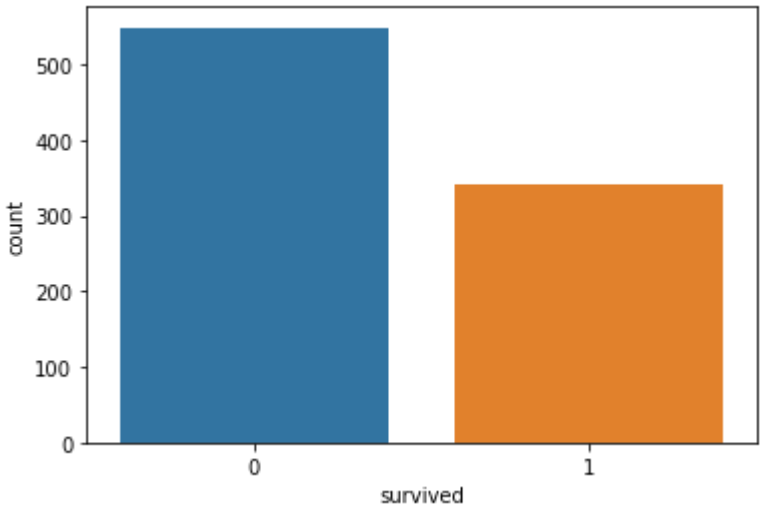
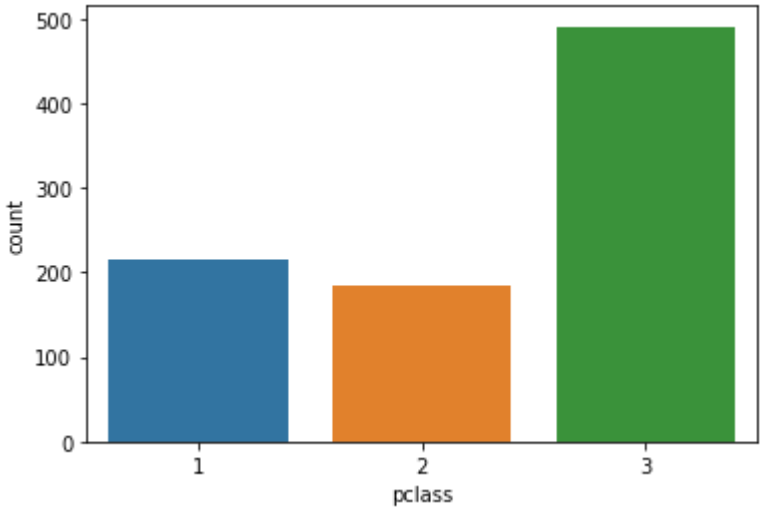
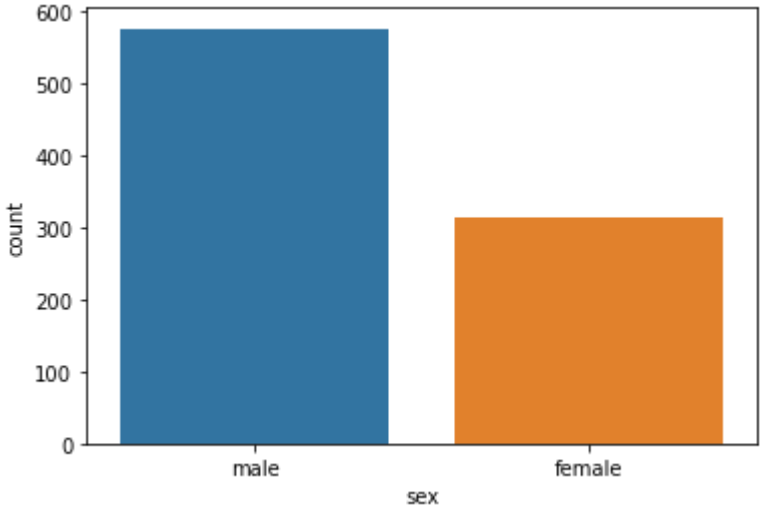
In [50]:
```
#DIGITAL ASSIGNMENT - 2
#20MID0170
#GIRISH KUMAR A
```

In [51]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
import numpy as np
# Load the dataset
data = pd.read_csv('titanic.csv')
```

In [37]:
```
# Visualizations

# Univariate Analysis
sns.countplot(x='survived', data=data)
plt.show()

sns.countplot(x='pclass', data=data)
plt.show()

sns.countplot(x='sex', data=data)
plt.show()
```

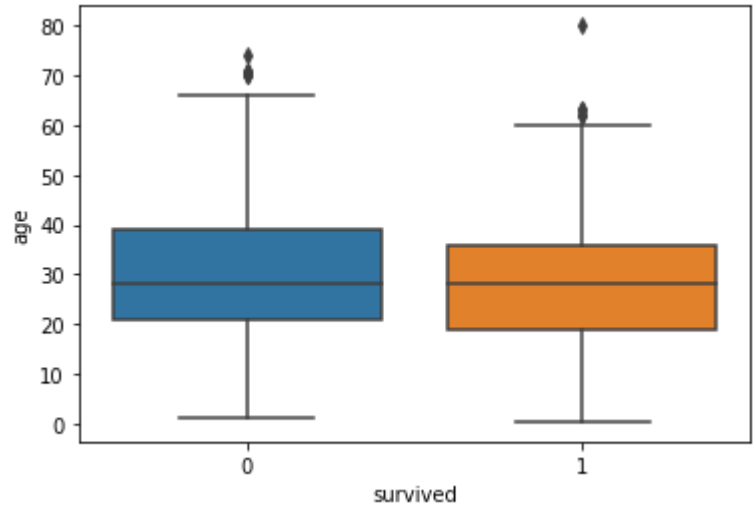


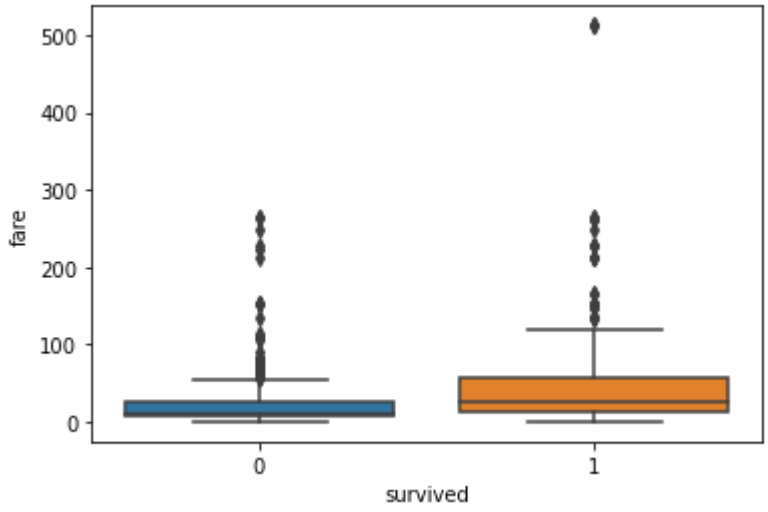


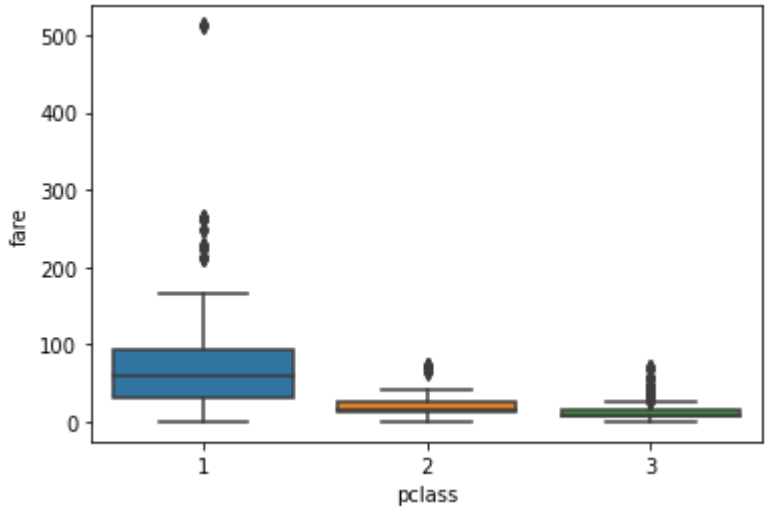


In [38]:

```
# Bi-Variate Analysis
sns.boxplot(x='survived', y='age', data=data)
plt.show()

sns.boxplot(x='survived', y='fare', data=data)
plt.show()

sns.boxplot(x='pclass', y='fare', data=data)
plt.show()
```

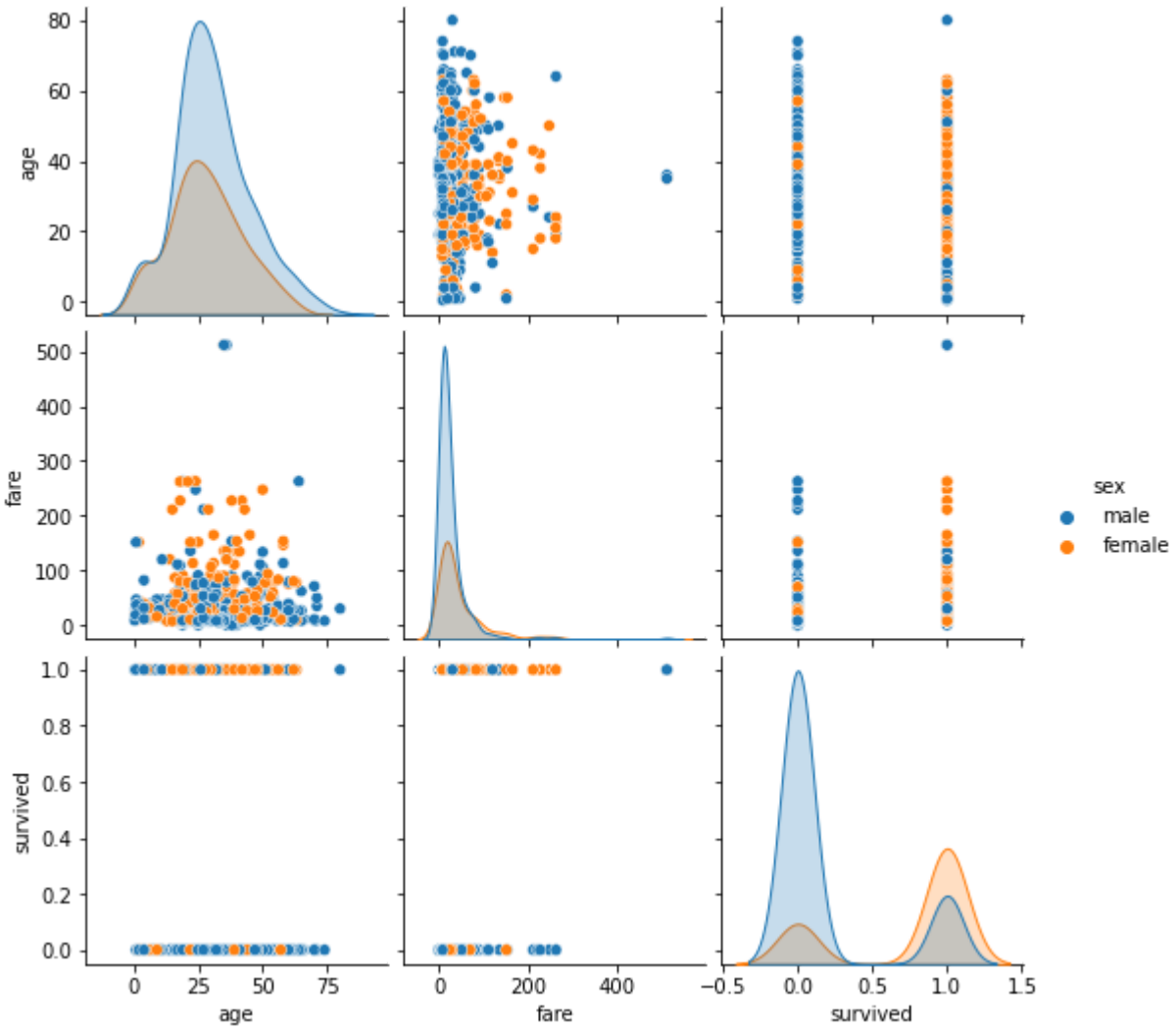






In [39]:

```
# Multi-Variate Analysis
sns.pairplot(data=data, vars=['age', 'fare', 'survived'], hue='sex')
plt.show()
```



In [40]:
```
# Descriptive Statistics
print(data.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
```

In [41]:
```
# Handle Missing Values
data['age'].fillna(data['age'].mean(), inplace=True)
data['embarked'].fillna(data['embarked'].mode()[0], inplace=True)
```

In [42]:
```
# Find and Replace Outliers
# You can use various methods to find and replace outliers, such as Z-score, IQR, or do
# Define a function to detect outliers based on Z-score
def find_replace_outliers_zscore(df, column):
    # Calculate Z-score for the column
    z_scores = (df[column] - df[column].mean()) / df[column].std()
```

```
    # Set a threshold for outliers (e.g., Z-score > 3 or Z-score < -3)
    threshold = 3
    # Find the indices of outliers
    outlier_indices = np.abs(z_scores) > threshold
    # Replace outliers with the median value of the column
    df.loc[outlier_indices, column] = df[column].median()

# Apply the function to each numeric column with outliers
numeric_columns = ['age', 'fare']
for column in numeric_columns:
    find_replace_outliers_zscore(data, column)

# Print the updated DataFrame with replaced outliers
print(data)
```

```
     survived  pclass     sex        age  sibsp  parch     fare embarked  \
0           0       3    male  22.000000      1      0   7.2500        S
1           1       1  female  38.000000      1      0  71.2833        C
2           1       3  female  26.000000      0      0   7.9250        S
3           1       1  female  35.000000      1      0  53.1000        S
4           0       3    male  35.000000      0      0   8.0500        S
..        ...     ...     ...        ...    ...    ...      ...      ...
886         0       2    male  27.000000      0      0  13.0000        S
887         1       1  female  19.000000      0      0  30.0000        S
888         0       3  female  29.699118      1      2  23.4500        S
889         1       1    male  26.000000      0      0  30.0000        C
890         0       3    male  32.000000      0      0   7.7500        Q

      class    who  adult_male deck  embark_town alive  alone
0     Third    man        True  NaN  Southampton    no  False
1     First  woman       False    C    Cherbourg   yes  False
2     Third  woman       False  NaN  Southampton   yes   True
3     First  woman       False    C  Southampton   yes  False
4     Third    man        True  NaN  Southampton    no   True
..      ...    ...         ...  ...          ...   ...    ...
886  Second    man        True  NaN  Southampton    no   True
887   First  woman       False    B  Southampton   yes   True
888   Third  woman       False  NaN  Southampton    no  False
889   First    man        True    C    Cherbourg   yes   True
890   Third    man        True  NaN   Queenstown    no   True

[891 rows x 15 columns]
```

In [49]:

```
# Check for Categorical Columns and Perform Encoding
encoder = LabelEncoder()
data['sex'] = encoder.fit_transform(data['sex'])
data['embarked'] = encoder.fit_transform(data['embarked'])
data['class'] = encoder.fit_transform(data['class'])
data['alone'] = encoder.fit_transform(data['alone'])
print(data.head())
```

```
   survived  pclass  sex   age  sibsp  parch     fare  embarked  class    who  \
0         0       3    1  22.0      1      0   7.2500         2      2    man
1         1       1    0  38.0      1      0  71.2833         0      0  woman
2         1       3    0  26.0      0      0   7.9250         2      2  woman
3         1       1    0  35.0      1      0  53.1000         2      0  woman
4         0       3    1  35.0      0      0   8.0500         2      2    man

   adult_male deck  embark_town alive  alone
```

```
0        True  NaN  Southampton    no      0
1       False    C    Cherbourg   yes      0
2       False  NaN  Southampton   yes      1
3       False    C  Southampton   yes      0
4        True  NaN  Southampton    no      1
```

In [44]:

```
# Split the Data into Dependent and Independent Variables
X = data.drop('survived', axis=1)
y = data['survived']
# Print the independent variables (X)
print("Independent Variables (X):")
print(X.head())

# Print the dependent variable (y)
print("\nDependent Variable (y):")
print(y.head())
```

```
Independent Variables (X):
   pclass  sex   age  sibsp  parch     fare  embarked  class    who  \
0       3    1  22.0      1      0   7.2500         2      2    man
1       1    0  38.0      1      0  71.2833         0      0  woman
2       3    0  26.0      0      0   7.9250         2      2  woman
3       1    0  35.0      1      0  53.1000         2      0  woman
4       3    1  35.0      0      0   8.0500         2      2    man

   adult_male deck  embark_town alive  alone
0        True  NaN  Southampton    no      0
1       False    C    Cherbourg   yes      0
2       False  NaN  Southampton   yes      1
3       False    C  Southampton   yes      0
4        True  NaN  Southampton    no      1

Dependent Variable (y):
0    0
1    1
2    1
3    1
4    0
Name: survived, dtype: int64
```

In [45]:

```
# Exclude non-numeric columns before scaling
numeric_cols = X.select_dtypes(include='number').columns
X_numeric = X[numeric_cols]

# Print the numeric columns of X
print("Numeric Columns of X:")
print(X_numeric.head())
```

```
Numeric Columns of X:
   pclass  sex   age  sibsp  parch     fare  embarked  class  alone
0       3    1  22.0      1      0   7.2500         2      2      0
1       1    0  38.0      1      0  71.2833         0      0      0
2       3    0  26.0      0      0   7.9250         2      2      1
3       1    0  35.0      1      0  53.1000         2      0      0
4       3    1  35.0      0      0   8.0500         2      2      1
```

In [46]:

```
# Scale the Independent Variables
scaler = StandardScaler()
```

```
X_scaled = scaler.fit_transform(X_numeric)

print("Scaled Independent Variables:")
print(X_scaled)
```

```
Scaled Independent Variables:
[[ 0.82737724  0.73769513 -0.59270449 ...  0.58595414  0.82737724
  -1.2316449 ]
 [-1.56610693 -1.35557354  0.69508685 ... -1.9423032  -1.56610693
  -1.2316449 ]
 [ 0.82737724 -1.35557354 -0.27075665 ...  0.58595414  0.82737724
   0.81192233]
 ...
 [ 0.82737724 -1.35557354  0.02697408 ...  0.58595414  0.82737724
  -1.2316449 ]
 [-1.56610693  0.73769513 -0.27075665 ... -1.9423032  -1.56610693
   0.81192233]
 [ 0.82737724  0.73769513  0.2121651  ... -0.67817453  0.82737724
   0.81192233]]
```

In [47]:

```
# Split the Data into Training and Testing Sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_
print("Training set shape:", X_train.shape, y_train.shape)
print("Testing set shape:", X_test.shape, y_test.shape)
```

```
Training set shape: (712, 9) (712,)
Testing set shape: (179, 9) (179,)
```