

1. IMPORT THE LIBRARIES

```
In [5]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

2. IMPORT THE DATASET

```
In [6]: df=pd.read_csv("Titanic-Dataset.csv")
```

```
In [7]: df

Out[7]:
```

	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	Cummings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Hekkinen, Miss. Laina	female	26.0	0	0	STON/O2 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...
886	887	0	2	Moreira, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	S
890	891	0	3	Doeley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows x 12 columns

```
In [8]: df.head()
```

```
Out[8]:
```

	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	Cummings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Hekkinen, Miss. Laina	female	26.0	0	0	STON/O2 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
In [9]: df.tail()
```

```
Out[9]:
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
886	887	0	2	Moreira, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	S
890	891	0	3	Doeley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

```
In [10]: df.shape
```

```
Out[10]: (891, 12)
```

```
In [11]: 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
```

```
In [12]: df.describe()
```

```
Out[12]:
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.309642	29.699118	0.520000	0.301594	32.200000
std	261.362842	0.486992	0.839071	14.526467	1.107143	0.896067	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	664.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

```
In [13]: corr=df.corr()
```

```
corr
```

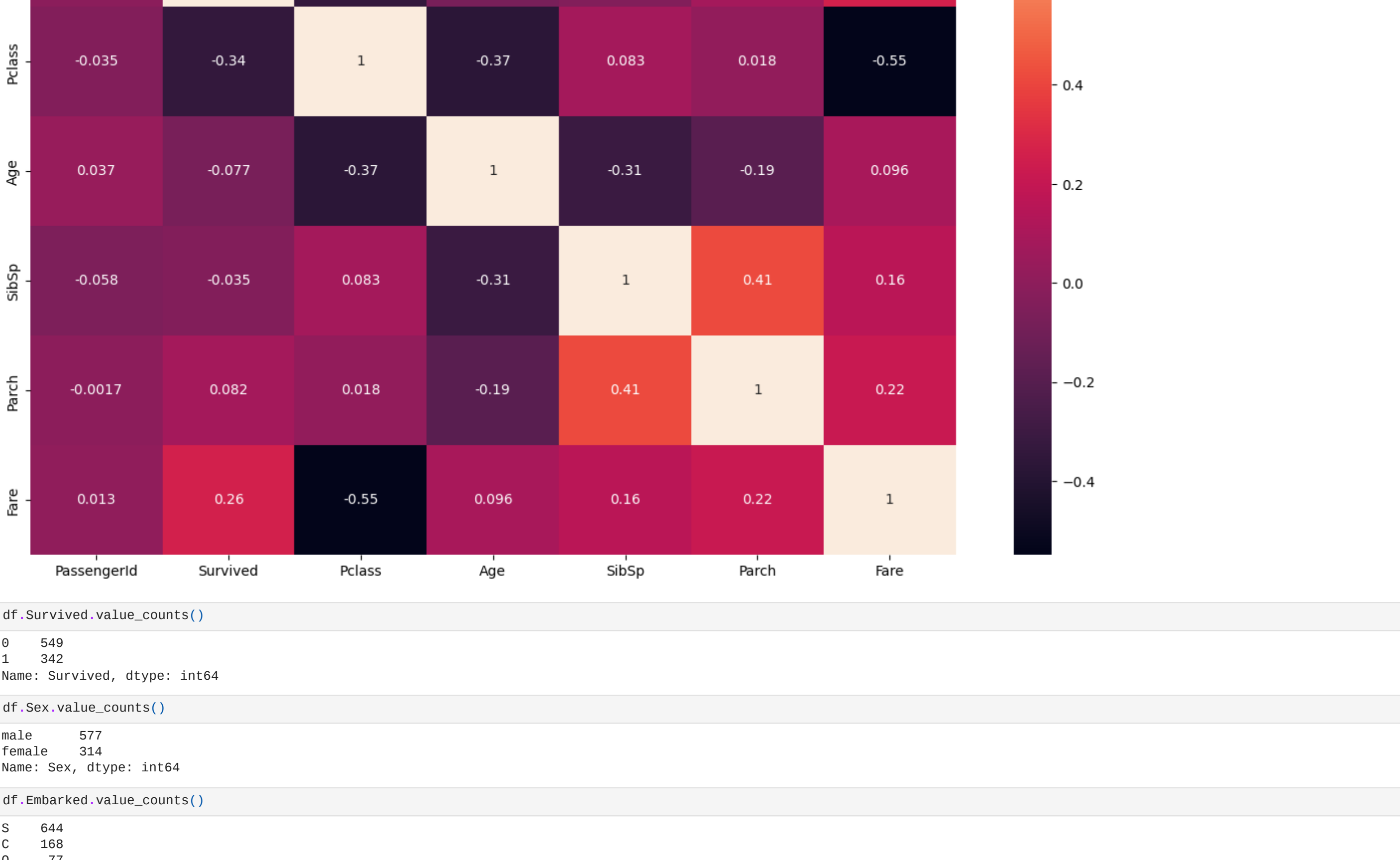
<ipython-input-13-7d5195e2bfad>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.

```
corr=df.corr()
```

```
Out[13]:
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.095007	-0.035144	0.038847	-0.057527	-0.001652	0.012658
Survived	-0.095007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083061	0.018443	-0.549500
Age	0.038847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083061	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

```
In [14]: plt.subplots(figsize=(15,10))
sns.heatmap(corr,annot=True)
```



```
In [15]: df.Survived.value_counts()
```

```
Out[15]:
```

Survived	count
0	549
1	342

Name: Survived, dtype: int64

```
In [16]: df.Sex.value_counts()
```

```
Out[16]:
```

Sex	count
male	577
female	314

Name: Sex, dtype: int64

```
In [17]: df.Embarked.value_counts()
```

```
Out[17]:
```

Embarked	count
S	644
C	168
Q	77

Name: Embarked, dtype: int64

3. CHECK FOR NULL VALUES

```
In [18]: df.isnull().any()
```

```
Out[18]:
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
PassengerId	False	False	False	False	False	False	False	False	False	False	False	False
Survived	False	False	False	False	False	False	False	False	False	False	False	False
Pclass	False	False	False	False	False	False	False	False	False	False	False	False
Name	False	False	False	False	False	False	False	False	False	False	False	False
Sex	False	False	False	False	False	False	False	False	False	False	False	False
Age	True	True	True	True	True	True	True	True	True	True	True	True
SibSp	False	False	False	False	False	False	False	False	False	False	False	False
Parch	False	False	False	False	False	False	False	False	False	False	False	False
Ticket	False	False	False	False	False	False	False	False	False	False	False	False
Fare	False	False	False	False	False	False	False	False	False	False	False	False
Cabin	True	True	True	True	True	True	True	True	True	True	True	True
Embarked	True	True	True	True	True	True	True	True	True	True	True	True
dtype: bool												

```
In [19]: df.isnull().sum()
```

```
Out[19]:
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
PassengerId	0	0	0	0	0	0	0	0	0	0	0	0
Survived	0	0	0	0	0	0	0	0	0	0	0	0
Pclass	0	0	0	0	0	0	0	0	0	0	0	0
Name	0	0	0	0	0	0	0	0	0	0	0	0
Sex	0	0	0	0	0	0	0	0	0	0	0	0
Age	177	0	0	0	0	0	0	0	0	0	0	0
SibSp	0	0	0	0	0	0	0	0	0	0	0	0
Parch	0	0	0	0	0	0	0	0	0	0	0	0
Ticket	0	0	0	0	0	0	0	0	0	0	0	0
Fare	0	0	0	0	0	0	0	0	0	0	0	0
Cabin	687	0	0	0	0	0	0	0	0	0	0	0
Embarked	2	0	0	0	0	0	0	0	0	0	0	0
dtype: int64												

```
Fill null values in the 'Age' column with the mean age
```

```
In [20]: mean_age = df['Age'].mean()
df['Age'].fillna(mean_age, inplace=True)
```

```
Fill null values in the 'Embarked' column with the most common value
```

```
In [21]: most_common_embarked = df['Embarked'].mode()[0]
df['Embarked'].fillna(most_common_embarked, inplace=True)
```

```
In [22]: df.drop(['Cabin'],axis=1, inplace=True)
```

```
In [23]: df.drop(['Ticket'],axis=1, inplace=True)
```

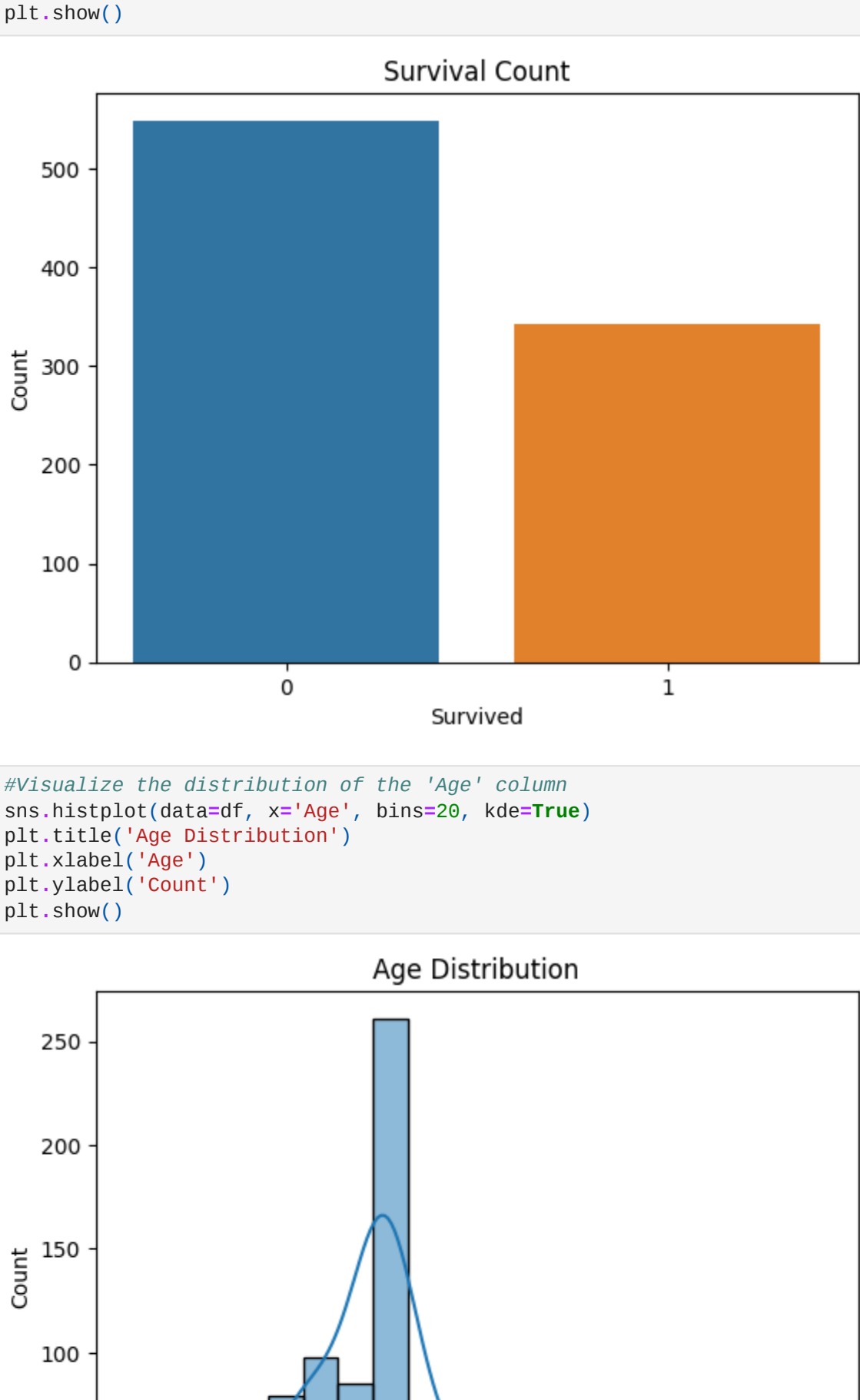
```
In [24]: df.drop(['Name'],axis=1, inplace=True)
```

```
In [25]: print(df.isnull().sum())
```

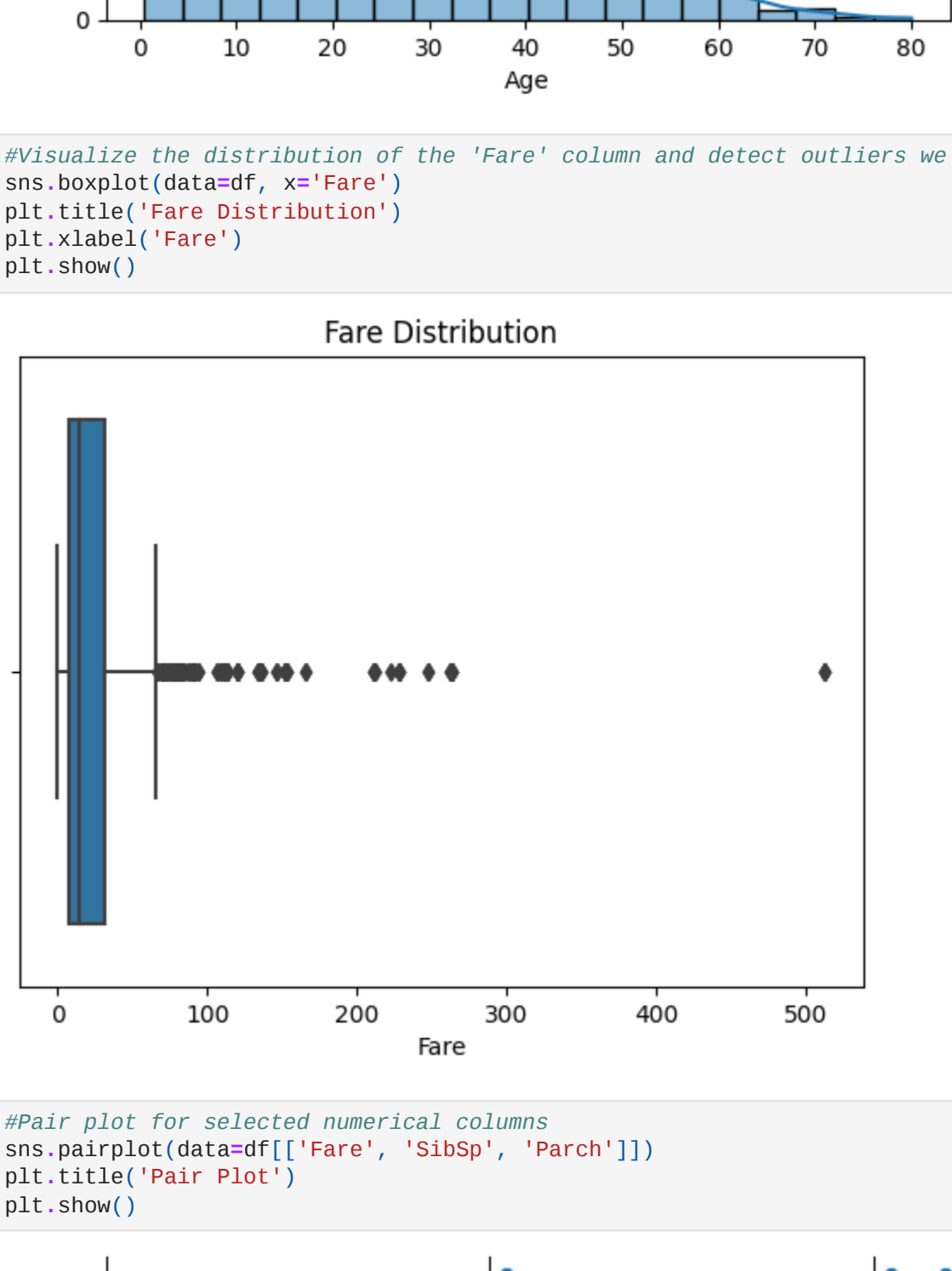
```
PassengerId    0
Survived        0
Pclass          0
Sex             0
Age            177
SibSp           0
Parch           0
Ticket          0
Fare            0
Cabin          687
Embarked        2
dtype: int64
```

4. Data Visualization

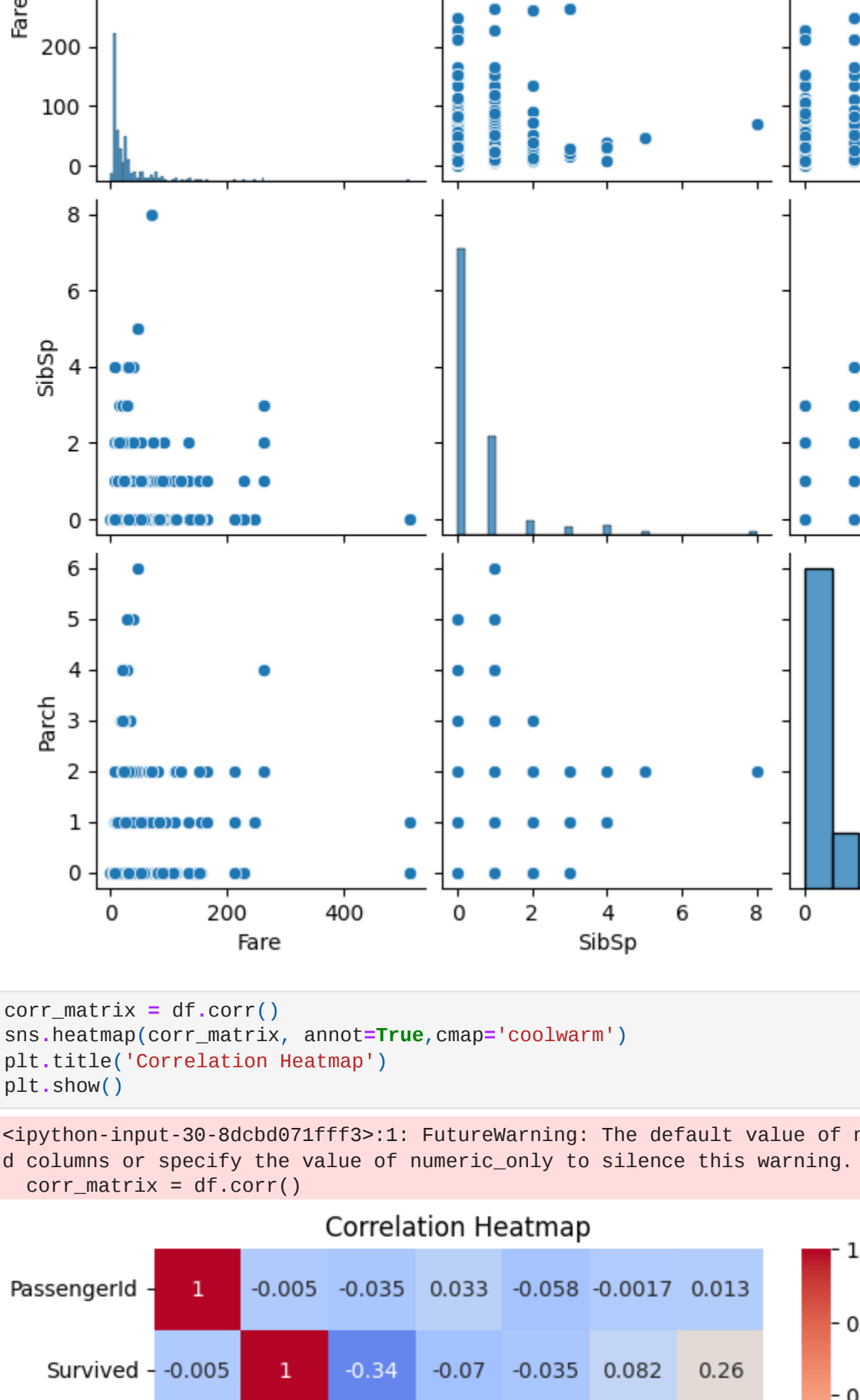
```
In [26]: #Visualize the distribution of the 'Survived' column (0 = Not Survived, 1 = Survived)
sns.countplot(data=df, x='Survived')
plt.title('Survival Count')
plt.xlabel('Survived')
plt.ylabel('Count')
plt.show()
```



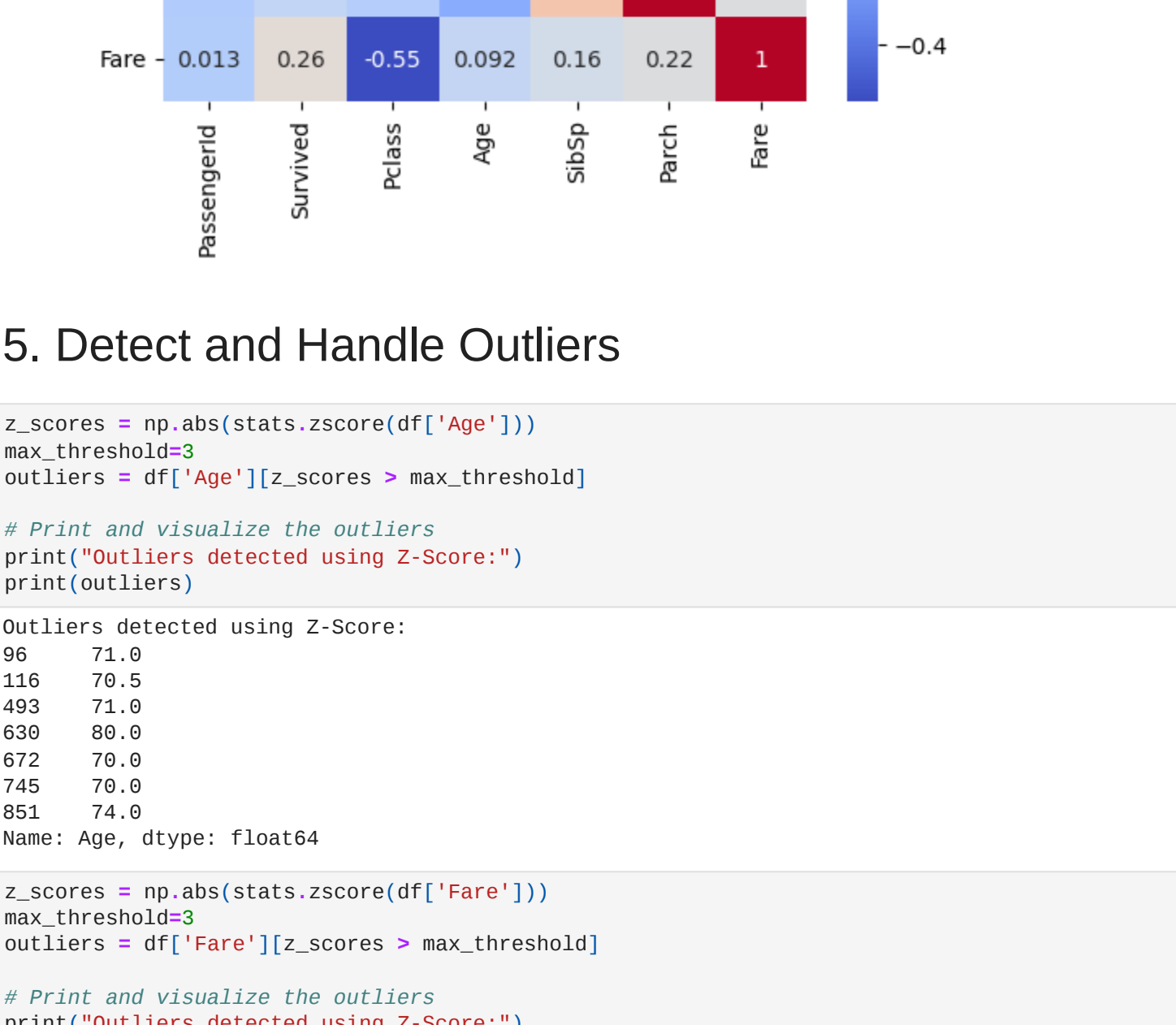
```
In [27]: #Visualize the distribution of the 'Age' column
sns.histplot(data=df, x='Age', bins=20, kde=True)
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```



```
In [28]: #Visualize the distribution of the 'Fare' column and detect outliers we will handle outliers in the next step
sns.boxplot(data=df, x='Fare')
plt.title('Fare Distribution')
plt.xlabel('Fare')
plt.show()
```



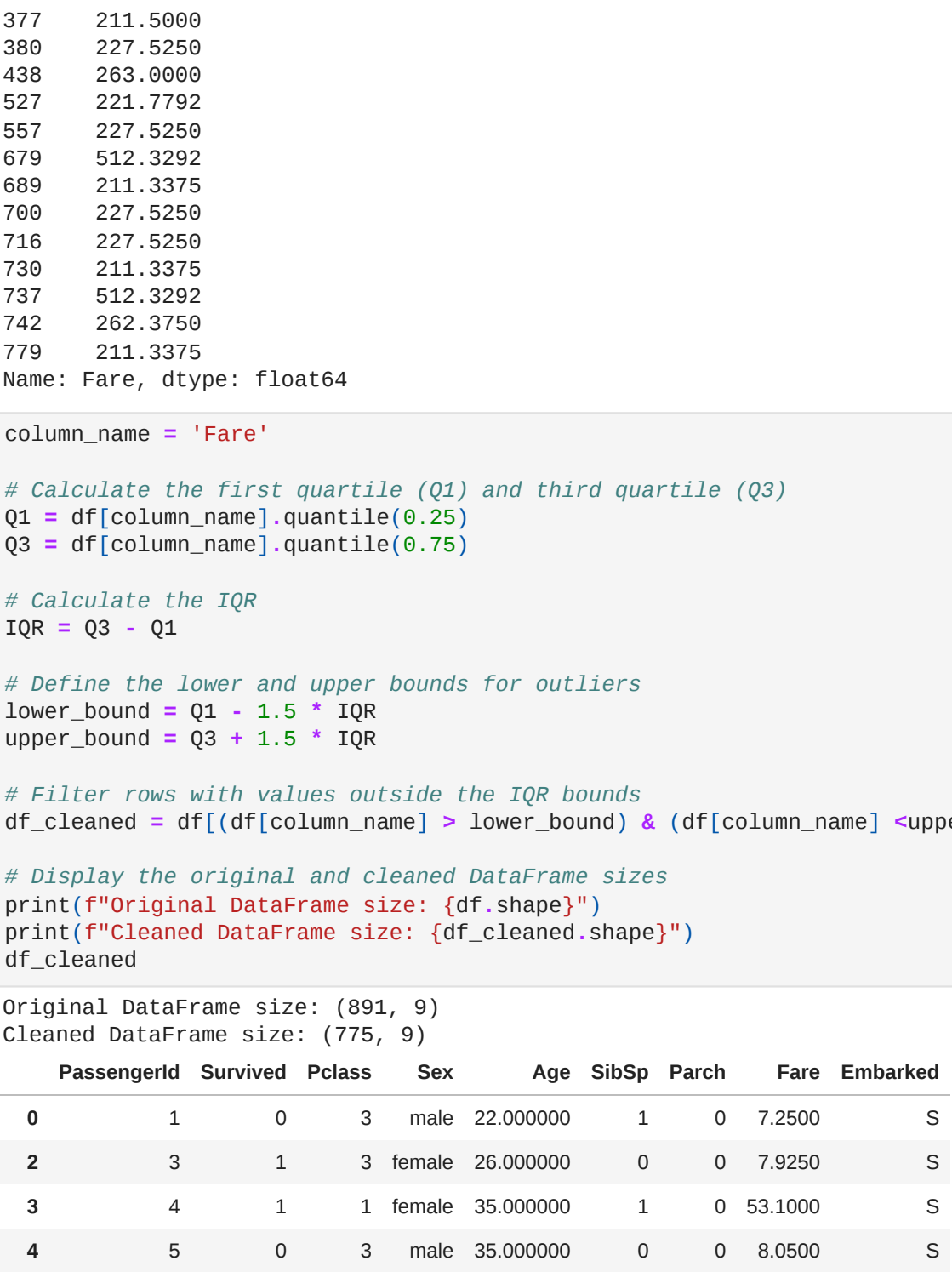
```
In [29]: #Pair plot for selected numerical columns
sns.pairplot(data=df[['Fare', 'SibSp', 'Parch']])
plt.title('Pair Plot')
plt.show()
```



```
In [30]: corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

<ipython-input-30-8dc0d71ff3>:1: 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.

```
corr_matrix = df.corr()
```



5. Detect and Handle Outliers

```
In [31]: z_scores = np.abs(stats.zscore(df['Age']))
max_threshold = df['Age'][z_scores > max_threshold]

# Print and visualize the outliers
print("Outliers detected using Z-Score:")
print(outliers)
```

```
Outliers detected using Z-Score:
96    71.8
116    79.5
483    71.8
638    86.8
672    70.2
745    70.8
851    74.8
Name: Age, dtype: float64
```

```
In [32]: z_scores = np.abs(stats.zscore(df['Fare']))
max_threshold = df['Fare'][z_scores > max_threshold]

# Print and visualize the outliers
print("Outliers detected using Z-Score:")
print(outliers)
```

```
Outliers detected using Z-Score:
27    263.0600
88    263.0600
138    247.5208
248    512.3292
299    247.5208
311    512.3292
344    263.0600
377    211.5809
389    227.5259
438    263.0600
527    221.7792
537    227.5259
679    512.3292
689    211.3375
780    227.5259
716    227.5259
730    211.3375
737    512.3292
742    262.3758
779    211.3375
Name: Fare, dtype: float64
```

```
In [33]: column_name = 'Fare'

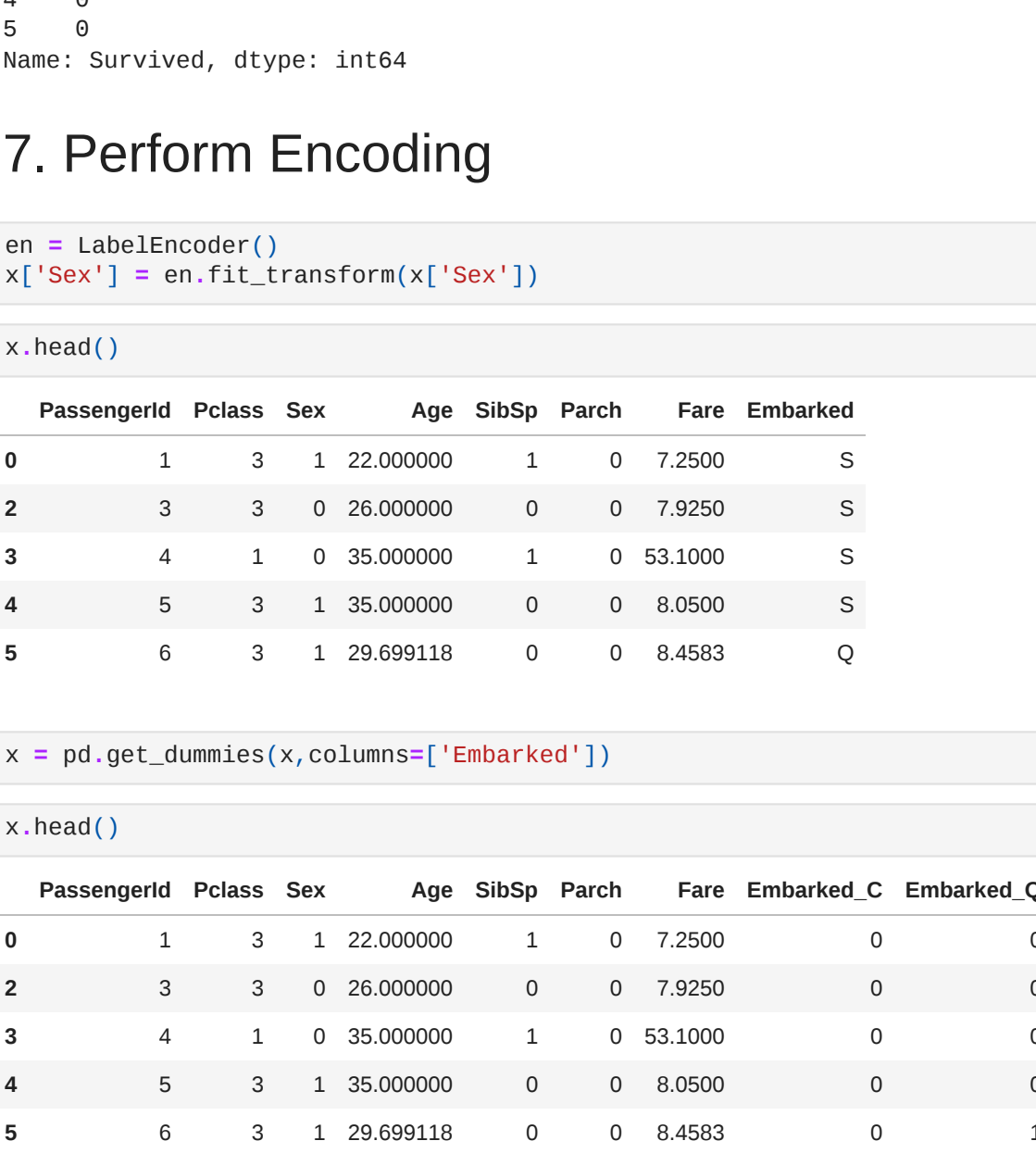
# Calculate the first quartile (Q1) and third quartile (Q3)
Q1 = df[column_name].quantile(0.25)
Q3 = df[column_name].quantile(0.75)

# Calculate the IQR
IQR = Q3 - Q1

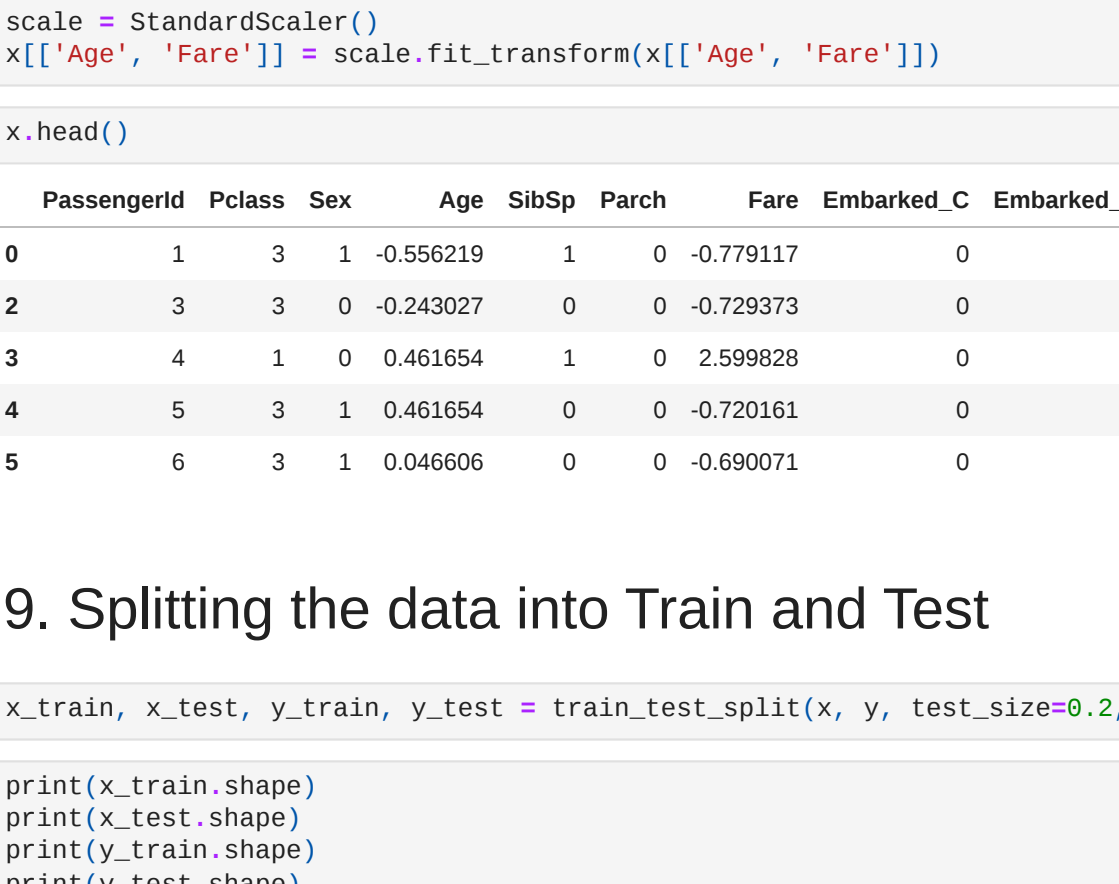
# Define the lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter rows with values outside the IQR bounds
df_cleaned = df[(df[column_name] > lower_bound) & (df[column_name] < upper_bound)]

# Display the original and cleaned DataFrame sizes
print(f"Original DataFrame size: {df.shape}")
print(f"Cleaned DataFrame size: {df_cleaned.shape}")
df_cleaned
```



```
In [34]: sns.boxplot(df_cleaned)
```



```
In [35]: df=df_cleaned
```

```
In [36]: y=df['Survived']
```

```
In [37]: x.head()
```

```
Out[37]:
```

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	3	1	22.000000	1	0	7.2500	S
2	3	3	0	26.000000	0	0	7.9250	S
3	4	1	0	35.000000	1	0	53.1000	S
4	5	3	1	35.000000	0	0	8.0500	S
5	6	3	1	29.699118	0	0	8.4583	Q

```
In [41]: X = pd.get_dummies(x.columns=[ 'Embarked' ])
```

```
Out[41]:
```

	PassengerId
--	-------------