In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```python
df = pd.read_csv("D:\Datasets\kaggle_Titanic_train.csv")
test = pd.read_csv("D:\Datasets\kaggle_Titanic_test.csv")
```

In [3]:
```python
print(df.shape)
print(test.shape)
```

```
(891, 12)
(418, 11)
```

In [4]:
```python
df.head()
```

Out[4]:

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

In [5]:
```
test.head()
```

Out[5]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Em
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	

In [6]:
```
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 [7]: test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Pclass       418 non-null    int64  
 2   Name         418 non-null    object 
 3   Sex          418 non-null    object 
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64  
 6   Parch        418 non-null    int64  
 7   Ticket       418 non-null    object 
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object 
 10  Embarked     418 non-null    object 
dtypes: float64(2), int64(4), object(5)
memory usage: 36.1+ KB
```

```
In [8]: df.columns
```

```
Out[8]: Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
               'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
              dtype='object')
```

```
In [9]: test.columns
```

```
Out[9]: Index(['PassengerId', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch',
               'Ticket', 'Fare', 'Cabin', 'Embarked'],
              dtype='object')
```

Data Understanding

- PassengerId : Unique PassengerId
- Survived : Passenger survived Yes(1) or No(0)
- Pclass : Passenger class (1st=1,2nd=2,3rd=3)
- Name : Passenger Name
- Sex : Male/Female
- Age : Passenger Age
- SibSp : # of siblings / spouses aboard the Titanic
- Parch : # of parents / children aboard the Titanic
- Ticket : Passenger ticket number
- Fare : Passenger fare
- Cabin : Passenger Cabin number
- Embarked : Port of Embarkation

```
In [10]: df['PassengerId'].nunique()
```

```
Out[10]: 891
```

```
In [11]: df['Survived'].value_counts()
```

```
Out[11]: 0    549
         1    342
         Name: Survived, dtype: int64
```

```
In [12]: df['Pclass'].value_counts()
```

```
Out[12]: 3    491
         1    216
         2    184
         Name: Pclass, dtype: int64
```

```
In [13]: df['Name'].nunique()
```

```
Out[13]: 891
```

```
In [14]: df['Sex'].value_counts()
```

```
Out[14]: male      577
         female    314
         Name: Sex, dtype: int64
```

```
In [15]: df['SibSp'].value_counts()
```

```
Out[15]: 0    608
         1    209
         2     28
         4     18
         3     16
         8      7
         5      5
         Name: SibSp, dtype: int64
```

```
In [16]: df['Parch'].value_counts()
```

```
Out[16]: 0    678
         1    118
         2     80
         5      5
         3      5
         4      4
         6      1
         Name: Parch, dtype: int64
```

```
In [17]: df['Embarked'].value_counts()
```

```
Out[17]: S    644
         C    168
         Q     77
         Name: Embarked, dtype: int64
```

In [18]:

```
continous=['PassengerId', 'Age', 'Fare']
discrete_categorical=['Name', 'Sex', 'Ticket', 'Cabin', 'Embarked']
discrete_count=['Survived', 'Pclass', 'SibSp', 'Parch']
```

Exploratory Data Analysis(EDA)

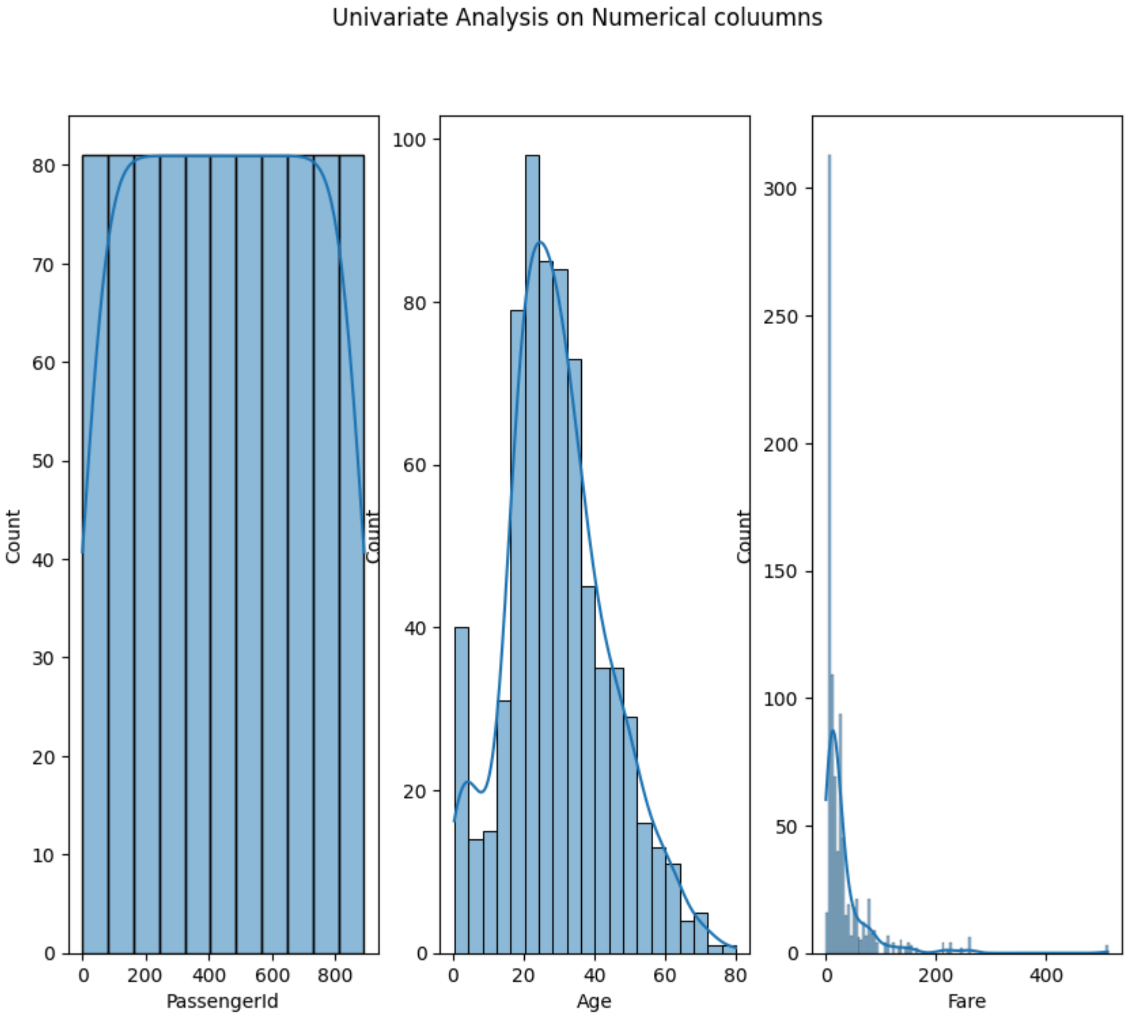
In [19]:

```
df[continous].describe()
```

Out[19]:

	PassengerId	Age	Fare
count	891.000000	714.000000	891.000000
mean	446.000000	29.699118	32.204208
std	257.353842	14.526497	49.693429
min	1.000000	0.420000	0.000000
25%	223.500000	20.125000	7.910400
50%	446.000000	28.000000	14.454200
75%	668.500000	38.000000	31.000000
max	891.000000	80.000000	512.329200

In [20]:
```
plt.rcParams['figure.figsize']=(10,8)

plt.subplot(1,3,1)
sns.histplot(df['PassengerId'],kde=True)

plt.subplot(1,3,2)
sns.histplot(df['Age'],kde=True)

plt.subplot(1,3,3)
sns.histplot(df['Fare'],kde=True)

plt.suptitle('Univariate Analysis on Numerical coluumns')
plt.show()
```

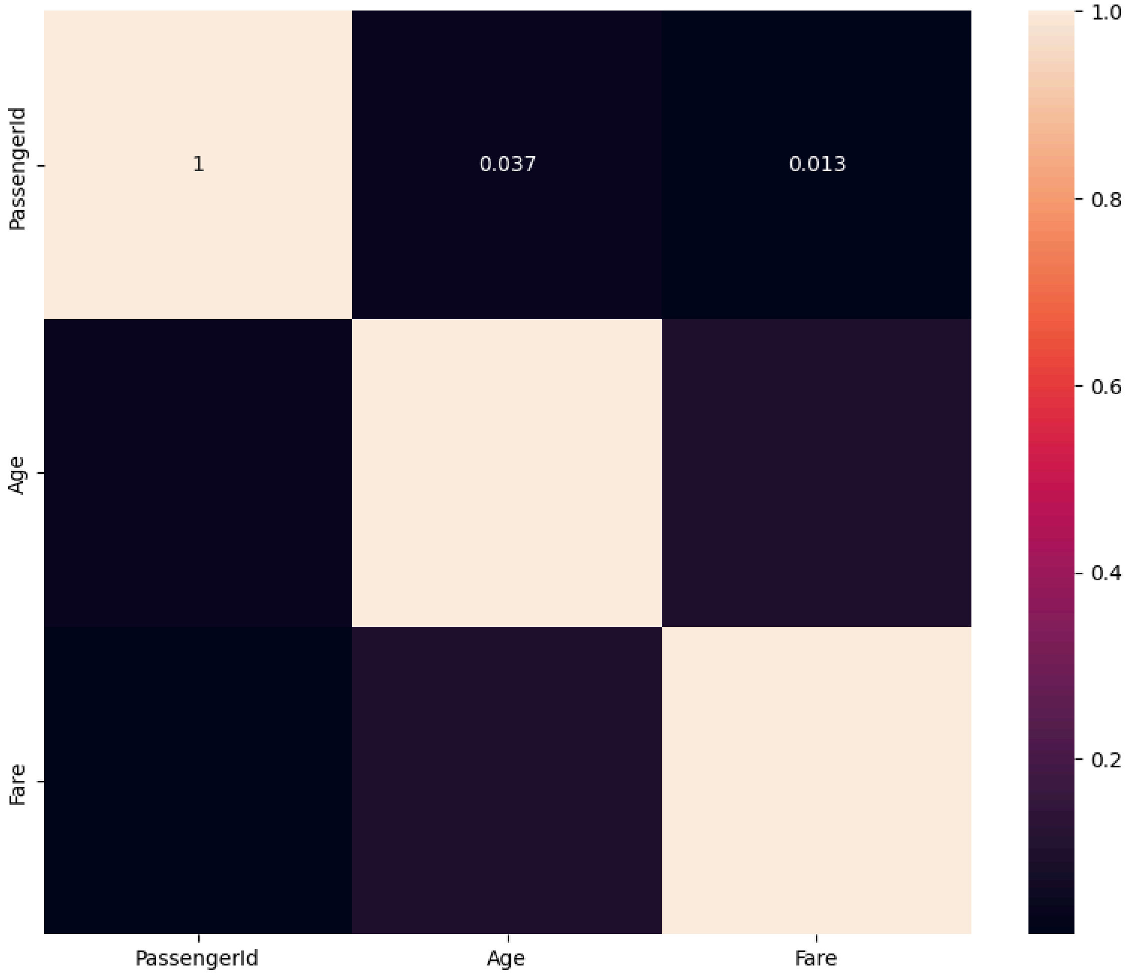


In [21]:
```
df[continous].corr()
```

Out[21]:

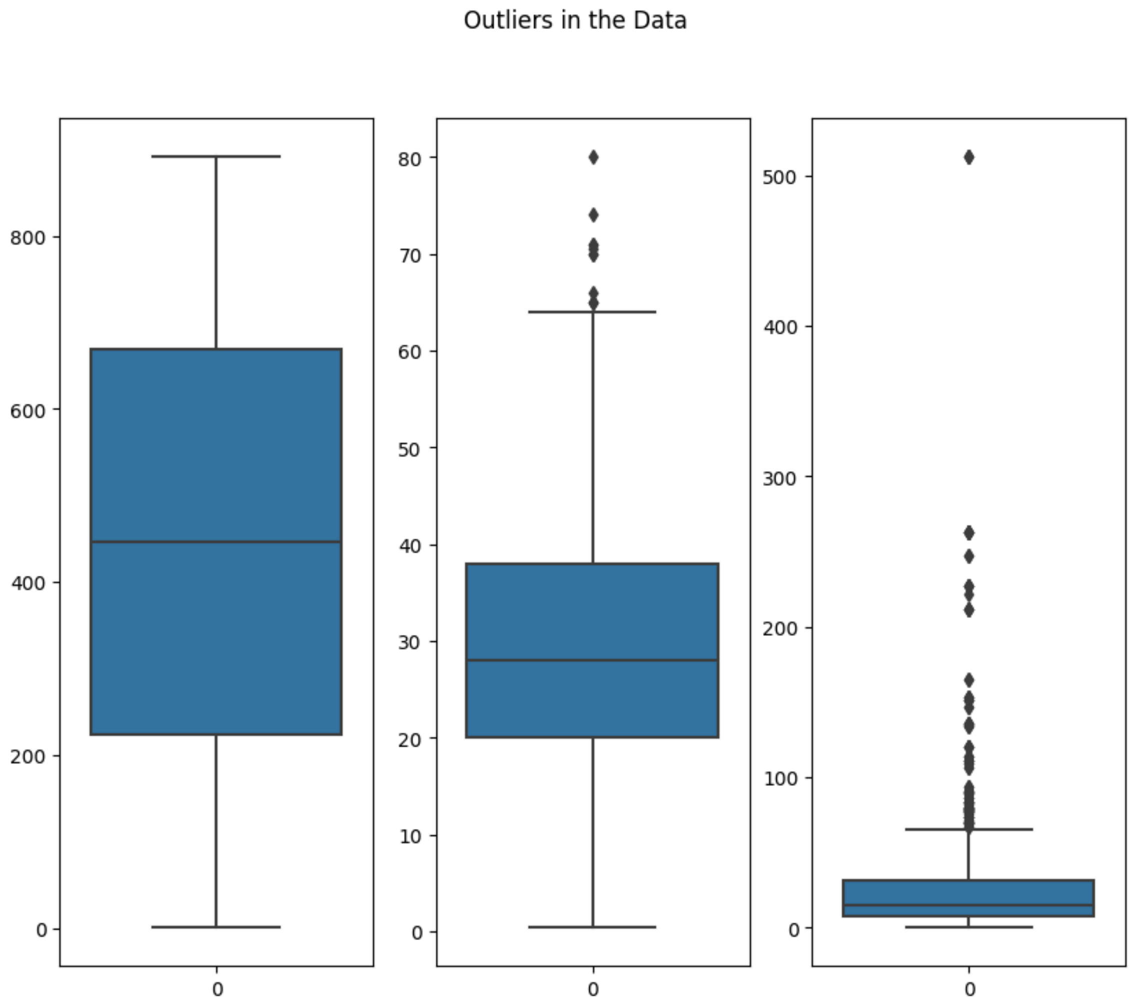
	PassengerId	Age	Fare
PassengerId	1.000000	0.036847	0.012658
Age	0.036847	1.000000	0.096067
Fare	0.012658	0.096067	1.000000

In [22]:
```
sns.heatmap(df[continous].corr(),annot=True)
plt.show()
```



In [23]:

```
plt.subplot(1,3,1)
sns.boxplot(df['PassengerId'])

plt.subplot(1,3,2)
sns.boxplot(df['Age'])

plt.subplot(1,3,3)
sns.boxplot(df['Fare'])

plt.suptitle('Outliers in the Data')
plt.show()
```



In [24]:

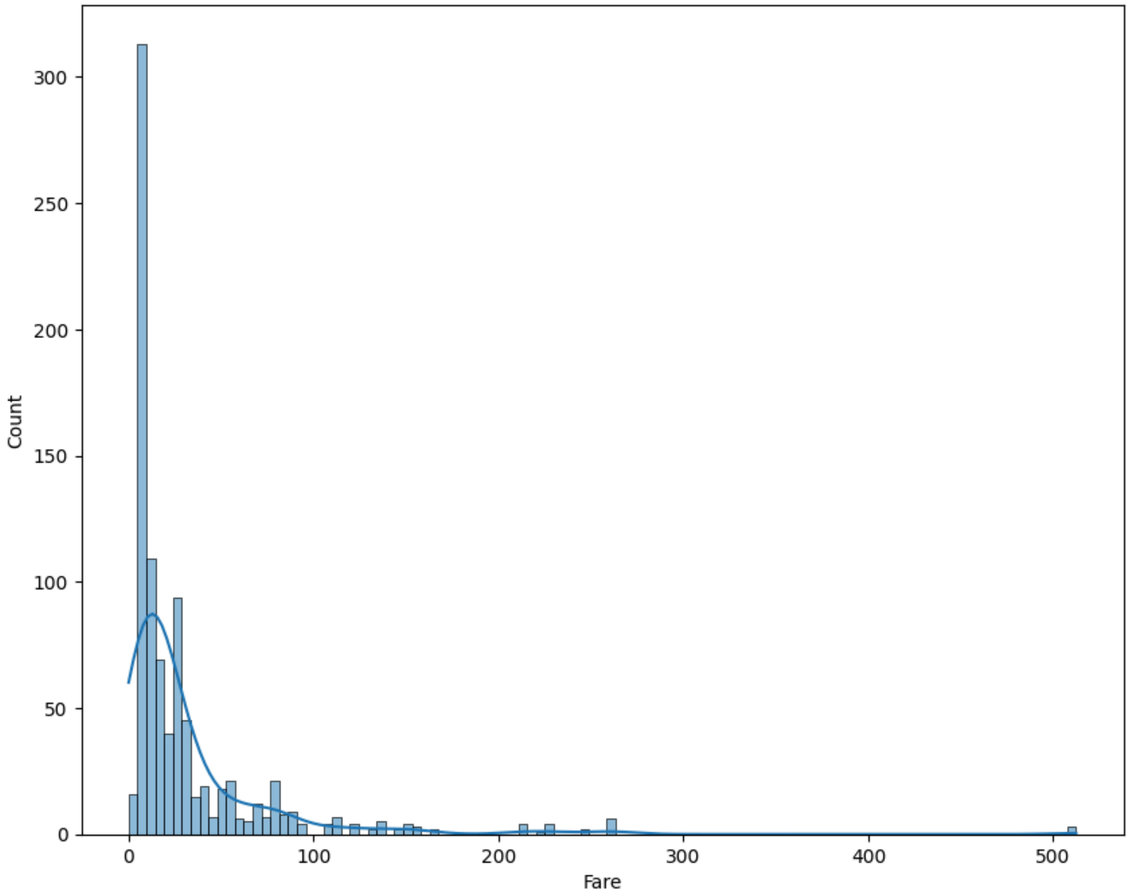
```
df[continous].skew()
```

Out[24]:

```
PassengerId    0.000000
Age            0.389108
Fare           4.787317
dtype: float64
```

In [25]:

```
sns.histplot(df['Fare'],kde=True)
plt.show()
```



In [26]:

```
df[discrete_categorical].describe()
```

Out[26]:

	Name	Sex	Ticket	Cabin	Embarked
count	891	891	891	204	889
unique	891	2	681	147	3
top	Braund, Mr. Owen Harris	male	347082	B96 B98	S
freq	1	577	7	4	644

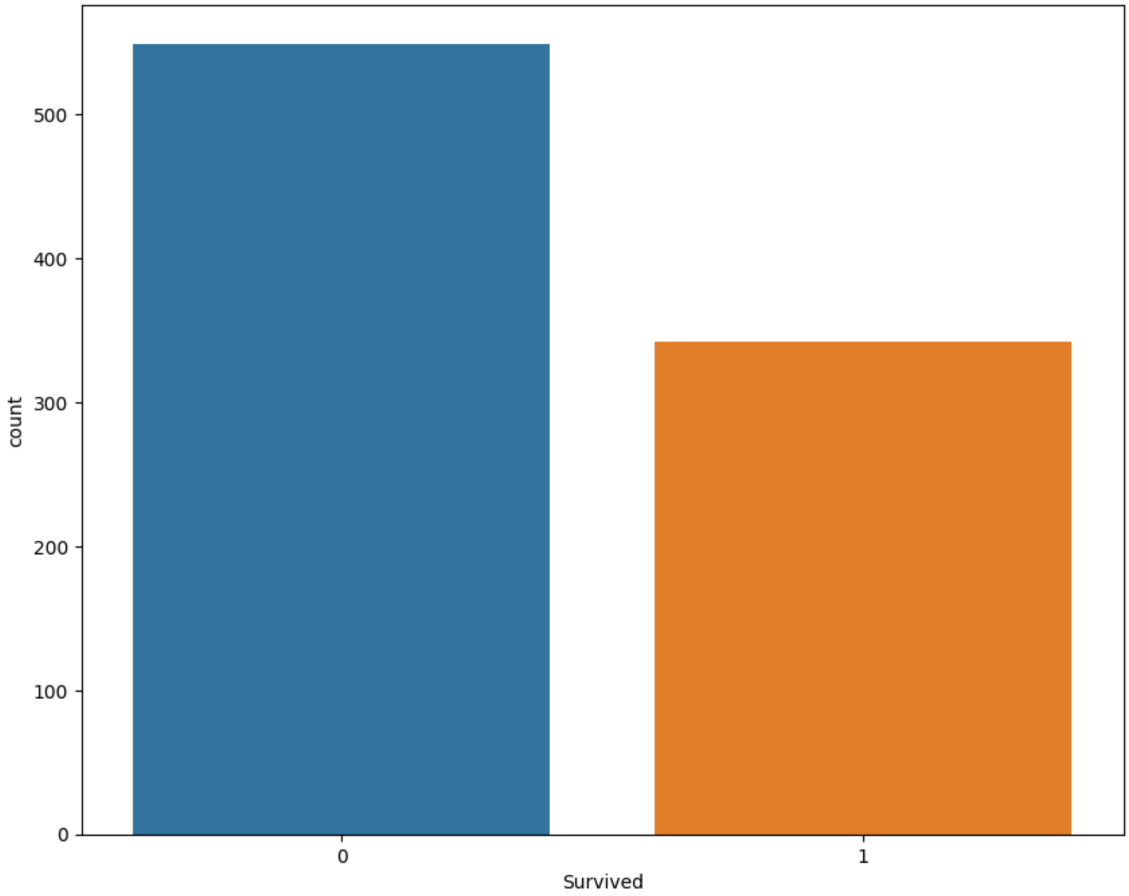
In [27]:

```
df[discrete_count].describe()
```

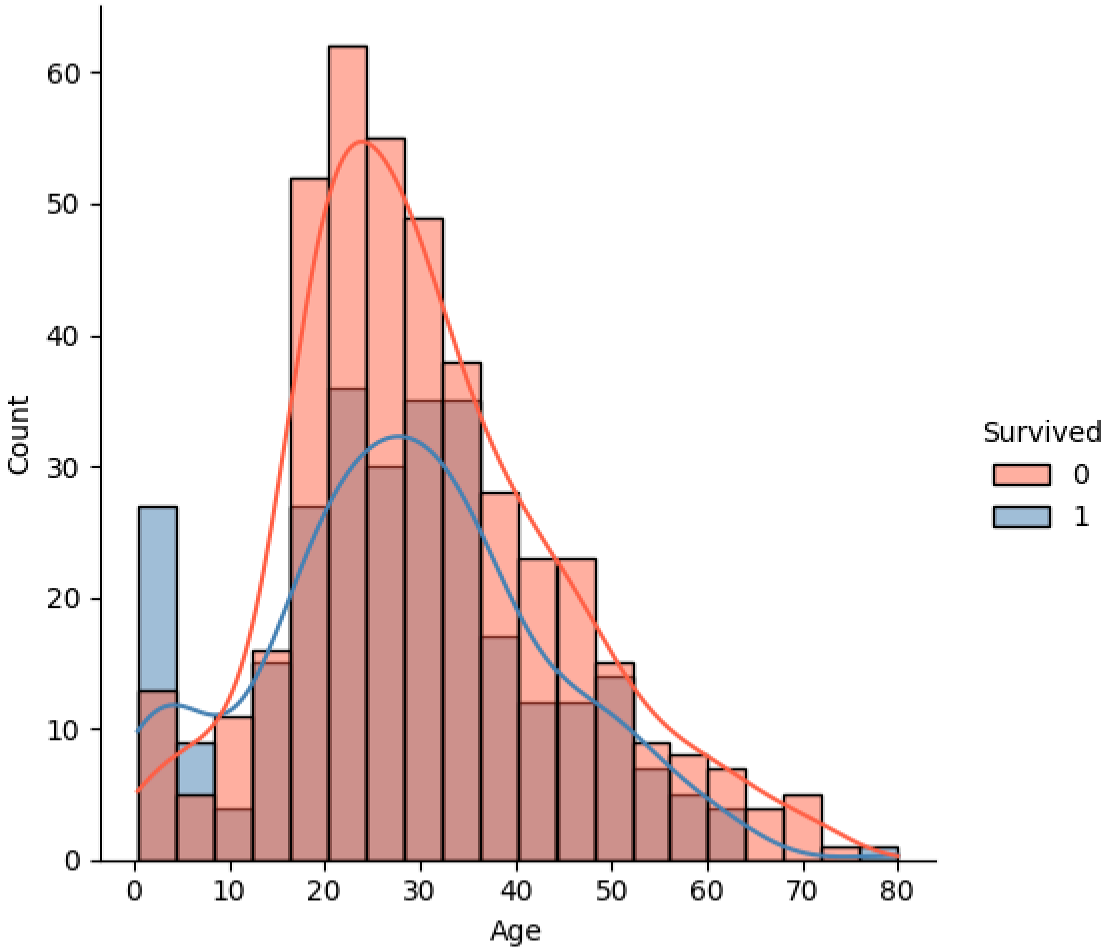
Out[27]:

	Survived	Pclass	SibSp	Parch
count	891.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	0.523008	0.381594
std	0.486592	0.836071	1.102743	0.806057
min	0.000000	1.000000	0.000000	0.000000
25%	0.000000	2.000000	0.000000	0.000000
50%	0.000000	3.000000	0.000000	0.000000
75%	1.000000	3.000000	1.000000	0.000000
max	1.000000	3.000000	8.000000	6.000000

In [28]:
```
sns.countplot(x=df['Survived'])
plt.show()
```

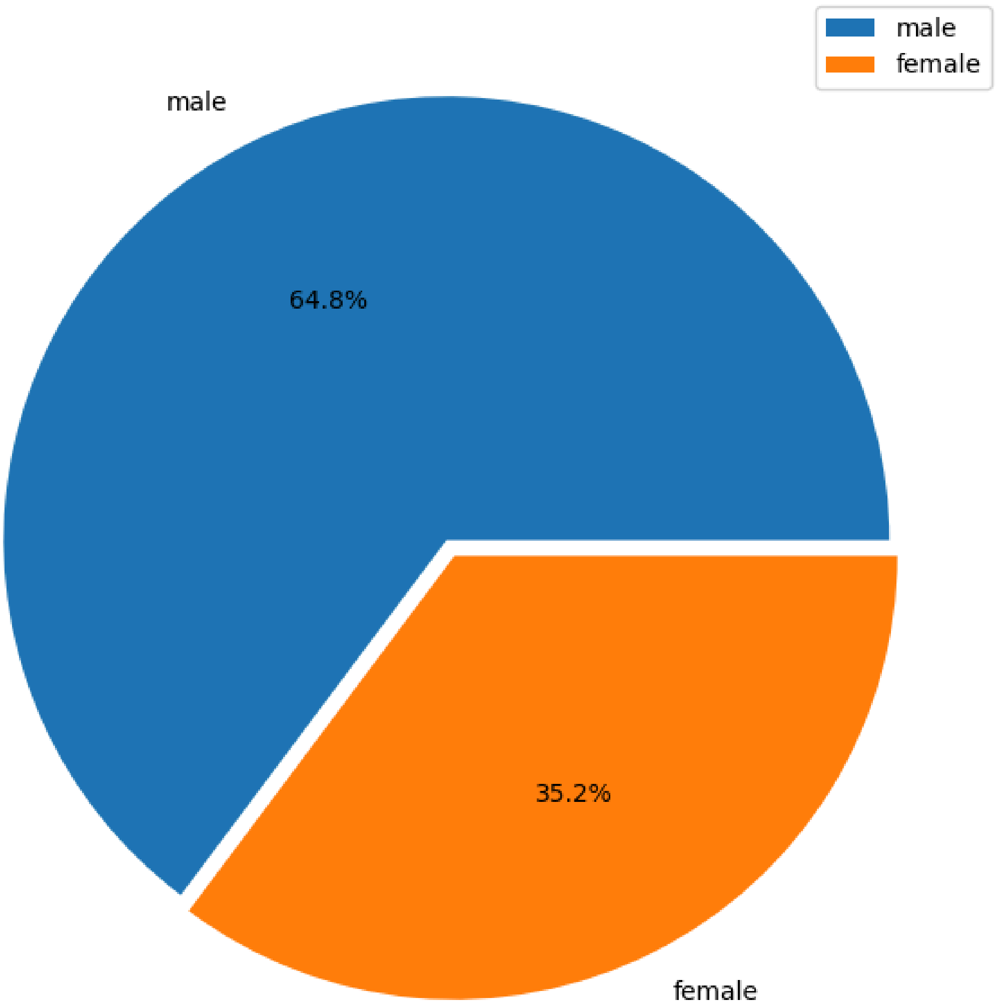


In [29]:
```
custom_palette = ['#FF6347', '#4682B4', '#3CB371']
sns.displot(x='Age', hue='Survived', data=df,kde=True,palette=custom_palett
plt.show()
```

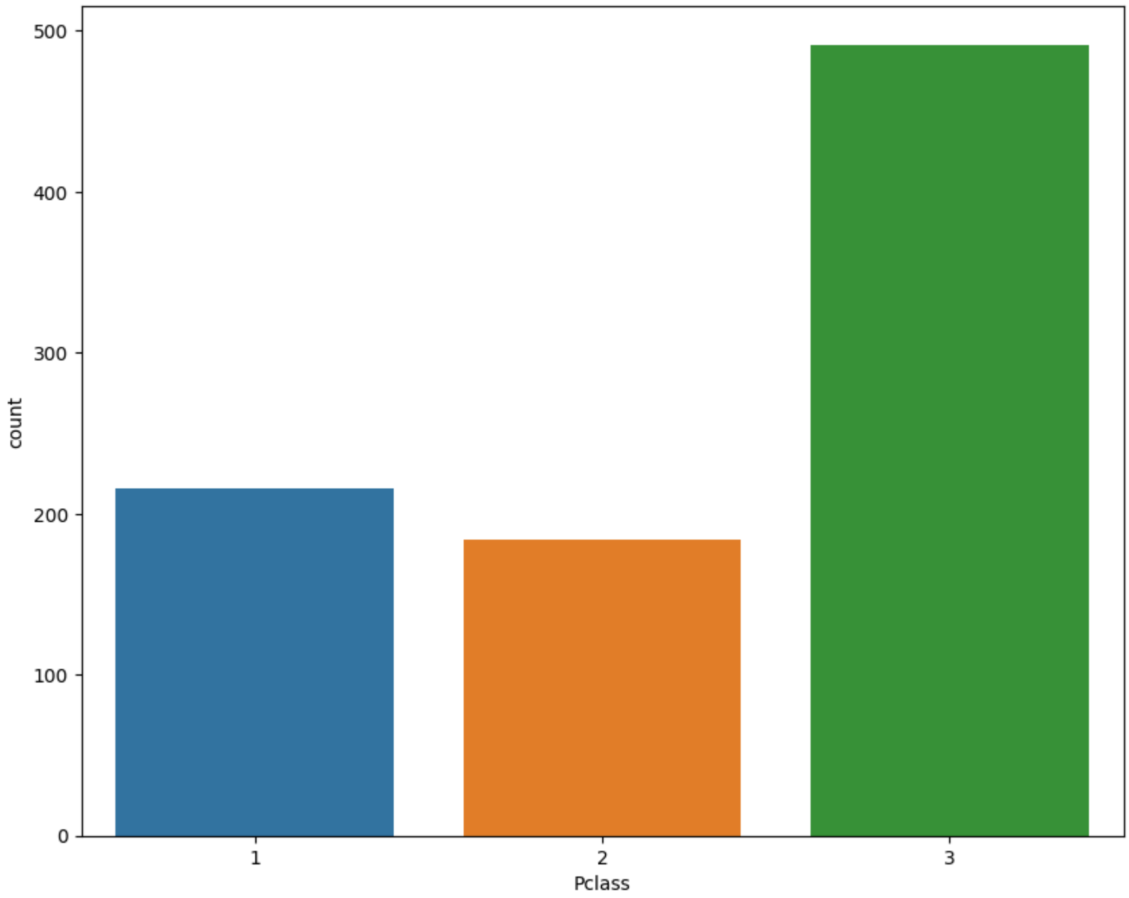


In [30]:

```
import matplotlib.pyplot as plt

# Assuming df is your DataFrame
sex_counts = df['Sex'].value_counts()
sex_labels = df['Sex'].unique()

# Create the explode list with the same length as the number of unique valu
explode = [0.02] * len(sex_labels)

plt.pie(x=sex_counts, labels=sex_labels, autopct='%0.1f%%', explode=explode
plt.legend()
plt.show()
```

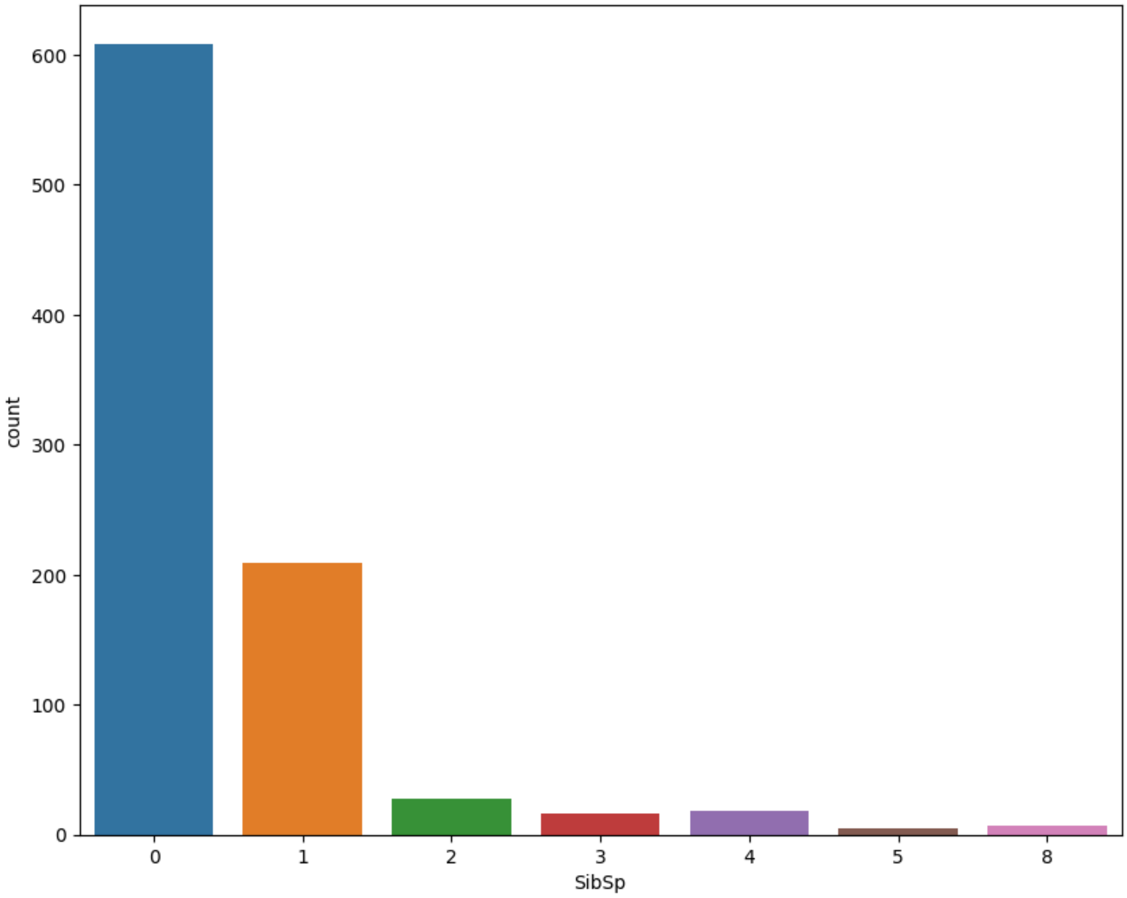


In [31]:
```
sns.countplot(x=df['Pclass'])
plt.show()
```

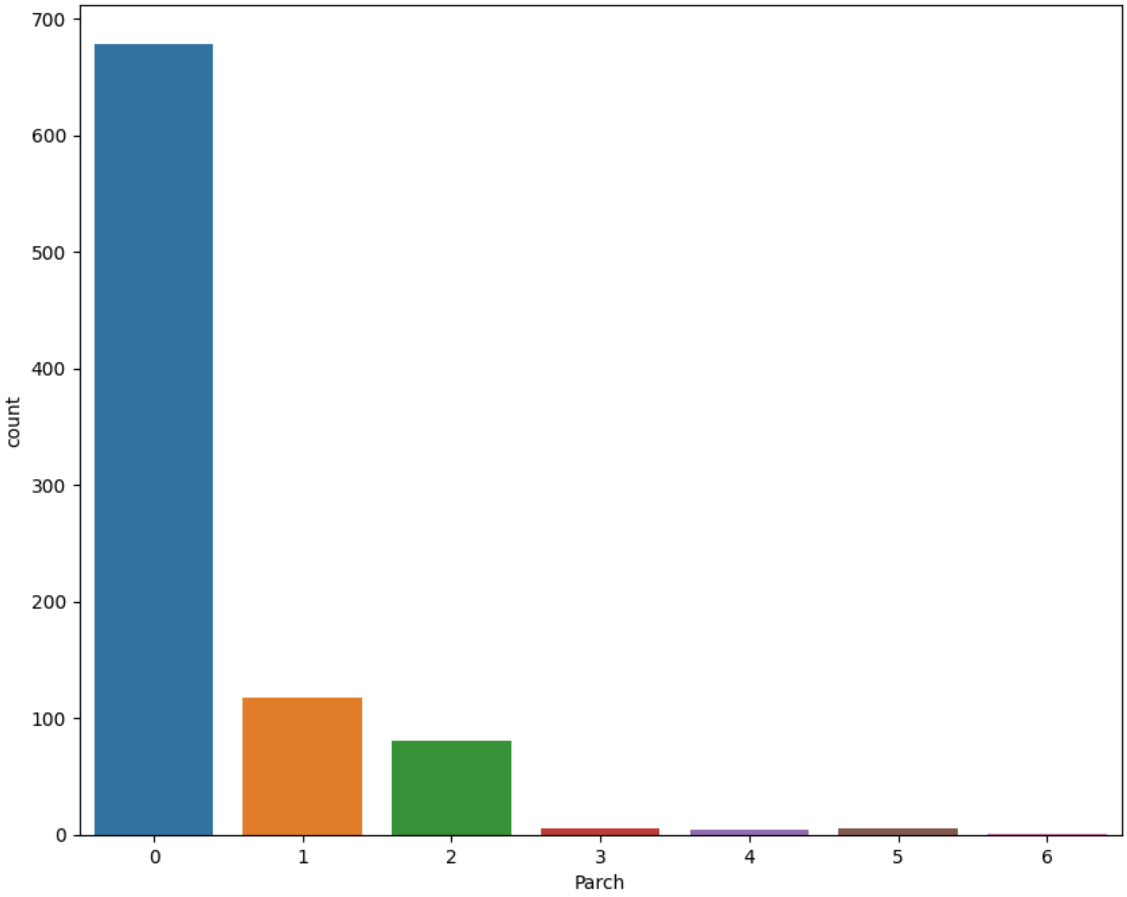


In [32]:

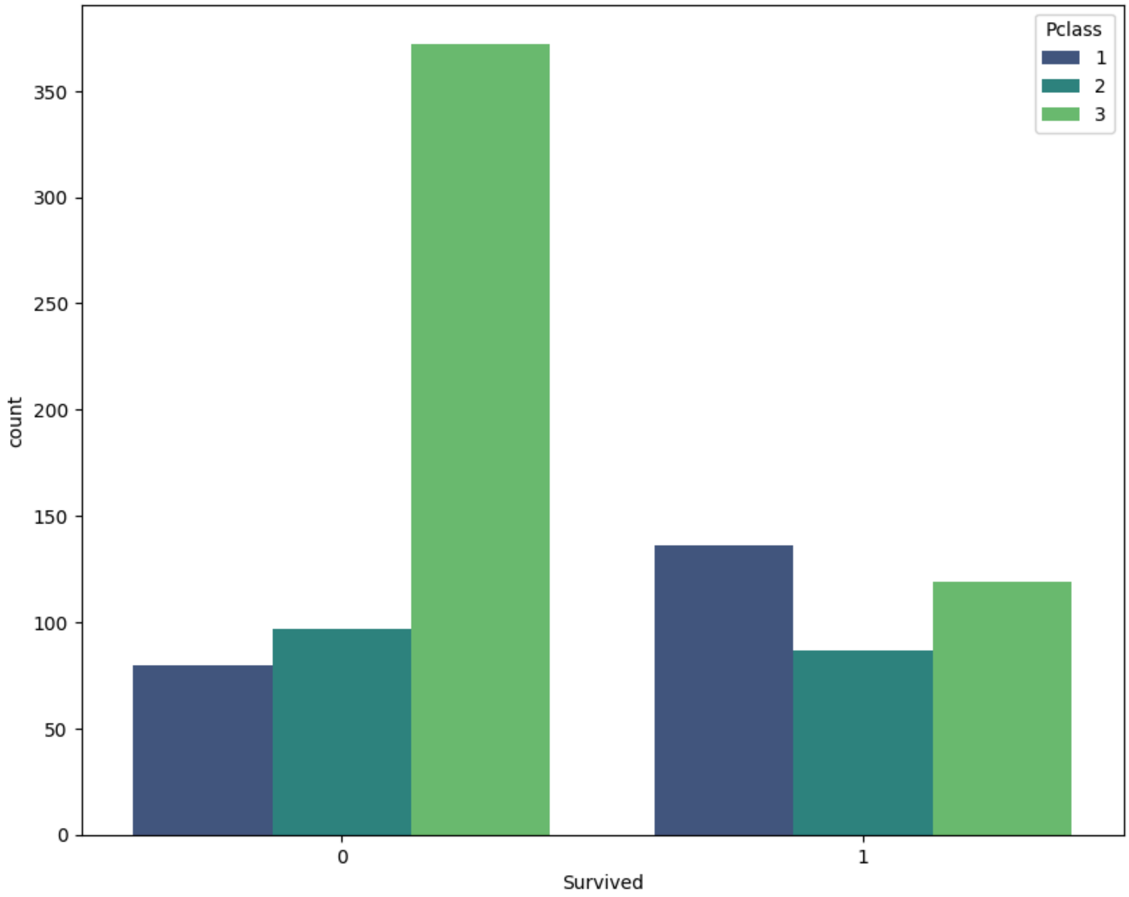
```
sns.countplot(x=df['SibSp'])
plt.show()
```



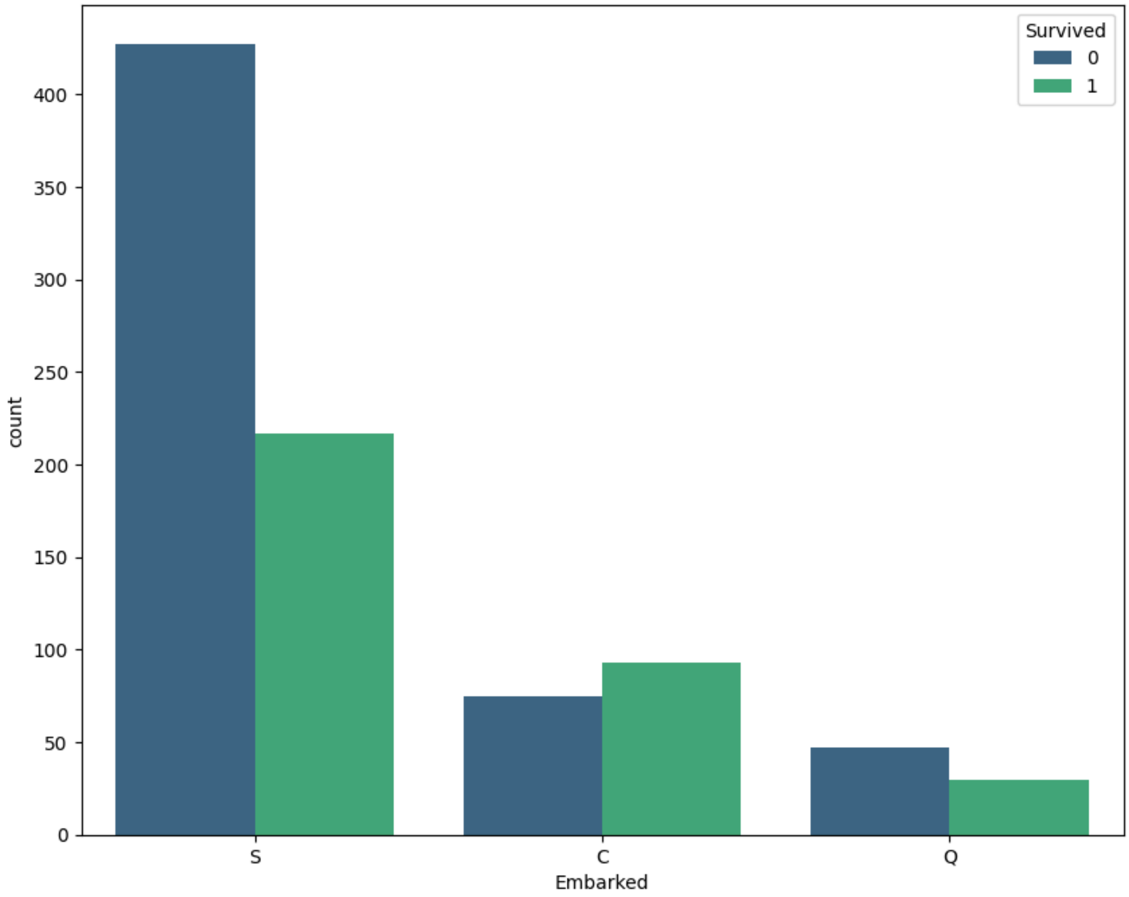
In [33]:
```
sns.countplot(x=df['Parch'])
plt.show()
```



```
In [34]: sns.countplot(x='Survived', hue='Pclass', data=df,palette='viridis')
         plt.show()
```



In [35]:
```
sns.countplot(x='Embarked', hue='Survived', data=df,palette='viridis')
plt.show()
```



Data prepration

Missing Value Treatments on Train data

In [36]:
```
#checking no.of missing values
df.isnull().sum()
```

Out[36]:
```
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 [37]:

```
#checking percentage of missing values
df.isnull().sum()/len(df)*100
```

Out[37]:

```
PassengerId     0.000000
Survived        0.000000
Pclass          0.000000
Name            0.000000
Sex             0.000000
Age            19.865320
SibSp           0.000000
Parch           0.000000
Ticket          0.000000
Fare            0.000000
Cabin          77.104377
Embarked        0.224467
dtype: float64
```

In [38]:

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

In [39]:

```
df.head()
```

Out[39]:

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

In [40]:

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

Out[40]:

```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

In [41]:

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

In [42]:

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

In [43]:

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

Out[43]:

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

Missing Value Treatments on Test data

In [44]:

```
#checking no.of missing values
test.isnull().sum()
```

Out[44]:

```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

In [45]:

```
#checking percentage of missing values
test.isnull().sum()/len(df)*100
```

Out[45]:

```
PassengerId     0.000000
Pclass          0.000000
Name            0.000000
Sex             0.000000
Age             9.652076
SibSp           0.000000
Parch           0.000000
Ticket          0.000000
Fare            0.112233
Cabin          36.700337
Embarked        0.000000
dtype: float64
```

In [46]:

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

In [47]:

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

In [48]:
```
test['Age'].fillna(test['Age'].mean(),inplace=True)
```

In [49]:
```
test.isnull().sum()
```

Out[49]:
```
PassengerId    0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

In [50]:
```
test[continous].skew()
```

Out[50]:
```
PassengerId    0.000000
Age            0.512711
Fare           3.691600
dtype: float64
```

outlier treatment on Train data

In [51]:
```
# Calculate the first and third quartiles
Q1 = np.percentile(df['Fare'], 25)
Q3 = np.percentile(df['Fare'], 75)

# Calculate the interquartile range (IQR)
IQR = Q3 - Q1

# Define the lower and upper bounds for outliers
outlier_low = Q1 - 1.5 * IQR
outlier_high = Q3 + 1.5 * IQR

# Filter out the outliers
df = df[(df['Fare'] > outlier_low) & (df['Fare'] < outlier_high)]
```

In [52]:
```
df[continous].skew()
```

Out[52]:
```
PassengerId   -0.007285
Age            0.435012
Fare           1.430672
dtype: float64
```

In [53]:
```
import numpy as np
import pandas as pd
from scipy.stats import boxcox

# Assuming df is your DataFrame and continuous is the list of continuous co
continuous = ['PassengerId', 'Age', 'Fare']

# Apply log1p transformation to columns with skewness > 1
df[continuous] = df[continuous].apply(lambda x: np.log1p(x) if x.skew() > 1

# Apply sqrt transformation to columns with 0.5 < skewness <= 1
df[continuous] = df[continuous].apply(lambda x: np.sqrt(x) if 0.5 < x.skew(

# Apply Box-Cox transformation to remaining columns with positive values
df[continuous] = df[continuous].apply(lambda x: boxcox(x + 1)[0] if x.skew(
```

In [54]:
```
df[continous].skew()
```

Out[54]:
```
PassengerId   -0.291521
Age            0.070739
Fare           0.100643
dtype: float64
```

outlier treatment on Test data

In [55]:
```
# Calculate the first and third quartiles
Q1 = np.percentile(test['Fare'], 25)
Q3 = np.percentile(test['Fare'], 75)

# Calculate the interquartile range (IQR)
IQR = Q3 - Q1

# Define the lower and upper bounds for outliers
outlier_low = Q1 - 1.5 * IQR
outlier_high = Q3 + 1.5 * IQR

# Filter out the outliers
test = test[(test['Fare'] > outlier_low) & (test['Fare'] < outlier_high)]
```

In [56]:
```
test[continous].skew()
```

Out[56]:
```
PassengerId    0.026021
Age            0.316123
Fare           1.578135
dtype: float64
```

In [57]:
```
import numpy as np
import pandas as pd
from scipy.stats import boxcox

# Assuming test is your DataFrame and continuous is the list of continuous
continuous = ['PassengerId','Age','Fare']

# Apply log1p transformation to columns with skewness > 1
test[continuous] = test[continuous].apply(lambda x: np.log1p(x) if x.skew()

# Apply sqrt transformation to columns with 0.5 < skewness <= 1
test[continuous] = test[continuous].apply(lambda x: np.sqrt(x) if 0.5 < x.s

# Apply Box-Cox transformation to remaining columns with positive values
test[continuous] = test[continuous].apply(lambda x: boxcox(x + 1)[0] if x.s
```

In [58]:
```
test[continous].skew()
```

Out[58]:
```
PassengerId   -0.026821
Age            0.102480
Fare           0.102564
dtype: float64
```

In [59]:
```
df.head()
```

Out[59]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	0.894223	0	3	Braund, Mr. Owen Harris	male	15.099941	1	0	A/5 21171	3.0
2	2.353145	1	3	Heikkinen, Miss. Laina	female	17.424536	0	0	STON/O2. 3101282	3.1
3	2.996462	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	22.457233	1	0	113803	6.9
4	3.602789	0	3	Allen, Mr. William Henry	male	22.457233	0	0	373450	3.2
5	4.179988	0	3	Moran, Mr. James	male	19.522803	0	0	330877	3.2

Encoding

In [60]:
```
df['Sex'].replace({'female':0,'male':1},inplace=True)
test['Sex'].replace({'female':0,'male':1},inplace=True)
```

In [61]:
```
df['Sex'].value_counts()
```

Out[61]:
```
1    531
0    244
Name: Sex, dtype: int64
```

In [62]:
```
test['Sex'].value_counts()
```

Out[62]:
```
1    240
0    123
Name: Sex, dtype: int64
```

In [63]:
```
df.drop(columns=['Name','Ticket'],inplace=True)
```

In [64]:
```
test.drop(columns=['Name','Ticket'],inplace=True)
```

Dummy Encoding

In [71]:
```
dum = pd.get_dummies(df['Embarked'],drop_first=True)
df = pd.concat([df,dum],axis='columns')
df
```

Out[71]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Q
0	0.894223	0	3	1	15.099941	1	0	3.047679	S	0
2	2.353145	1	3	0	17.424536	0	0	3.191541	S	0
3	2.996462	1	1	0	22.457233	1	0	6.966224	S	0
4	3.602789	0	3	1	22.457233	0	0	3.217180	S	0
5	4.179988	0	3	1	19.522803	0	0	3.298914	Q	1
...	...	...	...	...	...	...	...	...	...	...
886	169.677207	0	2	1	17.996302	0	0	4.050646	S	0
887	169.813249	1	1	0	13.311662	0	0	5.707060	S	0
888	169.949246	0	3	0	19.522803	1	2	5.194585	S	0
889	170.085199	1	1	1	17.424536	0	0	5.707060	C	0
890	170.221106	0	3	1	20.806226	0	0	3.155138	Q	1

775 rows × 11 columns

In [72]:

```
dum = pd.get_dummies(test['Embarked'],drop_first=True)
test = pd.concat([test,dum],axis='columns')
test
```

Out[72]:

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Q	S
0	98.698430	3	1	26.345771	0	0	1.974408	Q	1	0
1	98.766304	3	0	34.888183	1	0	1.890767	S	0	1
2	98.834148	2	1	44.844549	0	0	2.135350	Q	1	0
3	98.901963	3	1	21.075824	0	0	2.050568	S	0	1
4	98.969747	3	0	17.483278	1	1	2.317223	S	0	1
...	...	...	...	...	...	...	...	...	...	...
412	124.546897	3	0	21.786093	0	0	1.969197	S	0	1
413	124.605310	3	1	23.391101	0	0	1.995297	S	0	1
415	124.722083	3	1	29.108457	0	0	1.916905	S	0	1
416	124.780443	3	1	23.391101	0	0	1.995297	S	0	1
417	124.838786	3	1	23.391101	1	1	2.781269	C	0	0

363 rows × 10 columns

In [79]:

```
test.drop(columns=['Embarked'],inplace=True)
```

In [80]:

```
test.head()
```

Out[80]:

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Q	S
0	98.698430	3	1	26.345771	0	0	1.974408	1	0
1	98.766304	3	0	34.888183	1	0	1.890767	0	1
2	98.834148	2	1	44.844549	0	0	2.135350	1	0
3	98.901963	3	1	21.075824	0	0	2.050568	0	1
4	98.969747	3	0	17.483278	1	1	2.317223	0	1

In [82]:

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

In [83]:

```
df.head()
```

Out[83]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Q	S
0	0.894223	0	3	1	15.099941	1	0	3.047679	0	1
2	2.353145	1	3	0	17.424536	0	0	3.191541	0	1
3	2.996462	1	1	0	22.457233	1	0	6.966224	0	1
4	3.602789	0	3	1	22.457233	0	0	3.217180	0	1
5	4.179988	0	3	1	19.522803	0	0	3.298914	1	0

X&y