impoting the libraries like pandas,seaborn and matplotlib loading the data set

In [4]:
```
import warnings
warnings.simplefilter("ignore",category=FutureWarning)
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(style="whitegrid")
df=pd.read_csv("C:/Users/RGUKT/OneDrive/Desktop/internship_task5/titanic_dataset.cs
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 |

The .info() method in pandas is used to give a quick overview of your DataFrames structure 1.number of rows and columns 2.column names and data types

In [5]:
```
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
```

describe() method is used to generate descriptive statistics of a dataframe or series. it provides a quick overview of the central tendency,dispersion,and shapeof a datasets's distribution,excluding NaN values by default By default it returns: count:number of non-null entries mean:Average value std:standard deviation min:minimum value 25%,50%,75% percentiles max:maximim value

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

Out[7]:

| | 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 |

The value_counts() method in Pandas is used to count the number of occurrences of each unique value in a Series i.e, a single column. It is mainly used for analyzing categorical data or discrete values. Frequency Distribution-Shows how many times each unique value appears in a column. Categorical Data Analysis-Helps understand how data is distributed across different categories (e.g., departments, product types, regions). Detect Imbalance-Useful in

identifying class imbalances in classification problems. Data Cleaning-Detect unexpected or misspelled categories or values

In [4]:
```
df.value_counts()
```

Out[4]:
```
PassengerId  Survived  Pclass  Name                                                
Sex     Age   SibSp  Parch  Ticket    Fare      Cabin  Embarked
2            1         1      Cumings, Mrs. John Bradley (Florence Briggs Thayer)
female  38.0  1      0      PC 17599  71.2833   C85    C           1
572          1         1      Appleton, Mrs. Edward Dale (Charlotte Lamson)
female  53.0  2      0      11769     51.4792   C101   S           1
578          1         1      Silvey, Mrs. William Baird (Alice Munger)
female  39.0  1      0      13507     55.9000   E44    S           1
582          1         1      Thayer, Mrs. John Borland (Marian Longstreth Morri
s)  female  39.0  1      1      17421     110.8833  C68    C           1
584          0         1      Ross, Mr. John Hugo
male    36.0  0      0      13049     40.1250   A10    C           1

..
328          1         2      Ball, Mrs. (Ada E Hall)
female  36.0  0      0      28551     13.0000   D      S           1
330          1         1      Hippach, Miss. Jean Gertrude
female  16.0  0      1      111361    57.9792   B18    C           1
332          0         1      Partner, Mr. Austen
male    45.5  0      0      113043    28.5000   C124   S           1
333          0         1      Graham, Mr. George Edward
male    38.0  0      1      PC 17582  153.4625  C91    S           1
890          1         1      Behr, Mr. Karl Howell
male    26.0  0      0      111369    30.0000   C148   C           1
Name: count, Length: 183, dtype: int64
```

The isnull().sum() method in Pandas is commonly used to detect missing (null or NaN) values in a DataFrame or Series. Identify columns with missing values Count how many values are missing per column

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
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
```

Calculates the mean of the Age column. Replaces every value in the Age column (including non-null ones) with that mean.

In [7]:
```
df['Age']=df['Age'].mean()
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

When you use .astype(str) before filling missing values, it converts NaN to the string 'nan', not Python's np.nan. So fillna('unknown') has no effect, because there are no actual NaN values anymore only the string ` Missing values are replaced correctly. Column is clean and explicitly of string type.

In [10]:
```
df['Embarked']=df['Embarked'].astype(str)
df['Embarked'].fillna('unknown',inplace=True)
df['Cabin']=df['Cabin'].astype(str)
df['Cabin'].fillna('unknown',inplace=True)
```

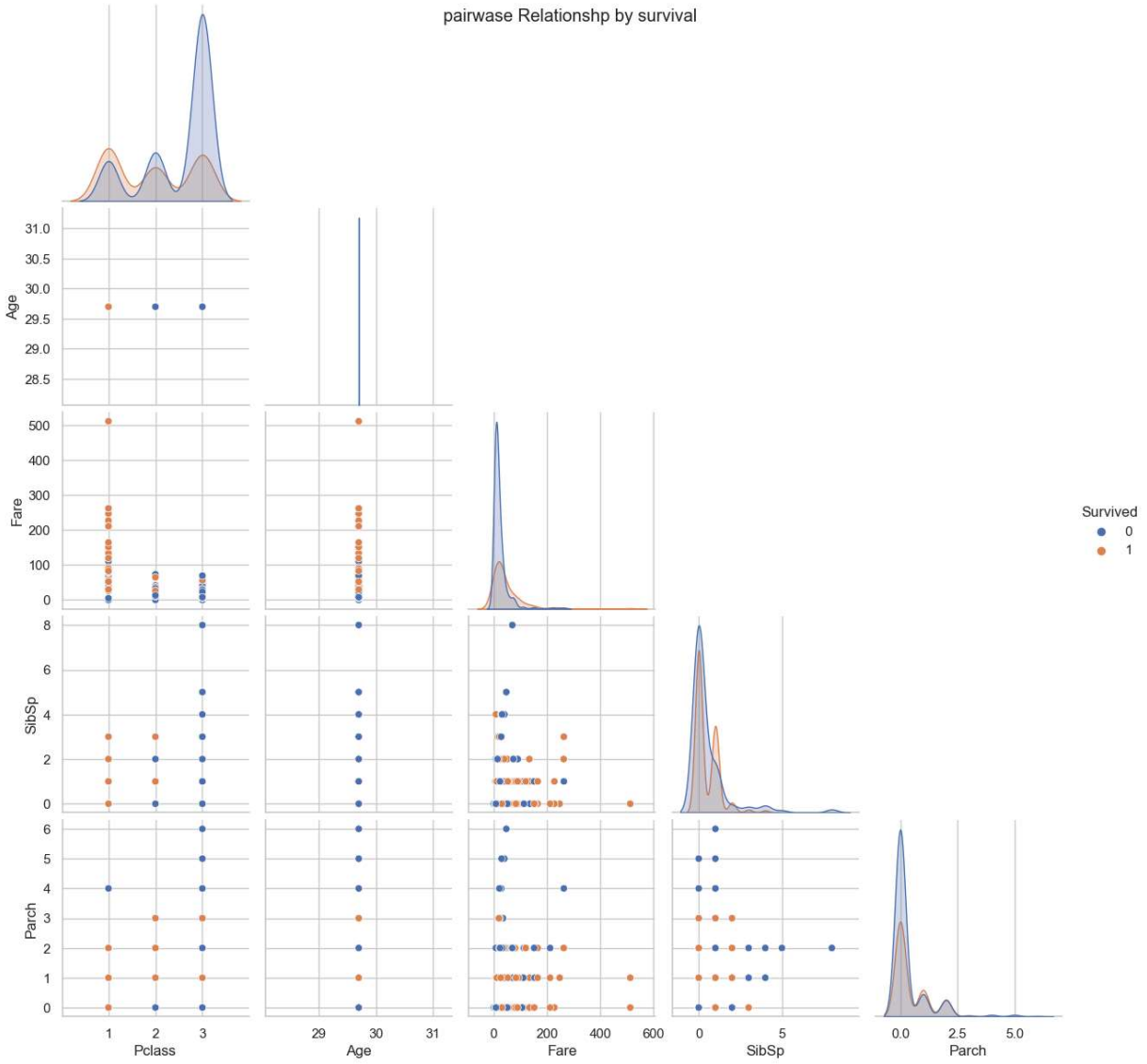
In [11]:
```
df.isnull().sum()
```

Out[11]:
```
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
```

Pick columns you want to study from your dataset – like Age, Fare, Class, etc. Use pairplot() from Seaborn to: Show how these features are related to each other using small scatter plots. Use different colors to show whether the person survived or not (based on the Survived column). Add a title at the top of the chart using plt.suptitle(). Women survived more than men. 1st class passengers survived more than 3rd class. Children and women in higher classes had the highest survival rates.

In [12]:
```
numeric_features=['Survived','Pclass','Age','Fare','SibSp','Parch']
sns.pairplot(df[numeric_features],hue='Survived',corner=True)
plt.suptitle("pairwase Relationshp by survival")
```

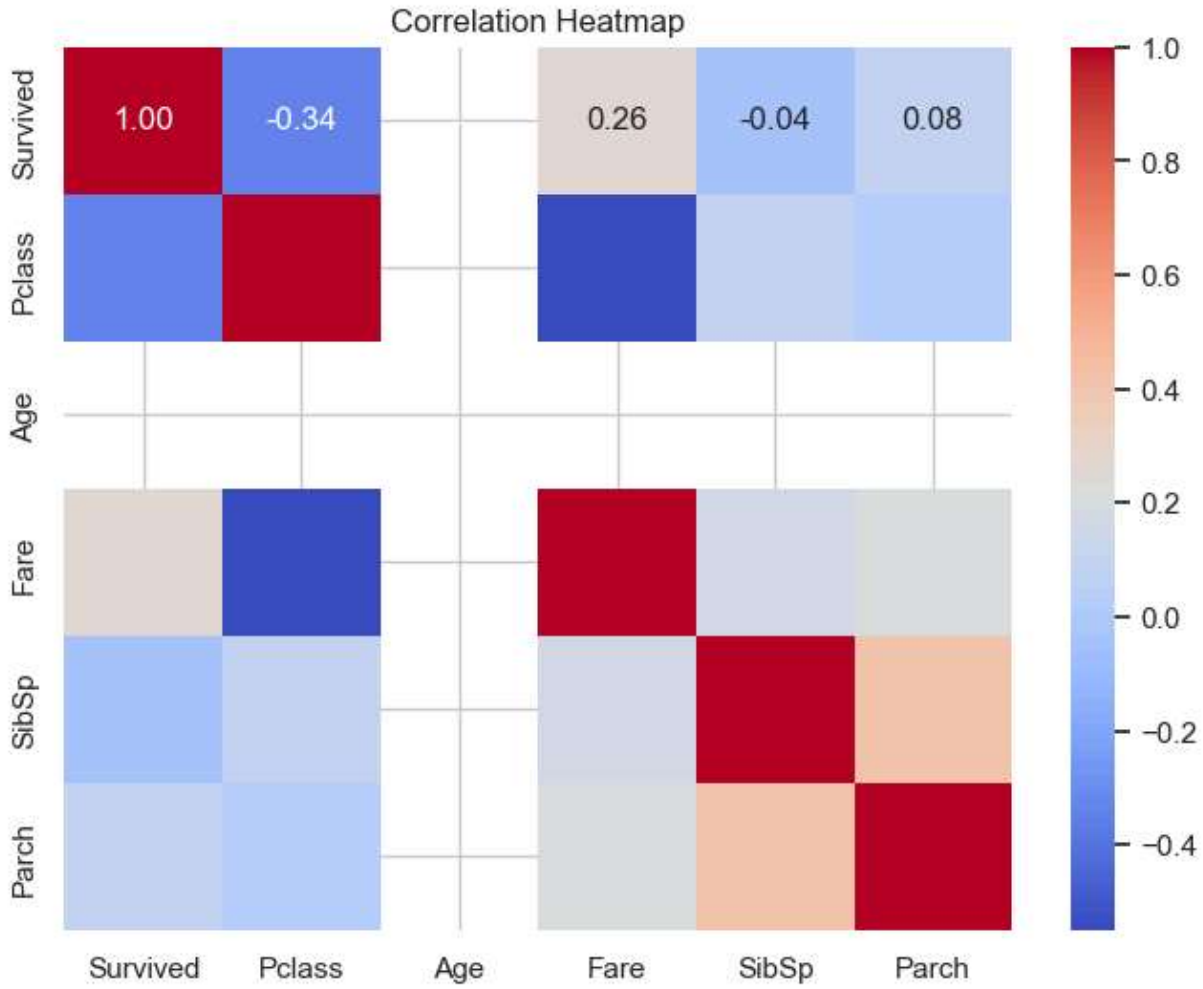
Out[12]: Text(0.5, 0.98, 'pairwase Relationshp by survival')




To see which features are related (positively or negatively). Helps decide which features might be useful for prediction (e.g. in a machine learning model). Example: Fare might have a strong positive correlation with Pclass.

In [13]:
```
corr = df[numeric_features].corr()
# Heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap")
```

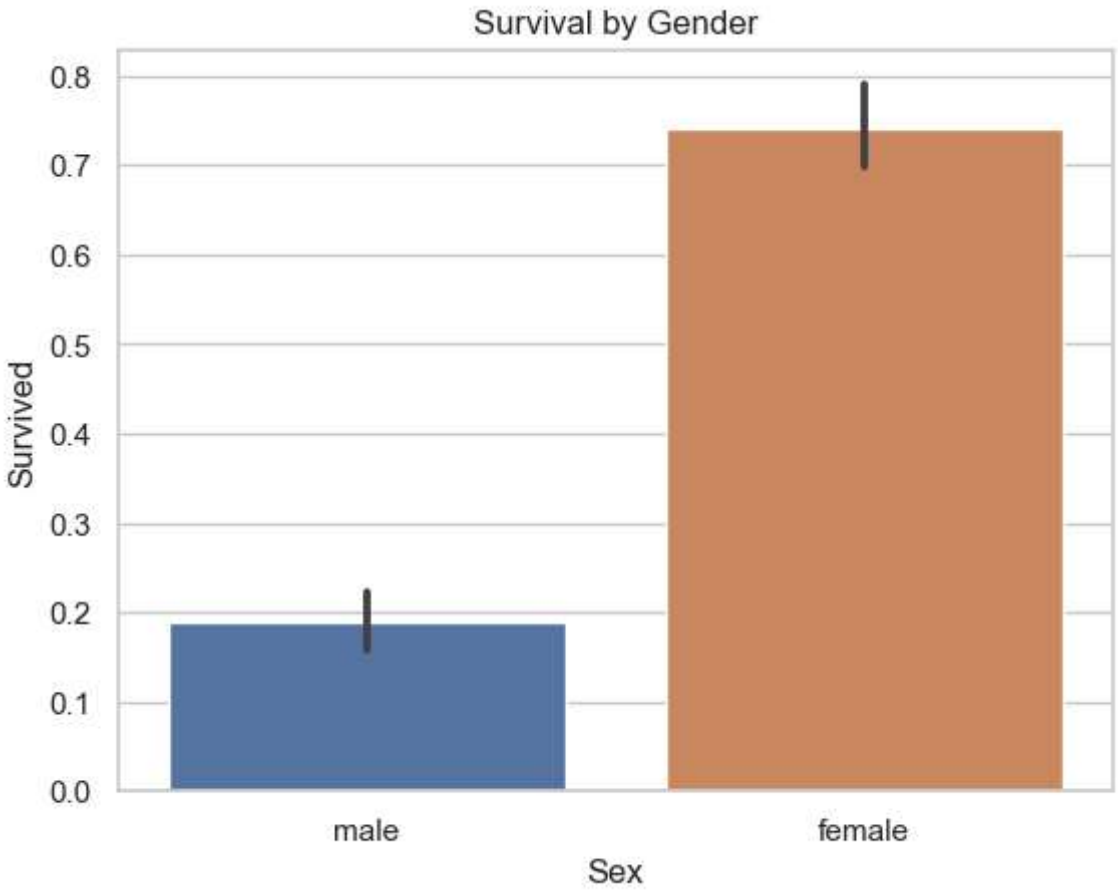
Out[13]: Text(0.5, 1.0, 'Correlation Heatmap')




Two bars: one for male and one for female The female bar will be higher, because women had a much higher survival rate on the Titanic.

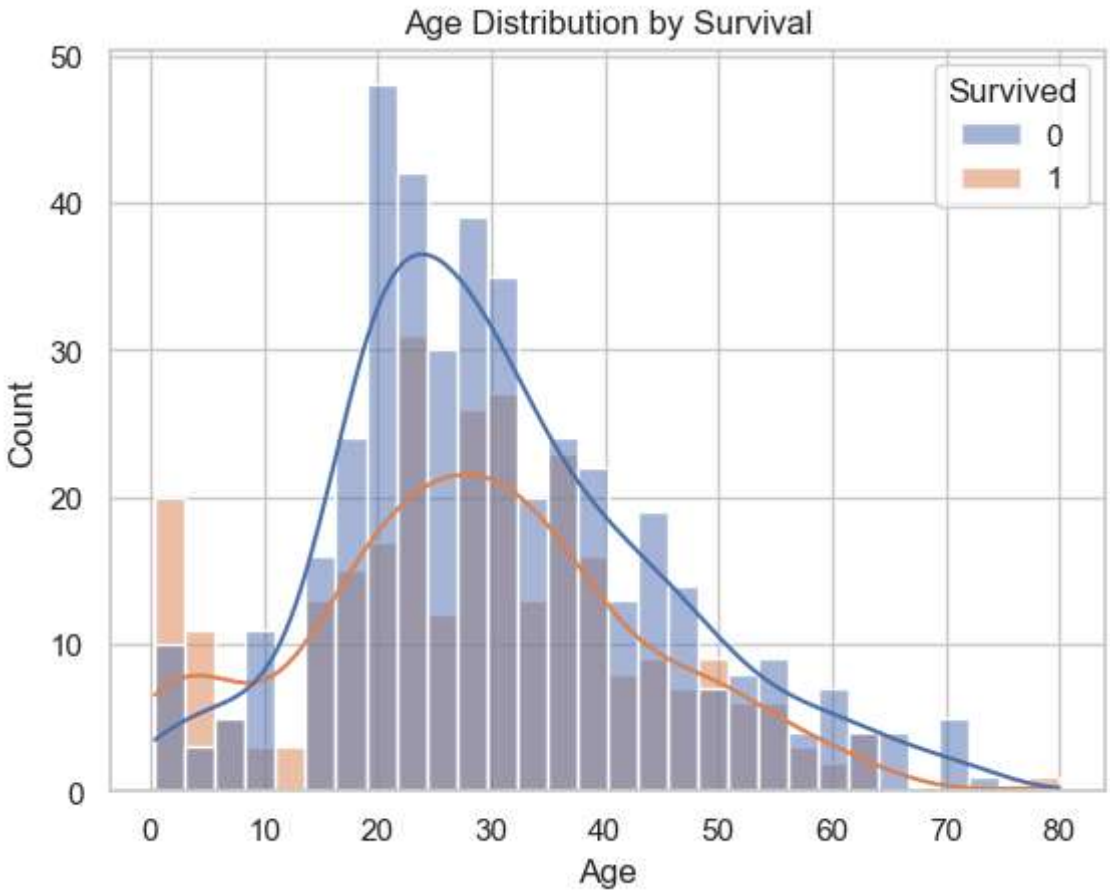
In [5]:
```
import seaborn as sns
import matplotlib.pyplot as plt

ax=sns.barplot(data=df, x='Sex',y='Survived')
ax.set_title("Survival by Gender")
plt.show()
```



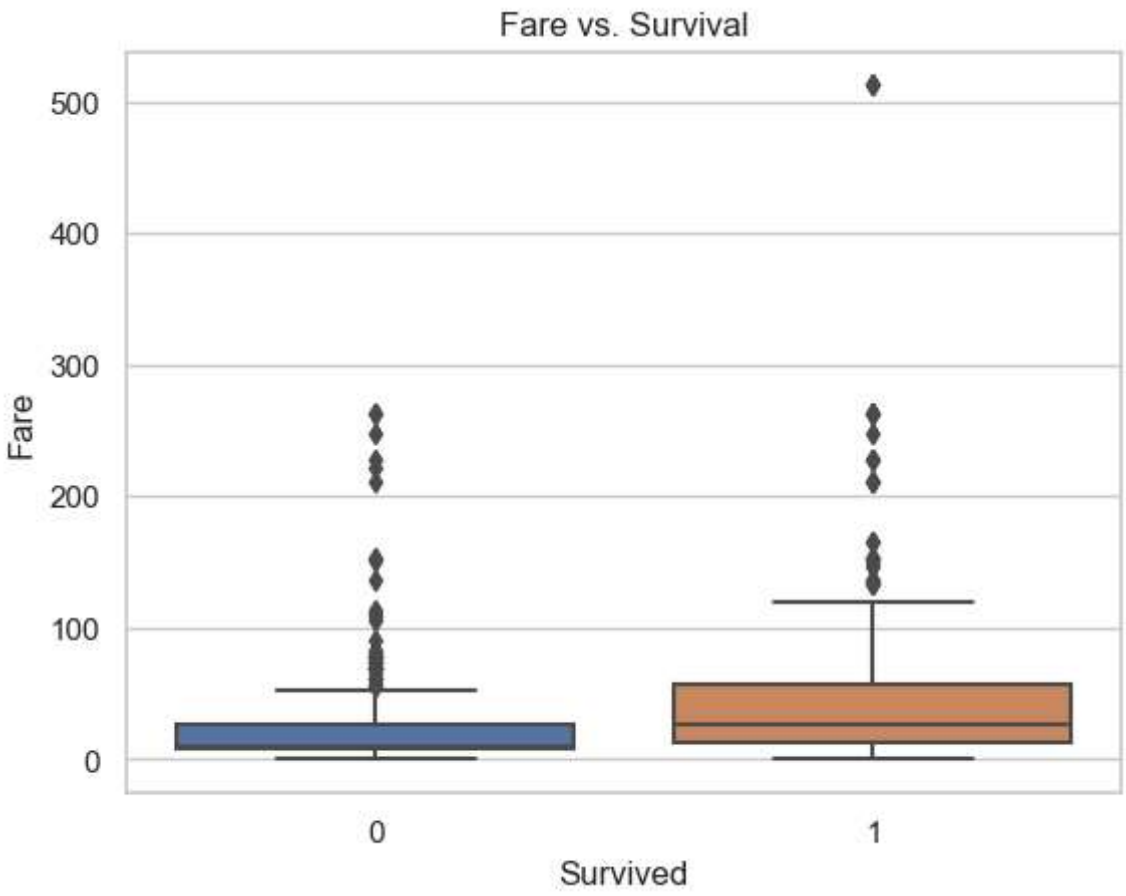

See if certain age groups survived more (e.g. children). Understand how age affects survival. Spot patterns or imbalances in the data.

In [6]:
```
sns.histplot(data=df, x='Age', hue='Survived', bins=30, kde=True)
plt.title("Age Distribution by Survival")
plt.show()
```




Whether people who paid higher fares had a better chance of survival Whether the fare distribution is different for survivors vs non-survivors Helps detect if Fare might be an important feature for prediction

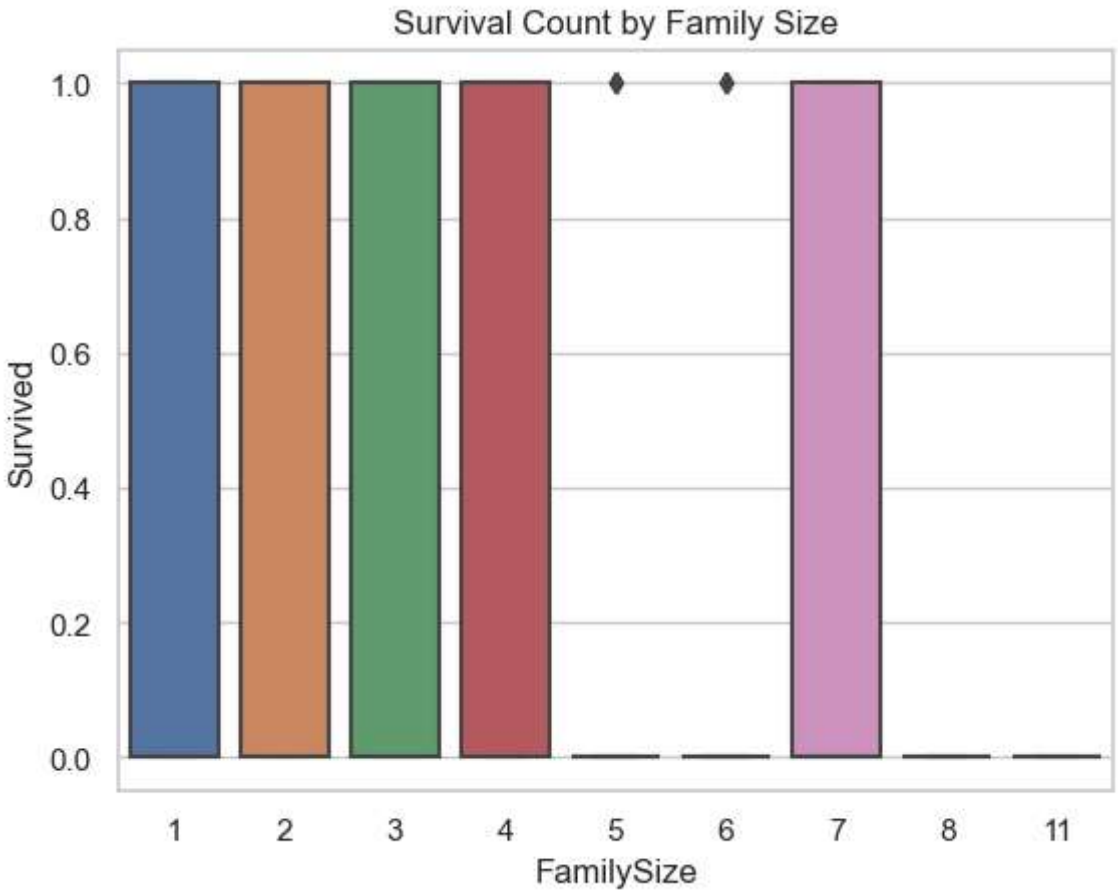
In [75]:
```
sns.boxplot(data=df, x='Survived', y='Fare')
plt.title("Fare vs. Survival")
plt.show()
```




How likely passengers were to survive based on the size of their family. Very small families (1–2 people) had a decent chance of survival. Medium-sized families may have done better. Very large families may have had a lower survival rate.

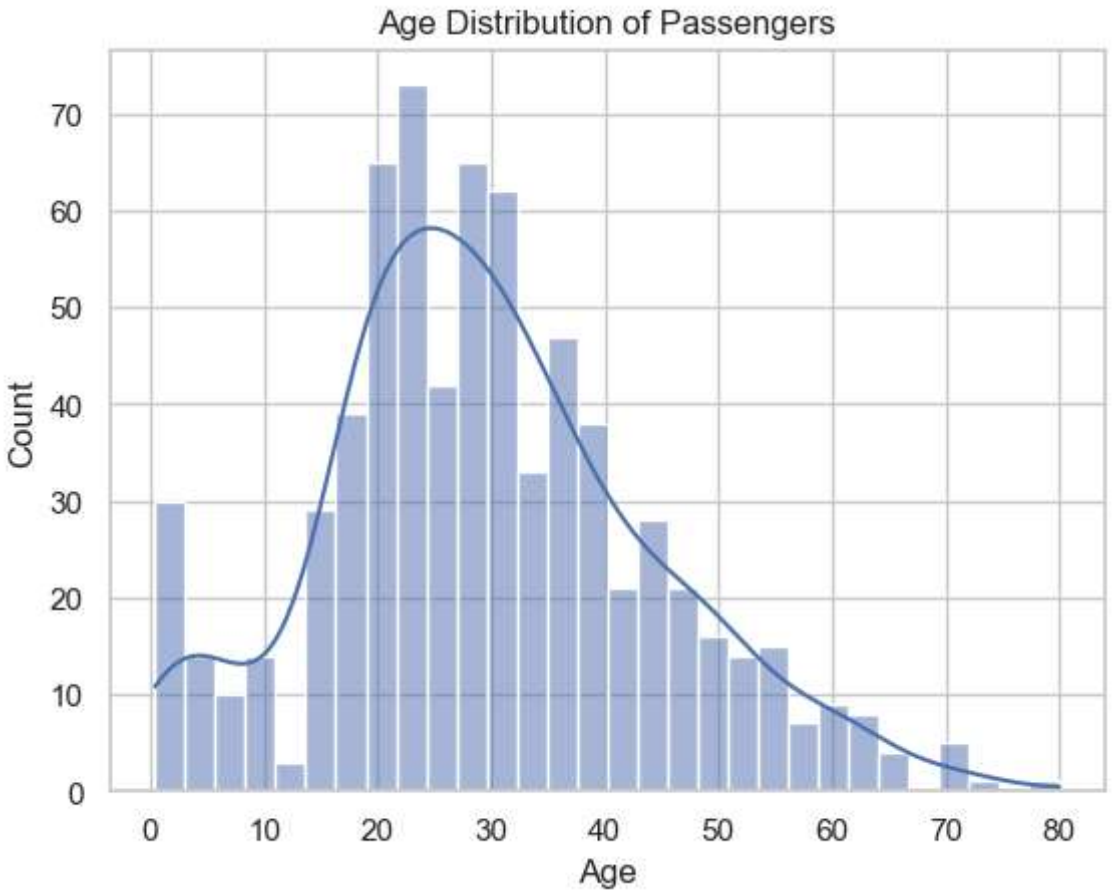
In [9]:
```
df['FamilySize'] = df['SibSp'] + df['Parch'] + 1

sns.boxplot(data=df, x='FamilySize', y='Survived')
plt.title("Survival Count by Family Size")
plt.show()
```



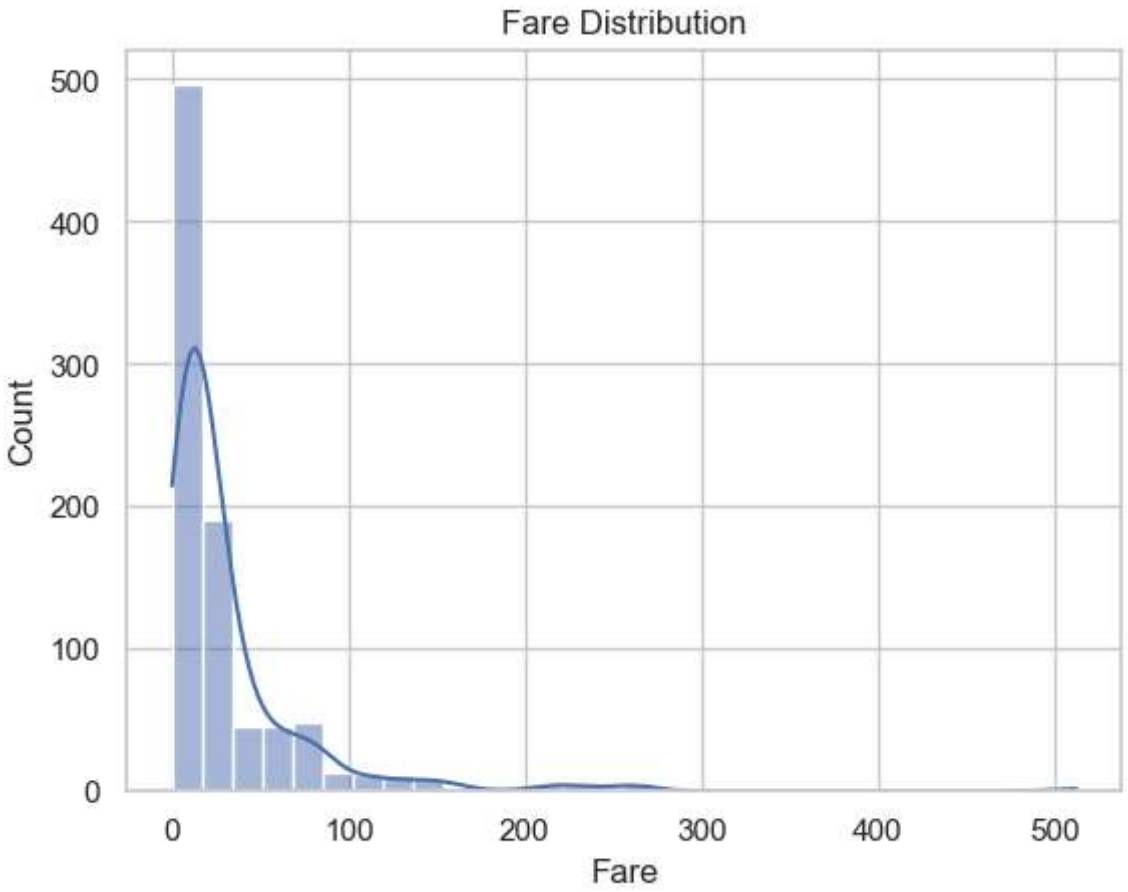

Helps you understand the age spread of passengers. Which age groups had more passengers (e.g., 20s–30s). Gaps or missing data in certain age ranges. If age distribution is skewed (not symmetrical).

In [8]:
```
sns.histplot(data=df, x='Age', bins=30, kde=True)
plt.title("Age Distribution of Passengers")
plt.xlabel("Age")
plt.ylabel("Count")
plt.show()
```



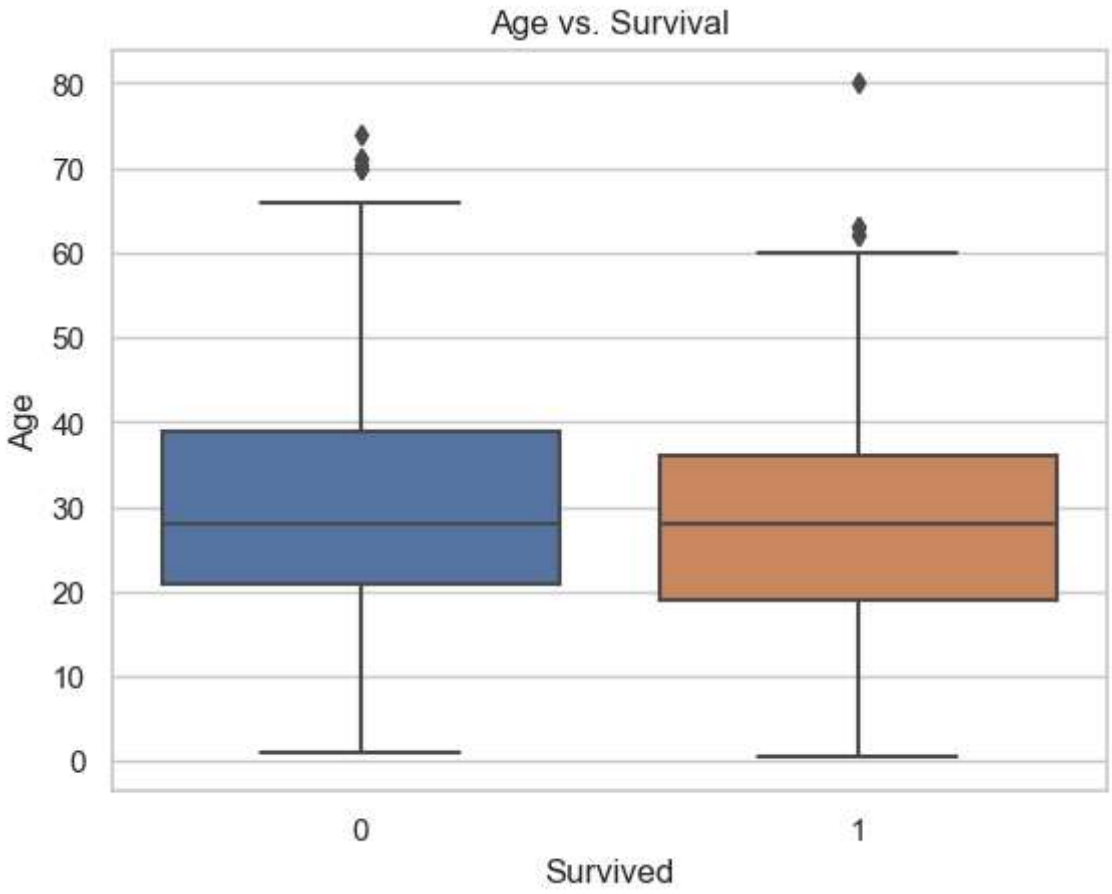

Most passengers paid low fares (you'll likely see a tall peak near the left). A few passengers paid very high fares (long tail to the right). This is a right-skewed distribution, which is common for price-related data.

In [78]:
```
sns.histplot(data=df, x='Fare', bins=30, kde=True)
plt.title("Fare Distribution")
plt.xlabel("Fare")
plt.ylabel("Count")
plt.show()
```



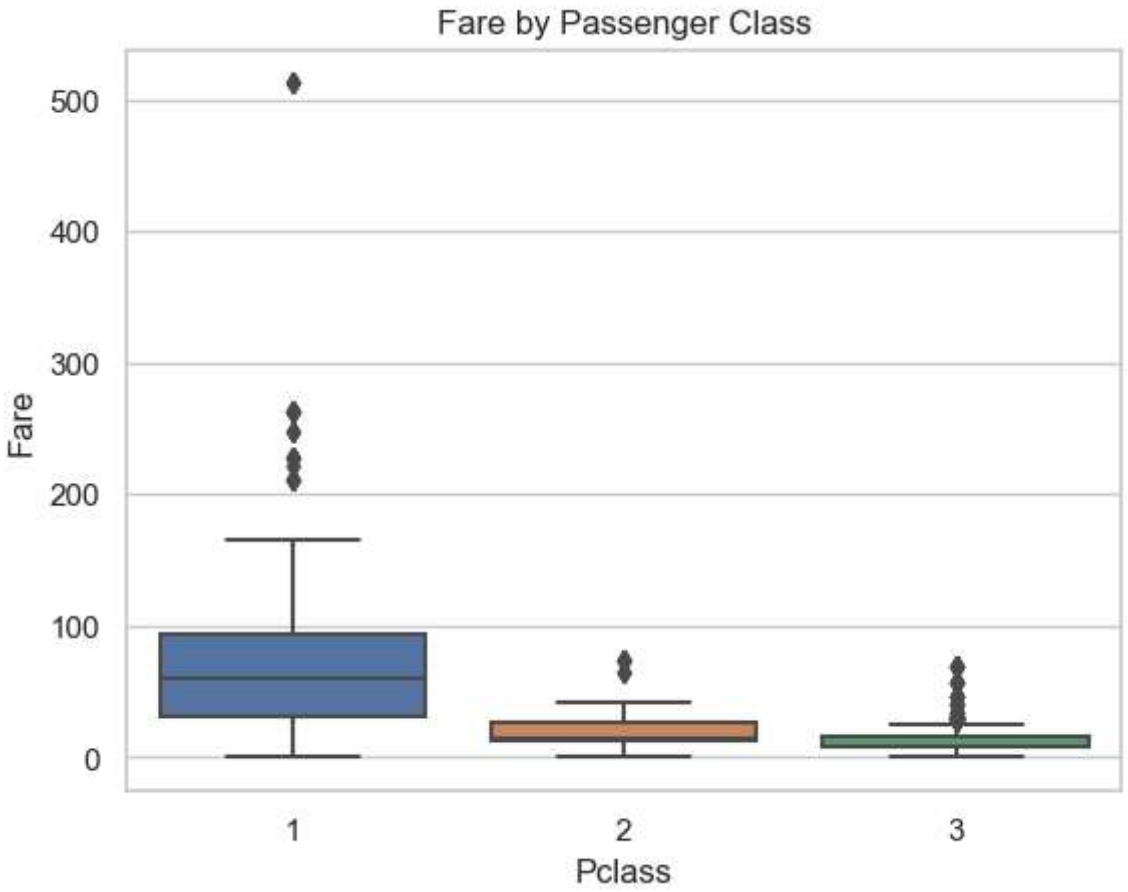

Survivors may have had a slightly lower median age. Some very young passengers (children) may be outliers who mostly survived.

In [10]:
```
sns.boxplot(data=df, x='Survived', y='Age')
plt.title("Age vs. Survival")
plt.xlabel("Survived")
plt.ylabel("Age")
plt.show()
```



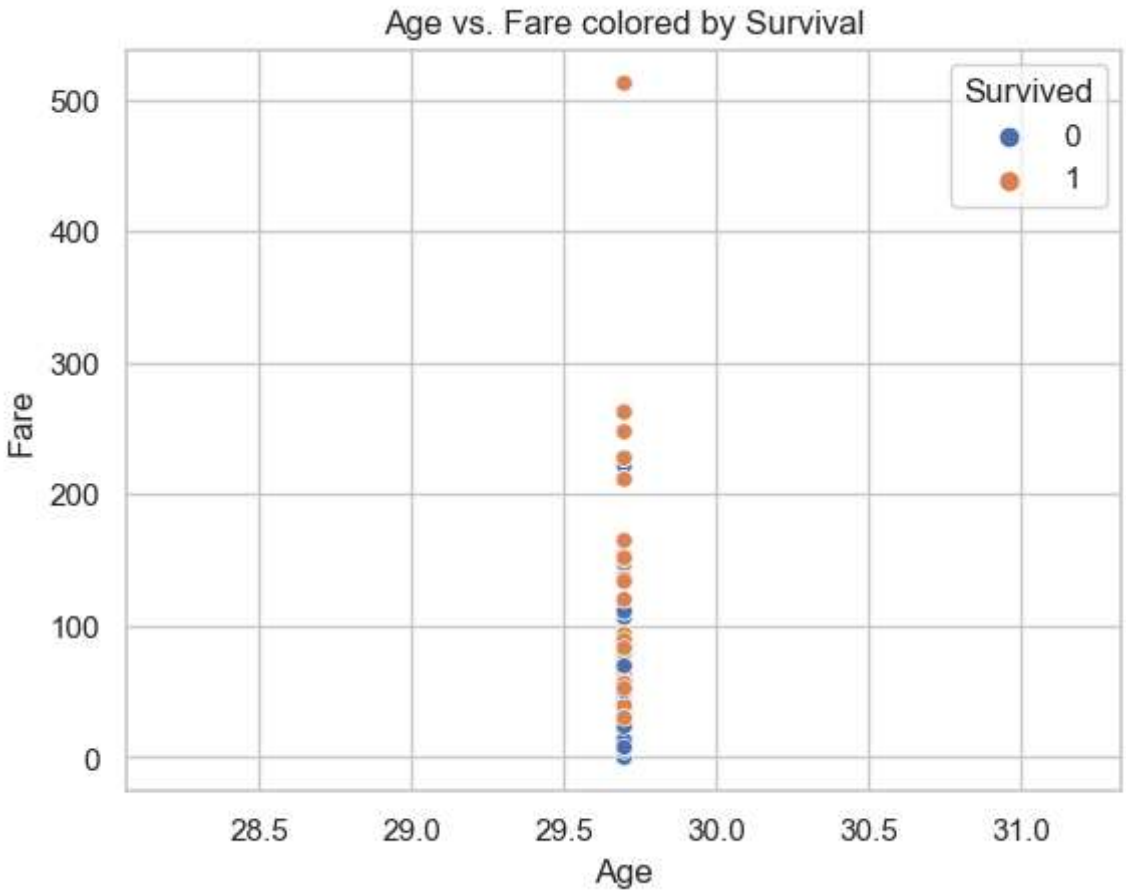

In [ ]:
```
1st class passengers paid the highest fares (tallest box)
3rd class passengers paid the lowest fares
There's a big difference in fare ranges between the classes
You may notice outliers in 1st class who paid much more
```

In [80]:
```
sns.boxplot(data=df, x='Pclass', y='Fare')
plt.title("Fare by Passenger Class")
plt.xlabel("Pclass")
plt.ylabel("Fare")
plt.show()
```



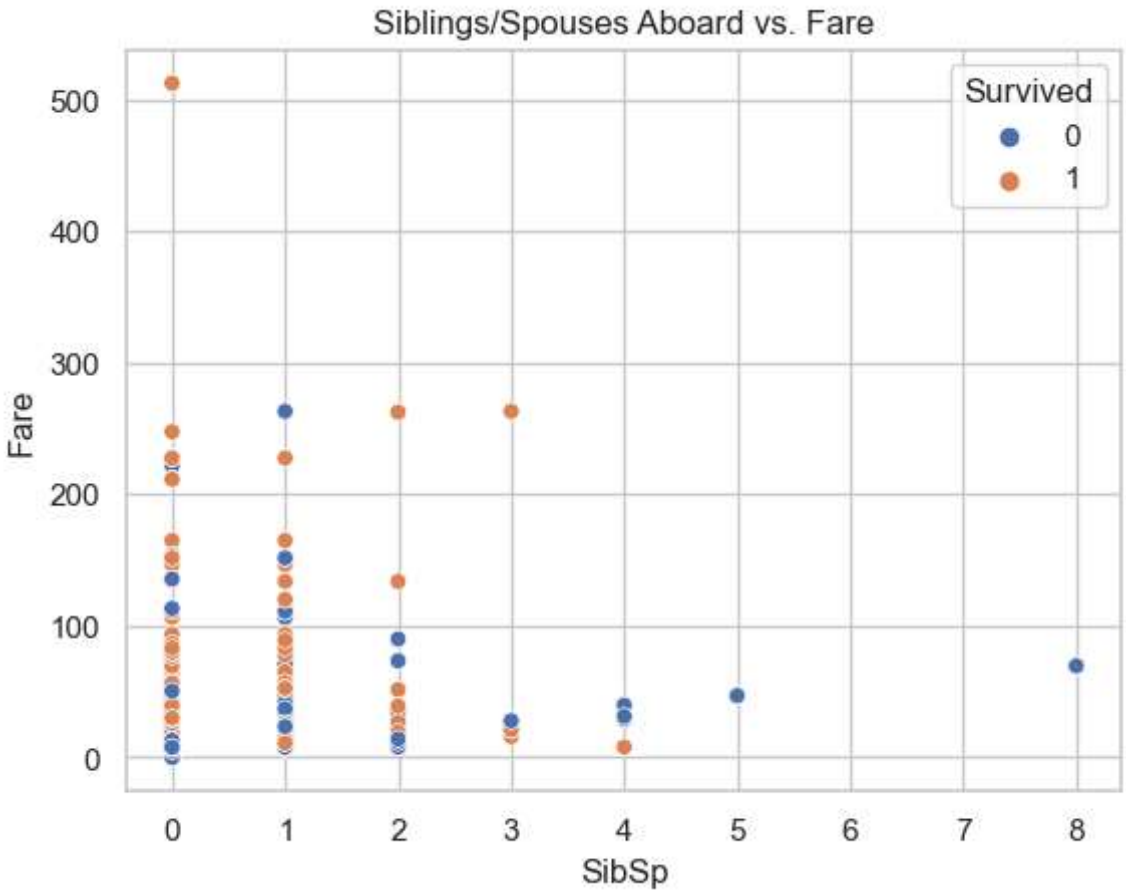

How age and fare relate to survival. Whether people who paid more or were younger were more likely to survive. If there are clusters or outliers (e.g., someone very young with a high fare).

In [81]:
```
sns.scatterplot(data=df, x='Age', y='Fare', hue='Survived')
plt.title("Age vs. Fare colored by Survival")
plt.xlabel("Age")
plt.ylabel("Fare")
plt.show()
```




People with 0 or 1 SibSp might have paid less (possibly traveling alone or as a couple). Survivors might appear more in certain SibSp + Fare combinations (e.g., small families in 1st class).

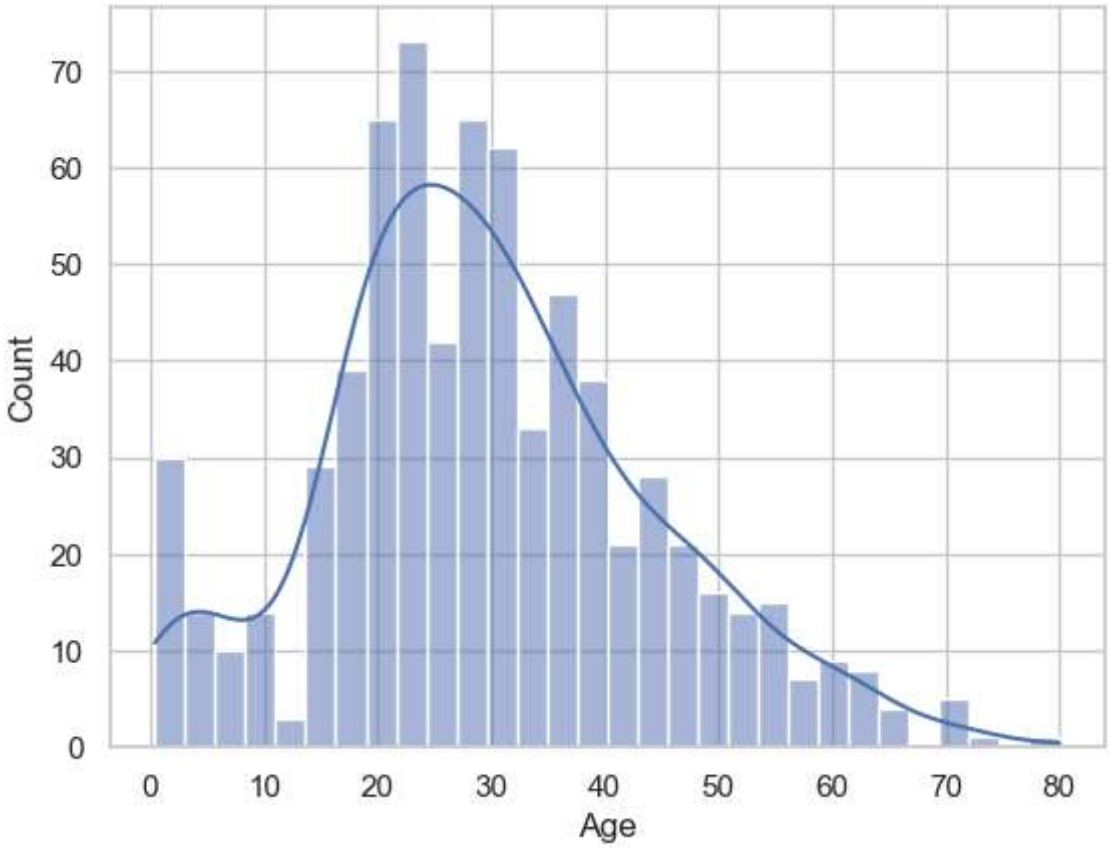
In [82]:
```
sns.scatterplot(data=df, x='SibSp', y='Fare', hue='Survived')
plt.title("Siblings/Spouses Aboard vs. Fare")
plt.show()
```




Helps you understand which age groups were most common on board. Shows whether the age distribution is skewed (e.g., more young people or more older adults). Useful for detecting gaps or outliers in the data.

In [12]:
```
sns.histplot(data=df, x='Age', bins=30, kde=True)
```

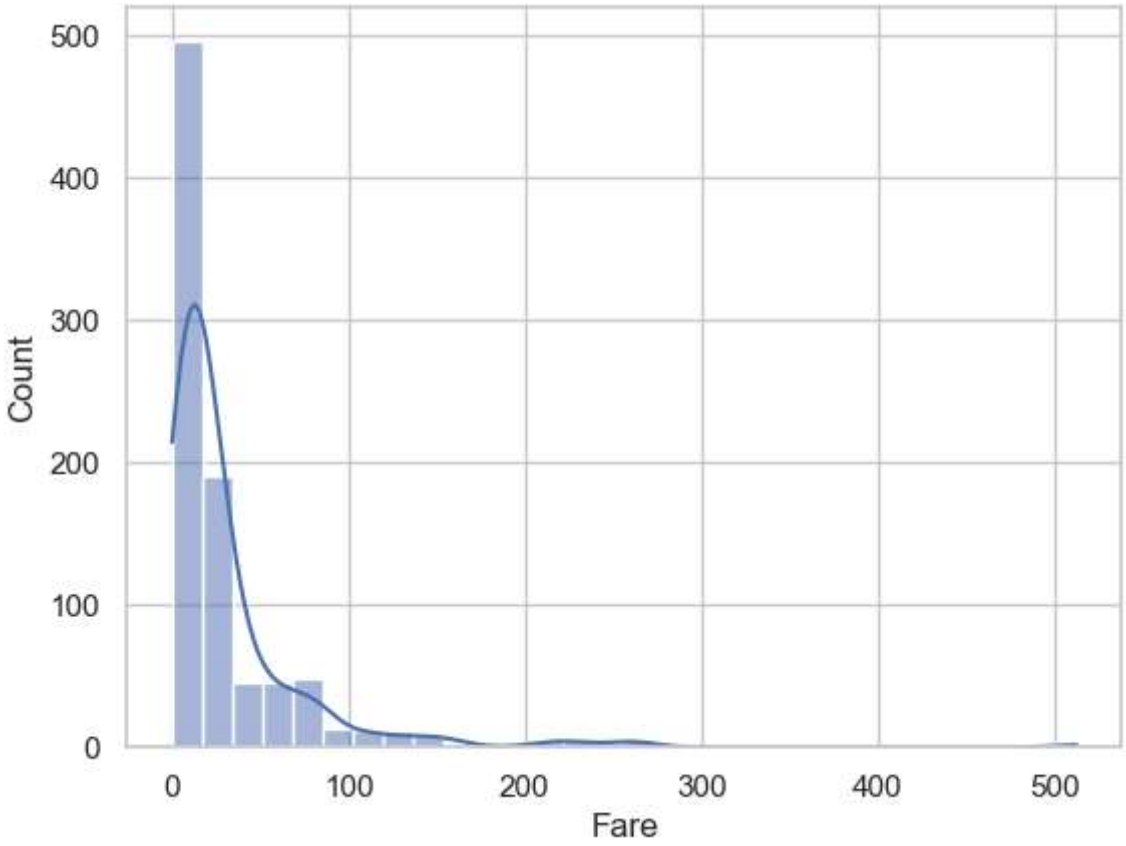
Out[12]:
```
<Axes: xlabel='Age', ylabel='Count'>
```




What fare ranges were most common. If most people paid low or high fares. Any outliers (e.g., a few people paid very high fares).

In [15]:
```
sns.histplot(data=df, x='Fare', bins=30, kde=True)
```

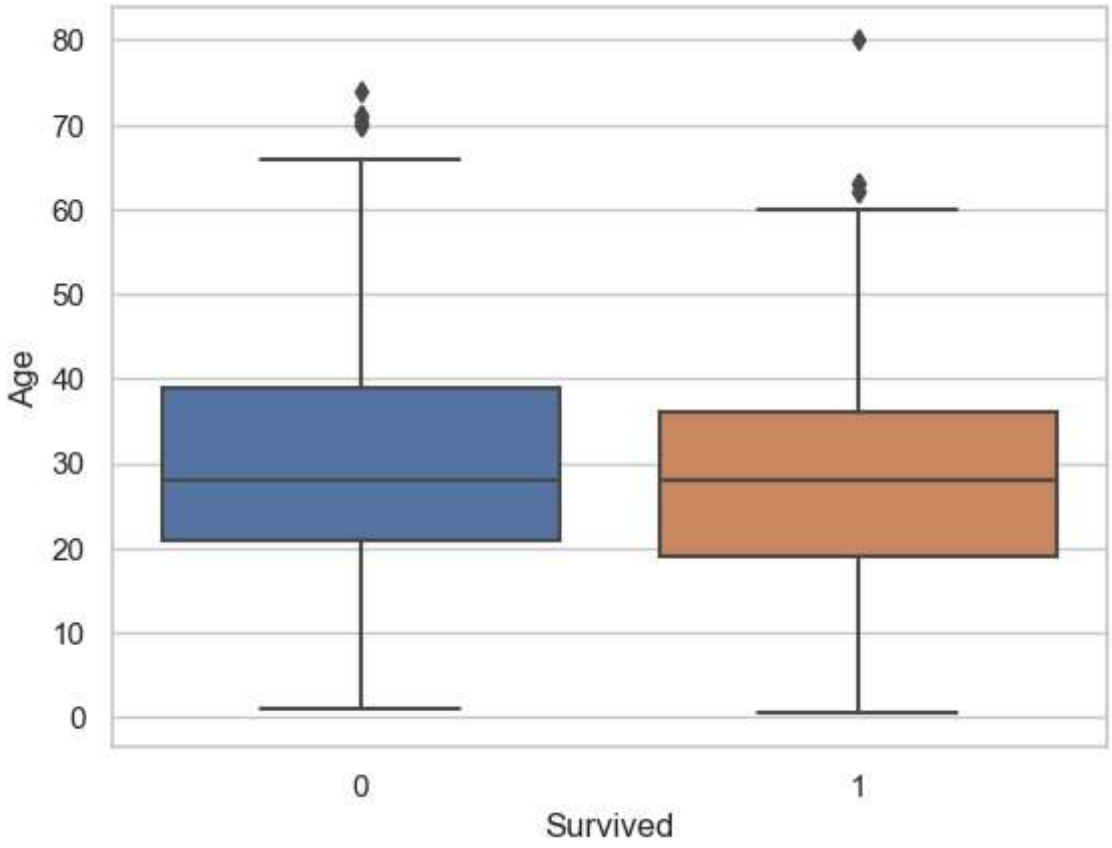
Out[15]:
```
<Axes: xlabel='Fare', ylabel='Count'>
```




Survivors have a slightly lower median age Some very young children (outliers) among the survivors

In [16]:
```
sns.boxplot(data=df, x='Survived', y='Age')
```

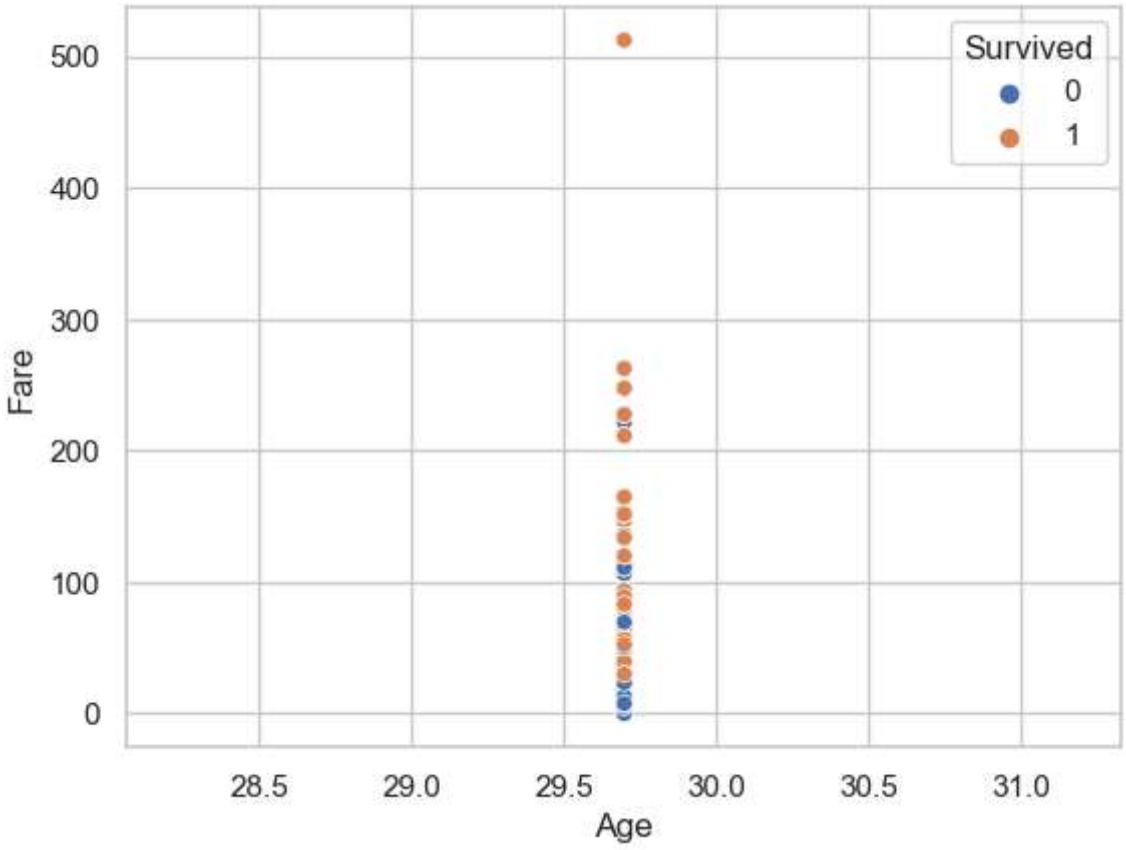
Out[16]:
```
<Axes: xlabel='Survived', ylabel='Age'>
```




Survivors often paid higher fares (likely 1st class) Young children who survived may stand out as outliers

In [87]:
```
sns.scatterplot(data=df, x='Age', y='Fare', hue='Survived')
```

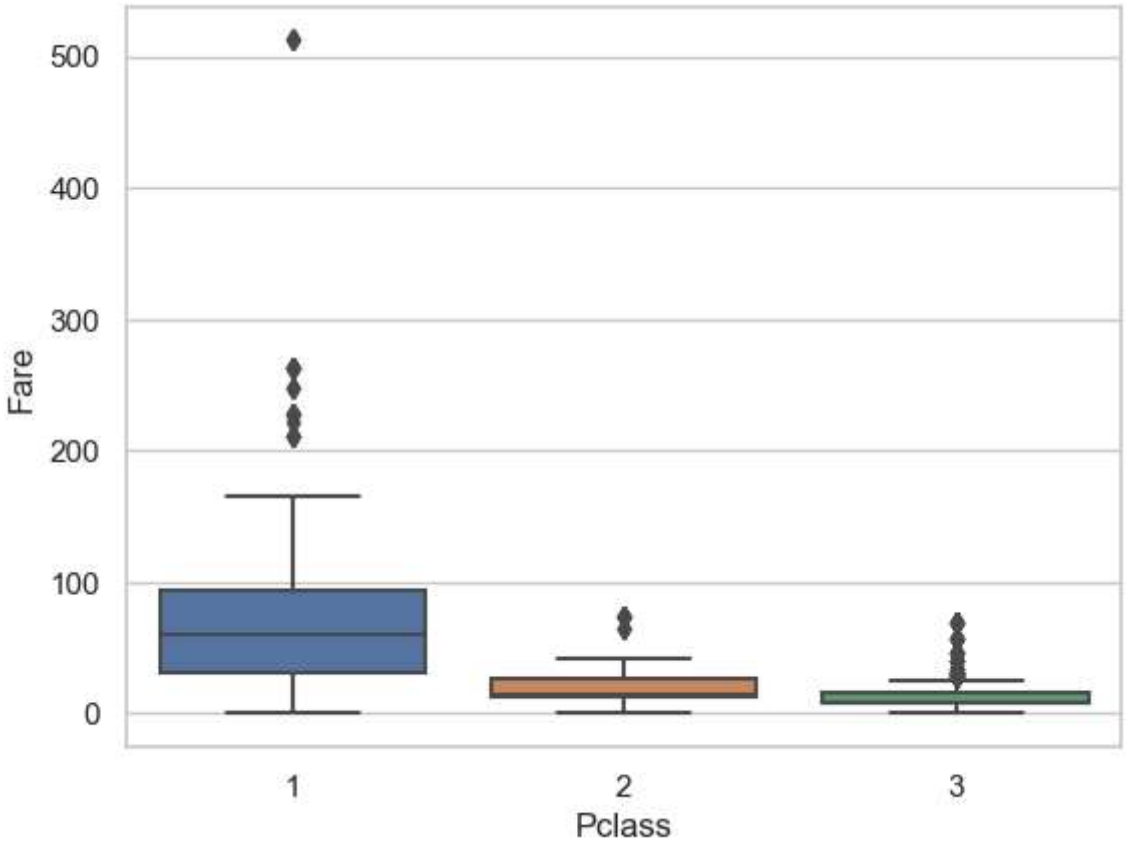
Out[87]:
```
<Axes: xlabel='Age', ylabel='Fare'>
```




How fare varies by class: 1st class → highest fares 3rd class → lowest fares Whether there is a wide range of prices within each class If some passengers paid unusually high fares

In [17]:
```
sns.boxplot(data=df, x='Pclass', y='Fare')
```

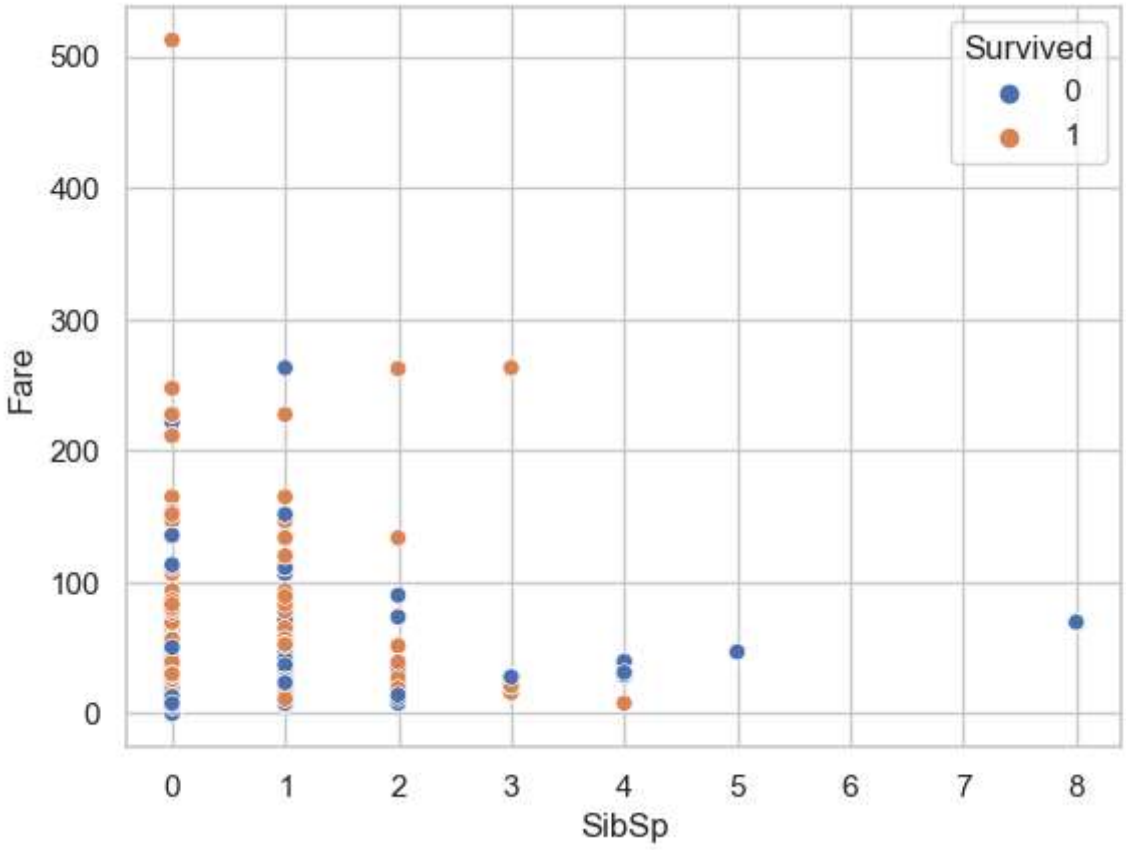
Out[17]:
```
<Axes: xlabel='Pclass', ylabel='Fare'>
```




Creates a scatter plot showing the relationship between: SibSp (number of siblings/spouses on board) Fare (ticket price) Color (hue='Survived') shows whether the passenger survived or not

In [88]:
```
sns.scatterplot(data=df, x='SibSp', y='Fare', hue='Survived')
```

Out[88]:
```
<Axes: xlabel='SibSp', ylabel='Fare'>
```




In [ ]:
```
1.Gender and Survival
Female passengers had a significantly higher survival rate than males.
This reflects the women and children first evacuation policy.
2.Passenger Class and Survival
Passengers in 1st class had the highest survival rates, followed by 2nd and 3rd cla
Lower-class passengers were less likely to survive, possibly due to restricted acce
3.Age and Survival
Children and young adults had higher survival rates.
The median age of survivors was lower compared to non-survivors.
4.Fare and Survival
Higher fares were associated with higher survival, again linking class and wealth t
Fare distribution is heavily right-skewed with a few outliers in 1st class.
5.Embarkation Port and Survival
Most passengers embarked from port 'S' (Southampton).
Passengers from port 'C' (Cherbourg) had better survival rates – likely due to clas
6.Cabin and Deck Information
Most cabin data was missing, but for available data, decks B and D had higher survi
These decks typically housed 1st-class passengers.
7.Family Size Impact
Passengers with small family sizes (2–4 members) had higher survival rates.
Solo travelers and large families were less likely to survive.
8.Missing Data Handling
Missing values in Embarked and Cabin were filled with 'Unknown'.
Deck was extracted from Cabin and used to explore survival by location
```