TITANIC SURVIVAL PREDICTION

- Utilize the Titanic dataset to construct a predictive model determining if a passenger survived the Titanic disaster.
- This project serves as an introductory exercise, offering accessible data for analysis.
- The dataset comprises passenger details encompassing age, gender, ticket class, fare, cabin, and survival outcome.
- By applying this data, you can embark on a classic project that provides insights into survival patterns among Titanic passengers.

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [4]:
```
df = pd.read_csv('tested.csv')
df
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
413	1305	0	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	NaN	S
414	1306	1	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C105	C
415	1307	0	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500	NaN	S
416	1308	0	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500	NaN	S
417	1309	0	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583	NaN	C

418 rows × 12 columns

In [5]:
```
df.head()
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

In [6]:
```
df.tail()
```

Out[6]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
413	1305	0	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	NaN	S
414	1306	1	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C105	C
415	1307	0	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500	NaN	S
416	1308	0	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500	NaN	S
417	1309	0	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583	NaN	C

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Survived     418 non-null    int64  
 2   Pclass       418 non-null    int64  
 3   Name         418 non-null    object 
 4   Sex          418 non-null    object 
 5   Age          332 non-null    float64
 6   SibSp        418 non-null    int64  
 7   Parch        418 non-null    int64  
 8   Ticket       418 non-null    object 
 9   Fare         417 non-null    float64
 10  Cabin        91 non-null     object 
 11  Embarked     418 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 39.3+ KB
```

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

Out[8]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	418.000000	418.000000	418.000000	332.000000	418.000000	418.000000	417.000000
mean	1100.500000	0.363636	2.265550	30.272590	0.447368	0.392344	35.627188
std	120.810458	0.481622	0.841838	14.181209	0.896760	0.981429	55.907576
min	892.000000	0.000000	1.000000	0.170000	0.000000	0.000000	0.000000
25%	996.250000	0.000000	1.000000	21.000000	0.000000	0.000000	7.895800
50%	1100.500000	0.000000	3.000000	27.000000	0.000000	0.000000	14.454200
75%	1204.750000	1.000000	3.000000	39.000000	1.000000	0.000000	31.500000
max	1309.000000	1.000000	3.000000	76.000000	8.000000	9.000000	512.329200

In [9]:
```
df.describe(include="object")
```

Out[9]:

	Name	Sex	Ticket	Cabin	Embarked
count	418	418	418	91	418
unique	418	2	363	76	3
top	Peter, Master. Michael J	male	PC 17608	B57 B59 B63 B66	S
freq	1	266	5	3	270

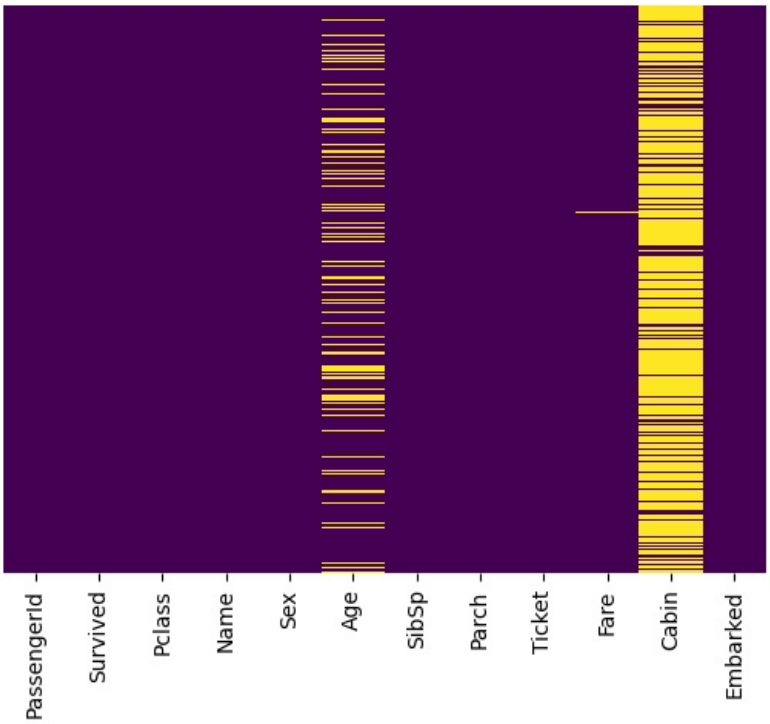
In [10]:
```
df.isnull().sum()
```

Out[10]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

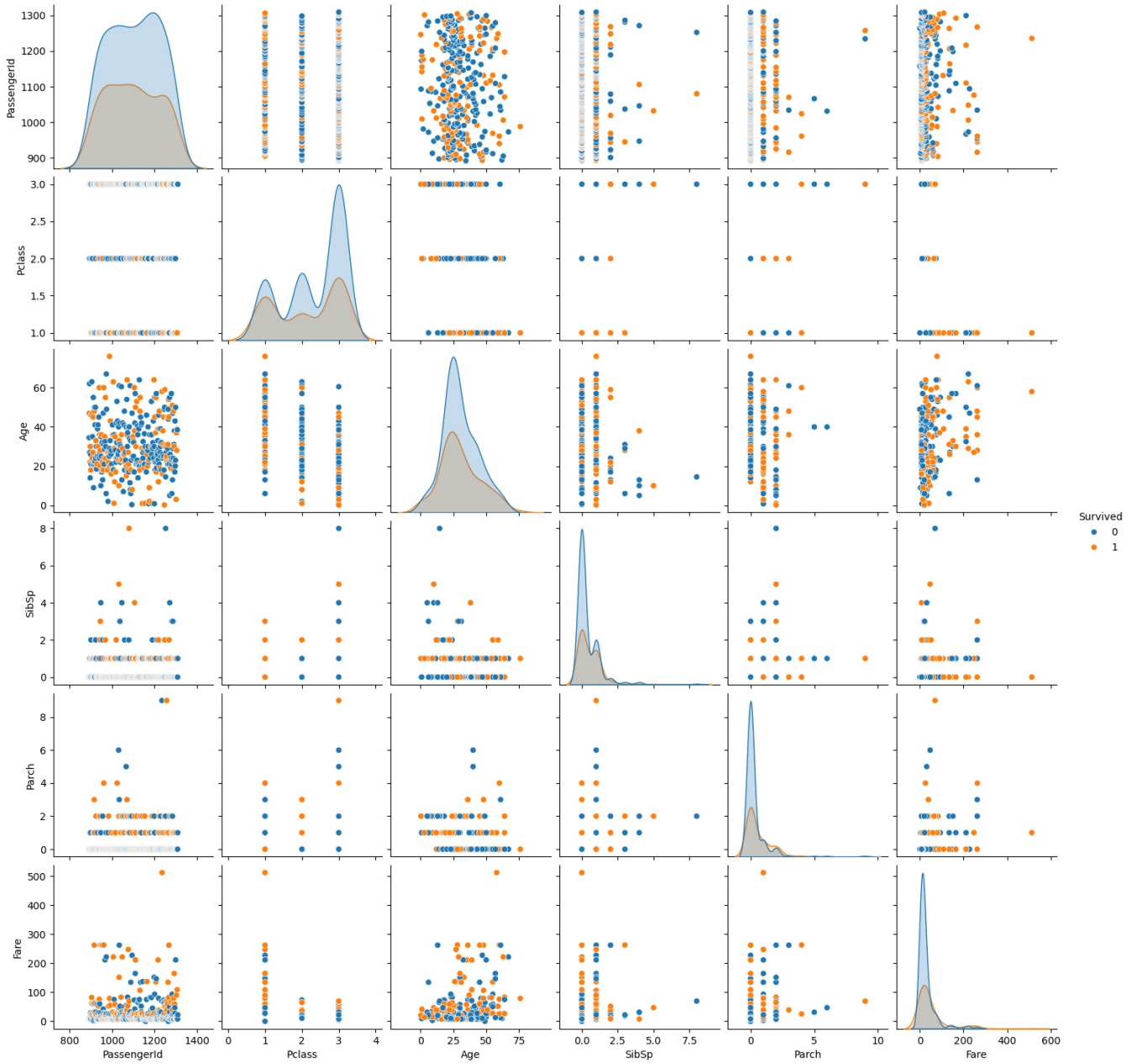
Data Visualization

In [11]:
```
#visualization of null value using heatmap
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

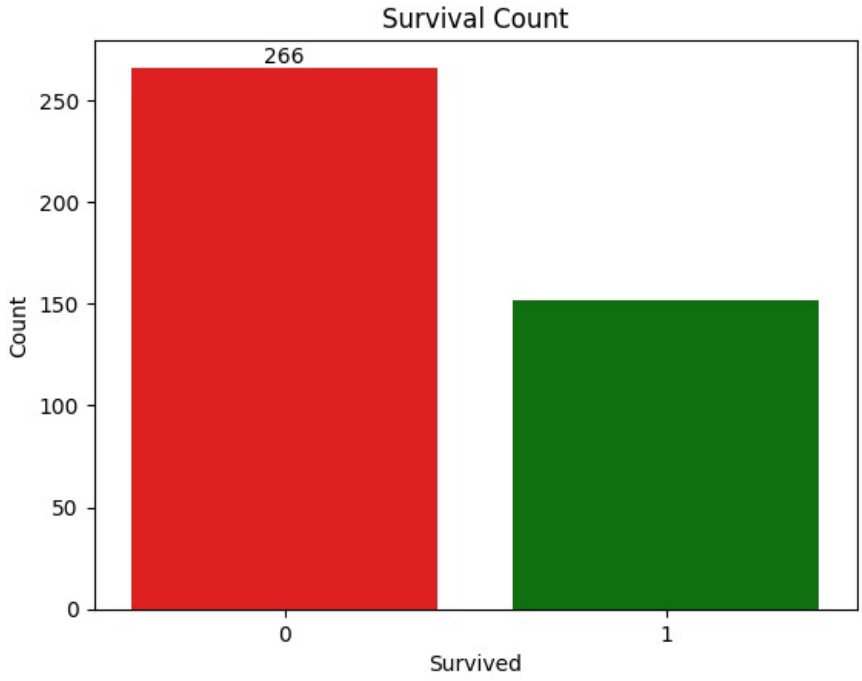
Out[11]:
```
<Axes: >
```



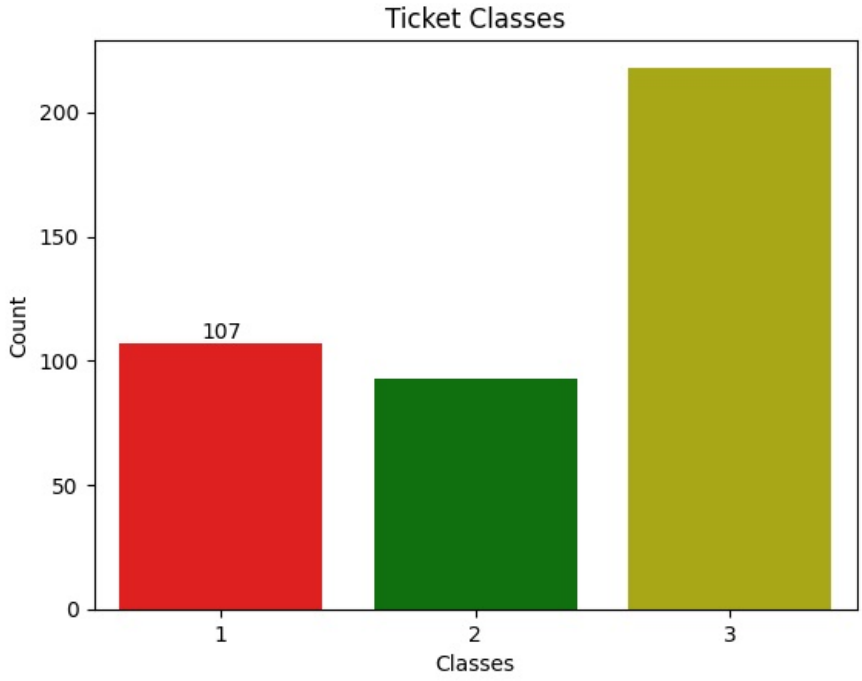
In [12]:
```
# Visualize the data distribution
sns.pairplot(df, hue='Survived')
plt.show()
```



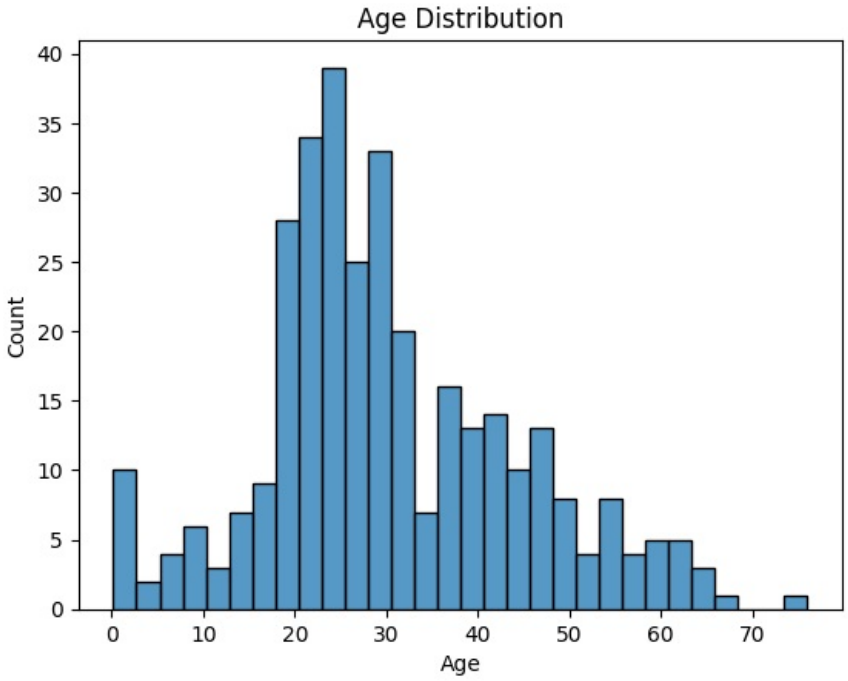
In [13]:
```
# Survival Count
data1 = df["Survived"].value_counts().reset_index()
data1.columns = ["Survived", "Count"]
bar1 = sns.barplot(x=data1["Survived"], y=data1["Count"], palette=['r', 'g'])
bar1.bar_label(bar1.containers[0])
plt.title("Survival Count")
plt.xlabel("Survived")
plt.ylabel("Count")
plt.show()
```



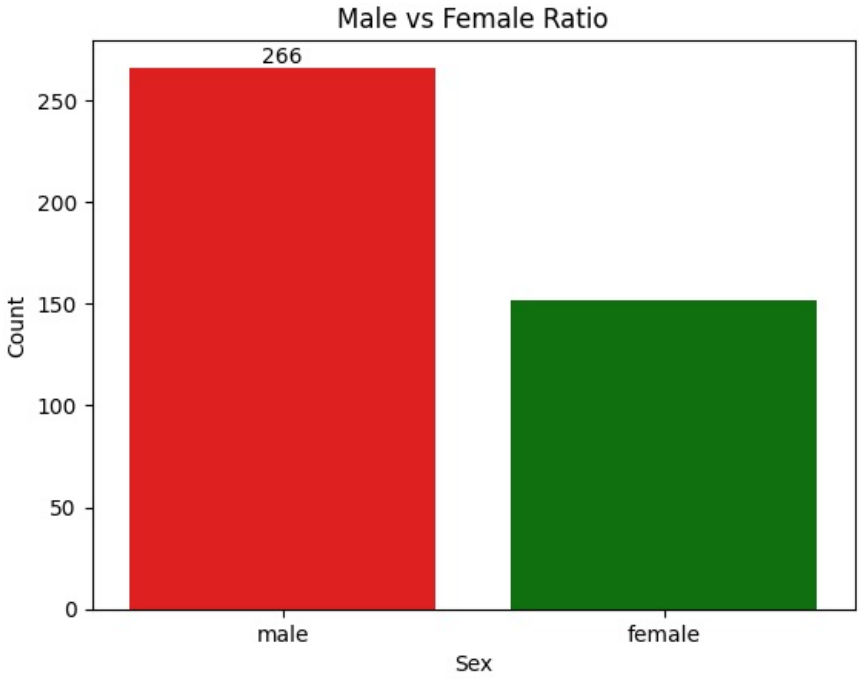
In [14]:
```
# Ticket Classes
data2 = df["Pclass"].value_counts().reset_index()
data2.columns = ["Pclass", "Count"]
bar2 = sns.barplot(x=data2["Pclass"], y=data2["Count"], palette=['r', 'g', 'y'])
bar2.bar_label(bar2.containers[0])
plt.title("Ticket Classes")
plt.xlabel("Classes")
plt.ylabel("Count")
plt.show()
```



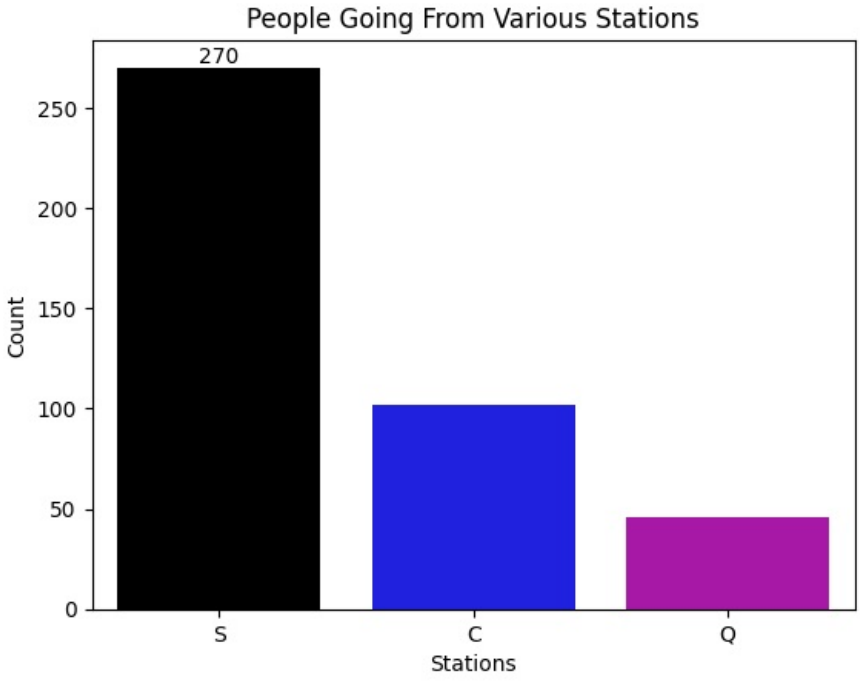
In [15]:
```
# Age Distribution
sns.histplot(x=df["Age"], bins=30)
plt.xlabel("Age")
plt.ylabel("Count")
plt.title("Age Distribution")
plt.show()
```



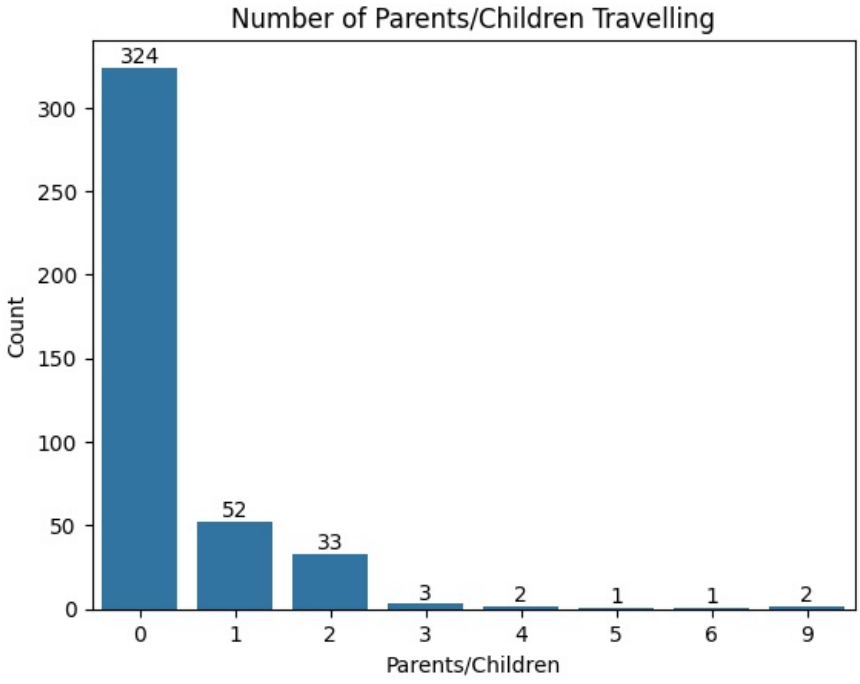
In [16]:
```
# Male vs Female Ratio
data3 = df["Sex"].value_counts().reset_index()
data3.columns = ["Sex", "Count"]
bar3 = sns.barplot(x=data3["Sex"], y=data3["Count"], palette=['r', 'g'])
bar3.bar_label(bar3.containers[0])
plt.title("Male vs Female Ratio")
plt.xlabel("Sex")
plt.ylabel("Count")
plt.show()
```



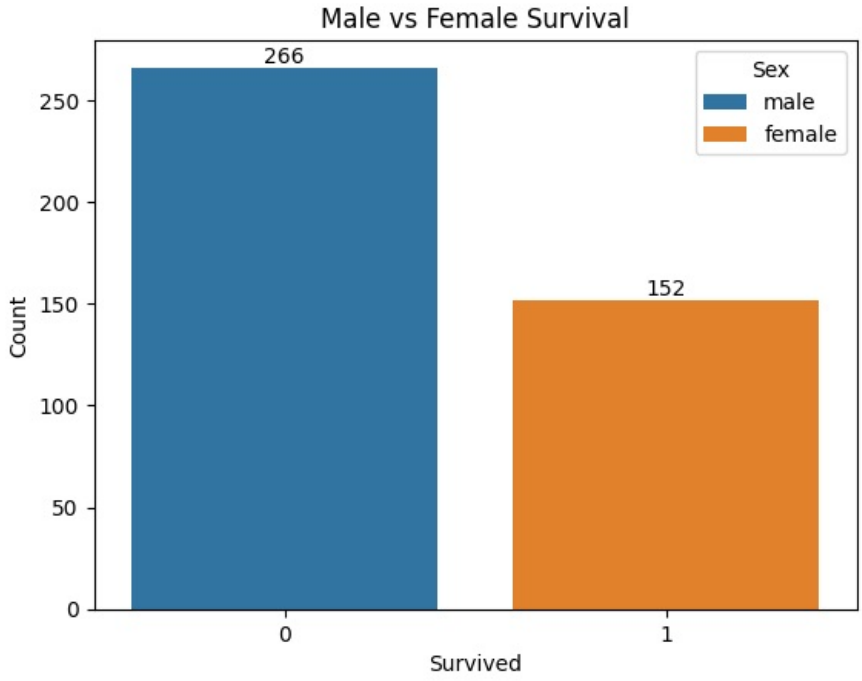
In [17]:
```
# People Going From Various Stations
data4 = df["Embarked"].value_counts().reset_index()
data4.columns = ["Embarked", "Count"]
bar4 = sns.barplot(x=data4["Embarked"], y=data4["Count"], palette=['k', 'b', 'm'])
bar4.bar_label(bar4.containers[0])
plt.title("People Going From Various Stations")
plt.xlabel("Stations")
plt.ylabel("Count")
plt.show()
```



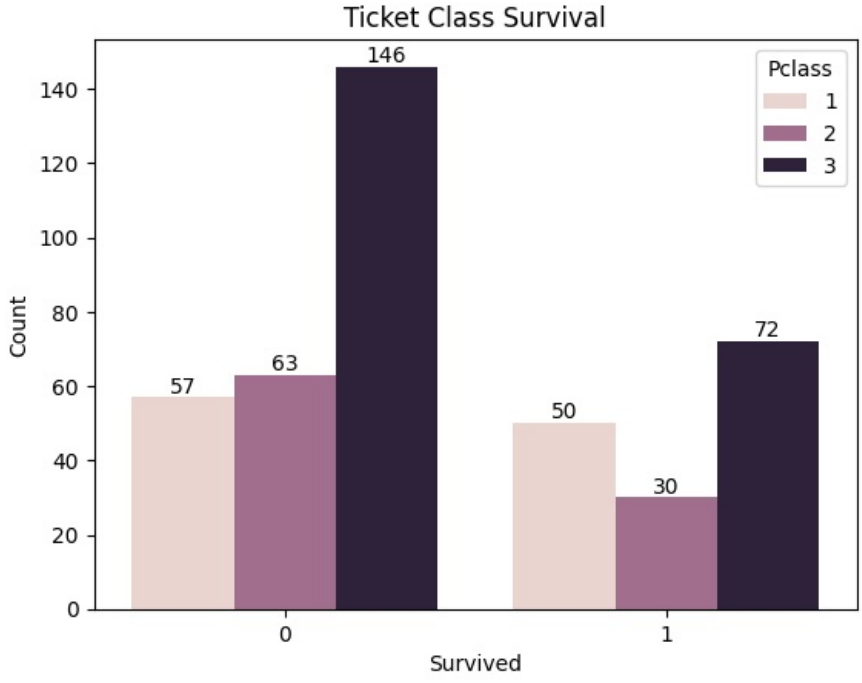
In [18]:
```
# Number of Parents/Children Travelling
data5 = df["Parch"].value_counts().reset_index()
data5.columns = ["Parch", "Count"]
bar5 = sns.barplot(x=data5["Parch"], y=data5["Count"])
bar5.bar_label(bar5.containers[0])
plt.title("Number of Parents/Children Travelling")
plt.xlabel("Parents/Children")
plt.ylabel("Count")
plt.show()
```



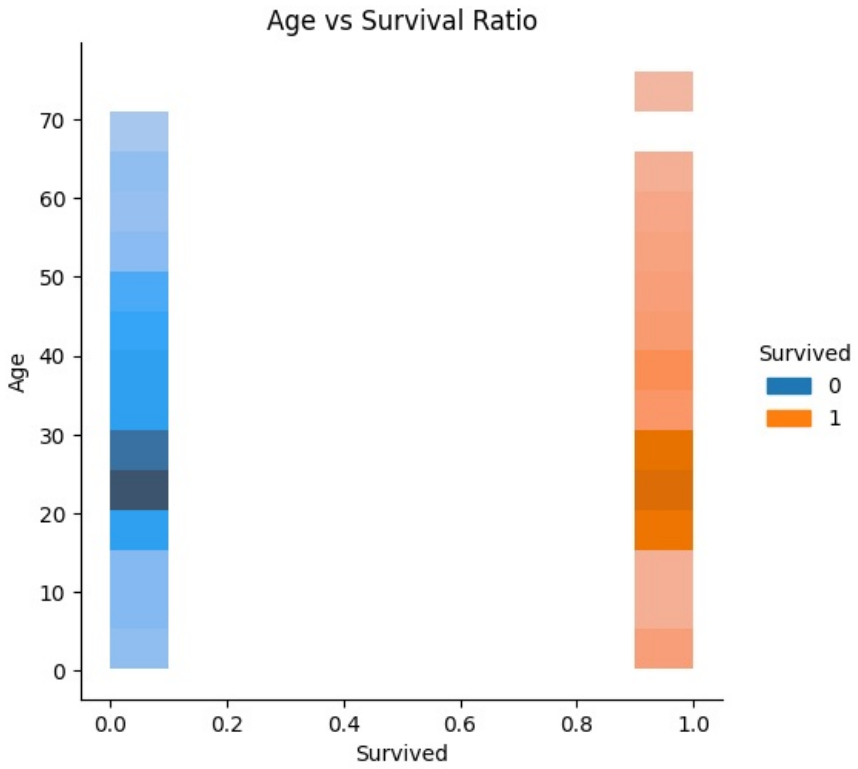
In [19]:
```
# Male vs Female Survival
data8 = df[["Survived", "Sex"]].value_counts().reset_index()
data8.columns = ["Survived", "Sex", "Count"]
bar8 = sns.barplot(x=data8["Survived"], y=data8["Count"], hue=data8["Sex"])
bar8.bar_label(bar8.containers[0])
bar8.bar_label(bar8.containers[1])
plt.title("Male vs Female Survival")
plt.xlabel("Survived")
plt.ylabel("Count")
plt.show()
```



In [20]:
```
# Ticket Class Survival
data9 = df[["Survived", "Pclass"]].value_counts().reset_index()
data9.columns = ["Survived", "Pclass", "Count"]
bar9 = sns.barplot(x=data9["Survived"], y=data9["Count"], hue=data9["Pclass"])
bar9.bar_label(bar9.containers[0])
bar9.bar_label(bar9.containers[1])
bar9.bar_label(bar9.containers[2])
plt.title("Ticket Class Survival")
plt.xlabel("Survived")
plt.ylabel("Count")
plt.show()
```



In [21]:
```
# Age vs Survival Ratio
sns.displot(x=df["Survived"], y=df["Age"], hue=df["Survived"])
plt.title("Age vs Survival Ratio")
plt.show()
```



Data Preprocessing

In [22]:
```
# Fill missing Age values with median
df['Age'].fillna(df['Age'].median(), inplace=True)
# Replace missing Fare value with the median Fare
df['Fare'].fillna(df['Fare'].median(), inplace=True)
```

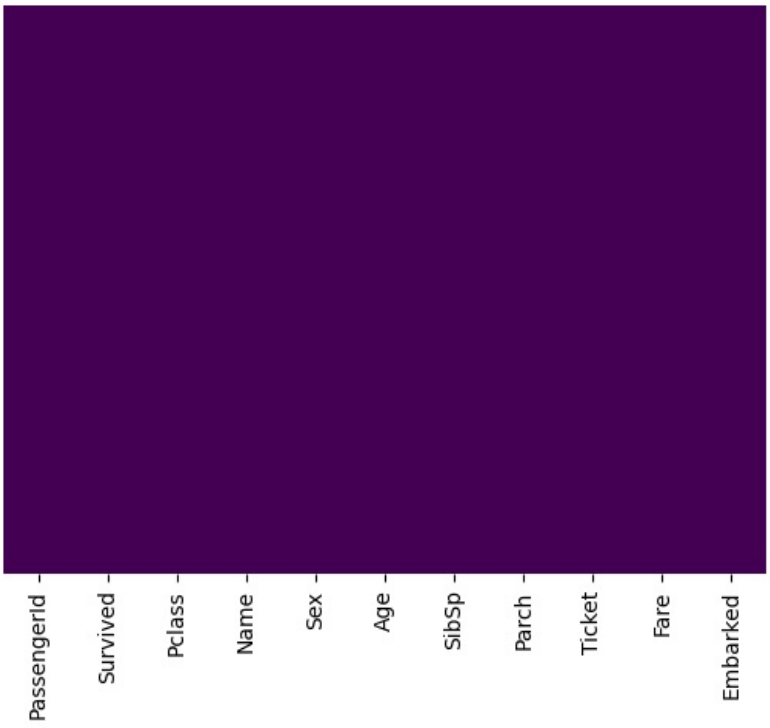
In [23]:
```
# Drop Cabin column due to too many missing values
df.drop(columns=['Cabin'], inplace=True)
```

In [24]:
```
df.isnull().sum()
```

Out[24]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

In [25]:
```
# Again, visualize the null values clearly using a heatmap.
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

Out[25]:
```
<Axes: >
```



In [26]:
```
# Convert categorical variables to numerical
df['Sex'] = df['Sex'].map({'male': 0, 'female': 1})
df['Embarked'] = df['Embarked'].map({'S': 0, 'C': 1, 'Q': 2})
```

In [27]:
```
# Split the dataset
X = df.drop(columns=['Survived', 'PassengerId', 'Name', 'Ticket'])
y = df['Survived']
```

In [28]:
```
X
```

Out[28]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	0	34.5	0	0	7.8292	2
1	3	1	47.0	1	0	7.0000	0
2	2	0	62.0	0	0	9.6875	2
3	3	0	27.0	0	0	8.6625	0
4	3	1	22.0	1	1	12.2875	0
...	...	...	...	...	...	...	...
413	3	0	27.0	0	0	8.0500	0
414	1	1	39.0	0	0	108.9000	1
415	3	0	38.5	0	0	7.2500	0
416	3	0	27.0	0	0	8.0500	0
417	3	0	27.0	1	1	22.3583	1

418 rows × 7 columns

In [29]:
```
y
```

Out[29]:
```
0      0
1      1
2      0
3      0
4      1
      ..
413    0
414    1
415    0
416    0
417    0
Name: Survived, Length: 418, dtype: int64
```

In [30]:
```
# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Building the Model

In [31]:
```
# Initialize and train the model
model = LogisticRegression()#max_iter=200
model.fit(X_train, y_train)
```

Out[31]:

▾ LogisticRegression

```
LogisticRegression()
```

In [32]:
```
# Predictions
y_pred = model.predict(X_test)
```

In [33]:
```
y_pred
```

Out[33]:
```
array([0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0,
       0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1])
```

In [34]:
```
y_test
```

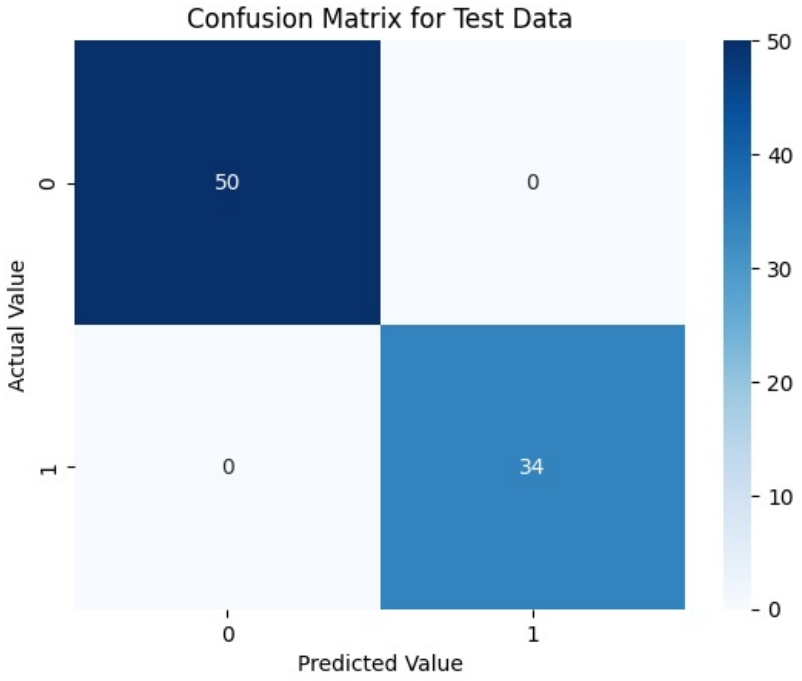
Out[34]:
```
321    0
324    1
388    0
56     0
153    1
      ..
57     0
126    0
24     1
17     0
66     1
Name: Survived, Length: 84, dtype: int64
```

In [35]:
```
# Evaluate the model performance matrix using confusion matrix.
cm = confusion_matrix(y_test, y_pred)
```

In [36]:
```
print(f'Confusion Matrix:\n{cm}')
```

```
Confusion Matrix:
[[50  0]
 [ 0 34]]
```

In [37]:
```
# Confusion matrix
conf_matrix = confusion_matrix(y_test,y_pred )
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='d')
plt.xlabel('Predicted Value')
plt.ylabel('Actual Value')
plt.title('Confusion Matrix for Test Data')
plt.show()
```



In [38]:
```
#check accuracy of our model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
```

In [39]:
```
print(f'Accuracy: {accuracy}')
print(f'Classification Report:\n{report}')
```

```
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       1.00      1.00      1.00        34

    accuracy                           1.00        84
   macro avg       1.00      1.00      1.00        84
weighted avg       1.00      1.00      1.00        84
```

In [ ]:

In [ ]:

In [ ]: