# EDA ON Titanic Survival Predictions

## Load the Titanic dataset

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from flask import Flask, render_template, request
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
```

```
titanic_df = pd.read_csv('/content/Titanic-Dataset.csv')
```

## Display the first few rows of the dataset

```
print(titanic_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
```

## info about the dataset

```
print(titanic_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
None
```

## Describe the numerical features

```
print(titanic_df.describe())
```

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

            Parch        Fare  
count  891.000000  891.000000  
mean     0.381594   32.204208  
std      0.806057   49.693429  
min      0.000000    0.000000  
25%      0.000000    7.910400  
50%      0.000000   14.454200  
75%      0.000000   31.000000  
max      6.000000  512.329200  
```

## Check the unique values for categorical features

```
for col in titanic_df.select_dtypes(include=['object']).columns:
  print(f"Unique values for {col}: {titanic_df[col].unique()}")
```

```
Unique values for Name: ['Braund, Mr. Owen Harris'
 'Cumings, Mrs. John Bradley (Florence Briggs Thayer)'
 'Heikkinen, Miss. Laina' 'Futrelle, Mrs. Jacques Heath (Lily May Peel)'
 'Allen, Mr. William Henry' 'Moran, Mr. James' 'McCarthy, Mr. Timothy J'
 'Palsson, Master. Gosta Leonard'
 'Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)'
 'Nasser, Mrs. Nicholas (Adele Achem)' 'Sandstrom, Miss. Marguerite Rut'
 'Bonnell, Miss. Elizabeth' 'Saundercock, Mr. William Henry'
 'Andersson, Mr. Anders Johan' 'Vestrom, Miss. Hulda Amanda Adolfina'
 'Hewlett, Mrs. (Mary D Kingcome) ' 'Rice, Master. Eugene'
 'Williams, Mr. Charles Eugene'
 'Vander Planke, Mrs. Julius (Emelia Maria Vandemoortele)'
 'Masselmani, Mrs. Fatima' 'Fynney, Mr. Joseph J' 'Beesley, Mr. Lawrence'
 'McGowan, Miss. Anna "Annie"' 'Sloper, Mr. William Thompson'
 'Palsson, Miss. Torborg Danira'
 'Asplund, Mrs. Carl Oscar (Selma Augusta Emilia Johansson)'
 'Emir, Mr. Farred Chehab' 'Fortune, Mr. Charles Alexander'
 'O\'Dwyer, Miss. Ellen "Nellie"' 'Todoroff, Mr. Lalio'
 'Uruchurtu, Don. Manuel E'
 'Spencer, Mrs. William Augustus (Marie Eugenie)'
 'Glynn, Miss. Mary Agatha' 'Wheadon, Mr. Edward H'
 'Meyer, Mr. Edgar Joseph' 'Holverson, Mr. Alexander Oskar'
 'Mamee, Mr. Hanna' 'Cann, Mr. Ernest Charles'
 'Vander Planke, Miss. Augusta Maria' 'Nicola-Yarred, Miss. Jamila'
 'Ahlin, Mrs. Johan (Johanna Persdotter Larsson)'
 'Turpin, Mrs. William John Robert (Dorothy Ann Wonnacott)'
 'Kraeff, Mr. Theodor' 'Laroche, Miss. Simonne Marie Anne Andree'
 'Devaney, Miss. Margaret Delia' 'Rogers, Mr. William John'
 'Lennon, Mr. Denis' "O'Driscoll, Miss. Bridget" 'Samaan, Mr. Youssef'
 'Arnold-Franchi, Mrs. Josef (Josefine Franchi)'
 'Panula, Master. Juha Niilo' 'Nosworthy, Mr. Richard Cater'
 'Harper, Mrs. Henry Sleeper (Myna Haxtun)'
 'Faunthorpe, Mrs. Lizzie (Elizabeth Anne Wilkinson)'
 'Ostby, Mr. Engelhart Cornelius' 'Woolner, Mr. Hugh' 'Rugg, Miss. Emily'
 'Novel, Mr. Mansouer' 'West, Miss. Constance Mirium'
 'Goodwin, Master. William Frederick' 'Sirayanian, Mr. Orsen'
 'Icard, Miss. Amelie' 'Harris, Mr. Henry Birkhardt'
 'Skoog, Master. Harald' 'Stewart, Mr. Albert A'
 'Moubarek, Master. Gerios' 'Nye, Mrs. (Elizabeth Ramell)'
 'Crease, Mr. Ernest James' 'Andersson, Miss. Erna Alexandra'
 'Kink, Mr. Vincenz' 'Jenkin, Mr. Stephen Curnow'
 'Goodwin, Miss. Lillian Amy' 'Hood, Mr. Ambrose Jr'
 'Chronopoulos, Mr. Apostolos' 'Bing, Mr. Lee' 'Moen, Mr. Sigurd Hansen'
 'Staneff, Mr. Ivan' 'Moutal, Mr. Rahamin Haim'
 'Caldwell, Master. Alden Gates' 'Dowdell, Miss. Elizabeth'
 'Waelens, Mr. Achille' 'Sheerlinck, Mr. Jan Baptist'
 'McDermott, Miss. Brigdet Delia' 'Carrau, Mr. Francisco M'
 'Ilett, Miss. Bertha'
 'Backstrom, Mrs. Karl Alfred (Maria Mathilda Gustafsson)'
 'Ford, Mr. William Neal' 'Slocovski, Mr. Selman Francis'
 'Fortune, Miss. Mabel Helen' 'Celotti, Mr. Francesco'
 'Christmann, Mr. Emil' 'Andreasson, Mr. Paul Edvin'
 'Chaffee, Mr. Herbert Fuller' 'Dean, Mr. Bertram Frank'
 'Coxon, Mr. Daniel' 'Shorney, Mr. Charles Joseph'
 'Goldschmidt, Mr. George B' 'Greenfield, Mr. William Bertram'
 'Doling, Mrs. John T (Ada Julia Bone)' 'Kantor, Mr. Sinai'
 'Petranec, Miss. Matilda' 'Petroff, Mr. Pastcho ("Pentcho")'
```

## Understand the target variable (Survived) distribution

```
print(titanic_df['Survived'].value_counts())
```

```
Survived
0    549
1    342
Name: count, dtype: int64
```

## Handle missing values

```
titanic_df['Age'].fillna(titanic_df['Age'].median(), inplace=True)
titanic_df['Embarked'].fillna(titanic_df['Embarked'].mode()[0], inplace=True)
titanic_df.drop('Cabin', axis=1, inplace=True)
```

```
<ipython-input-8-306afe30fe31>:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation i


  titanic_df['Age'].fillna(titanic_df['Age'].median(), inplace=True)
<ipython-input-8-306afe30fe31>:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation i


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

## Feature Engineering

```
titanic_df['FamilySize'] = titanic_df['SibSp'] + titanic_df['Parch'] + 1
```

## Encode categorical variables

```
le = LabelEncoder()
titanic_df['Sex'] = le.fit_transform(titanic_df['Sex'])
titanic_df['Embarked'] = le.fit_transform(titanic_df['Embarked'])

print(titanic_df.head())
print(titanic_df.info())
```

```
   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  Parch  \
0                            Braund, Mr. Owen Harris    1  22.0      1      0
1  Cumings, Mrs. John Bradley (Florence Briggs Th...    0  38.0      1      0
2                             Heikkinen, Miss. Laina    0  26.0      0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)    0  35.0      1      0
4                           Allen, Mr. William Henry    1  35.0      0      0

             Ticket     Fare  Embarked  FamilySize
0         A/5 21171   7.2500         2           2
1          PC 17599  71.2833         0           2
2  STON/O2. 3101282   7.9250         2           1
3            113803  53.1000         2           2
4            373450   8.0500         2           1
<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    int64  
 5   Age          891 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  Embarked     891 non-null    int64  
 11  FamilySize   891 non-null    int64  
dtypes: float64(2), int64(8), object(2)
memory usage: 83.7+ KB
None
```

## Correlation analysis

```
numerical_features = titanic_df.select_dtypes(include=[np.number])
correlation_matrix = numerical_features.corr()
print(correlation_matrix['Survived'].sort_values(ascending=False))
```

```
Survived       1.000000
Fare           0.257307
Parch          0.081629
PassengerId   -0.005007
SibSp         -0.035322
Age           -0.077221
Pclass        -0.338481
Name: Survived, dtype: float64
```

## Analyze survival rate by different features

## Survival rate by Pclass

```
print(titanic_df.groupby('Pclass')['Survived'].mean())
```

```
Pclass
1    0.629630
2    0.472826
3    0.242363
Name: Survived, dtype: float64
```

## Survival rate by Sex

```
print(titanic_df.groupby('Sex')['Survived'].mean())
```

```
Sex
female    0.742038
male      0.188908
Name: Survived, dtype: float64
```

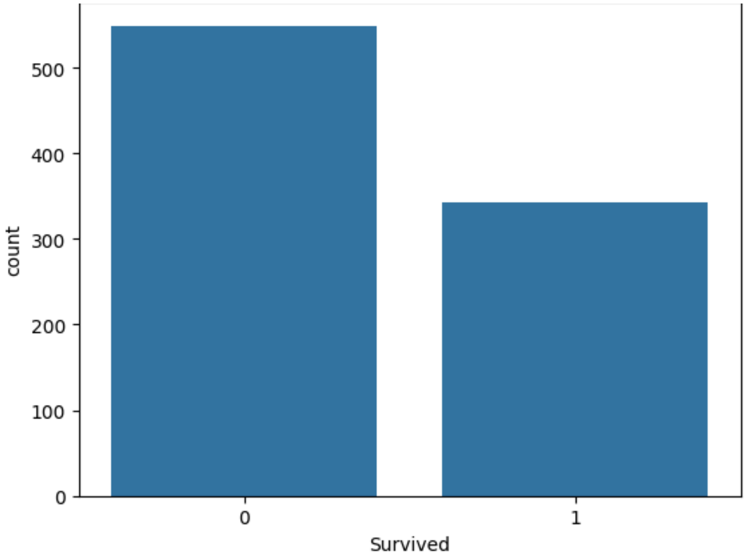
## Survival rate by Embarked

```
print(titanic_df.groupby('Embarked')['Survived'].mean())
```

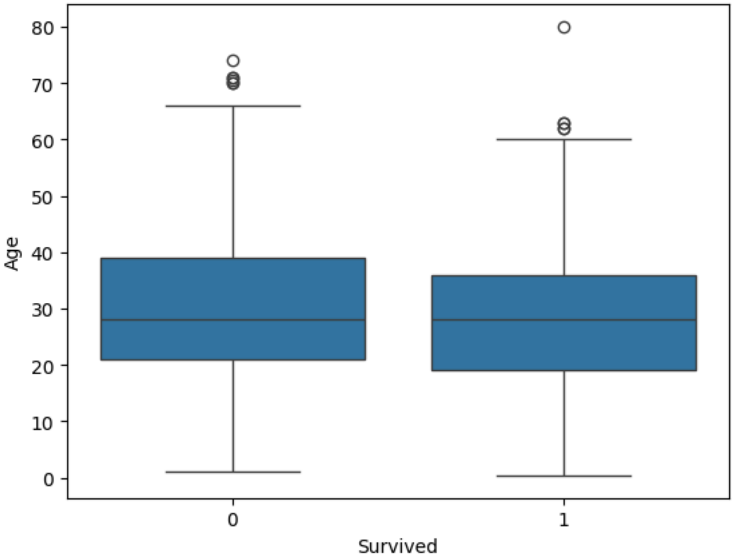
```
Embarked
C    0.553571
Q    0.389610
S    0.336957
Name: Survived, dtype: float64
```

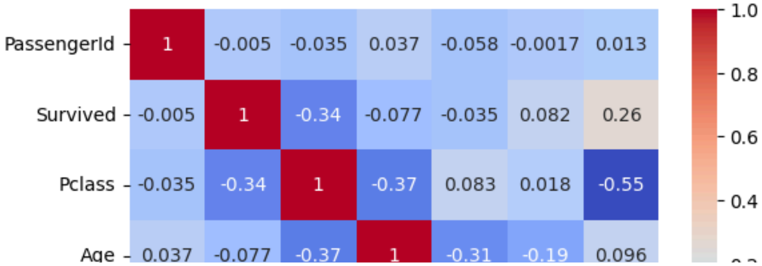
## Visualizations

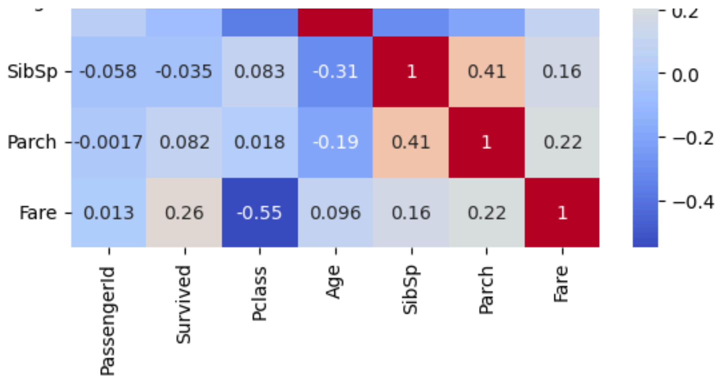
```
sns.countplot(x='Survived', data=titanic_df)
plt.show()
sns.boxplot(x='Survived', y='Age', data=titanic_df)
plt.show()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.show()
```



```
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWarning: SeriesGroupBy.grouper is deprecated and will be removed in a future version of pandas.
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
```







## Analyze survival rate based on title extracted from the Name column

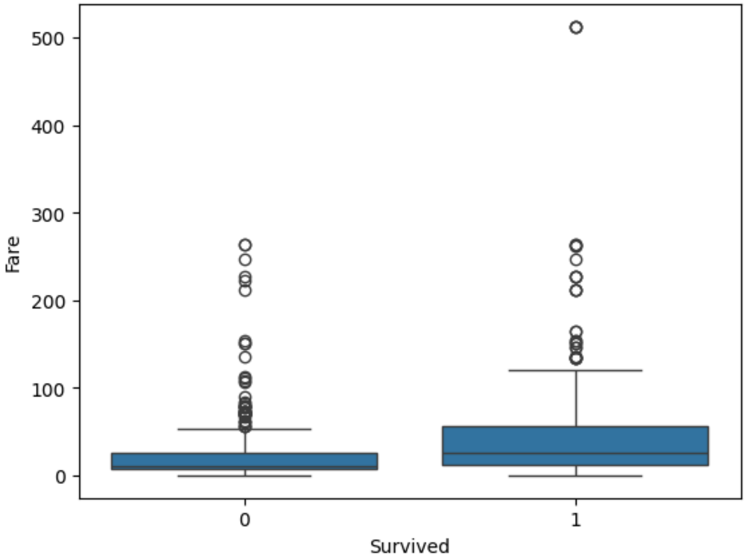
```
titanic_df['Title'] = titanic_df['Name'].str.extract(' ([A-Za-z]+)\.', expand=False)
print(titanic_df.groupby('Title')['Survived'].mean())
```

```
Title
Capt        0.000000
Col         0.500000
Countess    1.000000
Don         0.000000
Dr          0.428571
Jonkheer    0.000000
Lady        1.000000
Major       0.500000
Master      0.575000
Miss        0.697802
Mlle        1.000000
Mme         1.000000
Mr          0.156673
Mrs         0.792000
Ms          1.000000
Rev         0.000000
Sir         1.000000
Name: Survived, dtype: float64
```

## Investigate the relationship between Fare and survival rate

```
print(titanic_df.groupby('Survived')['Fare'].mean())
sns.boxplot(x='Survived', y='Fare', data=titanic_df)
plt.show()
```

```
Survived
0    22.117887
1    48.395408
Name: Fare, dtype: float64
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWarning: SeriesGroupBy.grouper is deprecated and will be removed in a future version of pandas.
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
```



## Create new features based on existing ones

```
titanic_df['AgeGroup'] = pd.cut(titanic_df['Age'], bins=[0, 18, 30, 50, float('inf')], labels=['Child', 'Young Adult', 'Adult', 'Senior'])
titanic_df['FareRange'] = pd.qcut(titanic_df['Fare'], q=4, labels=['Low', 'Medium', 'High', 'Very High'])
```

## Interaction term between Pclass and Fare

```
titanic_df['PclassFare'] = titanic_df['Pclass'] * titanic_df['Fare']
```

## Explore different machine learning models and compare their performance

```
X = titanic_df[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked', 'FamilySize', 'Title', 'AgeGroup', 'FareRange', 'PclassFare']]
y = titanic_df['Survived']

X = pd.get_dummies(X, columns=['Title', 'AgeGroup', 'FareRange'], dummy_na=True)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

## Define features (X) and target (y)

```
X = titanic_df.drop('Survived', axis=1)
y = titanic_df['Survived']
```

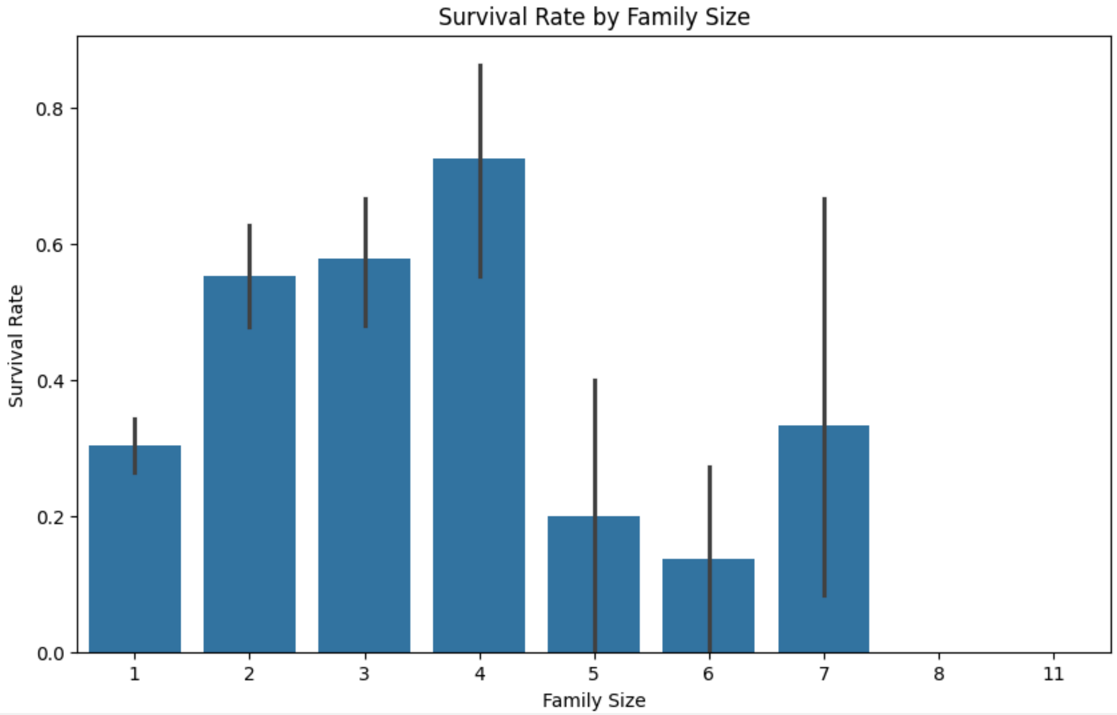
## Split data into training and testing sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

## Survival Rate by Age Group

## Survival Rate by Family Size

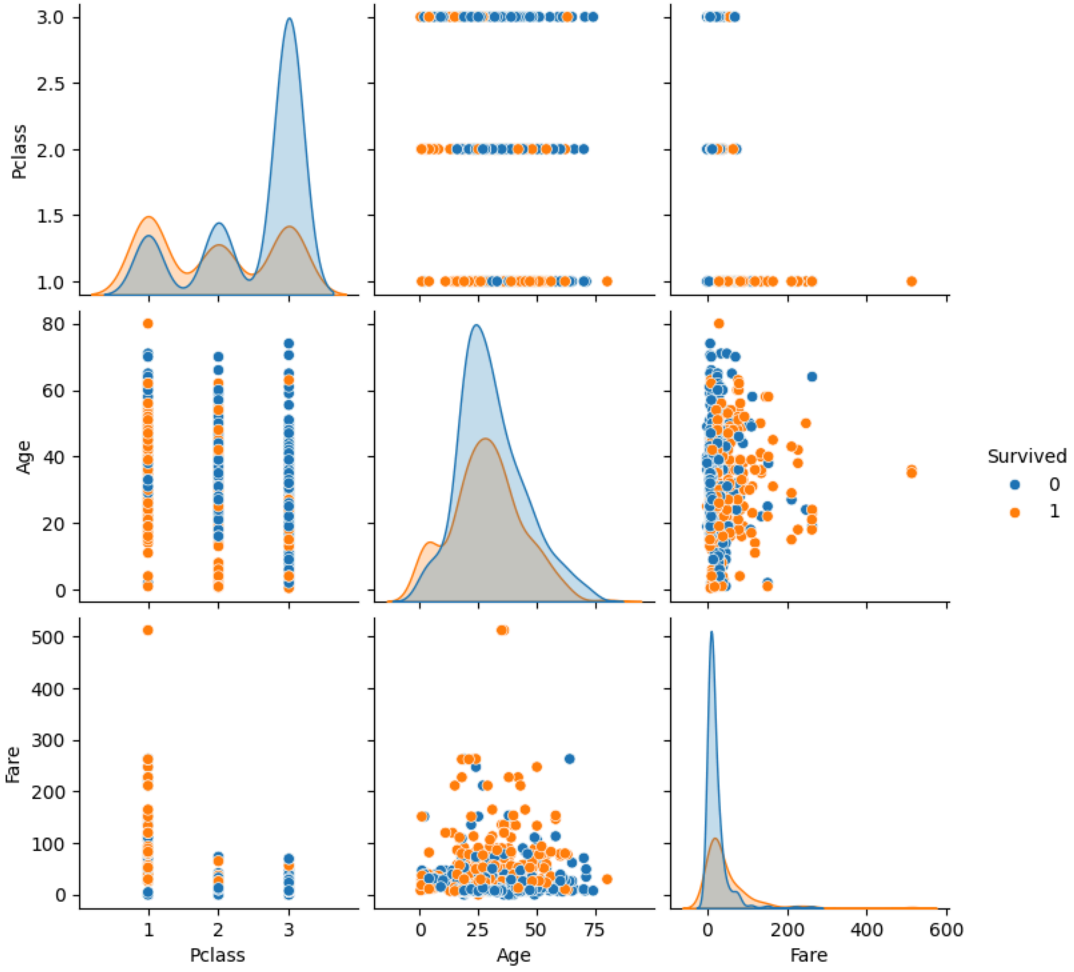
```
titanic_df['FamilySize'] = titanic_df['SibSp'] + titanic_df['Parch'] + 1

plt.figure(figsize=(10, 6))
sns.barplot(x='FamilySize', y='Survived', data=titanic_df)
plt.title('Survival Rate by Family Size')
plt.xlabel('Family Size')
plt.ylabel('Survival Rate')
plt.show()
```



## Pairplot for selected features

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

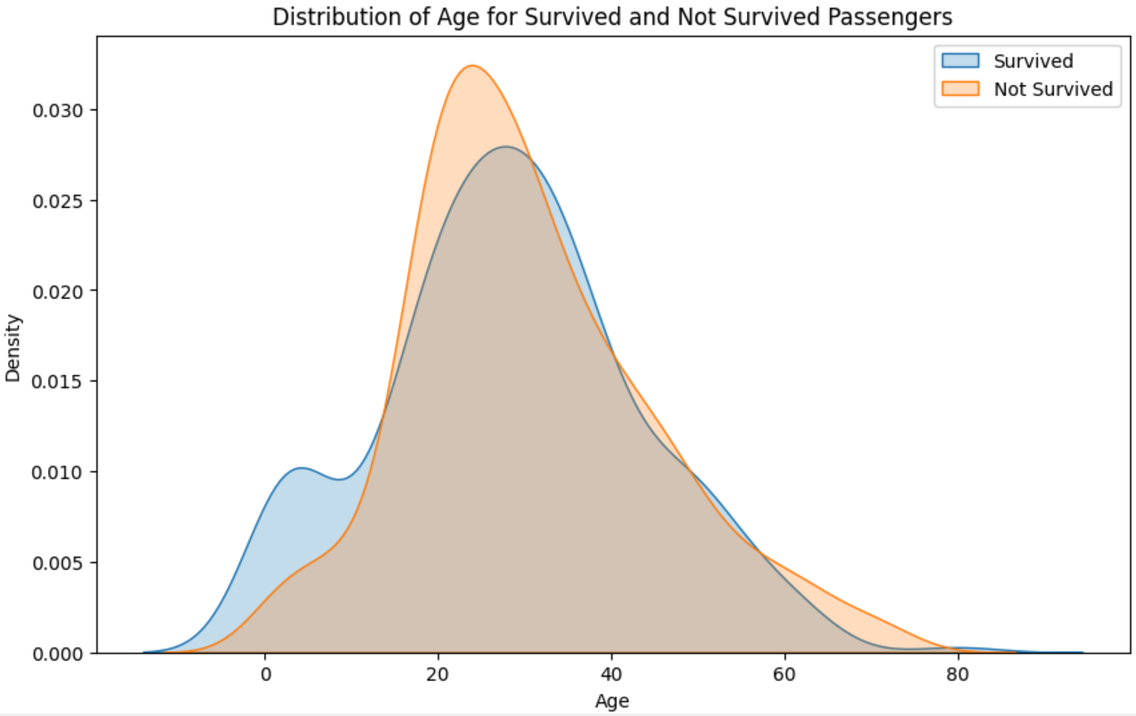
```
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future \
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future \
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future \
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future \
  data_subset = grouped_data.get_group(pd_key)
```



## Distribution of Age for Survived and Not Survived passengers

```
plt.figure(figsize=(10, 6))
sns.kdeplot(titanic_df[titanic_df['Survived'] == 1]['Age'], label='Survived', shade=True)
sns.kdeplot(titanic_df[titanic_df['Survived'] == 0]['Age'], label='Not Survived', shade=True)
plt.title('Distribution of Age for Survived and Not Survived Passengers')
plt.xlabel('Age')
plt.ylabel('Density')
plt.legend()
plt.show()
```

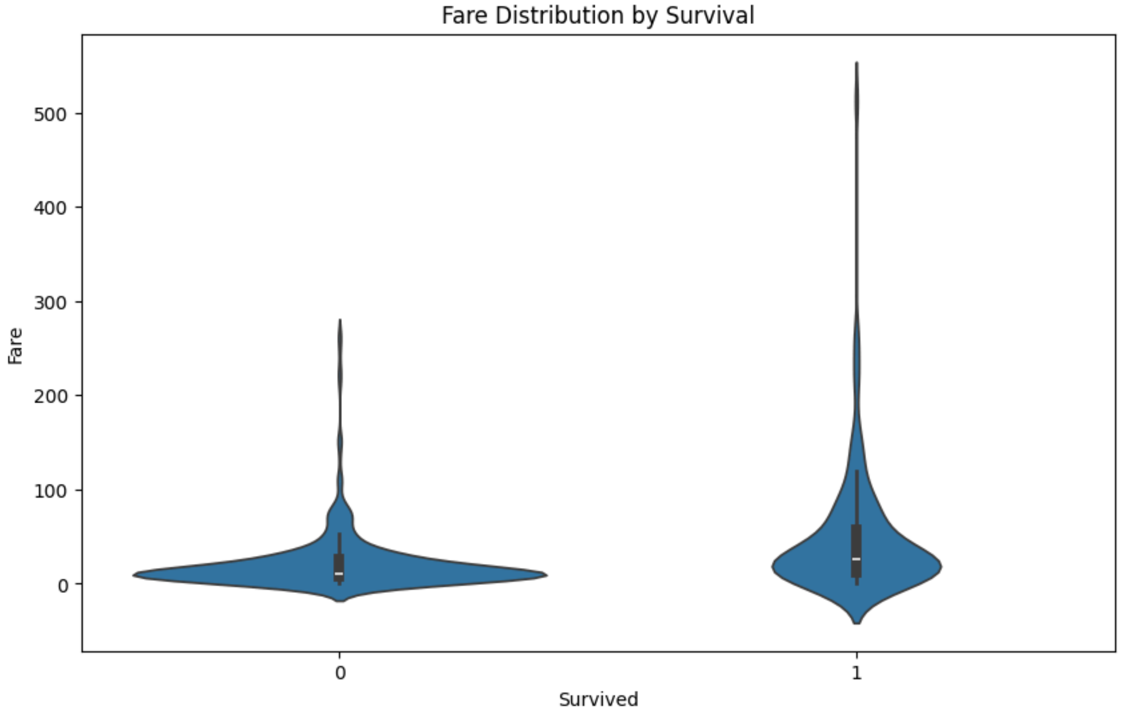
```
<ipython-input-25-5483e231d5cf>:2: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(titanic_df[titanic_df['Survived'] == 1]['Age'], label='Survived', shade=True)
<ipython-input-25-5483e231d5cf>:3: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(titanic_df[titanic_df['Survived'] == 0]['Age'], label='Not Survived', shade=True)
```



## Violin plot of Fare by Survived

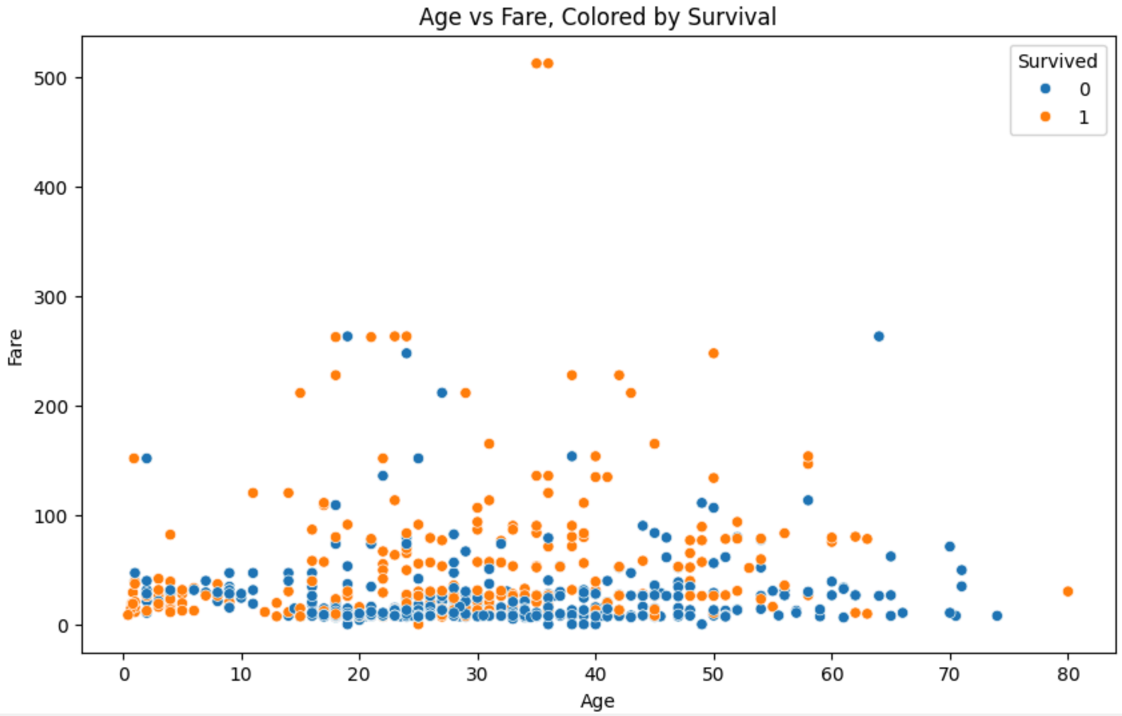
```
plt.figure(figsize=(10, 6))
sns.violinplot(x='Survived', y='Fare', data=titanic_df
plt.title('Fare Distribution by Survival')
plt.xlabel('Survived')
plt.ylabel('Fare')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future v
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future v
  data_subset = grouped_data.get_group(pd_key)
```



## Scatter plot of Age vs Fare, colored by Survived

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Age', y='Fare', hue='Survived', data=titanic_df)
plt.title('Age vs Fare, Colored by Survival')
plt.xlabel('Age')
plt.ylabel('Fare')
plt.show()
```



## Analyze Survival Rate by Ticket Class and Sex Combined

```
titanic_df['PclassSex'] = titanic_df['Pclass'].astype(str) + '_' + titanic_df['Sex'].astype(str)
print(titanic_df.groupby('PclassSex')['Survived'].mean())
```

```
PclassSex
1_female    0.968085
1_male      0.368852
2_female    0.921053
2_male      0.157407
3_female    0.500000
3_male      0.135447
Name: Survived, dtype: float64
```

## Investigate Survival Rate by Age and Fare Interaction

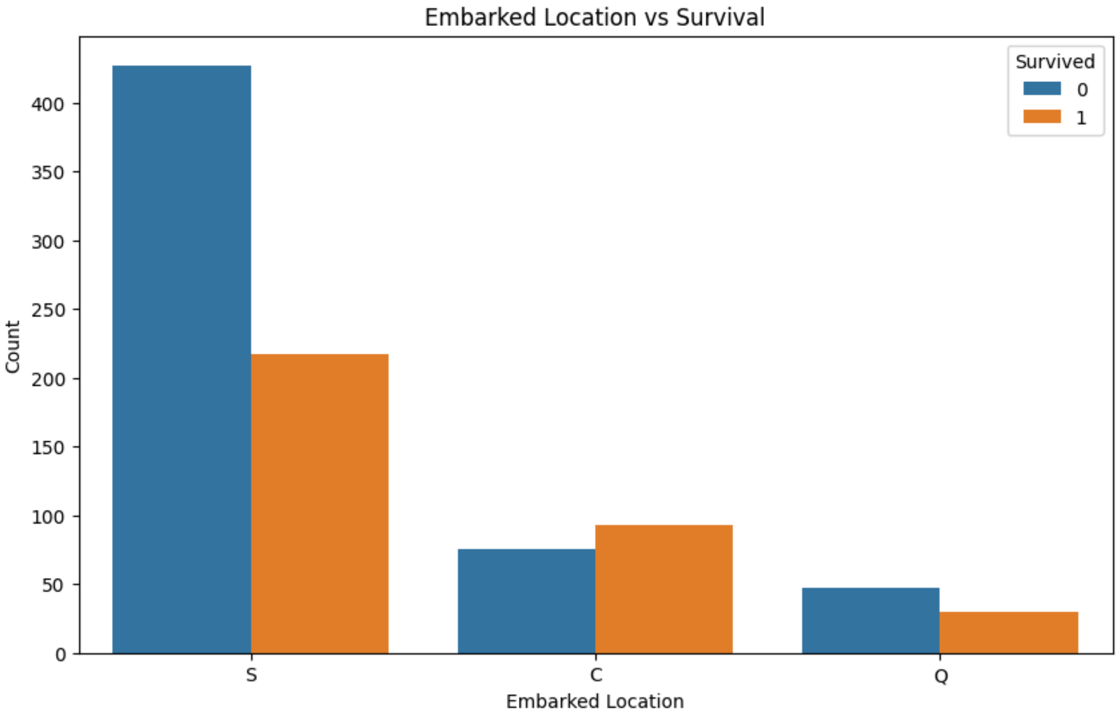
```
titanic_df['AgeRange'] = pd.cut(titanic_df['Age'], bins=[0, 10, 20, 30, 40, 50, float('inf')])
titanic_df['FareRange'] = pd.qcut(titanic_df['Fare'], q=4, labels=['Low', 'Medium', 'High', 'Very High'])
print(titanic_df.groupby(['AgeRange', 'FareRange'])['Survived'].mean())
```

```
AgeRange      FareRange
(0.0, 10.0]   Low               NaN
              Medium       0.875000
              High         0.617647
              Very High    0.454545
(10.0, 20.0]  Low          0.250000
              Medium       0.365854
              High         0.470588
              Very High    0.520000
(20.0, 30.0]  Low          0.238806
              Medium       0.250000
              High         0.488372
              Very High    0.636364
(30.0, 40.0]  Low          0.038462
              Medium       0.325581
              High         0.463415
              Very High    0.777778
(40.0, 50.0]  Low          0.000000
              Medium       0.368421
              High         0.400000
              Very High    0.516129
(50.0, inf]   Low          0.000000
              Medium       0.153846
              High         0.352941
              Very High    0.518519
Name: Survived, dtype: float64
<ipython-input-3-dbd9974b9a4a>:3: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain c
  print(titanic_df.groupby(['AgeRange', 'FareRange'])['Survived'].mean())
```

## Explore the distribution of Embarked locations for survived and non-survived passengers

```
plt.figure(figsize=(10, 6))
sns.countplot(x='Embarked', hue='Survived', data=titanic_df)
plt.title('Embarked Location vs Survival')
plt.xlabel('Embarked Location')
plt.ylabel('Count')
plt.show()
```

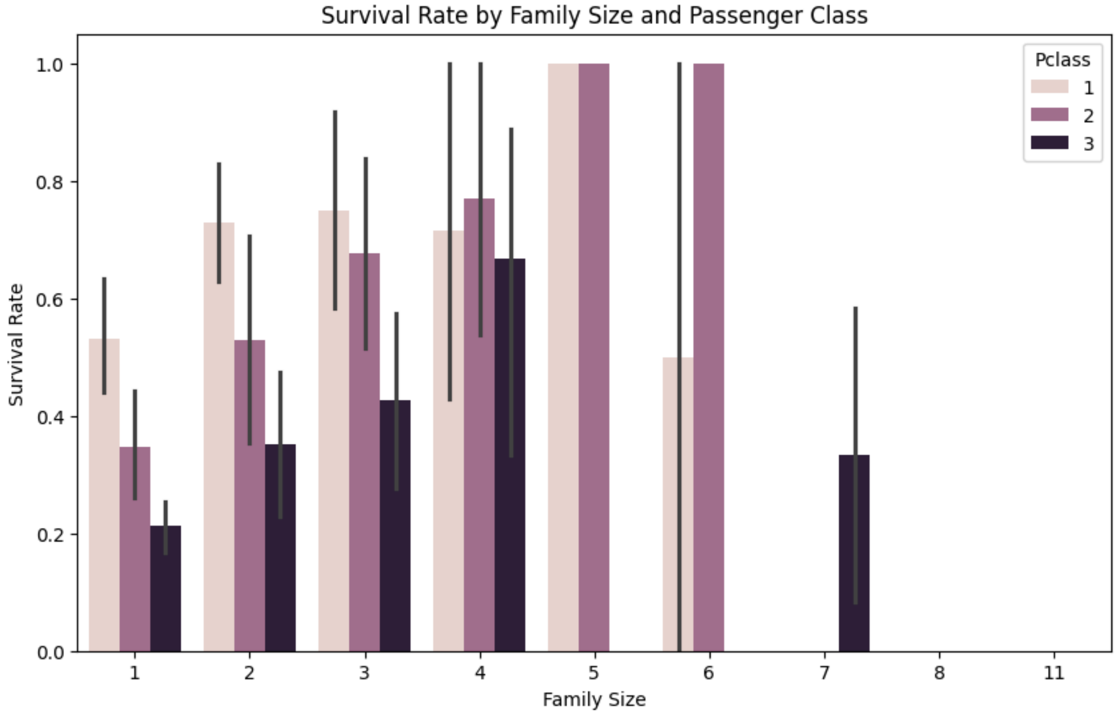
```
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future v
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future v
  data_subset = grouped_data.get_group(pd_key)
```



## Analyze the relationship between Family Size and Survival Rate for different passenger classes

```
titanic_df['FamilySize'] = titanic_df['SibSp'] + titanic_df['Parch'] + 1

plt.figure(figsize=(10, 6))
sns.barplot(x='FamilySize', y='Survived', hue='Pclass', data=titanic_df)
plt.title('Survival Rate by Family Size and Passenger Class')
plt.xlabel('Family Size')
plt.ylabel('Survival Rate')
plt.show()
```

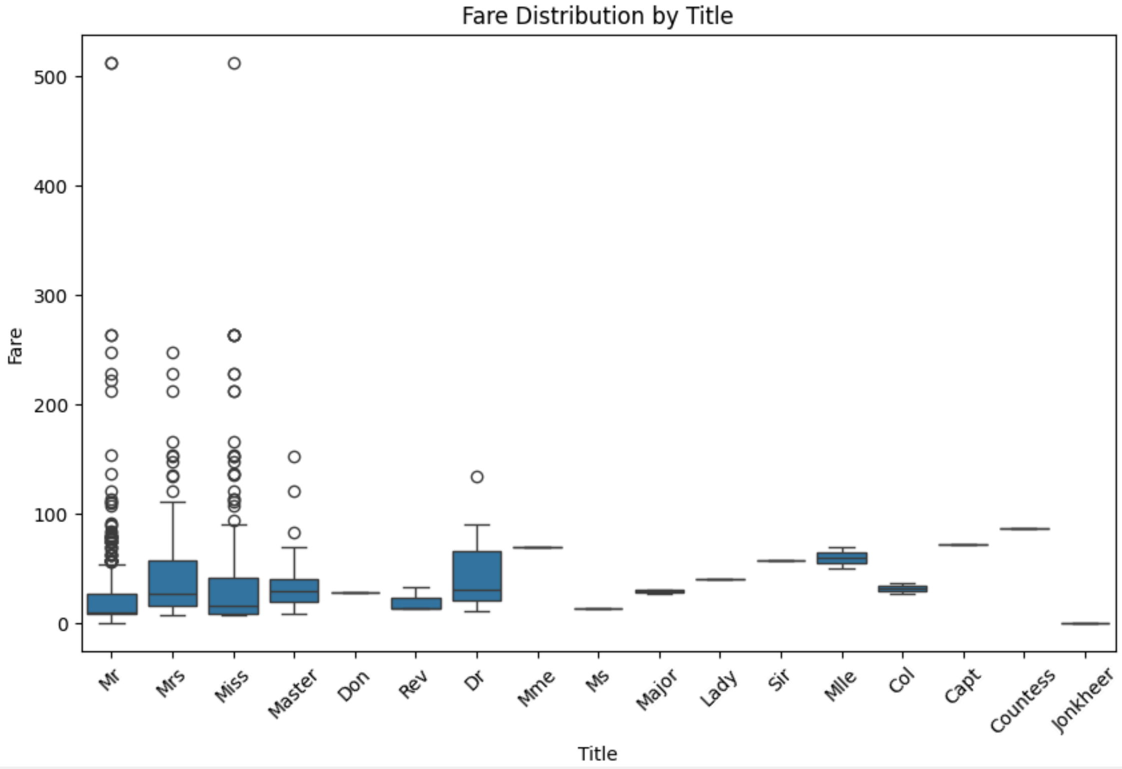
```
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future
  data_subset = grouped_data.get_group(pd_key)
```



## Explore the relationship between Title and Fare

```
titanic_df['Title'] = titanic_df['Name'].str.extract(' ([A-Za-z]+)\.', expand=False)
plt.figure(figsize=(10, 6))
sns.boxplot(x='Title', y='Fare', data=titanic_df)
plt.title('Fare Distribution by Title')
plt.xlabel('Title')
plt.ylabel('Fare')
plt.xticks(rotation=45)
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWarning: SeriesGroupBy.grouper is deprecated and will be removed in a future version of pandas.
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
```



## Create a correlation matrix including the new features created in the EDA process

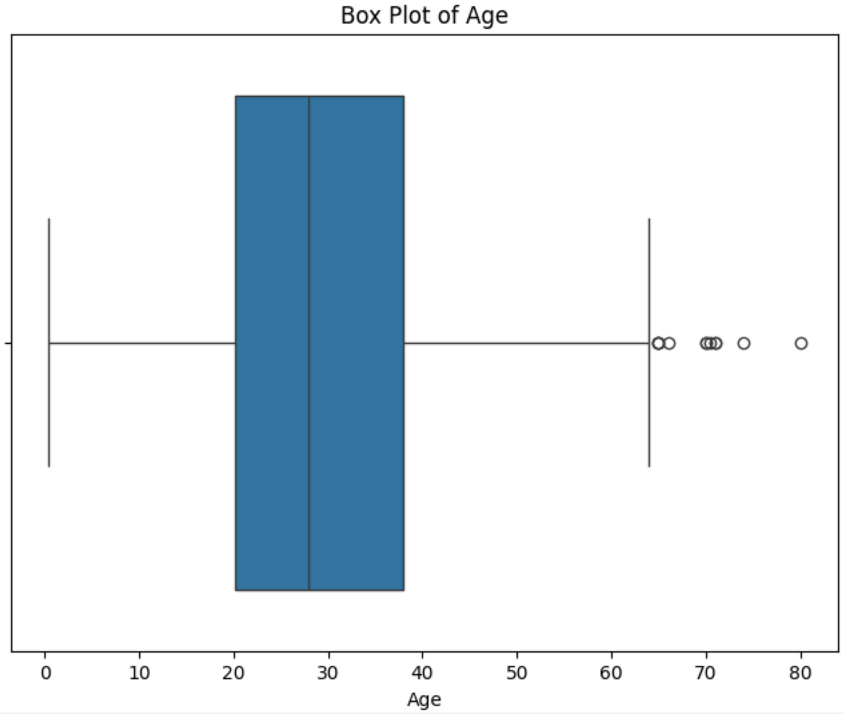
```
numerical_features = titanic_df.select_dtypes(include=['number'])
correlation_matrix = numerical_features.corr()
print(correlation_matrix['Survived'].sort_values(ascending=False))
```

```
Survived       1.000000
Fare           0.257307
Parch          0.081629
FamilySize     0.016639
PassengerId   -0.005007
SibSp         -0.035322
Age           -0.077221
Pclass        -0.338481
Name: Survived, dtype: float64
```

## Check for outliers in Age using a box plot

```
plt.figure(figsize=(8, 6))
sns.boxplot(x='Age', data=titanic_df)
plt.title('Box Plot of Age')
plt.show()
```

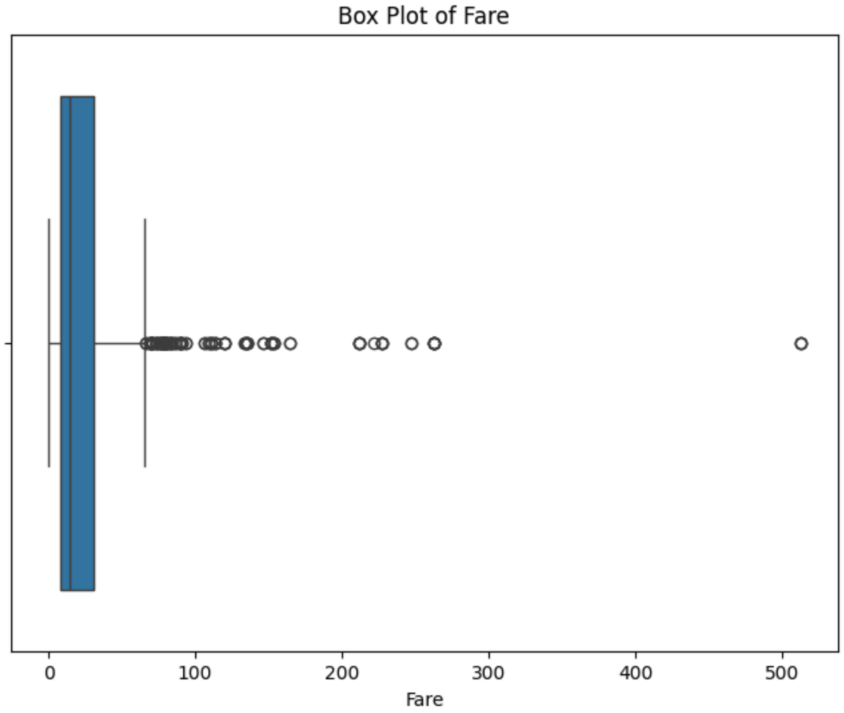
```
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWarning: SeriesGroupBy.grouper is deprecated and will be removed in a future version of pandas.
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
```



## Check for outliers in Fare using a box plot

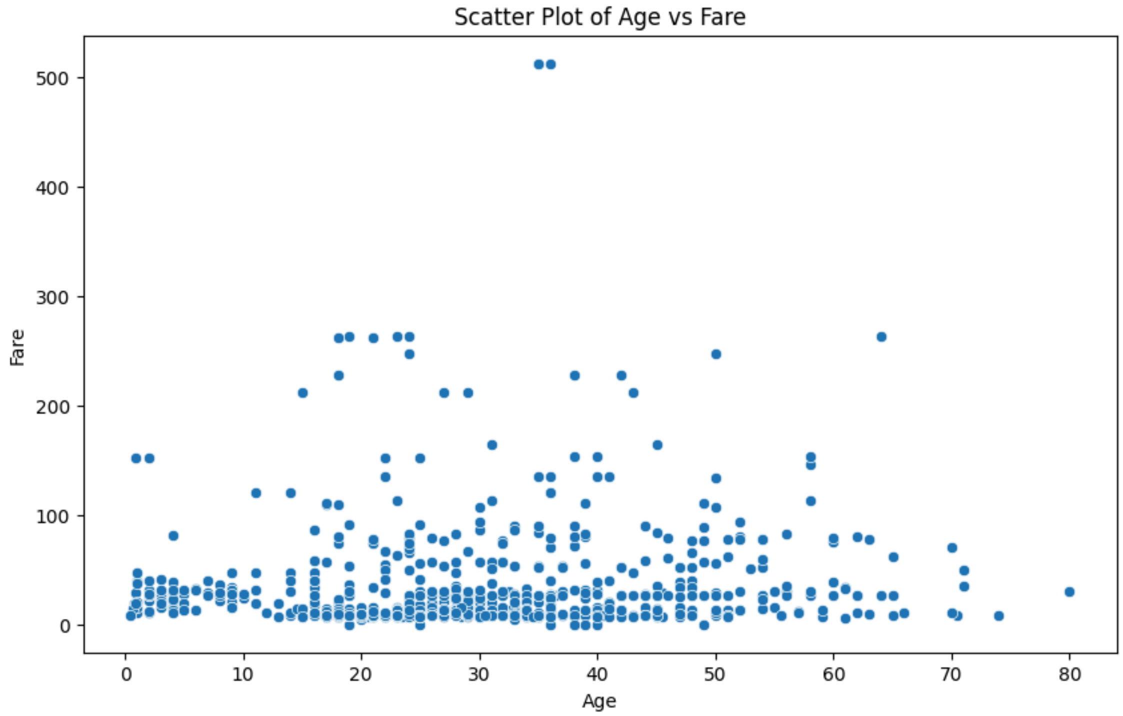
```
plt.figure(figsize=(8, 6))
sns.boxplot(x='Fare', data=titanic_df)
plt.title('Box Plot of Fare')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWarning: SeriesGroupBy.grouper is deprecated and will be removed in a future version of pandas.
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
```



## scatter plot to visualize outliers in Age and Fare together

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Age', y='Fare', data=titanic_df)
plt.title('Scatter Plot of Age vs Fare')
plt.show()
```



## IQR (Interquartile Range) to identify outliers more precisely

```
Q1_age = titanic_df['Age'].quantile(0.25)
Q3_age = titanic_df['Age'].quantile(0.75)
IQR_age = Q3_age - Q1_age
lower_bound_age = Q1_age - 1.5 * IQR_age
upper_bound_age = Q3_age + 1.5 * IQR_age

Q1_fare = titanic_df['Fare'].quantile(0.25)
Q3_fare = titanic_df['Fare'].quantile(0.75)
IQR_fare = Q3_fare - Q1_fare
lower_bound_fare = Q1_fare - 1.5 * IQR_fare
upper_bound_fare = Q3_fare + 1.5 * IQR_fare
```

## Identify outliers for Age and Fare

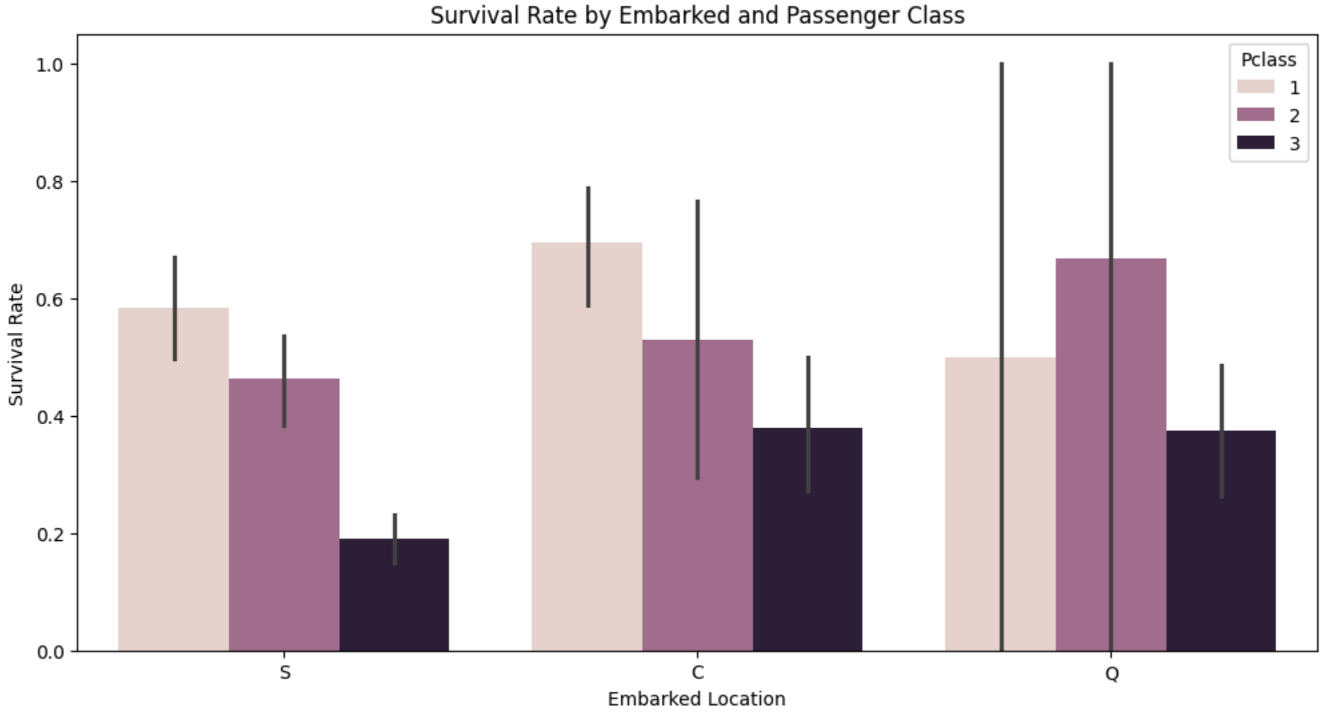
```
outliers_age = titanic_df[(titanic_df['Age'] < lower_bound_age) | (titanic_df['Age'] > upper_bound_age)]
outliers_fare = titanic_df[(titanic_df['Fare'] < lower_bound_fare) | (titanic_df['Fare'] > upper_bound_fare)]

print("Outliers in Age:")
print(outliers_age)
print("\nOutliers in Fare:")
print(outliers_fare)


print(titanic_df.groupby(['Embarked', 'Pclass'])['Survived'].mean())
print(titanic_df.groupby(['Embarked', 'Fare'])['Survived'].mean())
print(titanic_df.groupby(['Embarked', 'Pclass', 'Fare'])['Survived'].mean())
```

```
               7.2292      0.266667
               7.8958      0.200000
                              ...
      S        153.4625    0.666667
```

## Visualize the survival rate by Embarked, Pclass, and Fare

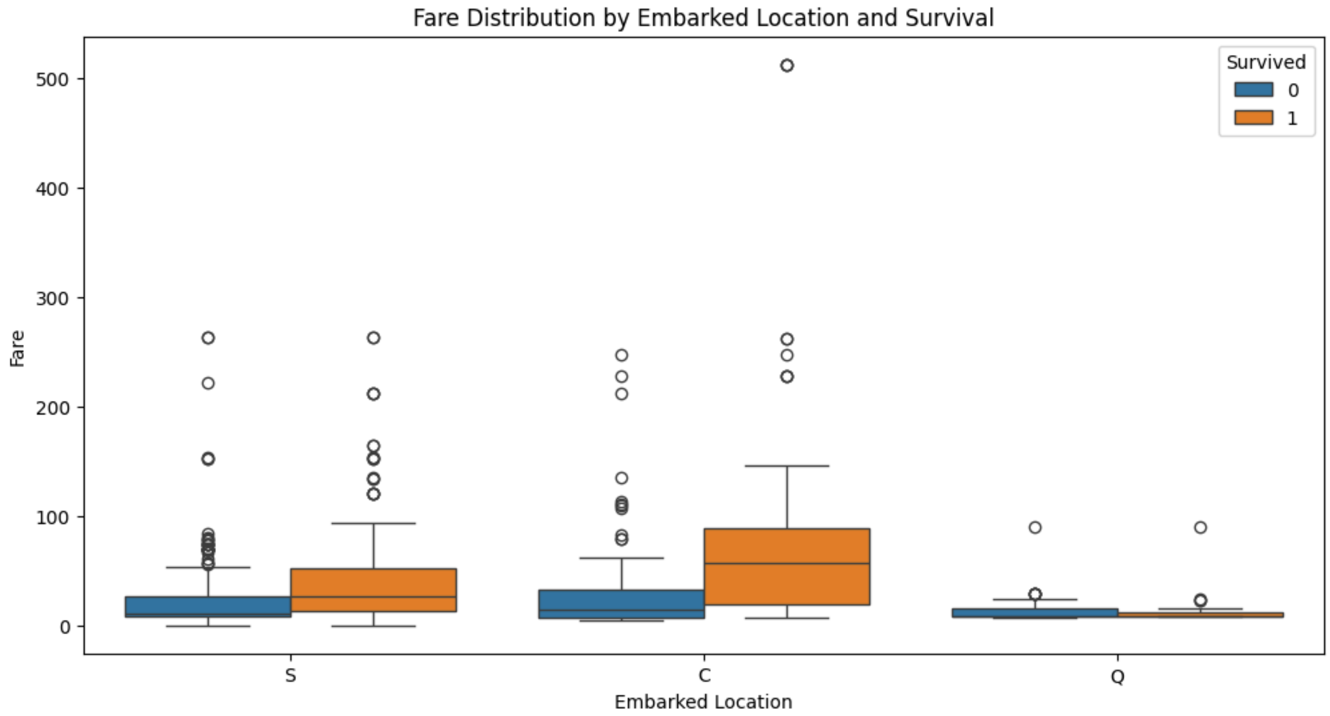
```
plt.figure(figsize=(12, 6))
sns.barplot(x='Embarked', y='Survived', hue='Pclass', data=titanic_df)
plt.title('Survival Rate by Embarked and Passenger Class')
plt.xlabel('Embarked Location')
plt.ylabel('Survival Rate')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future v
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future v
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future v
  data_subset = grouped_data.get_group(pd_key)
```



```
plt.figure(figsize=(12, 6))
sns.boxplot(x='Embarked', y='Fare', hue='Survived', data=titanic_df)
plt.title('Fare Distribution by Embarked Location and Survival')
plt.xlabel('Embarked Location')
plt.ylabel('Fare')
plt.show()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future \
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWarning: SeriesGroupBy.grouper is deprecated and will be removed in a future version of pandas.
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need to pass a length-1 tuple to get_group in a future \
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWarning: SeriesGroupBy.grouper is deprecated and will be removed in a future version of pandas.
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
```



## Define features (X) and target (y)

```
X = titanic_df.drop('Survived', axis=1)
y = titanic_df['Survived']
```

## Convert categorical features to numerical using one-hot encoding

```
X = pd.get_dummies(X, columns=['Sex', 'Embarked', 'Title', 'AgeGroup', 'FareRange'], dummy_na=True)
```