Exploratory Data Analysis - EDA

Titanic Dataset - Exploratory Data Analysis (EDA) and Data Cleaning

This notebook presents a thorough **data cleaning** and **exploratory data analysis (EDA)** of the Titanic dataset. The objective is to clean the dataset and uncover insights regarding the survival rates of passengers based on various factors.

Key Steps:

- **Data Cleaning**:
  - Address missing values in `Age` and `Embarked` columns
  - Remove the `Cabin` column due to excessive null values
  - Ensure data consistency and remove any duplicates
- **Exploratory Data Analysis (EDA)**:
  - Visualize survival distribution across different features such as **Gender**, **Passenger Class**, **Age Group**, and **Embarkation Point**
  - Perform correlation analysis between key variables

Insights:

- Understand the relationship between socio-economic factors and survival rates
- Identify patterns related to **gender**, **class**, **fare**, and **embarkation point** in determining survival likelihood

Libraries Used:

- `pandas`, `numpy`, `seaborn`, `matplotlib`

Importing the necessary libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Reading the Dataset

In [2]:
```
df = pd.read_csv("Titanic-Dataset.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Tic
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	21
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/ 3101
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373

In [6]:
```
df.shape
```

Out[6]:
```
(891, 12)
```

In [7]:
```
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 [8]:
```python
df.describe()
```

Out[8]:

	PassengerId	Survived	Pclass	Age	SibSp	Parc
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.00000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.38159
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.80605
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.00000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.00000
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.00000
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.00000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.00000

Checking for null/missing values in the dataset

- Checking for null or missing values is important to ensure data quality, prevent inaccuracies in analysis or models, and maintain data integrity.
- Handling missing values ***(e.g., through imputation or removal)*** is crucial for accurate insights and effective model performance.

In [9]:
```python
df.isnull().sum()
```

Out[9]:
```
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
```

`Observation & Inference:`

> There are 177 null values in the age column, 687 in Cabin, 2 in Embarked.

> Since the cabin column is not of much use and contains a lot of null values so we will drop it.

Dropping unnecessary columns

In [10]:
```python
df.drop(columns="Cabin",axis=1,inplace=True)
```

Imputing Missing Age Values with Column Mean

In [13]:
```python
df['Age'] = df['Age'].fillna(df['Age'].mean())
```

In [14]:
```python
df.fillna({'Age': df['Age'].mean()}, inplace=True)
```

Handled missing values in the 'Embarked' column by replacing them with its mode.

In [16]:
```python
df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])
```

In [17]:
```python
df.isnull().sum().sum()
```

Out[17]:
```
0
```

`Observation & Inference: - All the missing values are treated`

Checking for duplicate values in the dataset

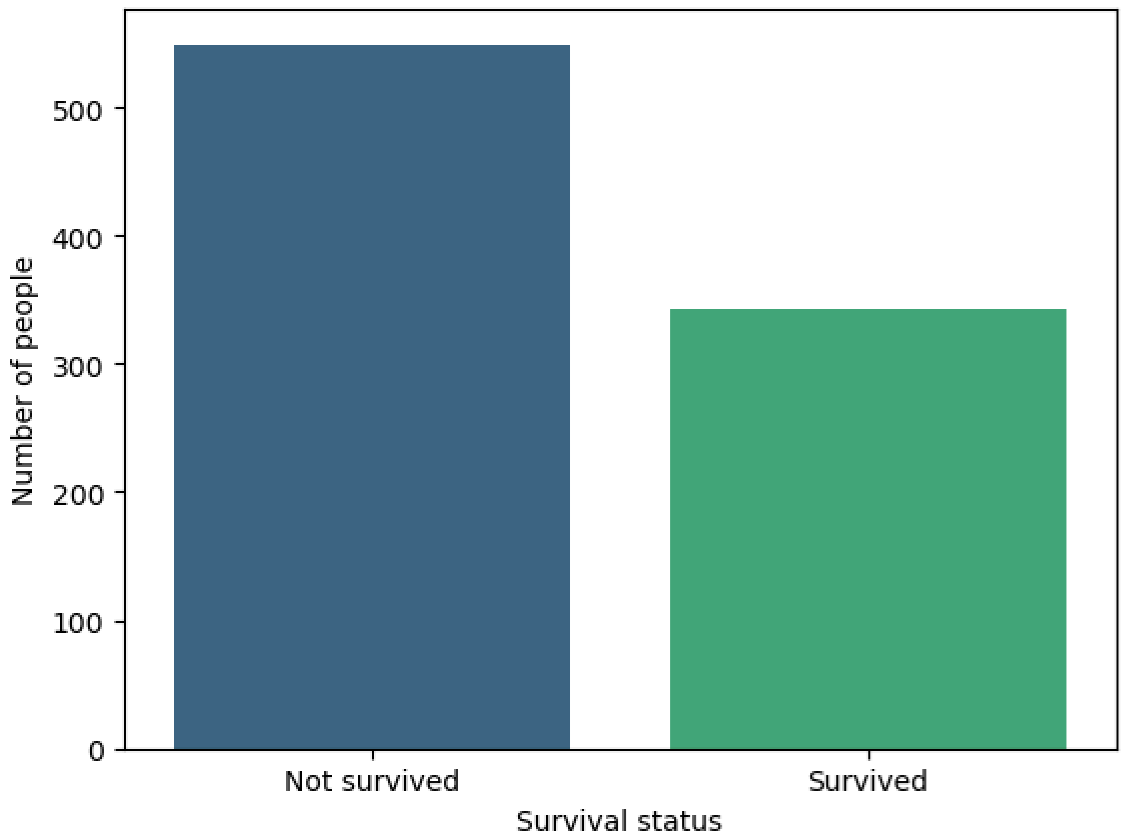
In [18]:
```python
df.duplicated().sum()
```

Out[18]:
```
0
```

` Observation & Inference: - No duplicate records are present`

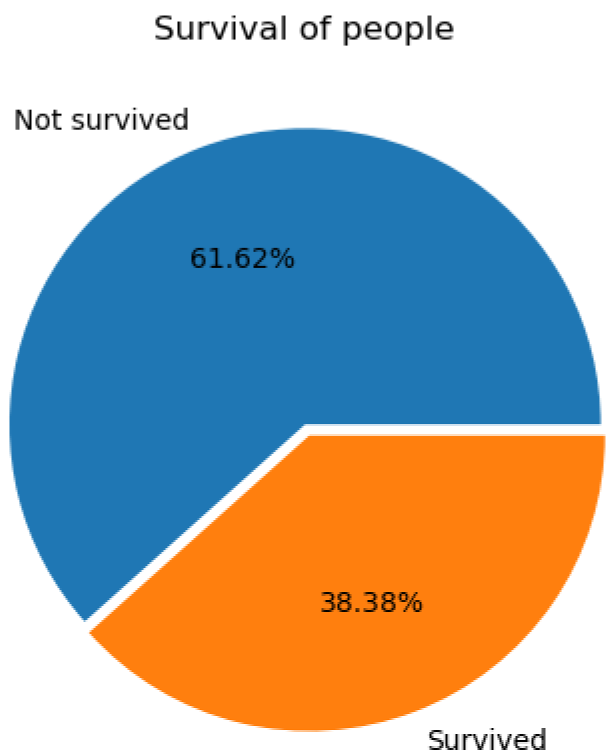
Checking the survival of people

In [20]:
```python
sns.countplot(x='Survived', hue='Survived', data=df, palette='viridis', lege
plt.xlabel("Survival status")
plt.ylabel("Number of people")
plt.xticks(ticks=[0,1], labels=['Not survived','Survived'])
plt.show()
```



Pie chart

In [21]:
```
plt.pie(df['Survived'].value_counts(),explode=[0,0.04],autopct="%1.2f%%",lab
plt.title("Survival of people")
plt.show()
```

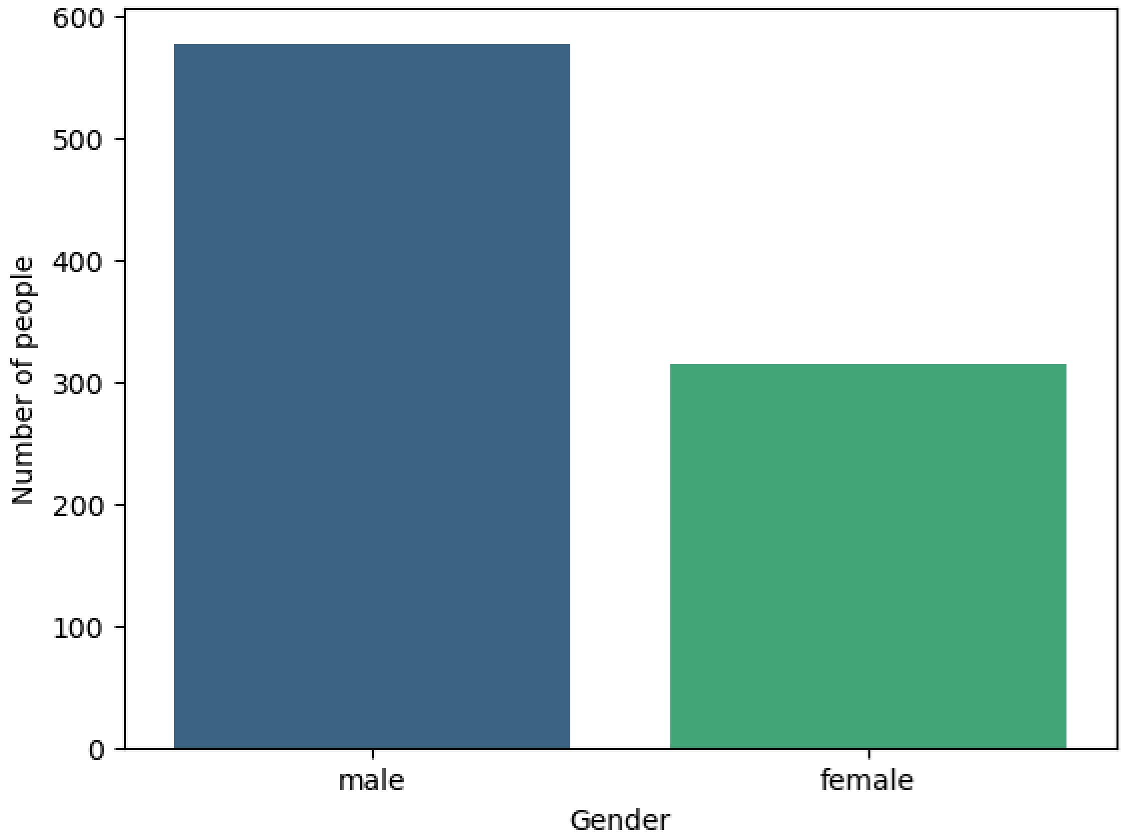


Visualization of people survived from different gender

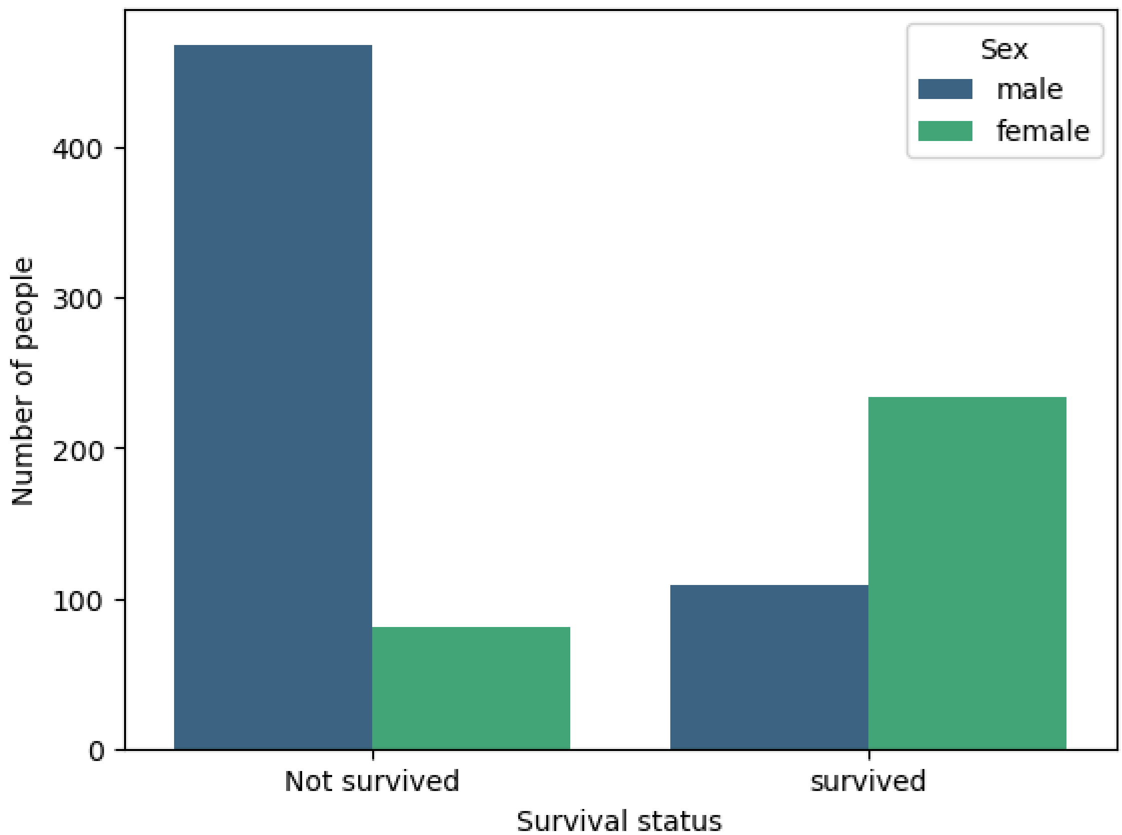
In [22]:
```
df['Sex'].unique()
```

Out[22]:
```
array(['male', 'female'], dtype=object)
```

In [35]:
```
sns.countplot(x='Sex', hue='Sex', data=df, palette='viridis', legend=False)
plt.xlabel("Gender")
plt.ylabel("Number of people")
plt.show()
```

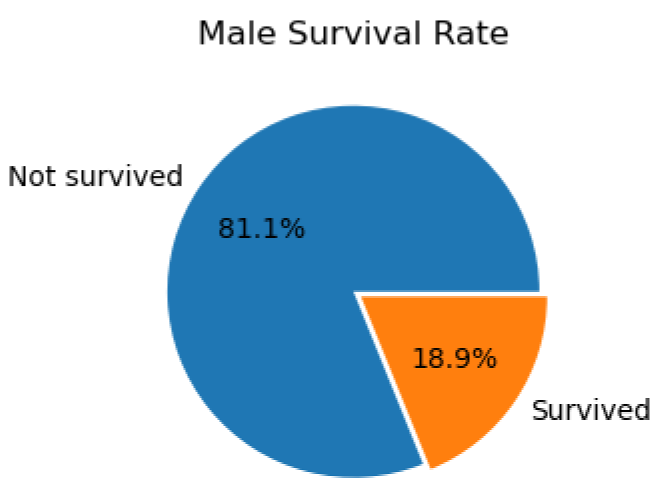


In [34]:
```
sns.countplot(x='Survived',hue='Sex',data=df,palette='viridis',)
plt.xlabel("Survival status")
plt.ylabel("Number of people")
plt.xticks(ticks=[0,1],labels=['Not survived','survived'])
plt.show()
```



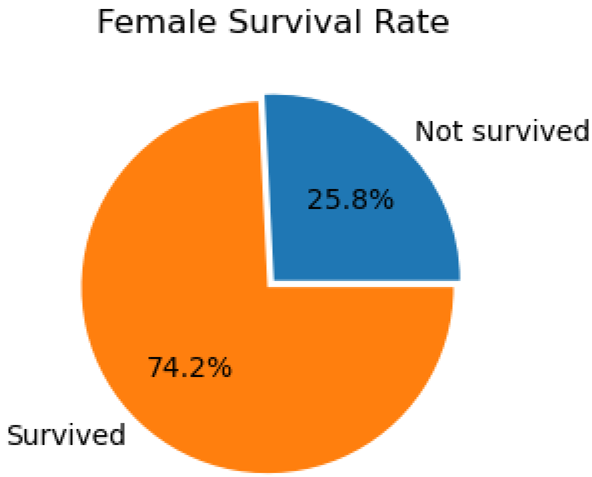
Pie Chart For Male Survival Rate

In [27]:
```
df[df['Sex'] == 'male'].Survived.groupby(df.Survived).count().plot(kind='pie
figsize=(3, 6),explode=[0,0.05],autopct='%1.1f%%',labels=["Not survived","Su
plt.ylabel("")
plt.title("Male Survival Rate")
plt.show()
```



Pie Chart For Female Survival Rate

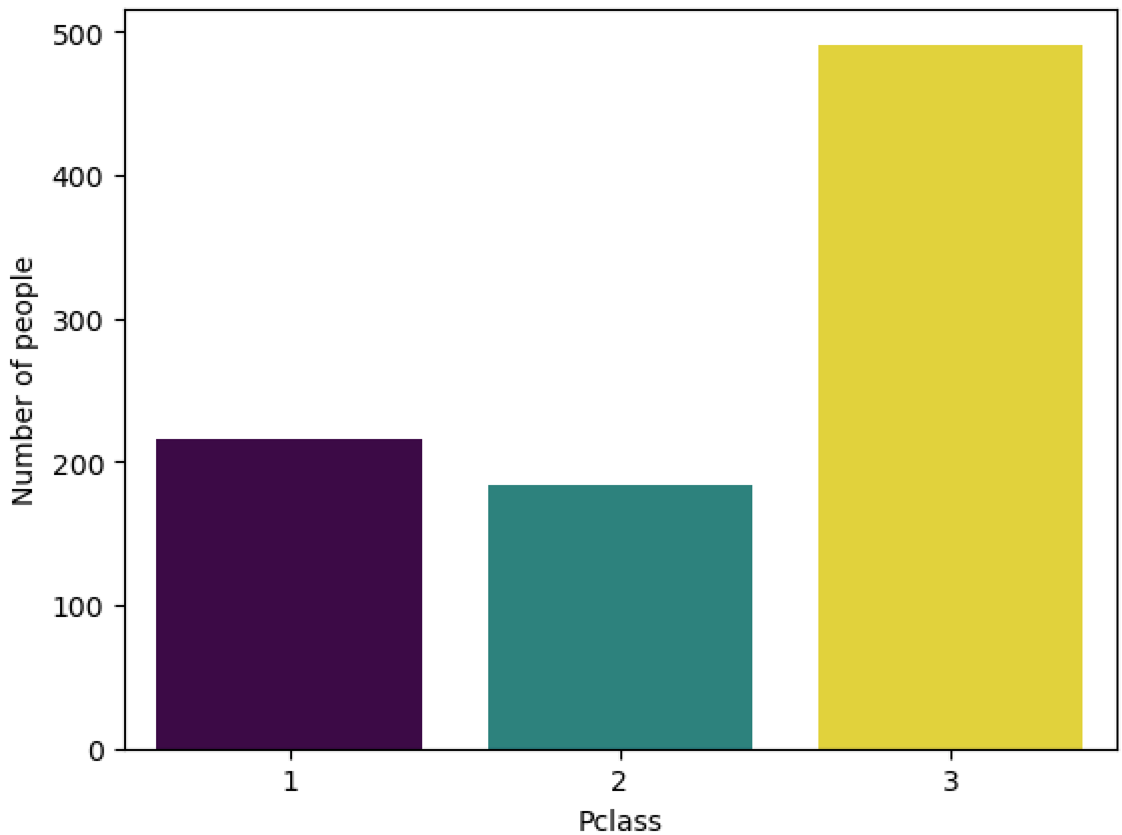
In [28]:
```
df[df['Sex'] == 'female'].Survived.groupby(df.Survived).count().plot(kind='p
figsize=(3, 6),explode=[0,0.05],autopct='%1.1f%%',labels=["Not survived","Su
plt.ylabel("")
plt.title("Female Survival Rate")
plt.show()
```



`Observation: Survival rate was female was much higher in comparison to male`

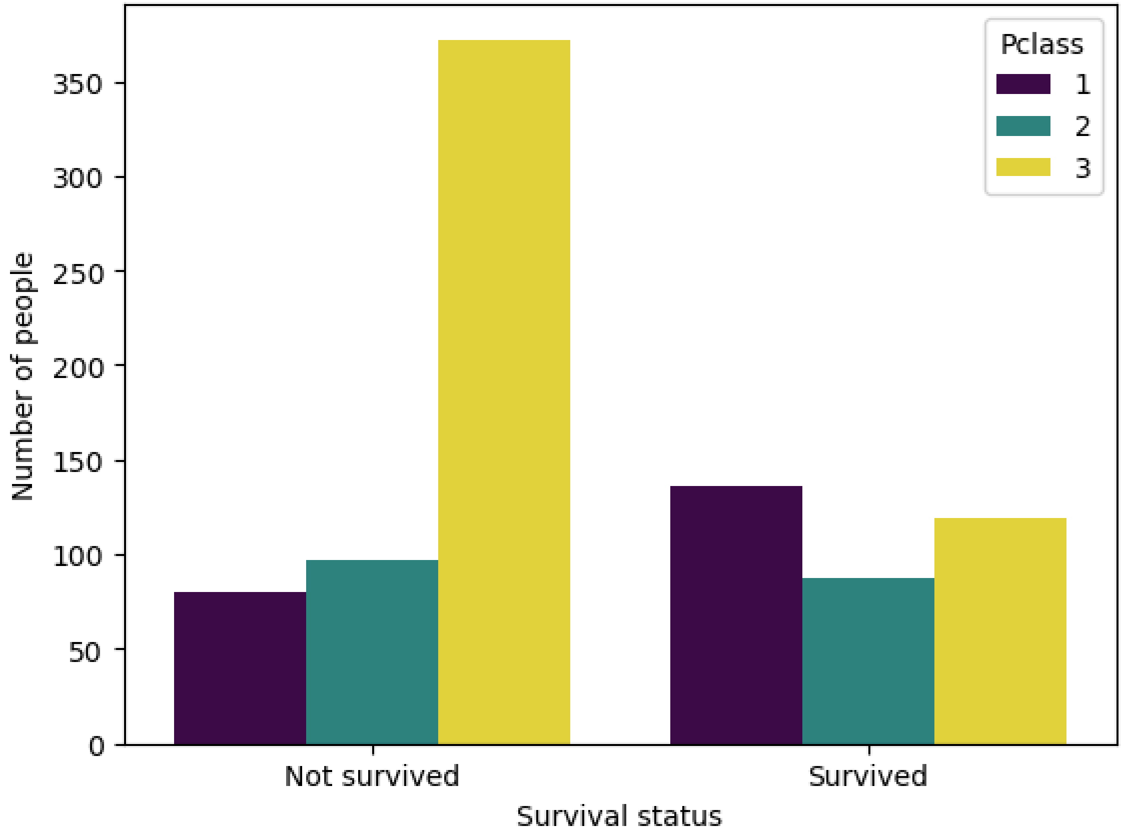
Visualizing the population of different passenger : Class

In [30]:
```
sns.countplot(x='Pclass', hue='Pclass', data=df, palette='viridis', legend=F
plt.xlabel("Pclass")
plt.ylabel("Number of people")
plt.show()
```

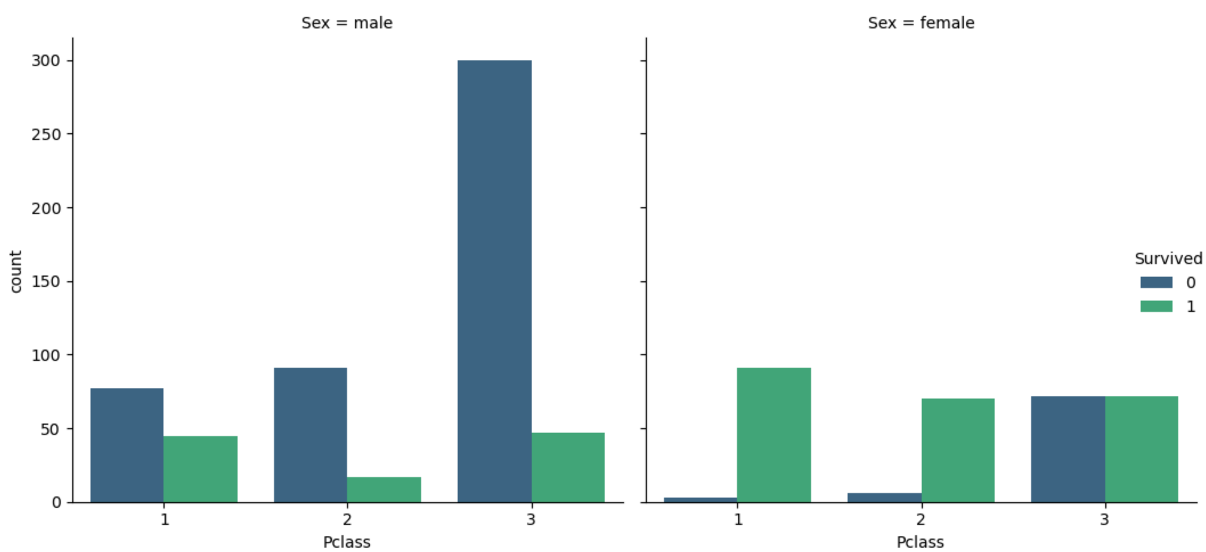


Visualization of people survived from different passenger class

In [32]:
```
sns.countplot(x='Survived', hue='Pclass', data=df, palette='viridis')
plt.xlabel("Survival status")
plt.ylabel("Number of people")
plt.xticks(ticks=[0, 1], labels=['Not survived', 'Survived'])
plt.show()
```



In [36]:
```
sns.catplot(x = 'Pclass', hue = 'Survived', col = 'Sex', kind = 'count', dat
df,palette='viridis' )
plt.tight_layout()
```

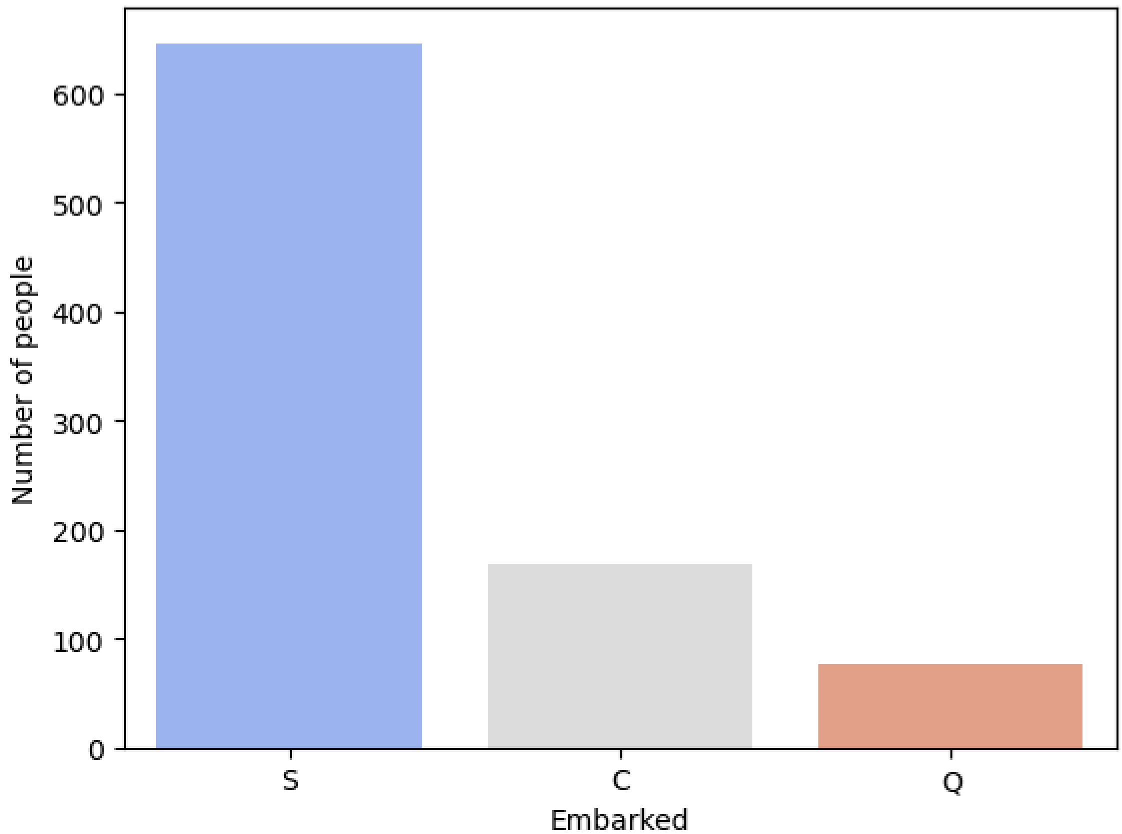


Observation: - Though population of P class 3 was the highest, yet they had the least survival rate

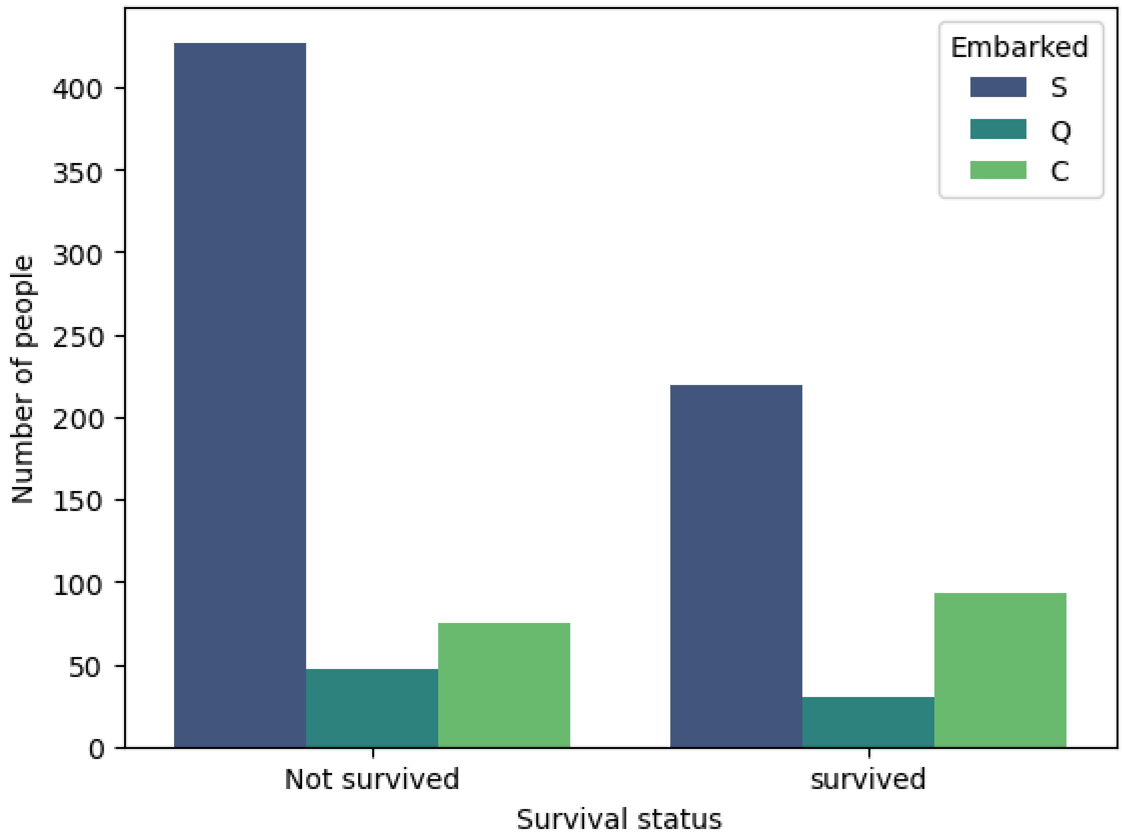
Males from P class 3 had the least survival rate

Visualization of people survived from different Embarkment

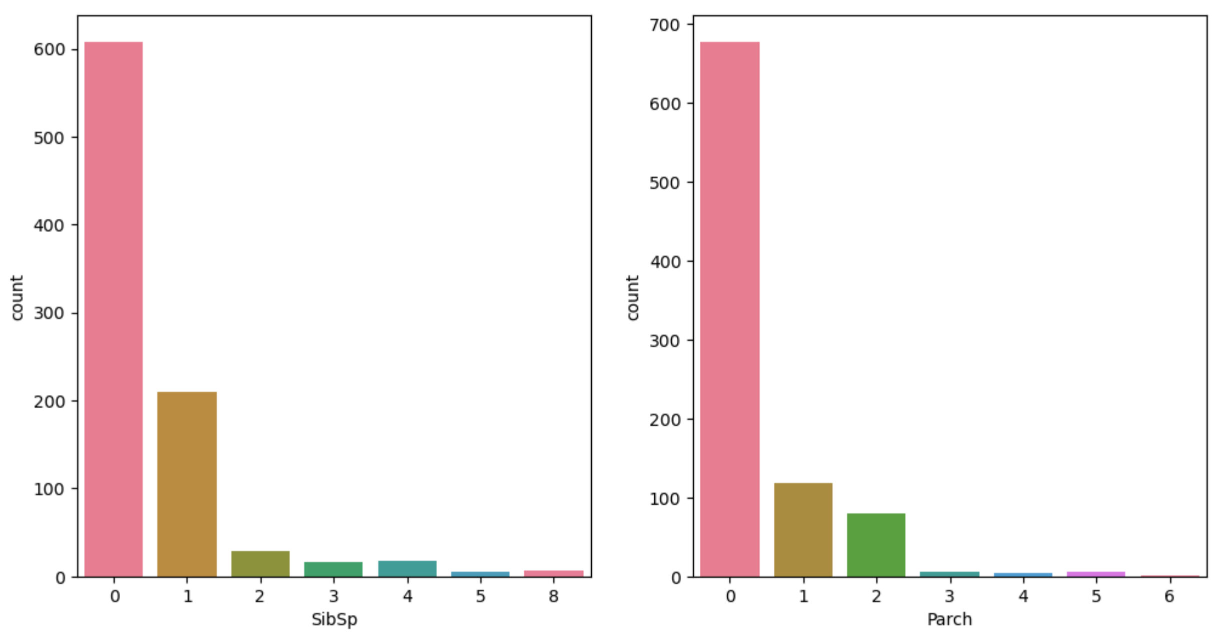
In [38]:
```
sns.countplot(x='Embarked', hue='Embarked', data=df, palette='coolwarm', leg
plt.xlabel("Embarked")
plt.ylabel("Number of people")
plt.show()
```



In [39]:
```
sns.countplot(x='Survived',hue='Embarked',data=df,palette='viridis',)
plt.xlabel("Survival status")
plt.ylabel("Number of people")
plt.xticks(ticks=[0,1],labels=['Not survived','survived'])
plt.show()
```

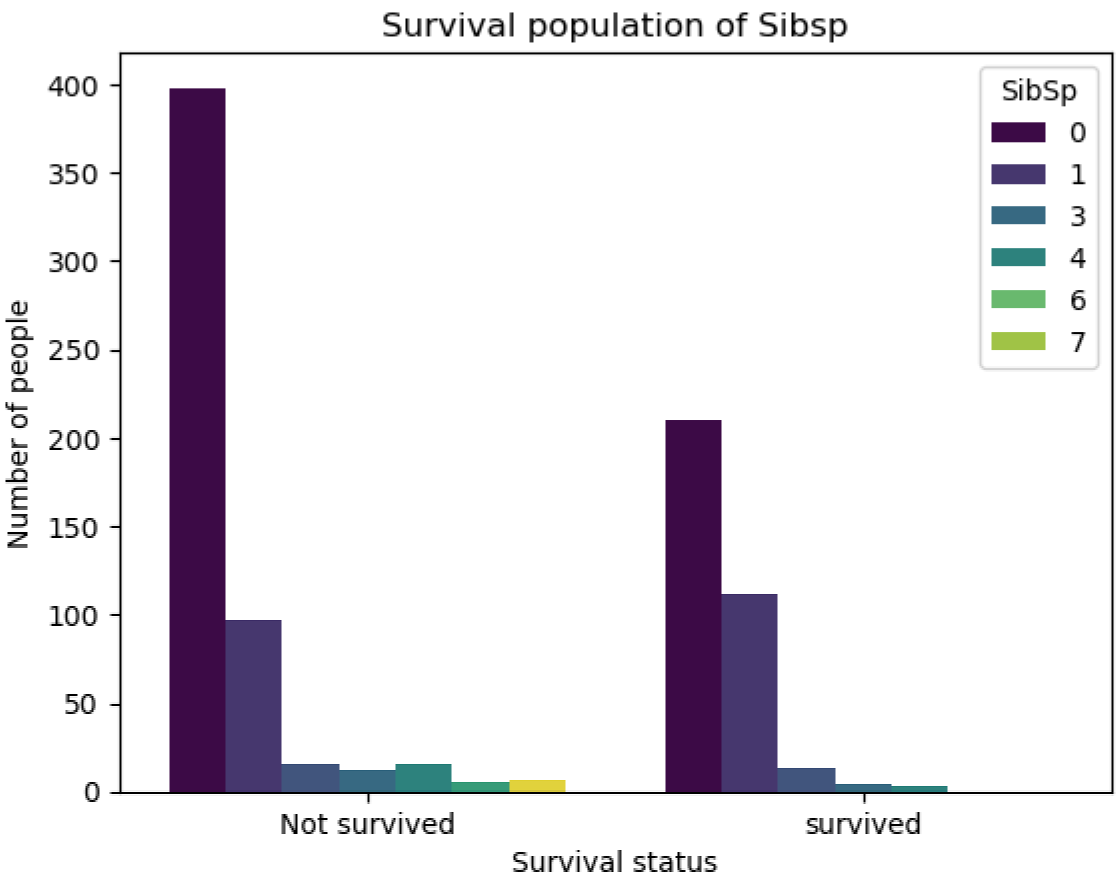


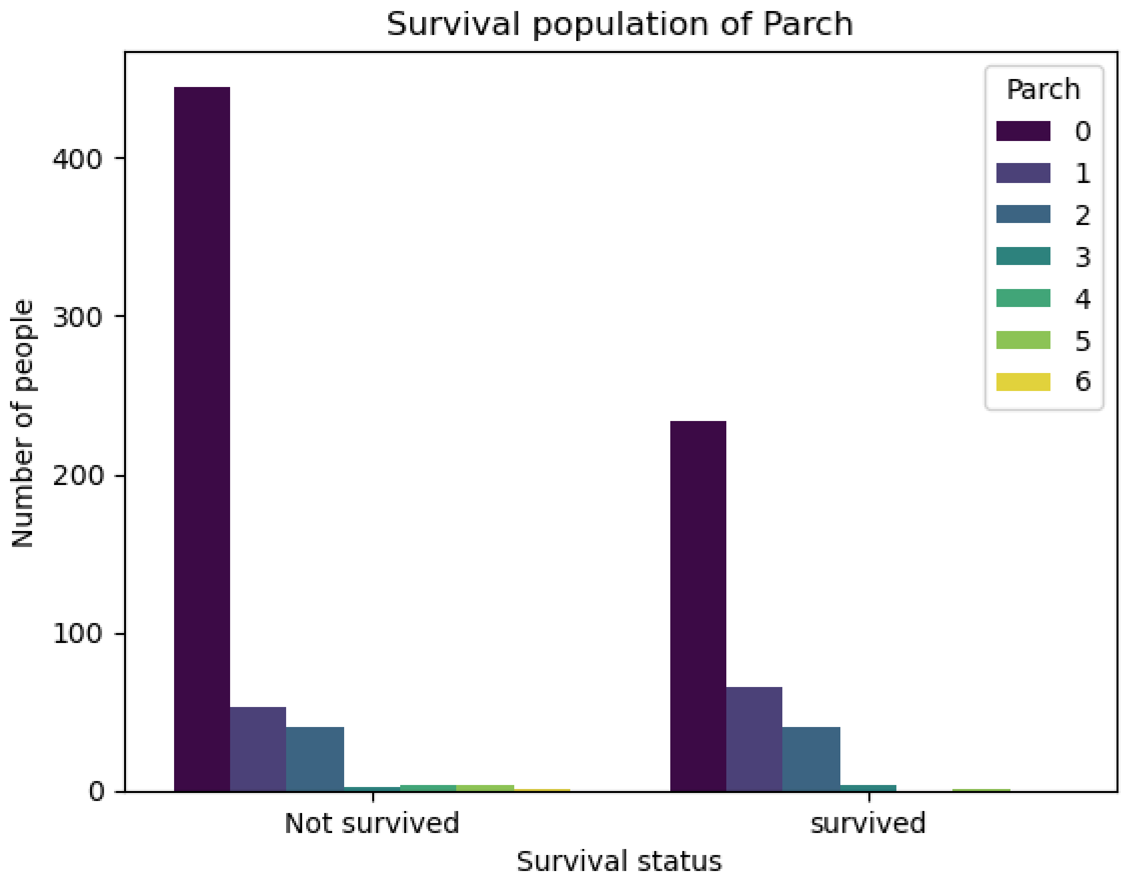
In [51]:
```
fig,axes = plt.subplots(1, 2, figsize=(12, 6))
sns.countplot(x='SibSp', data=df, ax=axes[0], hue='SibSp', palette='husl', l
sns.countplot(x='Parch', data=df, ax=axes[1], hue='Parch', palette='husl', l
plt.show()
```



In [41]:
```
sns.countplot(x ='Survived', hue='SibSp',data=df,palette='viridis')
plt.xticks(ticks=[0,1],labels=['Not survived','survived'])
plt.xlabel("Survival status")
plt.ylabel("Number of people")
plt.title("Survival population of Sibsp")
```

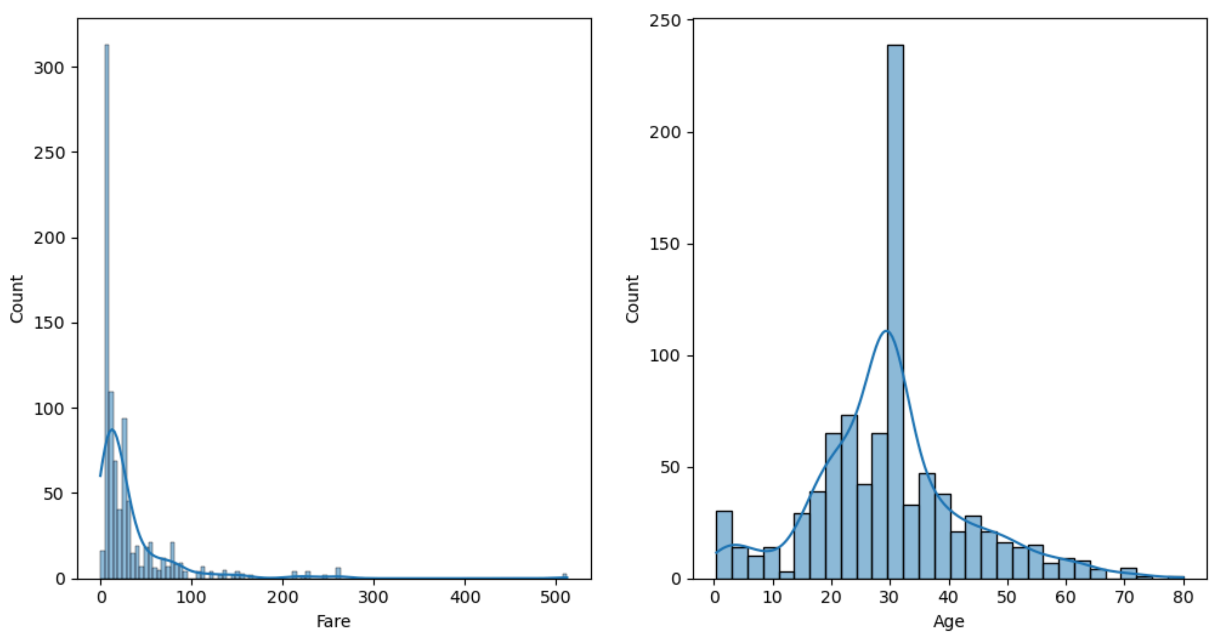
```
plt.show()

sns.countplot(x ='Survived',hue='Parch',data=df,palette='viridis')
plt.xticks(ticks=[0,1],labels=['Not survived','survived'])
plt.title("Survival population of Parch")
plt.xlabel("Survival status")
plt.ylabel("Number of people")
plt.show()
```





Distribution of Fare and Age

In [42]:
```
fig,axes = plt.subplots(1, 2, figsize=(12, 6))
sns.histplot(df['Fare'], kde=True,ax=axes[0])
sns.histplot(df['Age'].dropna(),kde=True,ax=axes[1])
plt.show()
```



Visualizing survival rate in different age category

Define cut points and label names

In [44]:
```
cut_points = [ 0, 5, 12, 18, 35, 60, 100]
label_names = [ 'Infant', "Child", 'Teenager', "Young Adult", 'Adult', 'Seni
```

Create the " Age_categories " column

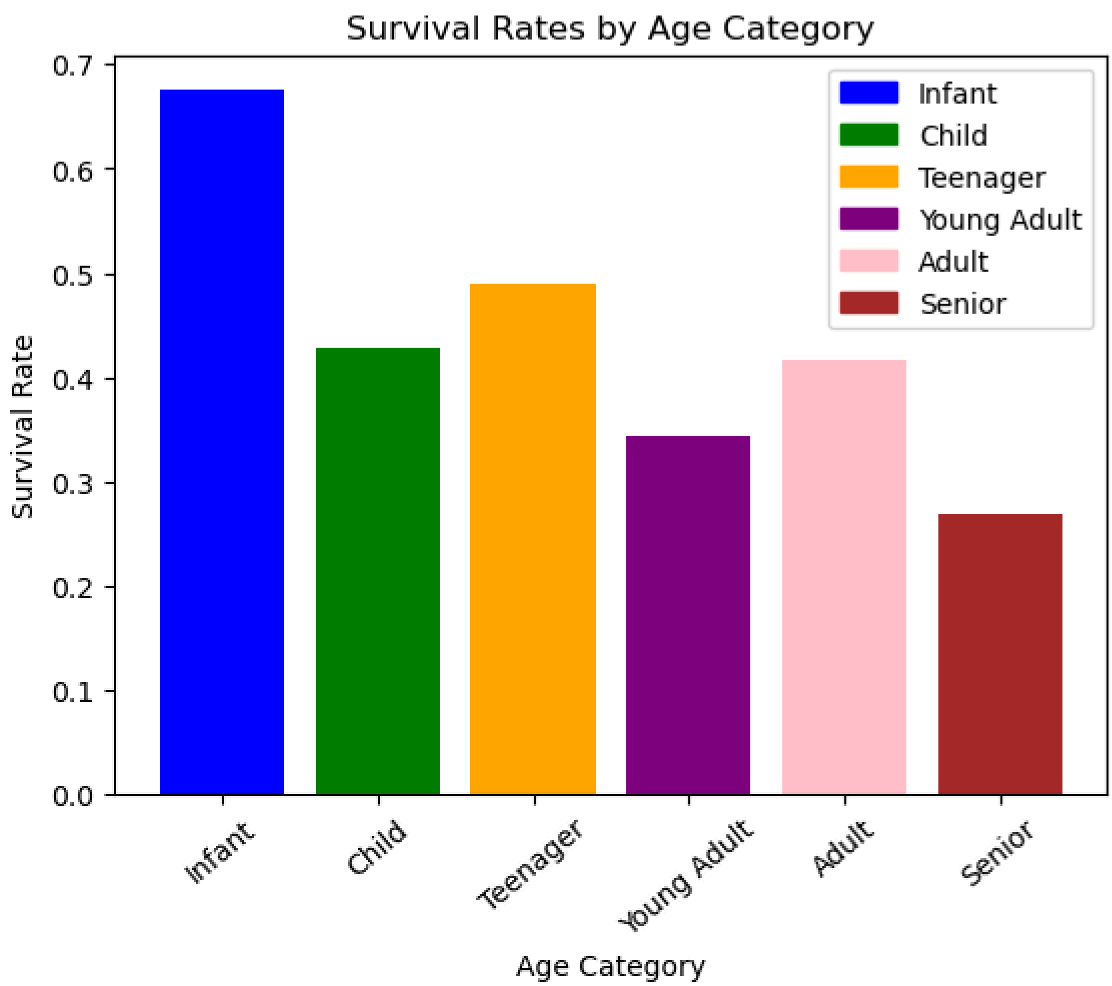
In [45]:
```
df['Age_categories'] = pd.cut(df['Age'], bins=cut_points, labels=label_names
```

Creating a pivot table for survival rates based on age categories

In [49]:
```
age_cat_pivot = df.pivot_table(index="Age_categories", values="Survived", ob
```

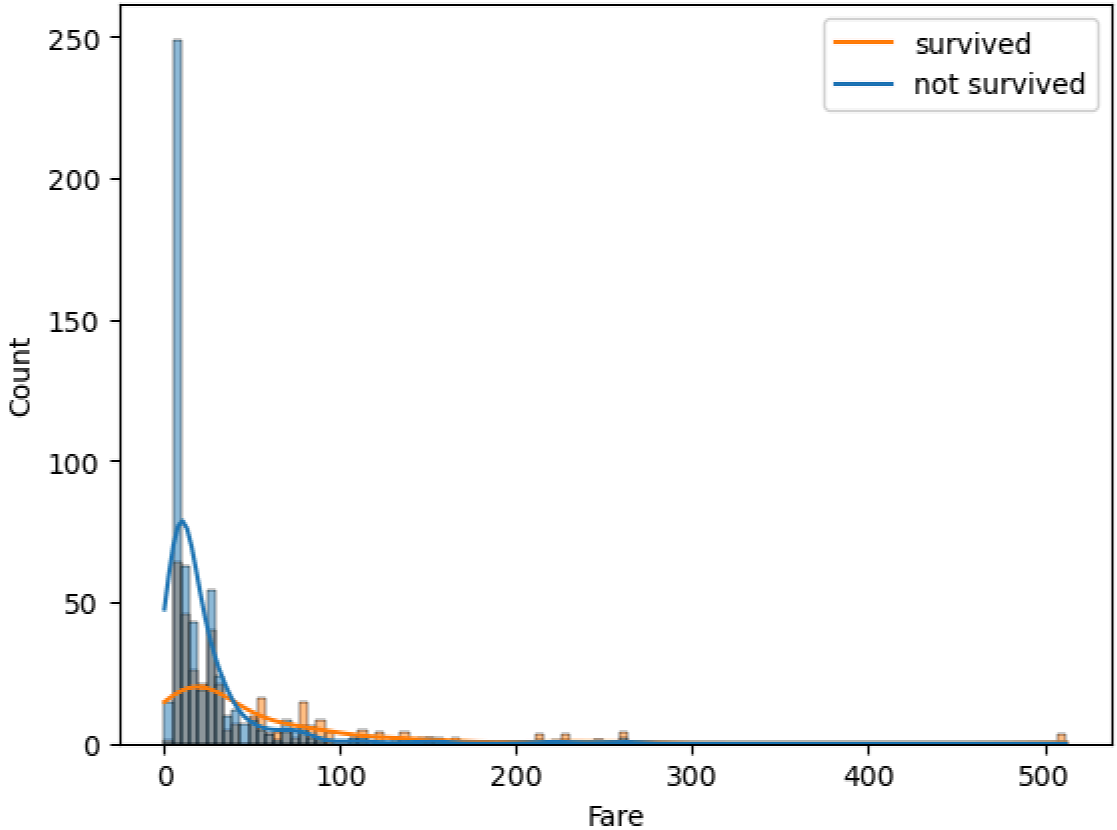
Plotting of the bar Chart

In [50]:
```
# Define colors for each bar
colors = ['blue', 'green', 'orange', 'purple', 'pink', 'brown']

# Plotting the bar chart with different colors for each bar
fig, ax = plt.subplots()
bars = ax.bar(age_cat_pivot.index, age_cat_pivot['Survived'], color=colors)

# Adding a legend with the specified colors
handles = [plt.Rectangle((0, 0), 1, 1, color=colors[i]) for i in
range(len(colors))]
ax.legend(handles, label_names)
ax.set_title('Survival Rates by Age Category')
ax.set_xlabel('Age Category')
ax.set_ylabel('Survival Rate')
plt.xticks(rotation=40)
plt.show()
```

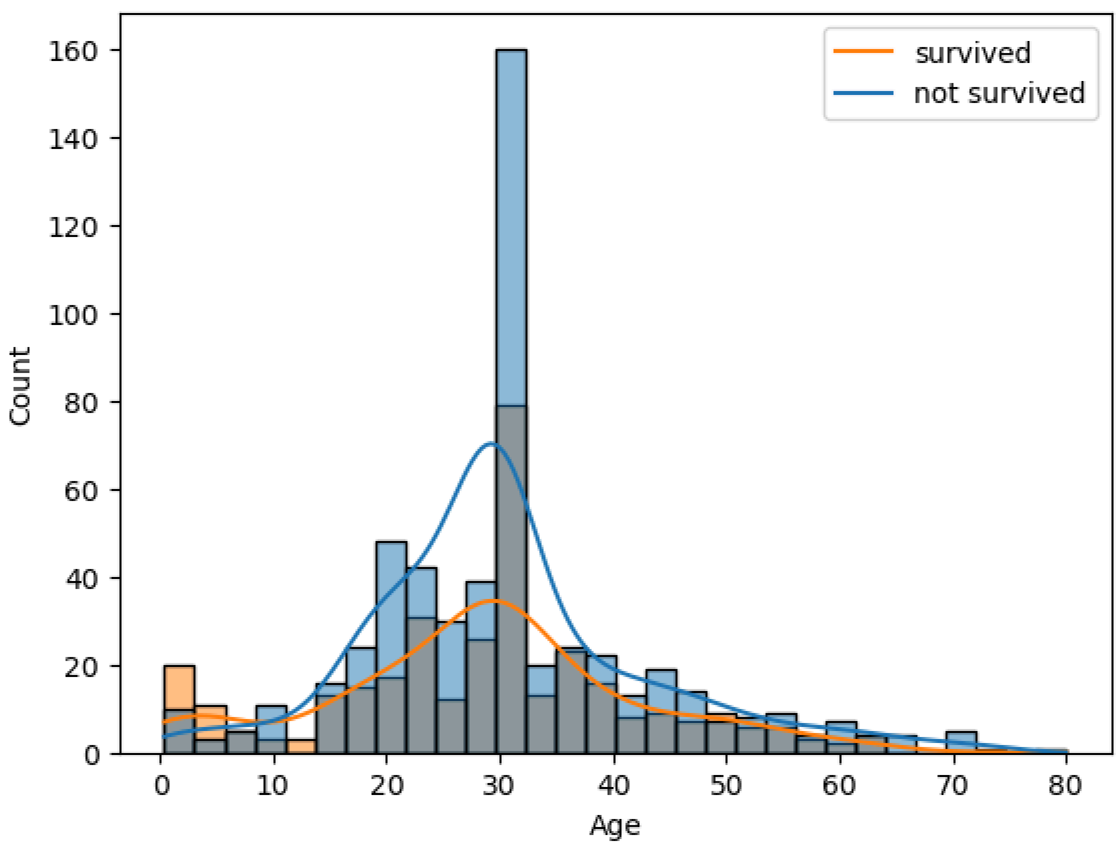



```
Observation : Young adult had the least survival rate
```

In [52]:
```
sns.histplot(x='Fare',hue='Survived',data=df,kde=True)
plt.legend(labels=['survived','not survived'])
plt.show()
```



In [53]:
```
sns.histplot(x='Age',hue='Survived',data=df,kde=True)
plt.legend(labels=['survived','not survived'])
plt.show()
```



Checking for correlation

In [56]:
```
df.replace({'Sex': {'male': 1, 'female': 0}, 'Embarked': {'S': 0, 'C': 1, 'Q
```

In [57]:
```
df.head()
```

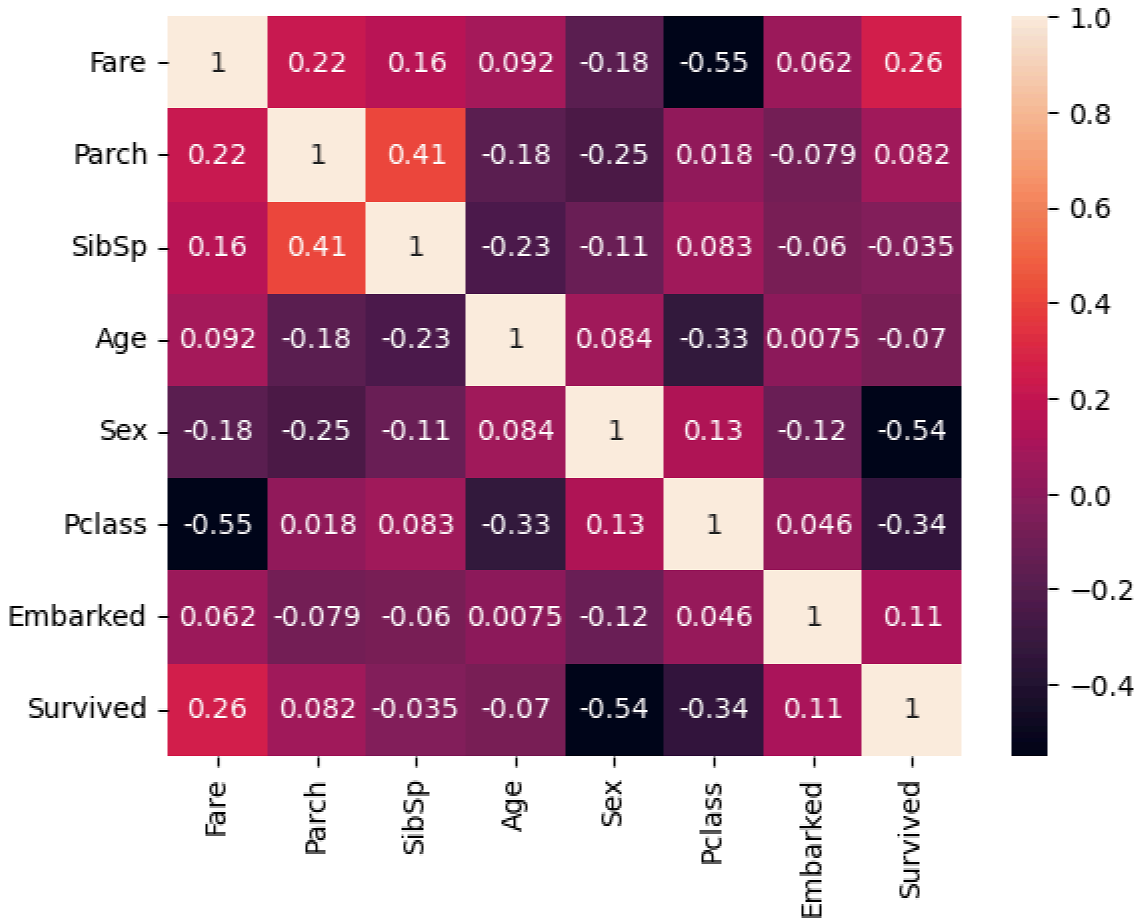
Out[57]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticke
0	1	0	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 2117
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	1	0	PC 1759
2	3	1	3	Heikkinen, Miss. Laina	0	26.0	0	0	STON/O2 310128
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	35.0	1	0	11380
4	5	0	3	Allen, Mr. William Henry	1	35.0	0	0	37345

In [58]:
```
df_num = df[['Fare','Parch','SibSp','Age','Sex','Pclass','Embarked','Survive
```

Plot : Heat Map

In [60]:
```
sns.heatmap(df_num.corr(),annot=True)
plt.show()
```




```
Observation: Fare, sex, Pclass, Embarked has correlation with
survived column
```

Conclusion:

Gender Disparity: Females had a higher survival rate, reflecting the "women and children first" policy during the Titanic disaster.

Passenger Class Disparity: Class 3, with the highest number of passengers, had the lowest survival rate, suggesting socio-economic status influenced survival chances.

Gender and Passenger Class Interaction: Males in Class 3 had the lowest survival rate, showing the combined impact of gender and socio-economic status on survival.

Age Factor: Young adults had the lowest survival rate, likely due to the prioritization of women and children.

Correlation with Survival: Factors like fare, sex, passenger class, and embarkation point were key to survival chances.