In [23]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

In [24]:
```
df = pd.read_csv('Titanic-Dataset.csv')
```

In [25]:
```
df.head(10)
```

Out[25]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708

In [26]:
```
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 [27]:
```
df.isnull().sum()
```

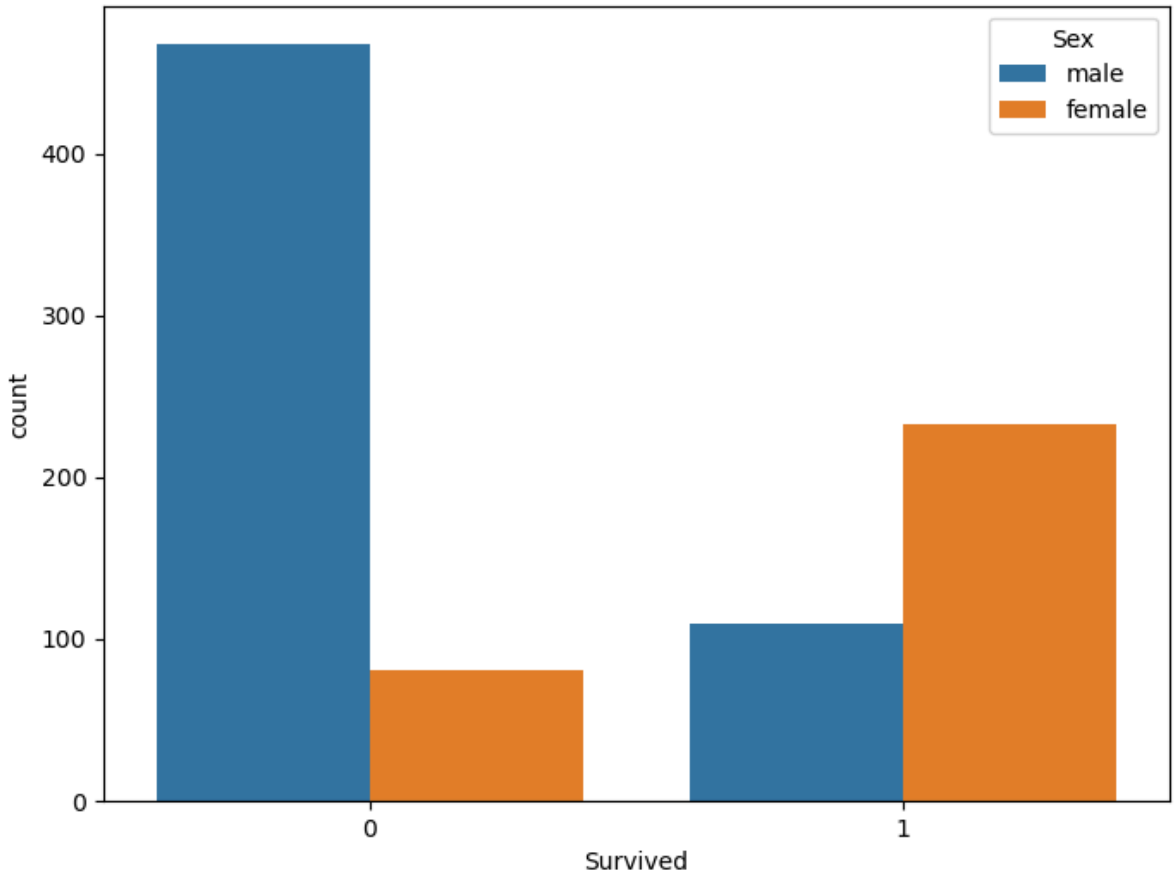
Out[27]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [28]:
```
df.size
```

Out[28]:
```
10692
```

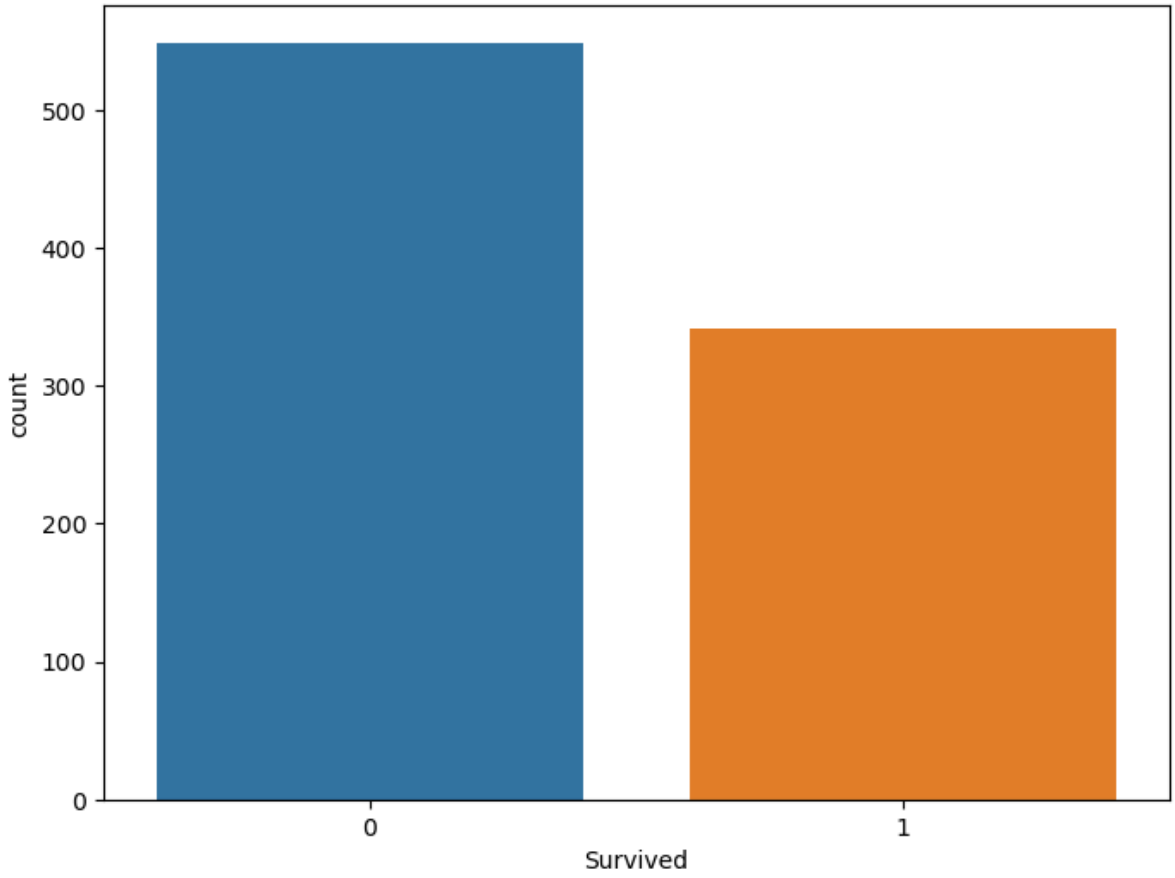
In [36]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.style.use("default")   # reset to normal white background
plt.figure(figsize=(8, 6))
sns.countplot(x="Survived", hue="Sex", data=df)
plt.show()
```

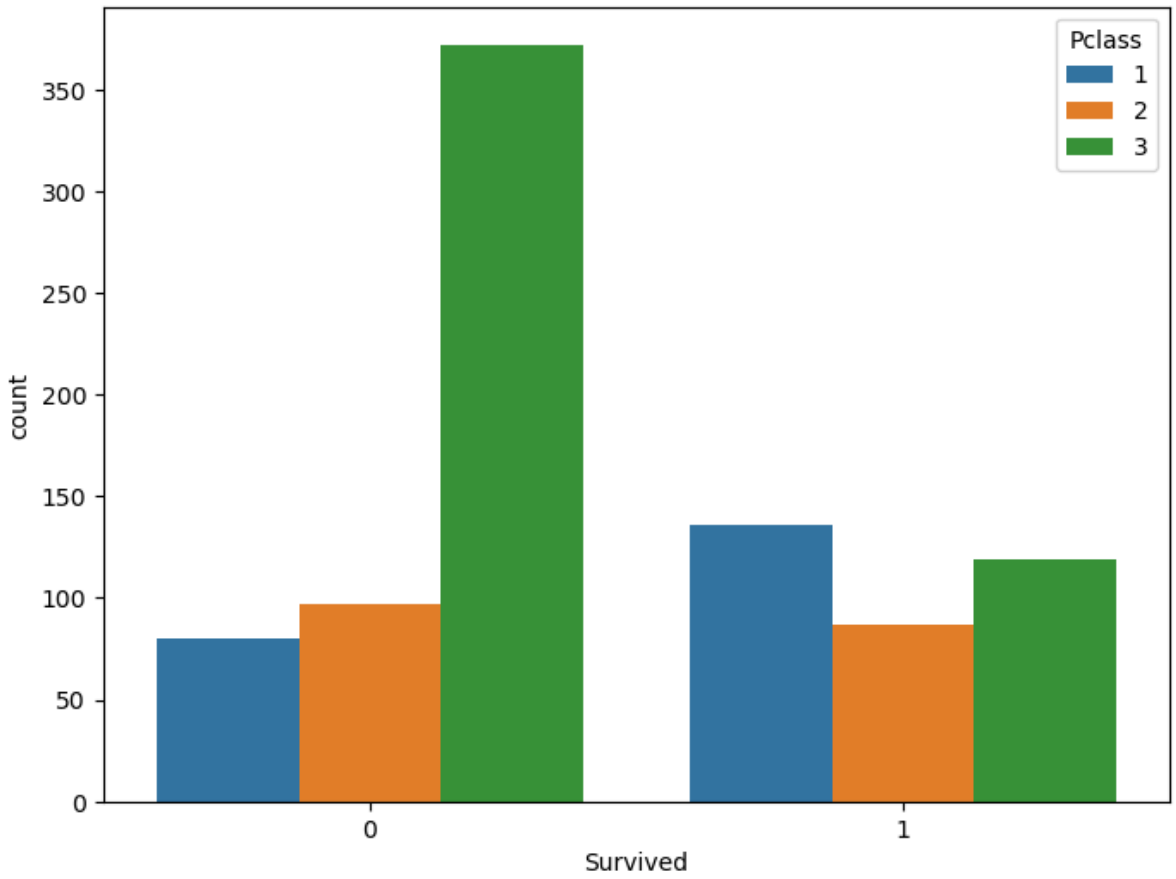


In [35]:
```
plt.figure(figsize=(8, 6))
plt.gca().set_facecolor("white")
sns.countplot(x = "Survived", data = df)
```

Out[35]:
```
<AxesSubplot:xlabel='Survived', ylabel='count'>
```

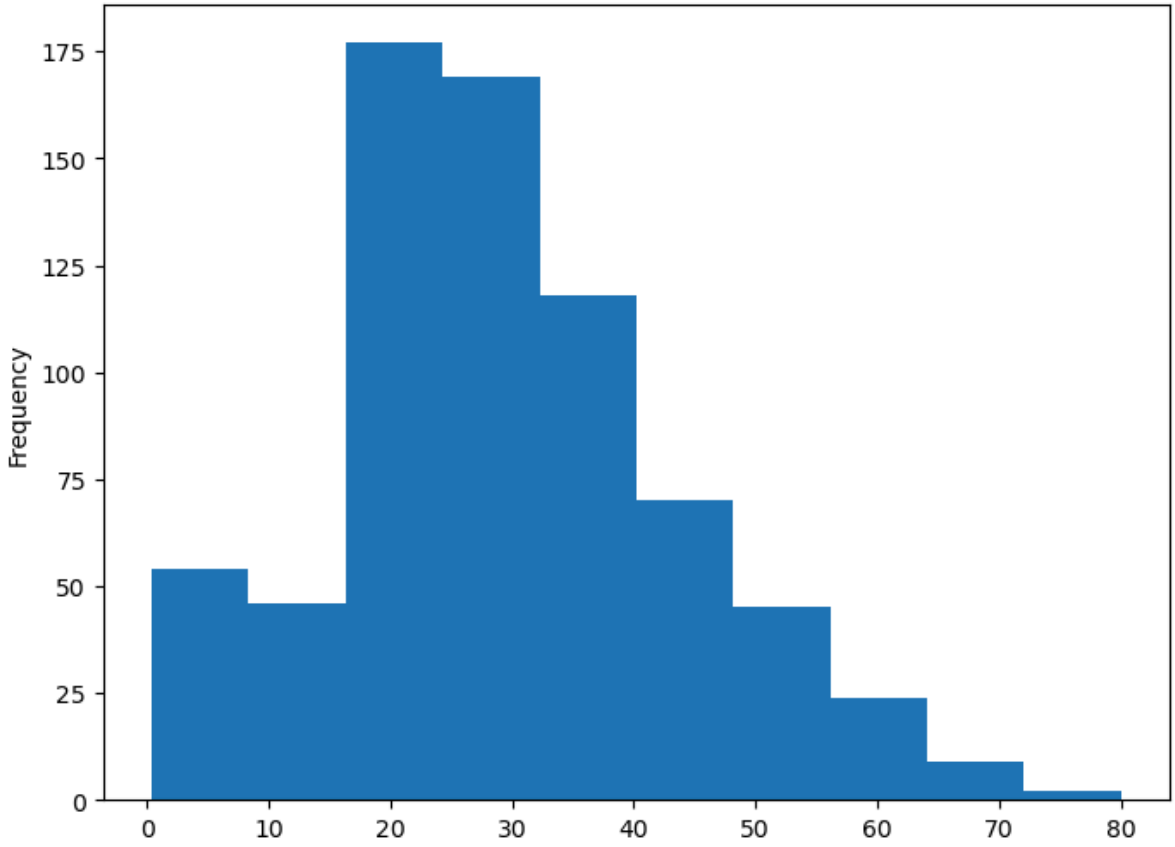


In [37]:
```
plt.figure(figsize=(8, 6))
sns.countplot(x="Survived", hue= "Pclass", data=df)
plt.show()
```



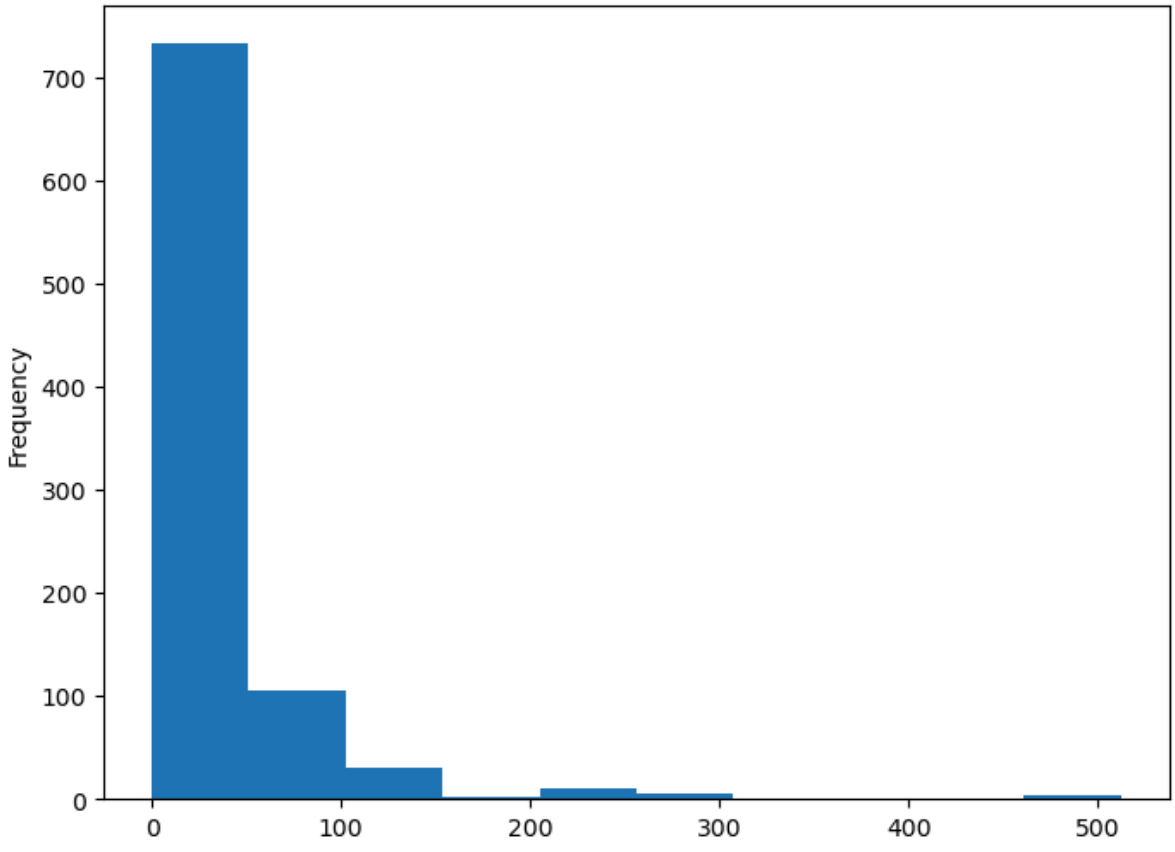
In [42]:
```
plt.figure(figsize=(8, 6))
df['Age'].plot.hist()
```

Out[42]:
```
<AxesSubplot:ylabel='Frequency'>
```



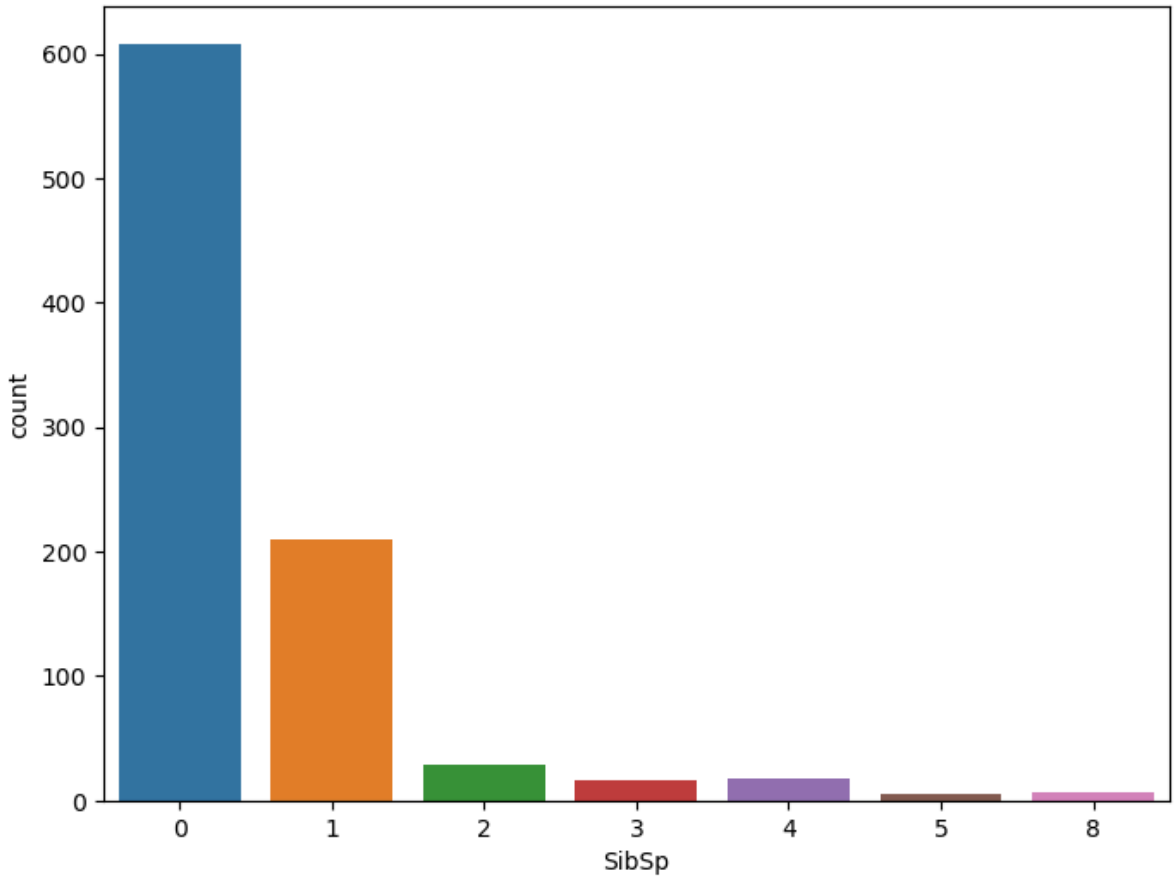
In [43]:
```
plt.figure(figsize=(8, 6))
df['Fare'].plot.hist()
```

Out[43]:
```
<AxesSubplot:ylabel='Frequency'>
```



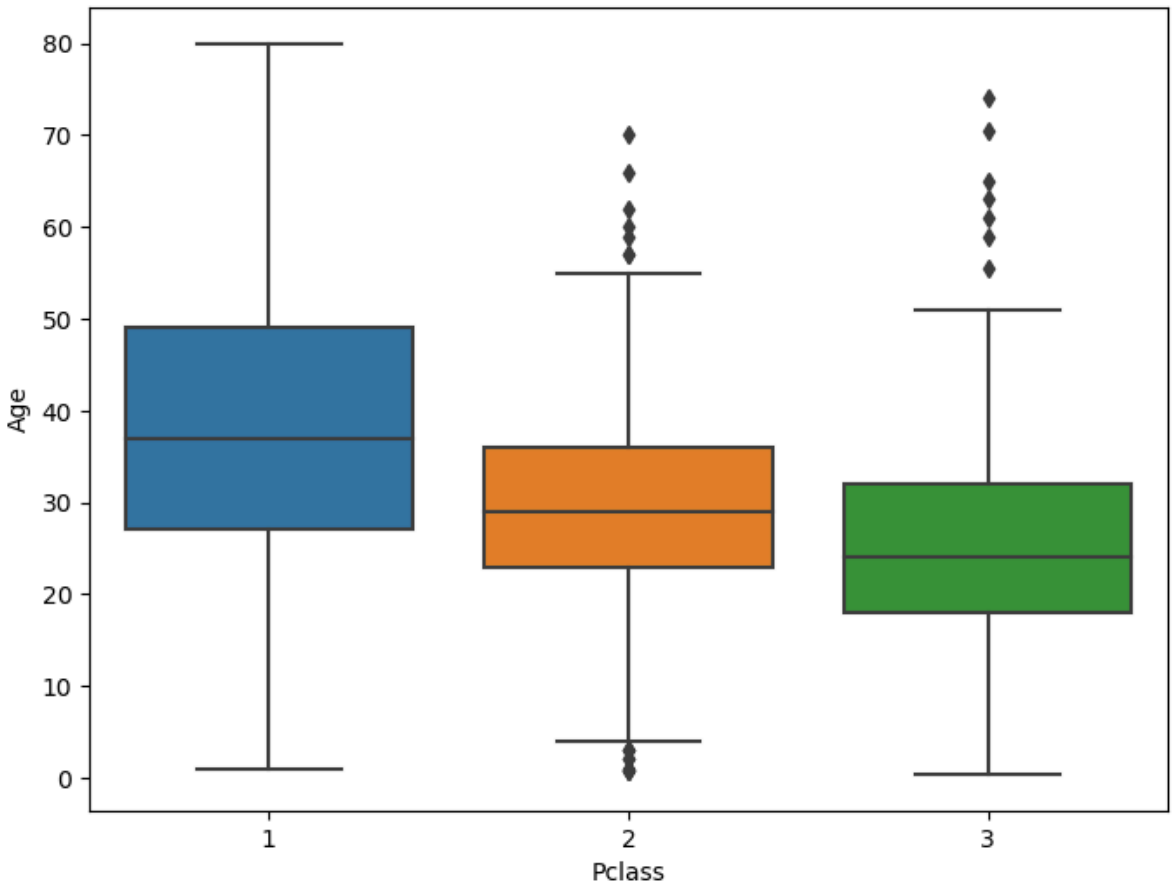
In [48]:
```
plt.figure(figsize=(8, 6))
sns.countplot(x = "SibSp", data = df)
```

Out[48]: <AxesSubplot:xlabel='SibSp', ylabel='count'>



In [49]:
```
plt.figure(figsize=(8, 6))
sns.boxplot(x = "Pclass", y = "Age" , data = df)
```

Out[49]: <AxesSubplot:xlabel='Pclass', ylabel='Age'>



In [51]:
```
df.drop("Cabin", axis = 1, inplace = True)
```

In [52]:
```
df
```

Out[52]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	F
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.28
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.10
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05
...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75

891 rows × 11 columns

In [53]:
```
df.isnull().sum()
```

Out[53]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Embarked         2
dtype: int64
```

In [54]:
```
df.size
```

Out[54]:
```
9801
```

In [55]:
```
df = df.dropna()
df
```

Out[55]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	F
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.28
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.10
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05
...	...	...	...	...	...	...	...	...	...	
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.12
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75

712 rows × 11 columns

create dummy variables

In [85]:
```
sex = pd.get_dummies(df["Sex"], drop_first=True)
Embarked = pd.get_dummies(df["Embarked"], drop_first=True)
Pclass = pd.get_dummies(df["Pclass"], drop_first=True)
df = pd.concat([df, Pclass ], axis=1)
```

In [91]:
```
df.drop(['PassengerId','Pclass','Name','Sex','Ticket','Embarked'], axis=1,
```

In [122…
```
 X.columns.astype(str)
```

Out[122]:
```
Index(['Age', 'SibSp', 'Parch', 'Fare', 'male', 'Q', 'S', '2', '3'], dtype
='object')
```

Train Data

In [118…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 712 entries, 0 to 890
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  712 non-null    int64
 1   Age       712 non-null    float64
 2   SibSp     712 non-null    int64
 3   Parch     712 non-null    int64
 4   Fare      712 non-null    float64
 5   male      712 non-null    uint8
 6   Q         712 non-null    uint8
 7   S         712 non-null    uint8
 8   2         712 non-null    uint8
 9   3         712 non-null    uint8
dtypes: float64(2), int64(3), uint8(5)
memory usage: 36.9 KB
```

In [126…
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [127…
```
from sklearn.linear_model import LogisticRegression
```

In [131…
```
X = df.drop("Survived", axis=1)   # Features
y = df["Survived"]                # Target
```

In [142…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ran
```

In [143…
```
from sklearn.linear_model import LogisticRegression
```

In [144…
```
logmodel =  LogisticRegression()
```

In [160…
```
df.head(2)
```

Out[160]:

	Survived	Age	SibSp	Parch	Fare	male	Q	S	2	3
0	0	22.0	1	0	7.2500	1	0	1	0	1
1	1	38.0	1	0	71.2833	0	0	0	0	0

In [161…
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classificatio

# Features & Target
X = df.drop("Survived", axis=1)  # features
y = df["Survived"]               # target

# Fix column names if mixed types
X.columns = X.columns.astype(str)
```

```
# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Model
logmodel = LogisticRegression(max_iter=1000)

# Train
logmodel.fit(X_train, y_train)

# Predict
y_pred = logmodel.predict(X_test)

# Evaluation
print("✅ Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
✅ Accuracy: 0.8111888111888111

Confusion Matrix:
 [[72 13]
 [14 44]]

Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.85      0.84        85
           1       0.77      0.76      0.77        58

    accuracy                           0.81       143
   macro avg       0.80      0.80      0.80       143
weighted avg       0.81      0.81      0.81       143
```

predict survival for a single new passenger

In [163…

```
import pandas as pd

new_passenger = pd.DataFrame([{
    "Age": 30,
    "SibSp": 0,
    "Parch": 0,
    "Fare": 50,
    "male": 1,
    "Q": 0,
    "S": 1,
    "2": 0,   # Pclass 2
    "3": 1    # Pclass 3
}])

prediction = logmodel.predict(new_passenger)
probability = logmodel.predict_proba(new_passenger)

print("Prediction:", prediction[0])
print("Survival Probability:", probability[0][1])
```

```
Prediction: 0
Survival Probability: 0.10293439657500156
```

In [170…

```
new_passenge_2 = pd.DataFrame([{
    "Age": 38.0,
    "SibSp": 1,
    "Parch": 0,
    "Fare": 71,
    "male": 0,
    "Q": 0,
    "S": 0,
    "2": 0,   # Pclass 2
    "3": 0    # Pclass 3
}])


rediction = logmodel.predict(new_passenge_2)
probability = logmodel.predict_proba(new_passenge_2)

print("Prediction:", prediction[0])
print("Survival Probability:", probability[0][1])
```

```
Prediction: 0
Survival Probability: 0.9124171917665683
```

RandomForestClassifier

In [172…

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classificatior
import pandas as pd

# Features & Target
X = df.drop("Survived", axis=1)
y = df["Survived"]

# Make sure all column names are strings
X.columns = X.columns.astype(str)

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Random Forest Model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Predictions
y_pred = rf_model.predict(X_test)

# --- Evaluation ---
print("✅ Random Forest Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# --- Feature Importances ---
importances = rf_model.feature_importances_
feature_names = X.columns

feat_df = pd.DataFrame({
    "Feature": feature_names,
    "Importance": importances
}).sort_values(by="Importance", ascending=False)
```

```
print("\n✅ Important Features:\n", feat_df)
```

```
✅ Random Forest Accuracy: 0.7902097902097902

Confusion Matrix:
 [[69 16]
 [14 44]]

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.81      0.82        85
           1       0.73      0.76      0.75        58

    accuracy                           0.79       143
   macro avg       0.78      0.79      0.78       143
weighted avg       0.79      0.79      0.79       143


✅ Important Features:
   Feature  Importance
0      Age    0.289684
3     Fare    0.254662
4     male    0.247501
8        3    0.071604
1    SibSp    0.050847
2    Parch    0.038903
7        2    0.022273
6        S    0.021602
5        Q    0.002924
```

In [ ]:

In [ ]: