Task 5: Exploratory Data Analysis (EDA)

Objective : Extract insights using visual and statistical exploration.

Import Libraries

```
# For data handling and analysis
import pandas as pd

# For data visualization
import matplotlib.pyplot as plt
import seaborn as sns

# Set Seaborn style for plots
sns.set(style="whitegrid")
```

Load the Dataset

```
# Load train.csv file
df = pd.read_csv("/content/train.csv")

# Show the first few rows to get a preview
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

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

Understand the Dataset

```
# Basic structure of data: columns, non-null values, and data types
df.info()

# Summary statistics for numeric columns
df.describe()

# Total number of rows and columns
df.shape

# Show column names
df.columns

# Count of unique values in each column
df.nunique()
```

```
<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
```

	0
PassengerId	891
Survived	2
Pclass	3
Name	891
Sex	2
Age	88
SibSp	7
Parch	7
Ticket	681
Fare	248
Cabin	147
Embarked	3

dtype: int64

Missing Value Detection

```
# Total missing values in each column
df.isnull().sum()

# Percentage of missing values per column
(df.isnull().sum() / len(df)) * 100
```

	0
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

Handle Missing Values (Basic Cleaning)

```
df.describe()
```

	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

```
# Fill missing Age values with median
df['Age'].fillna(df['Age'].median(), inplace=True)

# Fill missing Embarked values with the mode
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)

# Drop the 'Cabin' column due to too many missing values
df.drop('Cabin', axis=1, inplace=True)
```

```
<ipython-input-19-ee3788848f25>:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on wh

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


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

Check again for null values

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

	0
PassengerId	0
Survived	0
Pclass	0
Name	0
Sex	0
Age	0
SibSp	0
Parch	0
Ticket	0
Fare	0
Embarked	0

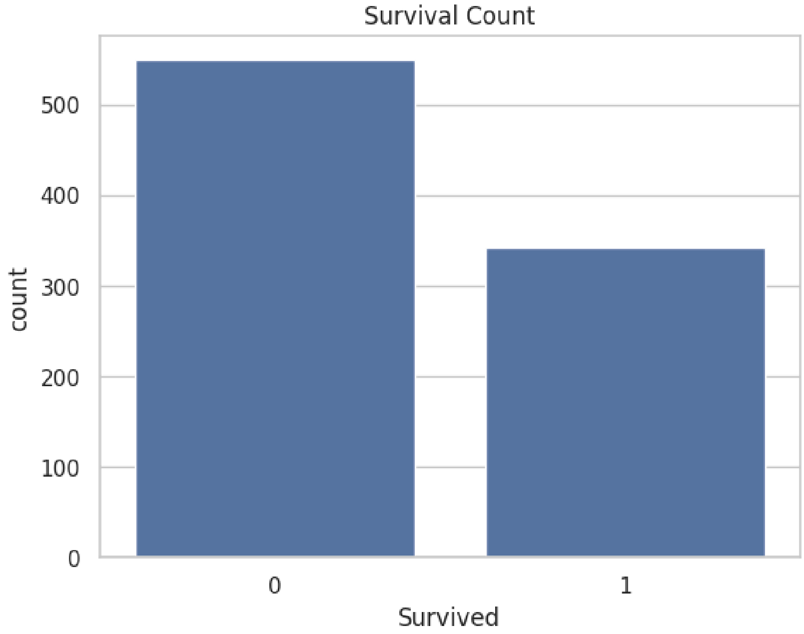
dtype: int64

Target Variable Analysis (Survived)

```
# Count values in Survived column
df['Survived'].value_counts()
```

```
# Plot count of survival
sns.countplot(x='Survived', data=df)
plt.title("Survival Count")
plt.show()

# Percentage of survived
df['Survived'].value_counts(normalize=True) * 100
```



Survived	proportion
0	61.616162
1	38.383838

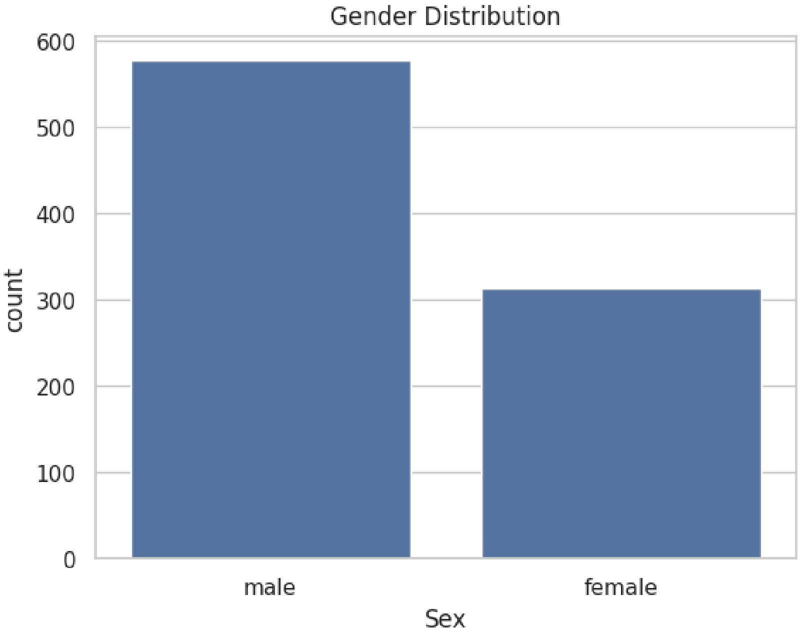
dtype: float64

Observation:

- Around 550 passengers did not survive.
- About 340 passengers survived.
- Majority of passengers died (approx. 62%).

Univariate Analysis (Single Variable)

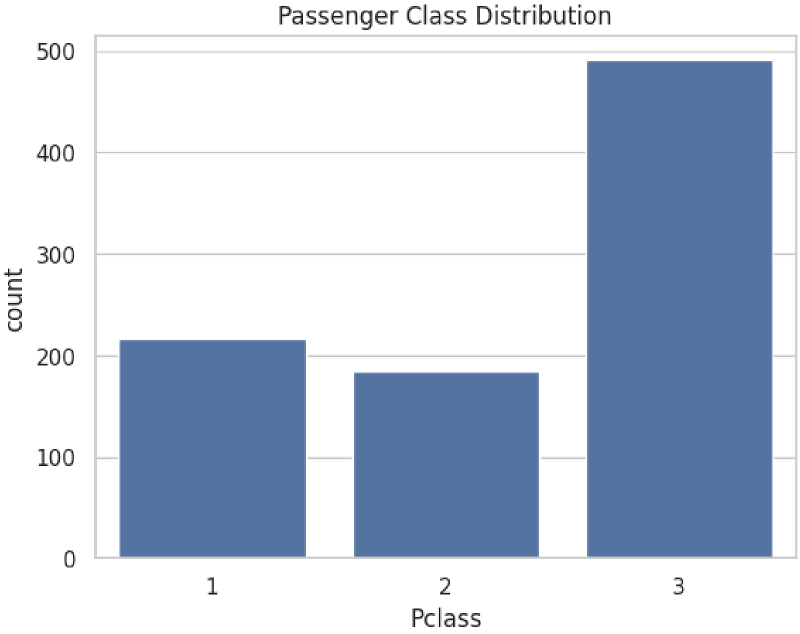
```
# Gender distribution
sns.countplot(x='Sex', data=df)
plt.title("Gender Distribution")
plt.show()
```



Observation:

- There were more male passengers than female passengers.

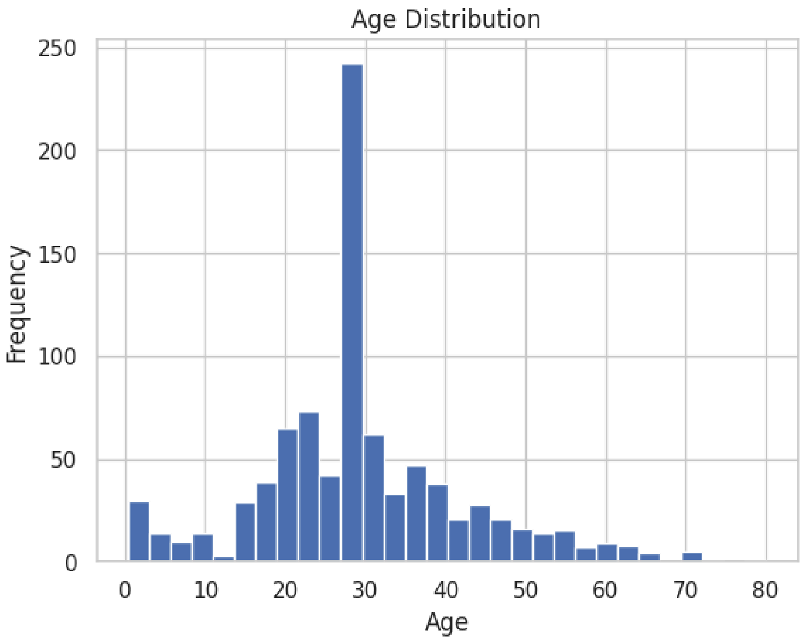
```
# Passenger class distribution
sns.countplot(x='Pclass', data=df)
plt.title("Passenger Class Distribution")
plt.show()
```



Observation:

- Most passengers traveled in 3rd class.

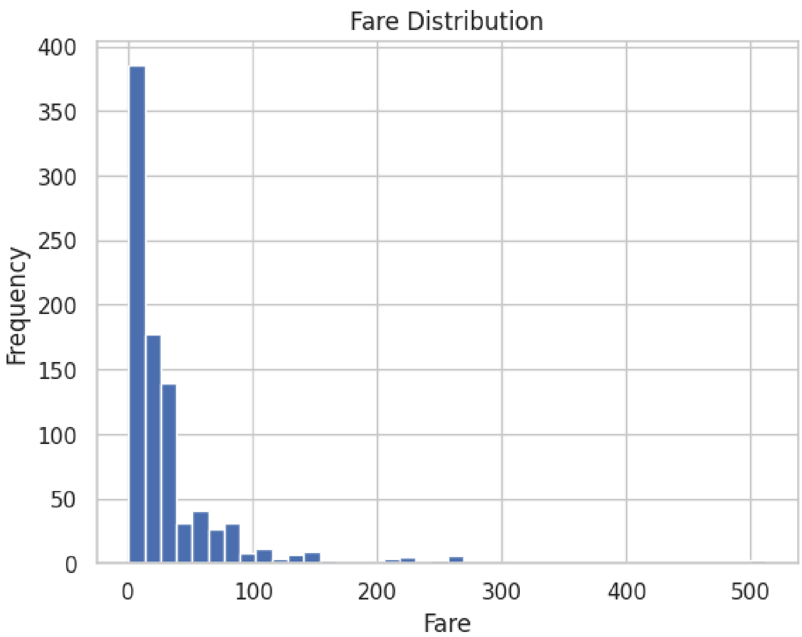
```
# Age distribution
df['Age'].hist(bins=30)
plt.title("Age Distribution")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.show()
```



Observation:

- Majority of passengers were aged 20 to 40 years.

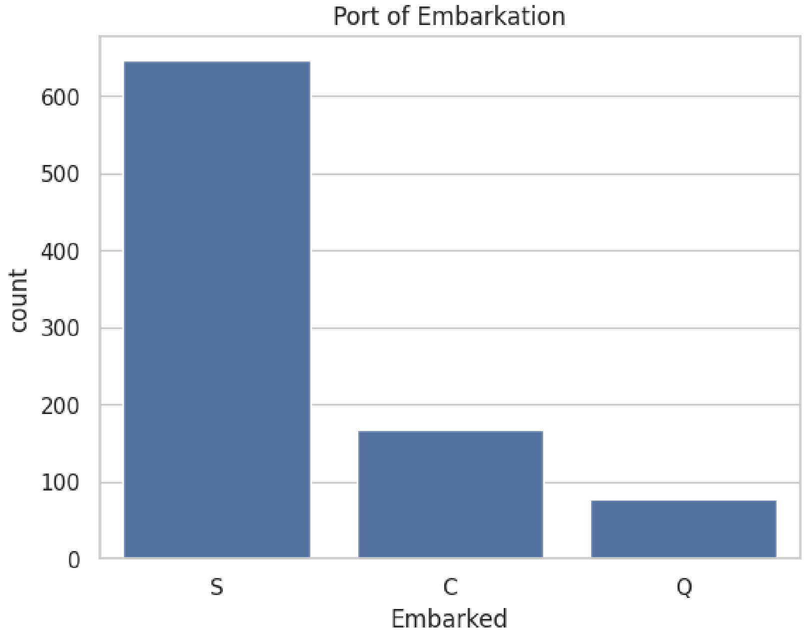
```
# Fare distribution
df['Fare'].hist(bins=40)
plt.title("Fare Distribution")
plt.xlabel("Fare")
plt.ylabel("Frequency")
plt.show()
```



Observation:

- Most passengers paid less than $100 for their ticket.

```
# Embarked distribution
sns.countplot(x='Embarked', data=df)
plt.title("Port of Embarkation")
plt.show()
```



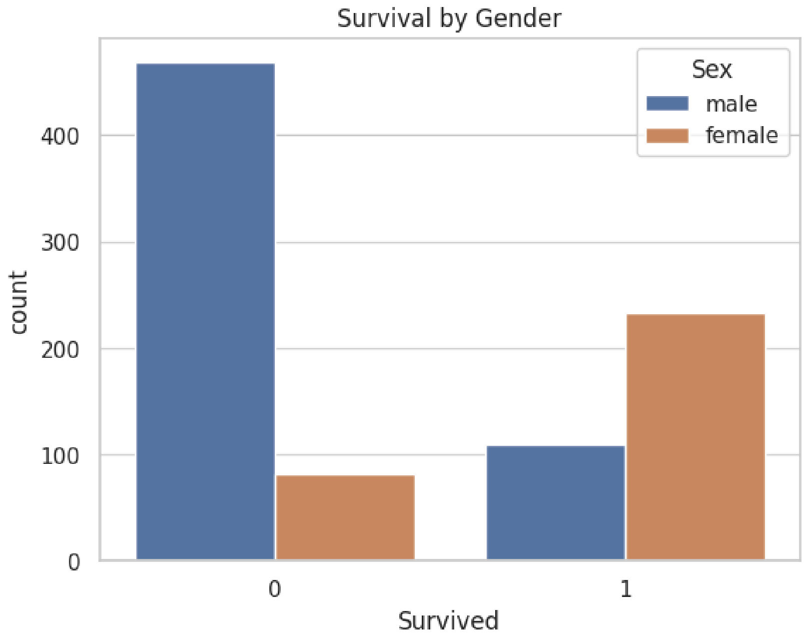
Observation:

- This can suggest that Southampton was a major departure port for the Titanic.

Bivariate Analysis (Two Variables Together)

Survival by Gender

```
sns.countplot(x='Survived', hue='Sex', data=df)
plt.title("Survival by Gender")
plt.show()
```

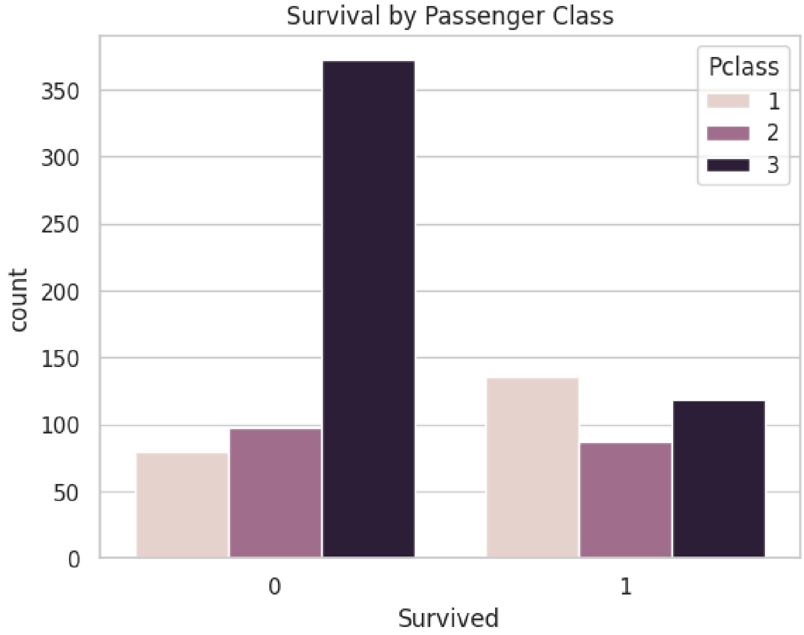


Observation:

- Females had a much higher survival rate than males.

Survival by Passenger Class

```
sns.countplot(x='Survived', hue='Pclass', data=df)
plt.title("Survival by Passenger Class")
plt.show()
```

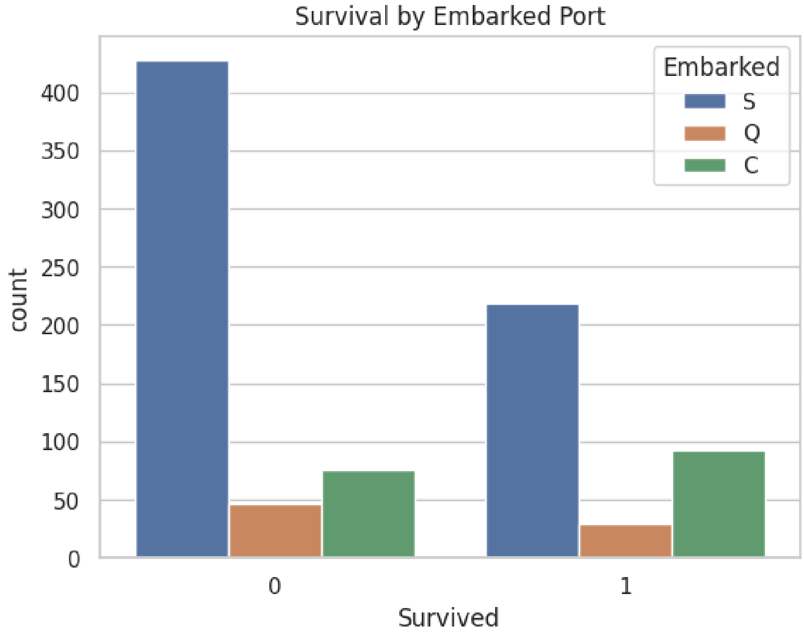


Observation:

- 1st class passengers were most likely to survive.

Survival by Embarked

```
sns.countplot(x='Survived', hue='Embarked', data=df)
plt.title("Survival by Embarked Port")
plt.show()
```



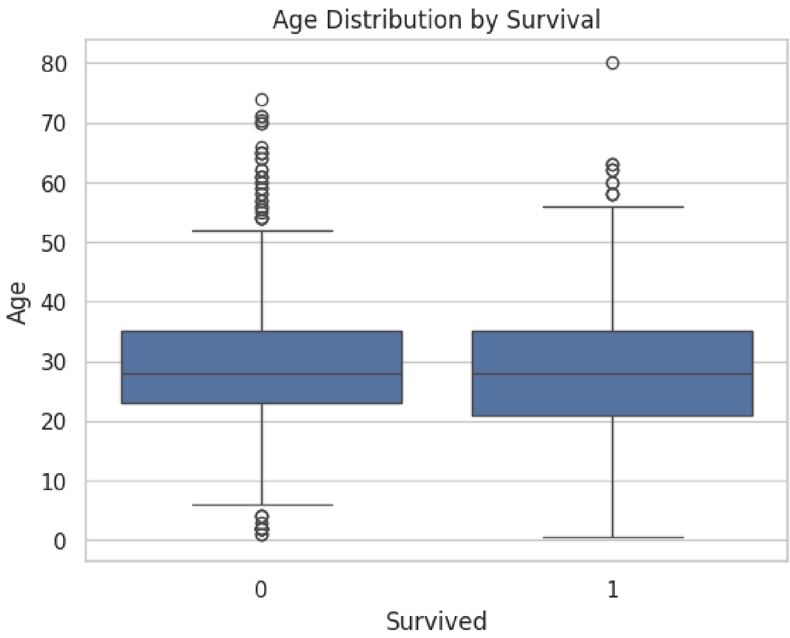
Observation:

- Passengers from Cherbourg (C) had the highest survival rate proportionally.
- Southampton (S) had the highest number of passengers, but many of them did not survive.
- Passengers from Queenstown (Q) were fewer and had a low survival count.

Boxplots (Distribution based on Survival)

Age vs Survival

```
sns.boxplot(x='Survived', y='Age', data=df)
plt.title("Age Distribution by Survival")
plt.show()
```

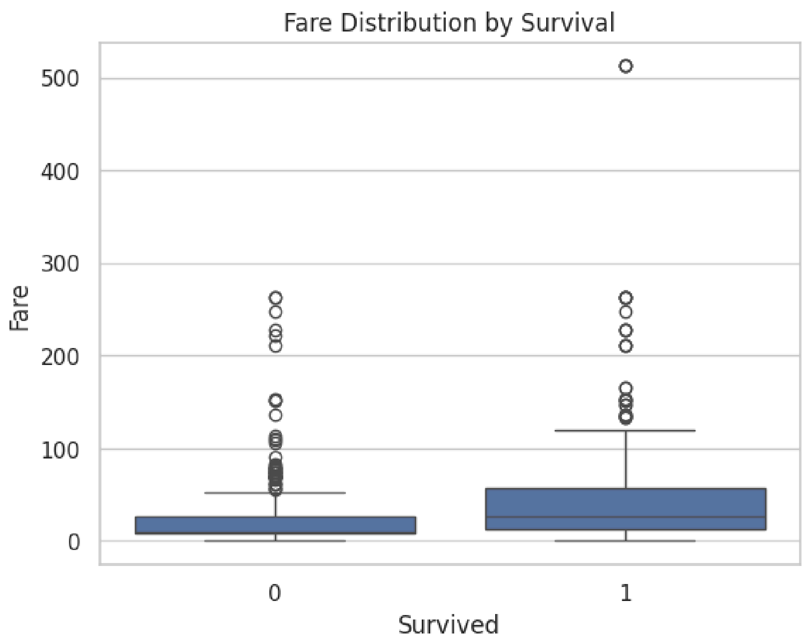


Observation:

- Younger passengers had slightly better survival chances.

Fare vs Survival

```
sns.boxplot(x='Survived', y='Fare', data=df)
plt.title("Fare Distribution by Survival")
plt.show()
```



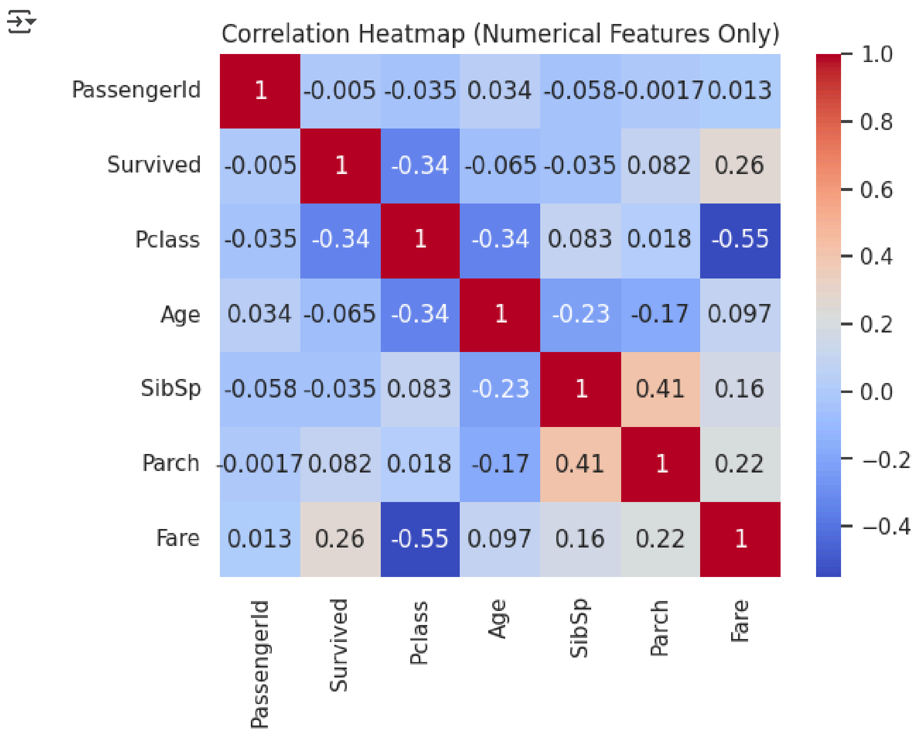
Observation:

- Those who paid higher fares had better survival rates.

Correlation Analysis

```
# This will automatically exclude non-numeric columns
correlation = df.corr(numeric_only=True)

# Heatmap
sns.heatmap(correlation, annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap (Numerical Features Only)")
plt.show()
```

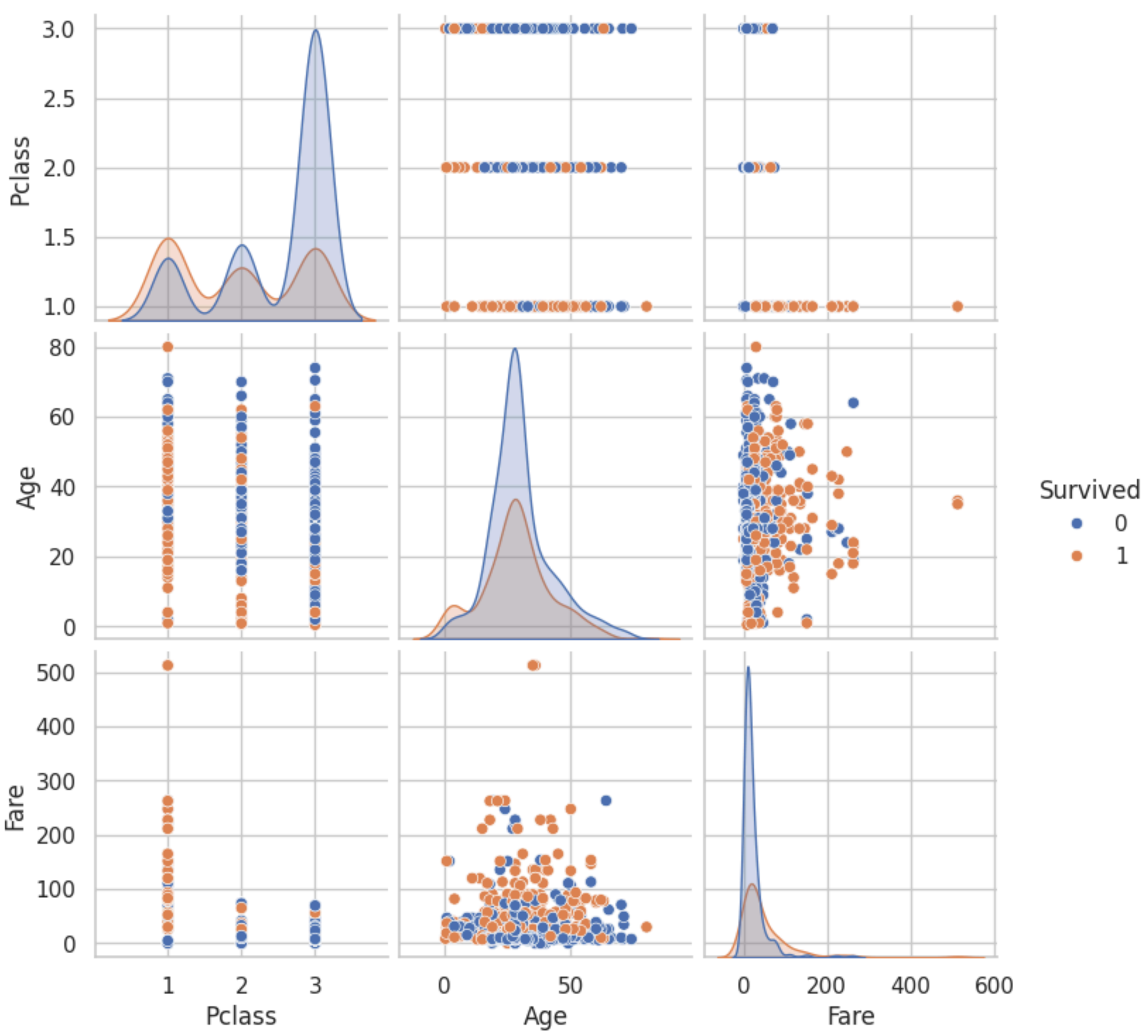


Observation:

- Strong negative correlation between Pclass and Survived
- Positive correlation between Fare and Survived
- Sex (encoded later) will also show strong correlation with survival

Pairplot (Explore Relationships)

```
sns.pairplot(df[['Survived', 'Pclass', 'Age', 'Fare']], hue='Survived')
plt.show()
```



Observation:

- Clear visual patterns: survival more common among higher fare and 1st class passengers.

Value Counts for Categorical Variables

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

Sex	count
male	577
female	314

dtype: int64

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

Pclass	count
3	491
1	216
2	184

dtype: int64

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

	count
Embarked	
S	646
C	168
Q	77

dtype: int64

GroupBy Aggregations

```
# Average survival by Sex
df.groupby('Sex')['Survived'].mean()
```

	Survived
Sex	
female	0.742038
male	0.188908

dtype: float64

```
# Average survival by Pclass
df.groupby('Pclass')['Survived'].mean()
```

	Survived
Pclass	
1	0.629630
2	0.472826