```
In [1]: import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
```

```
In [8]: df = pd.read_csv("train.csv")
        df.info()
        df.describe()
        df.isnull()
        df.sum()
        df.unique()
```

```
<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
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[8], line 5
      3 df.describe()
      4 df.isnull()
----> 5 df.sum()
      6 df.unique()

File E:\AC\anaconda\Lib\site-packages\pandas\core\frame.py:11670, in DataFrame.sum(self, axis, skipna, numeric_o
nly, min_count, **kwargs)
  11661 @doc(make_doc("sum", ndim=2))
  11662 def sum(
  11663     self,
   (...)
  11668     **kwargs,
  11669 ):
> 11670     result = super().sum(axis, skipna, numeric_only, min_count, **kwargs)
  11671     return result.__finalize__(self, method="sum")

File E:\AC\anaconda\Lib\site-packages\pandas\core\generic.py:12506, in NDFrame.sum(self, axis, skipna, numeric_o
nly, min_count, **kwargs)
  12498 def sum(
  12499     self,
  12500     axis: Axis | None = 0,
   (...)
  12504     **kwargs,
  12505 ):
> 12506     return self._min_count_stat_function(
  12507         "sum", nanops.nansum, axis, skipna, numeric_only, min_count, **kwargs
  12508     )

File E:\AC\anaconda\Lib\site-packages\pandas\core\generic.py:12489, in NDFrame._min_count_stat_function(self, na
me, func, axis, skipna, numeric_only, min_count, **kwargs)
  12486 elif axis is lib.no_default:
  12487     axis = 0
> 12489 return self._reduce(
  12490     func,
  12491     name=name,
  12492     axis=axis,
  12493     skipna=skipna,
  12494     numeric_only=numeric_only,
  12495     min_count=min_count,
  12496 )

File E:\AC\anaconda\Lib\site-packages\pandas\core\frame.py:11562, in DataFrame._reduce(self, op, name, axis, ski
pna, numeric_only, filter_type, **kwds)
  11558     df = df.T
  11560 # After possibly _get_data and transposing, we are now in the
  11561 #  simple case where we can use BlockManager.reduce
> 11562 res = df._mgr.reduce(blk_func)
  11563 out = df._constructor_from_mgr(res, axes=res.axes).iloc[0]
  11564 if out_dtype is not None and out.dtype != "boolean":

File E:\AC\anaconda\Lib\site-packages\pandas\core\internals\managers.py:1500, in BlockManager.reduce(self, func)
   1498 res_blocks: list[Block] = []
   1499 for blk in self.blocks:
-> 1500     nbs = blk.reduce(func)
   1501     res_blocks.extend(nbs)
   1503 index = Index([None])  # placeholder

File E:\AC\anaconda\Lib\site-packages\pandas\core\internals\blocks.py:404, in Block.reduce(self, func)
    398 @final
    399 def reduce(self, func) -> list[Block]:
    400     # We will apply the function and reshape the result into a single-row
    401     #  Block with the same mgr_locs; squeezing will be done at a higher level
    402     assert self.ndim == 2
--> 404     result = func(self.values)
    406     if self.values.ndim == 1:
    407         res_values = result

File E:\AC\anaconda\Lib\site-packages\pandas\core\frame.py:11481, in DataFrame._reduce.<locals>.blk_func(values, axis)
  11479         return np.array([result])
  11480 else:
> 11481     return op(values, axis=axis, skipna=skipna, **kwds)

File E:\AC\anaconda\Lib\site-packages\pandas\core\nanops.py:85, in disallow.__call__.<locals>._f(*args, **kwargs)
     81     raise TypeError(
     82         f"reduction operation '{f_name}' not allowed for this dtype"
     83     )
     84 try:
---> 85     return f(*args, **kwargs)
     86 except ValueError as e:
     87     # we want to transform an object array
     88     # ValueError message to the more typical TypeError
     89     # e.g. this is normally a disallowed function on
     90     # object arrays that contain strings
     91     if is_object_dtype(args[0]):

File E:\AC\anaconda\Lib\site-packages\pandas\core\nanops.py:404, in _datetimelike_compat.<locals>.new_func(values, axis, skipna, mask, **kwargs)
    401 if datetimelike and mask is None:
    402     mask = isna(values)
--> 404 result = func(values, axis=axis, skipna=skipna, mask=mask, **kwargs)
    406 if datetimelike:
    407     result = _wrap_results(result, orig_values.dtype, fill_value=iNaT)

File E:\AC\anaconda\Lib\site-packages\pandas\core\nanops.py:477, in maybe_operate_rowwise.<locals>.newfunc(values, axis, **kwargs)
    474         results = [func(x, **kwargs) for x in arrs]
    475         return np.array(results)
--> 477 return func(values, axis=axis, **kwargs)

File E:\AC\anaconda\Lib\site-packages\pandas\core\nanops.py:646, in nansum(values, axis, skipna, min_count, mask)
    643 elif dtype.kind == "m":
    644     dtype_sum = np.dtype(np.float64)
--> 646 the_sum = values.sum(axis, dtype=dtype_sum)
    647 the_sum = _maybe_null_out(the_sum, axis, mask, values.shape, min_count=min_count)
    649 return the_sum

File E:\AC\anaconda\Lib\site-packages\numpy\core\_methods.py:49, in _sum(a, axis, dtype, out, keepdims, initial, where)
     47 def _sum(a, axis=None, dtype=None, out=None, keepdims=False,
     48          initial=_NoValue, where=True):
---> 49     return umr_sum(a, axis, dtype, out, keepdims, initial, where)

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

In [ ]:

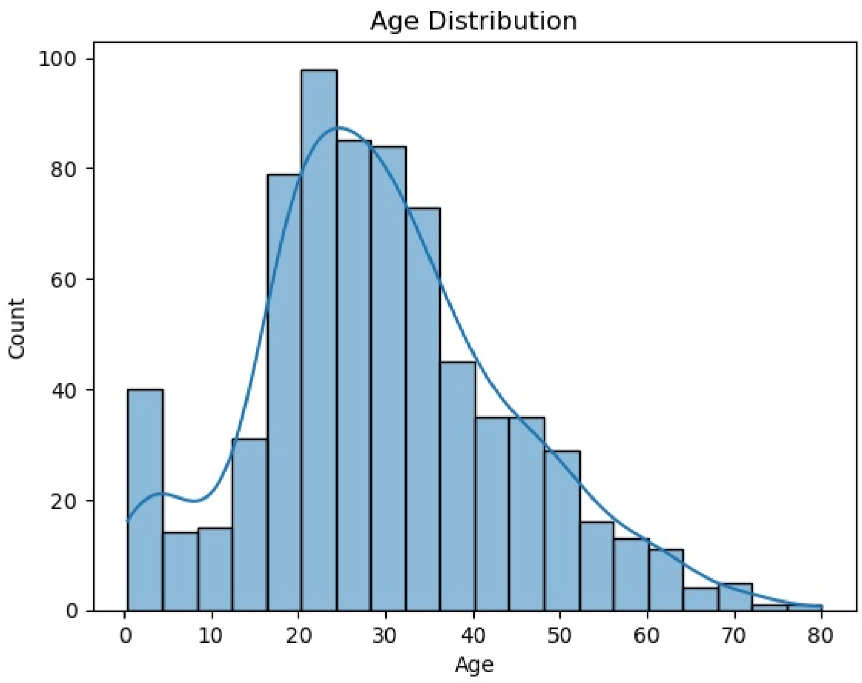
```
print(df['Survived'].value_counts())
print(df['Sex'].value_counts())
print(df['Pclass'].value_counts())
print(df['Embarked'].value_counts())
```

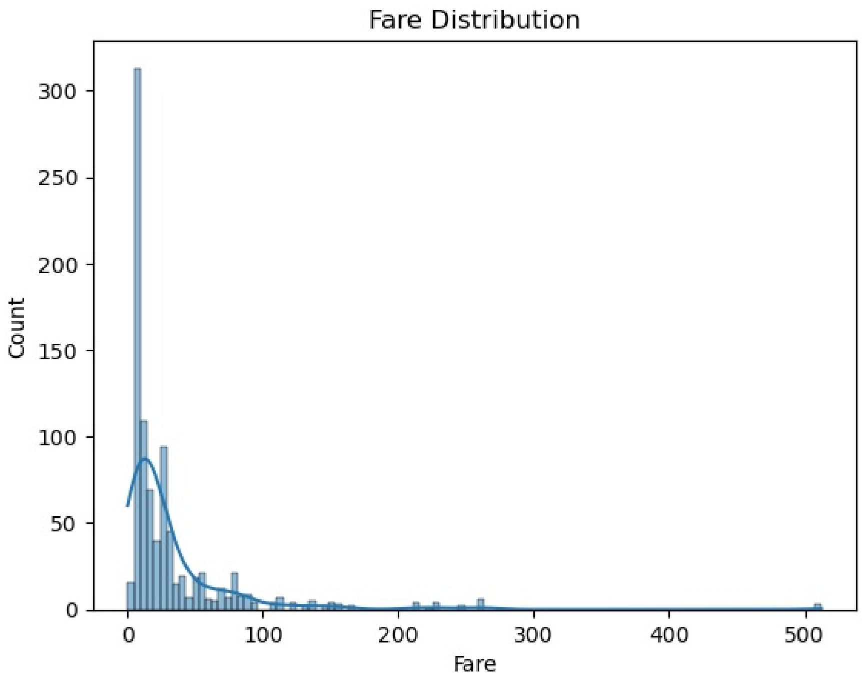
In [5]:

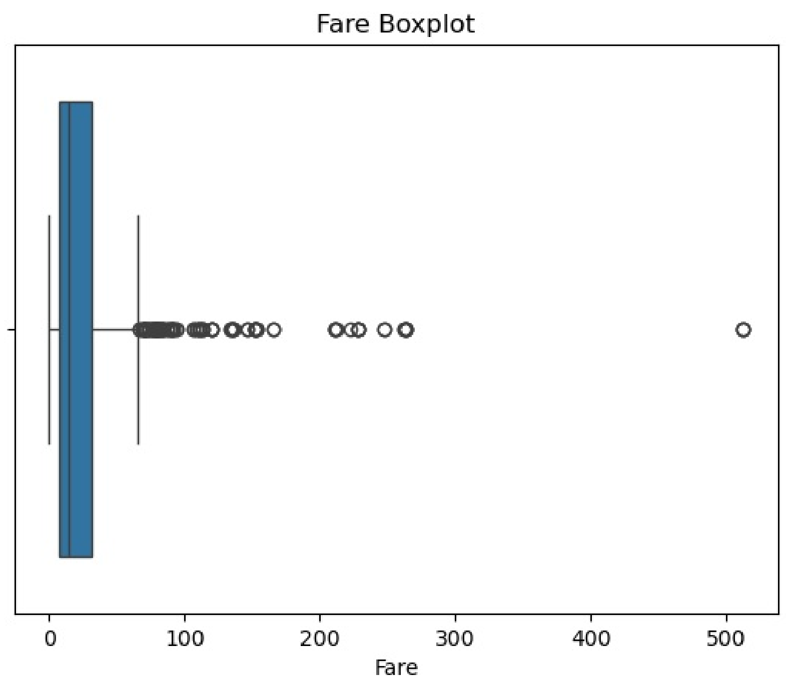
```
sns.histplot(df['Age'].dropna(), kde=True)
plt.title("Age Distribution")
plt.show()

sns.histplot(df['Fare'], kde=True)
plt.title("Fare Distribution")
plt.show()
```

```
sns.boxplot(x=df['Fare'])
plt.title("Fare Boxplot")
plt.show()
```

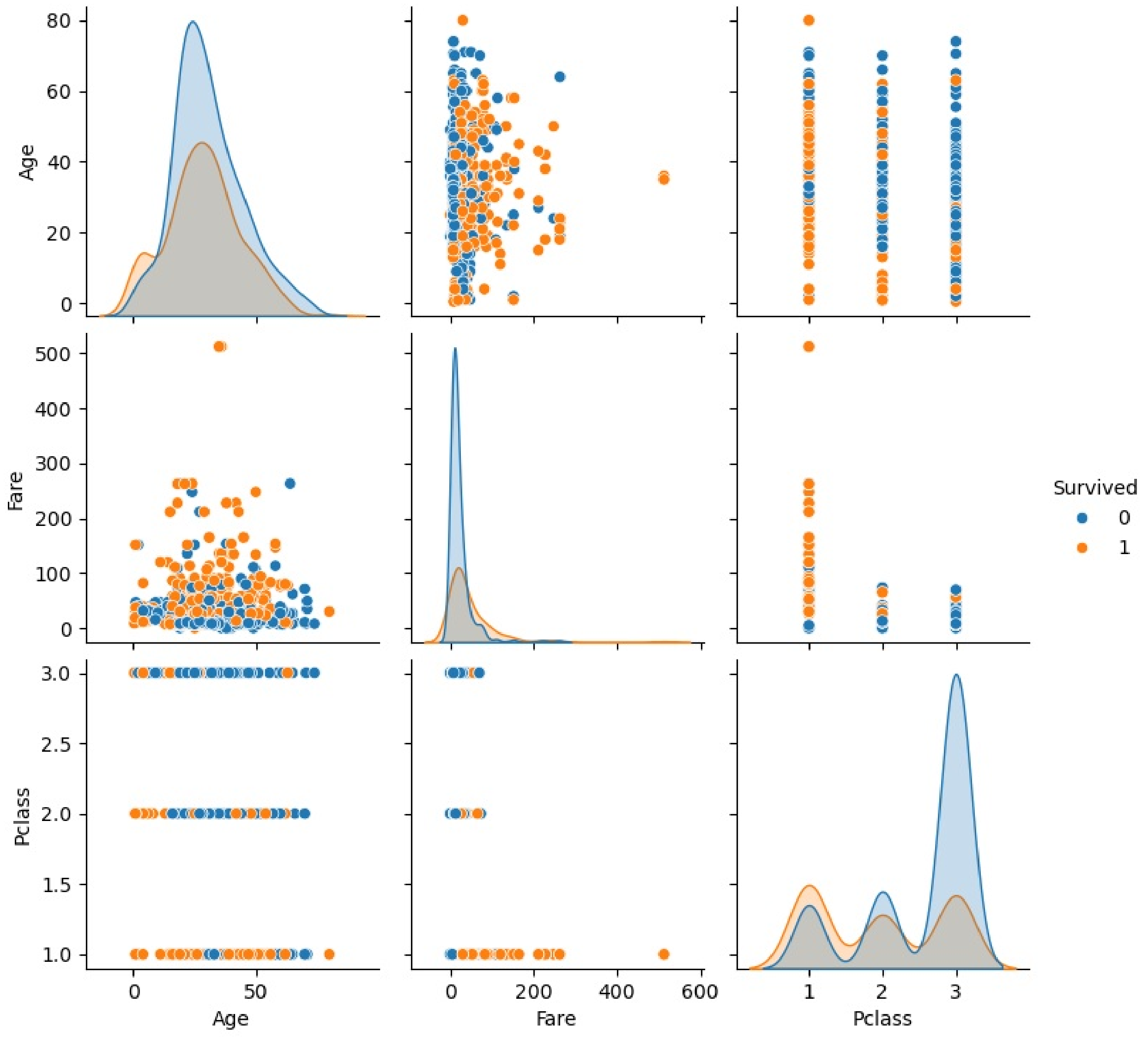


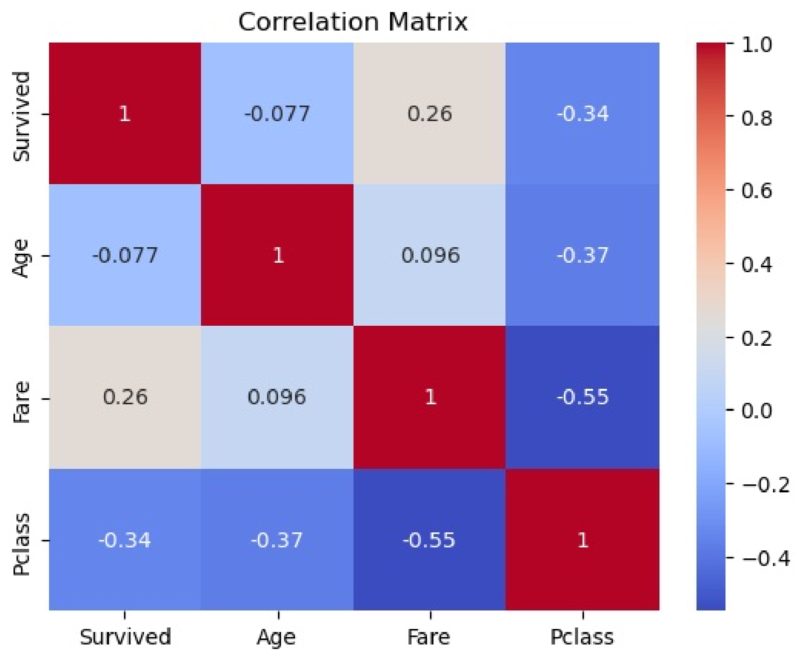




```
In [6]: sns.pairplot(df[['Survived', 'Age', 'Fare', 'Pclass']], hue='Survived')
        plt.show()


        corr = df[['Survived', 'Age', 'Fare', 'Pclass']].corr()
        sns.heatmap(corr, annot=True, cmap='coolwarm')
        plt.title("Correlation Matrix")
        plt.show()
```



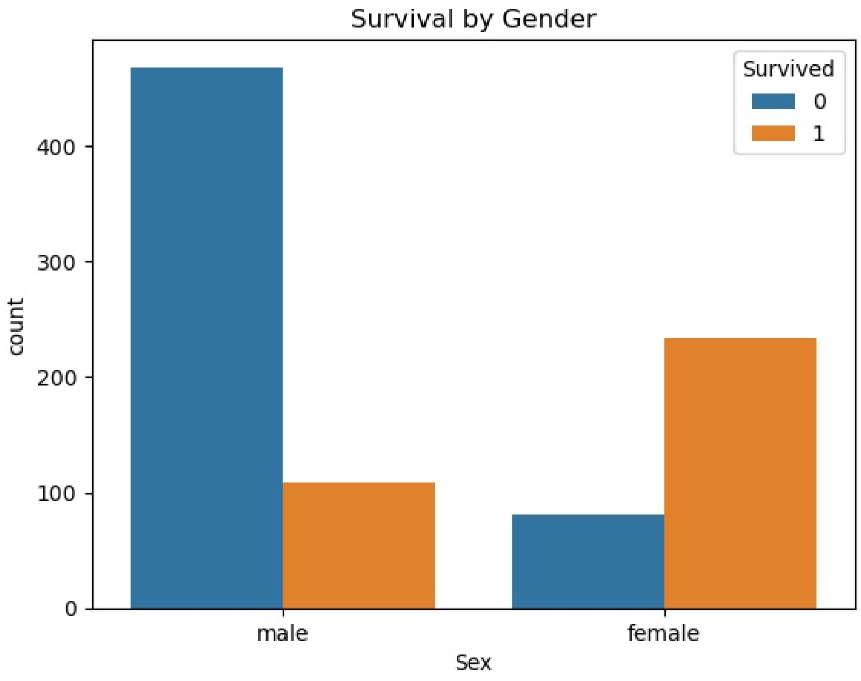


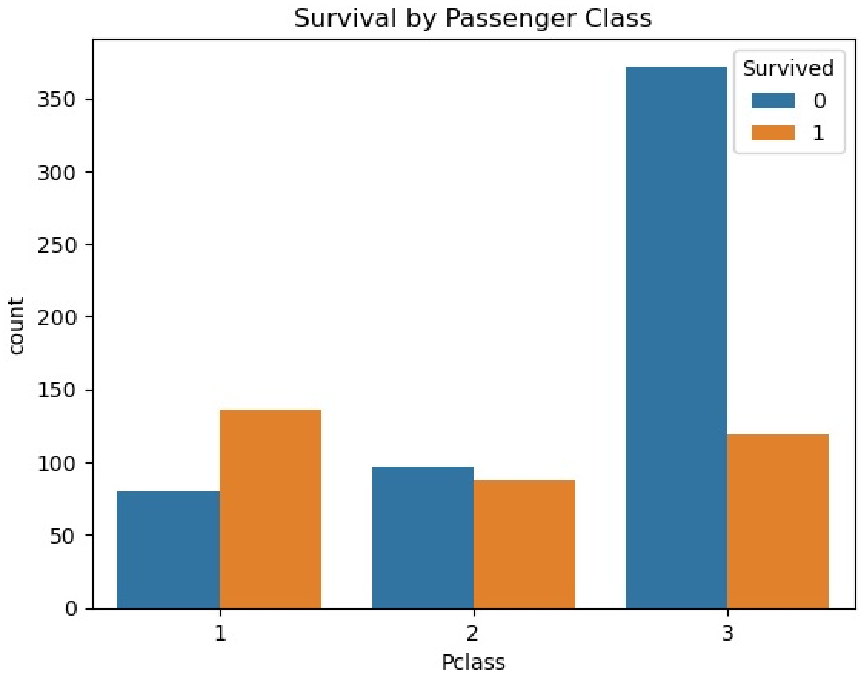
	Survived	Age	Fare	Pclass
Survived	1	-0.077	0.26	-0.34
Age	-0.077	1	0.096	-0.37
Fare	0.26	0.096	1	-0.55
Pclass	-0.34	-0.37	-0.55	1

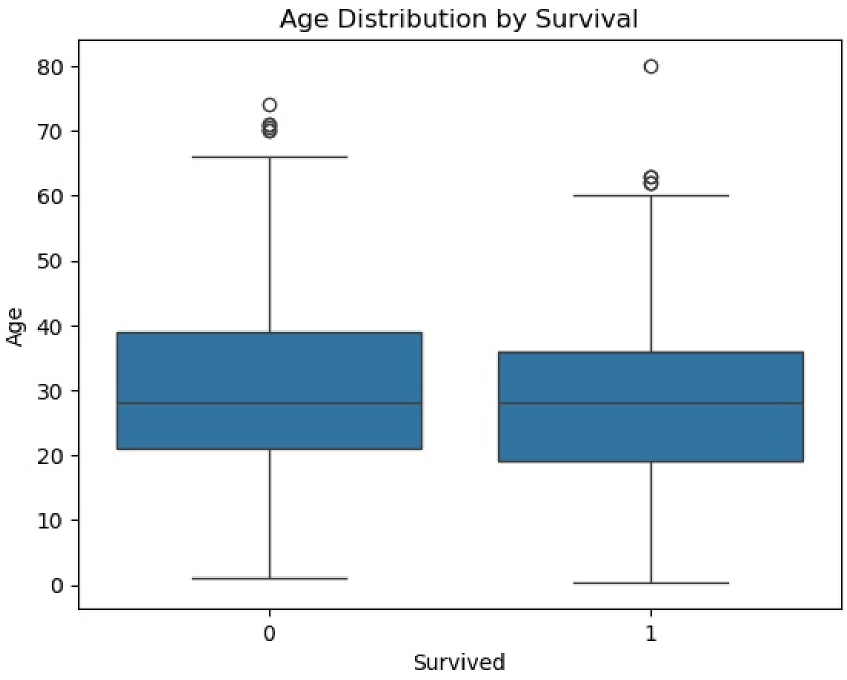
In [7]:
```
sns.countplot(x='Sex', hue='Survived', data=df)
plt.title("Survival by Gender")
plt.show()


sns.countplot(x='Pclass', hue='Survived', data=df)
plt.title("Survival by Passenger Class")
plt.show()


sns.boxplot(x='Survived', y='Age', data=df)
plt.title("Age Distribution by Survival")
plt.show()
```


Survival by Gender
Survived
0
1
count
400
300
200
100
0
male
female
Sex


Survival by Passenger Class
Survived
0
1
count
350
300
250
200
150
100
50
0
1
2
3
Pclass


Age Distribution by Survival
Age
80
70
60
50
40
30
20
10
0
0
1
Survived

```
In [ ]:
```

In [ ]:

```

```

In [ ]:

```

```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js