In [15]:

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

In [16]:

```
df=pd.read_csv('titanic.csv')
df
```

Out[16]:

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

891 rows × 15 columns

In [17]:

```
df.head()
```

Out[17]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_ma
0	0	3	male	22.0	1	0	7.2500	S	Third	man	Tru
1	1	1	female	38.0	1	0	71.2833	C	First	woman	Fals
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	Fals
3	1	1	female	35.0	1	0	53.1000	S	First	woman	Fals
4	0	3	male	35.0	0	0	8.0500	S	Third	man	Tru

In [18]:

```
df.tail()
```

Out[18]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_n
886	0	2	male	27.0	0	0	13.00	S	Second	man	T
887	1	1	female	19.0	0	0	30.00	S	First	woman	F
888	0	3	female	NaN	1	2	23.45	S	Third	woman	F
889	1	1	male	26.0	0	0	30.00	C	First	man	T
890	0	3	male	32.0	0	0	7.75	Q	Third	man	T

In [19]:

```
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
```

In [20]:

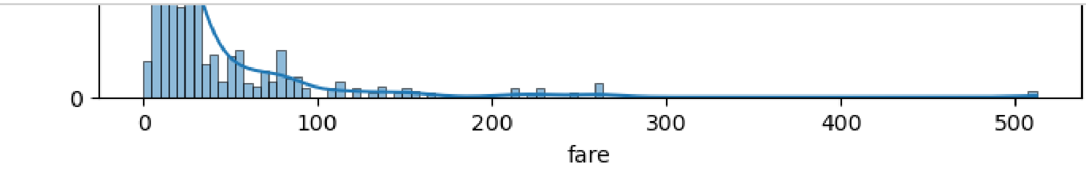
```
df['age']
```

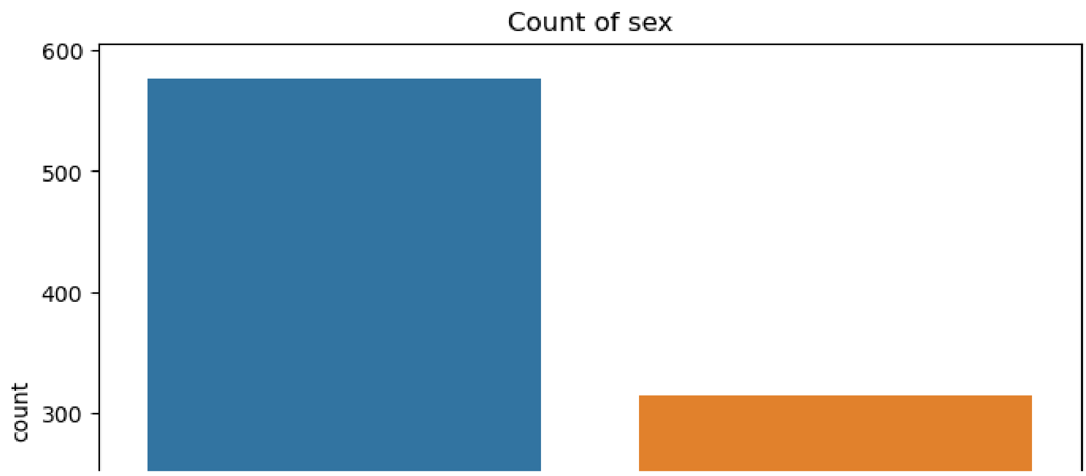
Out[20]:

```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888     NaN
889    26.0
890    32.0
Name: age, Length: 891, dtype: float64
```

In [21]:

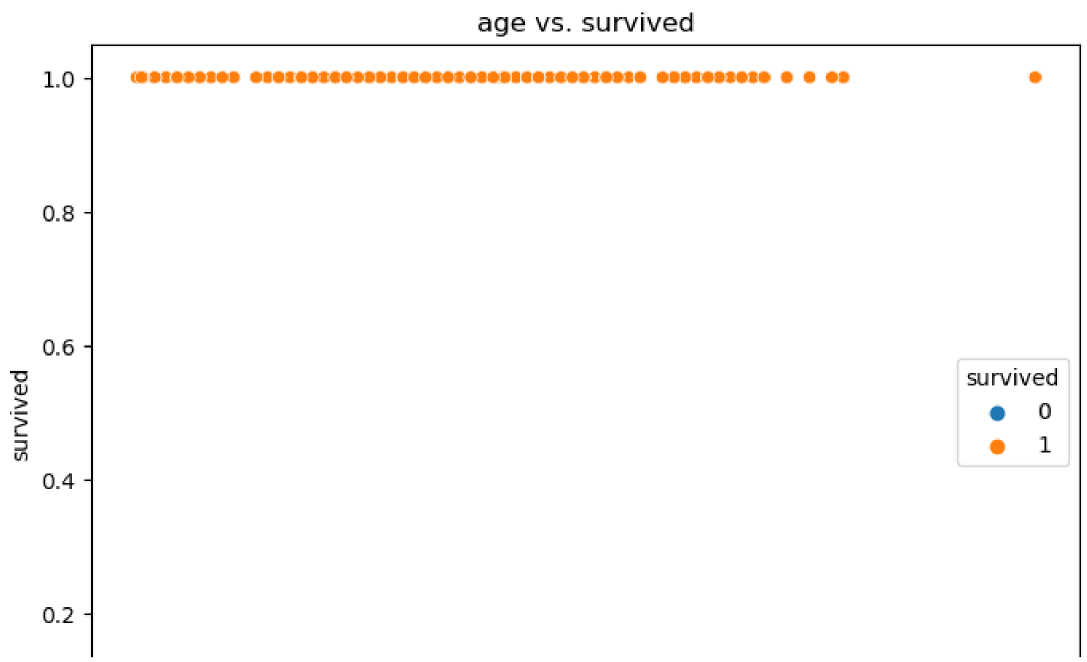
```
#Univariate Analysis

# Univariate analysis for numerical variables
numeric_vars = ['age', 'fare']
for var in numeric_vars:
    plt.figure(figsize=(8, 6))
    sns.histplot(data=df, x=var, kde=True)
    plt.title(f'Distribution of {var}')
    plt.show()

# Univariate analysis for categorical variables
categorical_vars = ['sex', 'embarked', 'pclass', 'survived']
for var in categorical_vars:
    plt.figure(figsize=(8, 6))
    sns.countplot(data=df, x=var)
    plt.title(f'Count of {var}')
    plt.show()
```





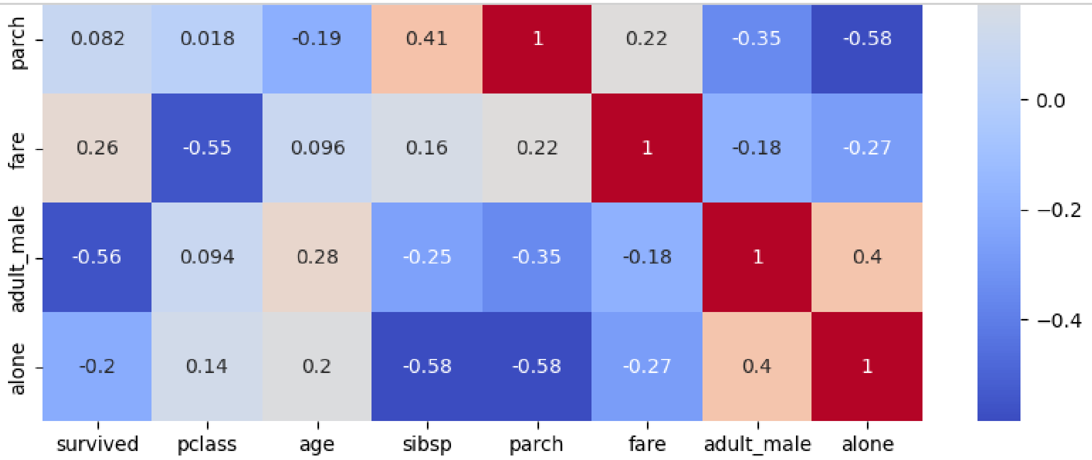
In [22]:

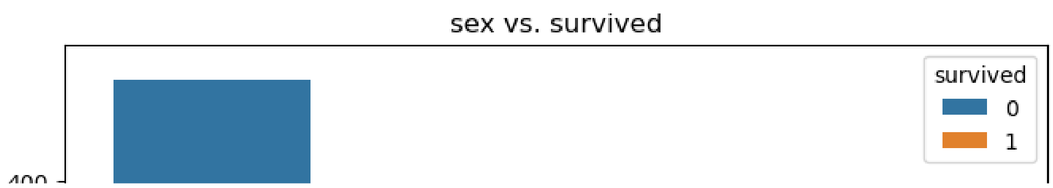
```
#Bivariate Analysis
# Bivariate analysis for numerical variables
numeric_vars = ['age', 'fare']
for var in numeric_vars:
    plt.figure(figsize=(8, 6))
    sns.scatterplot(data=df, x=var, y='survived', hue='survived')
    plt.title(f'{var} vs. survived')
    plt.show()

# Bivariate analysis for categorical variables
categorical_vars = ['sex', 'embarked', 'pclass']
for var in categorical_vars:
    plt.figure(figsize=(8, 6))
    sns.countplot(data=df, x=var, hue='survived')
    plt.title(f'{var} vs. survived')
    plt.show()
```



In [23]:

```
#Multivariate Analysis

# Multivariate analysis using scatter plot matrix
numeric_vars = ['age', 'fare']
sns.pairplot(data=df, vars=numeric_vars, hue='survived')
plt.title('Pairwise Scatter Plot of Numeric Variables')
plt.show()

# Multivariate analysis using a heatmap of correlation matrix
corr_matrix = df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()

# Multivariate analysis using a grouped bar plot
categorical_vars = ['sex', 'embarked', 'pclass']
for var in categorical_vars:
    plt.figure(figsize=(8, 6))
    sns.countplot(data=df, x=var, hue='survived')
    plt.title(f'{var} vs. survived')
    plt.show()
```





In [24]:

```
#Descriptive Statistics

# Compute descriptive statistics
statistics = df.describe()

# Display the descriptive statistics
print(statistics)
```

```
         survived      pclass         age       sibsp       parch          f
are
count  891.000000  891.000000  714.000000  891.000000  891.000000  891.000
000
mean     0.383838    2.308642   29.699118    0.523008    0.381594   32.204
208
std      0.486592    0.836071   14.526497    1.102743    0.806057   49.693
429
min      0.000000    1.000000    0.420000    0.000000    0.000000    0.000
000
25%      0.000000    2.000000   20.125000    0.000000    0.000000    7.910
400
50%      0.000000    3.000000   28.000000    0.000000    0.000000   14.454
200
75%      1.000000    3.000000   38.000000    1.000000    0.000000   31.000
000
max      1.000000    3.000000   80.000000    8.000000    6.000000  512.329
200
```

In [25]:

```
#Descriptive Statistics
#Measures of central tendency
#Mean

df.mean()
```

```
C:\Users\vijay\AppData\Local\Temp\ipykernel_2076\1208449491.py:5: FutureWa
rning: The default value of numeric_only in DataFrame.mean is deprecated.
In a future version, it will default to False. In addition, specifying 'nu
meric_only=None' is deprecated. Select only valid columns or specify the v
alue of numeric_only to silence this warning.
  df.mean()
```

Out[25]:

```
survived       0.383838
pclass         2.308642
age           29.699118
sibsp          0.523008
parch          0.381594
fare          32.204208
adult_male     0.602694
alone          0.602694
dtype: float64
```

In [26]:

```
#Median
df.median()
```

```
C:\Users\vijay\AppData\Local\Temp\ipykernel_2076\863618092.py:2: FutureWar
ning: The default value of numeric_only in DataFrame.median is deprecated.
In a future version, it will default to False. In addition, specifying 'nu
meric_only=None' is deprecated. Select only valid columns or specify the v
alue of numeric_only to silence this warning.
  df.median()
```

Out[26]:

```
survived       0.0000
pclass         3.0000
age           28.0000
sibsp          0.0000
parch          0.0000
fare          14.4542
adult_male     1.0000
alone          1.0000
dtype: float64
```

In [27]:

```
#Mode
df.mode()
```

Out[27]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	decl
0	0	3	male	24.0	0	0	8.05	S	Third	man	True	(

In [28]:

```
#Skewness
df.skew()
```

```
C:\Users\vijay\AppData\Local\Temp\ipykernel_2076\3944243432.py:2: FutureWa
rning: The default value of numeric_only in DataFrame.skew is deprecated.
In a future version, it will default to False. In addition, specifying 'nu
meric_only=None' is deprecated. Select only valid columns or specify the v
alue of numeric_only to silence this warning.
  df.skew()
```

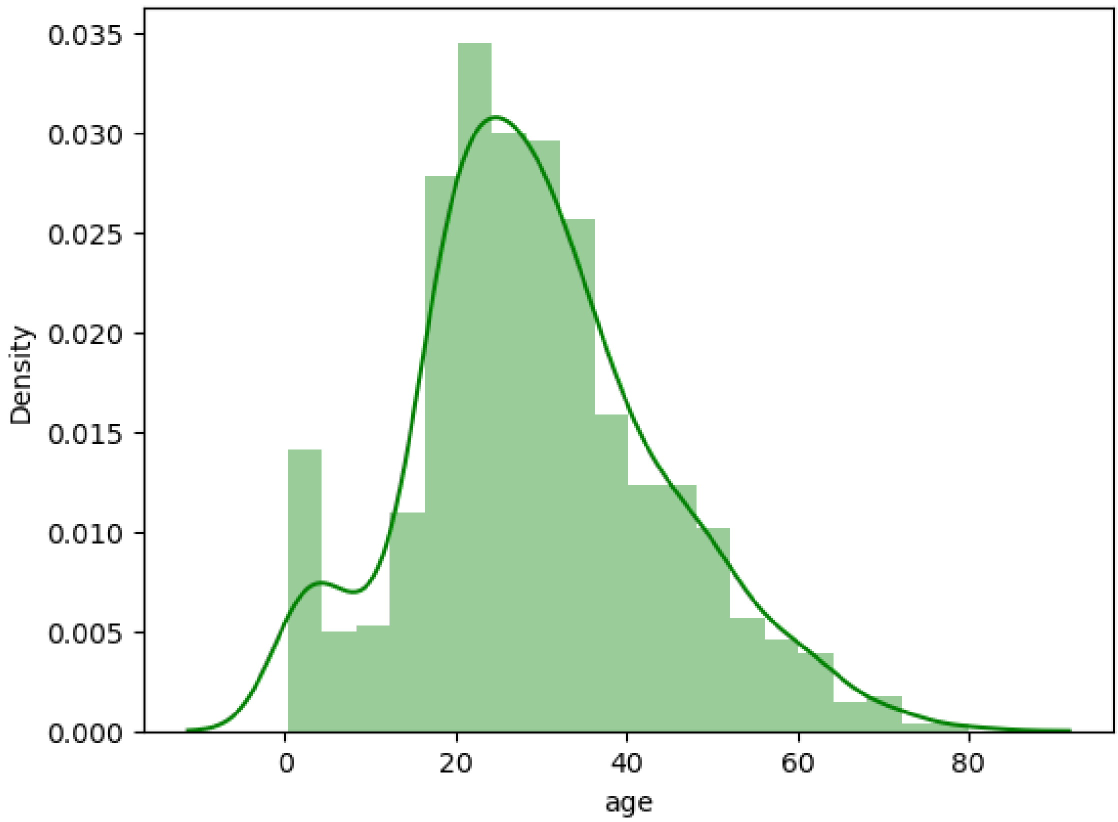
Out[28]:

```
survived      0.478523
pclass       -0.630548
age           0.389108
sibsp         3.695352
parch         2.749117
fare          4.787317
adult_male   -0.420431
alone        -0.420431
dtype: float64
```

In [29]:

```
# Distplot
print(sns.distplot(df['age'],color='green'))
```

```
C:\Users\vijay\AppData\Local\Temp\ipykernel_2076\3882639869.py:2: UserWarn
ing:

`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function wi
th
similar flexibility) or `histplot` (an axes-level function for histogram
s).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://
gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  print(sns.distplot(df['age'],color='green'))

Axes(0.125,0.11;0.775x0.77)
```

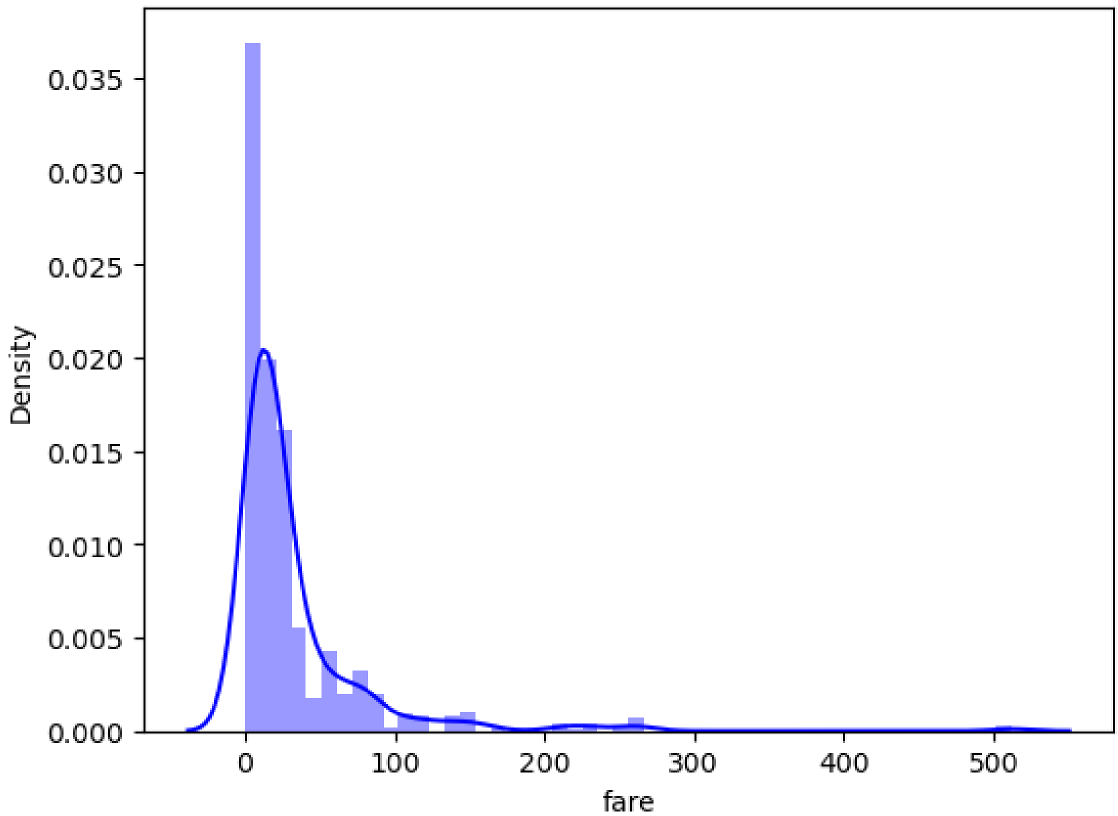


In [30]:

```
print(sns.distplot(df['fare'],color='blue'))
```

```
C:\Users\vijay\AppData\Local\Temp\ipykernel_2076\925598583.py:1: UserWarni
ng:

`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function wi
th
similar flexibility) or `histplot` (an axes-level function for histogram
s).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://
gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  print(sns.distplot(df['fare'],color='blue'))
```

```
Axes(0.125,0.11;0.775x0.77)
```



In [31]:

```
# kurtosis
df.kurt()
```

```
C:\Users\vijay\AppData\Local\Temp\ipykernel_2076\3536932851.py:2: FutureWa
rning: The default value of numeric_only in DataFrame.kurt is deprecated.
In a future version, it will default to False. In addition, specifying 'nu
meric_only=None' is deprecated. Select only valid columns or specify the v
alue of numeric_only to silence this warning.
  df.kurt()
```

Out[31]:

```
survived      -1.775005
pclass        -1.280015
age            0.178274
sibsp         17.880420
parch          9.778125
fare          33.398141
adult_male    -1.827345
alone         -1.827345
dtype: float64
```

In [32]:

```
#Range

df.max()
```

```
C:\Users\vijay\AppData\Local\Temp\ipykernel_2076\625711735.py:3: FutureWar
ning: The default value of numeric_only in DataFrame.max is deprecated. In
a future version, it will default to False. In addition, specifying 'numer
ic_only=None' is deprecated. Select only valid columns or specify the valu
e of numeric_only to silence this warning.
  df.max()
```

Out[32]:

```
survived             1
pclass               3
sex               male
age               80.0
sibsp                8
parch                6
fare          512.3292
class            Third
who              woman
adult_male        True
alive              yes
alone             True
dtype: object
```

In [33]:

```
df.min()
```

```
C:\Users\vijay\AppData\Local\Temp\ipykernel_2076\3962516015.py:1: FutureWa
rning: The default value of numeric_only in DataFrame.min is deprecated. I
n a future version, it will default to False. In addition, specifying 'num
eric_only=None' is deprecated. Select only valid columns or specify the va
lue of numeric_only to silence this warning.
  df.min()
```

Out[33]:

```
survived          0
pclass            1
sex          female
age            0.42
sibsp             0
parch             0
fare            0.0
class         First
who           child
adult_male    False
alive            no
alone         False
dtype: object
```

In [34]:

```
column = 'age'

# Find the range
column_range = df[column].max() - df[column].min()

# Print the range
print(f"The range for '{column}' is: {column_range}")
```

```
The range for 'age' is: 79.58
```

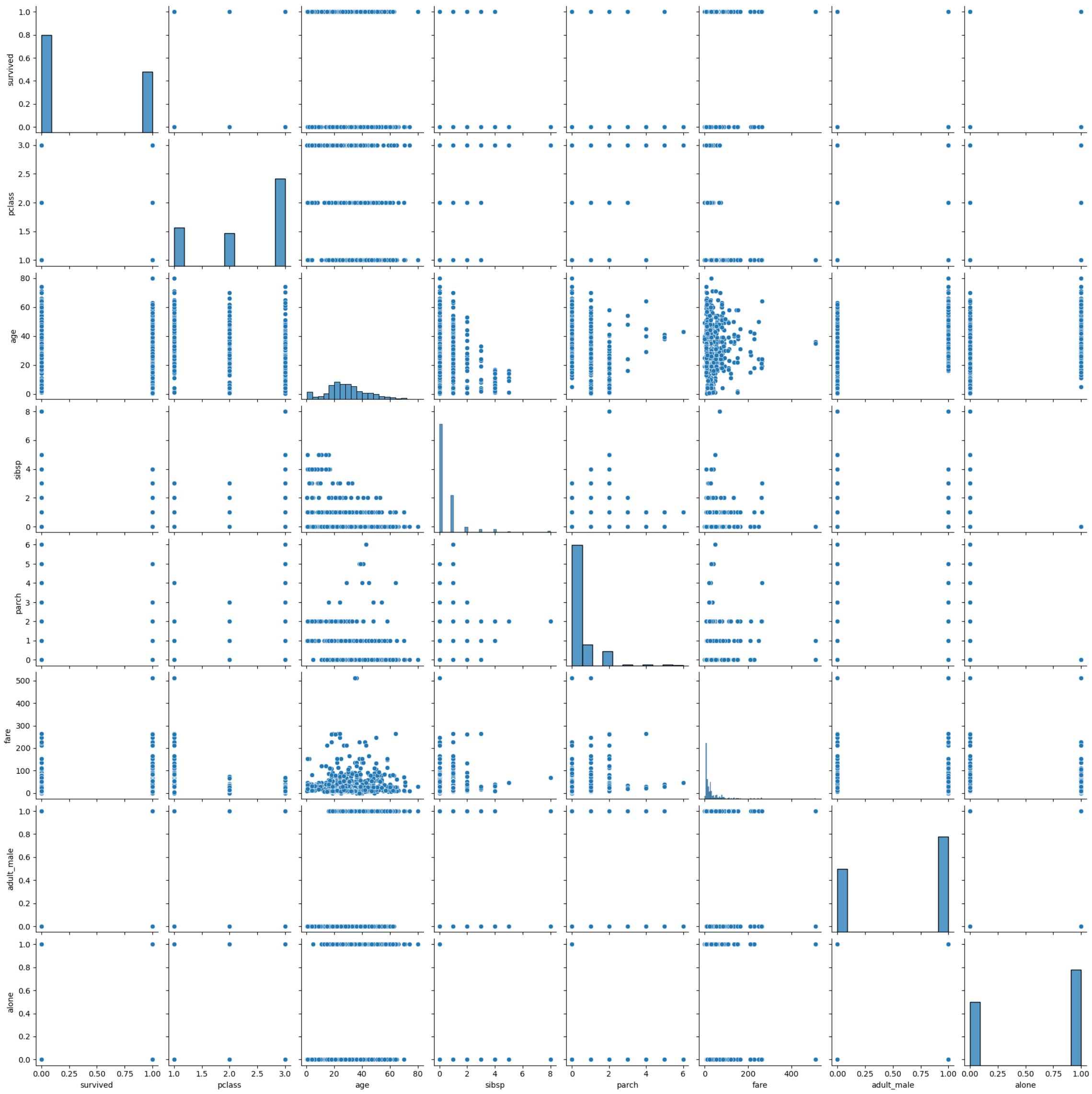
In [35]:

```
sns.pairplot(df)
```

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

Out[35]:

<seaborn.axisgrid.PairGrid at 0x1cc2ced02e0>



In [36]:

```
#Handling the Missing values

# Check for missing values
print(df.isnull().sum())

# Handling missing values for numerical columns
df['age'].fillna(df['age'].median(), inplace=True)
df['fare'].fillna(df['fare'].mean(), inplace=True)

# Handling missing values for categorical columns
df['embarked'].fillna(df['embarked'].mode()[0], inplace=True)

# Dropping rows with missing values
#data.dropna(inplace=True)

# Verify if missing values are handled
print(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
survived         0
pclass           0
sex              0
age              0
sibsp            0
parch            0
fare             0
embarked         0
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

In [37]:

```
# Find the outliers and replace the outliers

# Identify outliers in numerical columns
numeric_vars = ['age', 'fare']

for var in numeric_vars:
    # Calculate the IQR (Interquartile Range)
    Q1 = df[var].quantile(0.25)
    Q3 = df[var].quantile(0.75)
    IQR = Q3 - Q1

    # Determine the upper and lower bounds for outliers
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Identify outliers
    outliers = df[(df[var] < lower_bound) | (df[var] > upper_bound)]

    # Replace outliers with appropriate values
    df[var] = np.where((df[var] < lower_bound) | (df[var] > upper_bound), df[var].median

# Verify if outliers are replaced
for var in numeric_vars:
    # Calculate the IQR (Interquartile Range)
    Q1 = df[var].quantile(0.25)
    Q3 = df[var].quantile(0.75)
    IQR = Q3 - Q1

    # Determine the upper and lower bounds for outliers
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Identify outliers
    outliers = df[(df[var] < lower_bound) | (df[var] > upper_bound)]

    # Print the outliers (should be empty if outliers are replaced)
    print(f"Outliers in '{var}':")
    print(outliers)
```

```
Outliers in 'age':
     survived  pclass     sex   age  sibsp  parch     fare embarked   c
lass  \
6           0       1    male  54.0      0      0  51.8625        S   F
irst
10          1       3  female   4.0      1      1  16.7000        S   T
hird
24          0       3  female   8.0      3      1  21.0750        S   T
hird
43          1       2  female   3.0      1      2  41.5792        C  Se
cond
50          0       3    male   7.0      4      1  39.6875        S   T
hird
..        ...     ...     ...   ...    ...    ...      ...      ...
...
857         1       1    male  51.0      0      0  26.5500        S   F
irst
862         1       1  female  48.0      0      0  25.9292        S   F
irst
```

In [38]:

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


# Check for categorical columns
categorical_cols = df.select_dtypes(include=['object']).columns
print("Categorical columns:")
print(categorical_cols)

# Perform categorical encoding
#Label Encoding
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
for col in categorical_cols:
    df[col] = label_encoder.fit_transform(df[col])

#One-Hot Encoding
data = pd.get_dummies(df, columns=categorical_cols, drop_first=True)


# Display the encoded dataset
print(data.head())
```

```
Categorical columns:
Index(['sex', 'embarked', 'class', 'who', 'deck', 'embark_town', 'alive'],
dtype='object')
   survived  pclass   age  sibsp  parch     fare  adult_male  alone  sex_1  \
0         0       3  22.0      1      0   7.2500        True  False      1
1         1       1  38.0      1      0  14.4542       False  False      0
2         1       3  26.0      0      0   7.9250       False   True      0
3         1       1  35.0      1      0  53.1000       False  False      0
4         0       3  35.0      0      0   8.0500        True   True      1

   embarked_1  ...  deck_2  deck_3  deck_4  deck_5  deck_6  deck_7  \
0           0  ...       0       0       0       0       0       1
1           0  ...       1       0       0       0       0       0
2           0  ...       0       0       0       0       0       1
3           0  ...       1       0       0       0       0       0
4           0  ...       0       0       0       0       0       1

   embark_town_1  embark_town_2  embark_town_3  alive_1
0              0              1              0        0
1              0              0              0        1
2              0              1              0        1
3              0              1              0        1
4              0              1              0        0

[5 rows x 26 columns]
```

In [39]:

```
# Split the data into dependent and independent variables
X = df.drop('survived', axis=1)  # Independent variables (features)
y = df['survived']  # Dependent variable (target)

# Display the independent variables (features)
print("Independent variables (features):")
print(X.head())

# Display the dependent variable (target)
print("\nDependent variable (target):")
print(y.head())
```

```
Independent variables (features):
   pclass  sex   age  sibsp  parch     fare  embarked  class  who  adult_m
ale  \
0       3    1  22.0      1      0   7.2500         2      2    1         T
rue
1       1    0  38.0      1      0  14.4542         0      0    2        Fa
lse
2       3    0  26.0      0      0   7.9250         2      2    2        Fa
lse
3       1    0  35.0      1      0  53.1000         2      0    2        Fa
lse
4       3    1  35.0      0      0   8.0500         2      2    1         T
rue

   deck  embark_town  alive  alone
0     7            2      0  False
1     2            0      1  False
2     7            2      1   True
3     2            2      1  False
4     7            2      0   True

Dependent variable (target):
0    0
1    1
2    1
3    1
4    0
Name: survived, dtype: int64
```

In [40]:

```
#Scale the independent variables

from sklearn.preprocessing import StandardScaler

# Split the data into dependent and independent variables
X = df.drop('survived', axis=1)  # Independent variables (features)
y = df['survived']  # Dependent variable (target)

# Create a StandardScaler object
scaler = StandardScaler()

# Scale the independent variables
X_scaled = scaler.fit_transform(X)

# Convert the scaled variables back to a DataFrame (optional)
X_scaled = pd.DataFrame(X_scaled, columns=X.columns)

# Display the scaled variables
print(X_scaled.head())
```

```
     pclass       sex       age     sibsp     parch      fare  embarked  \
0  0.827377  0.737695 -0.661724  0.432793 -0.473674 -0.797554  0.585954
1 -1.566107 -1.355574  0.972921  0.432793 -0.473674 -0.230556 -1.942303
2  0.827377 -1.355574 -0.253063 -0.474545 -0.473674 -0.744429  0.585954
3 -1.566107 -1.355574  0.666425  0.432793 -0.473674  2.811012  0.585954
4  0.827377  0.737695  0.666425 -0.474545 -0.473674 -0.734591  0.585954

      class       who  adult_male      deck  embark_town     alive     alo
ne
0  0.827377 -0.355242    0.811922  0.512048     0.581114 -0.789272 -1.2316
45
1 -1.566107  1.328379   -1.231645 -1.914733    -1.938460  1.266990 -1.2316
45
2  0.827377  1.328379   -1.231645  0.512048     0.581114  1.266990  0.8119
22
3 -1.566107  1.328379   -1.231645 -1.914733     0.581114  1.266990 -1.2316
45
4  0.827377 -0.355242    0.811922  0.512048     0.581114 -0.789272  0.8119
22
```

In [42]:

```
from sklearn.model_selection import train_test_split

# Split the data into dependent and independent variables
X = df.drop('survived', axis=1)  # Independent variables (features)
y = df['survived']  # Dependent variable (target)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42

# Display the shapes of the training and testing sets
print("Shape of X_train:", X_train.shape)
print("Shape of X_test:", X_test.shape)
print("Shape of y_train:", y_train.shape)
print("Shape of y_test:", y_test.shape)
```

```
Shape of X_train: (712, 14)
Shape of X_test: (179, 14)
Shape of y_train: (712,)
Shape of y_test: (179,)
```

In [ ]:

```

```

In [ ]:

```

```