<u>**ADS Assignment 2**</u>

NAME: Aaron Paul.B

REG.NO: 20BCR7109

Titanic Ship Case Study

Problem Description: On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. Translated 32% survival rate.

- One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew.
- Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class.

The problem associated with the Titanic dataset is to predict whether a passenger survived the disaster or not. The dataset contains various features such as passenger class, age, gender, cabin, fare, and whether the passenger had any siblings or spouses on board. These features can be used to build a predictive model to determine the likelihood of a passenger surviving the disaster. The dataset offers opportunities for feature engineering, data visualization, and model selection, making it a valuable resource for developing and testing data analysis and machine learning skills.

Perform Below Tasks to complete the assignment:-

1. Download the dataset: Dataset

2. Load the dataset.

3. Perform Below Visualizations.

    - Univariate Analysis
    - Bi - Variate Analysis
    - Multi - Variate Analysis

4. Perform descriptive statistics on the dataset.

5. Handle the Missing values.

6. Find the outliers and replace the outliers

7. Check for Categorical columns and perform encoding.

8. Split the data into dependent and independent variables.

9. Scale the independent variables

10. Split the data into training and testing

1)

```
#1
# Download the dataset: Dataset


import numpy as np  import pandas as pd  import seaborn as sns
```

2)

```
#2
# Load the dataset.

from google.colab import files
uploaded = files.upload()
```

Choose Files titanic.csv

- **titanic.csv**(text/csv) - 57018 bytes, last modified: 5/28/2023 - 100% done

```
Saving titanic.csv to titanic.csv
```

```
df = pd.read_csv("titanic.csv")
df.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

```
[5]
df.info()
```

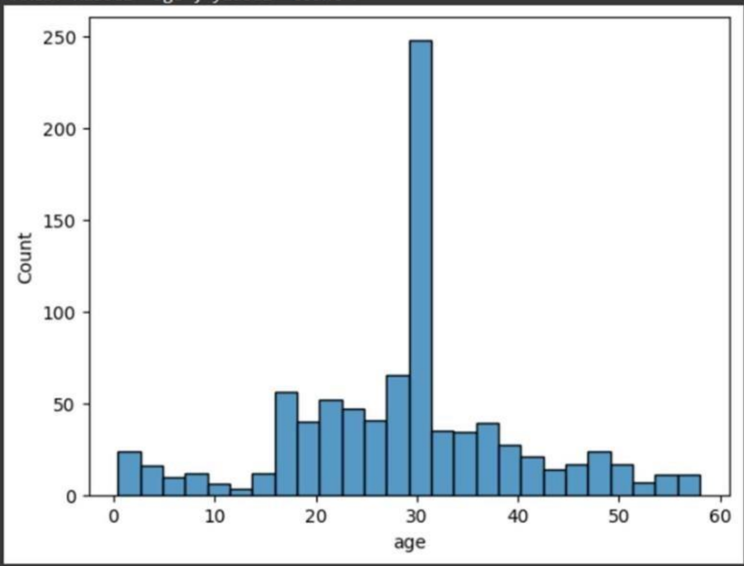
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    object
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    object
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), float64(2), int64(4), object(7)
memory usage: 92.4+ KB
```

3)

- Univariate Analysis

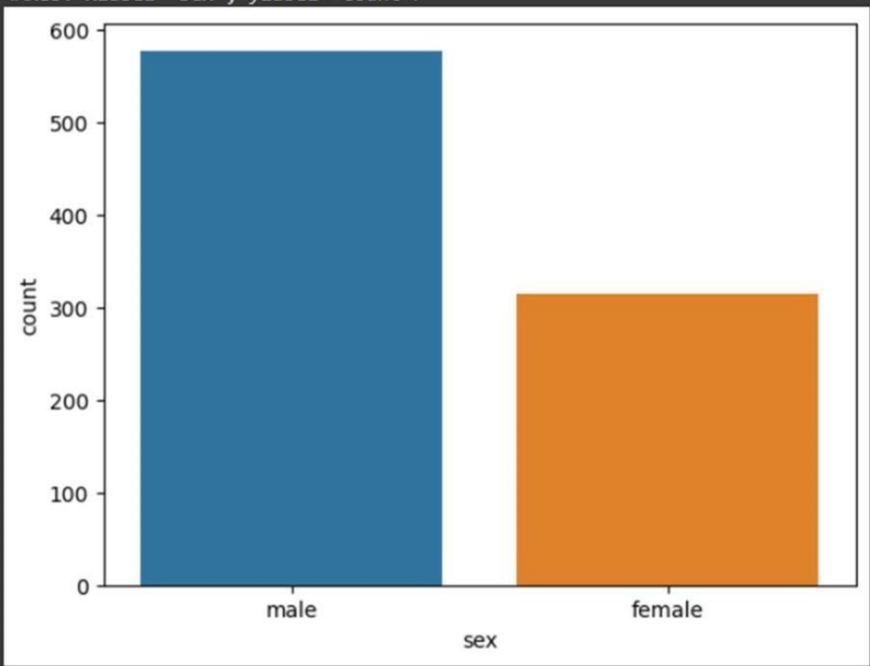
```python
sns.histplot(df['age'])
```

```
<Axes: xlabel='age', ylabel='Count'>
```



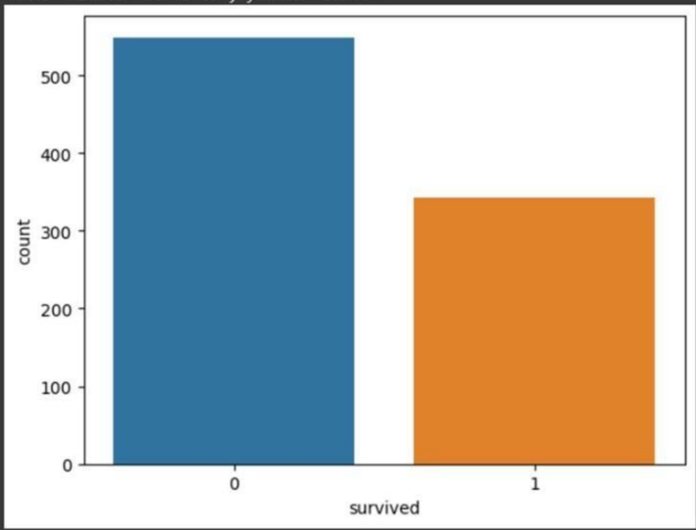

```python
sns.countplot(x = df['sex'])
```

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



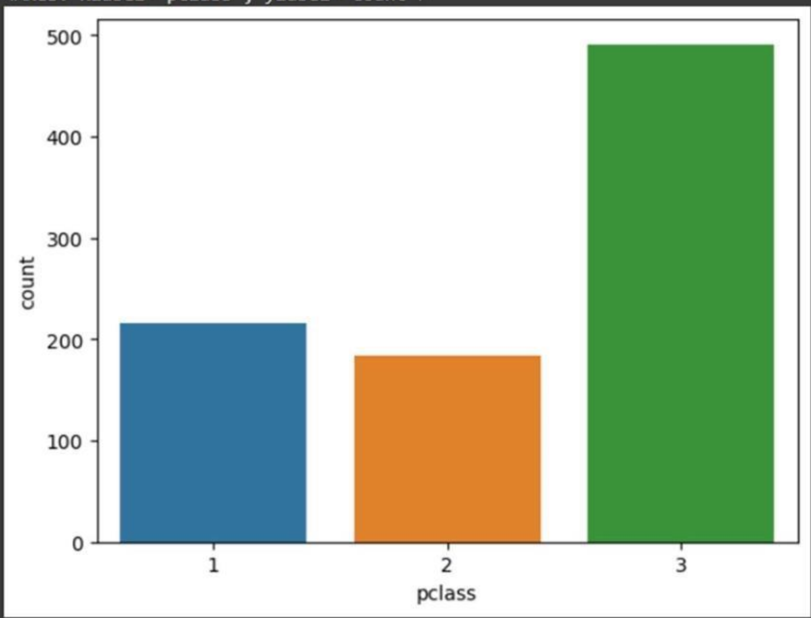
```
[8] sns.countplot(x = df['survived'])
```

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



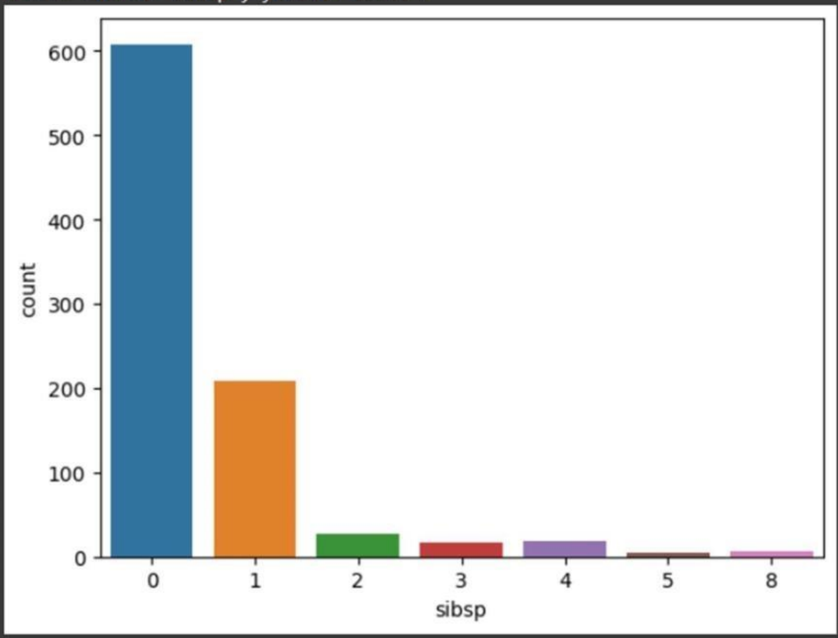

```
sns.countplot(x = df['pclass'])
```

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



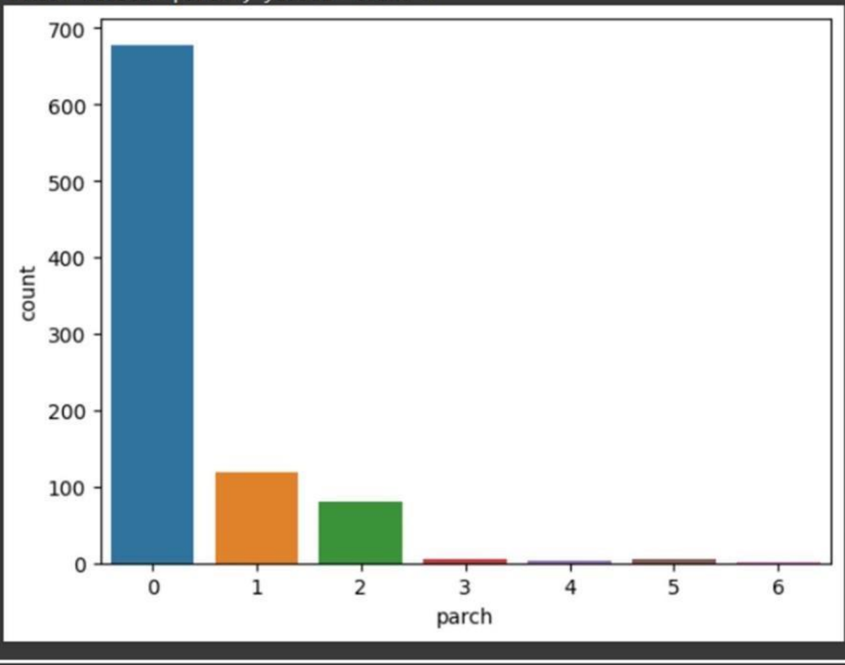
```
sns.countplot(x = df['sibsp'])
```

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



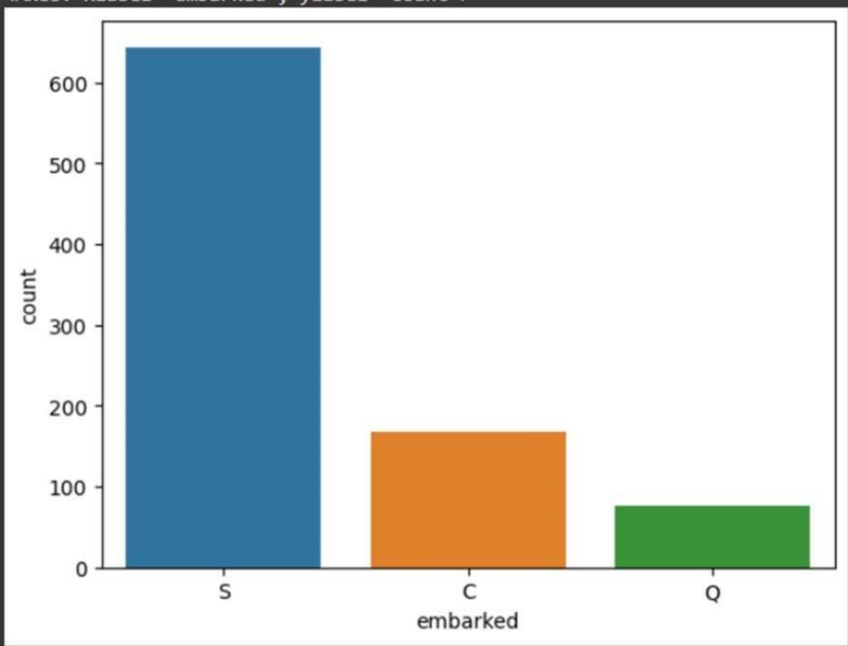
```
[11] sns.countplot(x = df['parch'])
```

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




```
sns.countplot(x = df['embarked'])
```

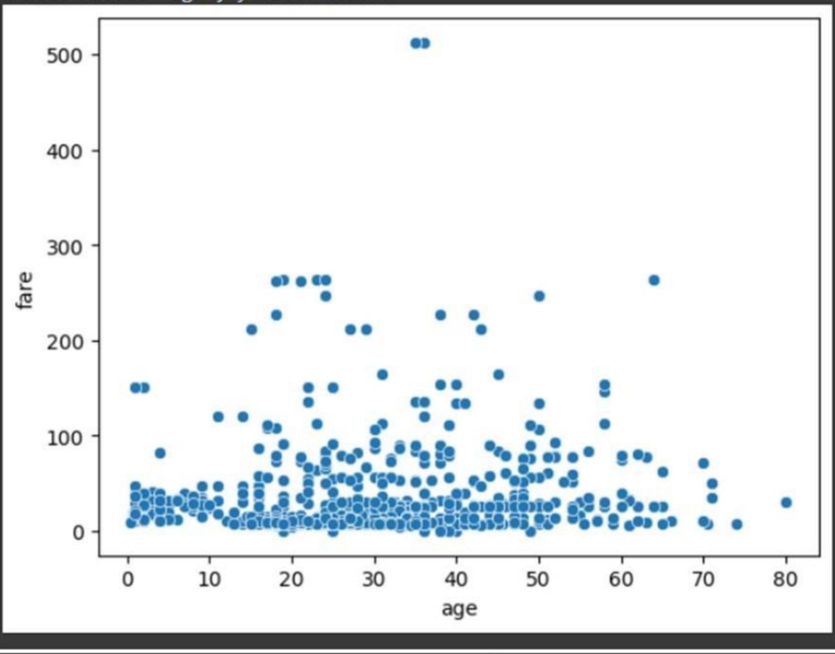
```
<Axes: xlabel='embarked', ylabel='count'>
```



- bivariate analysis

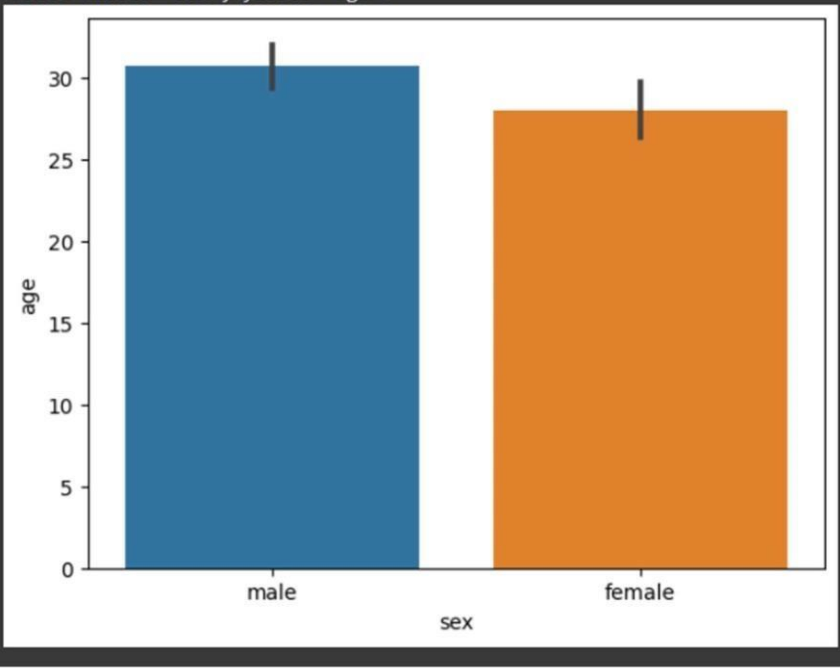
```
sns.scatterplot(data = df, x = 'age', y = 'fare')
```

```
<Axes: xlabel='age', ylabel='fare'>
```



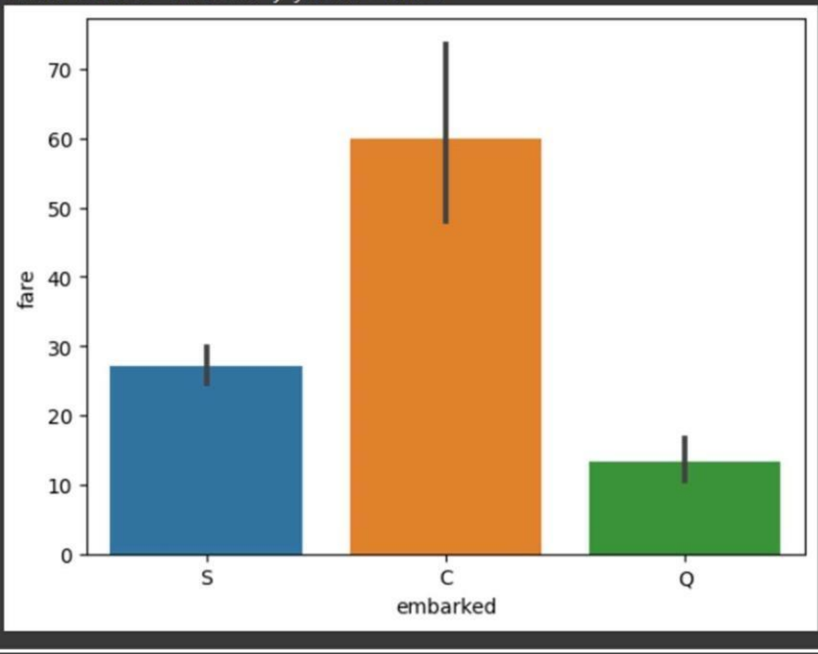

```
sns.barplot(data = df, x = 'sex', y = 'age')
```

```
<Axes: xlabel='sex', ylabel='age'>
```



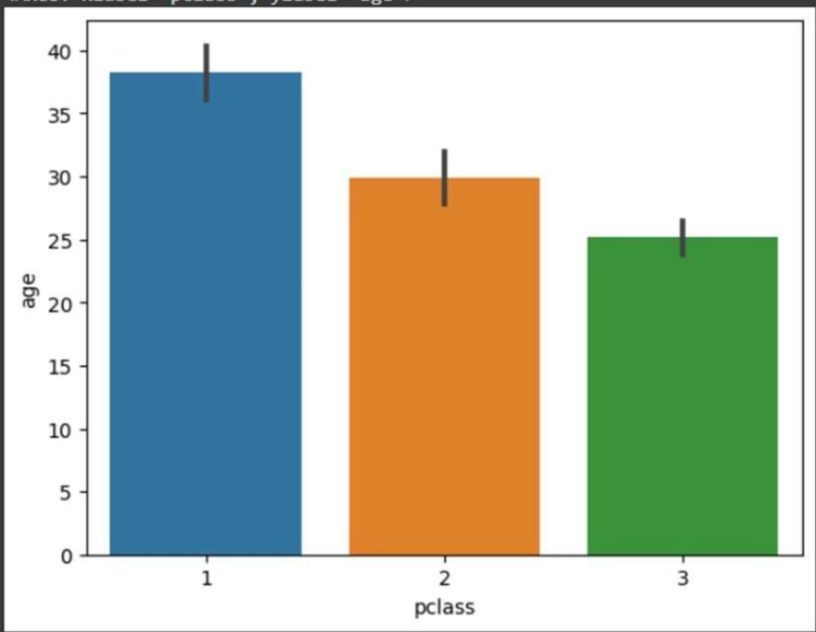
```python
sns.barplot(data = df, x = 'embarked', y = 'fare')
```

```
<Axes: xlabel='embarked', ylabel='fare'>
```



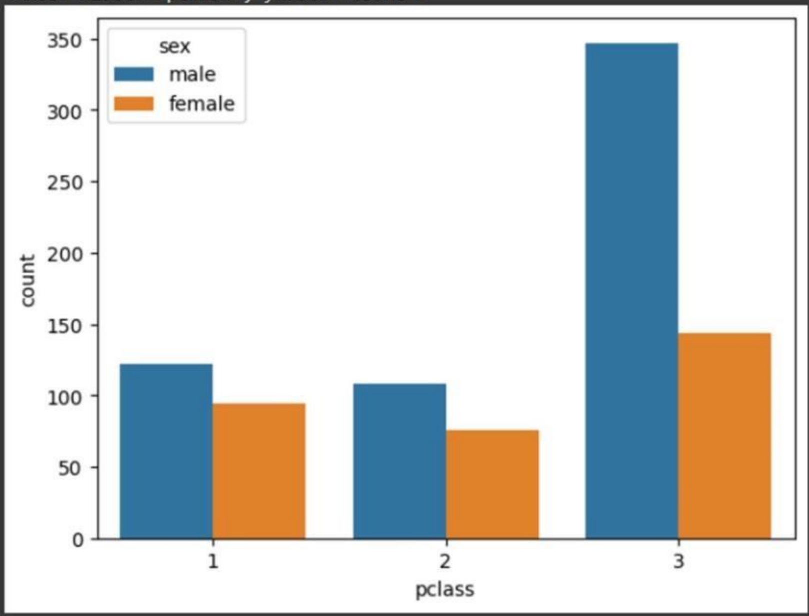

```python
sns.barplot(data = df, x = 'pclass', y = 'age')
```

```
<Axes: xlabel='pclass', ylabel='age'>
```



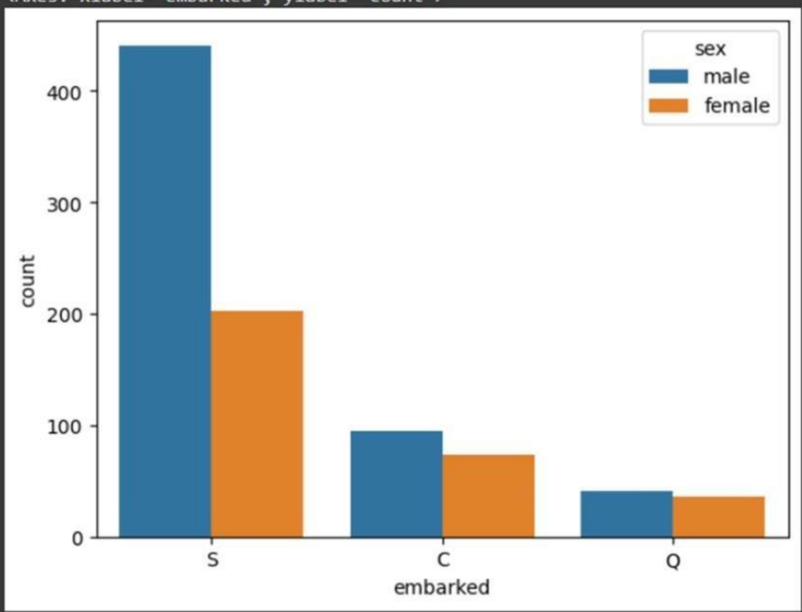
```
[17] sns.countplot(x = df['pclass'], hue = df['sex'])
```

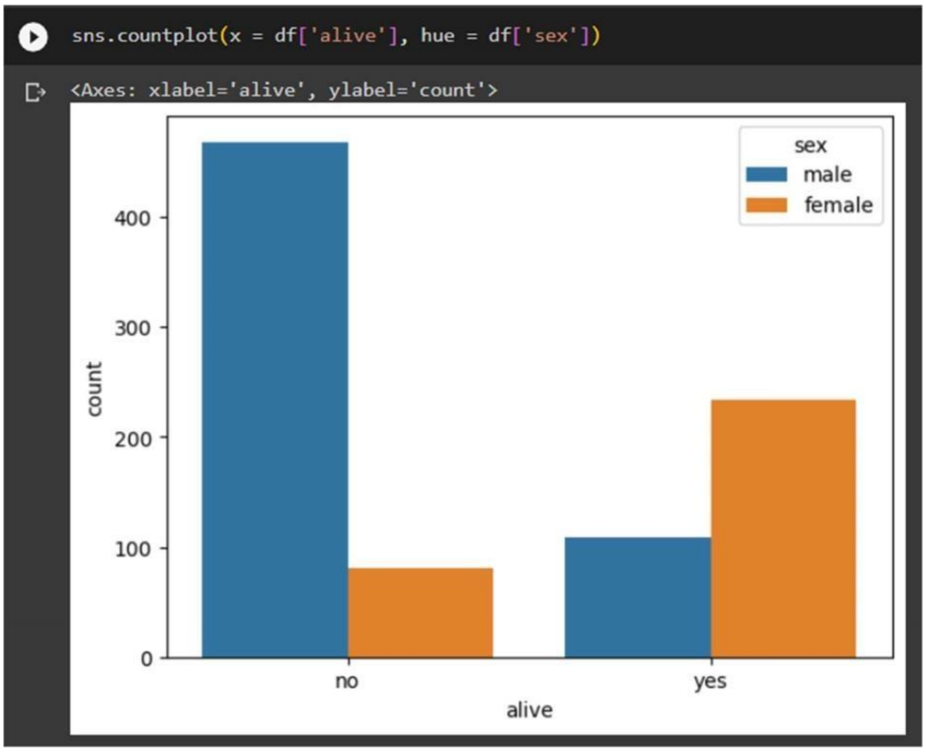
```
<Axes: xlabel='pclass', ylabel='count'>
```



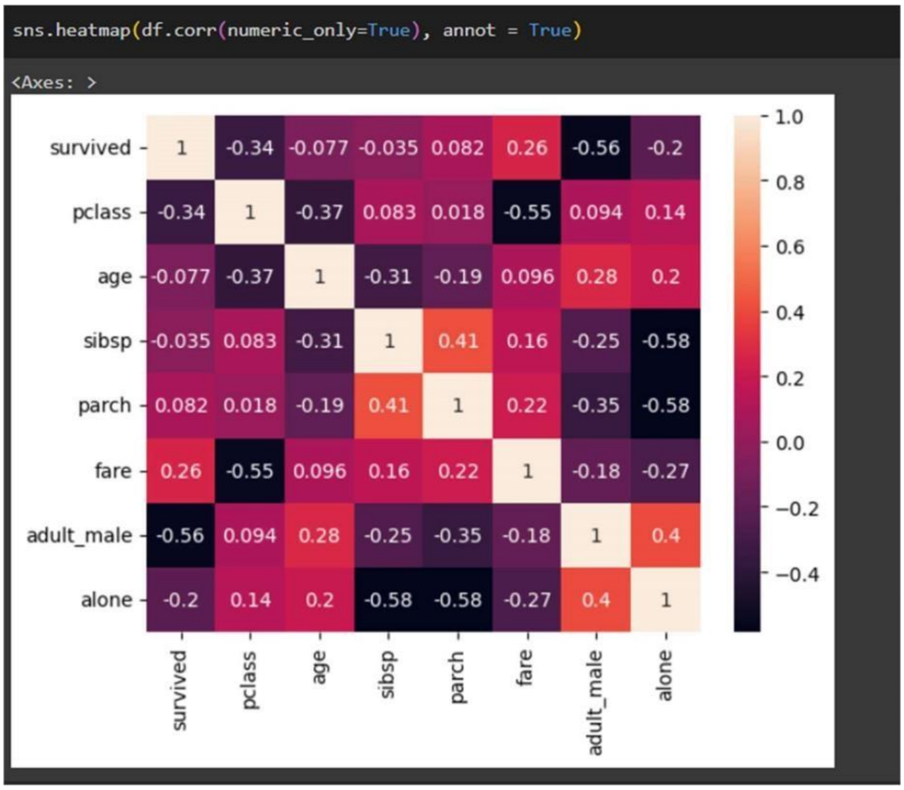

```
sns.countplot(x = df['embarked'], hue = df['sex'])
```

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





- mul variate analysis



4)

```
df.describe()
```

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

5)

```
df.isnull().sum()
```

```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

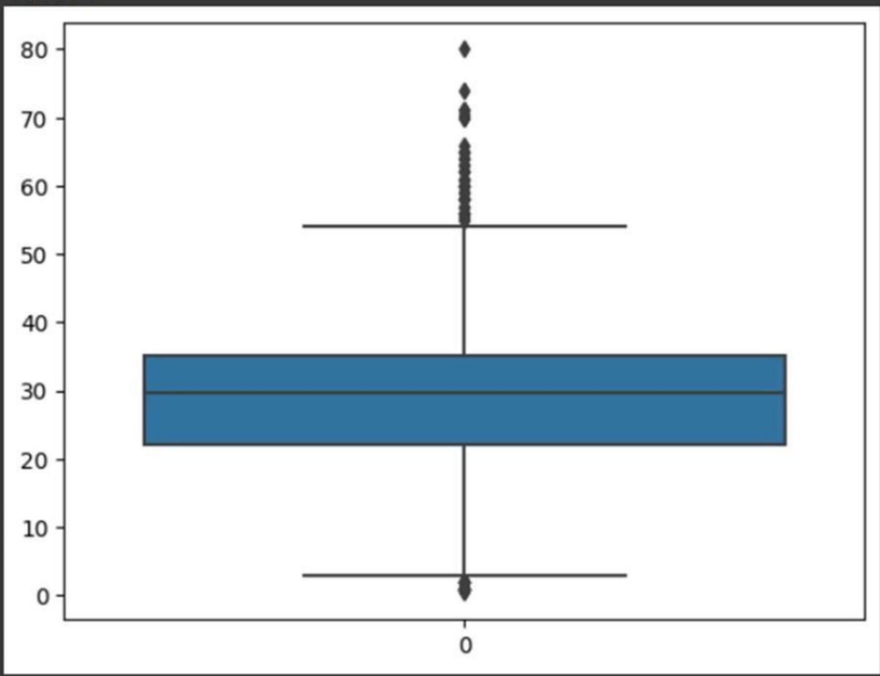
```
df.dropna(subset=['embark_town'], how='all', inplace = True)
df['age'] = df['age'].fillna(df['age'].mean())
df.drop(['deck'], axis = 1,inplace = True)
df.isnull().sum()
```

```
survived       0
pclass         0
sex            0
age            0
sibsp          0
parch          0
fare           0
embarked       0
class          0
who            0
adult_male     0
embark_town    0
alive          0
alone          0
dtype: int64
```

6)

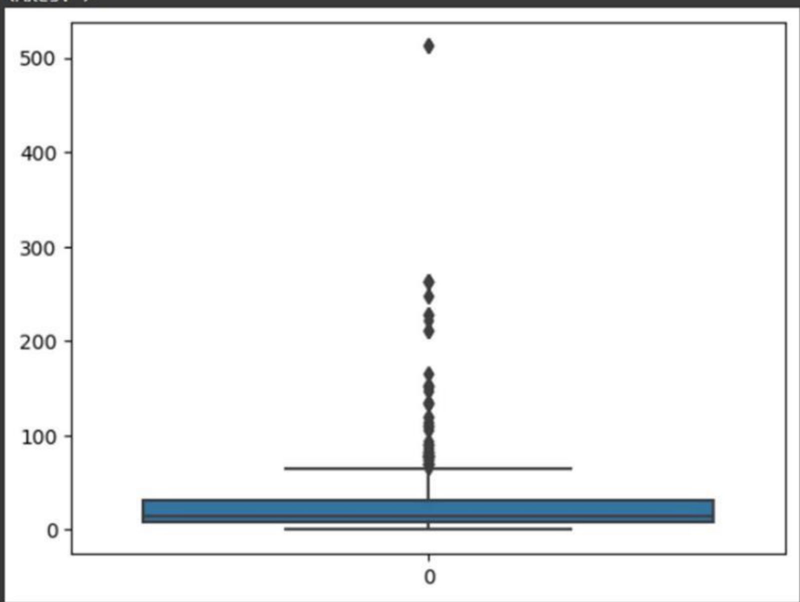
```python
sns.boxplot(df['age'])
```

```
<Axes: >
```



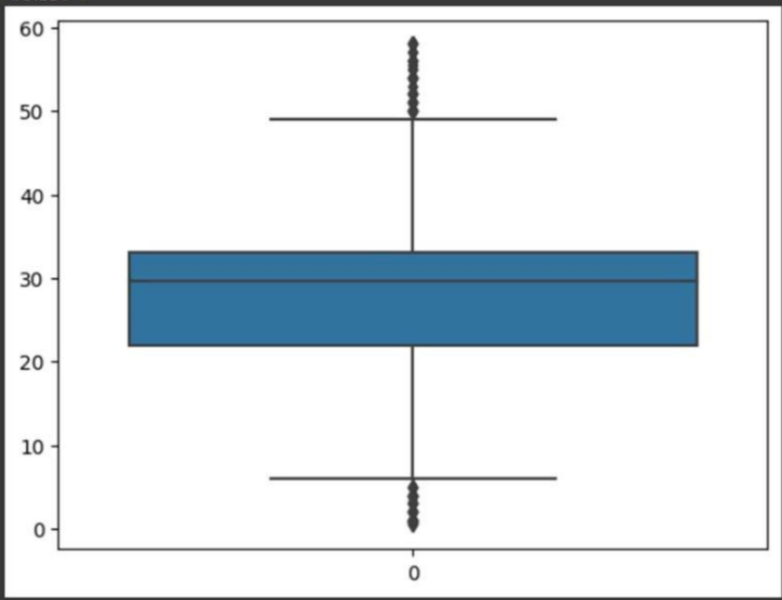

```python
[26] sns.boxplot(df['fare'])
```

```
<Axes: >
```



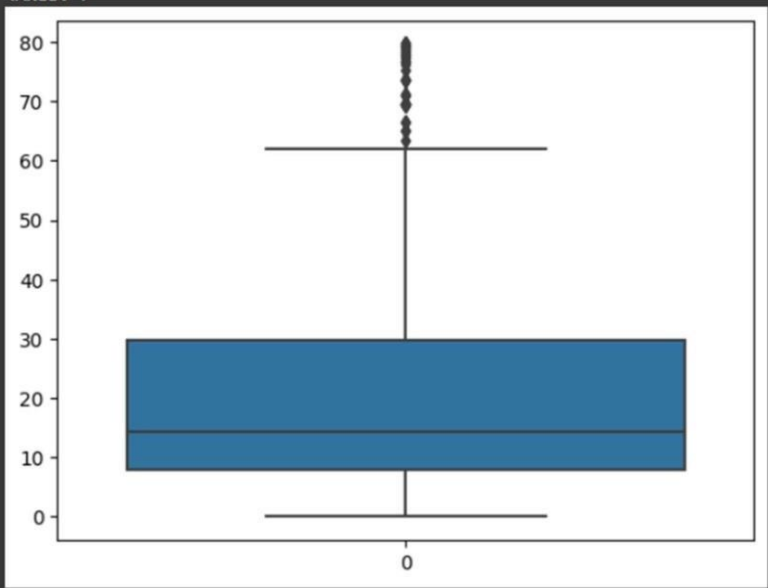
```python
median_age = df['age'].median()
df["age"] = np.where(df["age"] > 58, median_age, df['age'])
sns.boxplot(df['age'])
```

```
<Axes: >
```




```python
median_fare = df['fare'].median()
df["fare"] = np.where(df["fare"] > 80, median_age, df['fare'])
sns.boxplot(df['fare'])
```

```
<Axes: >
```



7)

```
#7
# Check for Categorical columns and perform encoding.

from sklearn.preprocessing import OneHotEncoder
encoding = pd.get_dummies(df, columns = ['sex','embarked','class','who','adult_male','embark_town','alone'])
encoding.head()
```

	survived	pclass	age	sibsp	parch	fare	alive	sex_female	sex_male	embarked_C	...	who_child	who_man	who_woman	ad
0	0	3	22.0	1	0	7.2500	no	0	1	0	...	0	1	0	
1	1	1	38.0	1	0	71.2833	yes	1	0	1	...	0	0	1	
2	1	3	26.0	0	0	7.9250	yes	1	0	0	...	0	0	1	
3	1	1	35.0	1	0	53.1000	yes	1	0	0	...	0	0	1	
4	0	3	35.0	0	0	8.0500	no	0	1	0	...	0	1	0	

5 rows × 25 columns

8)

```
df.columns
```

```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'embark_town', 'alive',
       'alone'],
      dtype='object')
```

```
X = encoding.drop(['survived', 'alive'], axis = 1)
X.head()
```

	pclass	age	sibsp	parch	fare	sex_female	sex_male	embarked_C	embarked_Q	embarked_S	...	who_child	who_
0	3	22.0	1	0	7.2500	0	1	0	0	1	...	0	
1	1	38.0	1	0	71.2833	1	0	1	0	0	...	0	
2	3	26.0	0	0	7.9250	1	0	0	0	1	...	0	
3	1	35.0	1	0	53.1000	1	0	0	0	1	...	0	
4	3	35.0	0	0	8.0500	0	1	0	0	1	...	0	

5 rows × 23 columns

```
y = df[['survived', 'alive']]
```

Run cell (Ctrl+Enter)
cell has not been executed in this session

executed by Pratham Mohanty
16:30 (1 hour ago)
executed in 0.499 s

1	1	yes
2	1	yes
3	1	yes
4	0	no

9)

```
[ ] #9
    # Scale the independent variables


    from sklearn.preprocessing import StandardScaler
    scaler = StandardScaler()
    x_std = scaler.fit_transform(X)
    x_std

    array([[ 0.82520863, -0.57985934,  0.43135024, ...,  0.61679395,
             1.22934919, -1.22934919],
           [-1.57221121,  0.83108889,  0.43135024, ..., -1.62128697,
             1.22934919, -1.22934919],
           [ 0.82520863, -0.22712228, -0.47519908, ...,  0.61679395,
            -0.81343853,  0.81343853],
           ...,
           [ 0.82520863,  0.09405298,  0.43135024, ...,  0.61679395,
             1.22934919, -1.22934919],
           [-1.57221121, -0.22712228, -0.47519908, ..., -1.62128697,
            -0.81343853,  0.81343853],
           [ 0.82520863,  0.3019833 , -0.47519908, ..., -1.62128697,
            -0.81343853,  0.81343853]])
```

10)

```
[ ] # 10
    #  Split the data into training and testing

    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, y['survived'], test_size=0.33)

[ ]
```