In [1]:

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

In [2]:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [4]:

```
df = pd.read_csv("C:\\Users\\waghm\\OneDrive\\Desktop\\Titanic datase.csv")
```

In [5]:

```
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
```

In [6]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [7]:

```
df.sample(5)
```

Out[7]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
607	608	1	1	Daniel, Mr. Robert Williams	male	27.0	0	0	113804	30.5000
44	45	1	3	Devaney, Miss. Margaret Delia	female	19.0	0	0	330958	7.8792
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	263.0000
688	689	0	3	Fischer, Mr. Eberhard Thelander	male	18.0	0	0	350036	7.7958
601	602	0	3	Slabenoff, Mr. Petco	male	NaN	0	0	349214	7.8958

In [8]:

```
df = df.drop(columns = ['PassengerId' , 'Name' , 'Ticket' , 'Cabin'])
df.head(5)
```

Out[8]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S

In [9]:

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

Out[9]:

```
Survived      0
Pclass        0
Sex           0
Age         177
SibSp         0
Parch         0
Fare          0
Embarked      2
dtype: int64
```

In [10]:

```
df.describe()
```

Out[10]:

	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [11]:

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

Out[11]:

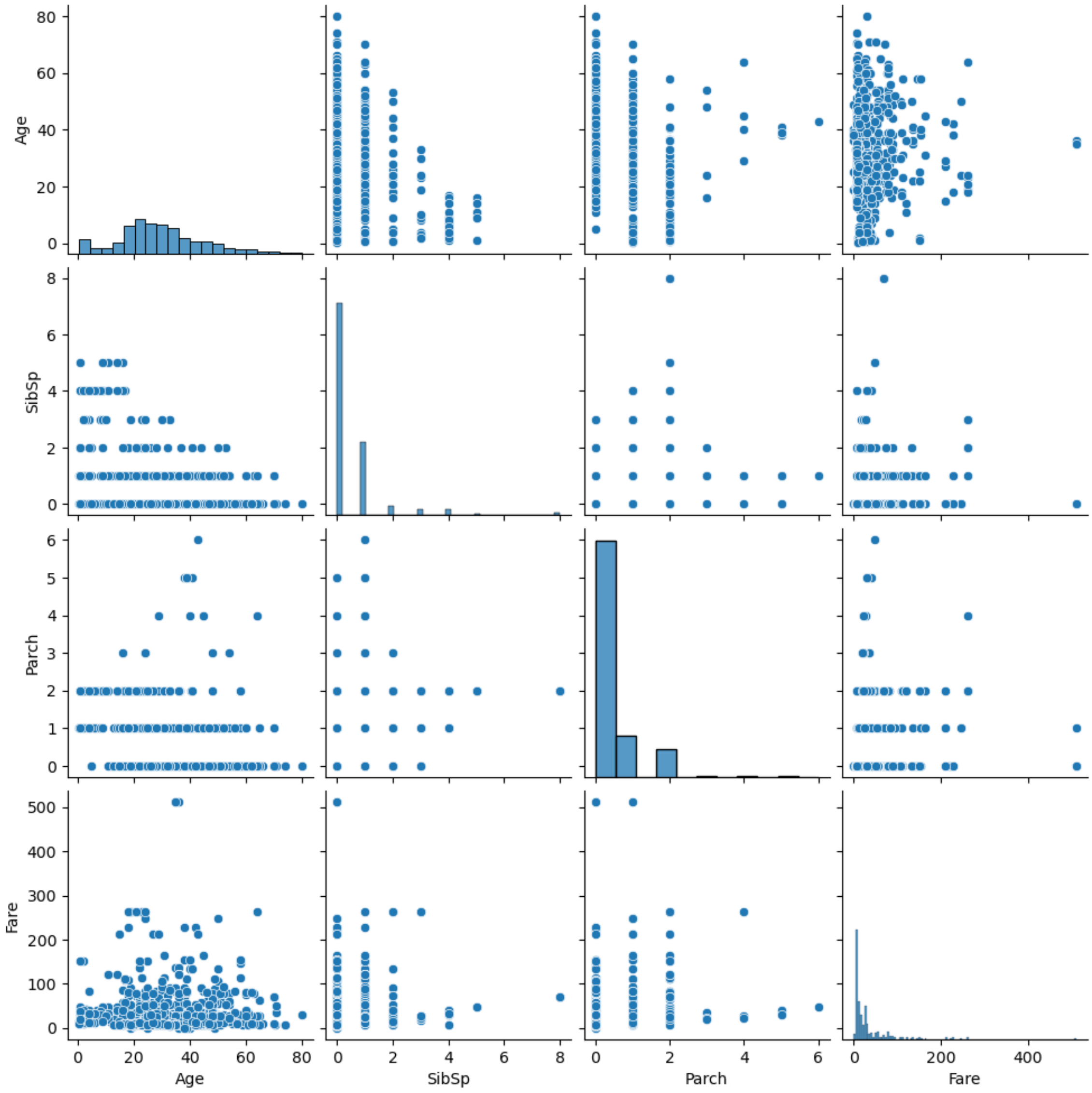
```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

In [12]:

```
sns.pairplot(df[['Age','SibSp','Parch','Fare']])
```

Out[12]:

```
<seaborn.axisgrid.PairGrid at 0x1abac772250>
```



In [13]:

```
y = df.iloc[:,:1]
y.shape
```

Out[13]:

```
(891, 1)
```

In [14]:

```
X = df.iloc[:,1:]
X.shape
```

Out[14]:

```
(891, 7)
```

In [15]:

```
X_train , X_test , y_train , y_test = train_test_split(X,y , random_state = 0 , test_siz
```

In [16]:

```
transformer  = ColumnTransformer(transformers = [
    ('tf1' , SimpleImputer(), ['Age'] ),
    ('tf2' , OneHotEncoder(sparse = False , drop = 'first' , dtype = np.int32) , ['Sex'
] , remainder = 'passthrough')
```

In [17]:

```
X_train_transformed = transformer.fit_transform(X_train)
X_test_transformed = transformer.transform(X_test)
```

In [18]:

```
X_train_transformed.shape , X_test_transformed.shape
```

Out[18]:

```
((623, 9), (268, 9))
```

In [19]:

```
X_train_transformed = pd.DataFrame(X_train_transformed)
X_test_transformed = pd.DataFrame(X_test_transformed)
```

In [20]:

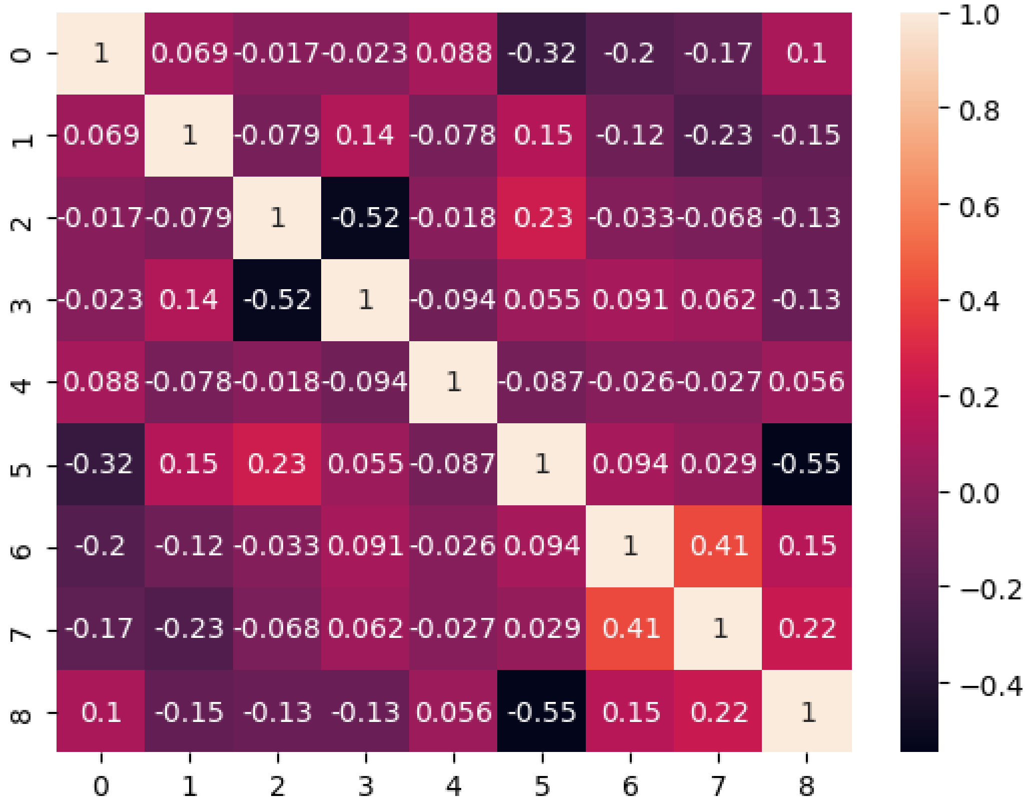
```
X_train_transformed.head()
```

Out[20]:

	0	1	2	3	4	5	6	7	8
0	51.000000	1.0	0.0	1.0	0.0	1.0	0.0	0.0	26.5500
1	49.000000	0.0	0.0	0.0	0.0	1.0	1.0	0.0	76.7292
2	1.000000	1.0	0.0	1.0	0.0	3.0	5.0	2.0	46.9000
3	54.000000	1.0	0.0	1.0	0.0	1.0	0.0	1.0	77.2875
4	29.915339	0.0	0.0	0.0	0.0	3.0	1.0	0.0	14.4583

In [35]:

```
sns.heatmap(X_train_transformed.corr(), annot = True)
plt.show()
```



In [23]:

```
scaler = StandardScaler()
```

In [24]:

```
X_train_scaled = scaler.fit_transform(X_train_transformed)
X_test_scaled = scaler.transform(X_test_transformed)
```

In [25]:

```
X_train_scaled = pd.DataFrame(X_train_scaled)
X_test_scaled = pd.DataFrame(X_test_scaled)
```

In [26]:

```
np.round(X_train.describe() , 1)
```

Out[26]:

	Pclass	Age	SibSp	Parch	Fare
count	623.0	502.0	623.0	623.0	623.0
mean	2.3	29.9	0.5	0.4	32.5
std	0.8	14.5	1.2	0.8	48.3
min	1.0	0.7	0.0	0.0	0.0
25%	1.5	21.0	0.0	0.0	7.9
50%	3.0	29.0	0.0	0.0	15.0
75%	3.0	38.0	1.0	0.0	31.4
max	3.0	80.0	8.0	6.0	512.3

In [27]:

```
np.round(X_train_scaled.describe() , 1)
```

Out[27]:

	0	1	2	3	4	5	6	7	8
count	623.0	623.0	623.0	623.0	623.0	623.0	623.0	623.0	623.0
mean	-0.0	-0.0	0.0	0.0	-0.0	0.0	0.0	-0.0	0.0
std	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
min	-2.2	-1.4	-0.3	-1.7	-0.1	-1.5	-0.5	-0.5	-0.7
25%	-0.5	-1.4	-0.3	-1.7	-0.1	-0.9	-0.5	-0.5	-0.5
50%	-0.0	0.7	-0.3	0.6	-0.1	0.8	-0.5	-0.5	-0.4
75%	0.5	0.7	-0.3	0.6	-0.1	0.8	0.4	-0.5	-0.0
max	3.8	0.7	3.2	0.6	17.6	0.8	6.4	6.7	10.0

In [28]:

```
model = LogisticRegression()
```

In [36]:

```
model.fit(X_train_scaled , y_train)
plt.show()
```

In [30]:

```
y_pred = model.predict(X_test_scaled)
y_pred
```

Out[30]:

```
array([0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,
       1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0,
       1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0,
       1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1,
       0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1], dtype=int64)
```

In [31]:

```
from sklearn.metrics import accuracy_score
```

In [32]:

```
print('Accuracy Score:' , accuracy_score(y_test , y_pred)*100)
```

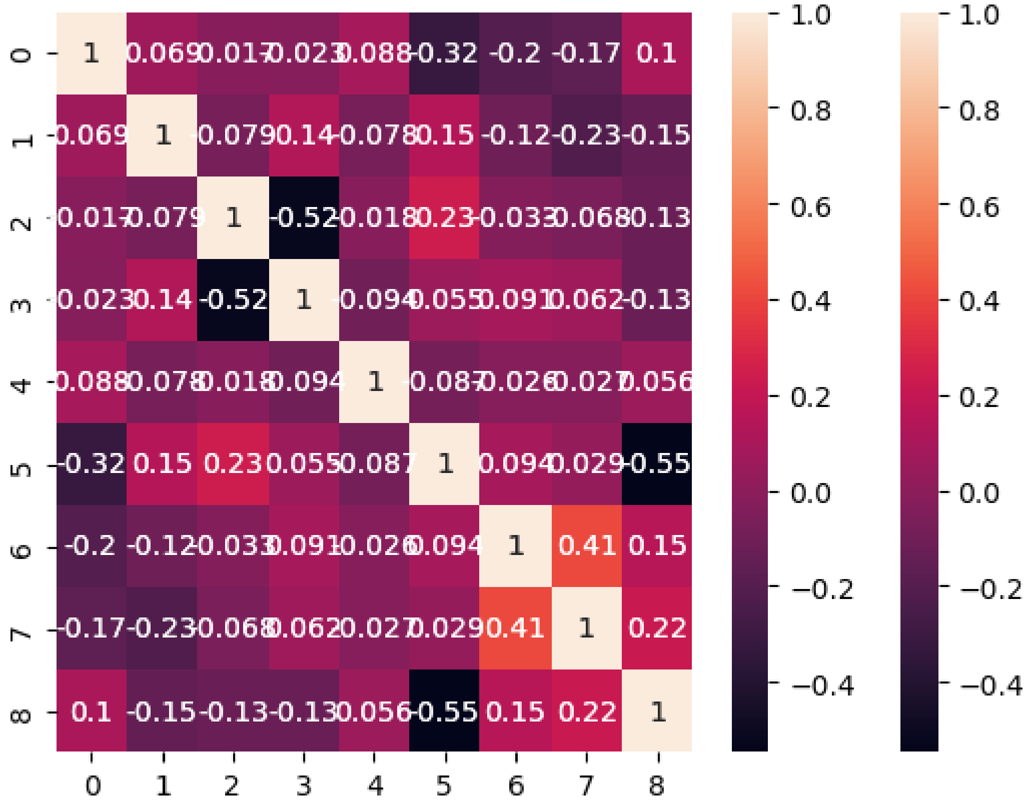
```
Accuracy Score: 79.47761194029852
```

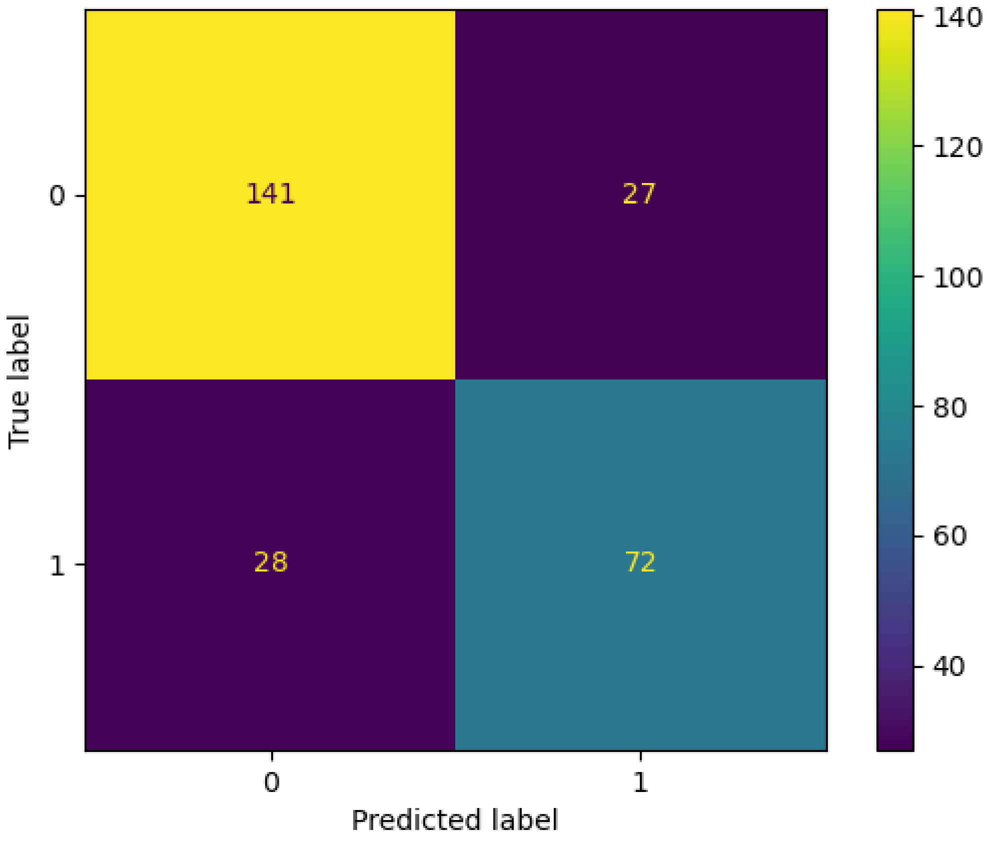
In [33]:

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
```

In [34]:

```
cm = confusion_matrix(y_test , y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix = cm)
disp.plot()
plt.show()
```





In [37]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [38]:

```
model2 = DecisionTreeClassifier()
```

In [39]:

```
model2.fit(X_train_transformed , y_train)
```

Out[39]:

```
DecisionTreeClassifier()
```

In [40]:

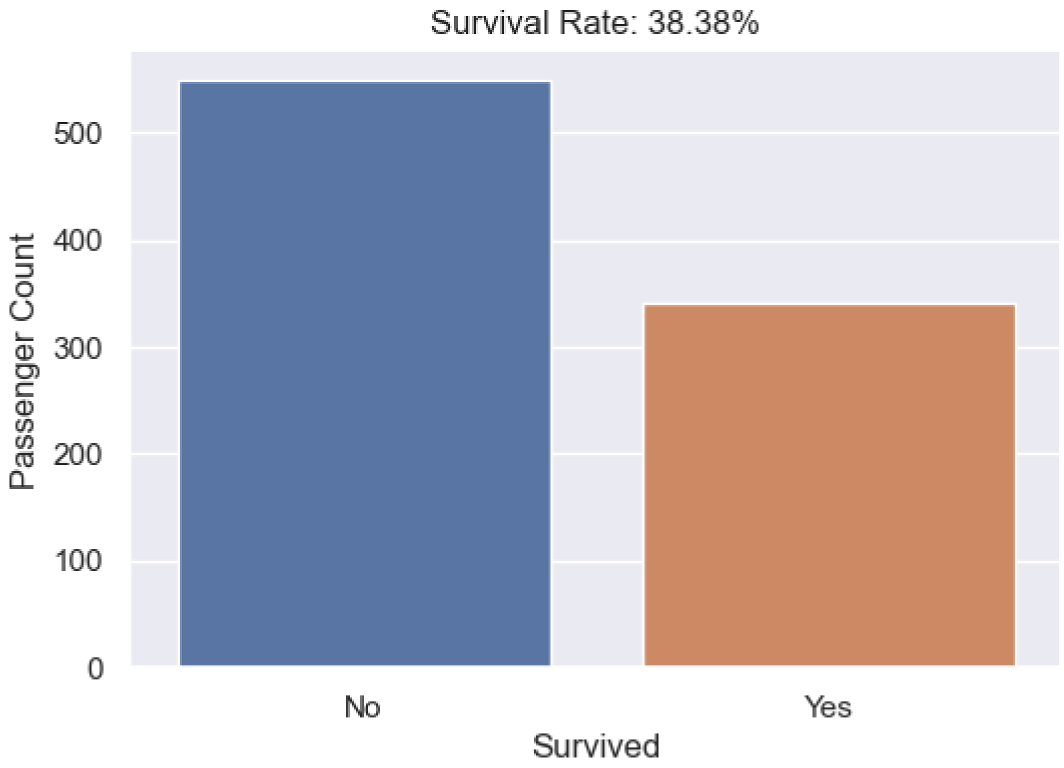
```
y_pred1 = model2.predict(X_test_transformed)
y_pred1
```

Out[40]:

```
array([0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0,
       1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1,
       0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1,
       0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0], dtype=int64)
```

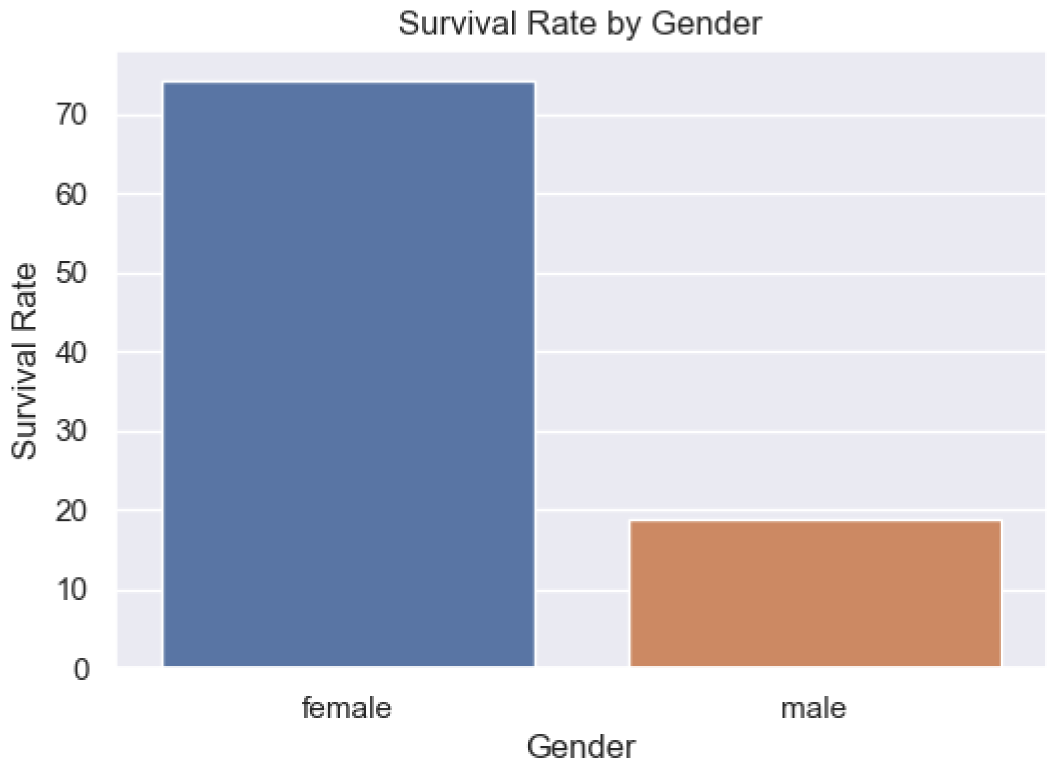
In [41]:

```
# Calculate the overall survival rate
survival_rate = df['Survived'].mean() * 100

# Create a bar plot to visualize the survival rate
sns.set(style='darkgrid')
plt.figure(figsize=(6, 4))
sns.countplot(x='Survived', data=df)
plt.xlabel('Survived')
plt.ylabel('Passenger Count')
plt.title('Survival Rate: {:.2f}%'.format(survival_rate))
plt.xticks([0, 1], ['No', 'Yes'])
plt.show()
```



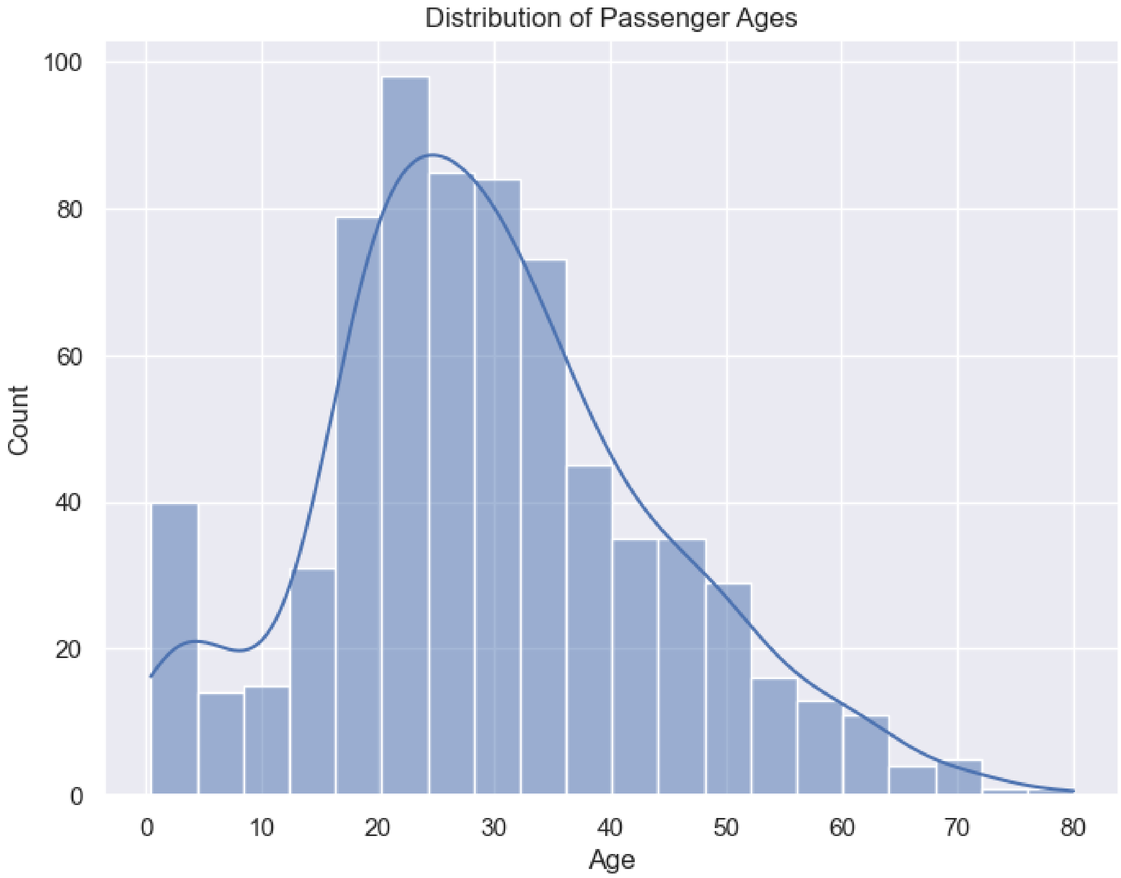
In [42]:

```
# Calculate the survival rate by gender
survival_by_gender = df.groupby('Sex')['Survived'].mean() * 100

# Create a bar plot to visualize the survival rate by gender
sns.set(style='darkgrid')
plt.figure(figsize=(6, 4))
sns.barplot(x=survival_by_gender.index, y=survival_by_gender.values)
plt.xlabel('Gender')
plt.ylabel('Survival Rate')
plt.title('Survival Rate by Gender')
plt.show()
```



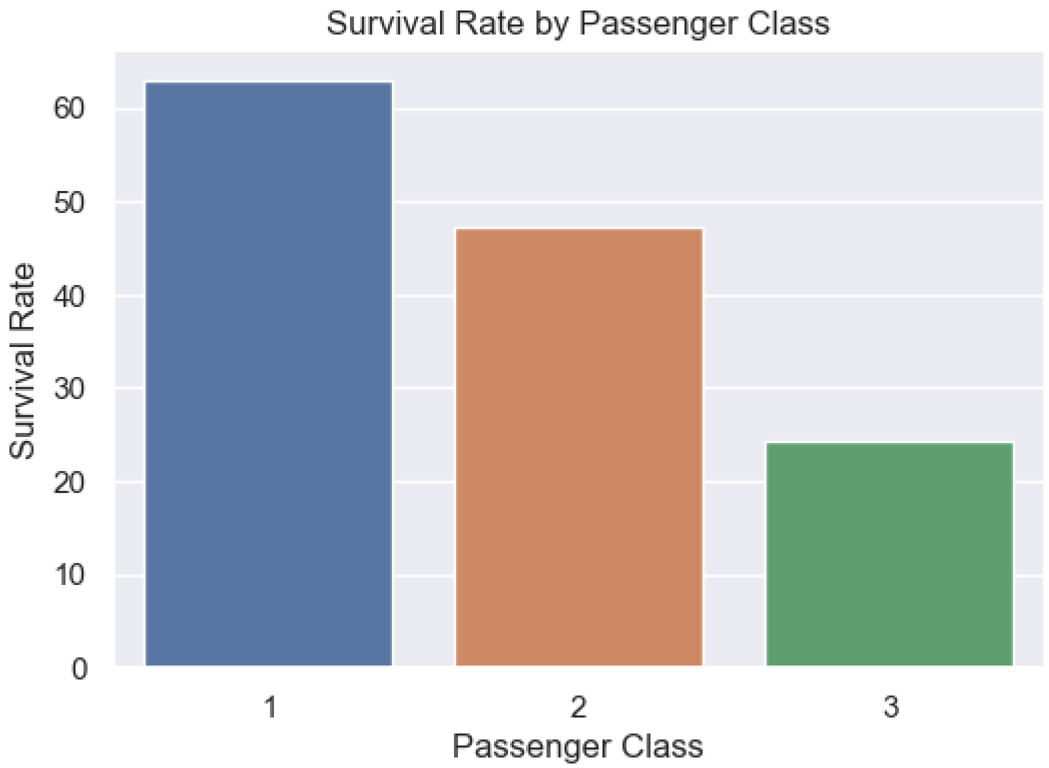
In [43]:

```
# Plot the distribution of passenger ages
sns.set(style='darkgrid')
plt.figure(figsize=(8, 6))
sns.histplot(data=df, x='Age', bins=20, kde=True)
plt.xlabel('Age')
plt.ylabel('Count')
plt.title('Distribution of Passenger Ages')
plt.show()
```



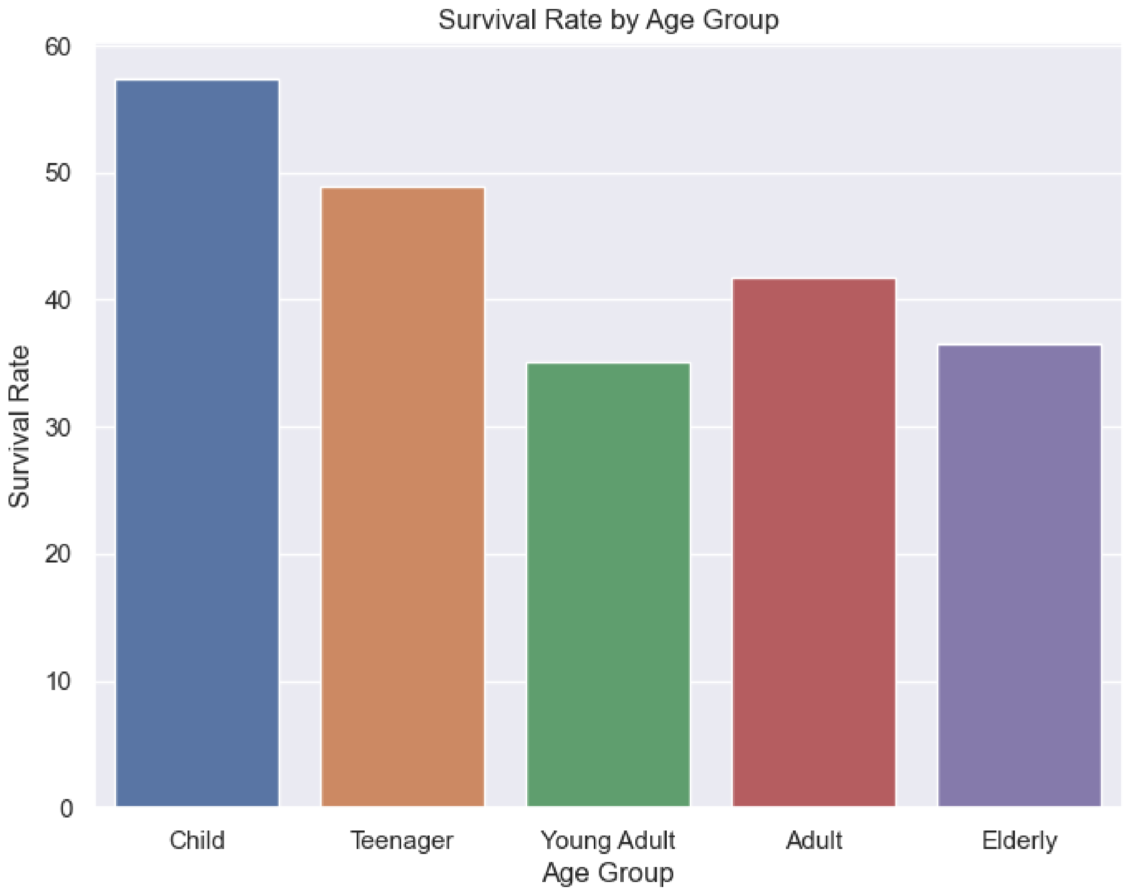
In [44]:

```
# Calculate the survival rates by passenger class
survival_by_class = df.groupby('Pclass')['Survived'].mean() * 100

# Create a bar plot to visualize the survival rates by passenger class
sns.set(style='darkgrid')
plt.figure(figsize=(6, 4))
sns.barplot(x=survival_by_class.index, y=survival_by_class.values)
plt.xlabel('Passenger Class')
plt.ylabel('Survival Rate')
plt.title('Survival Rate by Passenger Class')
plt.show()
```



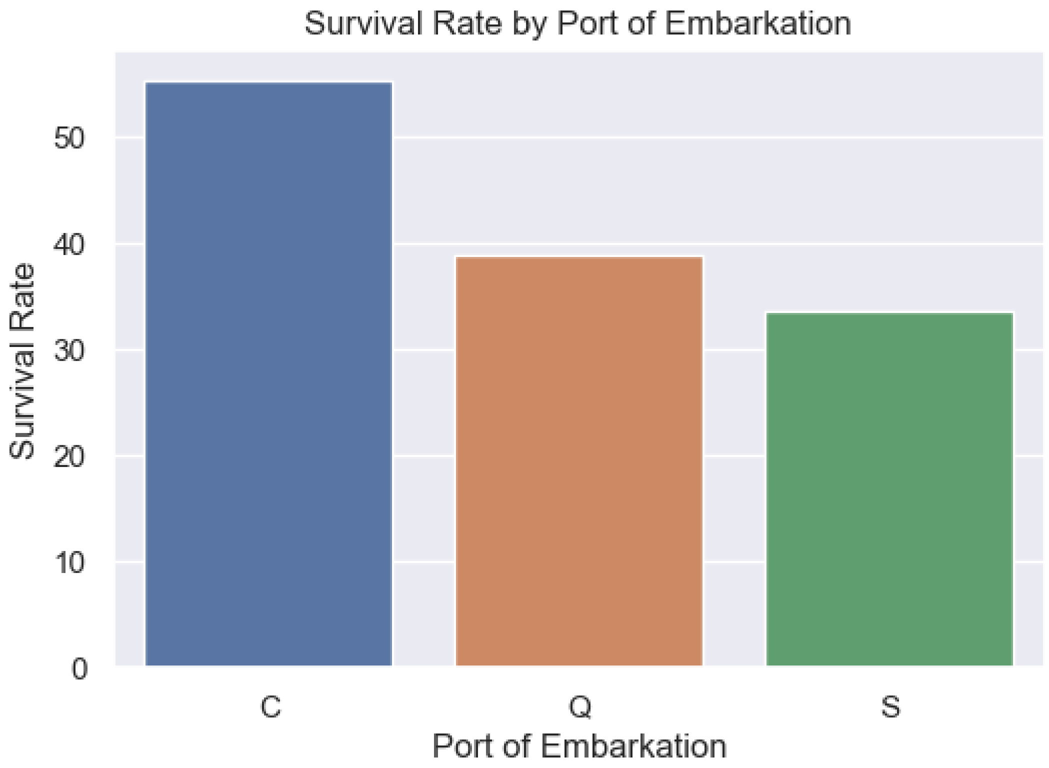
In [47]:

```
#Create age groups
age_bins = [0, 12, 18, 30, 50, 100]  # Define the age group boundaries
age_labels = ['Child', 'Teenager', 'Young Adult', 'Adult', 'Elderly']  # Define the age
df['AgeGroup'] = pd.cut(df['Age'], bins=age_bins, labels=age_labels, right=False)

# Calculate the survival rates by age group
survival_by_age_group = df.groupby('AgeGroup')['Survived'].mean() * 100

# Create a bar plot to visualize the survival rates by age group
sns.set(style='darkgrid')
plt.figure(figsize=(8, 6))
sns.barplot(x=survival_by_age_group.index, y=survival_by_age_group.values)
plt.xlabel('Age Group')
plt.ylabel('Survival Rate')
plt.title('Survival Rate by Age Group')
plt.show()
```



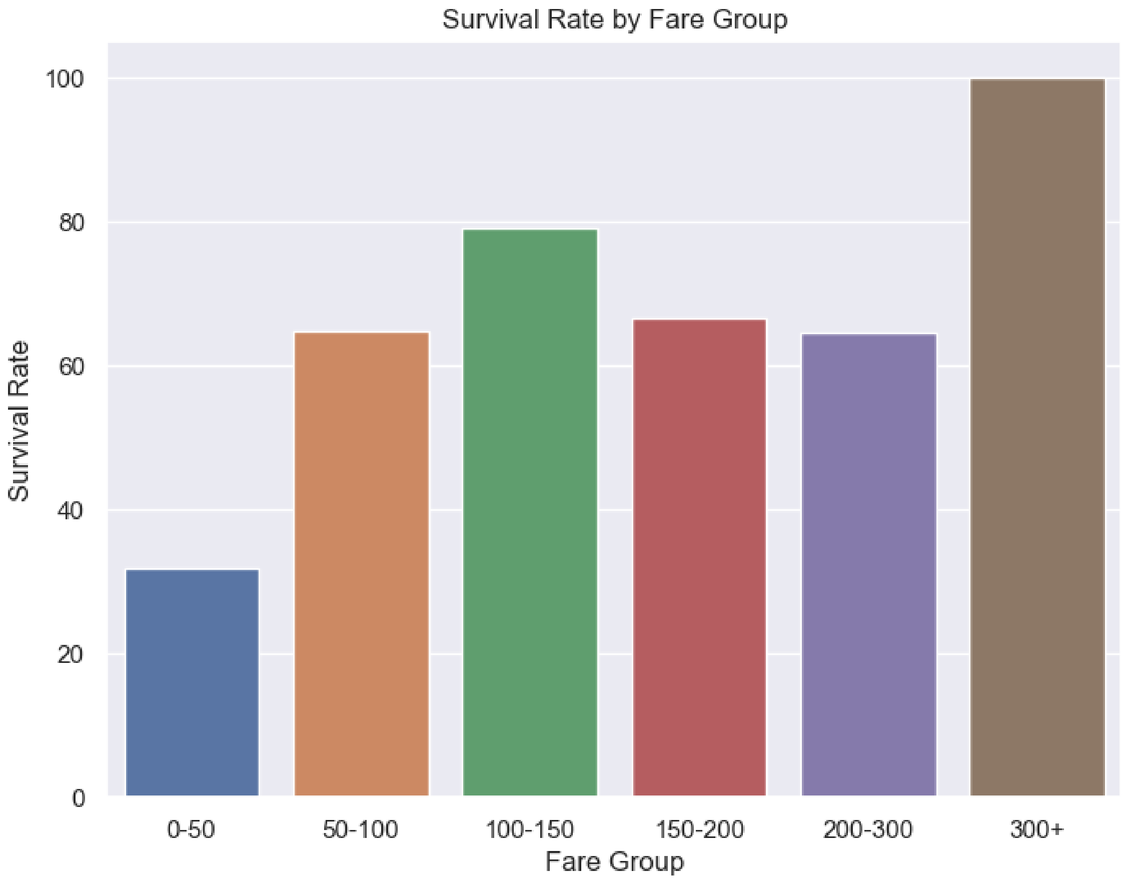
In [48]:

```
# Calculate the survival rates by port of embarkation
survival_by_embarkation = df.groupby('Embarked')['Survived'].mean() * 100

# Create a bar plot to visualize the survival rates by port of embarkation
sns.set(style='darkgrid')
plt.figure(figsize=(6, 4))
sns.barplot(x=survival_by_embarkation.index, y=survival_by_embarkation.values)
plt.xlabel('Port of Embarkation')
plt.ylabel('Survival Rate')
plt.title('Survival Rate by Port of Embarkation')
plt.show()
```



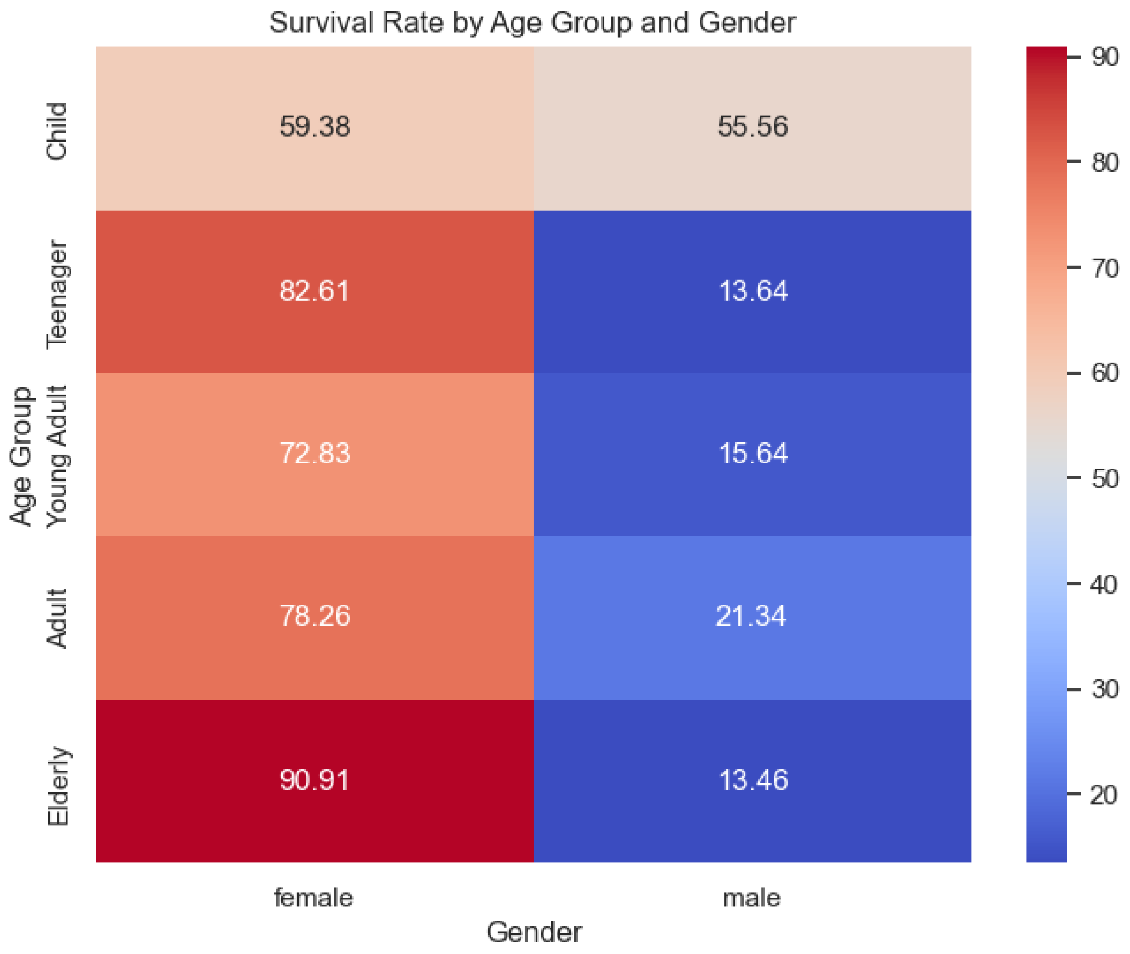
In [49]:

```
# Create fare groups
fare_bins = [0, 50, 100, 150, 200, 300, 1000]  # Define the fare group boundaries
fare_labels = ['0-50', '50-100', '100-150', '150-200', '200-300', '300+']  # Define the
df['FareGroup'] = pd.cut(df['Fare'], bins=fare_bins, labels=fare_labels, right=False)

# Calculate the survival rates by fare group
survival_by_fare_group = df.groupby('FareGroup')['Survived'].mean() * 100

# Create a bar plot to visualize the survival rates by fare group
sns.set(style='darkgrid')
plt.figure(figsize=(8, 6))
sns.barplot(x=survival_by_fare_group.index, y=survival_by_fare_group.values)
plt.xlabel('Fare Group')
plt.ylabel('Survival Rate')
plt.title('Survival Rate by Fare Group')
plt.show()
```



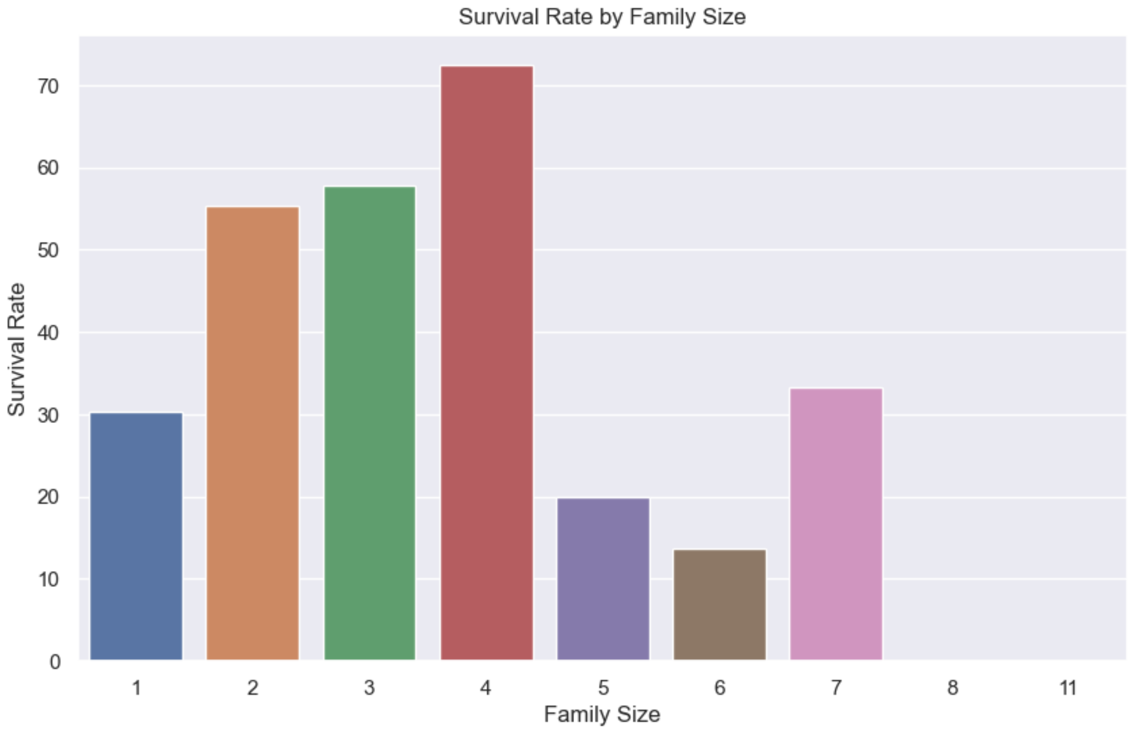
In [51]:

```
# Create age groups
age_bins = [0, 12, 18, 30, 50, 100]  # Define the age group boundaries
age_labels = ['Child', 'Teenager', 'Young Adult', 'Adult', 'Elderly']  # Define the age
df['AgeGroup'] = pd.cut(df['Age'], bins=age_bins, labels=age_labels, right=False)

# Calculate the survival rates by age group and gender
survival_by_age_gender = df.groupby(['AgeGroup', 'Sex'])['Survived'].mean() * 100

# Convert the survival rates into a pivot table for easier visualization
survival_pivot = survival_by_age_gender.unstack()

# Create a heatmap to visualize the survival rates by age group and gender
sns.set(style='darkgrid')
plt.figure(figsize=(8, 6))
sns.heatmap(data=survival_pivot, annot=True, cmap='coolwarm', fmt=".2f", cbar=True)
plt.xlabel('Gender')
plt.ylabel('Age Group')
plt.title('Survival Rate by Age Group and Gender')
plt.show()
```



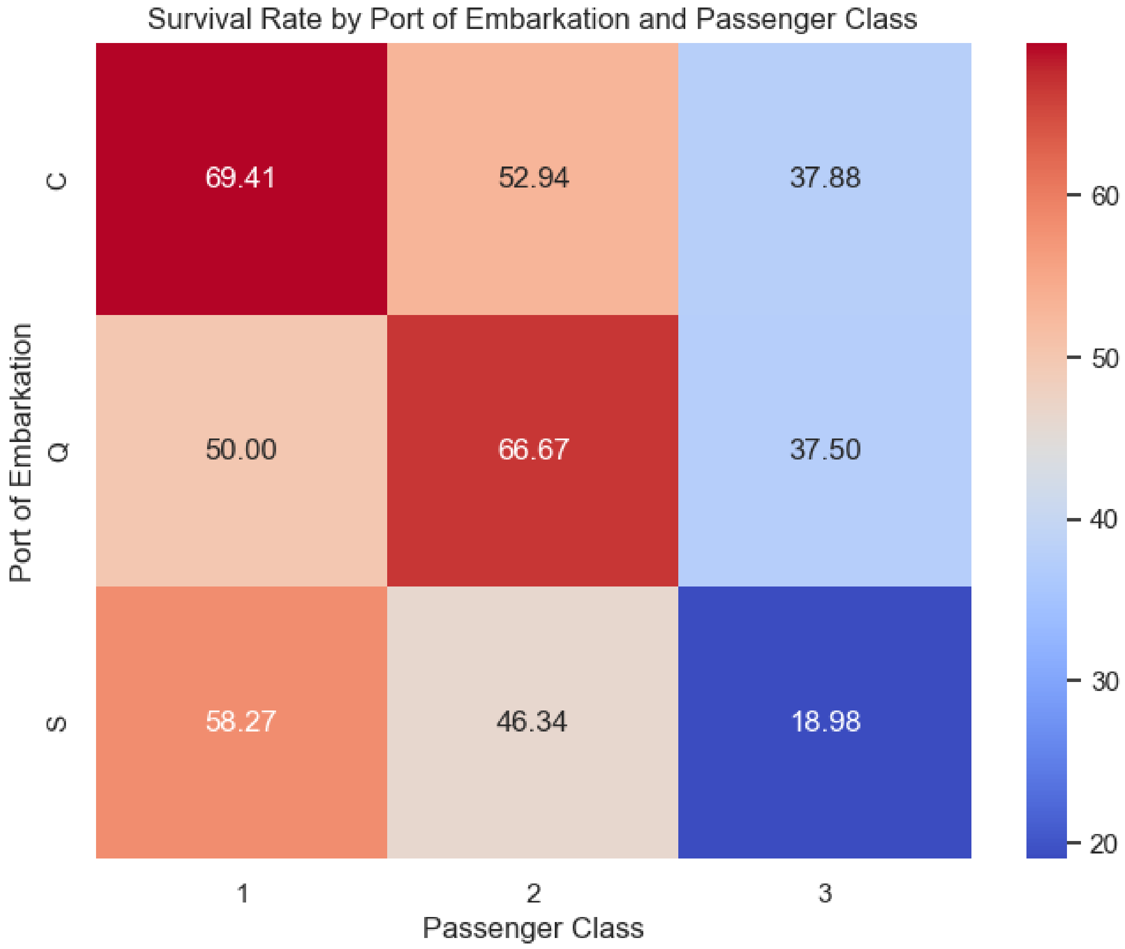
In [52]:

```
# Calculate the total number of family members for each passenger
df['FamilySize'] = df['SibSp'] + df['Parch'] + 1

# Calculate the survival rates by family size
survival_by_family_size = df.groupby('FamilySize')['Survived'].mean() * 100

# Create a bar plot to visualize the survival rates by family size
sns.set(style='darkgrid')
plt.figure(figsize=(10, 6))
sns.barplot(x=survival_by_family_size.index, y=survival_by_family_size.values)
plt.xlabel('Family Size')
plt.ylabel('Survival Rate')
plt.title('Survival Rate by Family Size')
plt.show()
```



In [53]:

```
# Calculate the survival rates by port of embarkation and passenger class
survival_by_embark_class = df.groupby(['Embarked', 'Pclass'])['Survived'].mean() * 100

# Convert the survival rates into a pivot table for easier visualization
survival_pivot = survival_by_embark_class.unstack()

# Create a heatmap to visualize the survival rates
sns.set(style='darkgrid')
plt.figure(figsize=(8, 6))
sns.heatmap(data=survival_pivot, annot=True, cmap='coolwarm', fmt=".2f", cbar=True)
plt.xlabel('Passenger Class')
plt.ylabel('Port of Embarkation')
plt.title('Survival Rate by Port of Embarkation and Passenger Class')
plt.show()
```



In [ ]: