In [71]:
```
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Load the dataset
df_train = pd.read_csv(r"C:\Users\Prajwal\Desktop\Mtech\Second Sem\ML\train.csv")
df_test = pd.read_csv(r"C:\Users\Prajwal\Desktop\Mtech\Second Sem\ML\test.csv")
```

In [72]:
```
df_train.head()
```

Out[72]:

	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
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

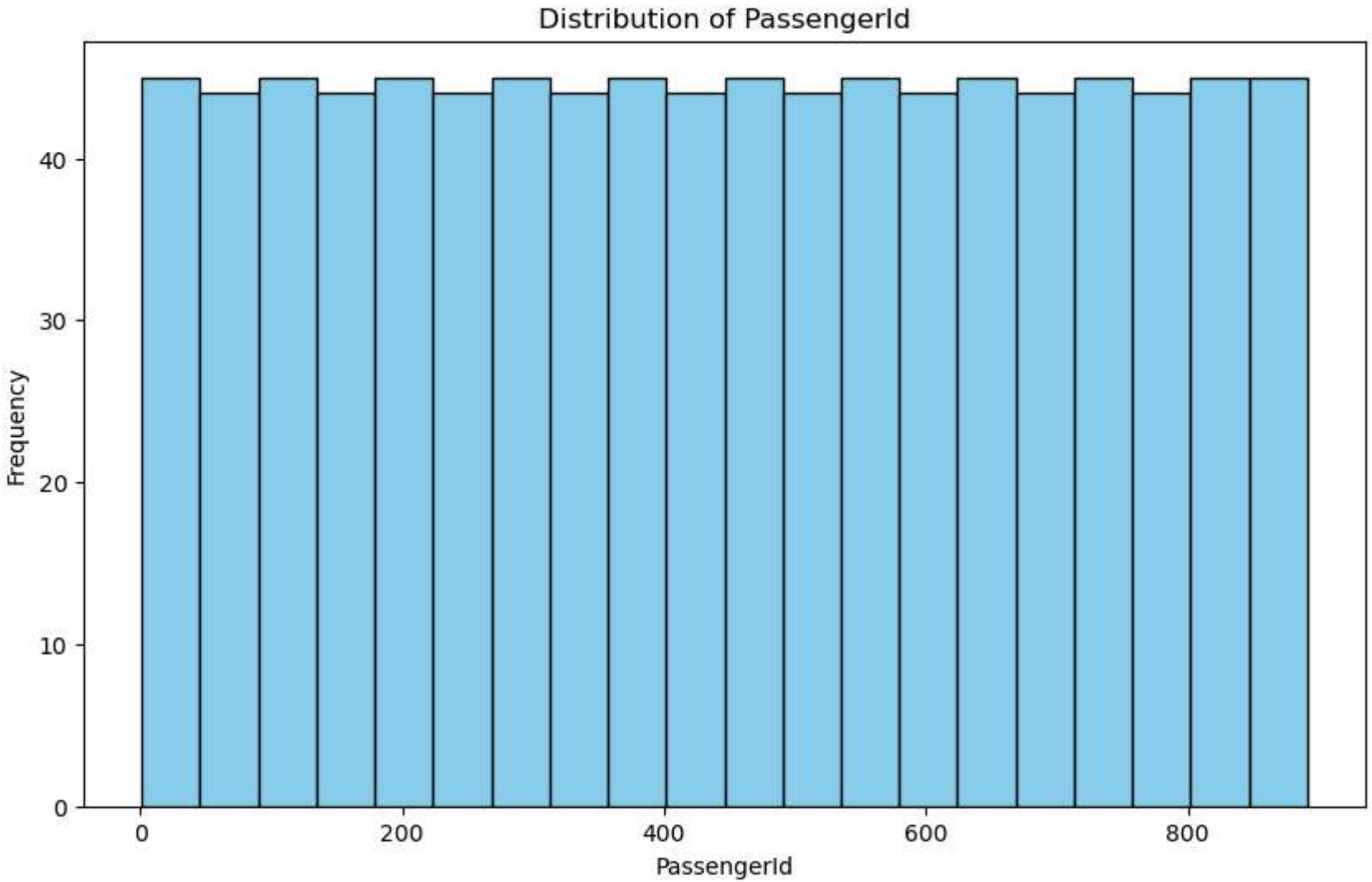
In [73]:
```
df_train.describe()
```

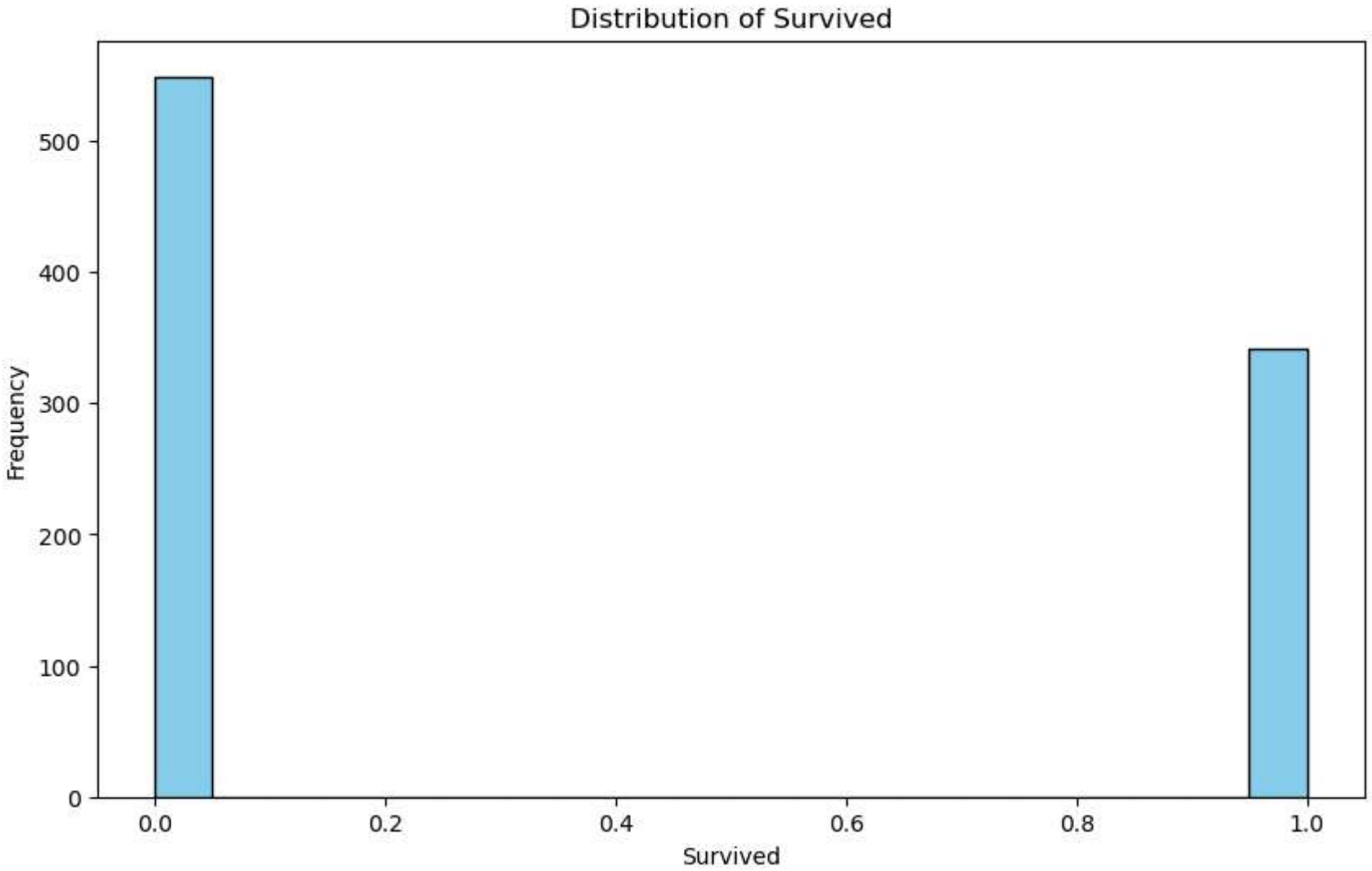
Out[73]:

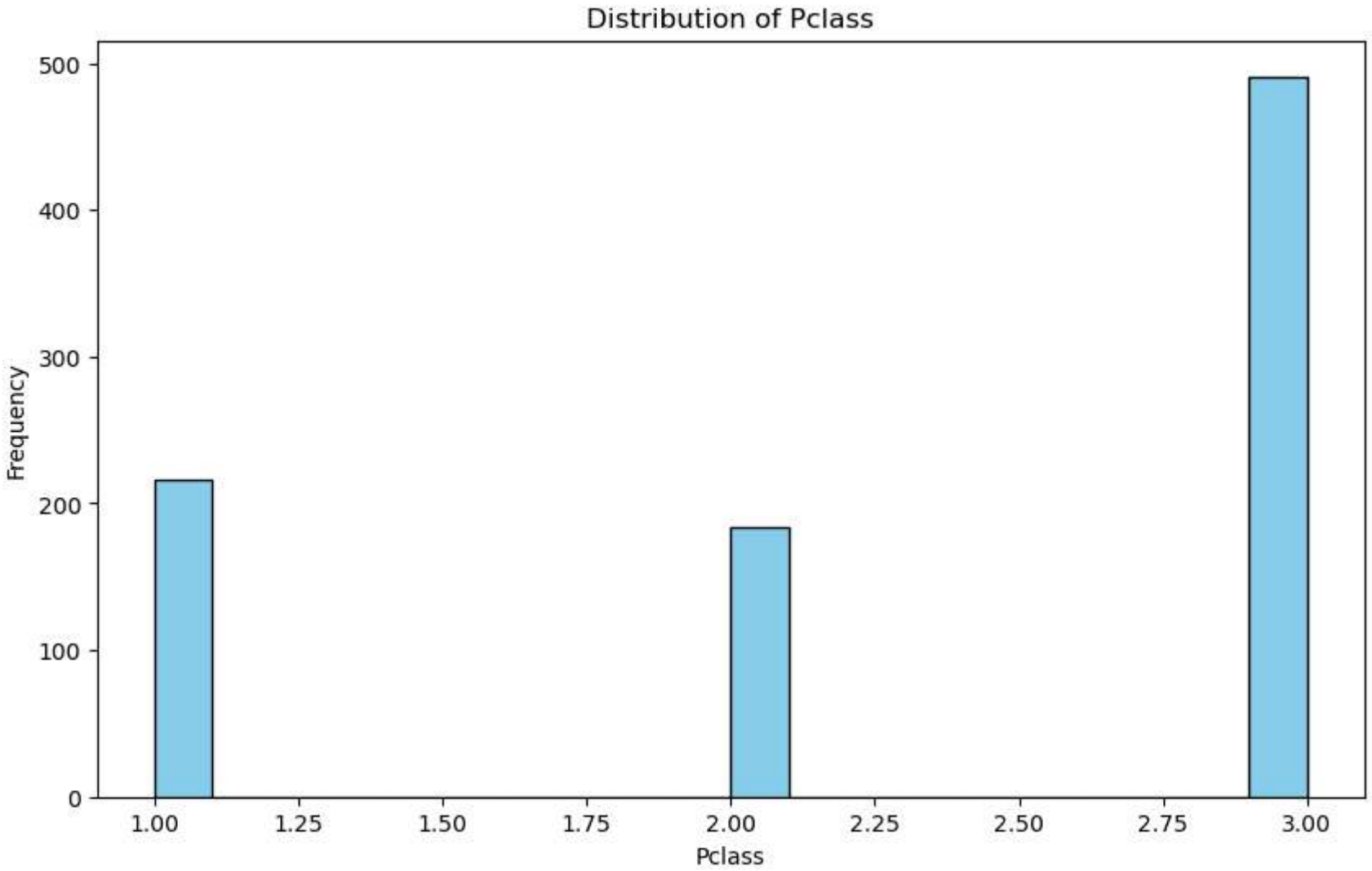
	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	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%	668.500000	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

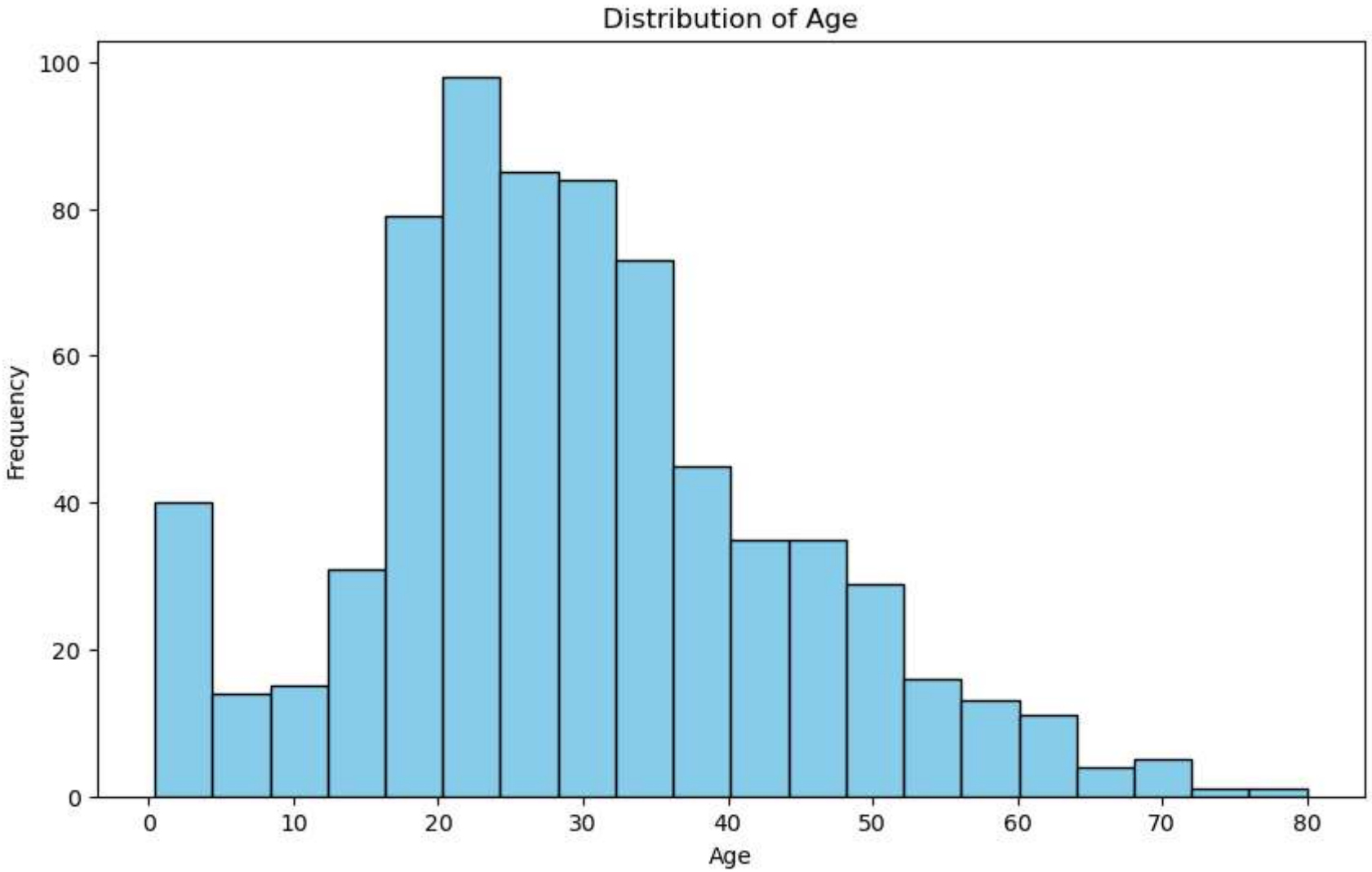
In [56]:

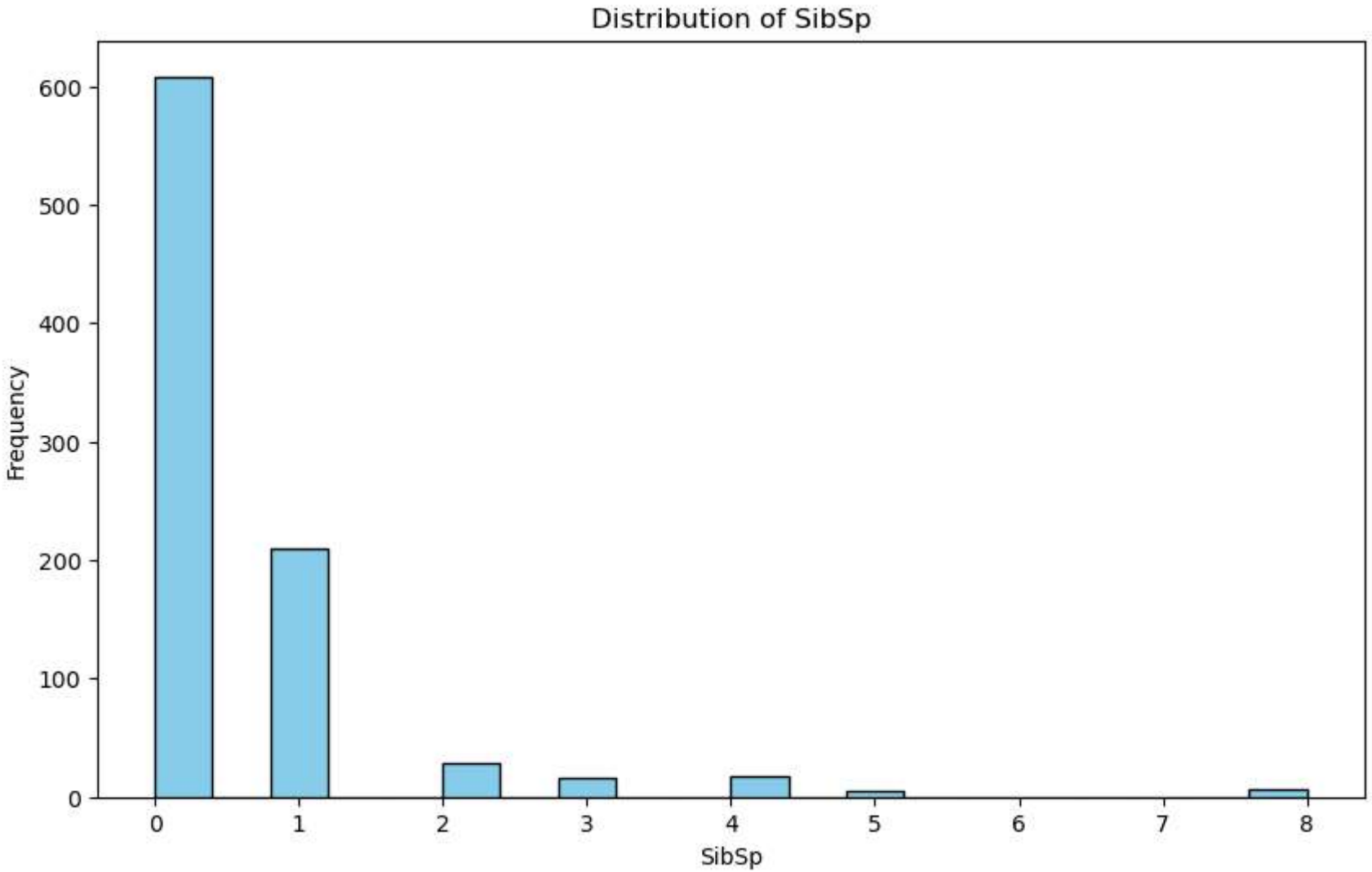
```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

# Load the Titanic dataset
train_data = pd.read_csv(r"C:\Users\Prajwal\Desktop\Mtech\Second Sem\ML\train.csv")

# Plot histograms for numerical columns
numerical_columns = train_data.select_dtypes(include=['int64', 'float64']).columns
for col in numerical_columns:
    plt.figure(figsize=(10, 6))
    plt.hist(train_data[col].dropna(), bins=20, color='skyblue', edgecolor='black')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.title(f'Distribution of {col}')
    plt.show()

# Plot bar plots for categorical columns
categorical_columns = train_data.select_dtypes(include=['object']).columns
for col in categorical_columns:
    plt.figure(figsize=(8, 5))
    sns.countplot(data=train_data, x=col, palette='Set2')
    plt.xlabel(col)
    plt.ylabel('Count')
    plt.title(f'Distribution of {col}')
    plt.xticks(rotation=45)
    plt.show()
```

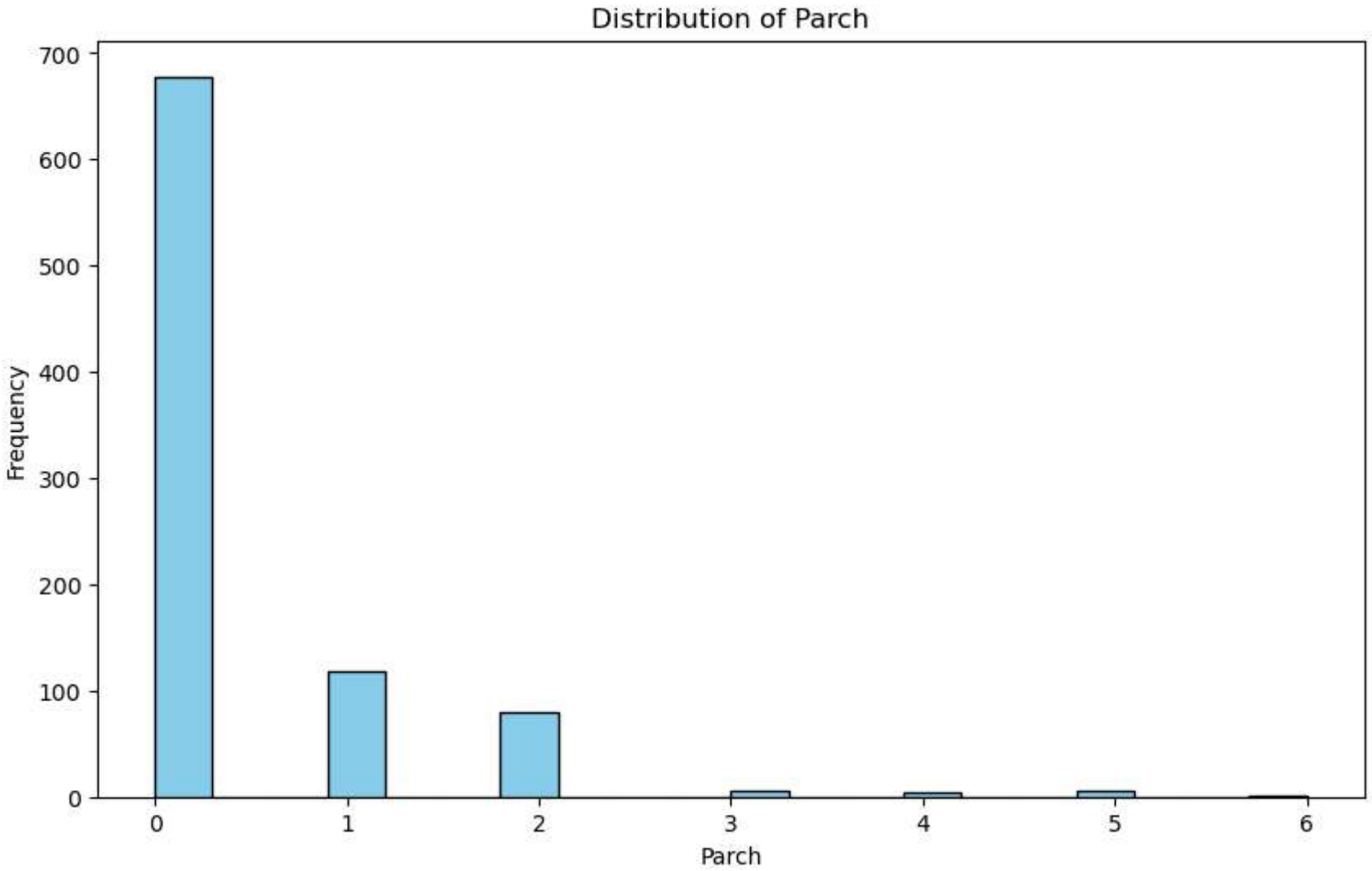

Distribution of PassengerId
Frequency
0
10
20
30
40
0
200
400
600
800
PassengerId

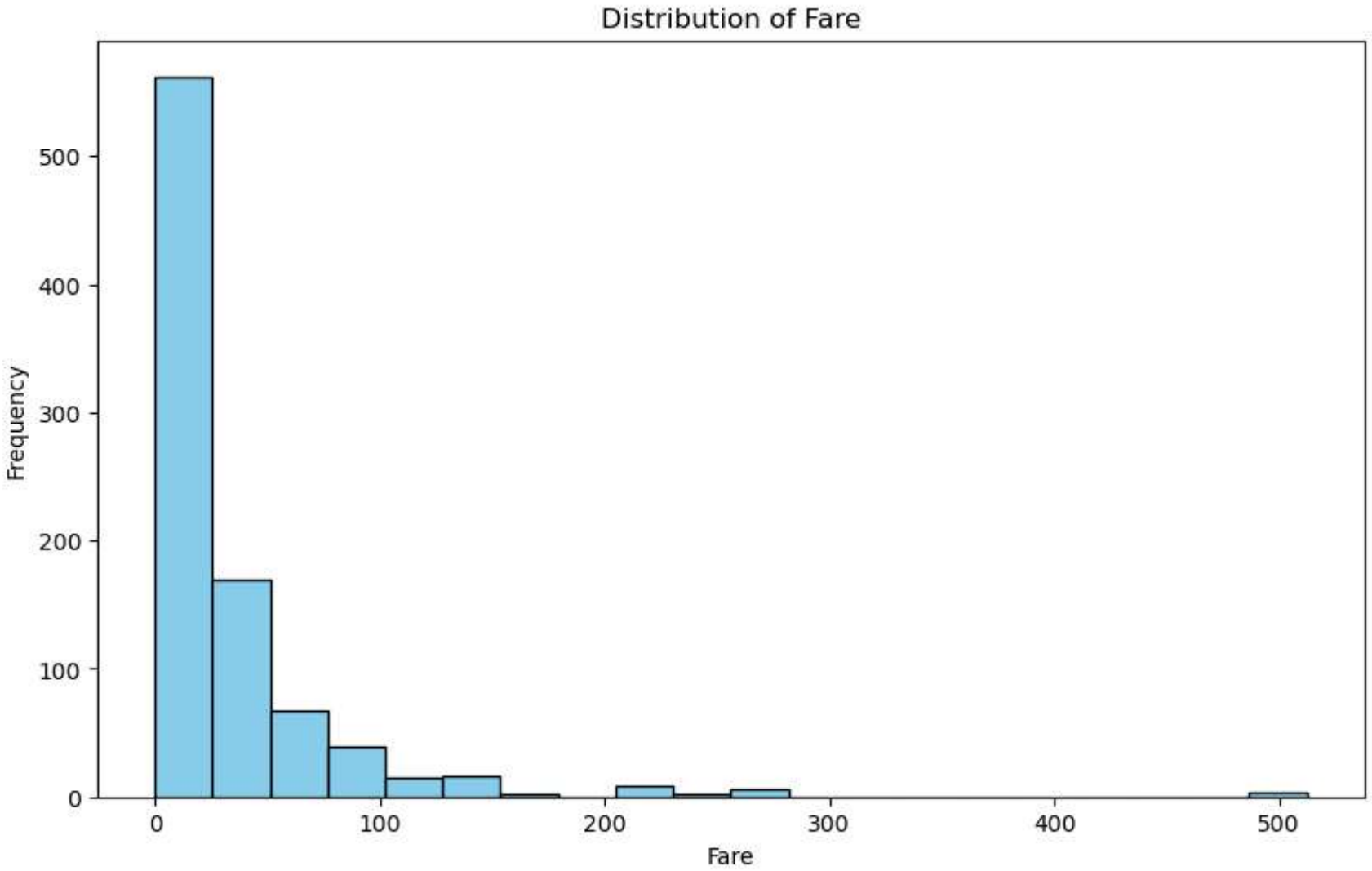

Distribution of Survived
Frequency
0
100
200
300
400
500
0.0
0.2
0.4
0.6
0.8
1.0
Survived

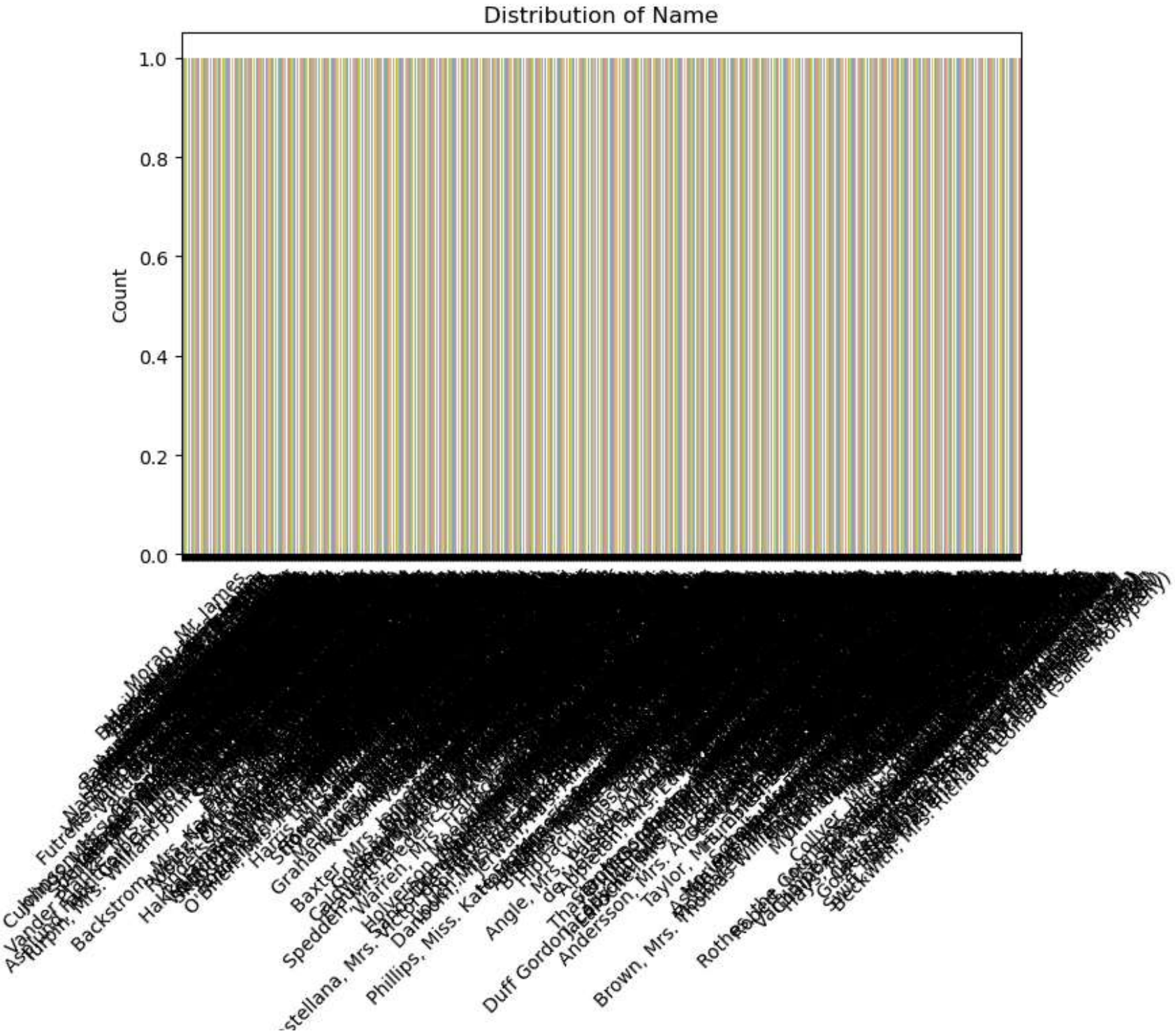

Distribution of Pclass
Frequency
Pclass
0
100
200
300
400
500
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00

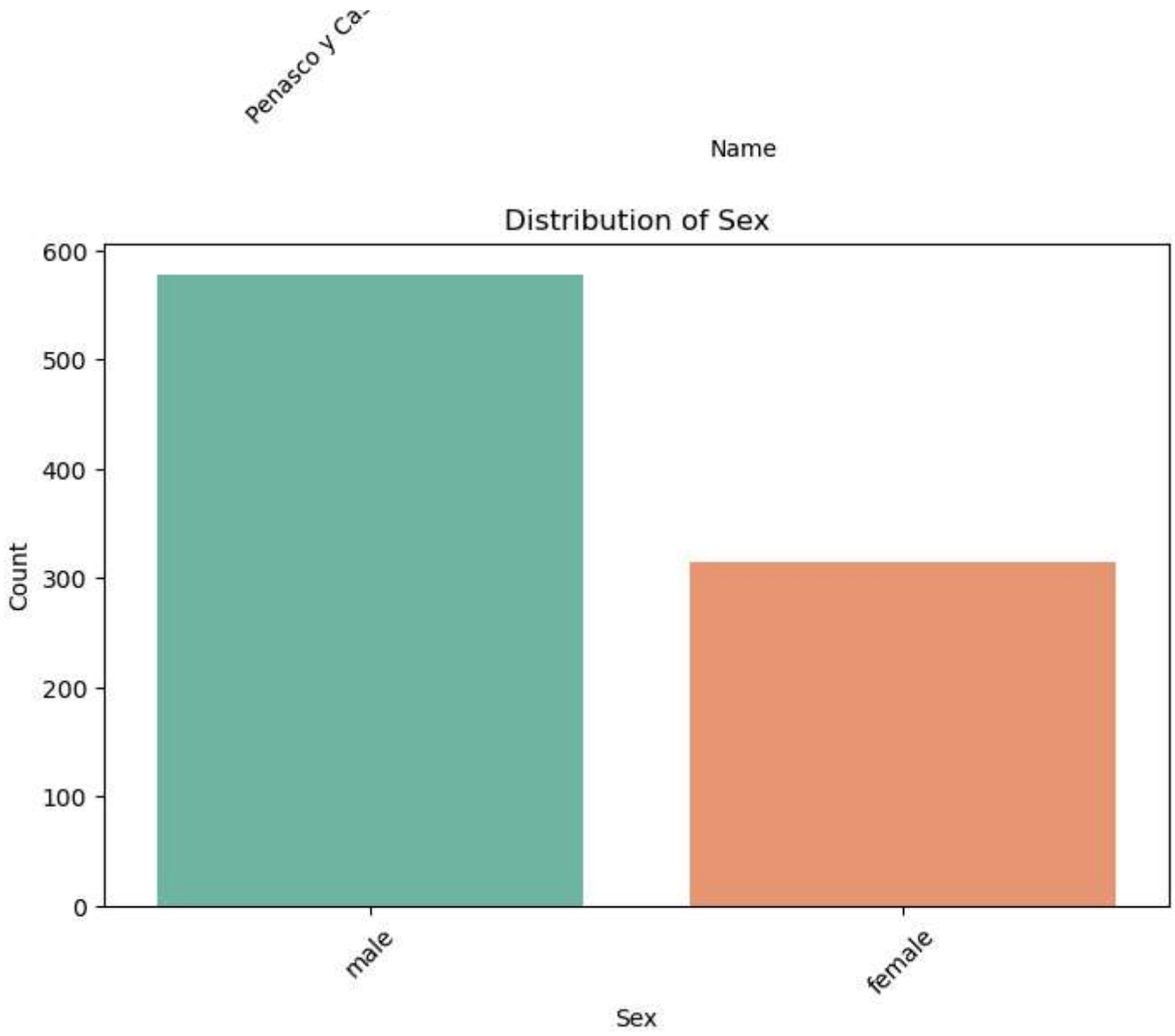

Distribution of Age
Frequency
Age
0
20
40
60
80
100
0
10
20
30
40
50
60
70
80

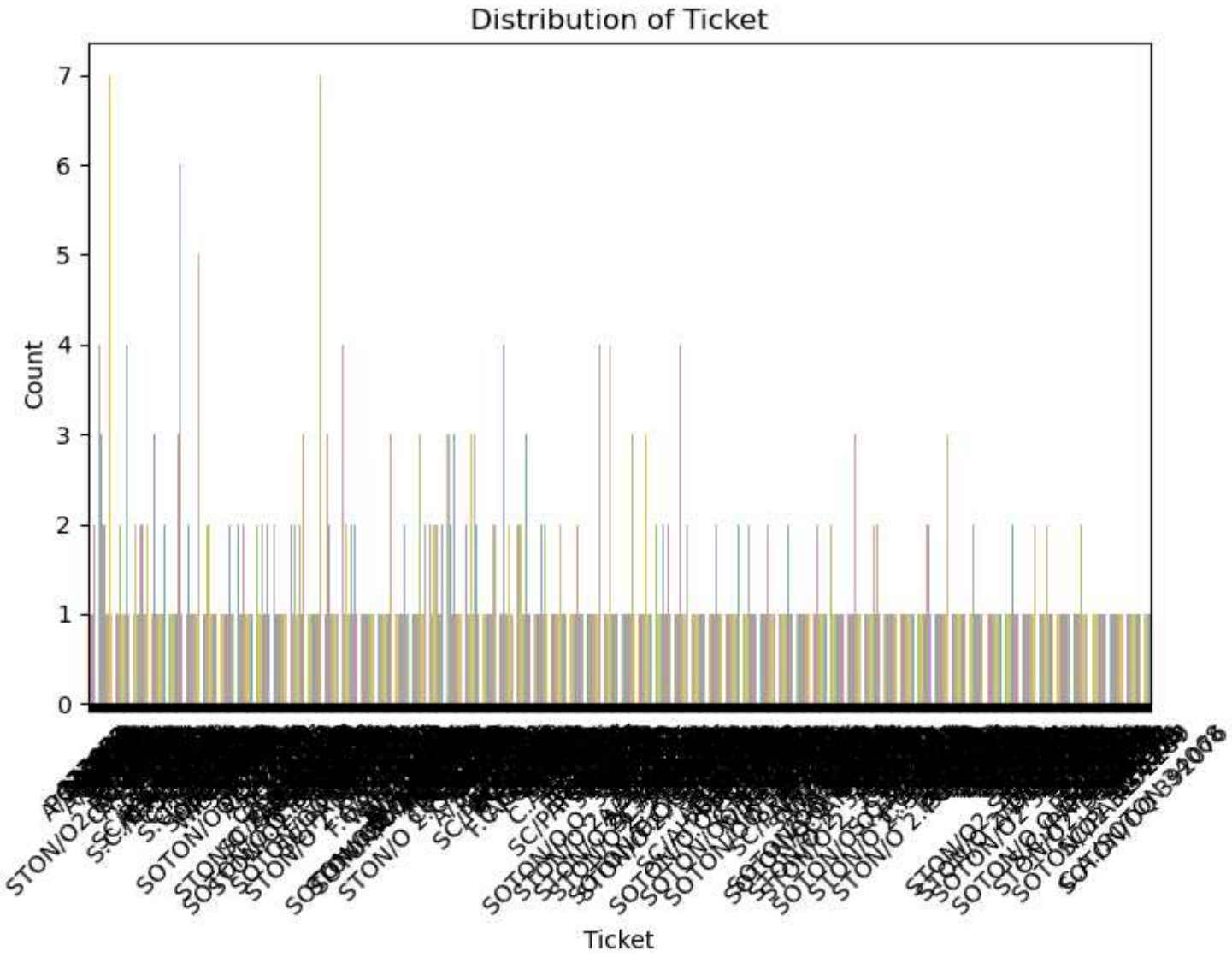

Distribution of SibSp
Frequency
SibSp
0
100
200
300
400
500
600
0
1
2
3
4
5
6
7
8

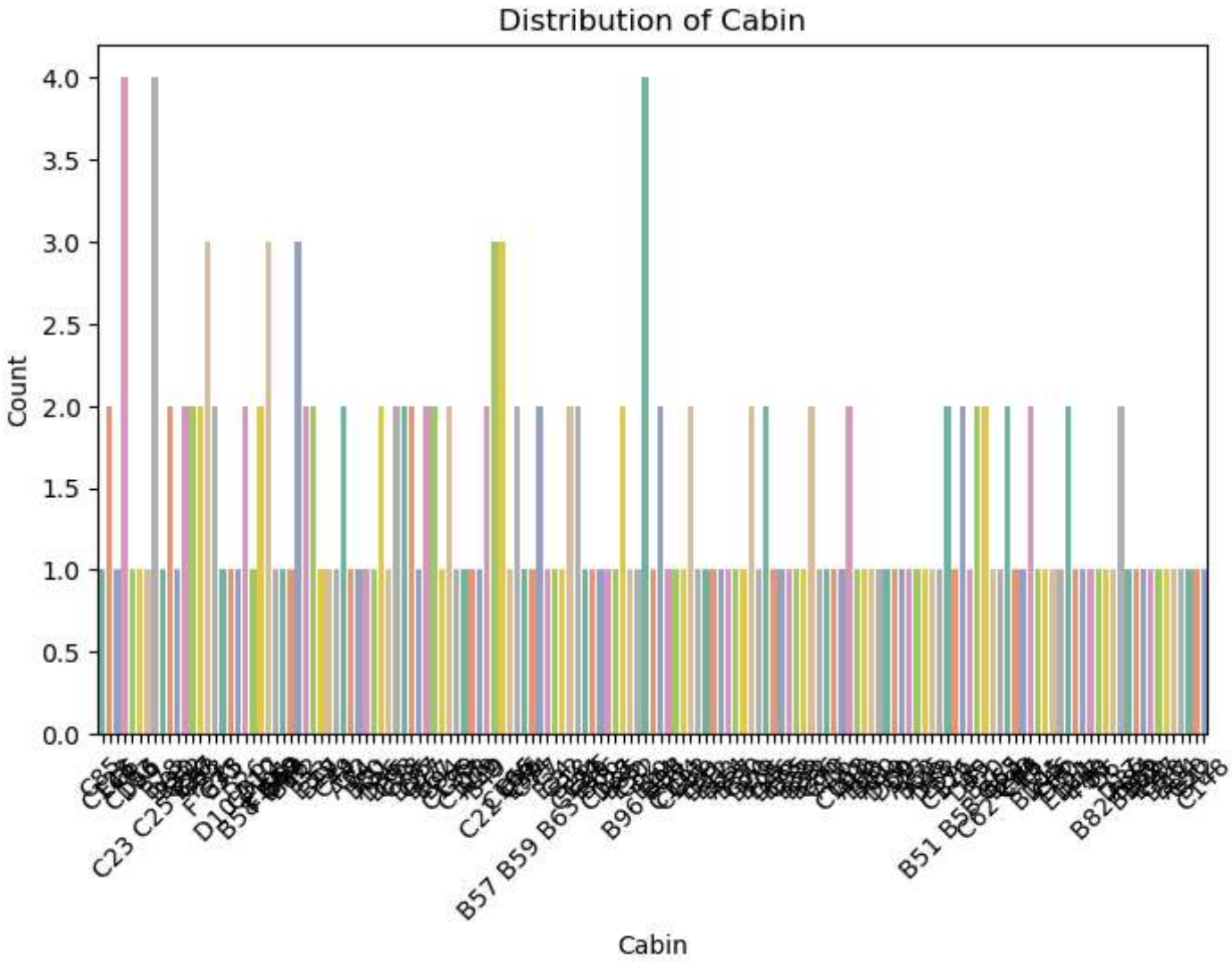

Distribution of Parch
Frequency
700
600
500
400
300
200
100
0
0
1
2
3
4
5
6
Parch

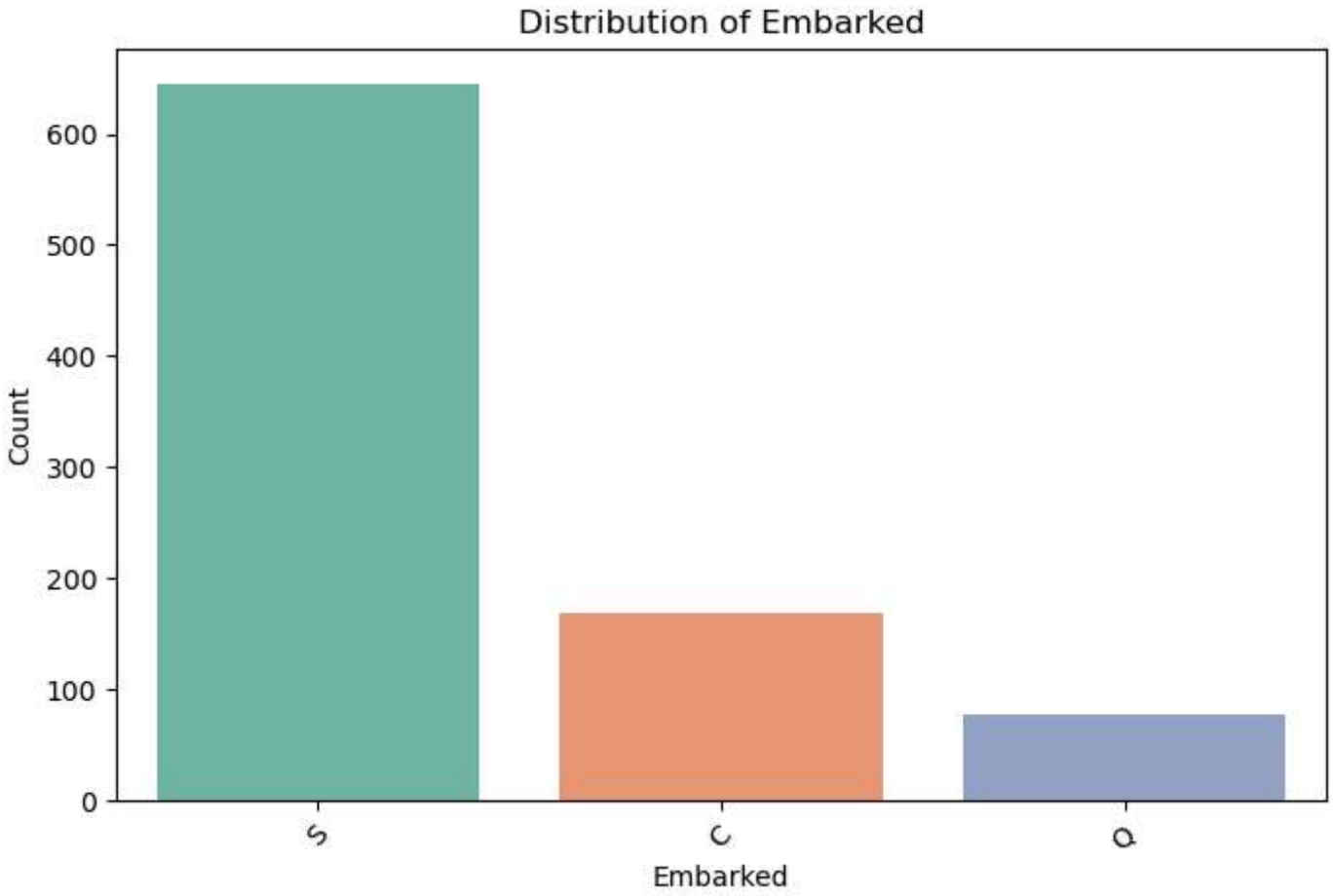

Distribution of Fare
Frequency
0
100
200
300
400
500
Fare
0
100
200
300
400
500


Distribution of Name
Count
1.0
0.8
0.6
0.4
0.2
0.0


Penasco y Ca
Name
Distribution of Sex
600
500
400
300
200
100
0
Count
male
female
Sex


Distribution of Ticket
Count
0
1
2
3
4
5
6
7
Ticket


Distribution of Cabin
Count
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
C23 C25
B57 B59 B63
B51 B5
B82
C148
Cabin



In [60]:
```
# Preprocessing
# Fill missing values
df_train['Age'].fillna(df_train['Age'].median(), inplace=True)
df_test['Age'].fillna(df_test['Age'].median(), inplace=True)
df_train['Embarked'].fillna(df_train['Embarked'].mode()[0], inplace=True)
df_test['Embarked'].fillna(df_test['Embarked'].mode()[0], inplace=True)
```

In [61]:
```
# Feature selection
features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']
X = pd.get_dummies(df_train[features])
print("Input matrix for the training data, X:\n", X)
```

```
Input matrix for the training data, X:
      Pclass   Age  SibSp  Parch     Fare  Sex_female  Sex_male  Embarked_C  \
0          3  22.0      1      0   7.2500       False      True       False
1          1  38.0      1      0  71.2833        True     False        True
2          3  26.0      0      0   7.9250        True     False       False
3          1  35.0      1      0  53.1000        True     False       False
4          3  35.0      0      0   8.0500       False      True       False
..       ...   ...    ...    ...      ...         ...       ...         ...
886        2  27.0      0      0  13.0000       False      True       False
887        1  19.0      0      0  30.0000        True     False       False
888        3  28.0      1      2  23.4500        True     False       False
889        1  26.0      0      0  30.0000       False      True        True
890        3  32.0      0      0   7.7500       False      True       False

     Embarked_Q  Embarked_S
0         False        True
1         False       False
2         False        True
3         False        True
4         False        True
..          ...         ...
886       False        True
887       False        True
888       False        True
889       False       False
890        True       False

[891 rows x 10 columns]
```

In [62]:
```
y = df_train['Survived']
print("Target Variable, y:\n", y)
```

```
Target Variable, y:
 0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

In [63]:
```
X_test = pd.get_dummies(df_test[features])
print("Input matrix for the test data, X_test:\n", X_test)
```

```
Input matrix for the test data, X_test:
      Pclass   Age  SibSp  Parch      Fare  Sex_female  Sex_male  Embarked_C  \
0          3  34.5      0      0    7.8292       False      True       False
1          3  47.0      1      0    7.0000        True     False       False
2          2  62.0      0      0    9.6875       False      True       False
3          3  27.0      0      0    8.6625       False      True       False
4          3  22.0      1      1   12.2875        True     False       False
..       ...   ...    ...    ...       ...         ...       ...         ...
413        3  27.0      0      0    8.0500       False      True       False
414        1  39.0      0      0  108.9000        True     False        True
415        3  38.5      0      0    7.2500       False      True       False
416        3  27.0      0      0    8.0500       False      True       False
417        3  27.0      1      1   22.3583       False      True        True

     Embarked_Q  Embarked_S
0          True       False
1         False        True
2          True       False
3         False        True
4         False        True
..          ...         ...
413       False        True
414       False       False
415       False        True
416       False        True
417       False       False

[418 rows x 10 columns]
```

In [64]:
```
# Split the data into training and testing sets
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
print("X_train:\n", X_train)
print("X_val:\n", X_val)
print("y_train:\n", y_train)
print("y_val:\n", y_val)
```

```
X_train:
      Pclass   Age  SibSp  Parch      Fare  Sex_female  Sex_male  Embarked_C  \
331       1  45.5      0      0   28.5000       False      True       False
733       2  23.0      0      0   13.0000       False      True       False
382       3  32.0      0      0    7.9250       False      True       False
704       3  26.0      1      0    7.8542       False      True       False
813       3   6.0      4      2   31.2750        True     False       False
..      ...   ...    ...    ...       ...         ...       ...         ...
106       3  21.0      0      0    7.6500        True     False       False
270       1  28.0      0      0   31.0000       False      True       False
860       3  41.0      2      0   14.1083       False      True       False
435       1  14.0      1      2  120.0000        True     False       False
102       1  21.0      0      1   77.2875       False      True       False

     Embarked_Q  Embarked_S
331       False        True
733       False        True
382       False        True
704       False        True
813       False        True
..          ...         ...
106       False        True
270       False        True
860       False        True
435       False        True
102       False        True

[712 rows x 10 columns]
X_val:
      Pclass   Age  SibSp  Parch     Fare  Sex_female  Sex_male  Embarked_C  \
709       3  28.0      1      1  15.2458       False      True        True
439       2  31.0      0      0  10.5000       False      True       False
840       3  20.0      0      0   7.9250       False      True       False
720       2   6.0      0      1  33.0000        True     False       False
39        3  14.0      1      0  11.2417        True     False        True
..      ...   ...    ...    ...     ...         ...       ...         ...
433       3  17.0      0      0   7.1250       False      True       False
773       3  28.0      0      0   7.2250       False      True        True
25        3  38.0      1      5  31.3875        True     False       False
84        2  17.0      0      0  10.5000        True     False       False
10        3   4.0      1      1  16.7000        True     False       False

     Embarked_Q  Embarked_S
709       False       False
439       False        True
```

```
840       False        True
720       False        True
39        False       False
..          ...         ...
433       False        True
773       False       False
25        False        True
84        False        True
10        False        True

[179 rows x 10 columns]
y_train:
 331    0
733    0
382    0
704    0
813    0
      ..
106    1
270    0
860    0
435    1
102    0
Name: Survived, Length: 712, dtype: int64
y_val:
 709    1
439    0
840    0
720    1
39     1
      ..
433    0
773    0
25     1
84     1
10     1
Name: Survived, Length: 179, dtype: int64
```

In [65]:

```python
from sklearn.model_selection import GridSearchCV

# Define the hyperparameters to tune
param_grid = {
    'max_depth': [3, 5, 7, 10],
    'min_samples_split': [2, 5, 7, 10],
    'min_samples_leaf': [1, 2, 4, 5, 7]
```

```
}

# Initialize the GridSearchCV object
grid_search = GridSearchCV(estimator=DecisionTreeClassifier(random_state=42),
                           param_grid=param_grid,
                           n_jobs=-1,
                           verbose=2)

# Perform the grid search
grid_search.fit(X_train, y_train)

# Get the best hyperparameters
best_params = grid_search.best_params_
print("Best Hyperparameters:", best_params)

# Use the best model for prediction
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
```

```
Fitting 5 folds for each of 80 candidates, totalling 400 fits
Best Hyperparameters: {'max_depth': 3, 'min_samples_leaf': 5, 'min_samples_split': 2}
```

In [66]:

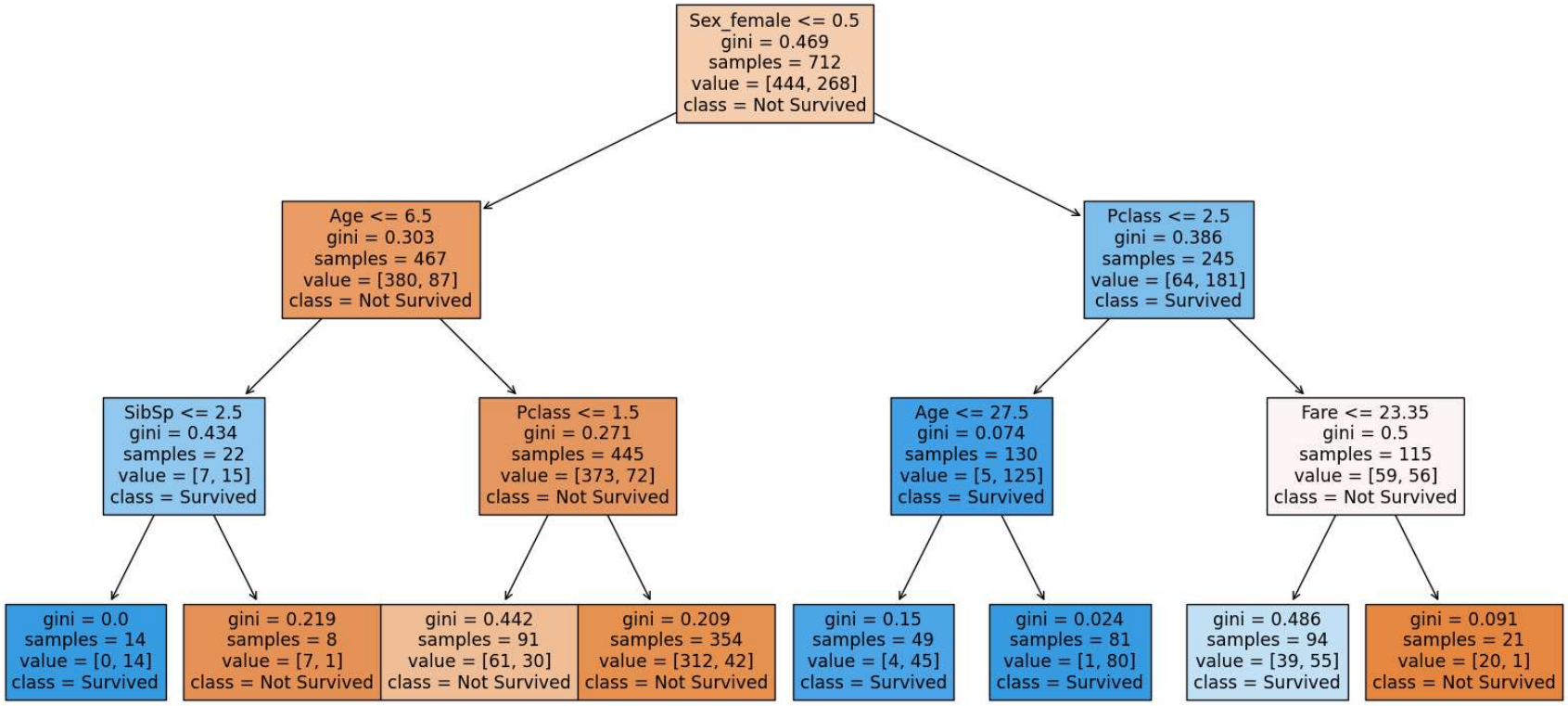
```
# Decision Tree Classifier
dt_classifier = DecisionTreeClassifier(max_depth=3, min_samples_split=2, min_samples_leaf=5, random_state=42)
dt_classifier.fit(X_train, y_train)

# Model evaluation
# Cross-validation
cv_scores = cross_val_score(dt_classifier, X, y, cv=11)
print(f"Cross-validation scores: {cv_scores}")
print(f"Mean CV accuracy: {cv_scores.mean()}")

# Predictions on the test set
y_pred = dt_classifier.predict(X_test)

# Output predictions to a file
output = pd.DataFrame({'PassengerId': df_test.PassengerId, 'Survived': y_pred})
output.to_csv(r"C:\Users\Prajwal\Desktop\Mtech\Second Sem\ML\titanic_predictions_dt_classification.csv", index=False)
```

```
Cross-validation scores: [0.82716049 0.81481481 0.82716049 0.80246914 0.86419753 0.83950617
 0.77777778 0.80246914 0.7654321  0.88888889 0.81481481]
Mean CV accuracy: 0.8204264870931538
```

In [67]:

```
from sklearn import tree
import matplotlib.pyplot as plt
```

```
# Get feature names as a list
feature_names = X.columns.tolist()

# Visualize the Decision Tree
plt.figure(figsize=(20,10))
tree.plot_tree(dt_classifier, filled=True, feature_names=feature_names, class_names=['Not Survived', 'Survived'])
plt.show()
```



In [68]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline

# Define the pipeline with an imputer and the random forest classifier
pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='mean')),  # Use mean to impute missing values
    ('classifier', RandomForestClassifier(random_state=42))
])
```

```
# Define the hyperparameters to tune
param_grid = {
    'classifier__n_estimators': [50, 100],
    'classifier__max_depth': [10, 20],
    'classifier__min_samples_split': [2, 5, 7, 10],
    'classifier__min_samples_leaf': [1, 2, 4, 5, 7]
}

# Initialize the GridSearchCV object
grid_search = GridSearchCV(estimator=pipeline,
                           param_grid=param_grid,
                           n_jobs=-1,
                           verbose=2)

# Perform the grid search
grid_search.fit(X_train, y_train)

# Get the best hyperparameters
best_params = grid_search.best_params_
print("Best Hyperparameters:", best_params)

# Use the best model for prediction
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

# Output predictions to a file
output = pd.DataFrame({'PassengerId': df_test.PassengerId, 'Survived': y_pred})
output.to_csv(r"C:\Users\Prajwal\Desktop\Mtech\Second Sem\ML\titanic_predictions_rf_classification.csv", index=False)
```

```
Fitting 5 folds for each of 80 candidates, totalling 400 fits
Best Hyperparameters: {'classifier__max_depth': 10, 'classifier__min_samples_leaf': 7, 'classifier__min_samples_split':
2, 'classifier__n_estimators': 100}
```

In [69]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
import numpy as np

# Random Forest Classifier
rf_classifier = RandomForestClassifier(n_estimators=100, max_depth=10, min_samples_split=2, min_samples_leaf=7, random_
rf_classifier.fit(X_train, y_train)

# Model evaluation

# Random Forest
rf_cv_scores = cross_val_score(rf_classifier, X, y, cv=11)
```

```
print(f"Cross-validation scores: {rf_cv_scores}")
print(f"Random Forest - Mean CV accuracy: {rf_cv_scores.mean()}")
```

```
Cross-validation scores: [0.80246914 0.81481481 0.80246914 0.82716049 0.90123457 0.85185185
 0.81481481 0.81481481 0.7654321  0.86419753 0.83950617]
Random Forest - Mean CV accuracy: 0.8271604938271605
```

In [70]:
```
# # Decision Tree
print(f"Decision Tree - Mean CV accuracy: {cv_scores.mean()}")
# Random Forest
print(f"Random Forest - Mean CV accuracy: {rf_cv_scores.mean()}")

# Compare the results
print(f"Decision Tree vs Random Forest: {np.mean(rf_cv_scores) - np.mean(cv_scores)}")
```

```
Decision Tree - Mean CV accuracy: 0.8204264870931538
Random Forest - Mean CV accuracy: 0.8271604938271605
Decision Tree vs Random Forest: 0.006734006734006703
```

In [ ]:
```

```