Name : GAGAN SAI

Register Number : 20MID0192 VIT - Vellore Titanic Ship Case Study: 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. Importing all the Libraries Required:

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import skew
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [2]:
```
# 2. Load the dataset
Titanic = pd.read_csv('C:/Users/gagan/Downloads/titanic.csv')
Titanic
```

Out[2]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	d
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	N
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	N
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	N
...	...	...	...	...	...	...	...	...	...	...	...	
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	N
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	N
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	N

891 rows × 15 columns

In [3]:

```
Titanic.head(15)
```

Out[3]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	de
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	Na
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	Na
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	Na
5	0	3	male	NaN	0	0	8.4583	Q	Third	man	True	Na
6	0	1	male	54.0	0	0	51.8625	S	First	man	True	
7	0	3	male	2.0	3	1	21.0750	S	Third	child	False	Na
8	1	3	female	27.0	0	2	11.1333	S	Third	woman	False	Na
9	1	2	female	14.0	1	0	30.0708	C	Second	child	False	Na
10	1	3	female	4.0	1	1	16.7000	S	Third	child	False	
11	1	1	female	58.0	0	0	26.5500	S	First	woman	False	
12	0	3	male	20.0	0	0	8.0500	S	Third	man	True	Na
13	0	3	male	39.0	1	5	31.2750	S	Third	man	True	Na
14	0	3	female	14.0	0	0	7.8542	S	Third	child	False	Na

In [4]:

```
print("Column Names : ",", ".join(Titanic.columns.to_list()))
```

```
Column Names :  survived, pclass, sex, age, sibsp, parch, fare, embarked, class, who,
adult_male, deck, embark_town, alive, alone
```

In [5]:
```
int64_col = Titanic.select_dtypes(include = 'int64')
print("Integer Columns: ", int64_col.columns.to_list())
float64_col = Titanic.select_dtypes(include = 'float64')
print("Float Columns : ", float64_col.columns.to_list())
object_col = Titanic.select_dtypes(include = 'object')
print("Object Columns : ", object_col.columns.to_list())
numeric_col = Titanic.select_dtypes('number')
print("Numeric Columns : ", numeric_col.columns.to_list())
```

```
Integer Columns:  ['survived', 'pclass', 'sibsp', 'parch']
Float Columns :  ['age', 'fare']
Object Columns :  ['sex', 'embarked', 'class', 'who', 'deck', 'embark_town', 'alive']
Numeric Columns :  ['survived', 'pclass', 'age', 'sibsp', 'parch', 'fare']
```

In [6]:
```
Titanic.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
```

In [7]:
```
for column in object_col:
    value_counts = Titanic[column].value_counts(dropna = False).reset_index()
    value_counts.columns = ['Value', 'Count']
    print(f"Value counts for column '{column}':\n{value_counts.to_string(index=False)}
```

```
Value counts for column 'sex':
 Value  Count
  male    577
female    314

Value counts for column 'embarked':
Value  Count
    S    644
    C    168
    Q     77
  NaN      2

Value counts for column 'class':
 Value  Count
 Third    491
 First    216
Second    184

Value counts for column 'who':
Value  Count
  man    537
woman    271
child     83

Value counts for column 'deck':
Value  Count
  NaN    688
    C     59
    B     47
    D     33
    E     32
    A     15
    F     13
    G      4

Value counts for column 'embark_town':
      Value  Count
Southampton    644
  Cherbourg    168
 Queenstown     77
        NaN      2

Value counts for column 'alive':
Value  Count
   no    549
  yes    342
```

In [8]:

```
male_count = Titanic[(Titanic['sex'] == 'male')].shape[0]
print("Total Male Count : ", male_count)
male_adult_count = Titanic[(Titanic['sex'] == 'male') & (Titanic['adult_male'] == True
print("Total Adult Male Count : ", male_adult_count)
male_survived = Titanic[(Titanic['sex'] == 'male') & (Titanic['survived'] == 1)].shape
print("Total male survived : ", male_survived)
adult_male_survived = Titanic[(Titanic['sex'] == 'male') & (Titanic['adult_male'] == T
print("Adult Male Survived Count : ", adult_male_survived)
female_count = Titanic[(Titanic['sex'] == 'female')].shape[0]
print("Total Female Count : ", female_count)
female_survived = Titanic[(Titanic['sex'] == 'female') & (Titanic['survived'] == 1)].s
print("Total Female Survived : ", female_survived)
```

```
print("Survival Percentage :")
print("For Male : ", round((male_survived/male_count)*100),"%")
print("For Female : ", round((female_survived/female_count)*100), "%")
```

```
Total Male Count :  577
Total Adult Male Count :  537
Total male survived :  109
Adult Male Survived Count :  88
Total Female Count :  314
Total Female Survived :  233
Survival Percentage :
For Male :  19 %
For Female :  74 %
```

In [9]:

```
male_survived_pclass = Titanic[Titanic['sex'] == 'male'].groupby('pclass')['survived']
print("Number of male survivors by Pclass:")
print(male_survived_pclass)
```

```
Number of male survivors by Pclass:
pclass
1    45
2    17
3    47
Name: survived, dtype: int64
```

In [10]:

```
import seaborn as sns

print(sns.__version__)
```

```
0.12.2
```

3. Visualisation :

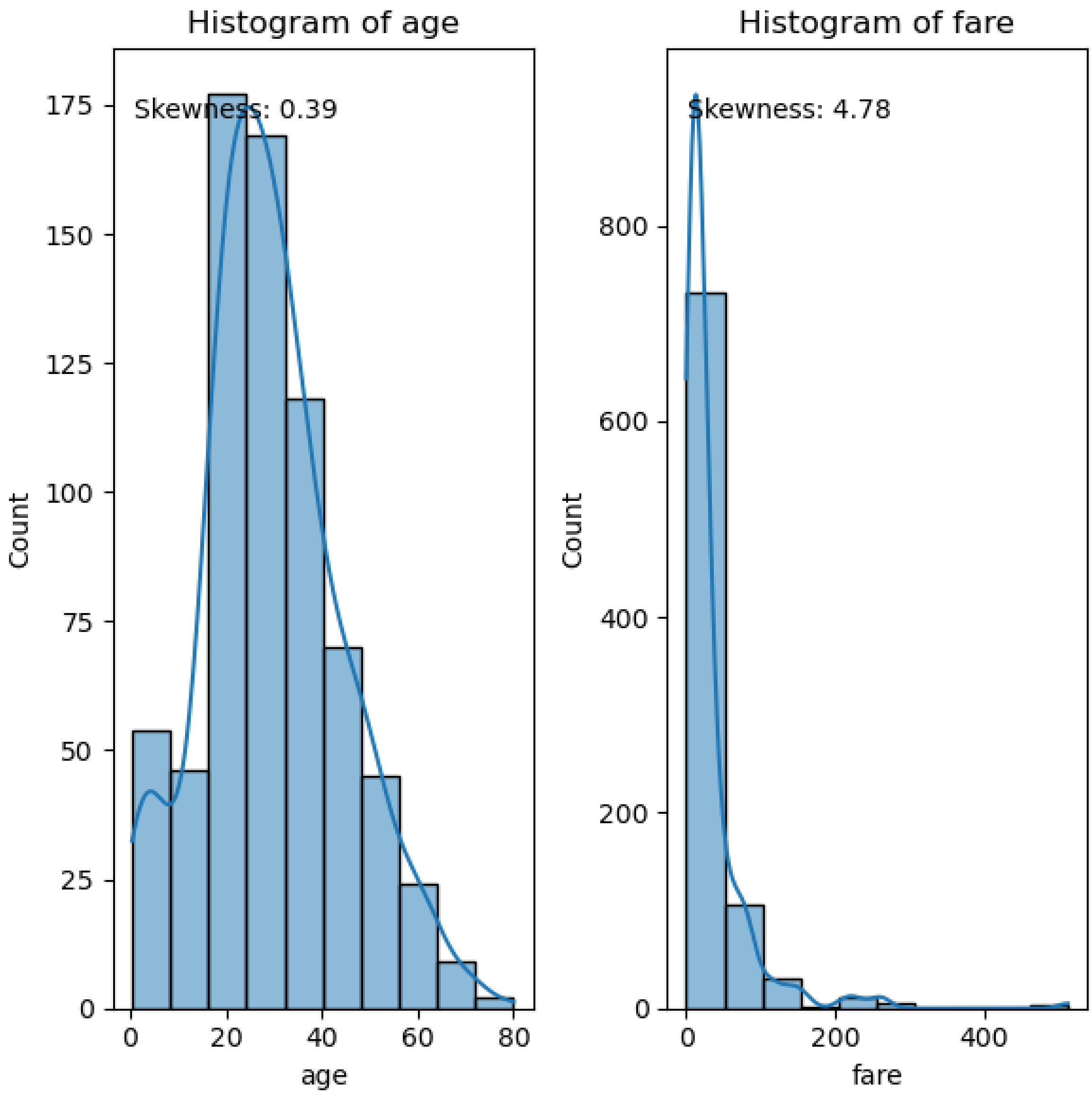
Univariate Analysis Bivariate Analysis Multivariate Analysis 3.1. Univariate Analysis:

In [11]:

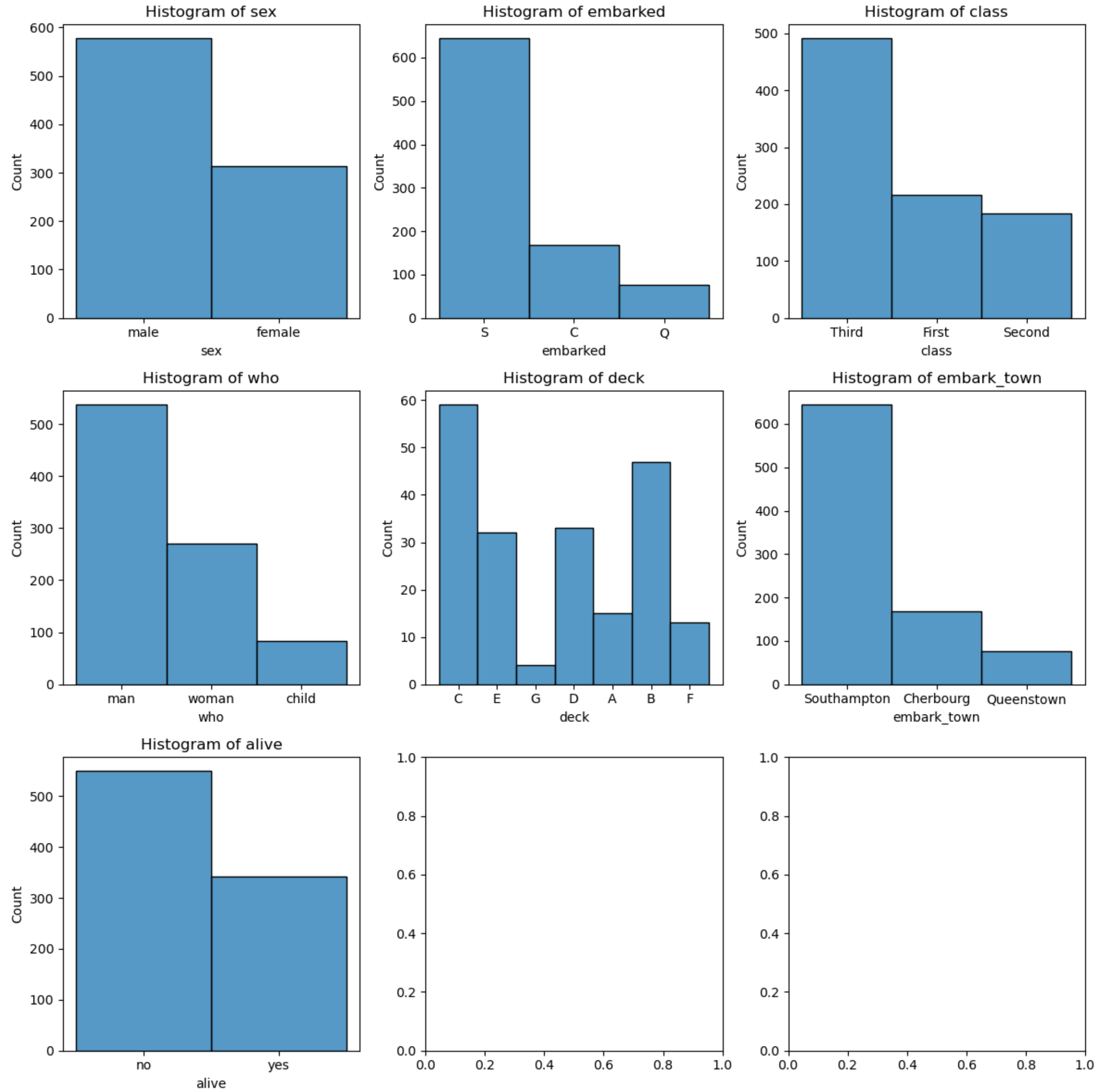
```
num_plots = len(float64_col)
num_rows = 1
num_cols = 2

fig, axes = plt.subplots(num_rows, num_cols, figsize=(6, 6))
axes = axes.flatten()

for i, column in enumerate(float64_col):
    if i < num_rows * num_cols:
        ax = axes[i]
        sns.histplot(data=Titanic, x=column, bins=10, stat='count', ax=ax, kde =True)
        ax.set_xlabel(column)
        ax.set_ylabel('Count')
        ax.set_title(f'Histogram of {column}')
        skewness = skew(Titanic[column].dropna())
        skewness_text = f'Skewness: {skewness:.2f}'
        ax.text(0.05, 0.95, skewness_text, transform=ax.transAxes, fontsize=10, vertic
    else:
        break
fig.tight_layout()
plt.show()
```



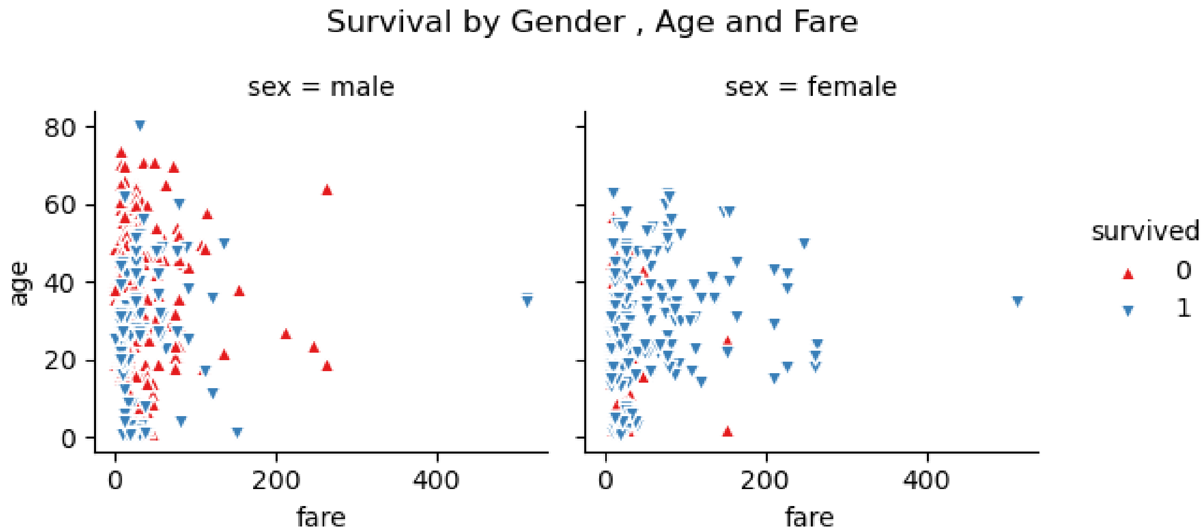
In [12]:
```
num_plots = len(object_col)
num_rows = 3
num_cols = 3
fig, axes = plt.subplots(num_rows, num_cols, figsize=(12, 12))
axes = axes.flatten()
for i, column in enumerate(object_col):
    if i < num_rows * num_cols:
        ax = axes[i]
        sns.histplot(data=Titanic, x=column, bins=10, stat='count', ax=ax)
        ax.set_xlabel(column)
        ax.set_ylabel('Count')
        ax.set_title(f'Histogram of {column}')
    else:
        break
fig.tight_layout()
plt.show()
```



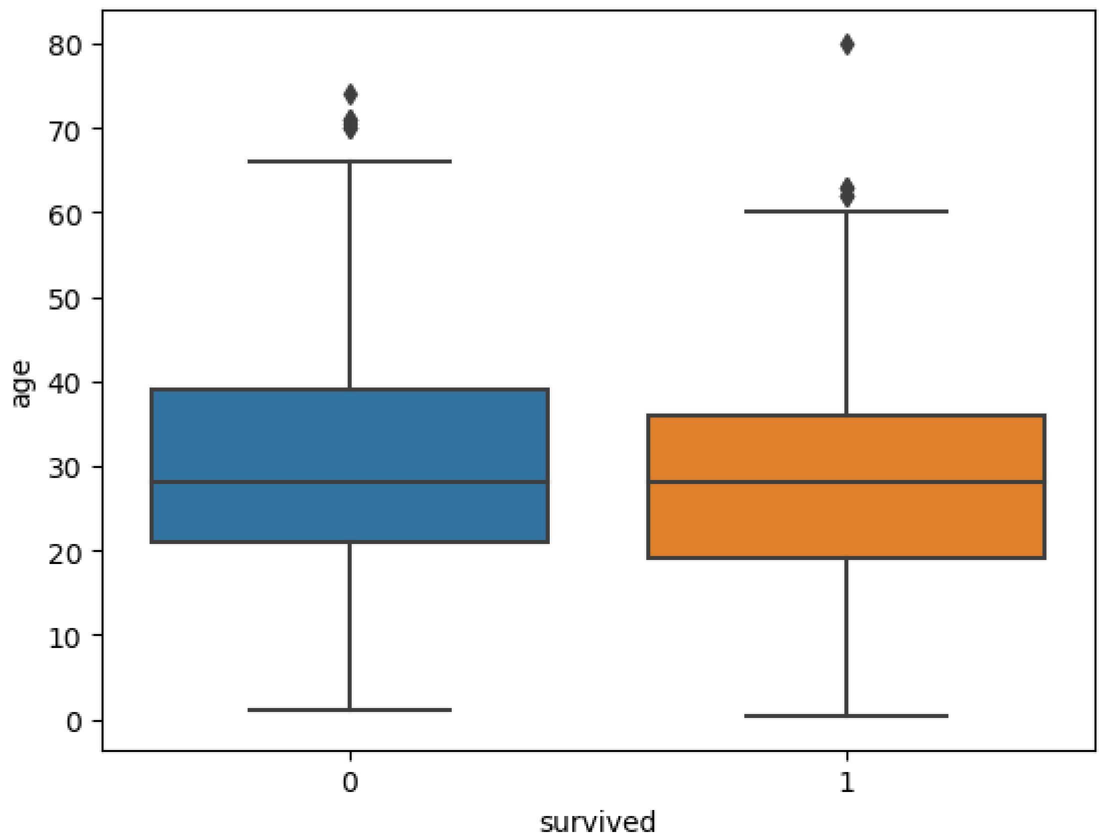
3.2. Bivariate Analysis:

In [13]:
```
g = sns.FacetGrid(Titanic, hue="survived", col="sex", margin_titles=True,
                palette="Set1",hue_kws=dict(marker=["^", "v"]))
g.map(plt.scatter, "fare", "age",edgecolor="w").add_legend()
plt.subplots_adjust(top=0.8)
g.fig.suptitle('Survival by Gender , Age and Fare')
```

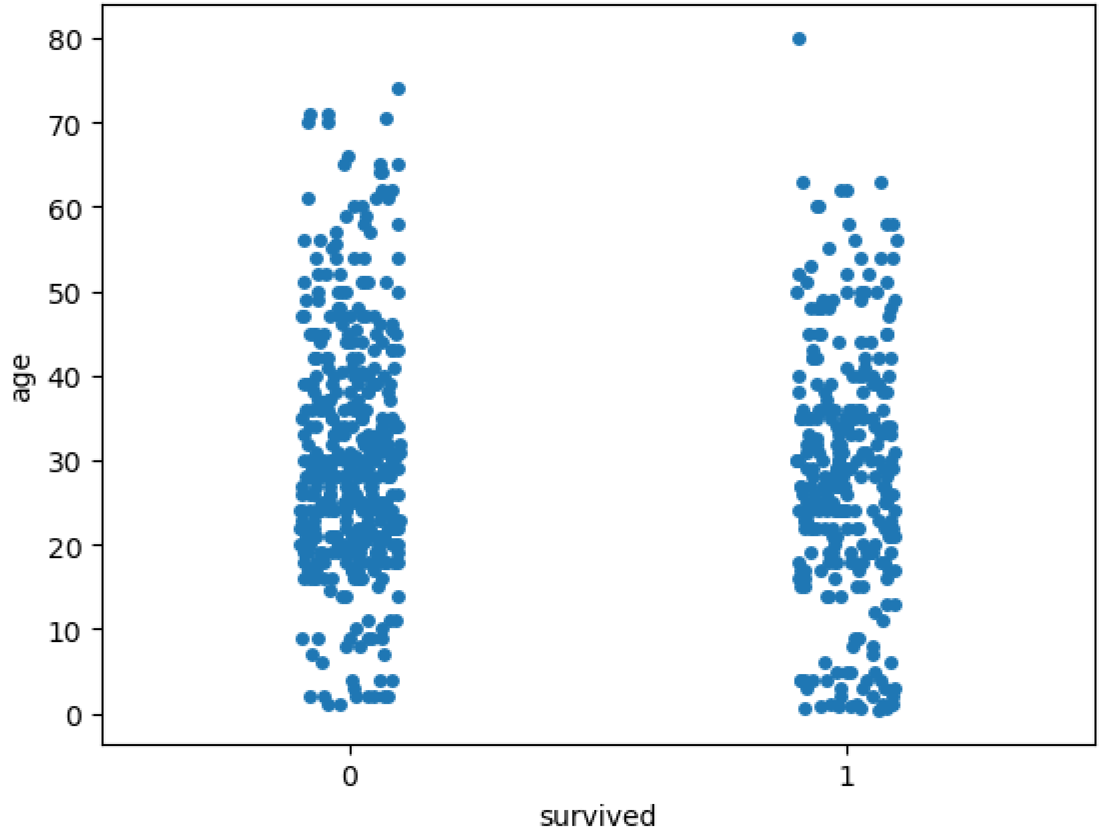
Out[13]:
```
Text(0.5, 0.98, 'Survival by Gender , Age and Fare')
```



In [14]:
```
ax = sns.boxplot(x="survived", y="age",
                data=Titanic)
```

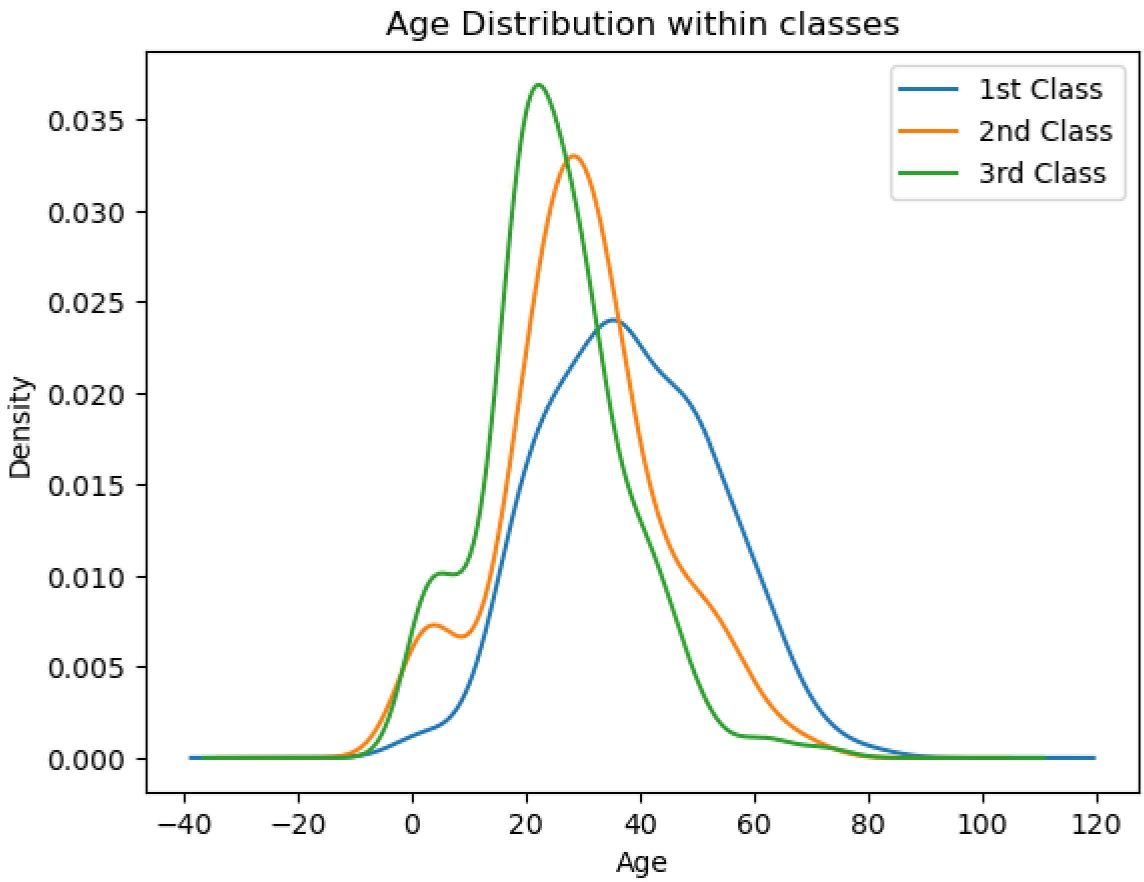


In [15]:
```
ax = sns.stripplot(x="survived", y="age",
                   data=Titanic, jitter=True,
                   edgecolor="gray")
```



In [16]:
```
Titanic.age[Titanic.pclass == 1].plot(kind='kde')
Titanic.age[Titanic.pclass == 2].plot(kind='kde')
Titanic.age[Titanic.pclass == 3].plot(kind='kde')
 # plots an axis lable
plt.xlabel("Age")
plt.title("Age Distribution within classes")
# sets our legend for our graph.
plt.legend(('1st Class', '2nd Class','3rd Class'),loc='best')
```

Out[16]: <matplotlib.legend.Legend at 0x1a342b63370>

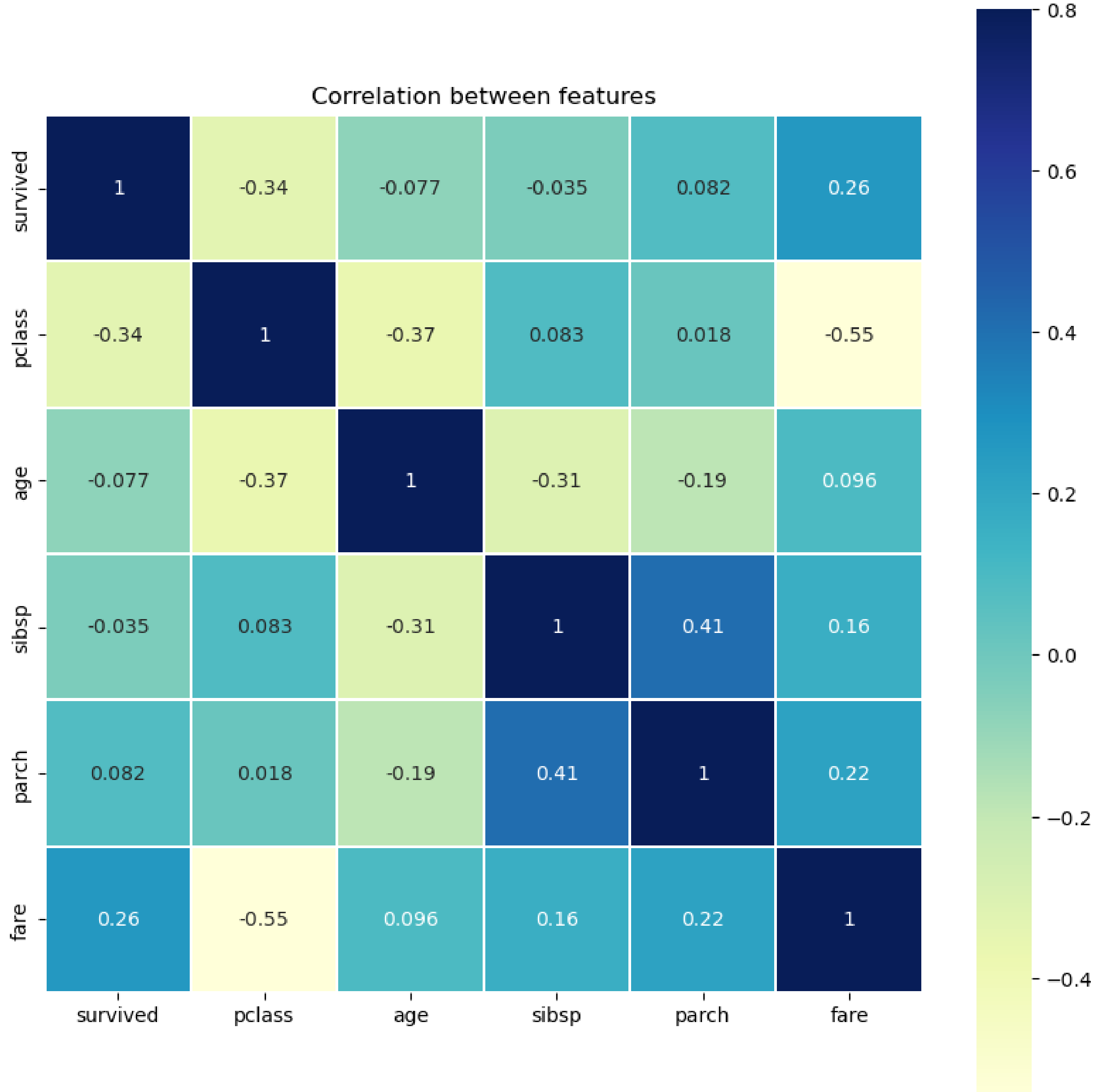


3.3. Multivariate Analysis:

In [17]:
```
corr=numeric_col.corr()#["Survived"]
plt.figure(figsize=(10, 10))

sns.heatmap(corr, vmax=.8, linewidths=0.01,
            square=True,annot=True,cmap='YlGnBu',linecolor="white")
plt.title('Correlation between features')
```

Out[17]:
```
Text(0.5, 1.0, 'Correlation between features')
```



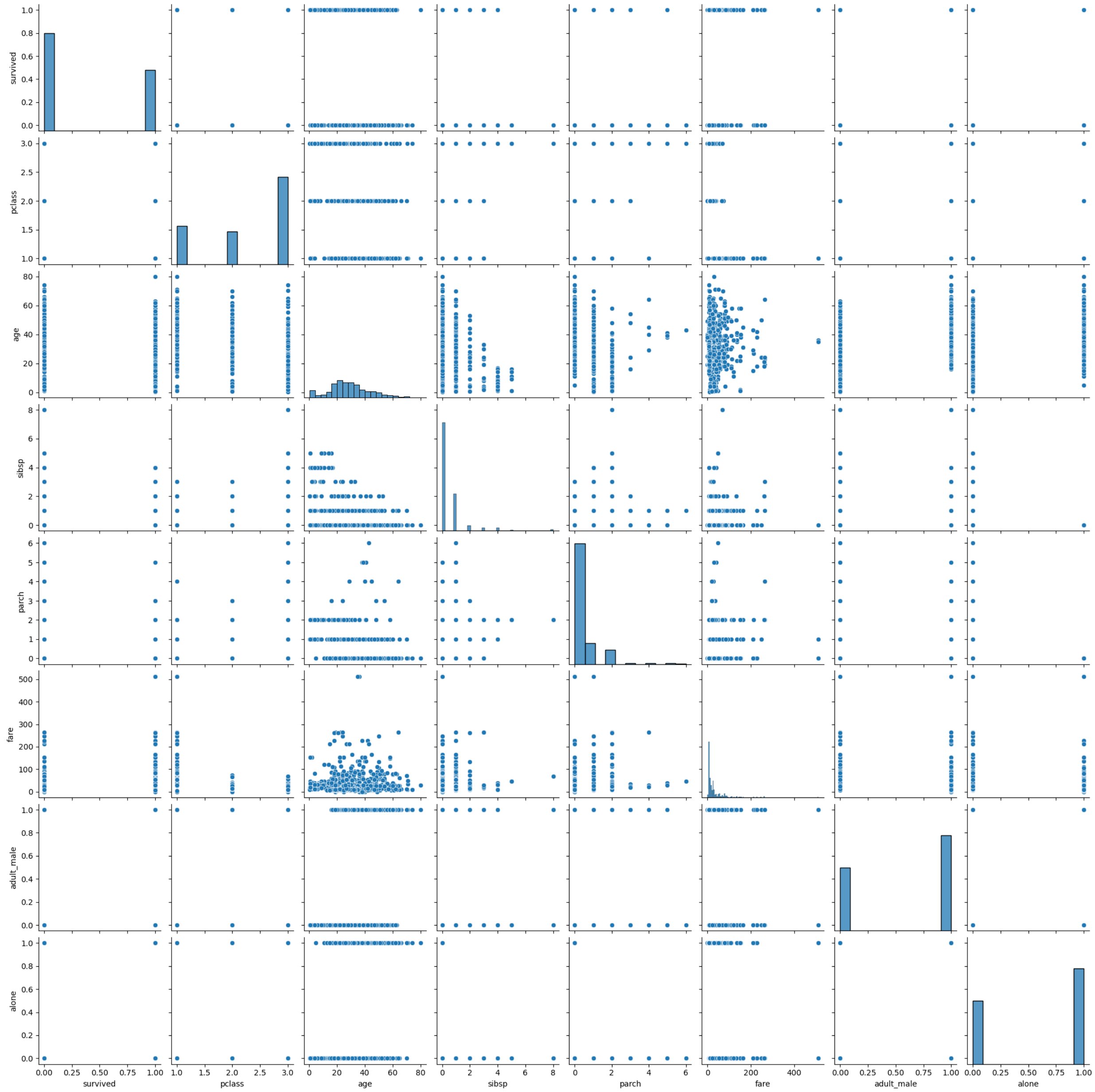
In [18]:

```
sns.pairplot(Titanic)
```

```
<__array_function__ internals>:180: RuntimeWarning: Converting input from bool to <cl
ass 'numpy.uint8'> for compatibility.
<__array_function__ internals>:180: RuntimeWarning: Converting input from bool to <cl
ass 'numpy.uint8'> for compatibility.
```

Out[18]:

```
<seaborn.axisgrid.PairGrid at 0x1a342e6ff10>
```



1. Peform Descriptive Statistics:
2. Mean, median, mode, variance, standard deviation, IQR
3. Describe().

In [19]:
```
for column in float64_col:
    quantile = Titanic[column].quantile(q=[0.25, 0.75])
    print(f"Quantile values for column '{column}':")
    print(quantile)

    q1 = quantile.iloc[0]
    q3 = quantile.iloc[1]
    IQR = q3 - q1

    print(f"Interquartile Range (IQR) for column '{column}': {IQR}")
    lower_extreme=quantile.iloc[1]-(1.5* IQR)
    print("Lower Extreme : ", lower_extreme)
    upper_extreme=quantile.iloc[0]+(1.5*IQR)
    print("Upper Extreme : ", upper_extreme,"\n")
```

```
for column in int64_col:
    quantile = Titanic[column].quantile(q=[0.25, 0.75])
    print(f"Quantile values for column '{column}':")
    print(quantile)

    q1 = quantile.iloc[0]
    q3 = quantile.iloc[1]
    IQR = q3 - q1

    print(f"Interquartile Range (IQR) for column '{column}': {IQR}")
    lower_extreme=quantile.iloc[1]-(1.5* IQR)
    print("Lower Extreme : ", lower_extreme)
    upper_extreme=quantile.iloc[0]+(1.5*IQR)
    print("Upper Extreme : ", upper_extreme,"\n")
```

```
Quantile values for column 'age':
0.25    20.125
0.75    38.000
Name: age, dtype: float64
Interquartile Range (IQR) for column 'age': 17.875
Lower Extreme :  11.1875
Upper Extreme :  46.9375

Quantile values for column 'fare':
0.25     7.9104
0.75    31.0000
Name: fare, dtype: float64
Interquartile Range (IQR) for column 'fare': 23.0896
Lower Extreme :  -3.6343999999999994
Upper Extreme :  42.5448

Quantile values for column 'survived':
0.25    0.0
0.75    1.0
Name: survived, dtype: float64
Interquartile Range (IQR) for column 'survived': 1.0
Lower Extreme :  -0.5
Upper Extreme :  1.5

Quantile values for column 'pclass':
0.25    2.0
0.75    3.0
Name: pclass, dtype: float64
Interquartile Range (IQR) for column 'pclass': 1.0
Lower Extreme :  1.5
Upper Extreme :  3.5

Quantile values for column 'sibsp':
0.25    0.0
0.75    1.0
Name: sibsp, dtype: float64
Interquartile Range (IQR) for column 'sibsp': 1.0
Lower Extreme :  -0.5
Upper Extreme :  1.5

Quantile values for column 'parch':
0.25    0.0
0.75    0.0
Name: parch, dtype: float64
Interquartile Range (IQR) for column 'parch': 0.0
Lower Extreme :  0.0
Upper Extreme :  0.0
```

In [20]:

```
print("Variance :\n\n", numeric_col.var())
print("Mean : \n\n", numeric_col.mean())
print("Median : \n\n", numeric_col.median())
print("Mode : \n\n", numeric_col.mode())
print("Standard Deviation : \n\n", numeric_col.std())
```

```
Variance :

 survived       0.236772
pclass         0.699015
age          211.019125
sibsp          1.216043
parch          0.649728
fare        2469.436846
dtype: float64
Mean :

 survived     0.383838
pclass       2.308642
age         29.699118
sibsp        0.523008
parch        0.381594
fare        32.204208
dtype: float64
Median :

 survived     0.0000
pclass       3.0000
age         28.0000
sibsp        0.0000
parch        0.0000
fare        14.4542
dtype: float64
Mode :

    survived  pclass   age  sibsp  parch  fare
0         0       3  24.0      0      0  8.05
Standard Deviation :

 survived     0.486592
pclass       0.836071
age         14.526497
sibsp        1.102743
parch        0.806057
fare        49.693429
dtype: float64
```

In [21]:

```
numeric_col.describe()
```

Out[21]:

	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

1. Handling Missing Values:

In [22]:
```
null_counts = Titanic.isnull().sum()
total_counts = Titanic.count()
dict_1 = {'Total Count' : total_counts, "Null Count" : null_counts}
null_table = pd.DataFrame(dict_1)
null_table.index.name = "Column Names"
print(null_table)
```

```
              Total Count  Null Count
Column Names
survived              891           0
pclass                891           0
sex                   891           0
age                   714         177
sibsp                 891           0
parch                 891           0
fare                  891           0
embarked              889           2
class                 891           0
who                   891           0
adult_male            891           0
deck                  203         688
embark_town           889           2
alive                 891           0
alone                 891           0
```

In [23]:
```
#For Embarked column:
Titanic["embarked"] = Titanic["embarked"].fillna('C')

#For Embark_town column:
Titanic["embark_town"] = Titanic["embark_town"].fillna('Cherbourg')
#For Deck column:
Titanic['deck'] = Titanic['deck'].fillna(Titanic['deck'].mode()[0])
#For Age Column:
Titanic['age'] = Titanic['age'].fillna(Titanic['age'].mean())
null_counts = Titanic.isnull().sum()
total_counts = Titanic.count()
dict_1 = {'Total Count' : total_counts, "Null Count" : null_counts}
null_table = pd.DataFrame(dict_1)
null_table.index.name = "Column Names"
print(null_table)
```

```
              Total Count  Null Count
Column Names
survived              891           0
pclass                891           0
sex                   891           0
age                   891           0
sibsp                 891           0
parch                 891           0
fare                  891           0
embarked              891           0
class                 891           0
who                   891           0
adult_male            891           0
deck                  891           0
embark_town           891           0
alive                 891           0
alone                 891           0
```

1. Finding Outliers and Removing it:

   A. Finding Outliers.

   B. Removing It. 6.1. Finding Outliers

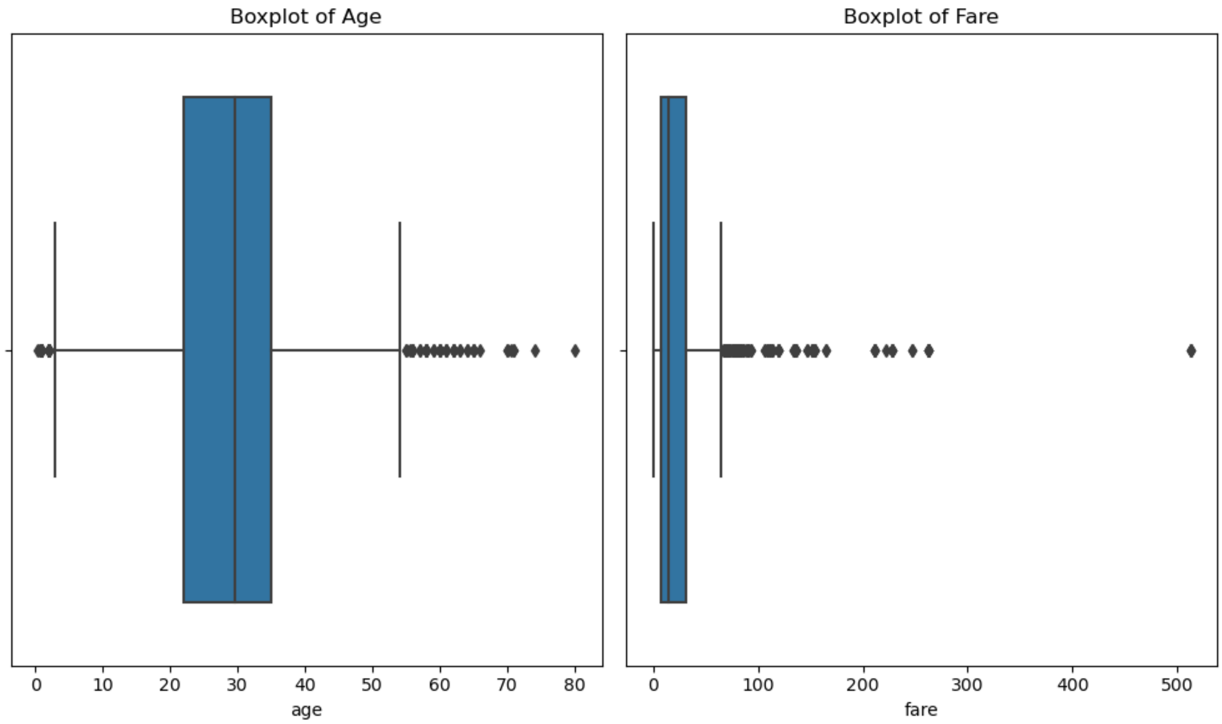
In [24]:
```
fig, axes = plt.subplots(1, 2, figsize=(10, 6))

# Boxplot for 'age'
sns.boxplot(data=Titanic, x='age', ax=axes[0])
axes[0].set_title('Boxplot of Age')

# Boxplot for 'fare'
sns.boxplot(data=Titanic, x='fare', ax=axes[1])
axes[1].set_title('Boxplot of Fare')

# Adjust the spacing between subplots
plt.tight_layout()

# Show the plots
plt.show()
```



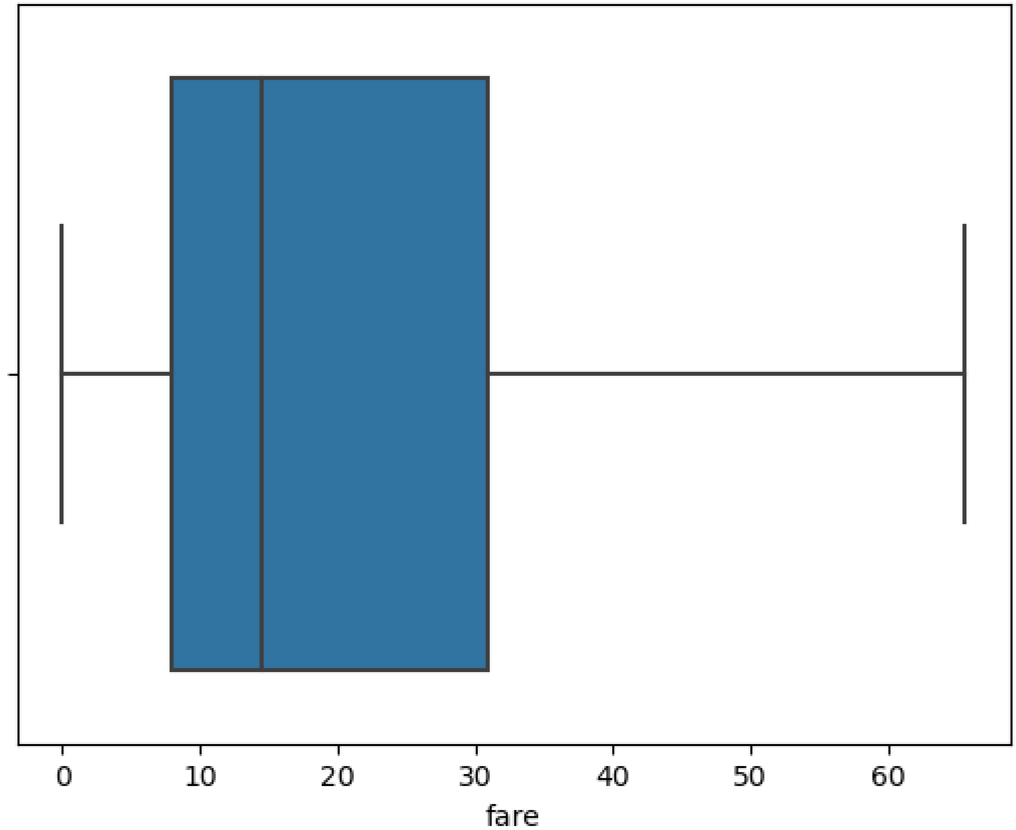
6.2. Removing(Handling) Outliers:

In [25]:
```
#From the above plot, can see fare is the column that have outliers which are to be re
#remaining numeric columns are not useful that much.
Q1 = Titanic['fare'].quantile(0.25)
Q3 = Titanic['fare'].quantile(0.75)
IQR = Q3 - Q1
whisker_width = 1.5
lower_whisker = Q1 -(whisker_width*IQR)
upper_whisker = Q3 + (whisker_width*IQR)
Titanic['fare']=np.where(Titanic['fare']>upper_whisker,upper_whisker,np.where(Titanic[
```

After removing most of the outliers.

In [26]:
```
# Boxplot for 'fare'
sns.boxplot(data=Titanic, x='fare')
```

Out[26]:
```
<Axes: xlabel='fare'>
```



1. Check for Categorical columns and perform encoding:

In [27]:
```
# Get the list of categorical columns
categorical_columns = Titanic.select_dtypes(include=['object']).columns

# Perform one-hot encoding
Titanic_encoded = pd.get_dummies(Titanic, columns=categorical_columns, drop_first=True
Titanic_encoded.head()
```

Out[27]:

	survived	pclass	age	sibsp	parch	fare	adult_male	alone	sex_male	embarked_Q	...	who_\
0	0	3	22.0	1	0	7.2500	True	False	1	0	...	
1	1	1	38.0	1	0	65.6344	False	False	0	0	...	
2	1	3	26.0	0	0	7.9250	False	True	0	0	...	
3	1	1	35.0	1	0	53.1000	False	False	0	0	...	
4	0	3	35.0	0	0	8.0500	True	True	1	0	...	

5 rows × 24 columns

1. Split the data into dependent and independent variables.
2. Scale the independent variables:

In [28]:
```
X = Titanic.drop('survived', axis=1)
y = Titanic['survived']
```

1. Split the data into training and testing:

In [29]:
```
# Scale the independent variables
scaler = StandardScaler()
X_encoded = pd.get_dummies(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, rando

X_scaled = scaler.fit_transform(X_train)
# Scaling should be applied to only to the training data and not to the whole dataset
```