```
# Load dataset
import pandas as pd
import numpy as np

df_titanic = pd.read_csv("/content/Titanic-Dataset.csv")

# Display first 5 rows
print("\nTitanic Dataset:\n", df_titanic.head())

# Check for missing values
print("\nMissing Values:\n", df_titanic.isnull().sum())

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

# Calculate survival rate by class
survival_by_class = df_titanic.groupby('Pclass')['Survived'].mean()
print("\nSurvival Rate by Passenger Class:\n", survival_by_class)

# Find the average age of survivors vs non-survivors
average_age = df_titanic.groupby('Survived')['Age'].mean()
print("\nAverage Age of Survivors vs Non-Survivors:\n", average_age)

# Sorting by Fare
sorted_fare = df_titanic.sort_values(by='Fare', ascending=False).head(10)
print("\nTop 10 Passengers with Highest Fare:\n", sorted_fare[['Name', 'Fare']])
```

```
                        Heikkinen, Miss. Laina  female  26.0      0
      Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
                       Allen, Mr. William Henry    male  35.0      0

'arch            Ticket     Fare Cabin Embarked
   0         A/5 21171   7.2500   NaN        S
   0          PC 17599  71.2833   C85        C
   0  STON/O2. 3101282   7.9250   NaN        S
   0            113803  53.1000  C123        S
   0            373450   8.0500   NaN        S

ing Values:
sengerId      0
ived         0
ss           0
             0
             0
           177
p            0
h            0
et           0
             0
n          687
rked         2
e: int64

ival Rate by Passenger Class:
ass
  0.629630
  0.472826
  0.242363
: Survived, dtype: float64

age Age of Survivors vs Non-Survivors:
vived
  30.028233
  28.291433
: Age, dtype: float64

 10 Passengers with Highest Fare:




thon-input-5-85157ac4ce43>:14: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignmen
```

```
behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values

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

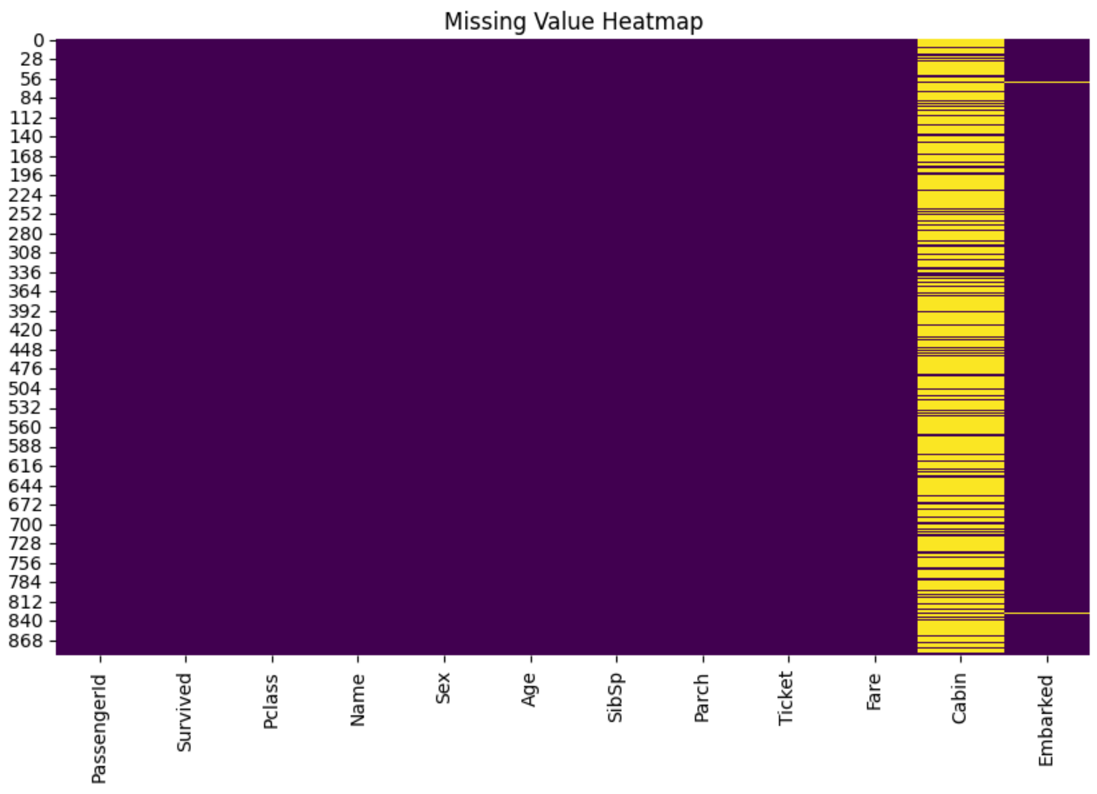
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

```
import pandas as pd
df_titanic = pd.read_csv("/content/Titanic-Dataset.csv")
print("\nInfo:")
print(df_titanic.info())

print("\nDescription:")
print(df_titanic.describe())

print("\nValue Counts (Survived):")
print(df_titanic['Survived'].value_counts())

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

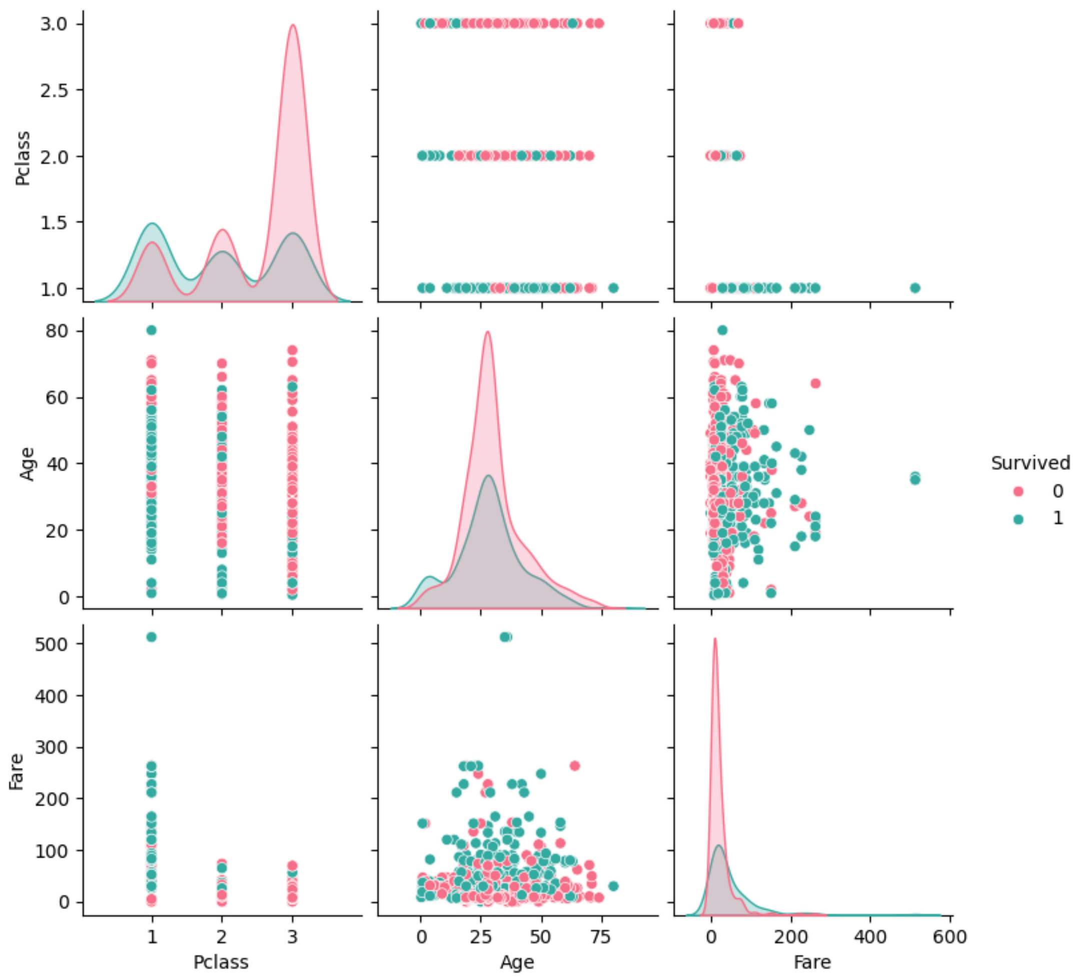
```
Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    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

Description:
       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

Value Counts (Survived):
Survived
0    549
1    342
Name: count, dtype: int64
<ipython-input-6-c91099ca056d>:13: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assig
```

```
    The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting va

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


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

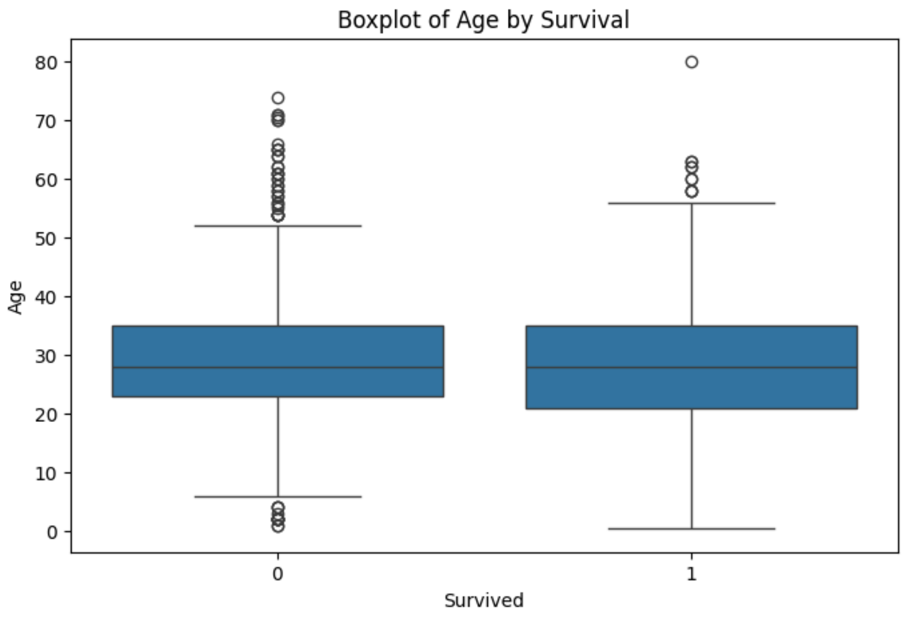
```
# Heatmap of missing values
plt.figure(figsize=(10,6))
sns.heatmap(df_titanic.isnull(), cbar=False, cmap='viridis')
plt.title("Missing Value Heatmap")
plt.show()
```



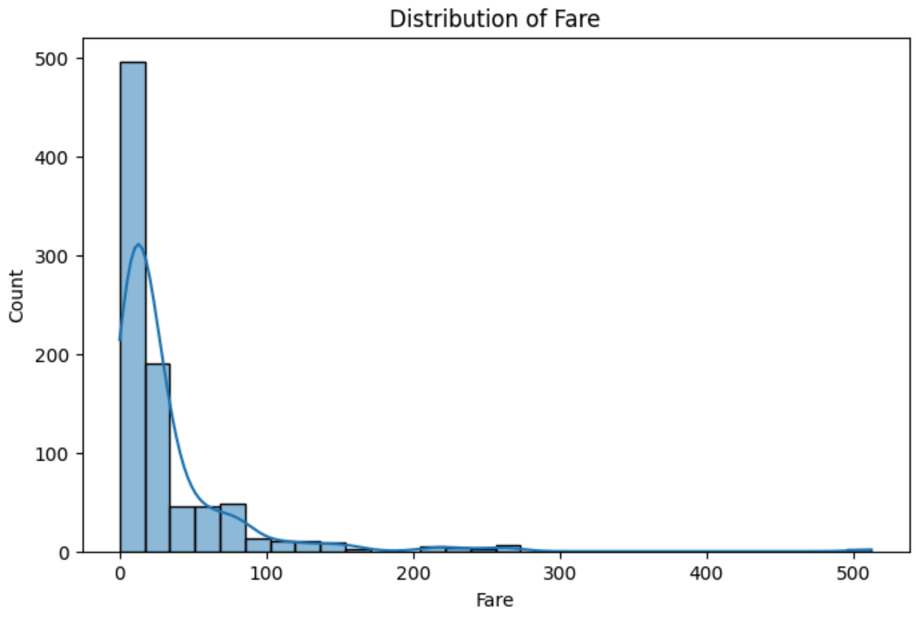
```
# Pairplot: Visualize relationships
sns.pairplot(df_titanic[['Survived', 'Pclass', 'Sex', 'Age', 'Fare']], hue='Survived', palette='husl')
plt.show()
```



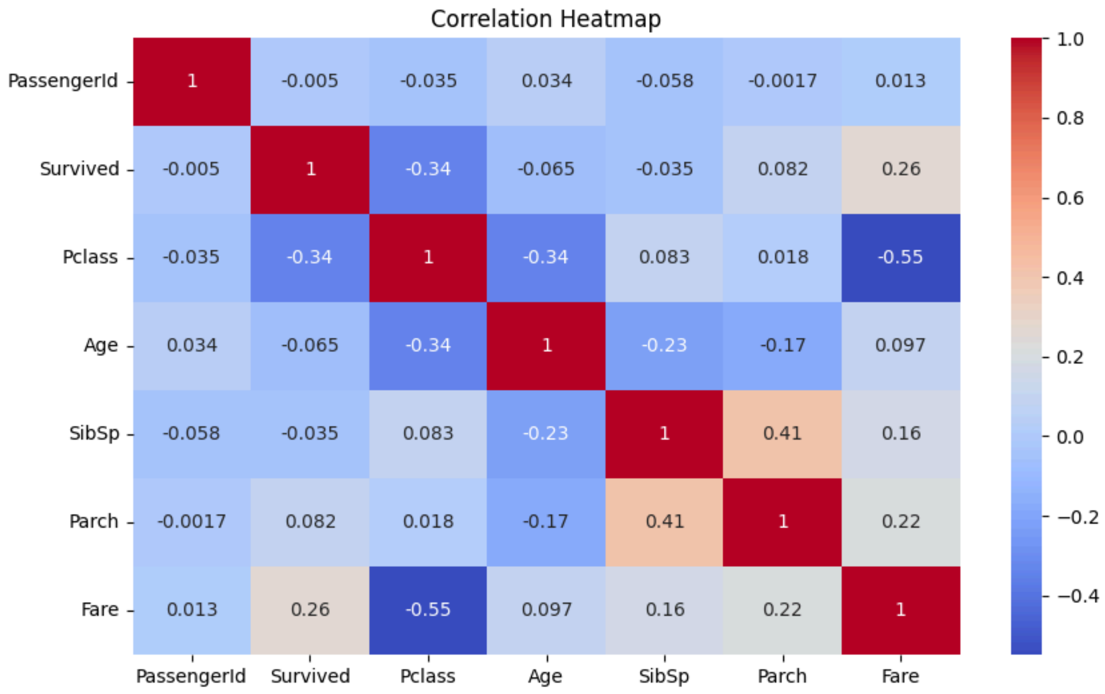
```
# Boxplot - Age vs Survived
plt.figure(figsize=(8,5))
sns.boxplot(x='Survived', y='Age', data=df_titanic)
plt.title("Boxplot of Age by Survival")
plt.show()
```



```
# Histogram of Fare
plt.figure(figsize=(8,5))
sns.histplot(df_titanic['Fare'], bins=30, kde=True)
plt.title("Distribution of Fare")
plt.show()
```



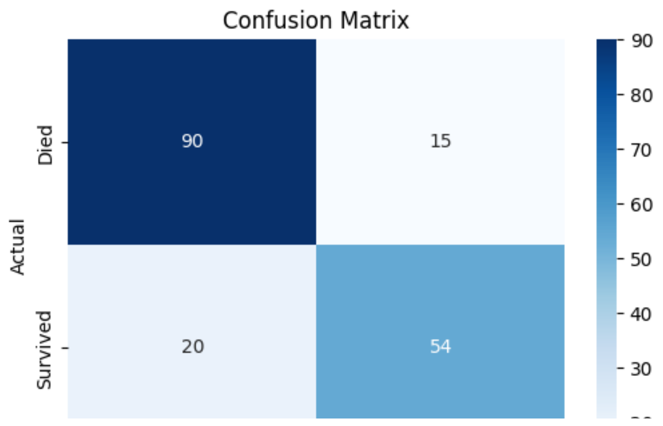
```
# Correlation heatmap
plt.figure(figsize=(10,6))
sns.heatmap(df_titanic.corr(numeric_only=True), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```



	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1	-0.005	-0.035	0.034	-0.058	-0.0017	0.013
Survived	-0.005	1	-0.34	-0.065	-0.035	0.082	0.26
Pclass	-0.035	-0.34	1	-0.34	0.083	0.018	-0.55
Age	0.034	-0.065	-0.34	1	-0.23	-0.17	0.097
SibSp	-0.058	-0.035	0.083	-0.23	1	0.41	0.16
Parch	-0.0017	0.082	0.018	-0.17	0.41	1	0.22
Fare	0.013	0.26	-0.55	0.097	0.16	0.22	1

```
# Encode categorical feature 'Sex'
df_titanic['Sex'] = df_titanic['Sex'].map({'male': 0, 'female': 1})

# Logistic Regression for confusion matrix
X = df_titanic[['Pclass', 'Sex', 'Age', 'Fare']]
y = df_titanic['Survived']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

```
# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Died', 'Survived'], yticklabels=['Died', 'Survived'])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```


Confusion Matrix
Actual
Died
Survived