```
import seaborn as sns

# Load Titanic dataset into a variable called 'titanic'
titanic = sns.load_dataset('titanic')

# Check if it loaded by showing the first 5 rows
print(titanic.head())
```

```
   survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
0         0       3    male  22.0      1      0   7.2500        S
Third
1         1       1  female  38.0      1      0  71.2833        C
First
2         1       3  female  26.0      0      0   7.9250        S
Third
3         1       1  female  35.0      1      0  53.1000        S
First
4         0       3    male  35.0      0      0   8.0500        S
Third

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

```
import pandas as pd

# Replace 'titanic.csv' with your file path if different
titanic = pd.read_csv('titanic.csv')

# Show first 5 rows to check
print(titanic.head())
```

```
---------------------------------------------------------------------------
------
FileNotFoundError                         Traceback (most recent call
last)
Cell In[2], line 4
      1 import pandas as pd
      3 # Replace 'titanic.csv' with your file path if different
----> 4 titanic = pd.read_csv('titanic.csv')
      6 # Show first 5 rows to check
      7 print(titanic.head())

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:1026,
in read_csv(filepath_or_buffer, sep, delimiter, header, names,
index_col, usecols, dtype, engine, converters, true_values,
false_values, skipinitialspace, skiprows, skipfooter, nrows,
```

```
na_values, keep_default_na, na_filter, verbose, skip_blank_lines,
parse_dates, infer_datetime_format, keep_date_col, date_parser,
date_format, dayfirst, cache_dates, iterator, chunksize, compression,
thousands, decimal, lineterminator, quotechar, quoting, doublequote,
escapechar, comment, encoding, encoding_errors, dialect, on_bad_lines,
delim_whitespace, low_memory, memory_map, float_precision,
storage_options, dtype_backend)
   1013 kwds_defaults = _refine_defaults_read(
   1014     dialect,
   1015     delimiter,
   (...)
   1022     dtype_backend=dtype_backend,
   1023 )
   1024 kwds.update(kwds_defaults)
-> 1026 return _read(filepath_or_buffer, kwds)

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:620,
in _read(filepath_or_buffer, kwds)
    617 _validate_names(kwds.get("names", None))
    619 # Create the parser.
--> 620 parser = TextFileReader(filepath_or_buffer, **kwds)
    622 if chunksize or iterator:
    623     return parser

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:1620,
in TextFileReader.__init__(self, f, engine, **kwds)
   1617     self.options["has_index_names"] = kwds["has_index_names"]
   1619 self.handles: IOHandles | None = None
-> 1620 self._engine = self._make_engine(f, self.engine)

File ~\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:1880,
in TextFileReader._make_engine(self, f, engine)
   1878     if "b" not in mode:
   1879         mode += "b"
-> 1880 self.handles = get_handle(
   1881     f,
   1882     mode,
   1883     encoding=self.options.get("encoding", None),
   1884     compression=self.options.get("compression", None),
   1885     memory_map=self.options.get("memory_map", False),
   1886     is_text=is_text,
   1887     errors=self.options.get("encoding_errors", "strict"),
   1888     storage_options=self.options.get("storage_options", None),
   1889 )
   1890 assert self.handles is not None
   1891 f = self.handles.handle

File ~\anaconda3\Lib\site-packages\pandas\io\common.py:873, in
get_handle(path_or_buf, mode, encoding, compression, memory_map,
is_text, errors, storage_options)
```

```
    868 elif isinstance(handle, str):
    869     # Check whether the filename is to be opened in binary
mode.
    870     # Binary mode does not support 'encoding' and 'newline'.
    871     if ioargs.encoding and "b" not in ioargs.mode:
    872         # Encoding
--> 873         handle = open(
    874             handle,
    875             ioargs.mode,
    876             encoding=ioargs.encoding,
    877             errors=errors,
    878             newline="",
    879         )
    880     else:
    881         # Binary mode
    882         handle = open(handle, ioargs.mode)

FileNotFoundError: [Errno 2] No such file or directory: 'titanic.csv'
```

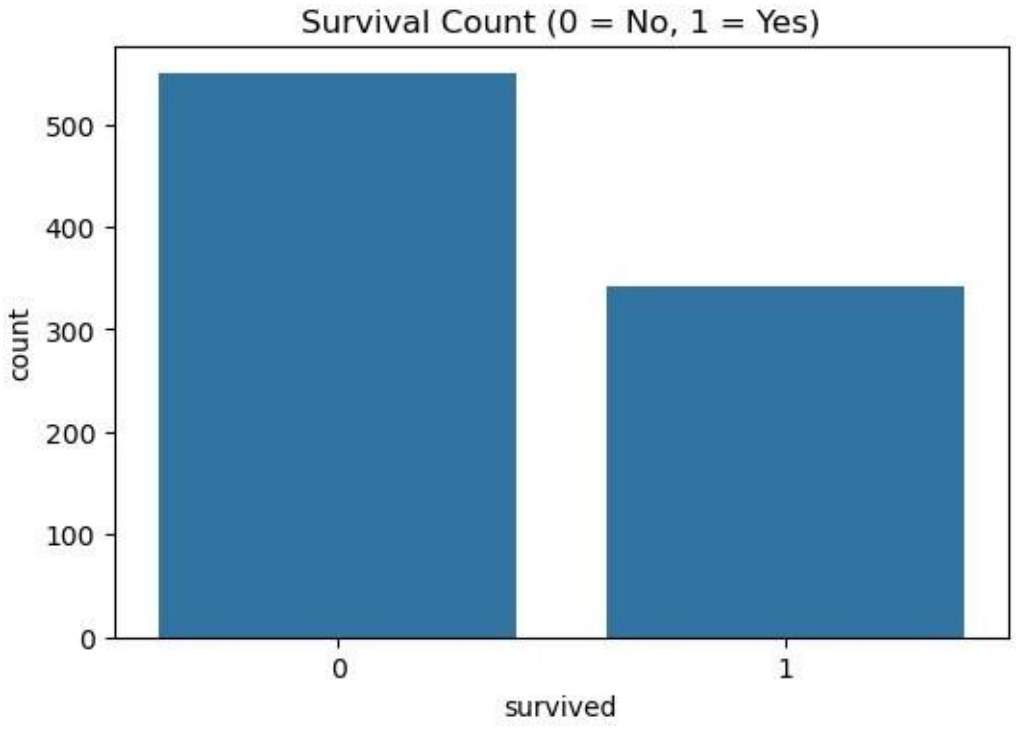
```
import seaborn as sns

# Load Titanic dataset from seaborn's built-in datasets
titanic = sns.load_dataset('titanic')

# Show first 5 rows to confirm loading
print(titanic.head())
```

```
   survived  pclass     sex   age  sibsp  parch     fare embarked
class  \
0         0       3    male  22.0      1      0   7.2500        S
Third
1         1       1  female  38.0      1      0  71.2833        C
First
2         1       3  female  26.0      0      0   7.9250        S
Third
3         1       1  female  35.0      1      0  53.1000        S
First
4         0       3    male  35.0      0      0   8.0500        S
Third

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

```
# Show info about dataset: columns, non-null counts, data types
titanic.info()
```

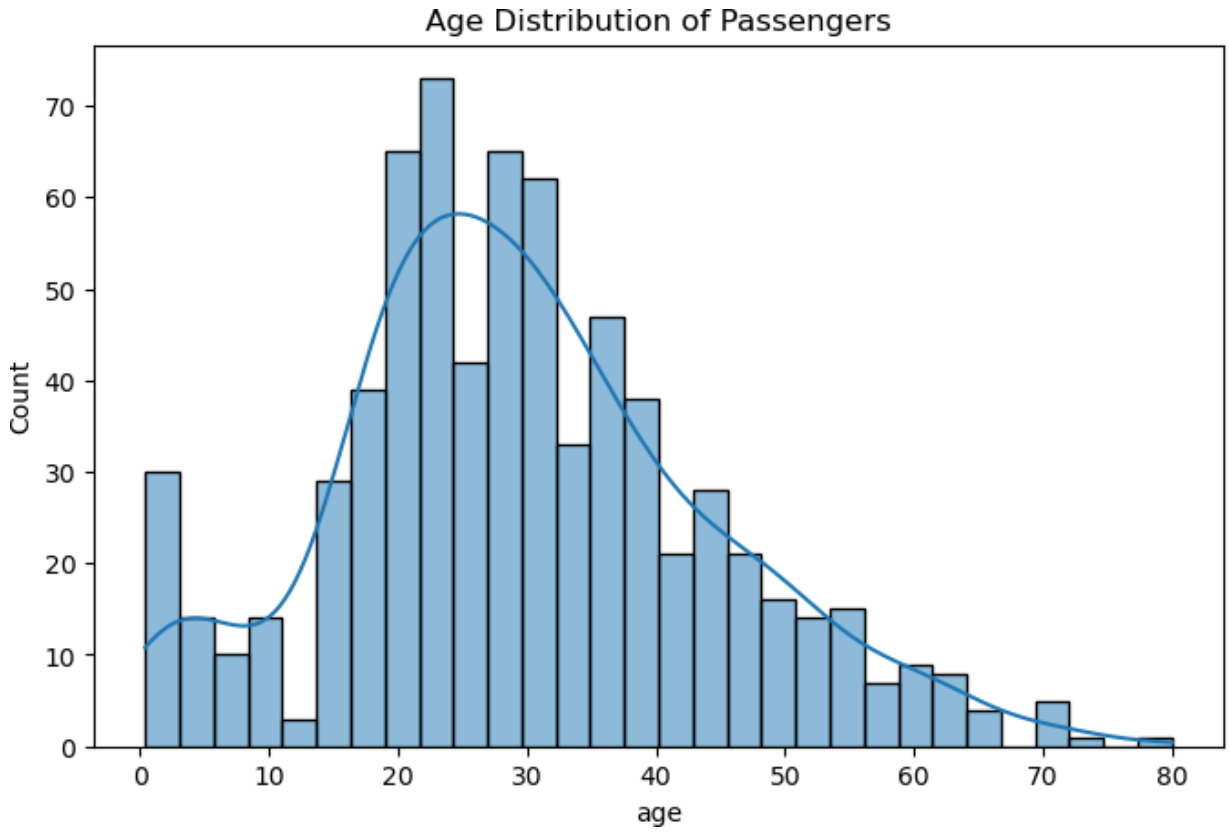
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype   
---  ------       --------------  -----   
 0   survived     891 non-null    int64   
 1   pclass       891 non-null    int64   
 2   sex          891 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    category
 9   who          891 non-null    object  
 10  adult_male   891 non-null    bool    
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object  
 13  alive        891 non-null    object  
 14  alone        891 non-null    bool    
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB

# Get summary statistics of numeric columns
titanic.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

import matplotlib.pyplot as plt
import seaborn as sns

# 1. Survival count plot
plt.figure(figsize=(6,4))
```

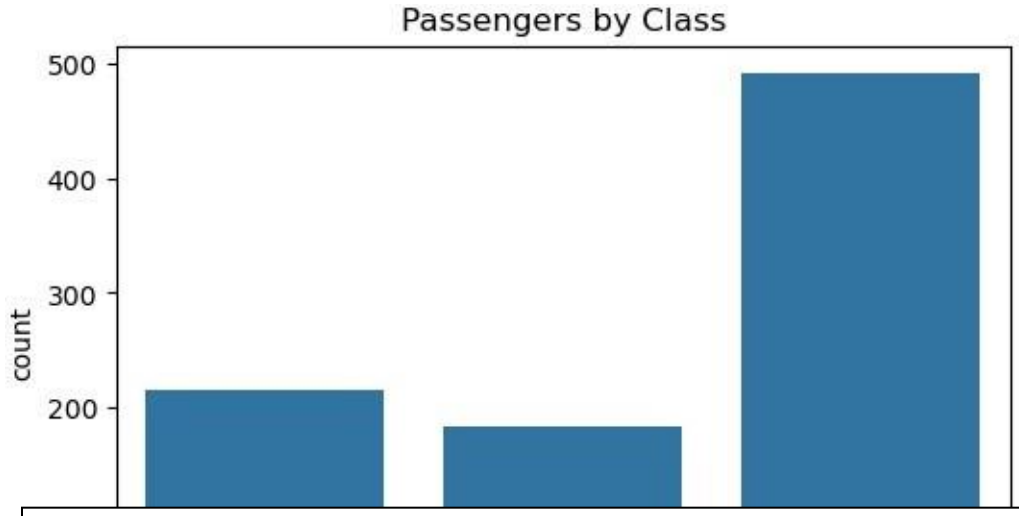
```
sns.countplot(x='survived', data=titanic)
plt.title('Survival Count (0 = No, 1 = Yes)')
plt.show()

# 2. Age distribution histogram
plt.figure(figsize=(8,5))
sns.histplot(titanic['age'].dropna(), bins=30, kde=True)
plt.title('Age Distribution of Passengers')
plt.show()

# 3. Passenger count by class
plt.figure(figsize=(6,4))
sns.countplot(x='class', data=titanic)
plt.title('Passengers by Class')
plt.show()
```



Visualization 1: Survival Count Plot
Observation:
The plot shows that more passengers did not survive (coded as 0) compared to those who survived (coded as 1). This indicates the high fatality rate of the Titanic disaster .

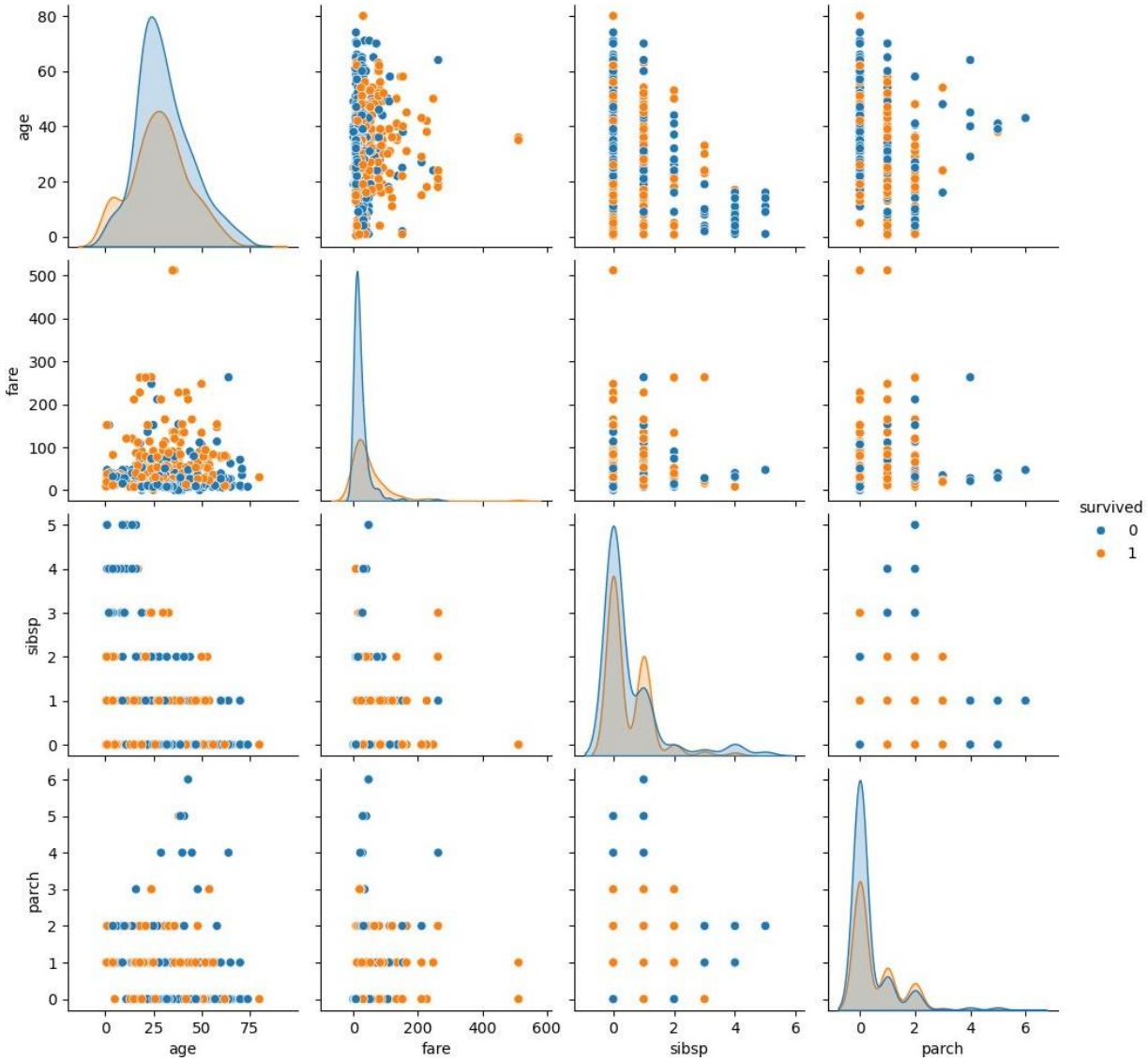
Visualization 2: Age Distribution Histogram
Observation:
Most passengers were between 20 to 40 years old, with fewer children and elderly passengers onboard.





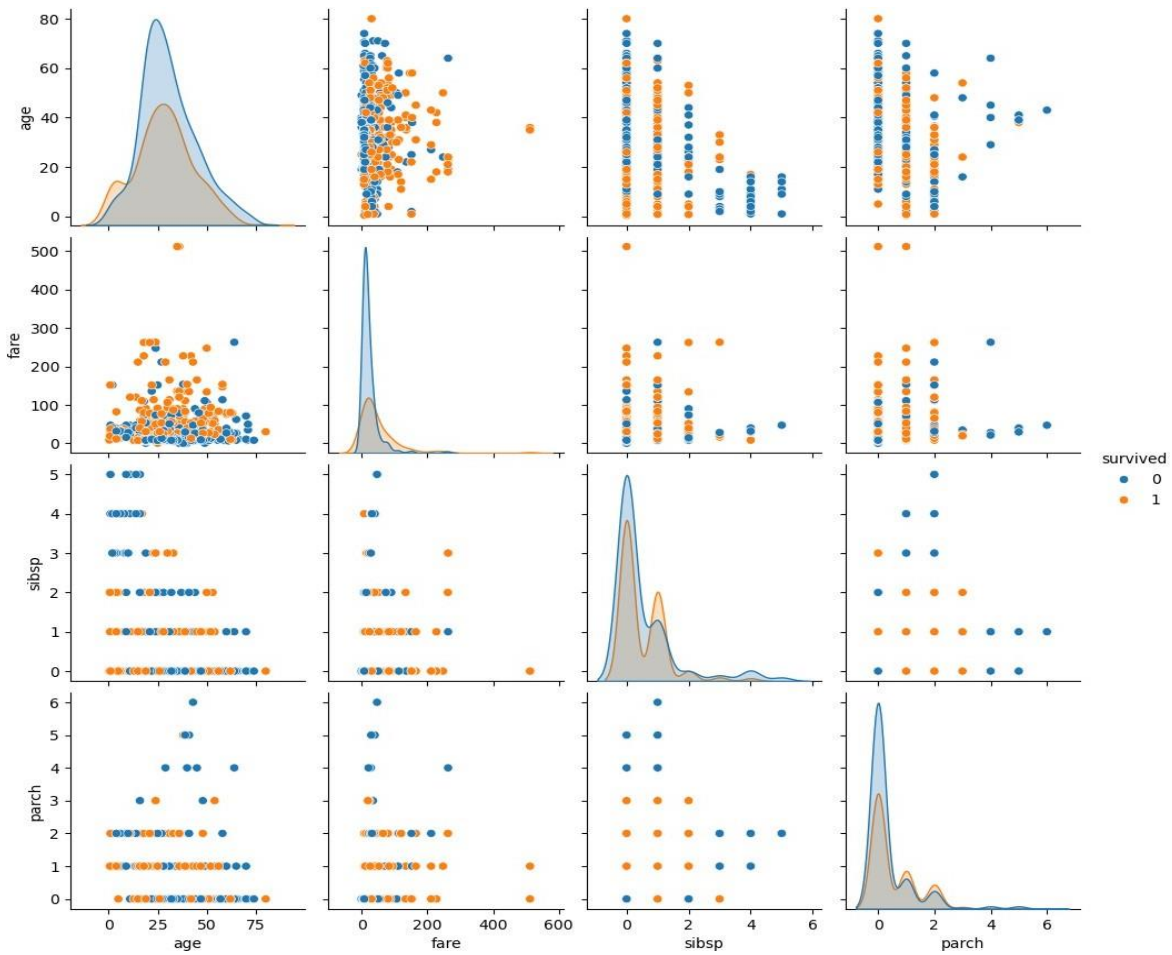
Visualization 3: Passenger Count by Class
Observation:
The majority of passengers were in 3rd class, followed by 1st and 2nd class. This distribution may affect survival rates due to class-based differences in safety measures.

```
# Pairplot colored by survival
sns.pairplot(titanic.dropna(subset=['age', 'fare', 'sibsp', 'parch']),

             vars=['age', 'fare', 'sibsp', 'parch'],
             hue='survived')
plt.show()

# Correlation heatmap of numeric columns
plt.figure(figsize=(8,6))
sns.heatmap(titanic.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



Visualization 4: Pairplot of Numeric Features
Observation:
Higher fares are associated with survivors. Age shows some variation, but no strong trend. Number of siblings/spouses and parents/children show limited correlation with survival

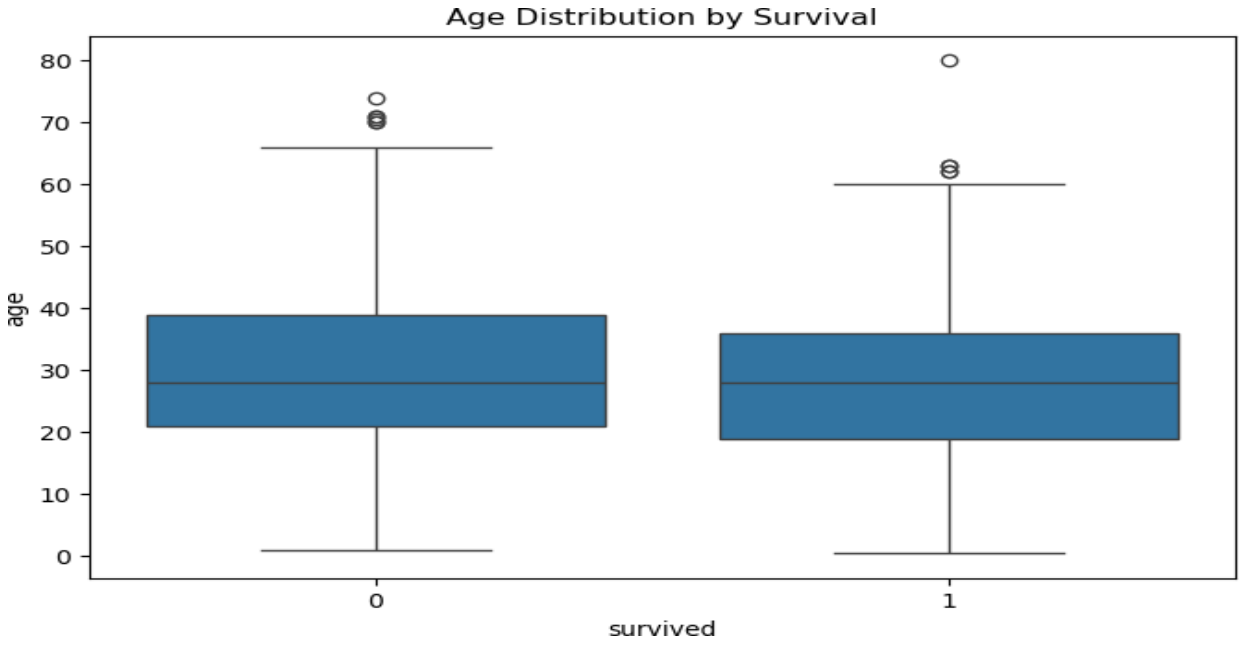
```
sns.pairplot(titanic.dropna(subset=['age', 'fare', 'sibsp', 'parch']),
             vars=['age', 'fare', 'sibsp', 'parch'],
             hue='survived')
plt.show()
```



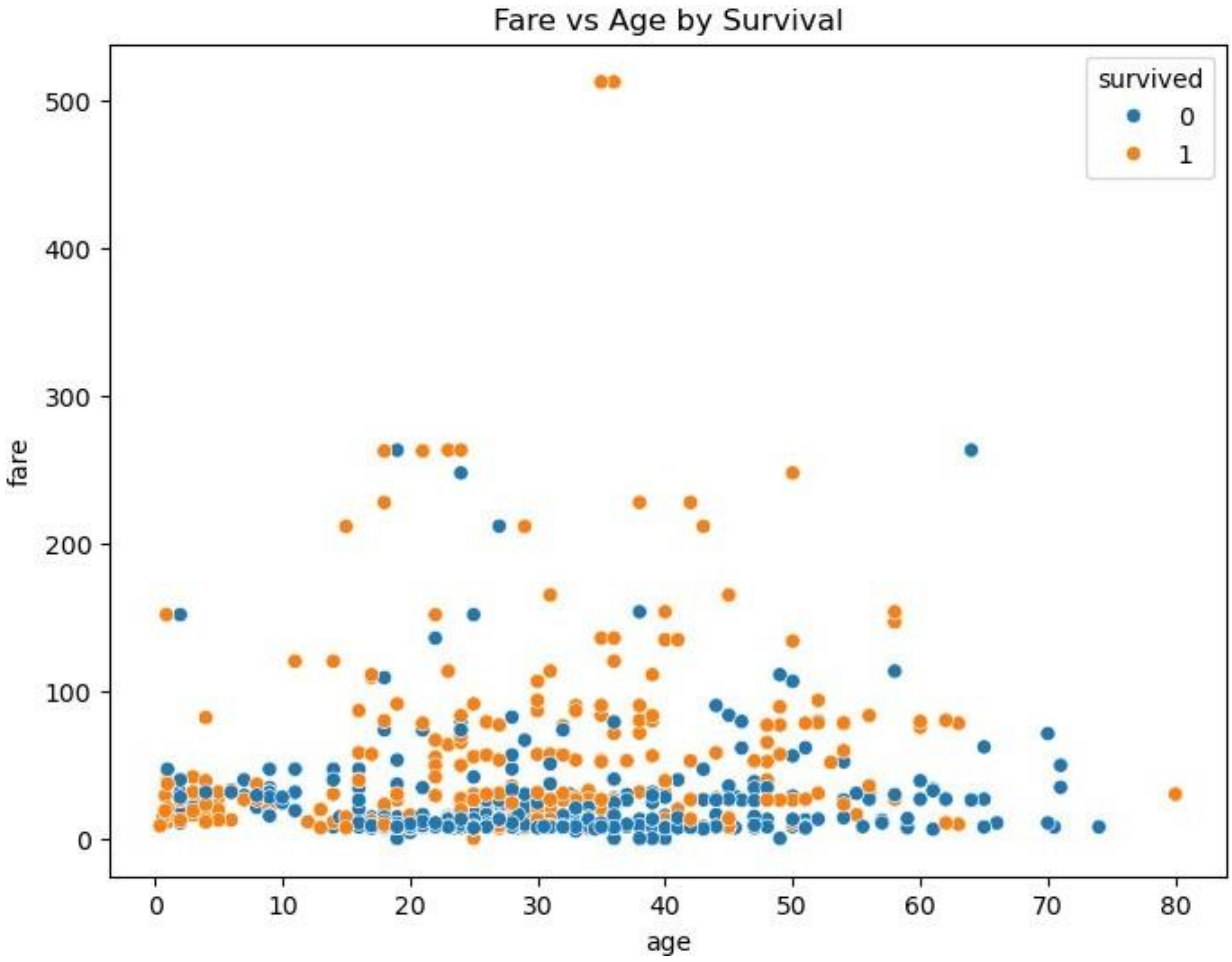
Visualization 5: Correlation Heatmap
Observation:
Fare and survival show a positive correlation, meaning higher fares relate to higher chances of survival. Other numeric features have weaker or no clear correlation with survival.

```
# Boxplot of Age vs Survival
plt.figure(figsize=(8,5))
sns.boxplot(x='survived', y='age', data=titanic)
plt.title('Age Distribution by Survival')
plt.show()

# Scatterplot of Fare vs Age colored by Survival
plt.figure(figsize=(8,6))
```

```
sns.scatterplot(x='age', y='fare', hue='survived', data=titanic)
plt.title('Fare vs Age by Survival')
plt.show()
```



Visualization 6: Boxplot of Age by Survival
Observation:
Survivors tend to be younger, with the median age lower than non-survivors. Older passengers had a higher mortality rate.



Visualization 7: Scatterplot of Fare vs Age by Survival
Observation:
Passengers who paid higher fares and were younger tended to survive more. This supports the link between socioeconomic status and survival.