```
Start coding or generate with AI.
```

End-to-End Exploratory Data Analysis (EDA) on the Titanic Dataset

Project Objective: To perform a comprehensive, step-by-step exploratory data analysis to understand the key factors that influenced survival on the Titanic. This notebook will serve as a complete guide, covering data loading, cleaning, analysis, feature engineering, and visualization.

Double-click (or enter) to edit

Step 1: Setup - Importing Libraries

Start by importing the essential Python libraries for data manipulation (`pandas`, `numpy`) and visualization (`matplotlib`, `seaborn`).

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Set plot style for better aesthetics
sns.set(style='whitegrid')
```

Step 2: Data Loading and Initial Inspection -load the dataset and take our first look at its structure, content, and overall health.

```
!git clone 'https://github.com/GeeksforgeeksDS/21-Days-21-Projects-Dataset'
```

```
Cloning into '21-Days-21-Projects-Dataset'...
remote: Enumerating objects: 22, done.
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 22 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (22/22), 1.40 MiB | 5.59 MiB/s, done.
Resolving deltas: 100% (3/3), done.
```

Locally: To import dataset locally: `df = pd.read_csv('Titanic-Dataset.csv')`

```
#Using Google Colab
df = pd.read_csv('/content/21-Days-21-Projects-Dataset/Datasets/Titanic-Dataset.csv')
df
```

	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
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S

Next steps: Generate code with df New interactive sheet

```
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

Next steps: Generate code with df New interactive sheet

```
df.tail()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C

```
df.shape
```

```
(891, 12)
```

```
# Get a concise summary of the dataframe
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
```

Interpretation of `.info()`:

- The dataset contains 891 entries (passengers) and 12 columns.
- **Missing Values Identified:** `Age`, `Cabin`, and `Embarked` have missing values. `Cabin` is missing a significant amount of data (~77%), which will require special attention.

```
# Get descriptive statistics for numerical columns
print("\nDescriptive Statistics:")
df.describe()
```

```
Descriptive Statistics:
```

	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

Interpretation of `.describe()`:

- **Survived:** About 38.4% of passengers in this dataset survived.
- **Age:** The age ranges from ~5 months to 80 years, with an average age of about 30.
- **Fare:** The fare is highly skewed, a mean of 32 dollars but a median of only 14.45 dollars.The maximum fare is over 512 dollars, indicating the presence of extreme outliers.

```
## Filter out the people who were Females and Embarked from Q
df[(df['Sex'] == 'female') & (df['Embarked'] == 'Q')]
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
22	23	1	3	McGowan, Miss. Anna "Annie"	female	15.0	0	0	330923	8.0292	NaN	Q
28	29	1	3	O'Dwyer, Miss. Ellen "Nellie"	female	NaN	0	0	330959	7.8792	NaN	Q
32	33	1	3	Glynn, Miss. Mary Agatha	female	NaN	0	0	335677	7.7500	NaN	Q
44	45	1	3	Devaney, Miss. Margaret Delia	female	19.0	0	0	330958	7.8792	NaN	Q
47	48	1	3	O'Driscoll, Miss. Bridget	female	NaN	0	0	14311	7.7500	NaN	Q
82	83	1	3	McDermott, Miss. Brigdet Delia	female	NaN	0	0	330932	7.7875	NaN	Q
109	110	1	3	Moran, Miss. Bertha	female	NaN	1	0	371110	24.1500	NaN	Q
156	157	1	3	Gilnagh, Miss. Katherine "Katie"	female	16.0	0	0	35851	7.7333	NaN	Q
186	187	1	3	O'Brien, Mrs. Thomas (Johanna "Hannah" Godfrey)	female	NaN	1	0	370365	15.5000	NaN	Q
198	199	1	3	Madigan, Miss. Margaret "Maggie"	female	NaN	0	0	370370	7.7500	NaN	Q
208	209	1	3	Carr, Miss. Helen "Ellen"	female	16.0	0	0	367231	7.7500	NaN	Q
241	242	1	3	Murphy, Miss. Katherine "Kate"	female	NaN	1	0	367230	15.5000	NaN	Q
264	265	0	3	Henry, Miss. Delia	female	NaN	0	0	382649	7.7500	NaN	Q
274	275	1	3	Healy, Miss. Hanora "Nora"	female	NaN	0	0	370375	7.7500	NaN	Q
289	290	1	3	Connolly, Miss. Kate	female	22.0	0	0	370373	7.7500	NaN	Q
300	301	1	3	Kelly, Miss. Anna Katherine "Annie Kate"	female	NaN	0	0	9234	7.7500	NaN	Q
303	304	1	2	Keane, Miss. Nora A	female	NaN	0	0	226593	12.3500	E101	Q
322	323	1	2	Slayter, Miss. Hilda Mary	female	30.0	0	0	234818	12.3500	NaN	Q
330	331	1	3	McCoy, Miss. Agnes	female	NaN	2	0	367226	23.2500	NaN	Q
358	359	1	3	McGovern, Miss. Mary	female	NaN	0	0	330931	7.8792	NaN	Q
359	360	1	3	Mockler, Miss. Helen Mary "Ellie"	female	NaN	0	0	330980	7.8792	NaN	Q
368	369	1	3	Jermyn, Miss. Annie	female	NaN	0	0	14313	7.7500	NaN	Q
412	413	1	1	Minahan, Miss. Daisy E	female	33.0	1	0	19928	90.0000	C78	Q
501	502	0	3	Canavan, Miss. Mary	female	21.0	0	0	364846	7.7500	NaN	Q
502	503	0	3	O'Sullivan, Miss. Bridget Mary	female	NaN	0	0	330909	7.6292	NaN	Q
573	574	1	3	Kelly, Miss. Mary	female	NaN	0	0	14312	7.7500	NaN	Q
593	594	0	3	Bourke, Miss. Mary	female	NaN	0	2	364848	7.7500	NaN	Q
612	613	1	3	Murphy, Miss. Margaret Jane	female	NaN	1	0	367230	15.5000	NaN	Q
653	654	1	3	O'Leary, Miss. Hanora	female	NaN	0	0	330919	7.8292	NaN	Q

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

	count
Cabin	
G6	4
C23 C25 C27	4
B96 B98	4
F2	3
D	3
...	...
E17	1
A24	1
C50	1
B42	1
C148	1

147 rows × 1 columns

dtype: int64

Step 3: Data Cleaning

```
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

```
# 1. Handle missing 'Age' values
# We use the median to fill missing ages because the age distribution can be skewed.
median = df['Age'].median()
print(median)
median_age = df['Age'].median()
df['Age'] = df['Age'].fillna(median_age)
```

```
28.0
```

```
# 2. Handle missing 'Embarked' values
# Since there are only two missing values, we'll fill them with the most common port of embarkation (the mode).
df['Embarked'].mode()[0]
mode_embarked = df['Embarked'].mode()[0]
print(mode_embarked)
df['Embarked'] = df['Embarked'].fillna(mode_embarked)
```

```
S
```

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

	0
PassengerId	0
Survived	0
Pclass	0
Name	0
Sex	0
Age	0
SibSp	0
Parch	0
Ticket	0
Fare	0
Cabin	687
Embarked	0

dtype: int64

```
# 3. Handle the 'Cabin' column
# With over 77% missing data, imputing is not a good idea. Instead, we'll create a new feature 'Has_Cabin'.
df['Has_Cabin'] = df['Cabin'].notna().astype(int) # 1 if has cabin, 0 if not
df.drop('Cabin', axis=1, inplace=True) # Drop the original column
```

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

	count
Has_Cabin	
0	687
1	204

dtype: int64

```
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
Has_Cabin	0

dtype: int64

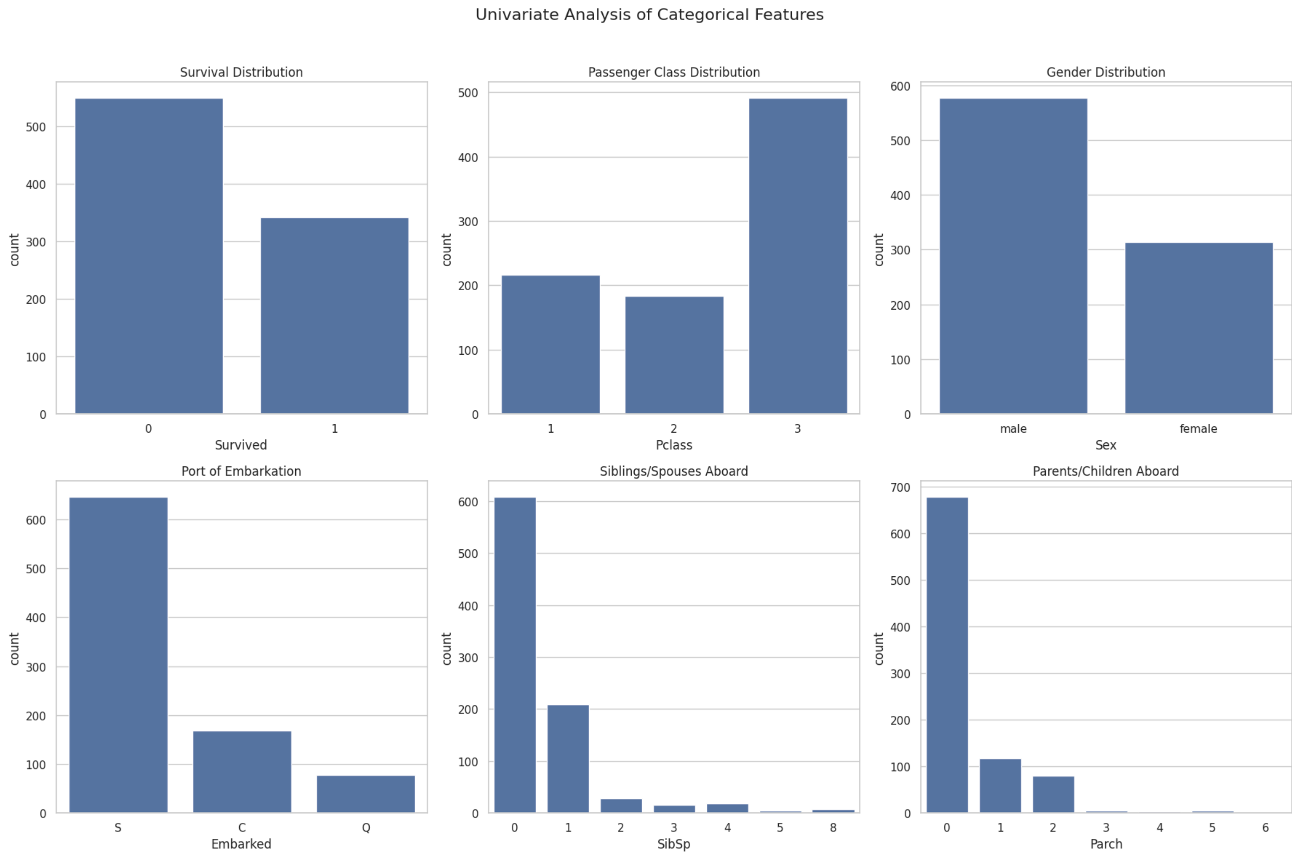
Step 4: Univariate Analysis

Analyze each variable individually to understand its distribution.

```
print("Analyzing categorical features:")

# Set up the figure for plotting
fig, axes = plt.subplots(2, 3, figsize=(18, 12))
fig.suptitle('Univariate Analysis of Categorical Features', fontsize=16)
```

```
# Plotting each categorical feature
sns.countplot(ax=axes[0, 0], x='Survived', data=df).set_title('Survival Distribution')
sns.countplot(ax=axes[0, 1], x='Pclass', data=df).set_title('Passenger Class Distribution')
sns.countplot(ax=axes[0, 2], x='Sex', data=df).set_title('Gender Distribution')
sns.countplot(ax=axes[1, 0], x='Embarked', data=df).set_title('Port of Embarkation')
sns.countplot(ax=axes[1, 1], x='SibSp', data=df).set_title('Siblings/Spouses Aboard')
sns.countplot(ax=axes[1, 2], x='Parch', data=df).set_title('Parents/Children Aboard')

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```

```
Analyzing categorical features:
```

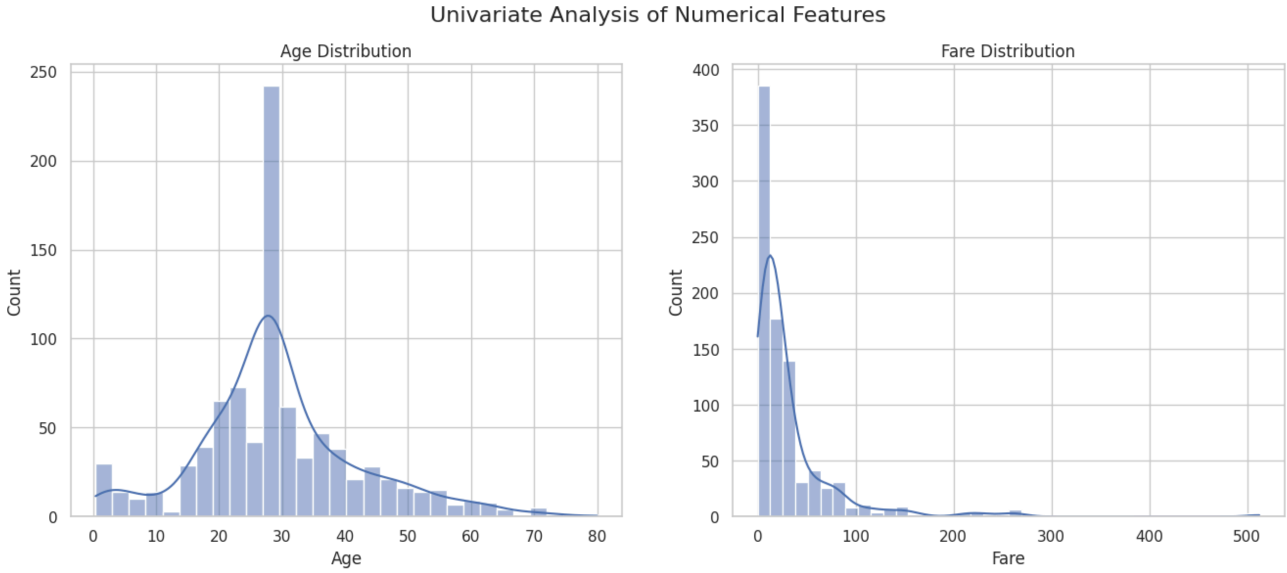


Key Insights (Categorical):

- **Survival:** Most passengers (over 500) did not survive.
- **Pclass:** The 3rd class was the most populated, followed by 1st and then 2nd.
- **Sex:** There were significantly more males than females.
- **Embarked:** The vast majority of passengers embarked from Southampton ('S').
- **SibSp & Parch:** Most passengers traveled alone.

```
print("\nAnalyzing numerical features:")

fig, axes = plt.subplots(1, 2, figsize=(16, 6))
fig.suptitle('Univariate Analysis of Numerical Features', fontsize=16)

# Plotting Age distribution
sns.histplot(ax=axes[0], data=df, x='Age', kde=True, bins=30).set_title('Age Distribution')

# Plotting Fare distribution
sns.histplot(ax=axes[1], data=df, x='Fare', kde=True, bins=40).set_title('Fare Distribution')

plt.show()
```

```
Analyzing numerical features:
```



Key Insights (Numerical):

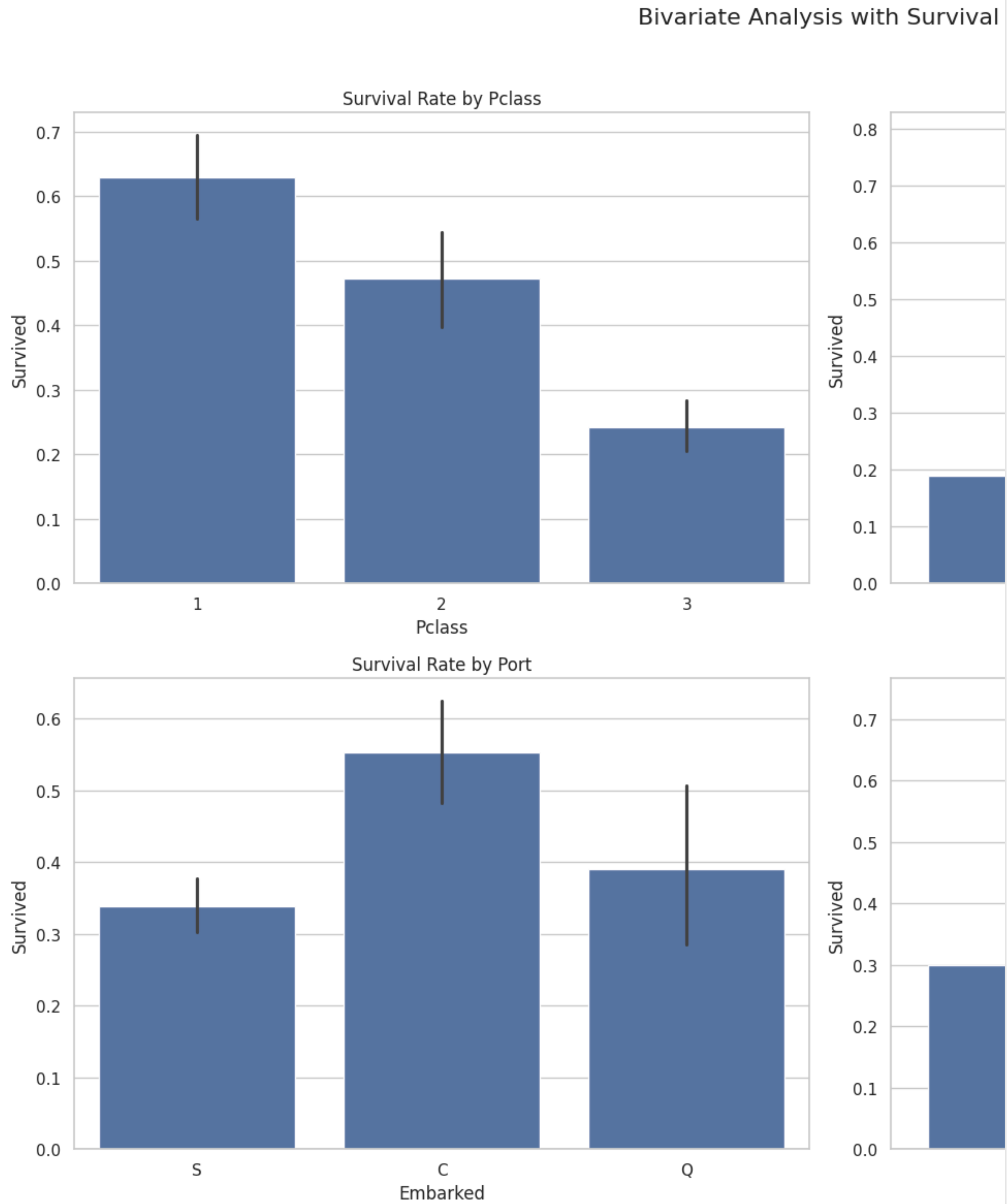
- **Age:** The distribution peaks around the 20-30 age range. Remember we filled missing values with the median (28), which contributes to the height of that central bar.
- **Fare:** The distribution is heavily right-skewed, confirming that most tickets were cheap, with a few very expensive exceptions.

Step 5: Bivariate Analysis

Here, we explore the relationship between two variables. Our primary focus will be on how each feature relates to our target variable, `Survived`.

```
print("Bivariate Analysis: Feature vs. Survival")

fig, axes = plt.subplots(2, 2, figsize=(16, 12))
fig.suptitle('Bivariate Analysis with Survival', fontsize=16)

# Pclass vs. Survived
sns.barplot(ax=axes[0, 0], x='Pclass', y='Survived', data=df).set_title('Survival Rate by Pclass')

# Sex vs. Survived
sns.barplot(ax=axes[0, 1], x='Sex', y='Survived', data=df).set_title('Survival Rate by Sex')

# Embarked vs. Survived
sns.barplot(ax=axes[1, 0], x='Embarked', y='Survived', data=df).set_title('Survival Rate by Port')

# Has_Cabin vs. Survived
sns.barplot(ax=axes[1, 1], x='Has_Cabin', y='Survived', data=df).set_title('Survival Rate by Cabin Availability')

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```

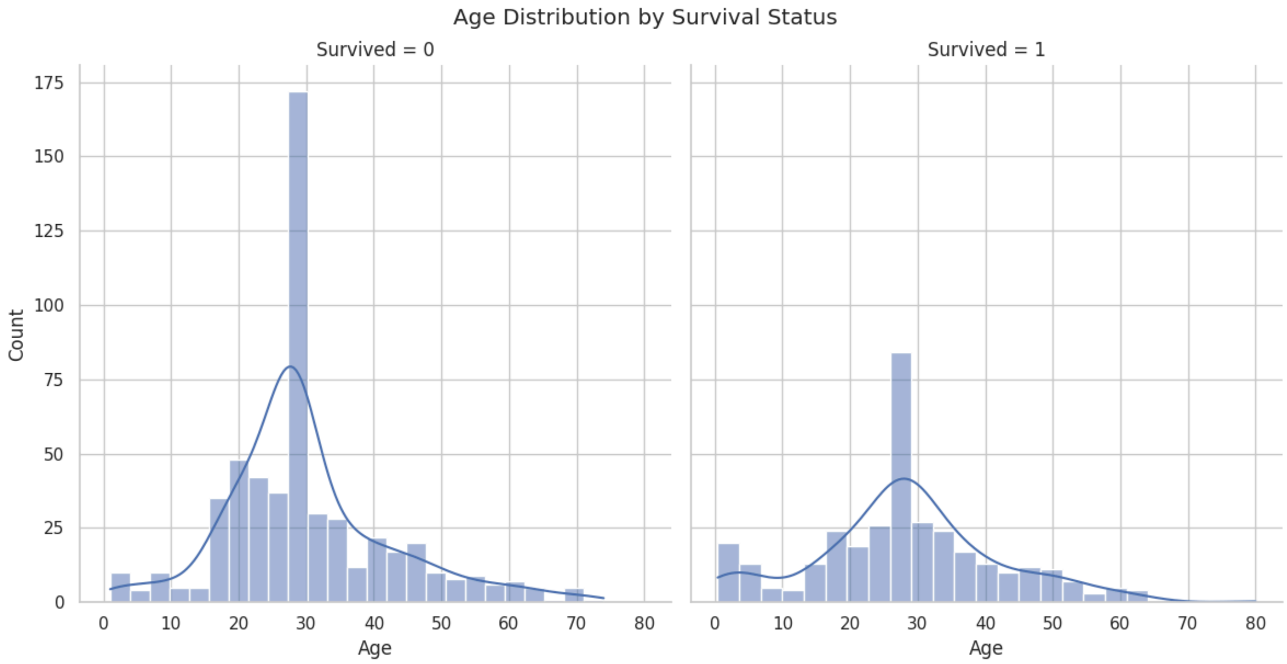
```
Bivariate Analysis: Feature vs. Survival
```



Key Insights (Bivariate):

- **Pclass:** A clear trend emerges: 1st class passengers had a >60% survival rate, while 3rd class passengers had less than 25%.
- **Sex:** This is the strongest predictor. Females had a survival rate of ~75%, while males had a rate below 20%.
- **Embarked:** Passengers embarking from Cherbourg ('C') had a higher survival rate than those from the other ports.
- **Has_Cabin:** Passengers with a registered cabin number had a much higher survival rate. This is likely correlated with being in 1st class.

```
# Age vs. Survival
g = sns.FacetGrid(df, col='Survived', height=6)
g.map(sns.histplot, 'Age', bins=25, kde=True)
plt.suptitle('Age Distribution by Survival Status', y=1.02)
plt.show()
```



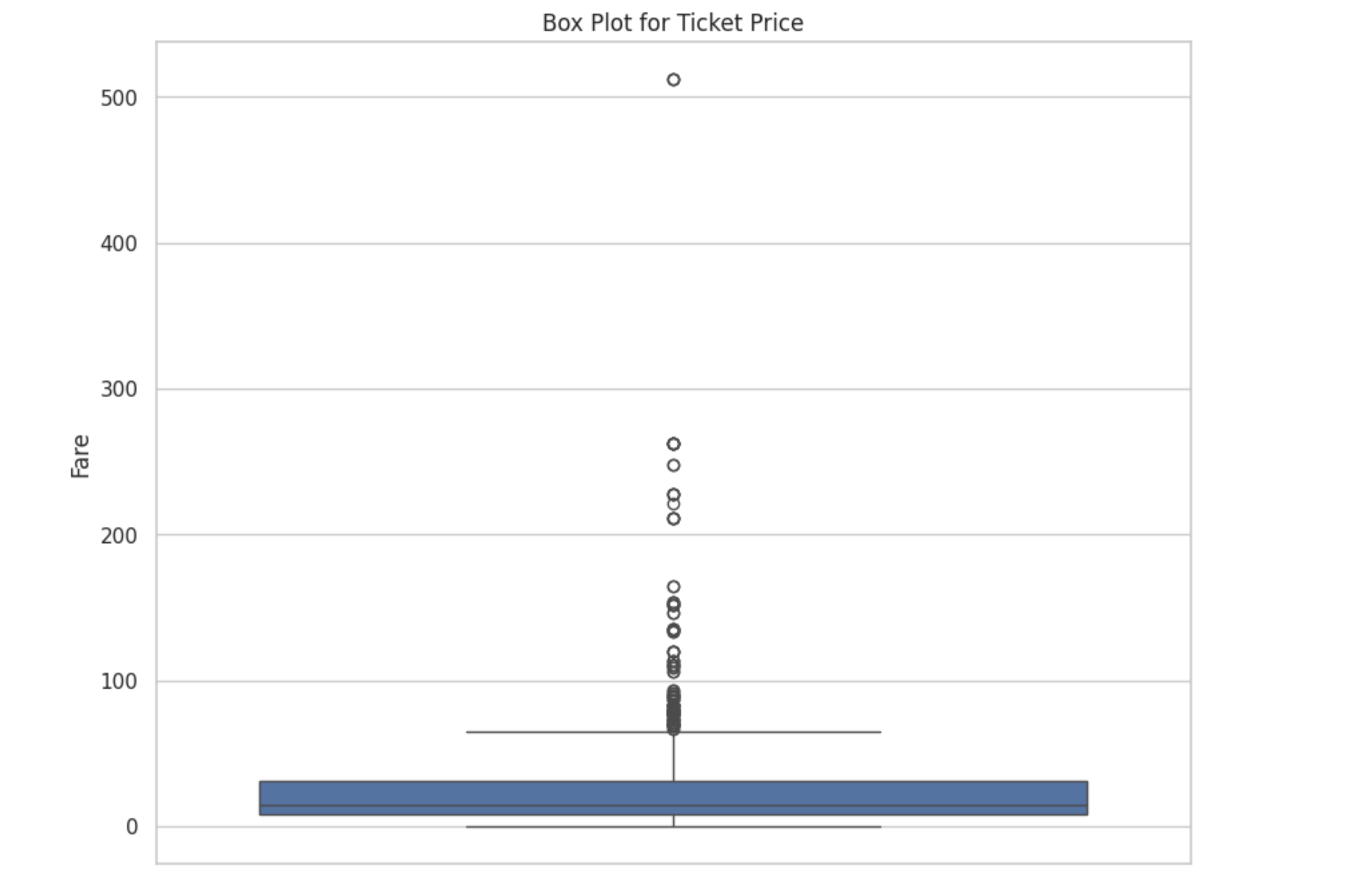
Key Insight (Age vs. Survival):

- Infants and young children had a higher probability of survival.
- A large portion of non-survivors were young adults (20-40).
- The oldest passengers (80 years) did not survive.

Deeper Dive: Outlier Analysis for 'Fare'

The `.describe()` function and histogram showed that `Fare` has extreme outliers. Let's visualize this clearly with a box plot.

```
plt.figure(figsize=(10,8))
sns.boxplot(y='Fare', data=df)
plt.title('Box Plot for Ticket Price')
plt.ylabel("Fare")
plt.show()
```



Observation: The box plot confirms the presence of significant outliers. Most fares are concentrated below $100, but there are several fares extending far beyond, with some even exceeding $500. These are likely first-class passengers who booked luxurious suites. For some machine learning models, handling these outliers (e.g., through log transformation) would be an important step.

Step 6: Feature Engineering

Creating new features from the existing ones to potentially uncover deeper insights and provide more useful information for a machine learning model.

Common Techniques:

1. **Combining Features:** Creating a new feature by combining others (e.g., `SibSp` + `Parch` = `FamilySize`).
2. **Extracting from Text:** Pulling out specific information from a text feature (e.g., extracting titles from the `Name` column).
3. **Binning:** Converting a continuous numerical feature into a categorical one (e.g., binning `Age` into groups like 'Child', 'Adult', 'Senior').

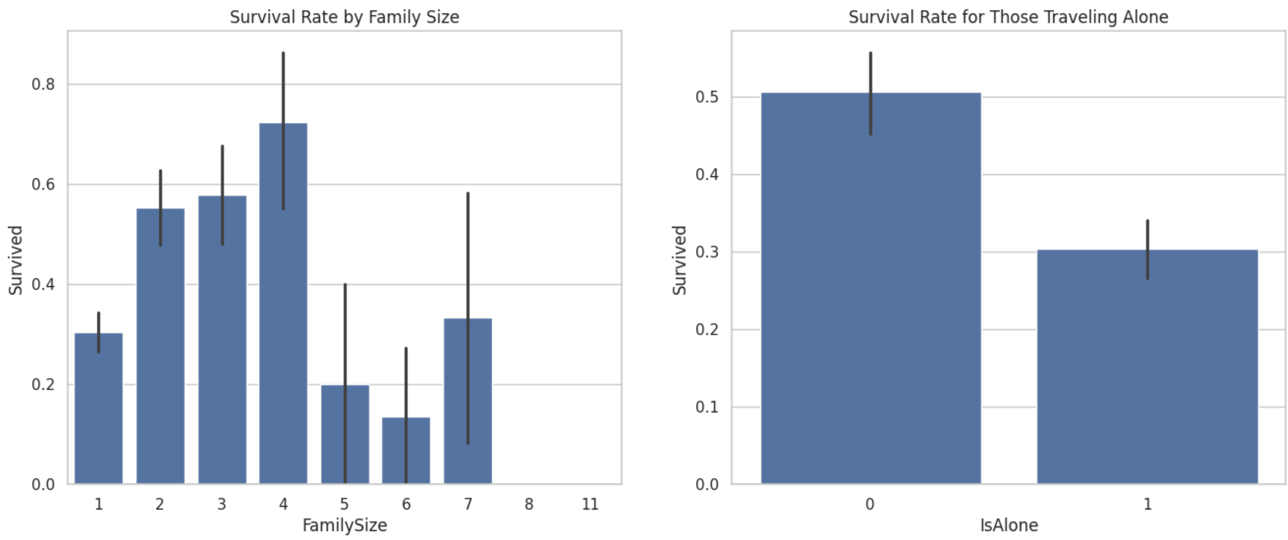
```
## Create a "familySIZE" column
df['FamilySize'] = df['SibSp'] + df['Parch'] + 1

# 2. Create an 'IsAlone' feature
df['IsAlone'] = 0
df.loc[df['FamilySize'] == 1, 'IsAlone'] = 1

print("Created 'FamilySize' and 'IsAlone' features:")
df[['FamilySize', 'IsAlone']].head()
```

```
Created 'FamilySize' and 'IsAlone' features:
```

	FamilySize	IsAlone
0	2	0
1	2	0
2	1	1
3	2	0
4	1	1

```
# Analyze the new family-related features against survival
fig, axes = plt.subplots(1, 2, figsize=(16, 6))
```

```
# Survival Rate by FamilySize
sns.barplot(ax=axes[0], x='FamilySize', y='Survived', data=df).set_title('Survival Rate by Family Size')

# Survival Rate by IsAlone
sns.barplot(ax=axes[1], x='IsAlone', y='Survived', data=df).set_title('Survival Rate for Those Traveling Alone')

plt.show()
```



Insight:

- Passengers who were alone (`IsAlone=1`) had a lower survival rate (~30%) than those in small families.
- Small families of 2 to 4 members had the highest survival rates.
- Very large families (5 or more) had a very poor survival rate. This might be because it was harder for large families to stay together and evacuate.

```
# 3. Extract 'Title' from the 'Name' column
df['Title'] = df['Name'].str.extract(r' ([A-Za-z]+)\.', expand=False)

# Let's see the different titles
print("Extracted Titles:")
df['Title'].value_counts()
```

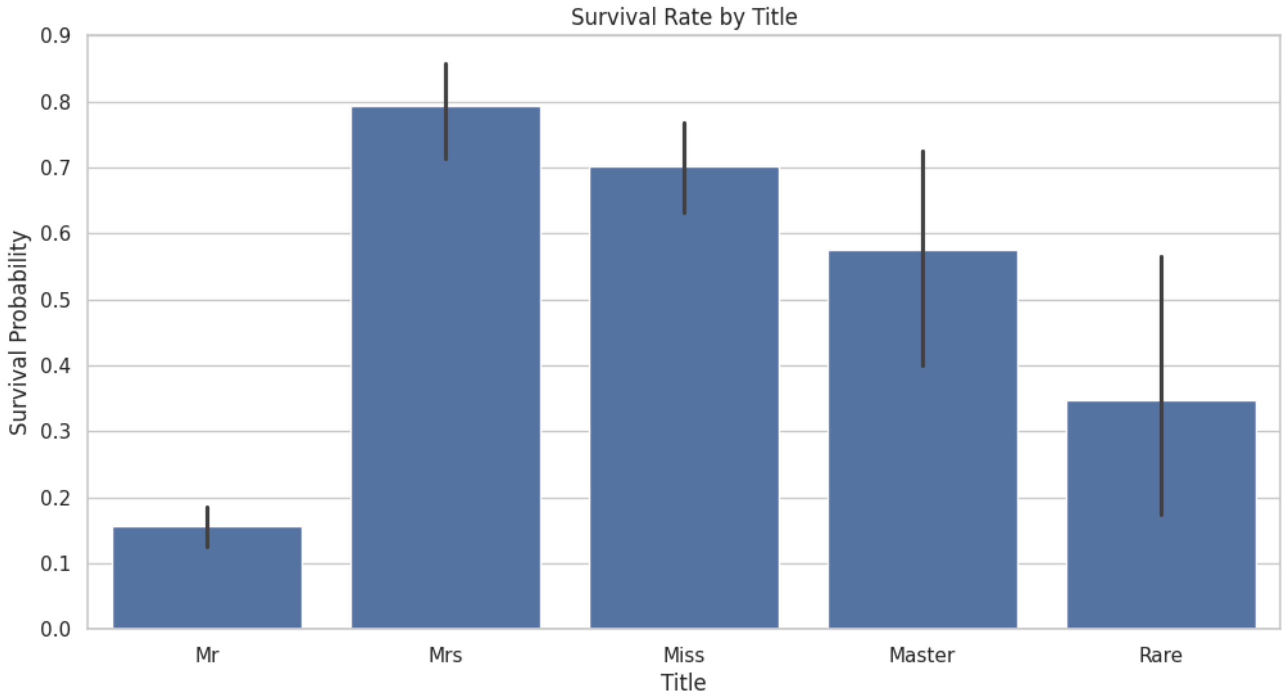
Extracted Titles:

Title	count
Mr	517
Miss	182
Mrs	125
Master	40
Dr	7
Rev	6
Col	2
Mlle	2
Major	2
Ms	1
Mme	1
Don	1
Lady	1
Sir	1
Capt	1
Countess	1
Jonkheer	1

dtype: int64

- Matches a space.
- Titles in the names are usually preceded by a space. ([A-Za-z]+): This is the capturing group.
- [A-Za-z]+: Matches one or more uppercase or lowercase letters. This captures the title itself (like Mr, Mrs, Miss, etc.).
- .: Matches a literal dot (.) which usually follows the title.

```
# Simplify the titles by grouping rare ones into a 'Rare' category
df['Title'] = df['Title'].replace(['Lady', 'Countess','Capt', 'Col','Don', 'Dr', 'Major', 'Rev', 'Sir', 'Jonkheer', 'Dona'],

df['Title'] = df['Title'].replace('Mlle', 'Miss')
df['Title'] = df['Title'].replace('Ms', 'Miss')
df['Title'] = df['Title'].replace('Mme', 'Mrs')

# Let's see the survival rate by the new, cleaned titles
plt.figure(figsize=(12, 6))
sns.barplot(x='Title', y='Survived', data=df)
plt.title('Survival Rate by Title')
plt.ylabel('Survival Probability')
plt.show()
```



Insight: The Title feature gives us powerful information. 'Mrs' and 'Miss' (females) had high survival rates. 'Mr' (males) had a very low survival rate. 'Master' (young boys) had a significantly higher survival rate than 'Mr', reinforcing the 'children first' idea. The 'Rare' titles, often associated with nobility or status, also had a mixed but generally higher survival rate than common men

Step 7: Multivariate Analysis

Exploring interactions between multiple variables simultaneously, including our new engineered features.

```
# Survival rate by Pclass and Sex
sns.catplot(x='Pclass', y='Survived', hue='Sex', data=df, kind='bar', height=6, aspect=1.5)
plt.title('Survival Rate by Pclass and Sex')
plt.ylabel('Survival Probability')
plt.show()

# Insights: Females in all classes had a significantly higher survival rate than males.
```

```
# Violin plot to see age distribution by sex and survival status
plt.figure(figsize=(14, 8))
sns.violinplot(x='Sex', y='Age', hue='Survived', data=df, split=True, palette={0: 'blue', 1: 'orange'})
plt.title('Age Distribution by Sex and Survival')
plt.show()
```