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

```
titanic_data=pd.read_csv('/content/titanic_train.csv')
len(titanic_data)
titanic_data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.250
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.283
				Heikkinen						

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

```
titanic_data.index
titanic_data.columns
```

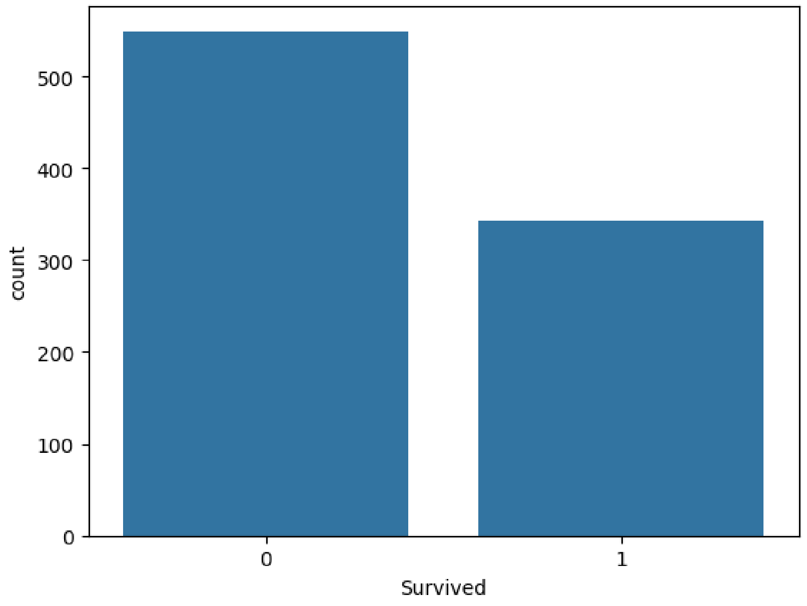
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

```
titanic_data.info()
titanic_data.describe()
```

```
<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
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.00000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.20420
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.69342
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.00000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.91040
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.45420
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.00000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.32920

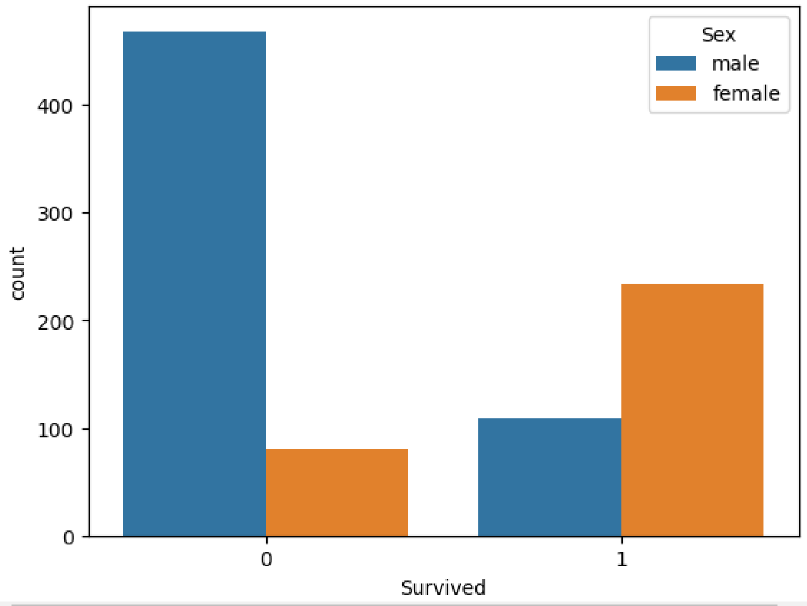
```
sns.countplot(x='Survived',data=titanic_data)

    <Axes: xlabel='Survived', ylabel='count'>
```



```
sns.countplot(x='Survived',data=titanic_data,hue='Sex')
titanic_data.isna()
```

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

891 rows × 12 columns

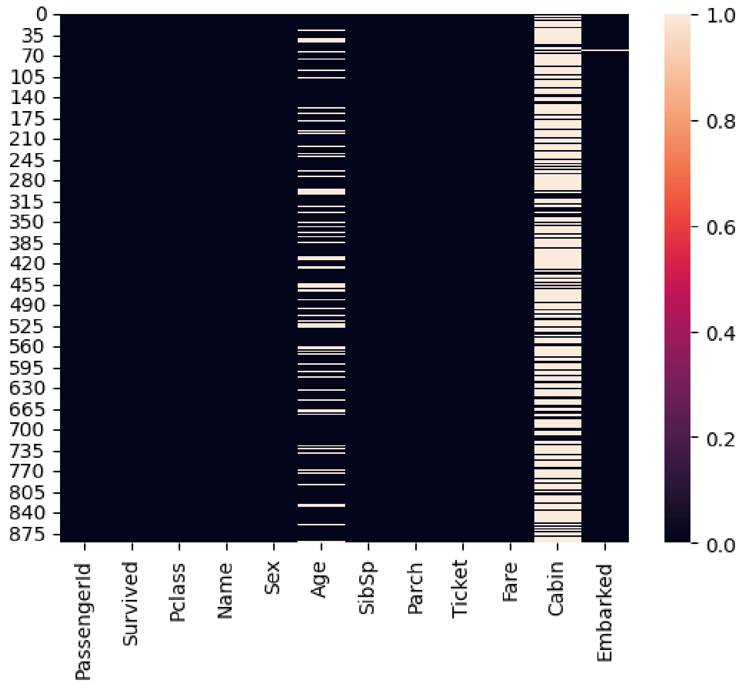


```
titanic_data.isna().sum()
```

```
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
```

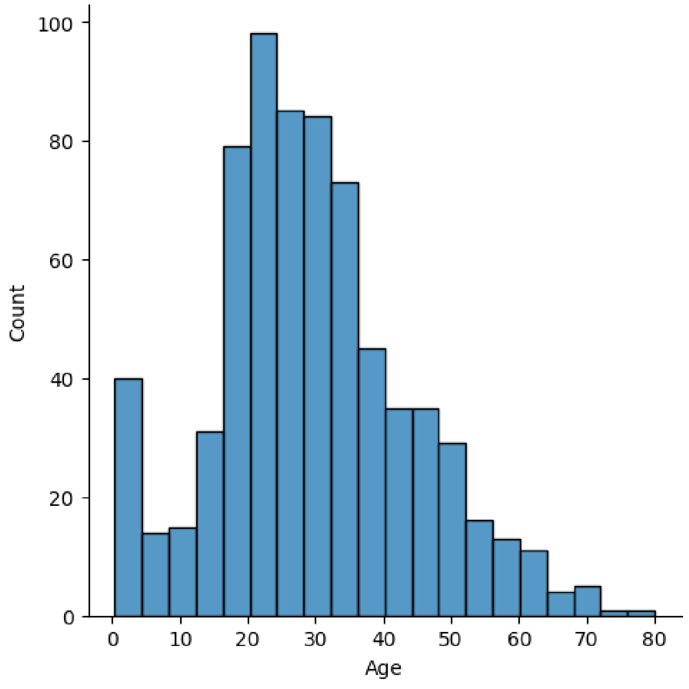
```
sns.heatmap(titanic_data.isna())
```

```
<Axes: >
```



```
(titanic_data['Age'].isna().sum()/len(titanic_data['Age']))*100
(titanic_data['Cabin'].isna().sum()/len(titanic_data['Cabin']))*100
sns.displot(x='Age',data=titanic_data)
```

```
<seaborn.axisgrid.FacetGrid at 0x7ee58515d810>
```

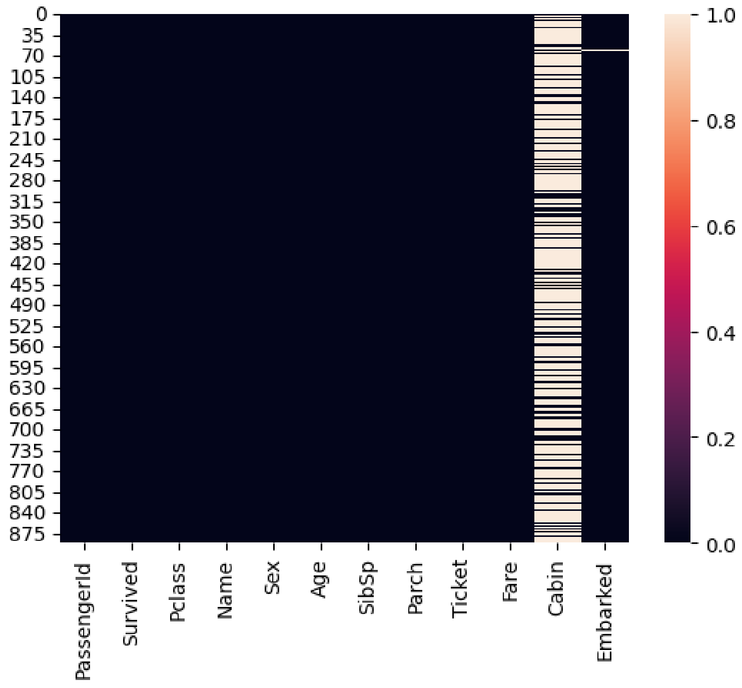


```
titanic_data['Age'].fillna(titanic_data['Age'].mean(),inplace=True)
titanic_data['Age'].isna().sum()
```

```
0
```

```
sns.heatmap(titanic_data.isna())
```

```
<Axes: >
```



```
titanic_data.drop('Cabin',axis=1,inplace=True)
titanic_data.head()
titanic_data.info()
titanic_data.dtypes
gender=pd.get_dummies(titanic_data['Sex'],drop_first=True)
titanic_data['Gender']=gender
titanic_data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 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          891 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  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(4)
memory usage: 76.7+ KB
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.250
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.283
				Heikkinen						

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

```
titanic_data.drop(['Name','Sex','Ticket','Embarked'],axis=1,inplace=True)
titanic_data.head()
x=titanic_data[['PassengerId','Pclass','Age','SibSp','Parch','Fare','Gender']]
y=titanic_data['Survived']
y
```

```
    0      0
    1      1
    2      1
    3      1
    4      0
          ..
    886    0
    887    1
    888    0
    889    1
    890    0
```

```
from sklearn.metrics import confusion_matrix
pd.DataFrame(confusion_matrix(y_test,predict),columns=['Predicted No','Predicted Yes'],index=['Actual No','Actual Yes'])
```

	Predicted No	Predicted Yes
Actual No	151	24
Actual Yes	37	83

```
#import train test split method
from sklearn.model_selection import train_test_split
#train test split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=42)
#import Logistic  Regression
from sklearn.linear_model import LogisticRegression
#Fit  Logistic Regression
lr=LogisticRegression()
lr.fit(x_train,y_train)
LogisticRegression()
#predict
predict=lr.predict(x_test)
from sklearn.metrics import classification_report
print(classification_report(y_test,predict))
from sklearn.metrics import confusion_matrix
pd.DataFrame(confusion_matrix(y_test,predict),columns=['Predicted No','Predicted Yes'],index=['Actual No','Actual Yes'])
```

```
                  precision    recall  f1-score   support

               0       0.80      0.86      0.83       175
               1       0.78      0.69      0.73       120

        accuracy                           0.79       295
       macro avg       0.79      0.78      0.78       295
    weighted avg       0.79      0.79      0.79       295

    /usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (s
    STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

    Increase the number of iterations (max_iter) or scale the data as shown in:
        https://scikit-learn.org/stable/modules/preprocessing.html
```