```
from google.colab import files
```

```
uploaded = files.upload()
```

```
Choose files TITANIC.csv
```

- **TITANIC.csv**(text/csv) - 29474 bytes, last modified: 11/08/2023 - 100% done

```
Saving TITANIC.csv to TITANIC.csv
```

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import os
for dirname, _, filenames in os.walk('TITANIC.csv'):
    for filename in filenames:
        print(os.path.join(dirname, filename))


# Load the dataset
tt = pd.read_csv('TITANIC.csv')
```

```
tt.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fa
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.82
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.00

```
tt.tail()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Em
413	1305	0	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	NaN	
414	1306	1	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C105	
				Saether								

```
tt.shape
```

```
(418, 12)
```

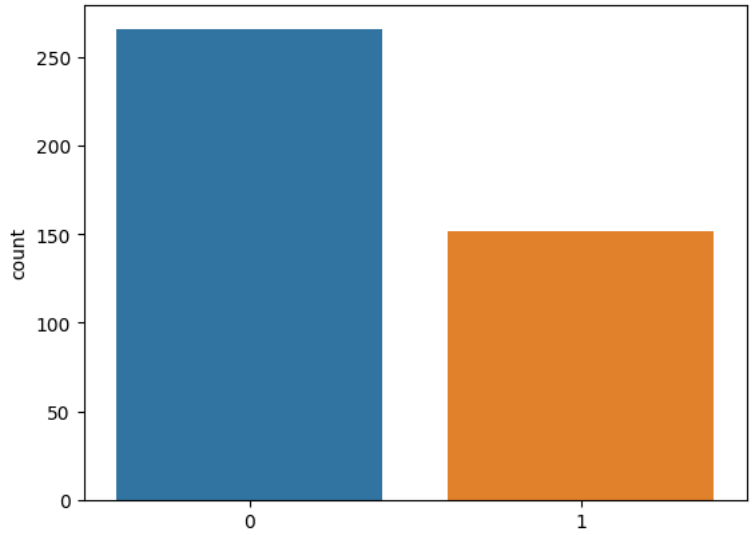
```
tt.describe()
```

	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

```
# Visualization libraries
import plotly.express as px
import seaborn as sns
```

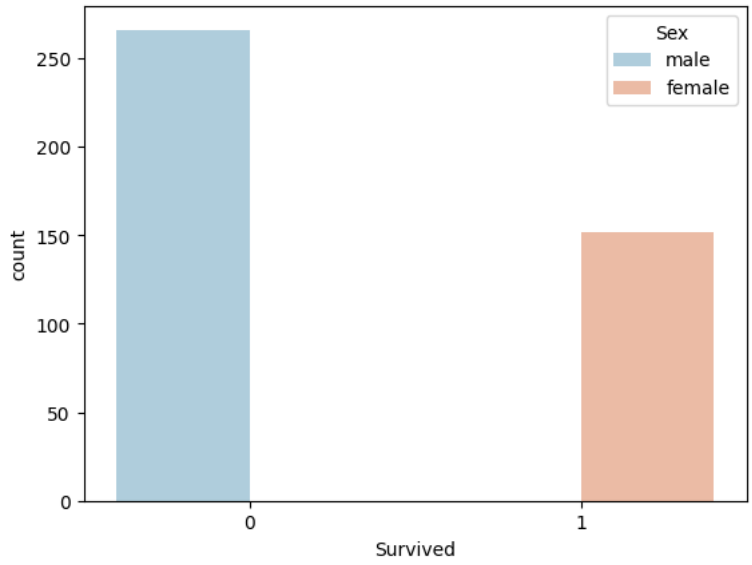
```
sns.countplot(x='Survived',data=tt)
```

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



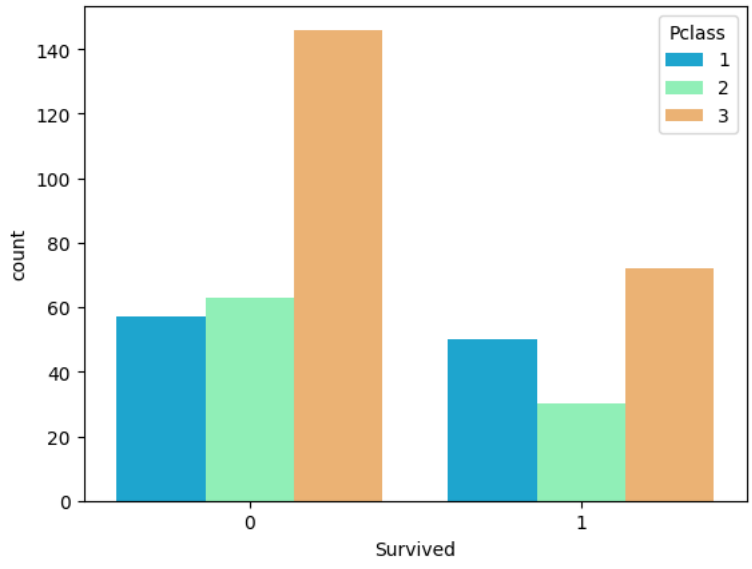

```
sns.countplot(x='Survived',hue='Sex',data=tt,palette='RdBu_r')
```

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



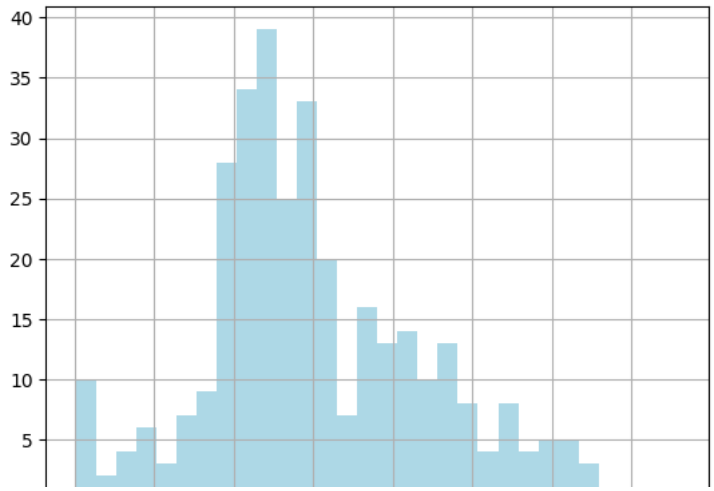

```
sns.countplot(x='Survived',hue='Pclass',data=tt,palette='rainbow')
```

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



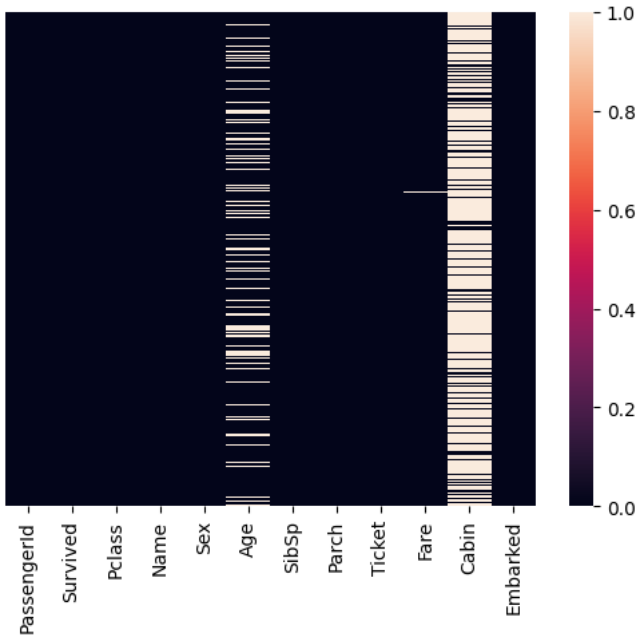

```
tt['Age'].hist(bins=30,color='lightblue')
```

```
<Axes: >
```

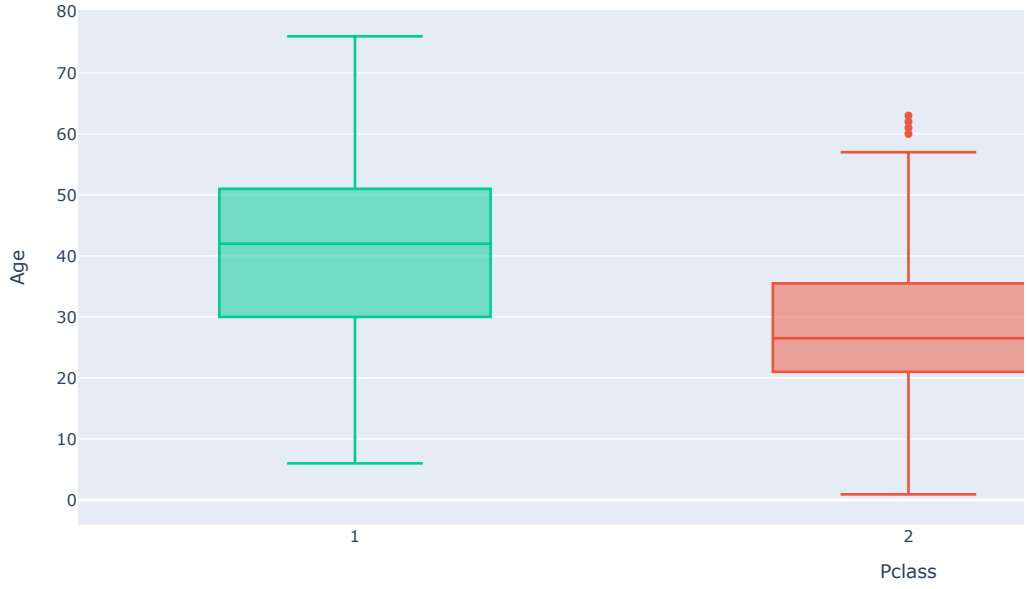



```
sns.heatmap(tt.isnull(),yticklabels=False)
```

```
<Axes: >
```




```
px.box(tt,x='Pclass',y='Age',color='Pclass')
```



```
# in the box plot age is factor of Pclass so we fill null values with meadian
tt.loc[(tt['Pclass'] == 1) & (tt['Age'].isnull()), 'Age'] = 42
tt.loc[(tt['Pclass'] == 2) & (tt['Age'].isnull()), 'Age'] = 26
tt.loc[(tt['Pclass'] == 3) & (tt['Age'].isnull()), 'Age'] = 24
```

```
tt=tt.drop(columns='Cabin')
```

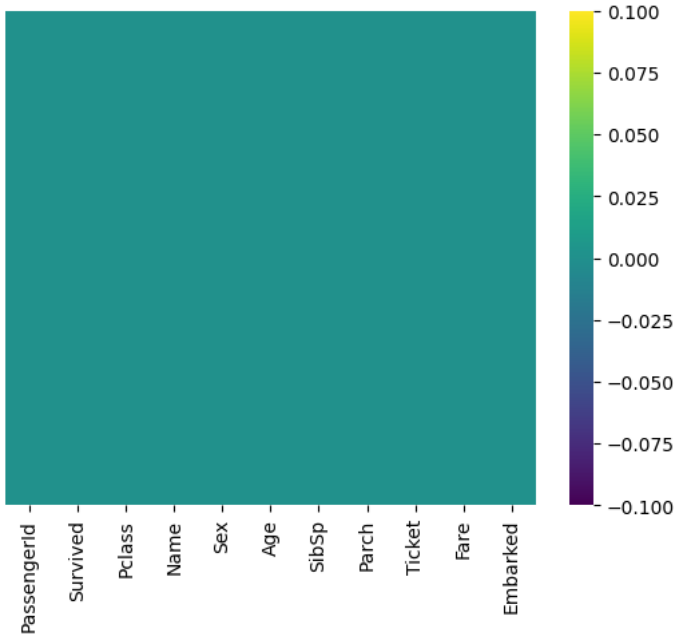
```
tt= tt.dropna()
```

```
tt.head()
```

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

```
sns.heatmap(tt.isnull(),yticklabels=False,cmap='viridis')
```

```
<Axes: >
```




```
tt['Age'] = tt['Age'].astype(int)
tt['Fare'] = tt['Fare'].astype(int)
```

```
# asign values to object using dictionary
tt['Embarked'] = tt['Embarked'].map({'Q': 0,'S':1,'C':2}).astype(int)
tt['Sex'] = tt['Sex'].map( {'female': 1,'male':0}).astype(int)
```

```
#drop columns for using dataset
ttn = tt.drop(['PassengerId','Name','Ticket'],axis = 1, inplace= True)
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
```

```
tt.head()
```

```
Survived  Pclass  Sex  Age  SibSp  Parch  Fare  Embarked
```

```
x= tt.drop(['Survived'],axis=1)
y= tt['Survived']
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=40)
clf = DecisionTreeClassifier()
clf.fit(x_train,y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
from sklearn.metrics import accuracy_score
#prediction on test data
y_pred = clf.predict(x_test)
#calculation
acc = accuracy_score(y_test,y_pred)
print('Accuracy:', acc)
```

```
Accuracy: 1.0
```