```
import pandas as pd   # linear algebra
import numpy as np    # data processing, CSV file I/O like (pd.read_csv)
import seaborn as sns  # for data visuvalization
import matplotlib.pyplot as plt

# Avoid Warning
import warnings
warnings.filterwarnings("ignore")



#training data
train_data = pd.read_csv("/content/titanic.csv") # 1) loading the dataset into the pandas data frame
sub_train_data = pd.read_csv("/content/titanic.csv")
train_data.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 `train_data` View recommended plots

```
# Test dataframe
print("Displaying the first few rows of the dataset: ")
sub_train_data.head(2) # here we want to see top 2 rows of sub_train_data
```

```
Displaying the first few rows of the dataset:
```

	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
				Cumings, Mrs. John					PC			

Next steps: Generate code with `sub_train_data` View recommended plots

```
print("summary information of the DataFrame i.e train_data\n")
train_data.info()
```

```
summary information of the DataFrame i.e train_data
```

```
<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
```

```
train_data.shape # will show the dimensions of the data set
```

```
(891, 12)
```

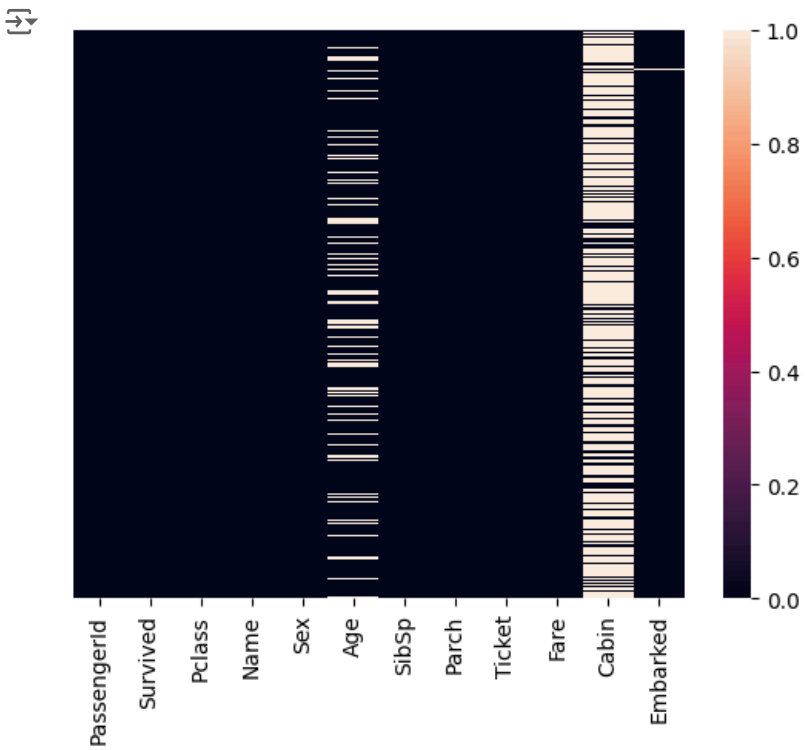
```
print("1. The Above is Data Loading and Inspection above\n")

print("2. Data Cleaning")
```

```
1. The Above is Data Loading and Inspection above

2. Data Cleaning
```

```
# Checking for NaN Values in data frame colunms
train_data.isnull().sum()
100 * train_data.isnull().sum() / len(train_data)  # To get the values in percentage
```

```
PassengerId     0.000000
Survived        0.000000
Pclass          0.000000
Name            0.000000
Sex             0.000000
Age            19.865320
SibSp           0.000000
Parch           0.000000
Ticket          0.000000
Fare            0.000000
Cabin          77.104377
Embarked        0.224467
dtype: float64
```

```
sns.heatmap(train_data.isnull(), yticklabels=False);
```




```
train_data.isnull()  # row wise finding of the NaN values
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	False	False	False	False	False	False	False	False	False	False	True	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	True	False
...	...	...	...	...	...	...	...	...	...	...	...	...
886	False	False	False	False	False	False	False	False	False	False	True	False
887	False	False	False	False	False	False	False	False	False	False	False	False
888	False	False	False	False	False	True	False	False	False	False	True	False
889	False	False	False	False	False	False	False	False	False	False	False	False
890	False	False	False	False	False	False	False	False	False	False	True	False

891 rows × 12 columns

```
# Identifying and list missing values and their percentages
missing_values = train_data.isnull().sum()
missing_percentages = 100 * train_data.isnull().sum() / len(train_data)

missing_table = pd.concat([missing_values, missing_percentages], axis=1, keys=['Missing Values', 'Percentage'])
print(missing_table)
```

```
             Missing Values  Percentage
PassengerId               0    0.000000
Survived                  0    0.000000
Pclass                    0    0.000000
Name                      0    0.000000
Sex                       0    0.000000
Age                     177   19.865320
SibSp                     0    0.000000
Parch                     0    0.000000
Ticket                    0    0.000000
Fare                      0    0.000000
Cabin                   687   77.104377
Embarked                  2    0.224467
```

```
# Droping 'Cabin' column (more than 50% missing)
train_data.drop('Cabin', axis=1, inplace=True)

# Handling missing values in 'Age'
# filling with the median age
train_data['Age'].fillna(train_data['Age'].median(), inplace=True)

# Handling missing values in 'Embarked'
# We'll fill with the mode (most frequent value)
train_data['Embarked'].fillna(train_data['Embarked'].mode()[0], inplace=True)

# Verification of handling all missing values
print(train_data.isnull().sum())
```

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

```
# Converting categorical columns to category type
categorical_columns = ['Pclass', 'Sex', 'Embarked', 'Ticket']
for col in categorical_columns:
    train_data[col] = train_data[col].astype('category')

# Converting 'Survived' column to binary (0 or 1)
train_data['Survived'] = train_data['Survived'].astype(int)

# Converting 'Age' and 'Fare' to float type
train_data['Age'] = train_data['Age'].astype(float)
train_data['Fare'] = train_data['Fare'].astype(float)

# Converting 'SibSp' and 'Parch' to integer type
train_data['SibSp'] = train_data['SibSp'].astype(int)
train_data['Parch'] = train_data['Parch'].astype(int)

# Keeping 'Name' as string (object) type
# Keeping 'PassengerId' as integer type

# Verifying the data types
print(train_data.dtypes)
```

```
PassengerId       int64
Survived          int64
```

```
Pclass         category
Name             object
Sex            category
Age             float64
SibSp             int64
Parch             int64
Ticket         category
Fare            float64
Embarked       category
dtype: object
```

```
# Checking for duplicates
duplicate_count = train_data.duplicated().sum()
print(f"Number of duplicate rows: {duplicate_count}")

# Removing duplicate rows
train_data.drop_duplicates(inplace=True)

# Verifying the removal
new_duplicate_count = train_data.duplicated().sum()
print(f"Number of duplicate rows after removal: {new_duplicate_count}")

# The shape of the DataFrame before and after duplicate removal
print(f"Shape of DataFrame before removing duplicates: {train_data.shape}")
train_data.drop_duplicates(inplace=True)
print(f"Shape of DataFrame after removing duplicates: {train_data.shape}")
```

```
Number of duplicate rows: 0
Number of duplicate rows after removal: 0
Shape of DataFrame before removing duplicates: (891, 11)
Shape of DataFrame after removing duplicates: (891, 11)
```

```
print("3. Exploratory Data Analysis (EDA)")
```

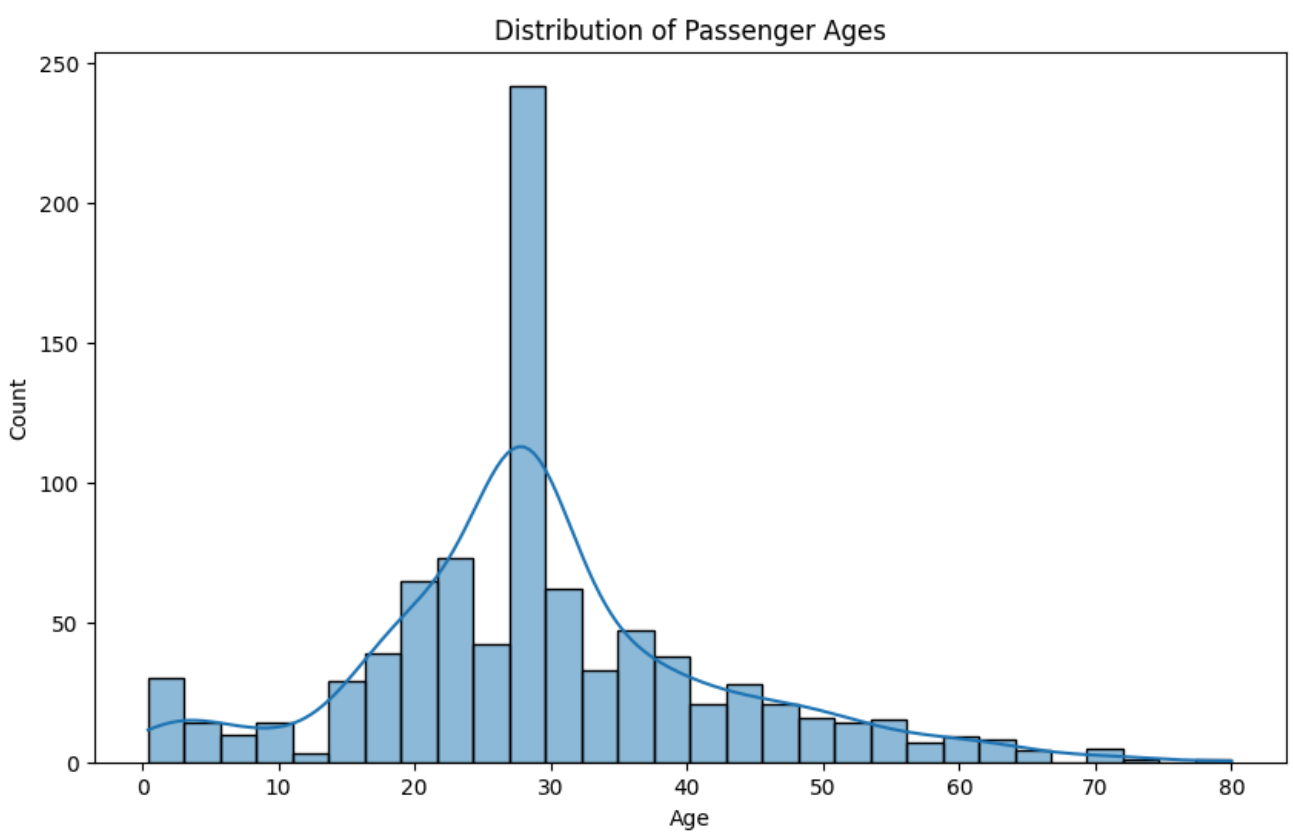
```
3. Exploratory Data Analysis (EDA)
```

```
# Summary statistics for numerical columns
numerical_cols = ['Age', 'Fare', 'SibSp', 'Parch']
summary_stats = train_data[numerical_cols].describe()
print(summary_stats)
```

```
              Age        Fare       SibSp       Parch
count  891.000000  891.000000  891.000000  891.000000
mean    29.361582   32.204208    0.523008    0.381594
std     13.019697   49.693429    1.102743    0.806057
min      0.420000    0.000000    0.000000    0.000000
25%     22.000000    7.910400    0.000000    0.000000
50%     28.000000   14.454200    0.000000    0.000000
75%     35.000000   31.000000    1.000000    0.000000
```

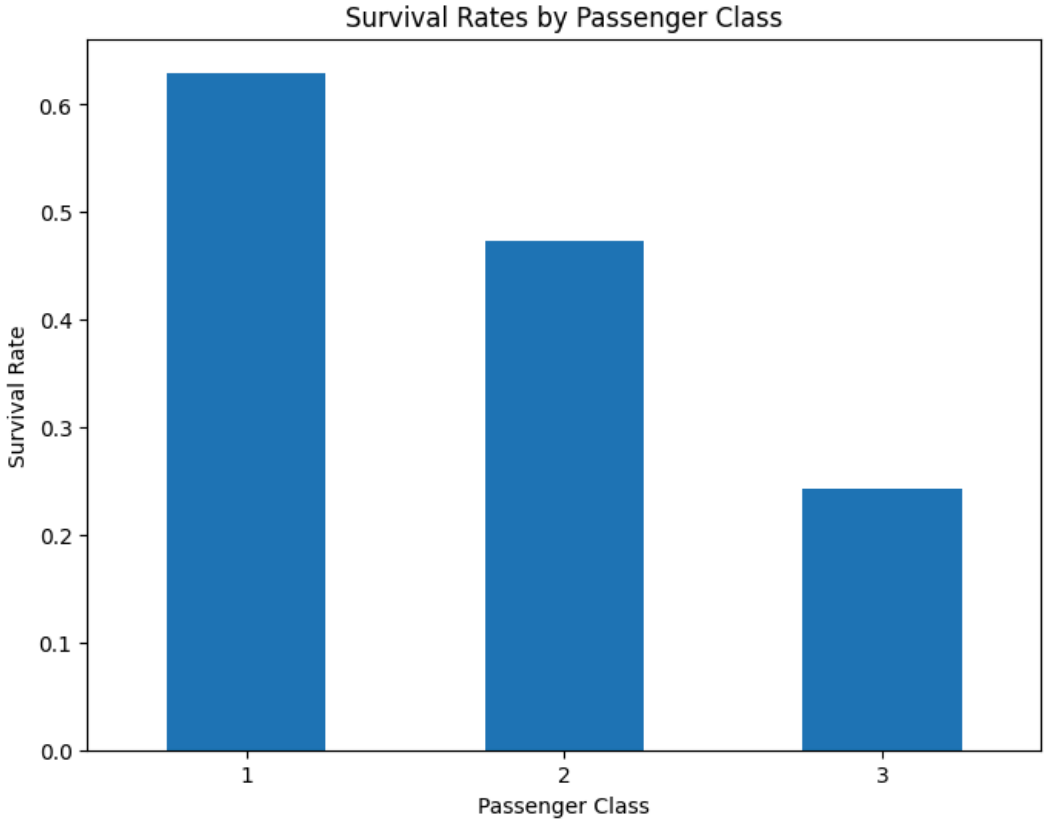
```
max     80.000000  512.329200    8.000000    6.000000
```

```
# Histogram of passenger ages
plt.figure(figsize=(10, 6))
sns.histplot(train_data['Age'], bins=30, kde=True)
plt.title('Distribution of Passenger Ages')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```



```
# Survival rates by passenger class
survival_by_class = train_data.groupby('Pclass')['Survived'].mean()
print("Survival rates by passenger class:")
print(survival_by_class)

# Visualizing survival rates by passenger class
plt.figure(figsize=(8, 6))
survival_by_class.plot(kind='bar')
plt.title('Survival Rates by Passenger Class')
plt.xlabel('Passenger Class')
plt.ylabel('Survival Rate')
plt.xticks(rotation=0)
plt.show()
```

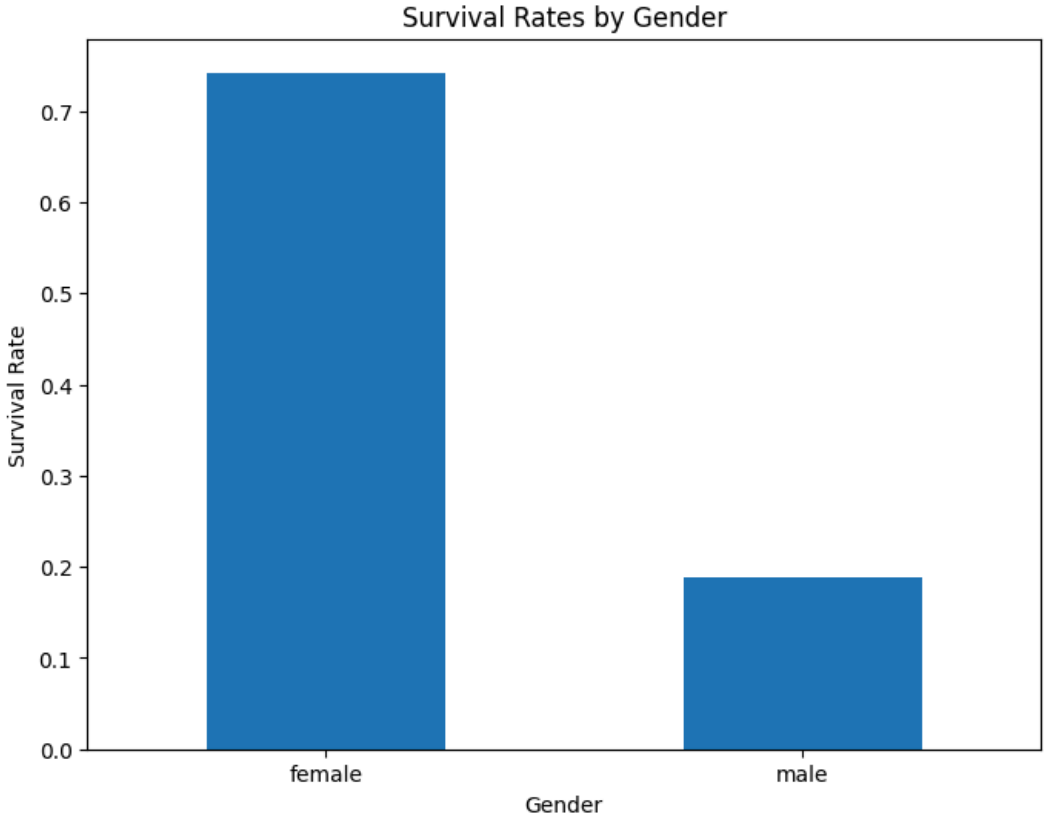
```
Survival rates by passenger class:
Pclass
1    0.629630
2    0.472826
3    0.242363
Name: Survived, dtype: float64
```



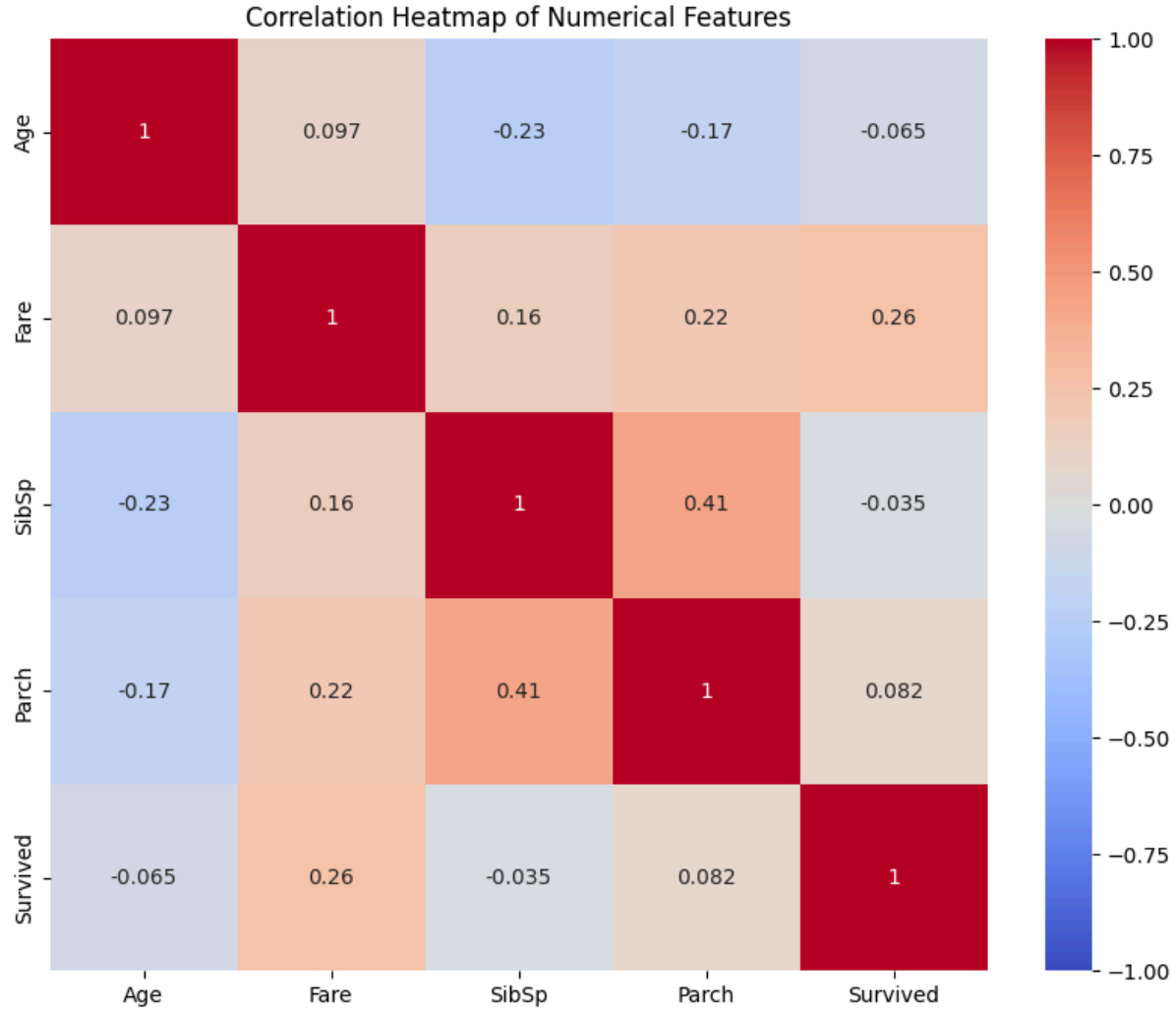
Double-click (or enter) to edit

```
# Survival rates by gender
survival_by_gender = train_data.groupby('Sex')['Survived'].mean()
print("\nSurvival rates by gender:")
print(survival_by_gender)

# Visualizing survival rates by gender
plt.figure(figsize=(8, 6))
survival_by_gender.plot(kind='bar')
plt.title('Survival Rates by Gender')
plt.xlabel('Gender')
plt.ylabel('Survival Rate')
plt.xticks(rotation=0)
plt.show()
```

```
Survival rates by gender:
Sex
female    0.742038
male      0.188908
Name: Survived, dtype: float64
```



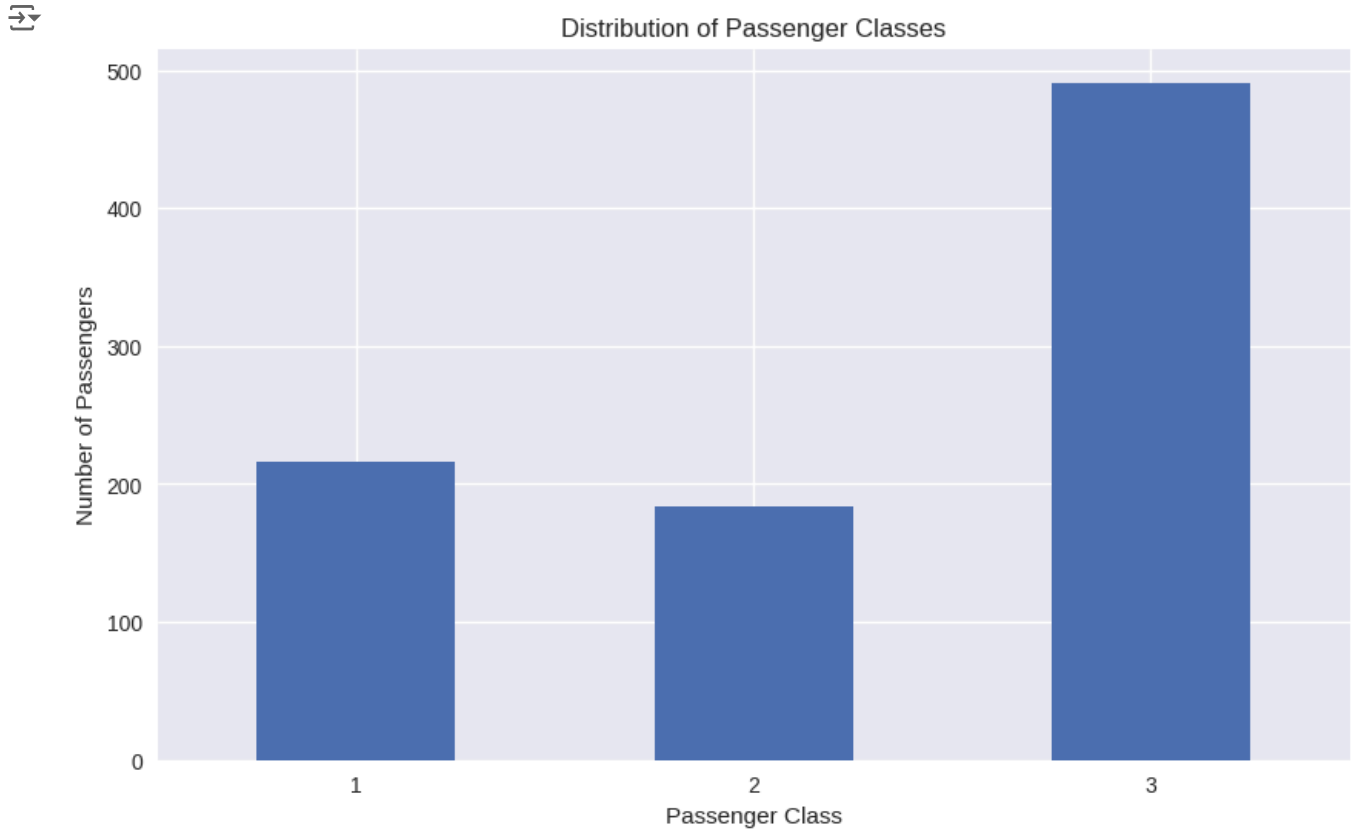

```
# Select numerical features for correlation analysis
numerical_features = ['Age', 'Fare', 'SibSp', 'Parch', 'Survived']

# Calculate the correlation matrix
correlation_matrix = train_data[numerical_features].corr()

# Create a heatmap to visualize correlations
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1, center=0)
plt.title('Correlation Heatmap of Numerical Features')
plt.show()
```



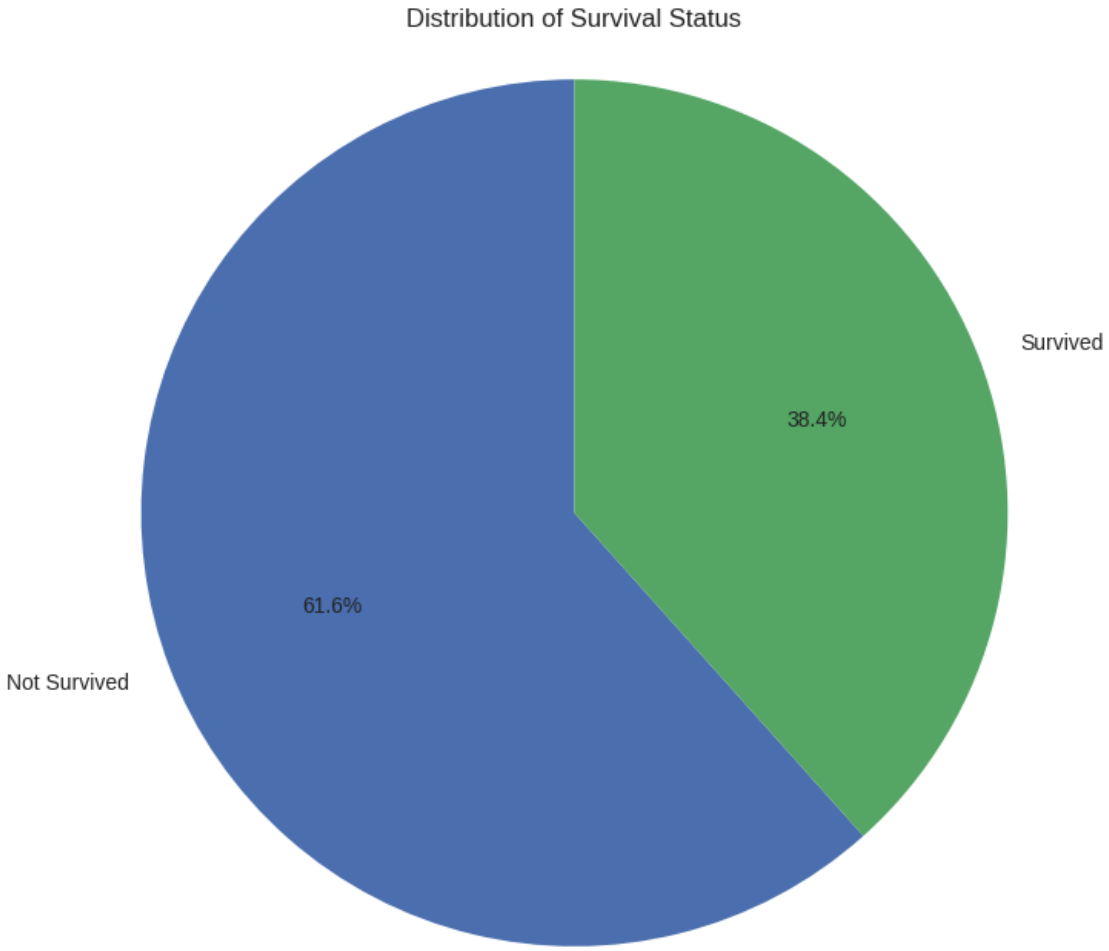

```
print("\n4. Data Visualization")
```

```
4. Data Visualization
```

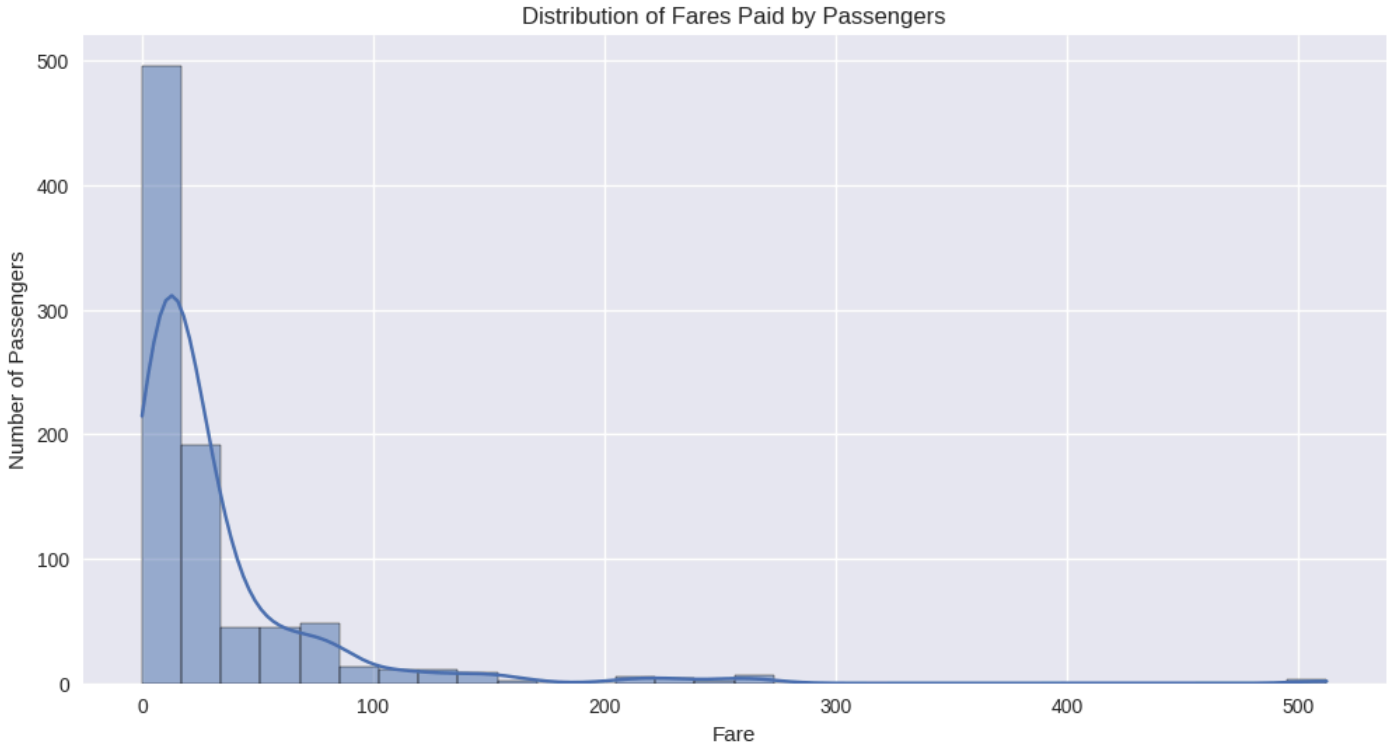
```
# Set a common style for all plots
plt.style.use('seaborn')

# 1. Distribution of passenger classes using a bar plot
plt.figure(figsize=(10, 6))
train_data['Pclass'].value_counts().sort_index().plot(kind='bar')
plt.title('Distribution of Passenger Classes')
plt.xlabel('Passenger Class')
plt.ylabel('Number of Passengers')
plt.xticks(rotation=0)
plt.show()
```




```
# 2. Distribution of survival status using a pie chart
plt.figure(figsize=(8, 8))
survival_counts = train_data['Survived'].value_counts()
plt.pie(survival_counts, labels=['Not Survived', 'Survived'], autopct='%1.1f%%', startangle=90)
plt.title('Distribution of Survival Status')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle
plt.show()
```




```
# 3. Distribution of fares paid by passengers using a histogram
plt.figure(figsize=(12, 6))
sns.histplot(train_data['Fare'], bins=30, kde=True)
plt.title('Distribution of Fares Paid by Passengers')
plt.xlabel('Fare')
plt.ylabel('Number of Passengers')
plt.show()
```




```
# 4. Box plots to compare fare distribution across different passenger classes
plt.figure(figsize=(12, 6))
sns.boxplot(x='Pclass', y='Fare', data=train_data)
plt.title('Fare Distribution Across Passenger Classes')
plt.xlabel('Passenger Class')
plt.ylabel('Fare')
plt.show()
```


```
print("\n5.Conclusion and Insights")
```

```
5.Conclusion and Insights
```

Comprehensive conclusion summarizing the key insights from your analysis.

1. Passenger Demographics:

- There were three classes of passengers, and most people were in third class.