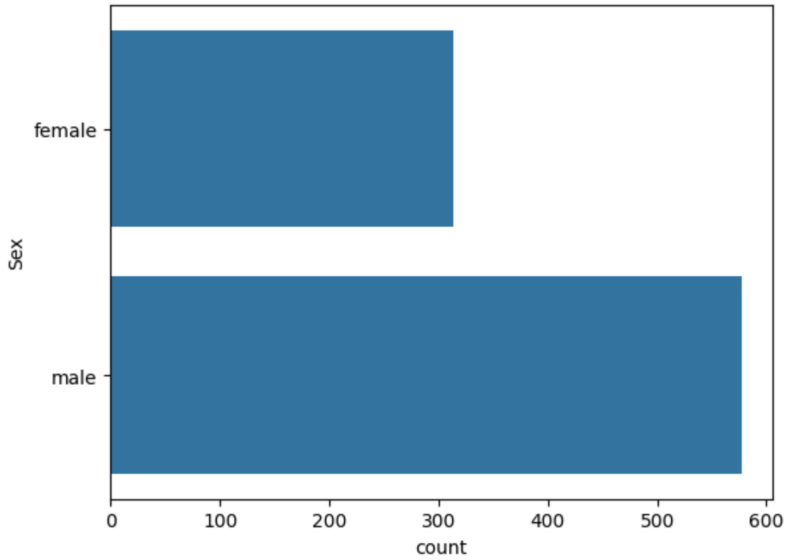
```
# Pclass column

print((df['Pclass'].value_counts()/891)*100)

sns.countplot(df['Pclass'])

# Conclusion : Pclass was the most crowded class
```

```
Pclass
3    55.106622
1    24.242424
2    20.650954
Name: count, dtype: float64
<Axes: xlabel='count', ylabel='Pclass'>
```



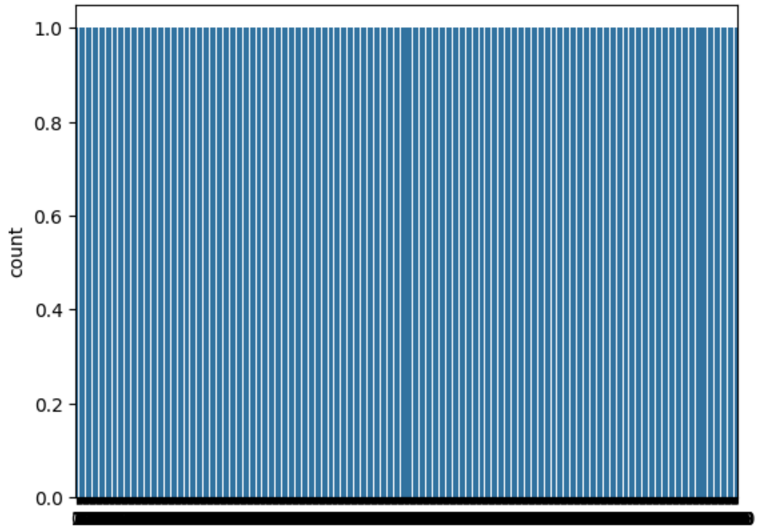
```
print((df['Sex'].value_counts()/891)*100)

sns.countplot(df['Sex'])
```

```
Sex
male      64.758698
female    35.241302
Name: count, dtype: float64
<Axes: xlabel='count', ylabel='Sex'>
```



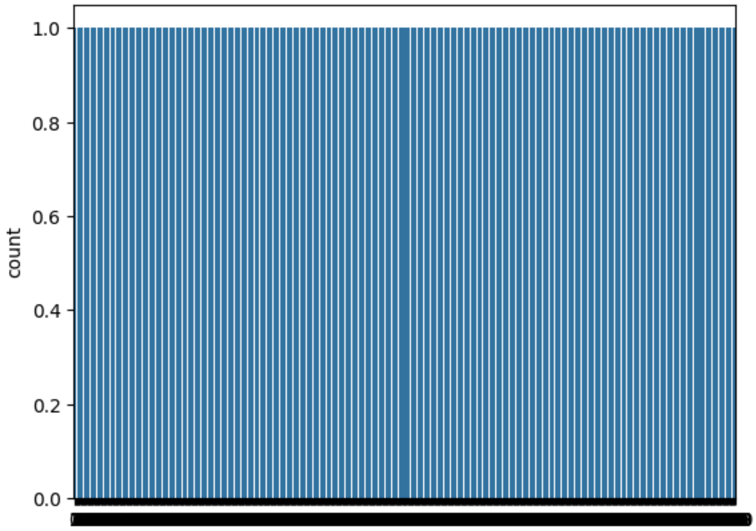
```
print(df['SibSp'].value_counts())

sns.countplot(df['SibSp'])
```

```
SibSp
0    608
1    209
2     28
4     18
3     16
8      7
5      5
Name: count, dtype: int64
<Axes: ylabel='count'>
```



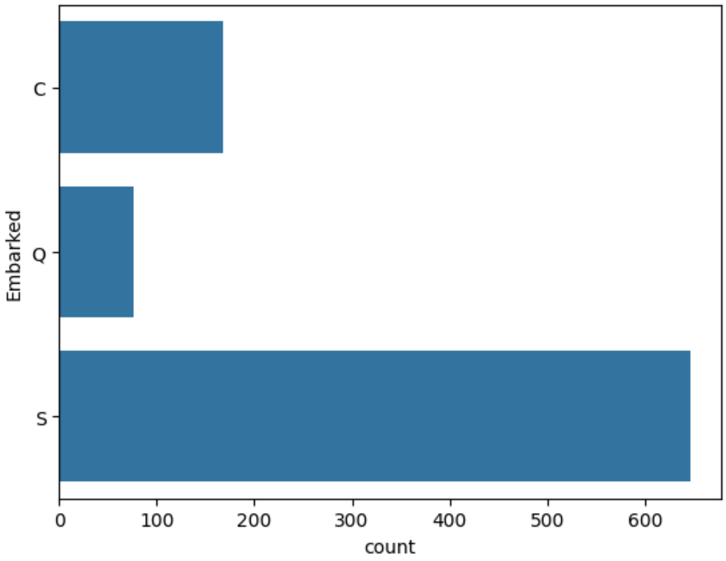
```
print((df['Parch'].value_counts()/891)*100)

sns.countplot(df['Parch'])
```

```
Parch
0    76.094276
1    13.243547
2     8.978676
5     0.561167
3     0.561167
4     0.448934
6     0.112233
Name: count, dtype: float64
<Axes: ylabel='count'>
```



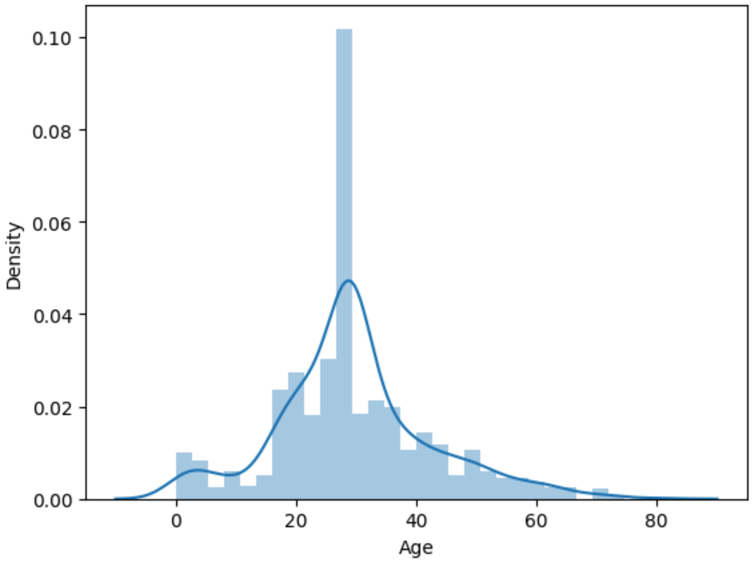
```
print((df['Embarked'].value_counts()/891)*100)

sns.countplot(df['Embarked'])
```

```
Embarked
S    72.502806
C    18.855219
Q     8.641975
Name: count, dtype: float64
<Axes: xlabel='count', ylabel='Embarked'>
```



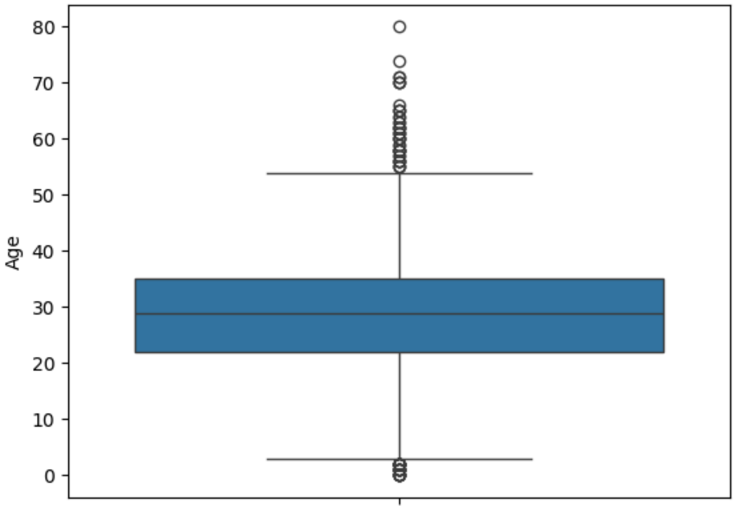
```
# Age column

sns.distplot(df['Age'])

print(df['Age'].skew())

print(df['Age'].kurt())
```

```
<ipython-input-24-ce823ca53eb8>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Age'])
0.45956263424701577
0.9865867453652877
```



```
sns.boxplot(df['Age'])
```

```
<Axes: ylabel='Age'>
```

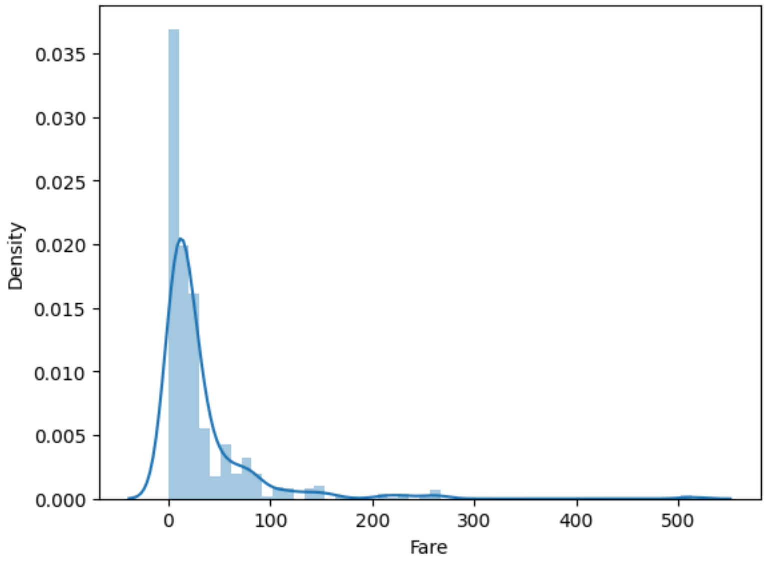


```
# Just out of curiosity

print("People with age in between 60 and 70 are",df[(df['Age']>60) & (df['Age']<70)].shape[0])
print("People with age greater than 70 and 75 are",df[(df['Age']>=70) & (df['Age']<=75)].shape[0])
print("People with age greater than 75 are",df[df['Age']>75].shape[0])

print('-'*50)

print("People with age between 0 and 1",df[df['Age']<1].shape[0])
```

```
People with age in between 60 and 70 are 15
People with age greater than 70 and 75 are 6
People with age greater than 75 are 1
--------------------------------------------------
People with age between 0 and 1 7
```

```
# Fare column

sns.distplot(df['Fare'])
```

```
<ipython-input-27-3001b72f0dd7>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Fare'])
<Axes: xlabel='Fare', ylabel='Density'>
```

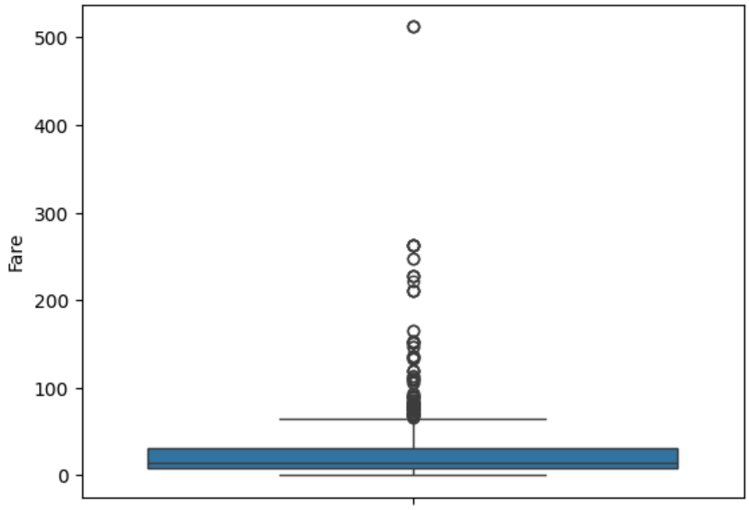


```
print(df['Fare'].skew())
print(df['Fare'].kurt())
```

```
4.787316519674893
33.39814088089868
```

```
sns.boxplot(df['Fare'])
```

```
<Axes: ylabel='Fare'>
```



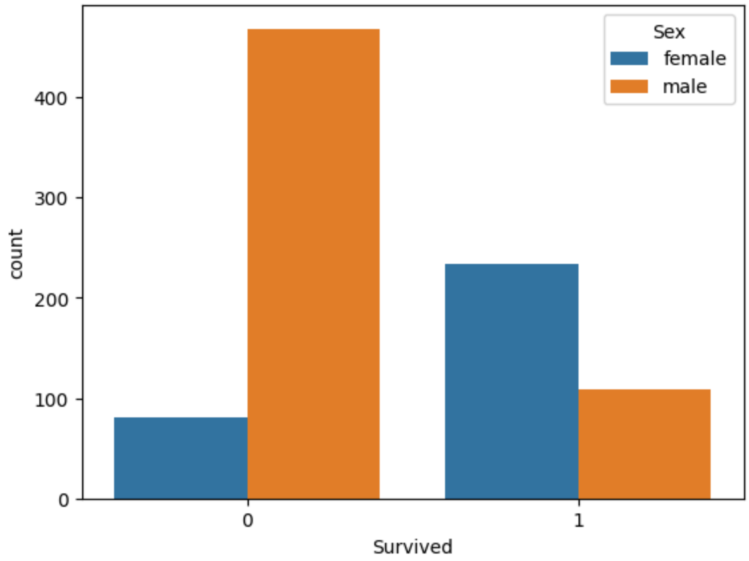
```
print("People with fare in between $200 and $300",df[(df['Fare']>200) & (df['Fare']<300)].shape[0])
print("People with fare in greater than $300",df[df['Fare']>300].shape[0])
```

```
People with fare in between $200 and $300 17
People with fare in greater than $300 3
```

```
# Survival with Sex

# Correcting the countplot call
sns.countplot(data=df, x='Survived', hue='Sex')

# Creating a crosstab and calculating percentages
survival_rate_by_sex = pd.crosstab(df['Sex'], df['Survived']).apply(lambda r: round((r/r.sum())*100, 1), axis=1)
```

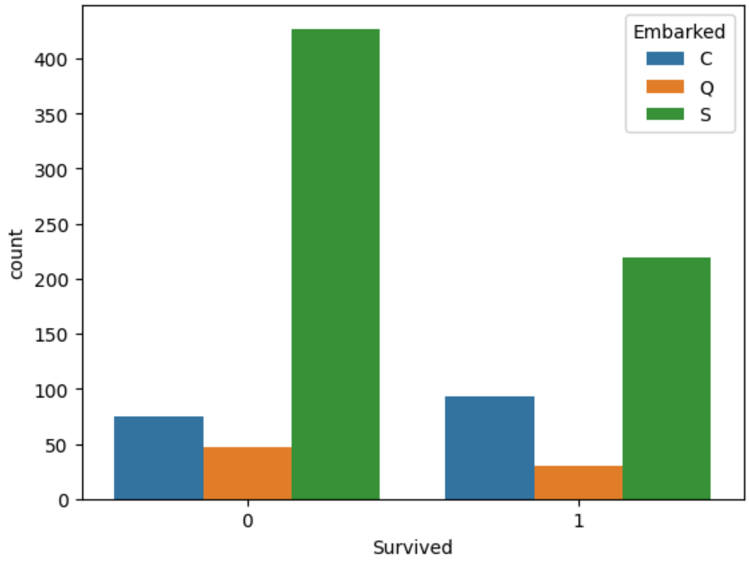
```
# Display the survival rates
print(survival_rate_by_sex)
```

```
Survived     0     1
Sex
female    25.8  74.2
male      81.1  18.9
```



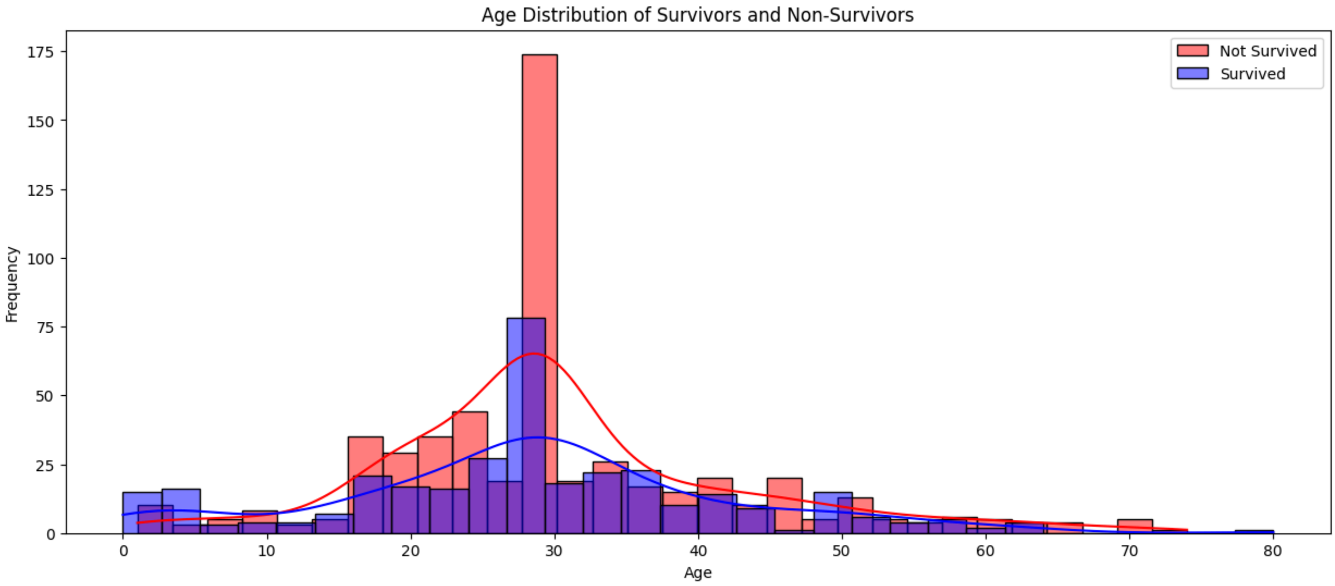
```
# Survival with Embarked

# Correcting the countplot call
sns.countplot(data=df, x='Survived', hue='Embarked')

# Creating a crosstab and calculating percentages
survival_rate_by_embarked = pd.crosstab(df['Embarked'], df['Survived']).apply(lambda r: round((r/r.sum())*100, 1), axis=1)

# Display the survival rates
print(survival_rate_by_embarked)
```

```
Survived     0     1
Embarked
C         44.6  55.4
Q         61.0  39.0
S         66.1  33.9
```



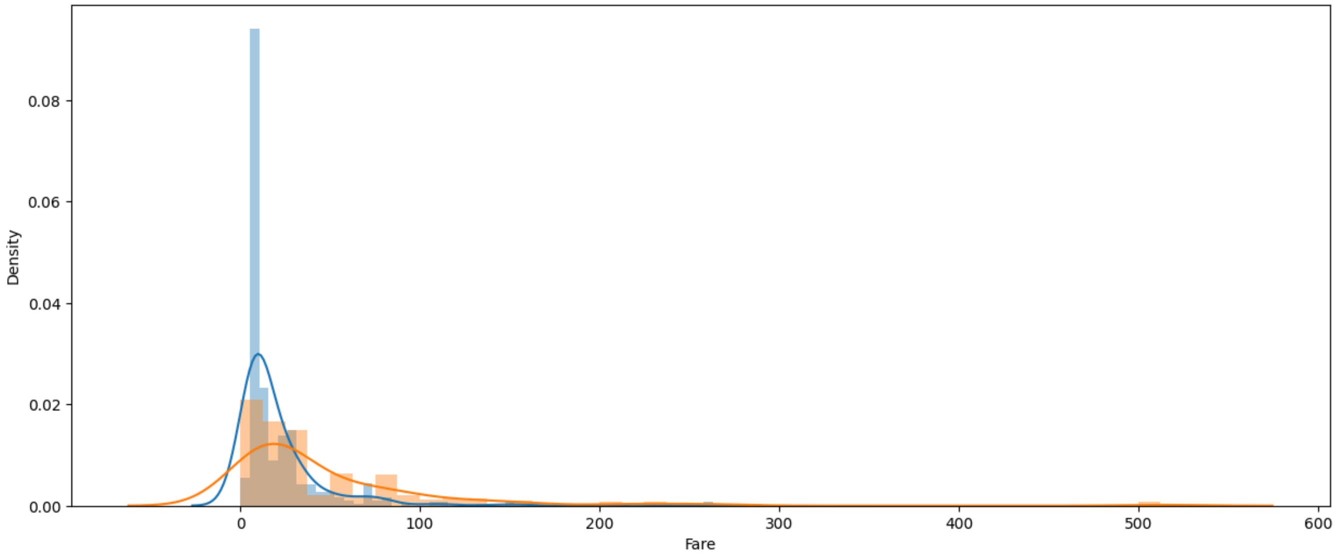
```
# Survived with Age

plt.figure(figsize=(15, 6))

# Plotting the distribution of Age for non-survivors
sns.histplot(df[df['Survived'] == 0]['Age'], kde=True, color='red', label='Not Survived', bins=30)

# Plotting the distribution of Age for survivors
sns.histplot(df[df['Survived'] == 1]['Age'], kde=True, color='blue', label='Survived', bins=30)
```

```
# Adding labels and title
plt.title('Age Distribution of Survivors and Non-Survivors')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.legend()

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



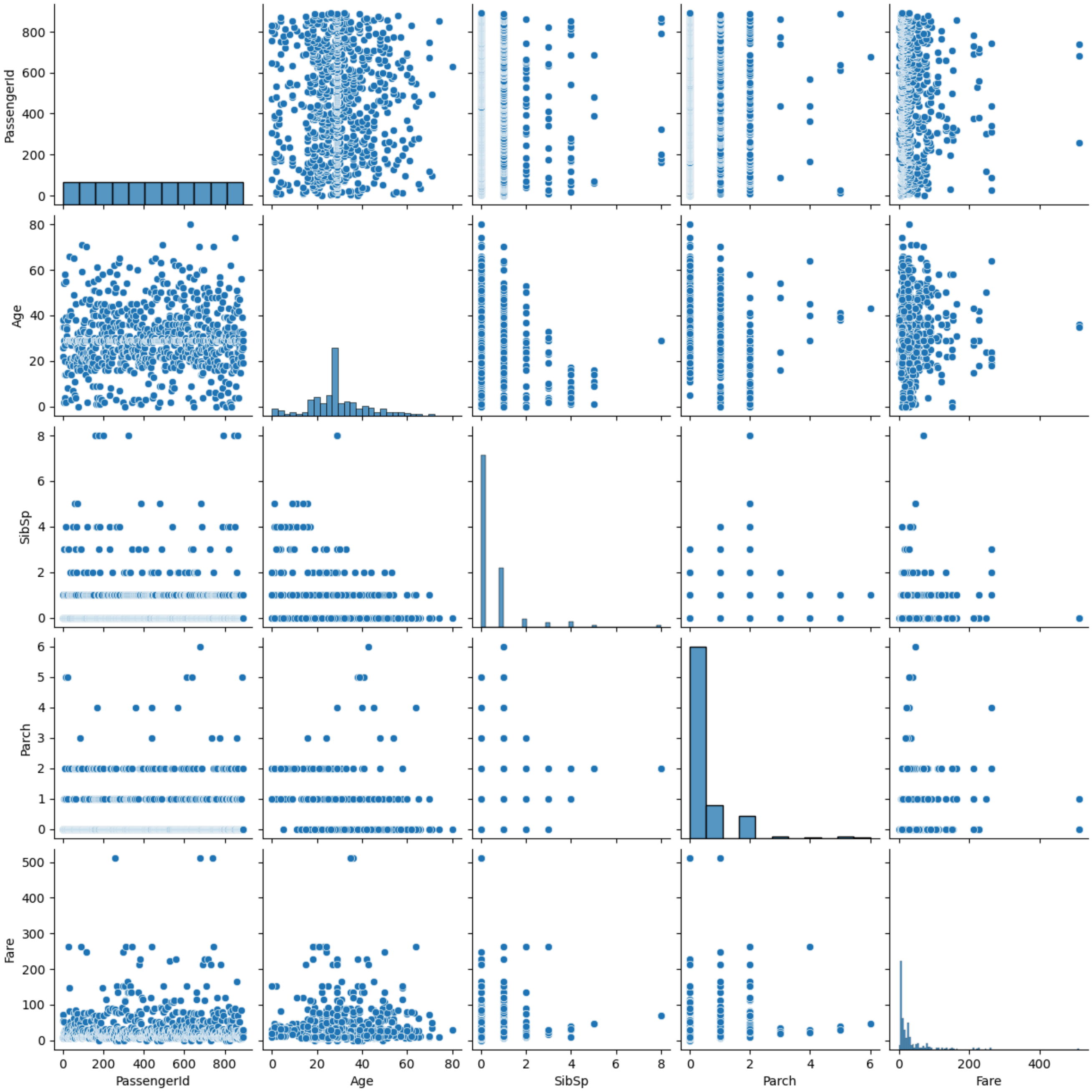
```
# Survived with Fare

plt.figure(figsize=(15,6))
sns.distplot(df[df['Survived']==0]['Fare'])
sns.distplot(df[df['Survived']==1]['Fare'])
```

```
<ipython-input-39-eeee0928512b>:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[df['Survived']==0]['Fare'])
<ipython-input-39-eeee0928512b>:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[df['Survived']==1]['Fare'])
<Axes: xlabel='Fare', ylabel='Density'>
```



```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7b4bd565e710>
```



Feature Engineering

```
# We will create a new column by the name of family which will be the sum of SibSp and Parch cols

df['family_size']=df['Parch'] + df['SibSp']
```

```
df.sample(5)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	family_size

```
# Now we will enginner a new feature by the name of family type

def family_type(number):
    if number==0:
        return "Alone"
    elif number>0 and number<=4:
        return "Medium"
    else:
        return "Large"
```

```
df['family_type']=df['family_size'].apply(family_type)
```

```
df.sample(5)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	family_size	family_type
227	228	0	3	Lovell, Mr. John Hall ("Henry")	male	20	0	0	A/5 21173	7.250	S	0	Alone
200	201	0	3	Vande Walle, Mr. Nestor Cyriel	male	28	0	0	345770	9.500	S	0	Alone
659	660	0	1	Newell, Mr. Arthur	male	58	0	2	35273	113.275	C	2	Medium

```
# Dropping SibSp, Parch and family_size

df.drop(columns=['SibSp','Parch','family_size'],inplace=True)
```

```
df.sample(5)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	Ticket	Fare	Embarked	family_type
442	443	0	3	Petterson, Mr. Johan Emil	male	25	347076	7.7750	S	Medium
302	303	0	3	Johnson, Mr. William Cahoone Jr	male	19	LINE	0.0000	S	Alone
223	224	0	3	Nenkoff, Mr. Christo	male	29	349234	7.8958	S	Alone
66	67	1	2	Nye, Mrs. (Elizabeth Ramell)	female	29	C.A. 29395	10.5000	S	Alone
185	186	0	1	Rood, Mr. Hugh Roscoe	male	29	113767	50.0000	S	Alone

```
pd.crosstab(df['family_type'], df['Survived']).apply(lambda r: round((r/r.sum())*100,1), axis=1)
```

Survived	0	1
family_type		
Alone	69.6	30.4
Large	85.1	14.9
Medium	44.0	56.0

```
# handling outliers in age(Almost normal)

df=df[df['Age']<(df['Age'].mean() + 3 * df['Age'].std())]
df.shape
```