ql8rmz3bn

December 14, 2023

Task -2

Perform data cleaning and exploratory data analysis (EDA) on a dataset of your choice, such as the Titanic dataset from Kaggle. Explore the relationships between variables and identify patterns and trends in the data.

Import necessary libraries

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

[ ]:
```
from google.colab import files
raw = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving titanic.csv to titanic.csv
```

[ ]:
```
data = pd.read_csv("titanic.csv")
```

[ ]:
```
df = data.copy()
```

[ ]:
```
df
```

[ ]:
```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           …         …     …
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                                  Name     Sex   Age  SibSp  \
0                              Braund, Mr. Owen Harris    male  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                               Heikkinen, Miss. Laina  female  26.0      0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                             Allen, Mr. William Henry    male  35.0      0
..                                                 …     …   …    …
886                              Montvila, Rev. Juozas    male  27.0      0
887                       Graham, Miss. Margaret Edith  female  19.0      0
888           Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
889                              Behr, Mr. Karl Howell    male  26.0      0
890                                Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket      Fare Cabin Embarked
0        0         A/5 21171   7.2500   NaN        S
1        0          PC 17599  71.2833   C85        C
2        0  STON/O2. 3101282   7.9250   NaN        S
3        0            113803  53.1000  C123        S
4        0            373450   8.0500   NaN        S
..     …               …      …   …      …
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      0            370376   7.7500   NaN        Q

[891 rows x 12 columns]
```

Exploratory data analysis

[ ]:
```
df.shape
```

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

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

[ ]:
```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
```

```
4                            Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

[ ]:

```
df.tail()
```

[ ]:

```
     PassengerId  Survived  Pclass                                      Name  \
886          887         0       2                     Montvila, Rev. Juozas
887          888         1       1              Graham, Miss. Margaret Edith
888          889         0       3  Johnston, Miss. Catherine Helen "Carrie"
889          890         1       1                     Behr, Mr. Karl Howell
890          891         0       3                       Dooley, Mr. Patrick

        Sex   Age  SibSp  Parch      Ticket   Fare Cabin Embarked
886    male  27.0      0      0      211536  13.00   NaN        S
887  female  19.0      0      0      112053  30.00   B42        S
888  female   NaN      1      2  W./C. 6607  23.45   NaN        S
889    male  26.0      0      0      111369  30.00  C148        C
890    male  32.0      0      0      370376   7.75   NaN        Q
```

[ ]:

```
df.columns
```

[ ]:

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

[ ]:

```
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
```

```
[ ]: df.describe().T
```

```
[ ]:              count        mean         std   min       25%       50%    75%  \
     PassengerId  891.0  446.000000  257.353842  1.00  223.5000  446.0000  668.5
     Survived     891.0    0.383838    0.486592  0.00    0.0000    0.0000    1.0
     Pclass       891.0    2.308642    0.836071  1.00    2.0000    3.0000    3.0
     Age          714.0   29.699118   14.526497  0.42   20.1250   28.0000   38.0
     SibSp        891.0    0.523008    1.102743  0.00    0.0000    0.0000    1.0
     Parch        891.0    0.381594    0.806057  0.00    0.0000    0.0000    0.0
     Fare         891.0   32.204208   49.693429  0.00    7.9104   14.4542   31.0

                       max
     PassengerId  891.0000
     Survived       1.0000
     Pclass         3.0000
     Age           80.0000
     SibSp          8.0000
     Parch          6.0000
     Fare         512.3292
```

Data Cleaning

```
[ ]: #check for missing values
     df.isnull().sum()
```

```
[ ]: 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
```

```
[ ]: # Handle missing values (for simplicity, we'll drop the missing values in this␣
      ↪example)
     df1 =df.dropna(subset=['Age','Cabin'])
```

[ ]:
```
# Confirm that there are no missing values anymore
df1.isnull().sum()
```

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

[ ]:
```
df1['Embarked'].fillna(df1['Embarked'].mode()[0], inplace=True)
```

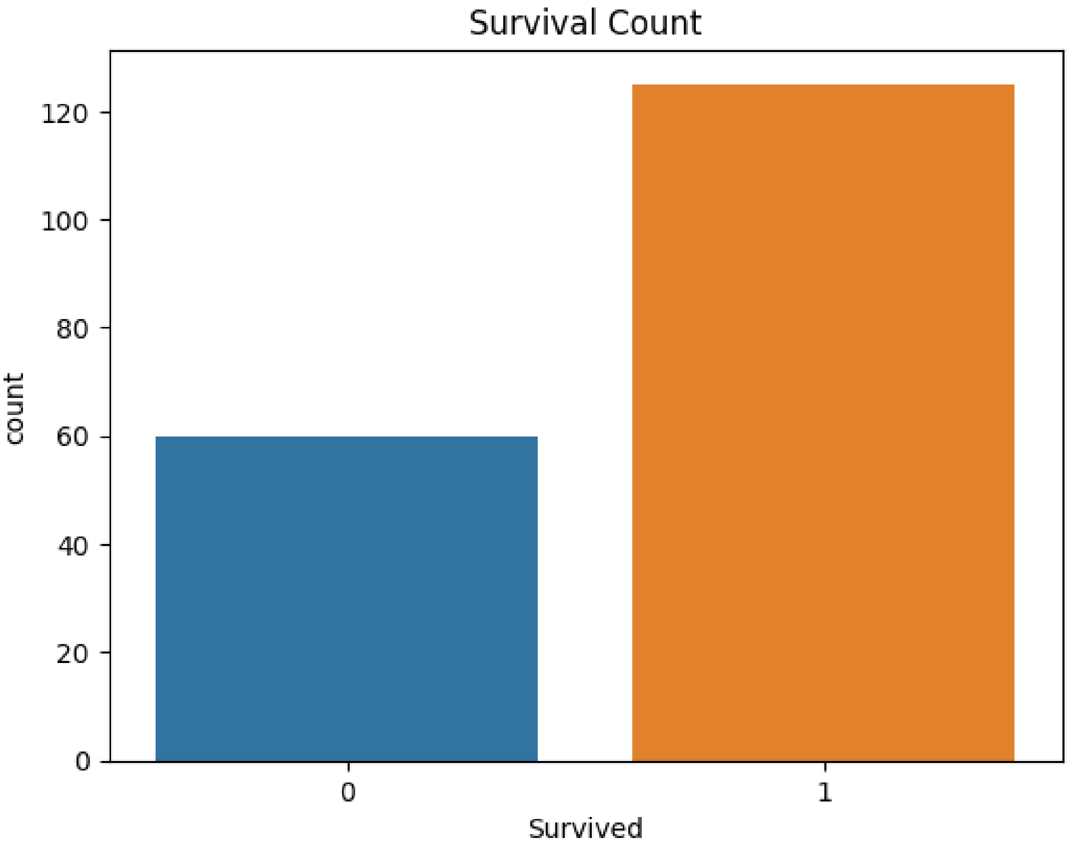
```
<ipython-input-15-96f9b2e3057e>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['Embarked'].fillna(df1['Embarked'].mode()[0], inplace=True)
```

[ ]:
```
df1.shape
```

[ ]:
```
(185, 12)
```

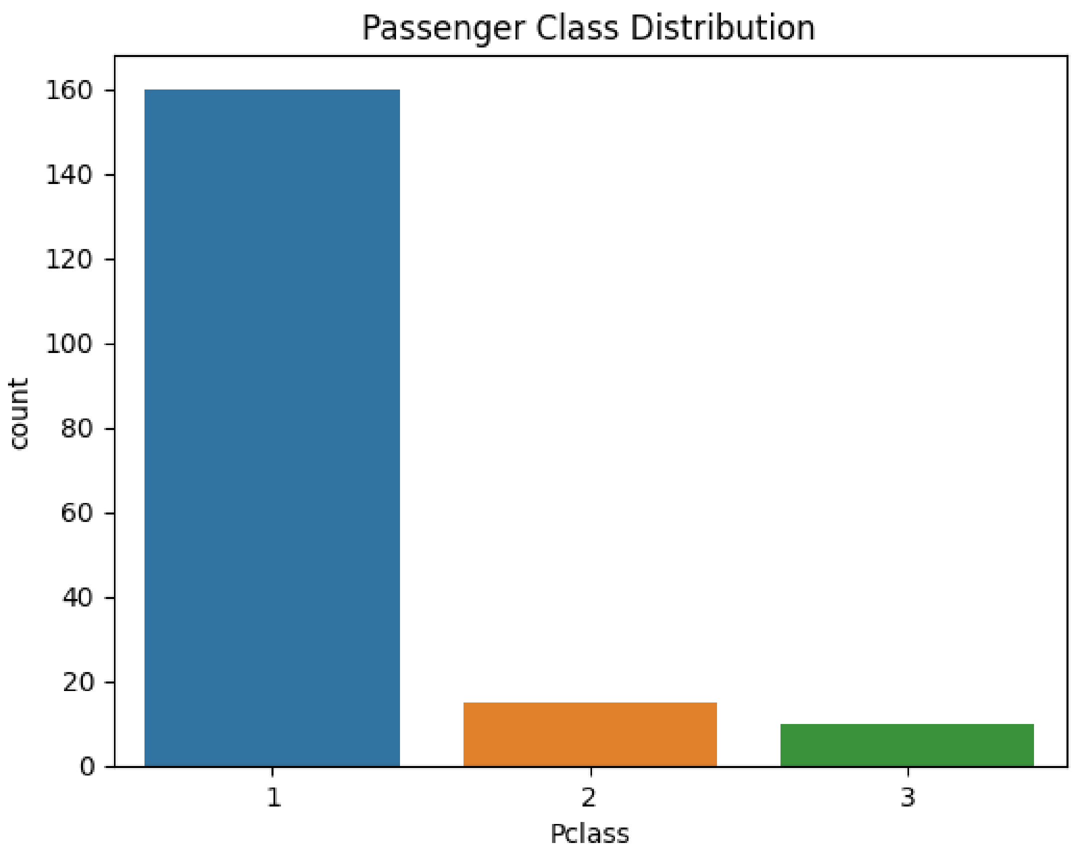
Visualization

[ ]:
```
# Countplot for survival
sns.countplot(x='Survived', data=df1)
plt.title('Survival Count')
plt.show()
```

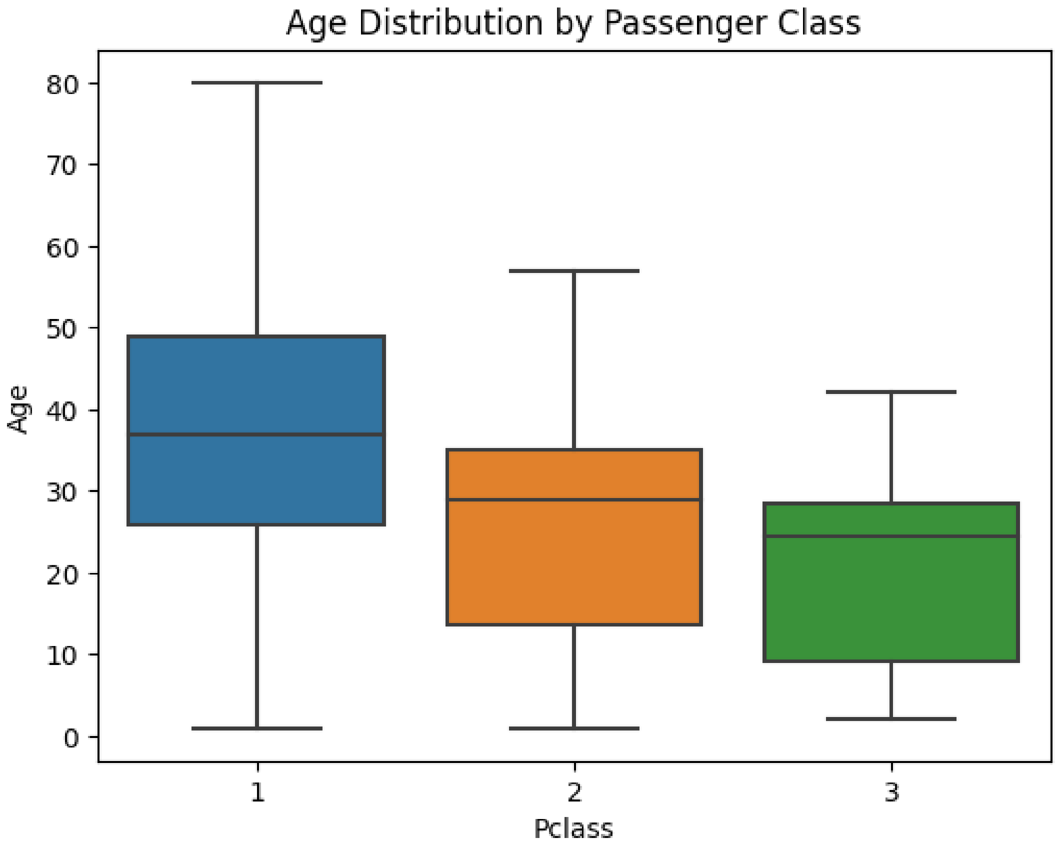


[ ]:

```
# Countplot for Pclass
sns.countplot(x='Pclass', data=df1)
plt.title('Passenger Class Distribution')
plt.show()
```

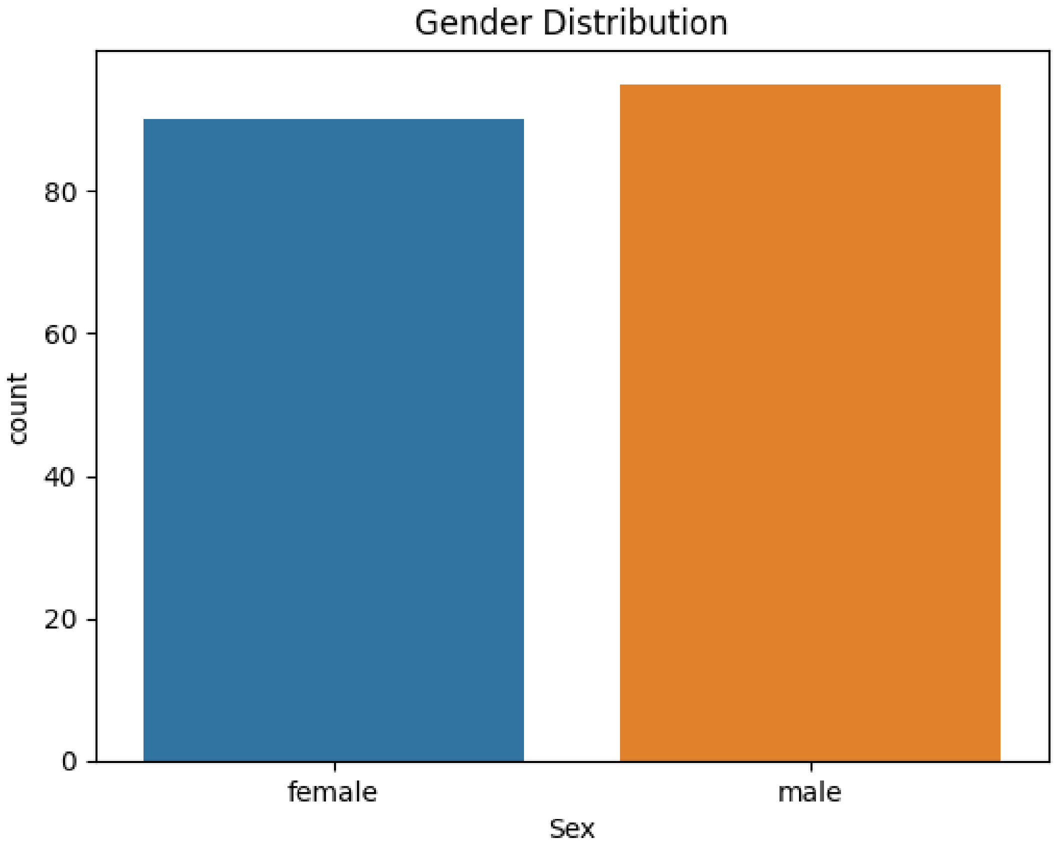


[ ]:
```
# Boxplot for Age distribution by Pclass
sns.boxplot(x='Pclass', y='Age', data=df1)
plt.title('Age Distribution by Passenger Class')
plt.show()
```

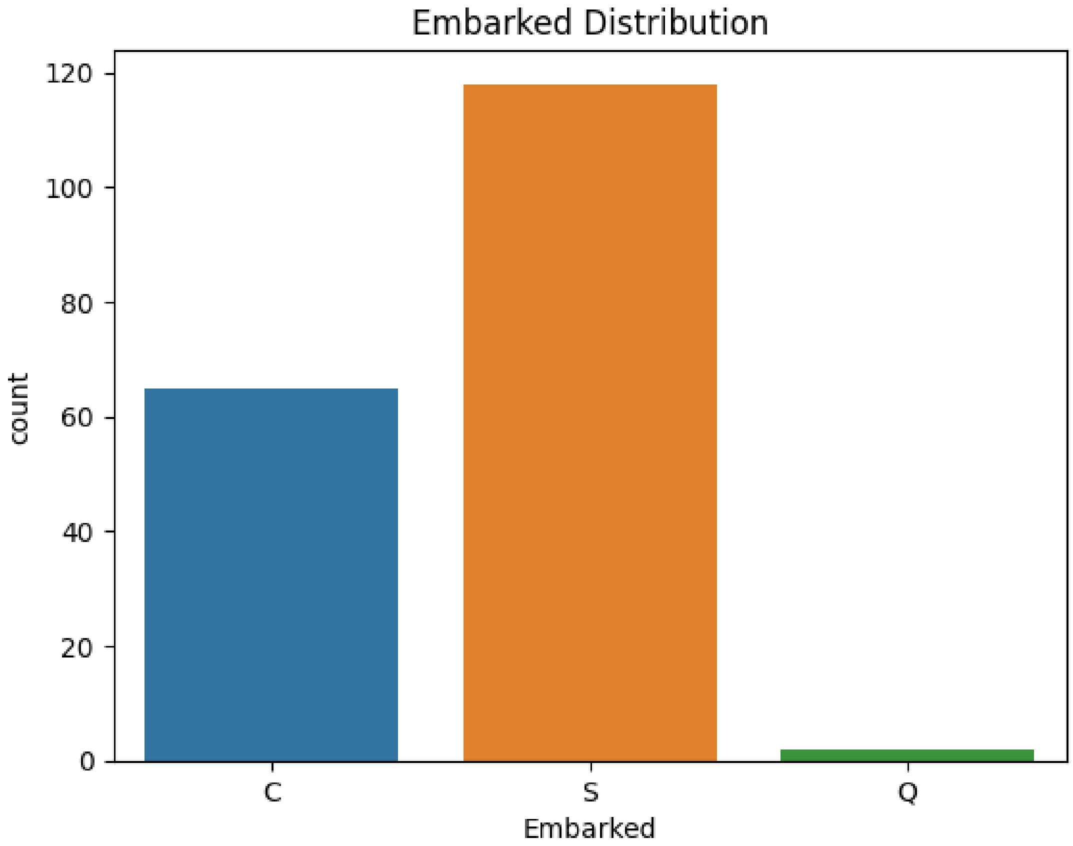


[ ]:

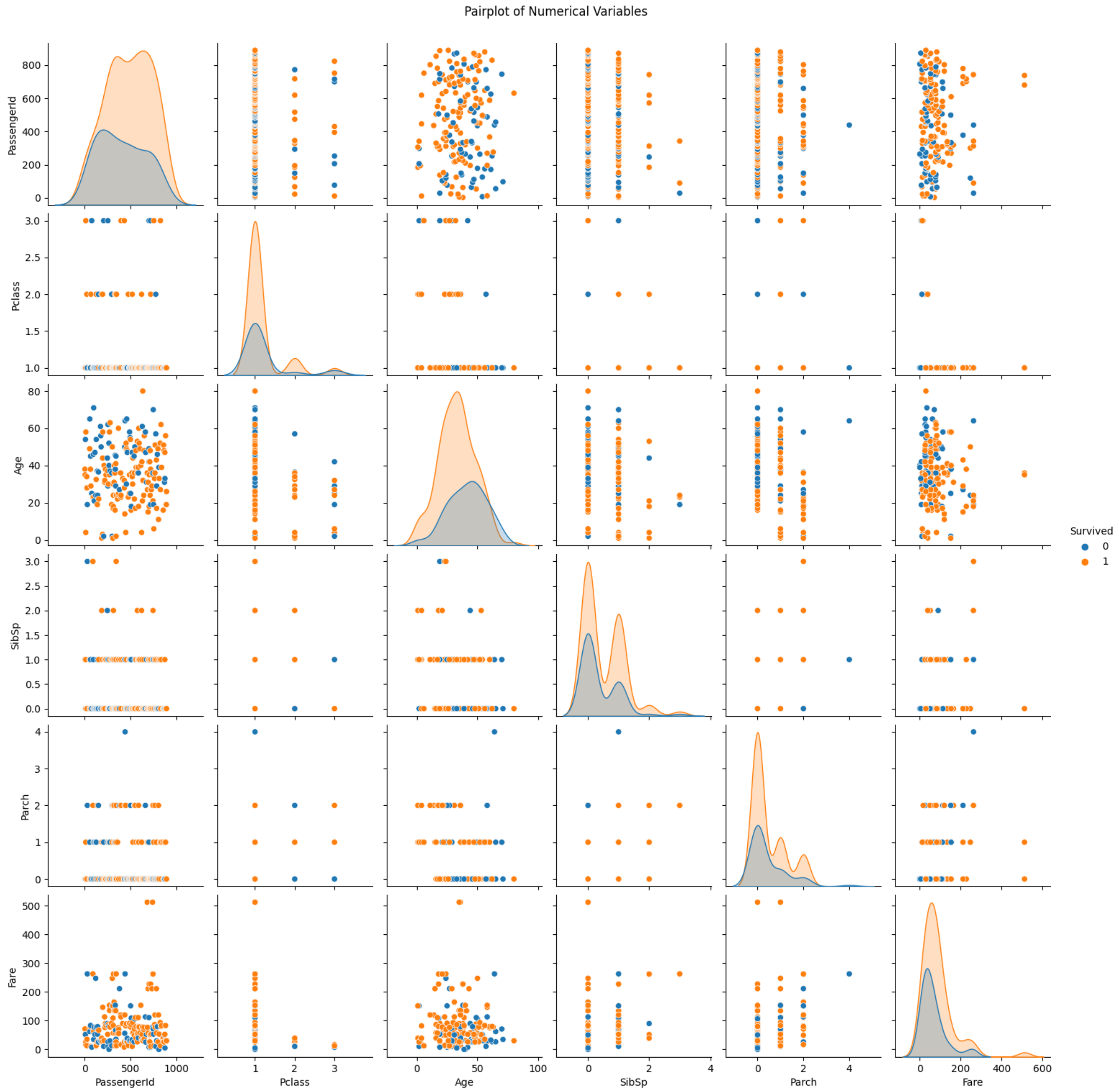
```
# Countplot for Sex
sns.countplot(x='Sex', data=df1)
plt.title('Gender Distribution')
plt.show()
```



```
[ ]: # Countplot for Embarked
     sns.countplot(x='Embarked', data=df1)
     plt.title('Embarked Distribution')
     plt.show()
```



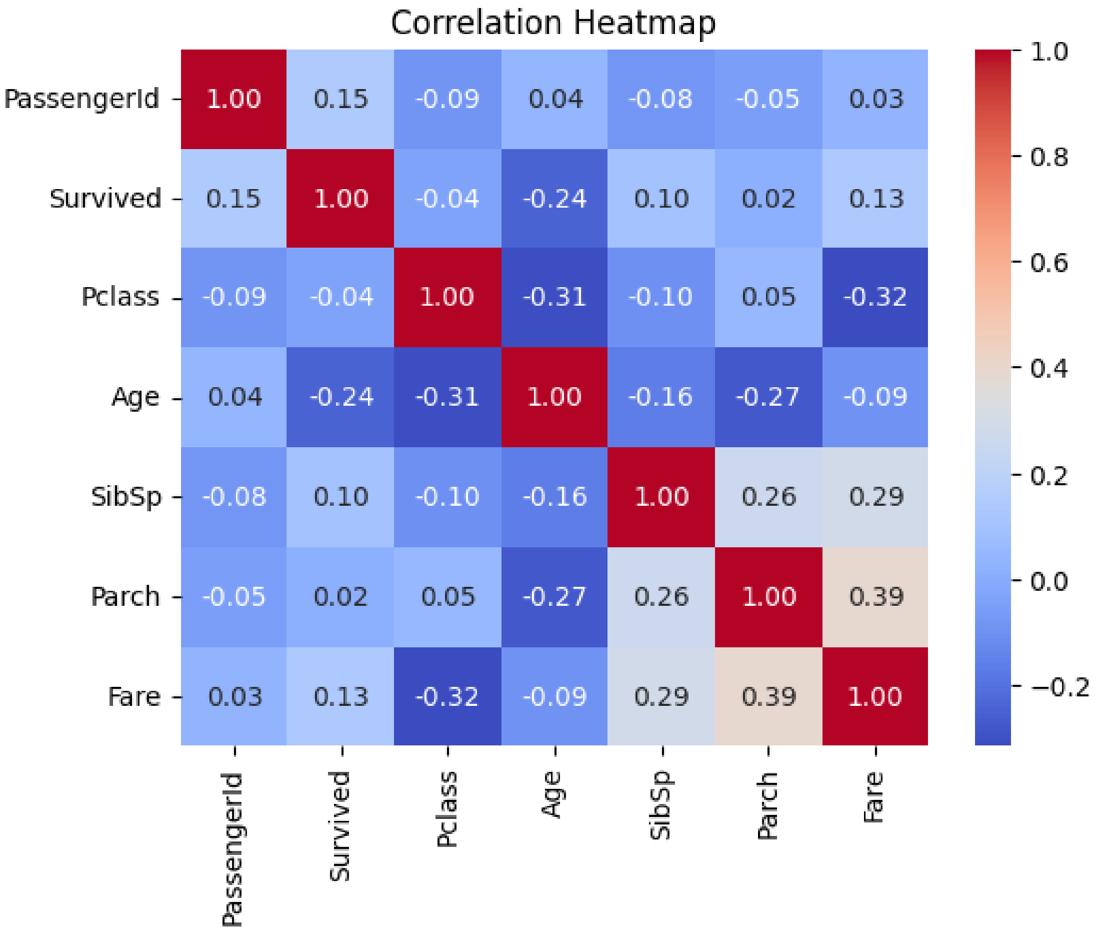
```
[ ]: # Pairplot to explore relationships between numerical variables
     sns.pairplot(df1, hue='Survived')
     plt.suptitle('Pairplot of Numerical Variables', y=1.02)
     plt.show()
```



[ ]:

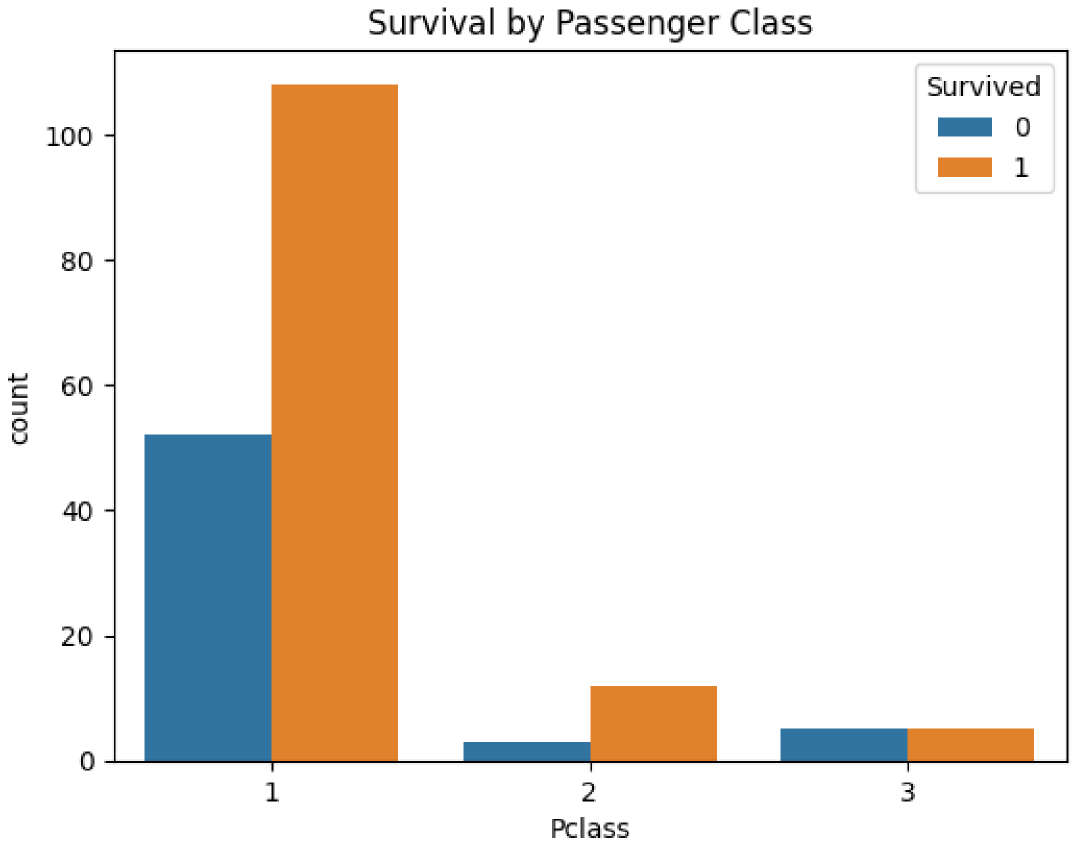
```
# Correlation heatmap
correlation_matrix = df1.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```

```
<ipython-input-23-fe033e8ba074>:2: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  correlation_matrix = df1.corr()
```



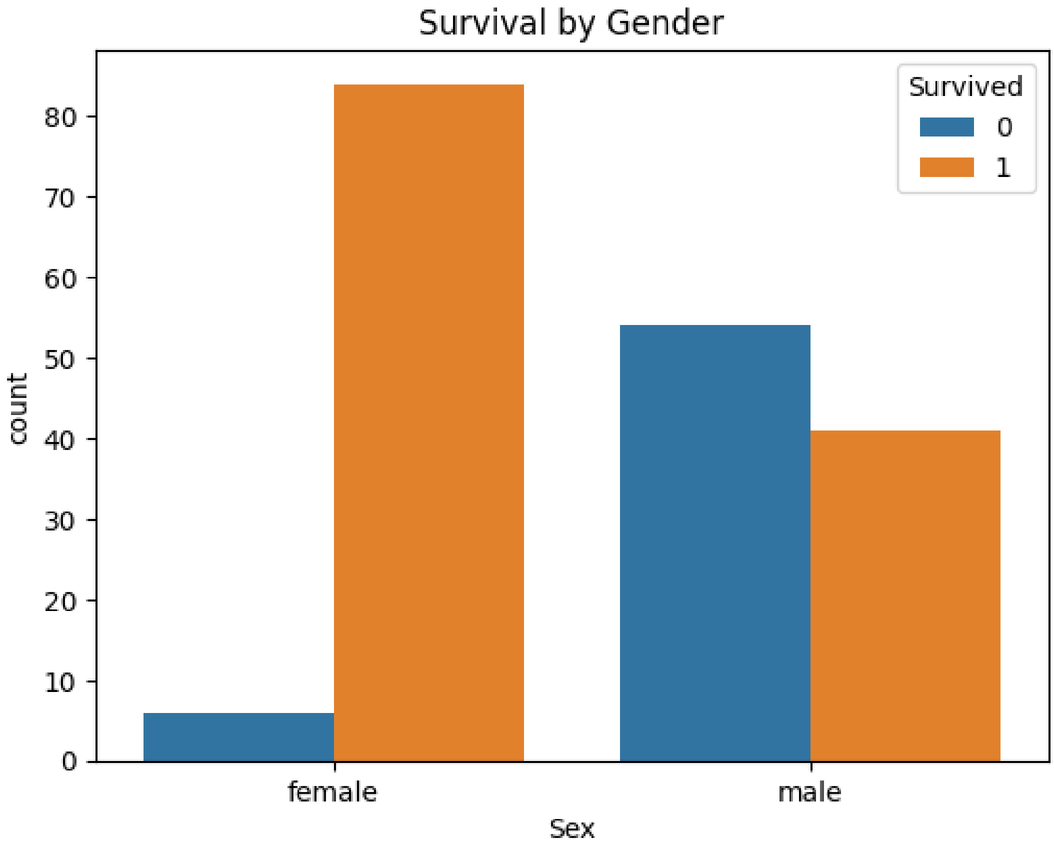
# Explore relationships and patterns

[ ]:
```
# Survival by Passenger Class
sns.countplot(x='Pclass', hue='Survived', data=df1)
plt.title('Survival by Passenger Class')
plt.show()
```

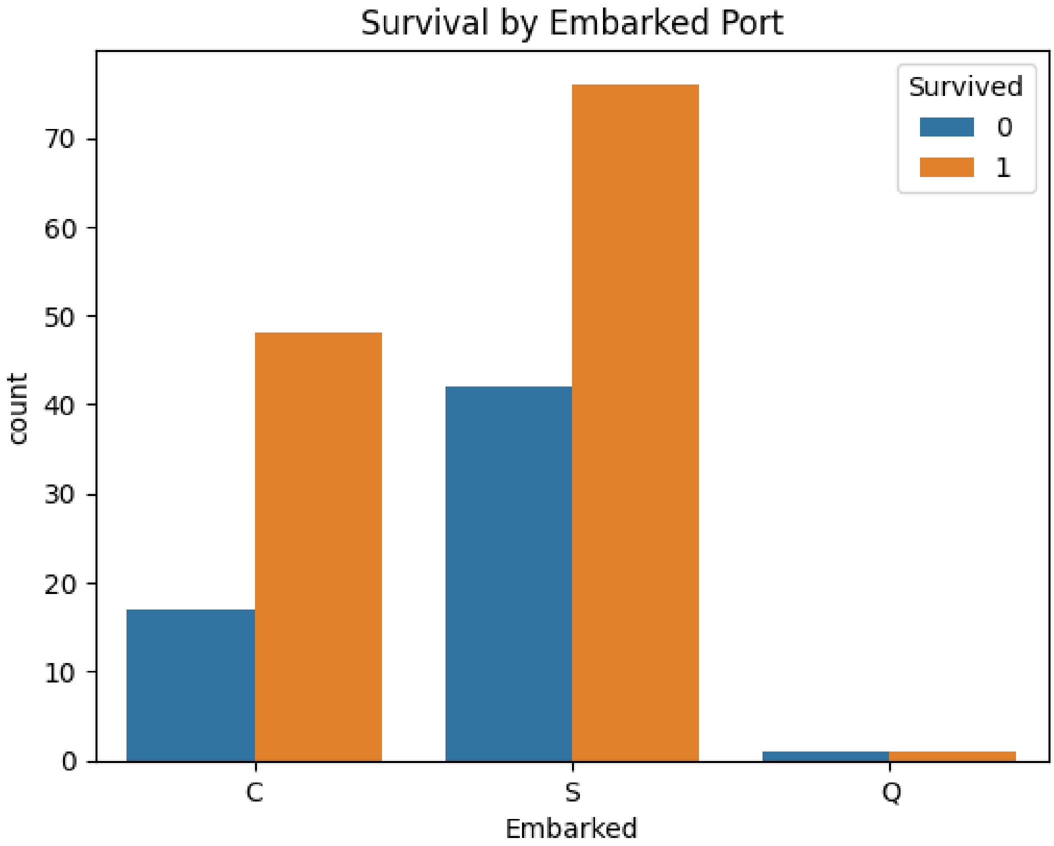


[ ]:

```
# Survival by Gender
sns.countplot(x='Sex', hue='Survived', data=df1)
plt.title('Survival by Gender')
plt.show()
```

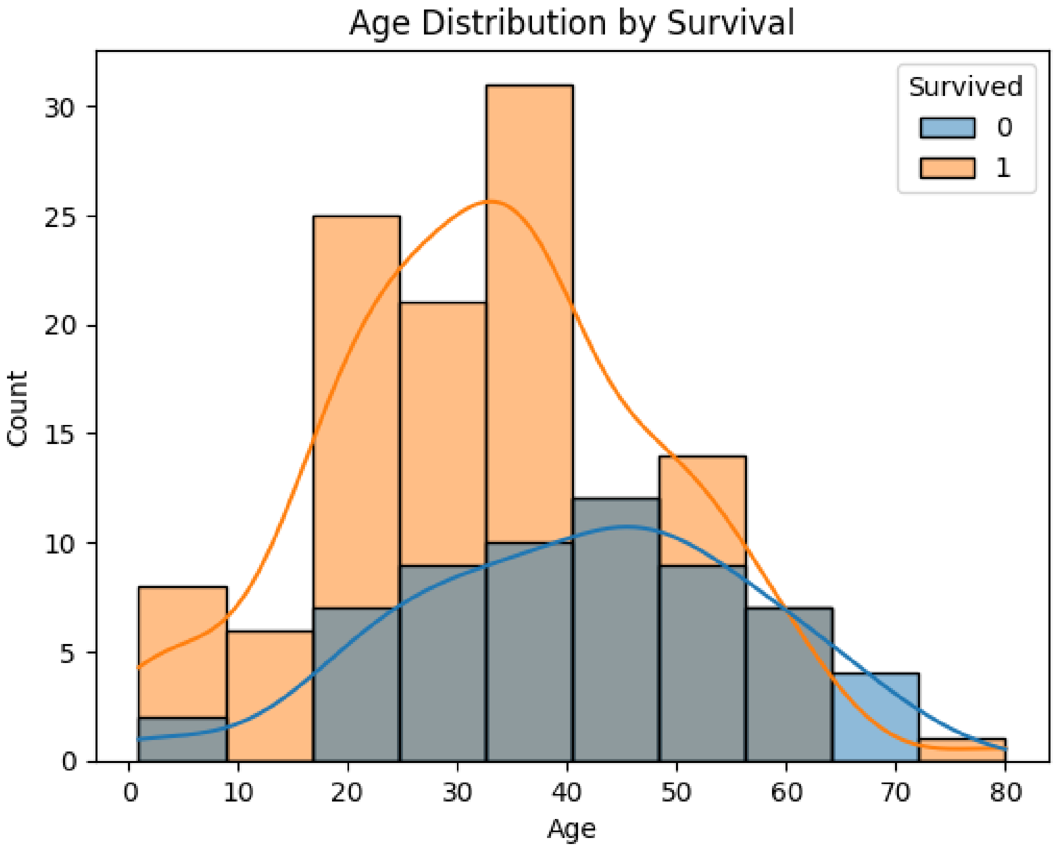


```
[ ]: # Survival by Embarked Port
     sns.countplot(x='Embarked', hue='Survived', data=df1)
     plt.title('Survival by Embarked Port')
     plt.show()
```

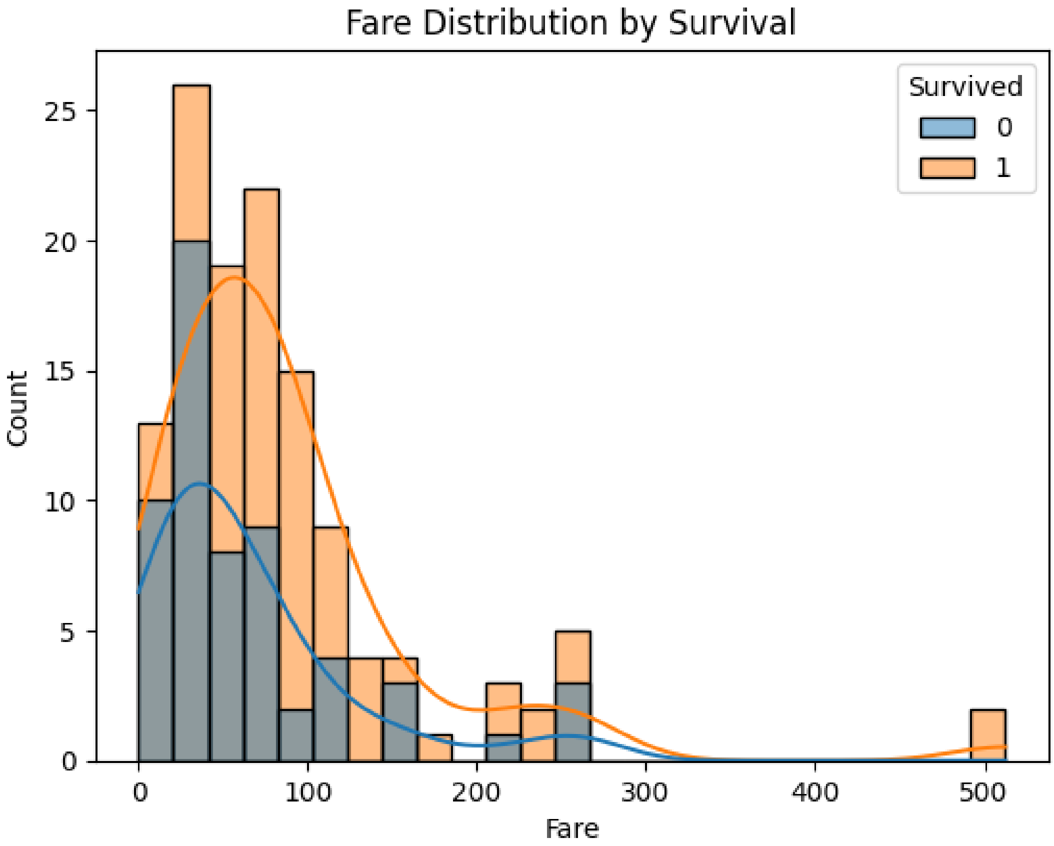


[ ]:

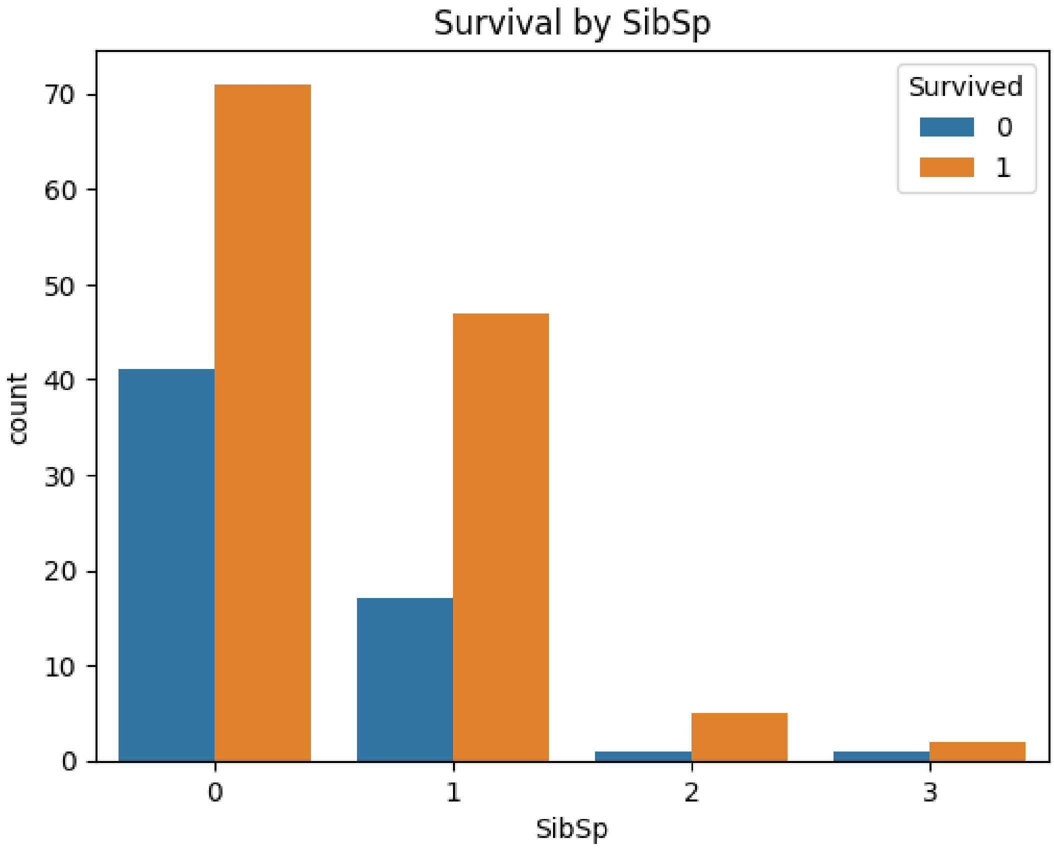
```
# Age distribution by Survival
sns.histplot(x='Age', hue='Survived', data=df1, kde=True)
plt.title('Age Distribution by Survival')
plt.show()
```



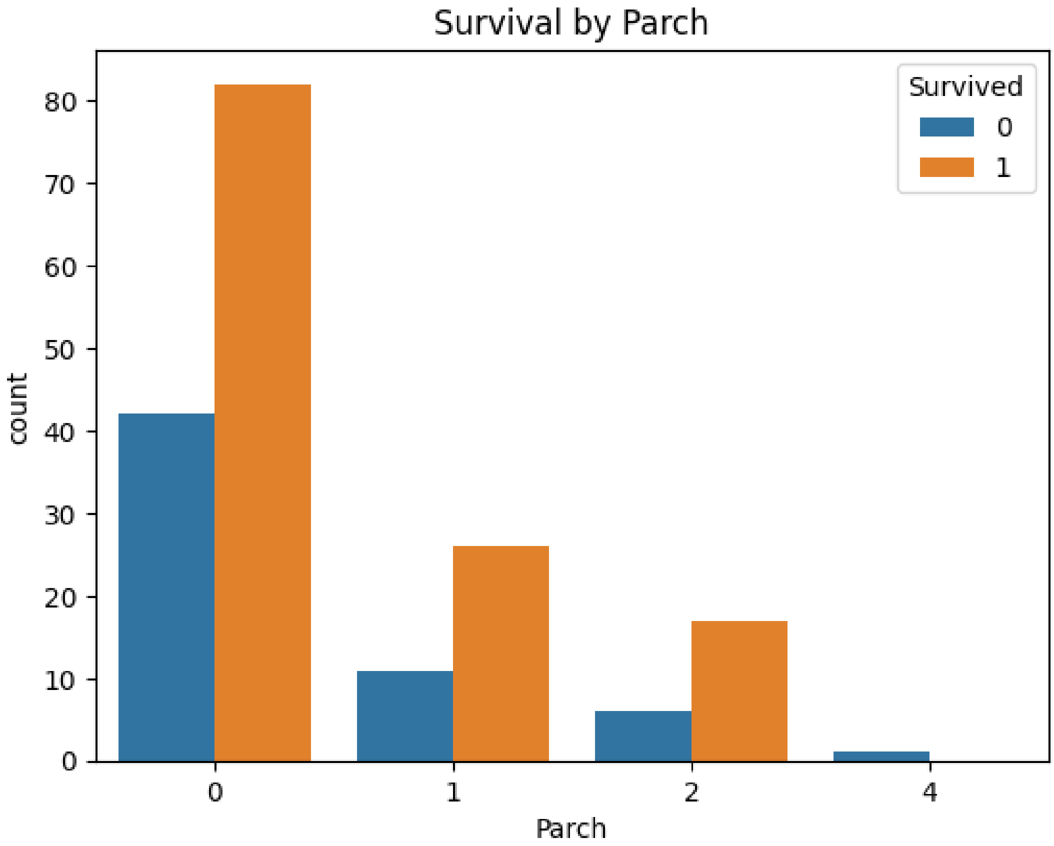
```
[ ]: # Fare distribution by Survival
     sns.histplot(x='Fare', hue='Survived', data=df1, kde=True)
     plt.title('Fare Distribution by Survival')
     plt.show()
```



[ ]:
```
# Survival by SibSp (Number of Siblings/Spouses Aboard)
sns.countplot(x='SibSp', hue='Survived', data=df1)
plt.title('Survival by SibSp')
plt.show()
```



[ ]:
```
# Survival by Parch (Number of Parents/Children Aboard)
sns.countplot(x='Parch', hue='Survived', data=df1)
plt.title('Survival by Parch')
plt.show()
```



```
[ ]: df1.head()
```

```
[ ]:     PassengerId  Survived  Pclass  \
    1             2         1       1
    3             4         1       1
    6             7         0       1
    10           11         1       3
    11           12         1       1

                                                    Name     Sex   Age  SibSp  \
    1   Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
    3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
    6                              McCarthy, Mr. Timothy J    male  54.0      0
    10                     Sandstrom, Miss. Marguerite Rut  female   4.0      1
    11                            Bonnell, Miss. Elizabeth  female  58.0      0

        Parch    Ticket     Fare Cabin Embarked
    1       0  PC 17599  71.2833   C85        C
    3       0    113803  53.1000  C123        S
    6       0     17463  51.8625   E46        S
```

```
10       1    PP 9549   16.7000    G6         S
11       0     113783   26.5500   C103        S
```

Label Encoding

[ ]:

```python
from sklearn.preprocessing import LabelEncoder
lb = LabelEncoder()
df1['Sex']=lb.fit_transform(df1['Sex'])
df1['Embarked']=lb.fit_transform(df1['Embarked'])
```

```
<ipython-input-32-591ec63db8c1>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['Sex']=lb.fit_transform(df1['Sex'])
<ipython-input-32-591ec63db8c1>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['Embarked']=lb.fit_transform(df1['Embarked'])
```

[ ]:

```python
#Train Test Split
# Select features and target variable
X = df1[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']]
y = df1['Survived']
```

[ ]:

```python
# Split the data into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
  ↪random_state=42)
```

Standardization

[ ]:

```python
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
X_train = ss.fit_transform(X_train)
X_test = ss.transform(X_test)
```

Model building - Logistic Regression

[ ]:

```python
# Create and train a logistic regression model
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(random_state=33)
model.fit(X_train, y_train)
```

```
[ ]: LogisticRegression(random_state=33)
```

```
[ ]: # Make predictions on the test set
     y_pred = model.predict(X_test)
```

Evaluation Metrics

```
[ ]: # Evaluate the model
     from sklearn.metrics import accuracy_score, confusion_matrix,␣
      ↪classification_report
     accuracy = accuracy_score(y_test, y_pred)*100
     conf_matrix = confusion_matrix(y_test, y_pred)
     classification_rep = classification_report(y_test, y_pred)

     print(f'Accuracy: {accuracy:.4f}')
     print(f'Confusion Matrix:\n{conf_matrix}')
     print(f'Classification Report:\n{classification_rep}')
```

```
Accuracy: 75.6757
Confusion Matrix:
[[11  4]
 [ 5 17]]
Classification Report:
              precision    recall  f1-score   support

           0       0.69      0.73      0.71        15
           1       0.81      0.77      0.79        22

    accuracy                           0.76        37
   macro avg       0.75      0.75      0.75        37
weighted avg       0.76      0.76      0.76        37
```

In this example, we use logistic regression as a classifier. We preprocess the data by handling missing values, encoding categorical variables using Label Encoding, and standardizing numerical features. Finally, we train the model, make predictions on the test set, and evaluate its performance using accuracy, confusion matrix, and classification report.

```
[ ]: #BY HARI
```

```
[ ]: #Happy coding
```