TASK-2

DATA CLEANING and PERFORMING EDA

In [3]:

```
#Load the required libraries
import pandas as pd
import numpy as np
import seaborn as sns
#Load the data
df = pd.read_csv('D:\\mlworld\\train.csv')
#View the data
df.head()
df
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Er
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
				Futrelle, Mrs. .								

In [14]:

```
df.shape
```

Out[14]:

```
(891, 12)
```

In [15]:

```
df.duplicated().sum()
```

Out[15]:

```
0
```

In [16]:

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

Out[16]:

```
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
```

HANDILING MISSING VALUES

replacing missing values

In [23]:

```
#Replace null values
df.replace(np.nan,'0',inplace = True)
#Check the changes now
df.isnull().sum()
```

Out[23]:

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

Summary Statistics:

In [34]:

```
#Describe the data
df.describe()
```

Out[34]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [35]:

```
#Basic information
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 [36]:

```
#unique values
df['Sex'].unique()
```

Out[36]:

```
array(['male', 'female'], dtype=object)
```

In [40]:

```
#unique values
df['Survived'].unique()
```

Out[40]:

```
array([0, 1], dtype=int64)
```

In [44]:

```
data['Fare'] = data['Fare'].astype(str)

# Select only categorical columns
categorical_columns = data.select_dtypes(include=["object"]).columns

# Calculate the frequency of each category for each categorical column
categorical_frequencies = {}
for column in categorical_columns:
    column_frequencies = data[column].value_counts()
    categorical_frequencies[column] = column_frequencies

# Display the results
for column, frequencies in categorical_frequencies.items():
    print(f"Frequency of Categories for '{column}':")
    print(frequencies)
    print("\n")
```

```
Frequency of Categories for 'Name':
Braund, Mr. Owen Harris                     1
Boulos, Mr. Hanna                           1
Frolicher-Stehli, Mr. Maxmillian            1
Gilinski, Mr. Eliezer                       1
Murdlin, Mr. Joseph                         1
                                           ..
Kelly, Miss. Anna Katherine "Annie Kate"    1
McCoy, Mr. Bernard                          1
Johnson, Mr. William Cahoone Jr             1
Keane, Miss. Nora A                         1
Dooley, Mr. Patrick                         1
Name: Name, Length: 891, dtype: int64


Frequency of Categories for 'Sex':
male      577
female    314
Name: Sex, dtype: int64


Frequency of Categories for 'Age':
0       177
24.0     30
22.0     27
18.0     26
28.0     25
       ...
36.5      1
55.5      1
0.92      1
23.5      1
74.0      1
Name: Age, Length: 89, dtype: int64


Frequency of Categories for 'Ticket':
347082      7
CA. 2343    7
1601        7
3101295     6
CA 2144     6
           ..
9234        1
19988       1
2693        1
PC 17612    1
370376      1
Name: Ticket, Length: 681, dtype: int64


Frequency of Categories for 'Fare':
8.05       43
13.0       42
7.8958     38
7.75       34
26.0       31
           ..
35.0        1
28.5        1
6.2375      1
```

```
14.0        1
10.5167     1
Name: Fare, Length: 248, dtype: int64


Frequency of Categories for 'Cabin':
0              687
C23 C25 C27      4
G6               4
B96 B98          4
C22 C26          3
              ...
E34              1
C7               1
C54              1
E36              1
C148             1
Name: Cabin, Length: 148, dtype: int64


Frequency of Categories for 'Embarked':
S    644
C    168
Q     77
0      2
Name: Embarked, dtype: int64
```

Data Visualization:

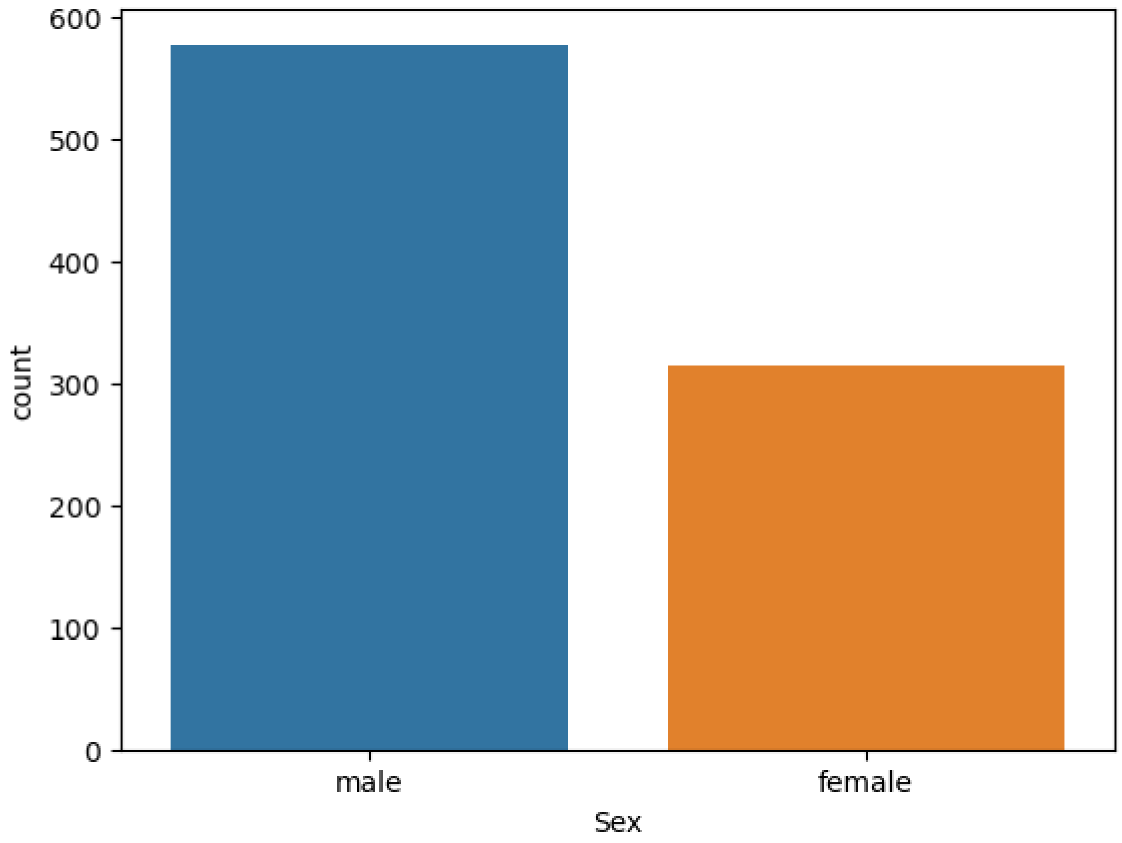
In [48]:

```
#Plot the unique values
sns.countplot(df['Sex']).unique()
```

```
C:\Users\91939\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Futu
reWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterp
retation.
  warnings.warn(

--------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_23040\884569769.py in <module>
      1 #Plot the unique values
----> 2 sns.countplot(df['Sex']).unique()

AttributeError: 'AxesSubplot' object has no attribute 'unique'
```

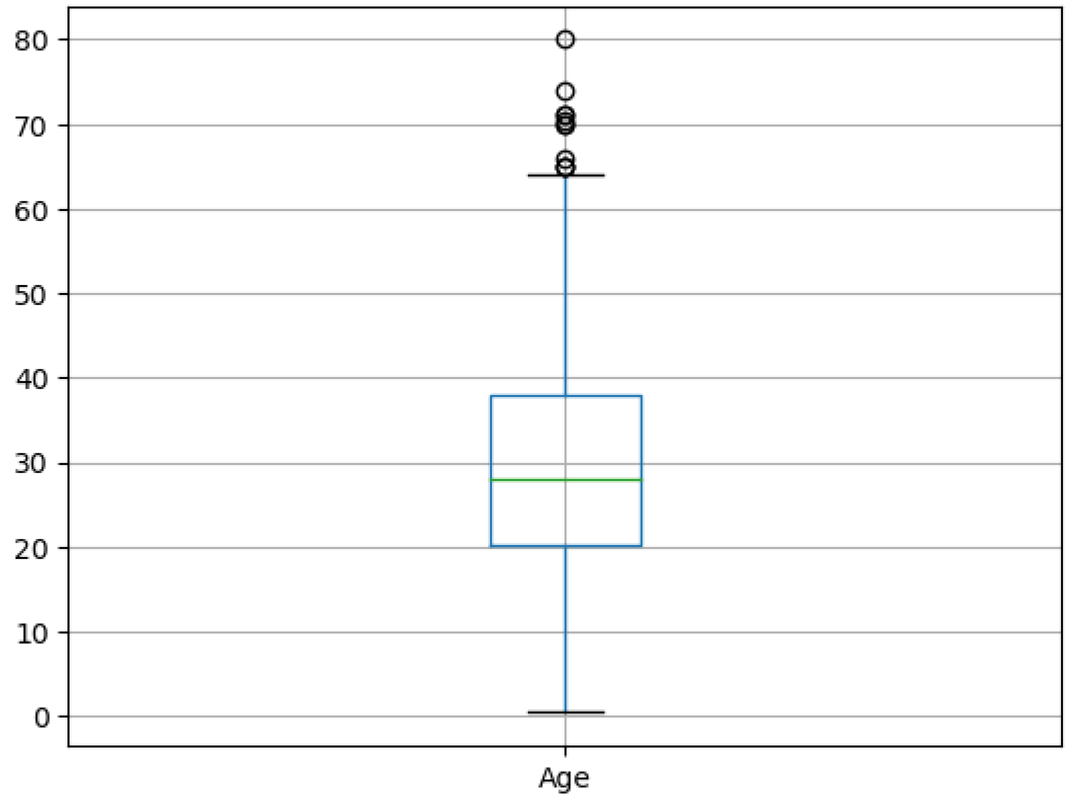


In [49]:

```
#Boxplot
df[['Age']].boxplot()
```

Out[49]:

```
<AxesSubplot:>
```



Correlation Analysis

In [50]:

```
df.corr()
```

Out[50]:

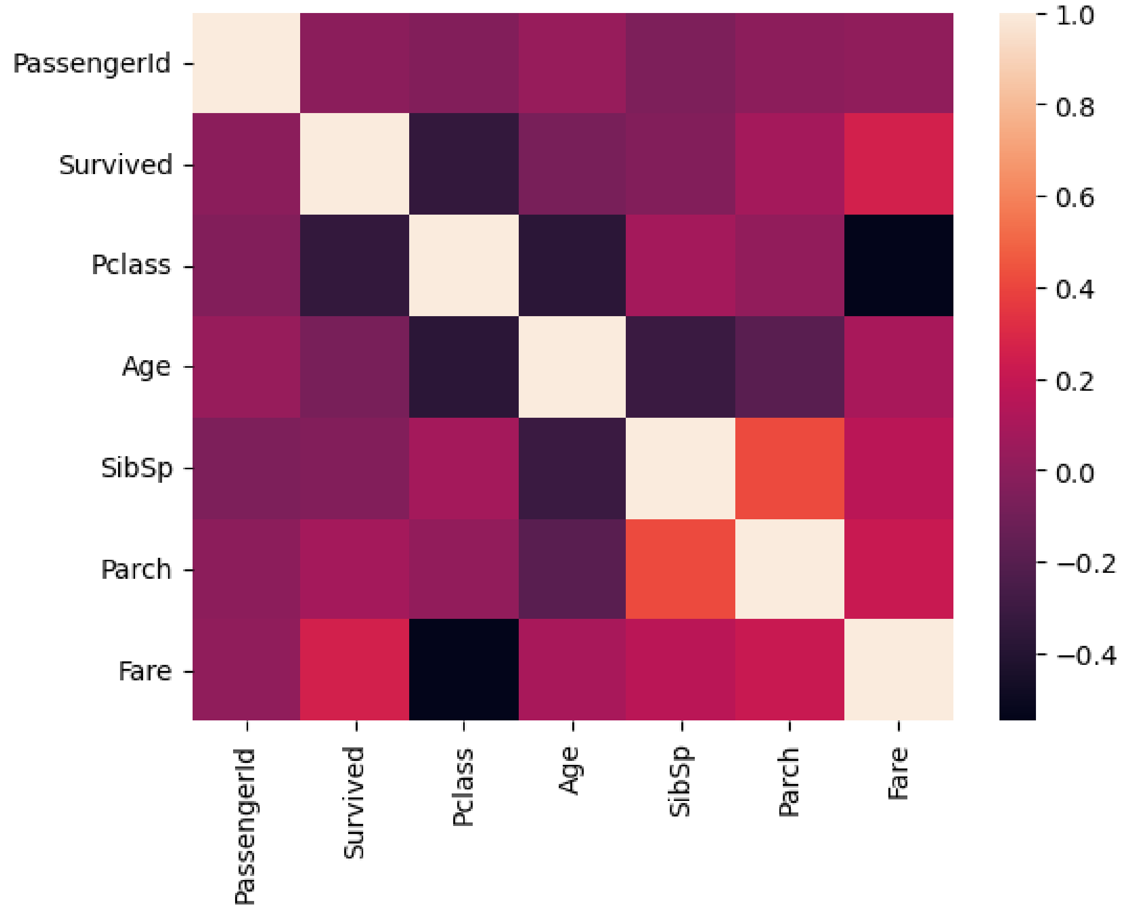
	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000

In [51]:

```
#Correlation plot
sns.heatmap(df.corr())
```

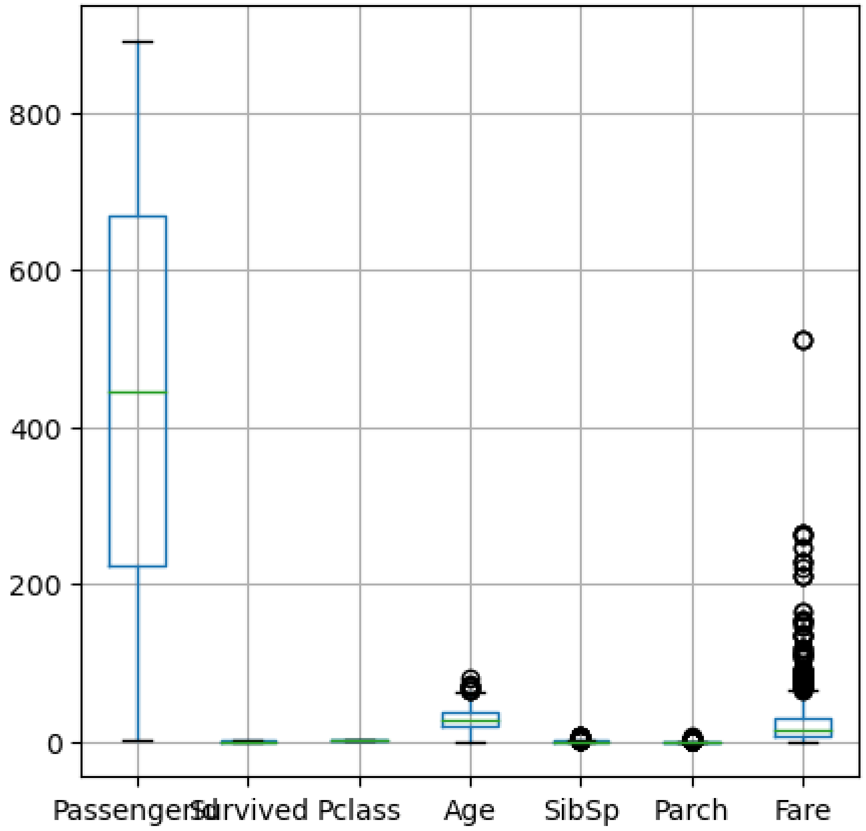
Out[51]:

```
<AxesSubplot:>
```



In [52]:

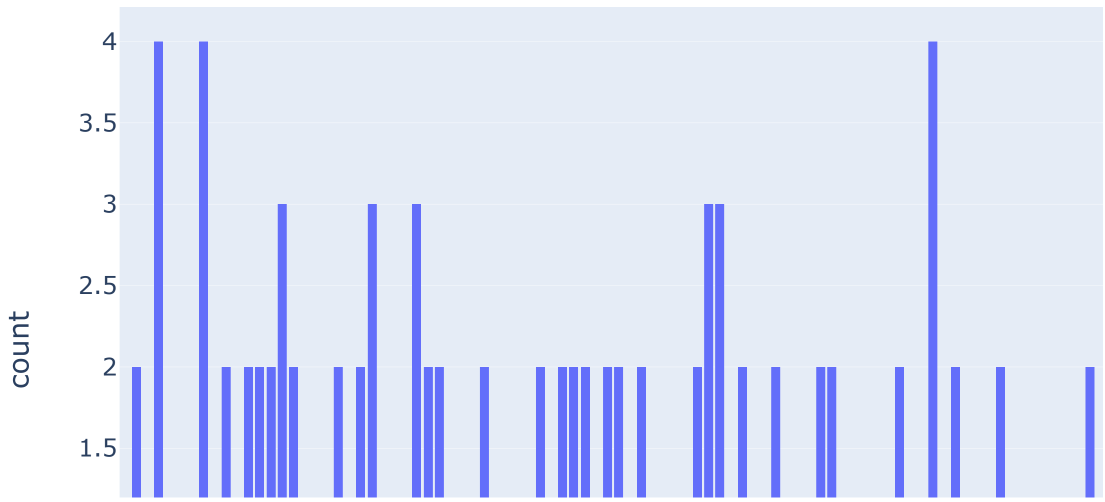
```
#add csv file to dataframe
df = pd.read_csv('D:\\mlworld\\train.csv')
#create boxplot
boxplot = df.boxplot(figsize = (5,5))
```



Visualize Outliers

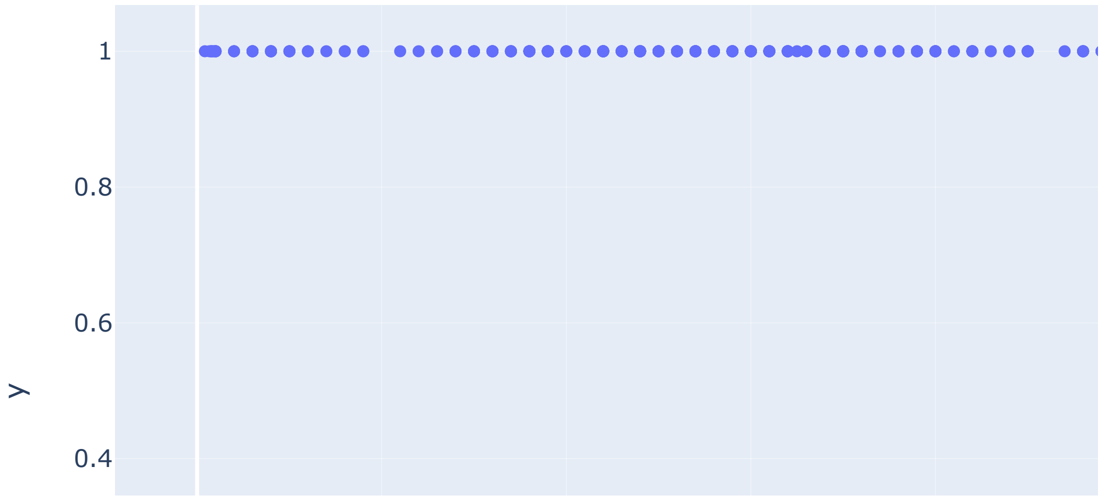
In [53]:

```
import plotly.express as px
fig = px.histogram(df, x='Cabin')
fig.show()
```



In [54]:

```
fig = px.scatter(x=df['Age'], y=df['Survived'])
fig.show()
```



In [55]:

```
def find_outliers_IQR(df):
 q1=df.quantile(0.25)
 q3=df.quantile(0.75)
 IQR=q3-q1
 outliers = df[((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]
 return outliers
outliers = find_outliers_IQR(df['Fare'])
print('number of outliers: '+ str(len(outliers)))
print('max outlier value: '+ str(outliers.max()))
print('min outlier value: '+ str(outliers.min()))
outliers
```

```
number of outliers: 116
max outlier value: 512.3292
min outlier value: 66.6
```

Out[55]:

```
1       71.2833
27     263.0000
31     146.5208
34      82.1708
52      76.7292
         ...   
846     69.5500
849     89.1042
856    164.8667
863     69.5500
879     83.1583
Name: Fare, Length: 116, dtype: float64
```

Handiling Outliers - (CAPPING THE OUTLIERS)

In [57]:

```
upper_limit = df['Fare'].mean() + 3*df['Fare'].std()
print(upper_limit)
lower_limit = df['Fare'].mean() - 3*df['Fare'].std()
print(lower_limit)
```

```
181.2844937601173
-116.87607782296811
```

In [ ]: