DATA CLEANING AND EXPLORATORY DATA ANALYSIS

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

from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving Titanic.csv to Titanic.csv
```

```
import pandas as pd

df = pd.read_csv("Titanic.csv")  # Change filename if needed
df.head()  # Display the first 5 rows
```

```
---------------------------------------------------------------------------
-----
UnicodeDecodeError                        Traceback (most recent call
last)
<ipython-input-3-b6657980fa9d> in <cell line: 0>()
      1 import pandas as pd
      2
----> 3 df = pd.read_csv("Titanic.csv")  # Change filename if needed
      4 df.head()  # Display the first 5 rows

/usr/local/lib/python3.11/dist-packages/pandas/io/parsers/readers.py
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)
   1024     kwds.update(kwds_defaults)
   1025
-> 1026     return _read(filepath_or_buffer, kwds)
   1027
   1028

/usr/local/lib/python3.11/dist-packages/pandas/io/parsers/readers.py
in _read(filepath_or_buffer, kwds)
    618
    619     # Create the parser.
```

```
--> 620     parser = TextFileReader(filepath_or_buffer, **kwds)
    621
    622     if chunksize or iterator:

/usr/local/lib/python3.11/dist-packages/pandas/io/parsers/readers.py
in __init__(self, f, engine, **kwds)
   1618
   1619         self.handles: IOHandles | None = None
-> 1620         self._engine = self._make_engine(f, self.engine)
   1621
   1622     def close(self) -> None:

/usr/local/lib/python3.11/dist-packages/pandas/io/parsers/readers.py
in _make_engine(self, f, engine)
   1896
   1897         try:
-> 1898             return mapping[engine](f, **self.options)
   1899         except Exception:
   1900             if self.handles is not None:

/usr/local/lib/python3.11/dist-packages/pandas/io/parsers/c_parser_wra
pper.py in __init__(self, src, **kwds)
     91             # Fail here loudly instead of in cython after
reading
     92             import_optional_dependency("pyarrow")
---> 93         self._reader = parsers.TextReader(src, **kwds)
     94
     95         self.unnamed_cols = self._reader.unnamed_cols

parsers.pyx in pandas._libs.parsers.TextReader.__cinit__()

parsers.pyx in pandas._libs.parsers.TextReader._get_header()

parsers.pyx in pandas._libs.parsers.TextReader._tokenize_rows()

parsers.pyx in
pandas._libs.parsers.TextReader._check_tokenize_status()

parsers.pyx in pandas._libs.parsers.raise_parser_error()

UnicodeDecodeError: 'utf-8' codec can't decode byte 0xff in position
37302: invalid start byte

df = pd.read_csv("Titanic.csv", encoding="ISO-8859-1")
df.head()

---------------------------------------------------------------------
-----
ParserError                               Traceback (most recent call
last)
<ipython-input-4-76b39af46e6b> in <cell line: 0>()
```

```
----> 1 df = pd.read_csv("Titanic.csv", encoding="ISO-8859-1")
      2 df.head()

/usr/local/lib/python3.11/dist-packages/pandas/io/parsers/readers.py
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)
   1024     kwds.update(kwds_defaults)
   1025
-> 1026     return _read(filepath_or_buffer, kwds)
   1027
   1028

/usr/local/lib/python3.11/dist-packages/pandas/io/parsers/readers.py
in _read(filepath_or_buffer, kwds)
    624
    625     with parser:
--> 626         return parser.read(nrows)
    627
    628

/usr/local/lib/python3.11/dist-packages/pandas/io/parsers/readers.py
in read(self, nrows)
   1921                     columns,
   1922                     col_dict,
-> 1923                 ) = self._engine.read(  # type: ignore[attr-
defined]
   1924                     nrows
   1925                 )

/usr/local/lib/python3.11/dist-packages/pandas/io/parsers/c_parser_wra
pper.py in read(self, nrows)
    232         try:
    233             if self.low_memory:
--> 234                 chunks = self._reader.read_low_memory(nrows)
    235                 # destructive to chunks
    236                 data = _concatenate_chunks(chunks)

parsers.pyx in pandas._libs.parsers.TextReader.read_low_memory()

parsers.pyx in pandas._libs.parsers.TextReader._read_rows()

parsers.pyx in pandas._libs.parsers.TextReader._tokenize_rows()
```

```
parsers.pyx in
pandas._libs.parsers.TextReader._check_tokenize_status()

parsers.pyx in pandas._libs.parsers.raise_parser_error()

ParserError: Error tokenizing data. C error: Expected 1 fields in line
4, saw 2
```

```
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>

Saving Titanic.xlsx.csv to Titanic.xlsx.csv
```

```
import pandas as pd

df = pd.read_csv("Titanic.xlsx.csv")  # Change filename if needed
df.head()  # Display the first 5 rows
```

```
---------------------------------------------------------------------------
-----
UnicodeDecodeError                        Traceback (most recent call
last)
<ipython-input-6-54b2744628e4> in <cell line: 0>()
      1 import pandas as pd
      2
----> 3 df = pd.read_csv("Titanic.xlsx.csv")  # Change filename if
needed
      4 df.head()  # Display the first 5 rows

/usr/local/lib/python3.11/dist-packages/pandas/io/parsers/readers.py
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)
   1024     kwds.update(kwds_defaults)
   1025
-> 1026     return _read(filepath_or_buffer, kwds)
   1027
   1028

/usr/local/lib/python3.11/dist-packages/pandas/io/parsers/readers.py
```

```
in _read(filepath_or_buffer, kwds)
    618
    619     # Create the parser.
--> 620     parser = TextFileReader(filepath_or_buffer, **kwds)
    621
    622     if chunksize or iterator:

/usr/local/lib/python3.11/dist-packages/pandas/io/parsers/readers.py
in __init__(self, f, engine, **kwds)
   1618
   1619         self.handles: IOHandles | None = None
-> 1620         self._engine = self._make_engine(f, self.engine)
   1621
   1622     def close(self) -> None:

/usr/local/lib/python3.11/dist-packages/pandas/io/parsers/readers.py
in _make_engine(self, f, engine)
   1896
   1897         try:
-> 1898             return mapping[engine](f, **self.options)
   1899         except Exception:
   1900             if self.handles is not None:

/usr/local/lib/python3.11/dist-packages/pandas/io/parsers/c_parser_wra
pper.py in __init__(self, src, **kwds)
     91             # Fail here loudly instead of in cython after
reading
     92             import_optional_dependency("pyarrow")
---> 93         self._reader = parsers.TextReader(src, **kwds)
     94
     95         self.unnamed_cols = self._reader.unnamed_cols

parsers.pyx in pandas._libs.parsers.TextReader.__cinit__()

parsers.pyx in pandas._libs.parsers.TextReader._get_header()

parsers.pyx in pandas._libs.parsers.TextReader._tokenize_rows()

parsers.pyx in
pandas._libs.parsers.TextReader._check_tokenize_status()

parsers.pyx in pandas._libs.parsers.raise_parser_error()

UnicodeDecodeError: 'utf-8' codec can't decode byte 0xff in position
54445: invalid start byte

from google.colab import files
uploaded = files.upload()

<IPython.core.display.HTML object>
```

```
Saving titanic.xlsx.csv to titanic.xlsx.csv
```

```
df = pd.read_csv("titanic.xlsx.csv")  # Change filename if needed
df.head()
```

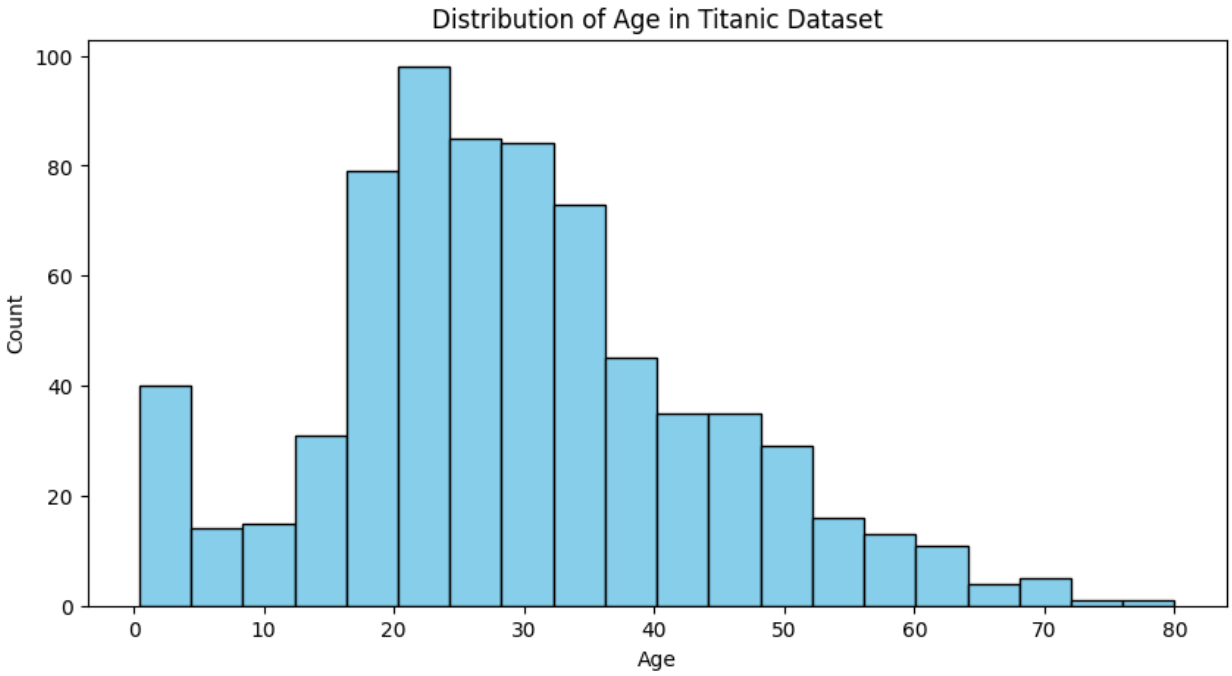

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 891,\n  \"fields\": [\
n    {\n      \"column\": \"PassengerId\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 257,\n        \"min\": 1,\n
\"max\": 891,\n        \"num_unique_values\": 891,\n
\"samples\": [\n          710,\n          440,\n          841\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Survived\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          1,\n
0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Pclass\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 1,\n        \"max\": 3,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          3,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Name\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 891,\n        \"samples\": [\n
\"Moubarek, Master. Halim Gonios (\\\"William George\\\")\",\n
\"Kvillner, Mr. Johan Henrik Johannesson\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Sex\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"female\",\n          \"male\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 14.526497332334044,\
n        \"min\": 0.42,\n        \"max\": 80.0,\n
\"num_unique_values\": 88,\n        \"samples\": [\n          0.75,\n
22.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"SibSp\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1,\n        \"min\": 0,\n        \"max\": 8,\n
\"num_unique_values\": 7,\n        \"samples\": [\n          1,\n
0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Parch\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 0,\n        \"max\": 6,\n
\"num_unique_values\": 7,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Ticket\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 681,\n        \"samples\": [\n
\"11774\",\n          \"248740\"\n        ],\n
```

\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Fare\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 49.693428597180905,\n
\"min\": 0.0,\n   \"max\": 512.3292,\n
\"num_unique_values\": 248,\n   \"samples\": [\n
11.2417,\n   51.8625\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"Cabin\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 147,\n
\"samples\": [\n   \"D45\",\n   \"B49\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Embarked\",\n   \"properties\":
{\n   \"dtype\": \"category\",\n   \"num_unique_values\":
3,\n   \"samples\": [\n   \"S\",\n   \"C\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe","variable_name":"df"}

```
df.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

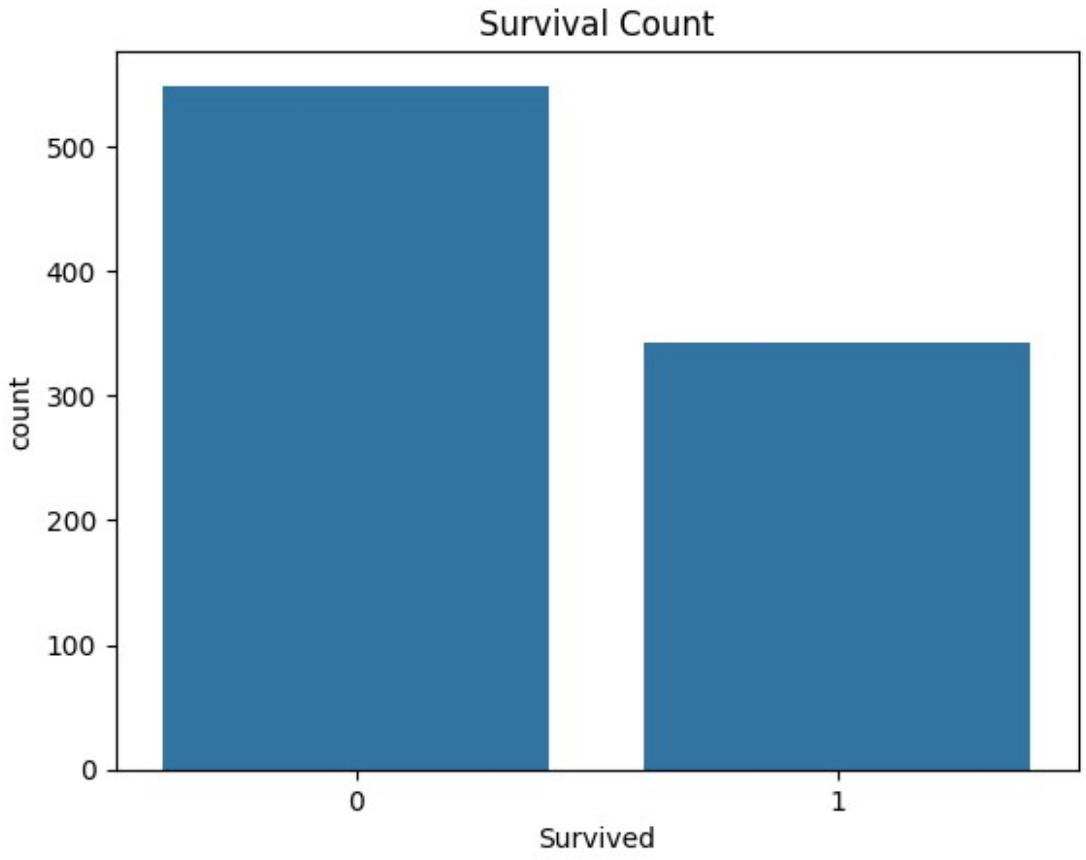
```
plt.figure(figsize=(10, 5))
plt.hist(df["Age"].dropna(), bins=20, color="skyblue",
edgecolor="black")
plt.xlabel("Age")
plt.ylabel("Count")
plt.title("Distribution of Age in Titanic Dataset")
plt.show()
```



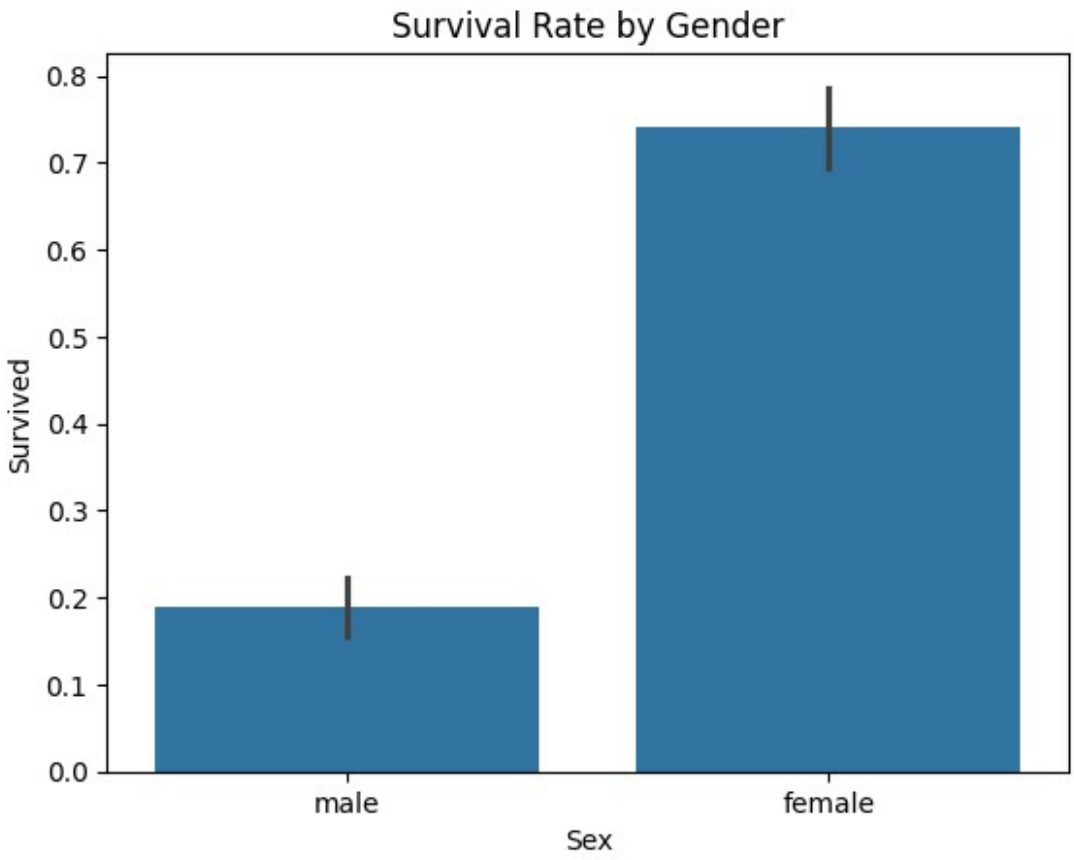

```
plt.figure(figsize=(6, 4))
sns.boxplot(x='class', y='fare', data=df, palette='coolwarm')
plt.title('Fare Prices by Class')
plt.show()

---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
<ipython-input-11-9cef617e10ff> in <cell line: 0>()
      1 plt.figure(figsize=(6, 4))
----> 2 sns.boxplot(x='class', y='fare', data=df, palette='coolwarm')
      3 plt.title('Fare Prices by Class')
      4 plt.show()

/usr/local/lib/python3.11/dist-packages/seaborn/categorical.py in
boxplot(data, x, y, hue, order, hue_order, orient, color, palette,
saturation, fill, dodge, width, gap, whis, linecolor, linewidth,
fliersize, hue_norm, native_scale, log_scale, formatter, legend, ax,
**kwargs)
   1595 ):
   1596
-> 1597     p = _CategoricalPlotter(
   1598         data=data,
   1599         variables=dict(x=x, y=y, hue=hue),

/usr/local/lib/python3.11/dist-packages/seaborn/categorical.py in
__init__(self, data, variables, order, orient, require_numeric, color,
legend)
```

```
     65     ):
     66
---> 67         super().__init__(data=data, variables=variables)
     68
     69         # This method takes care of some bookkeeping that is
necessary because the

/usr/local/lib/python3.11/dist-packages/seaborn/_base.py in
__init__(self, data, variables)
    632         # information for numeric axes would be information
about log scales.
    633         self._var_ordered = {"x": False, "y": False}  # alt.,
used DefaultDict
--> 634         self.assign_variables(data, variables)
    635
    636         # TODO Lots of tests assume that these are called to
initialize the

/usr/local/lib/python3.11/dist-packages/seaborn/_base.py in
assign_variables(self, data, variables)
    677             # to centralize / standardize data consumption
logic.
    678             self.input_format = "long"
--> 679             plot_data = PlotData(data, variables)
    680             frame = plot_data.frame
    681             names = plot_data.names

/usr/local/lib/python3.11/dist-packages/seaborn/_core/data.py in
__init__(self, data, variables)
     56
     57         data = handle_data_source(data)
---> 58         frame, names, ids = self._assign_variables(data,
variables)
     59
     60         self.frame = frame

/usr/local/lib/python3.11/dist-packages/seaborn/_core/data.py in
_assign_variables(self, data, variables)
    230                 else:
    231                     err += "An entry with this name does not
appear in `data`."
--> 232                 raise ValueError(err)
    233
    234             else:

ValueError: Could not interpret value `class` for `x`. An entry with
this name does not appear in `data`.

<Figure size 600x400 with 0 Axes>
```

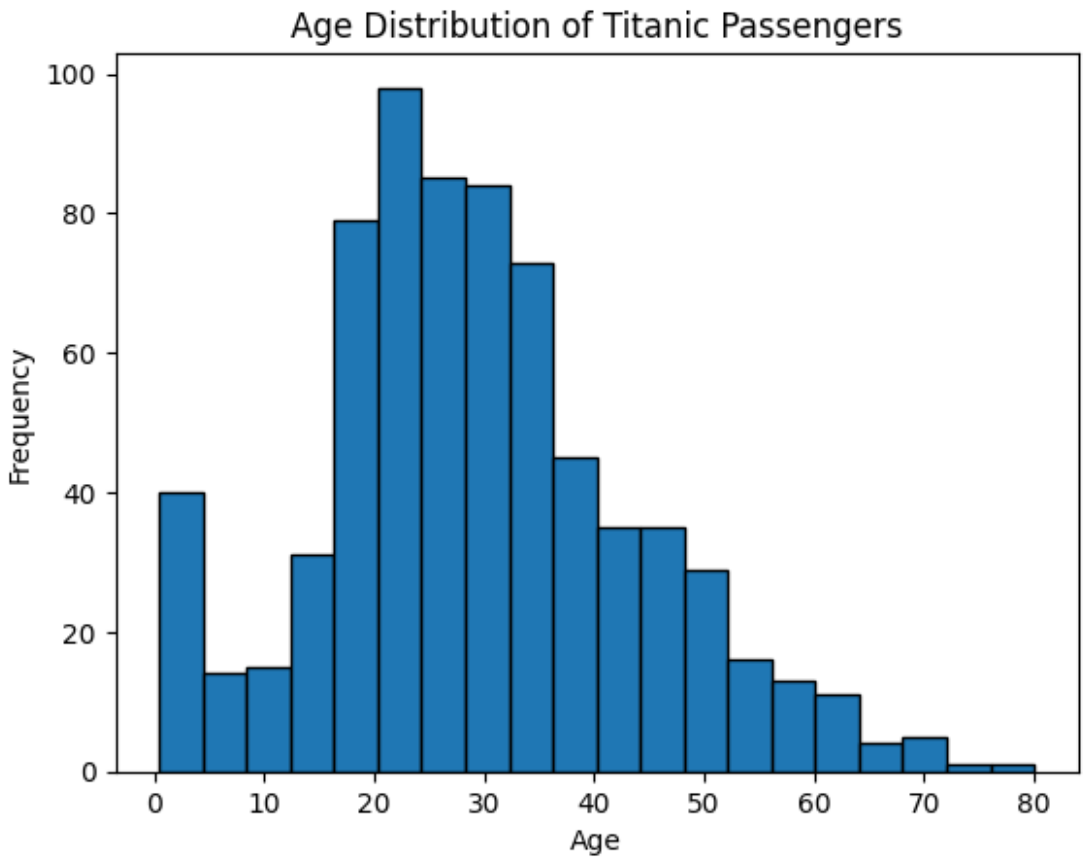
```
sns.countplot(x="Survived", data=df)
plt.title("Survival Count")
plt.show()
```




```
sns.barplot(x="Sex", y="Survived", data=df)
plt.title("Survival Rate by Gender")
plt.show()
```




```
plt.hist(df["Age"], bins=20, edgecolor="black")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.title("Age Distribution of Titanic Passengers")
plt.show()
```




```
import seaborn as sns

# Compute correlation matrix
corr = df.corr()

# Plot heatmap
plt.figure(figsize=(8,6))
sns.heatmap(corr, annot=True, cmap="coolwarm", linewidths=0.5)
plt.title("Feature Correlation Heatmap")
plt.show()
```

```
---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
<ipython-input-15-ea8798287da6> in <cell line: 0>()
      2
      3 # Compute correlation matrix
----> 4 corr = df.corr()
      5
      6 # Plot heatmap

/usr/local/lib/python3.11/dist-packages/pandas/core/frame.py in
```

```
corr(self, method, min_periods, numeric_only)
  11047         cols = data.columns
  11048         idx = cols.copy()
> 11049         mat = data.to_numpy(dtype=float, na_value=np.nan,
copy=False)
  11050
  11051         if method == "pearson":

/usr/local/lib/python3.11/dist-packages/pandas/core/frame.py in
to_numpy(self, dtype, copy, na_value)
   1991         if dtype is not None:
   1992             dtype = np.dtype(dtype)
-> 1993         result = self._mgr.as_array(dtype=dtype, copy=copy,
na_value=na_value)
   1994         if result.dtype is not dtype:
   1995             result = np.asarray(result, dtype=dtype)

/usr/local/lib/python3.11/dist-packages/pandas/core/internals/managers
.py in as_array(self, dtype, copy, na_value)
   1692                 arr.flags.writeable = False
   1693         else:
-> 1694             arr = self._interleave(dtype=dtype,
na_value=na_value)
   1695             # The underlying data was copied within
_interleave, so no need
   1696             # to further copy if copy=True or setting na_value

/usr/local/lib/python3.11/dist-packages/pandas/core/internals/managers
.py in _interleave(self, dtype, na_value)
   1751             else:
   1752                 arr = blk.get_values(dtype)
-> 1753             result[rl.indexer] = arr
   1754             itemmask[rl.indexer] = 1
   1755

ValueError: could not convert string to float: 'Braund, Mr. Owen
Harris'

corr = df.corr()

# Plot heatmap
plt.figure(figsize=(8,6))
sns.heatmap(corr, annot=True, cmap="coolwarm", linewidths=0.5)
plt.title("Feature Correlation Heatmap")
plt.show()

---------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
```

```
<ipython-input-16-9163d7049fed> in <cell line: 0>()
----> 1 corr = df.corr()
      2
      3 # Plot heatmap
      4 plt.figure(figsize=(8,6))
      5 sns.heatmap(corr, annot=True, cmap="coolwarm", linewidths=0.5)

/usr/local/lib/python3.11/dist-packages/pandas/core/frame.py in
corr(self, method, min_periods, numeric_only)
  11047         cols = data.columns
  11048         idx = cols.copy()
> 11049         mat = data.to_numpy(dtype=float, na_value=np.nan,
copy=False)
  11050
  11051         if method == "pearson":

/usr/local/lib/python3.11/dist-packages/pandas/core/frame.py in
to_numpy(self, dtype, copy, na_value)
   1991         if dtype is not None:
   1992             dtype = np.dtype(dtype)
-> 1993         result = self._mgr.as_array(dtype=dtype, copy=copy,
na_value=na_value)
   1994         if result.dtype is not dtype:
   1995             result = np.asarray(result, dtype=dtype)

/usr/local/lib/python3.11/dist-packages/pandas/core/internals/managers
.py in as_array(self, dtype, copy, na_value)
   1692                 arr.flags.writeable = False
   1693         else:
-> 1694             arr = self._interleave(dtype=dtype,
na_value=na_value)
   1695             # The underlying data was copied within
_interleave, so no need
   1696             # to further copy if copy=True or setting na_value

/usr/local/lib/python3.11/dist-packages/pandas/core/internals/managers
.py in _interleave(self, dtype, na_value)
   1751             else:
   1752                 arr = blk.get_values(dtype)
-> 1753             result[rl.indexer] = arr
   1754             itemmask[rl.indexer] = 1
   1755

ValueError: could not convert string to float: 'Braund, Mr. Owen
Harris'
```

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

```
import seaborn as sns
from sklearn.preprocessing import LabelEncoder

from google.colab import files
uploaded = files.upload()  # Upload Titanic.csv file

# Load dataset
df = pd.read_csv("titanic.xlsx.csv")

# Display first 5 rows
df.head()
```

```
<IPython.core.display.HTML object>
```

```
Saving titanic.xlsx.csv to titanic.xlsx (1).csv
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 891,\n  \"fields\": [\
n    {\n      \"column\": \"PassengerId\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 257,\n        \"min\": 1,\n
\"max\": 891,\n        \"num_unique_values\": 891,\n
\"samples\": [\n          710,\n          440,\n          841\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Survived\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          1,\n
0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Pclass\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 1,\n        \"max\": 3,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          3,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Name\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 891,\n        \"samples\": [\n
\"Moubarek, Master. Halim Gonios (\\\"William George\\\")\",\n
\"Kvillner, Mr. Johan Henrik Johannesson\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Sex\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"female\",\n          \"male\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 14.526497332334044,\
n        \"min\": 0.42,\n        \"max\": 80.0,\n
\"num_unique_values\": 88,\n        \"samples\": [\n          0.75,\n
22.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"SibSp\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1,\n        \"min\": 0,\n        \"max\": 8,\n
```

\"num_unique_values\": 7,\n   \"samples\": [\n   1,\n
0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Parch\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0,\n   \"min\": 0,\n   \"max\": 6,\n
\"num_unique_values\": 7,\n   \"samples\": [\n   0,\n
1\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Ticket\",\n   \"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 681,\n   \"samples\": [\n
\"11774\",\n   \"248740\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Fare\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 49.693428597180905,\n
\"min\": 0.0,\n   \"max\": 512.3292,\n
\"num_unique_values\": 248,\n   \"samples\": [\n
11.2417,\n   51.8625\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"Cabin\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 147,\n
\"samples\": [\n   \"D45\",\n   \"B49\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Embarked\",\n   \"properties\":
{\n   \"dtype\": \"category\",\n   \"num_unique_values\":
3,\n   \"samples\": [\n   \"S\",\n   \"C\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe","variable_name":"df"}

```
df.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

```
# Fill missing 'Age' values with median age
df["Age"].fillna(df["Age"].median(), inplace=True)

# Fill missing 'Embarked' values with most frequent value
df["Embarked"].fillna(df["Embarked"].mode()[0], inplace=True)
```

```
# Drop the 'Cabin' column (too many missing values)
df.drop(columns=["Cabin"], inplace=True)
```

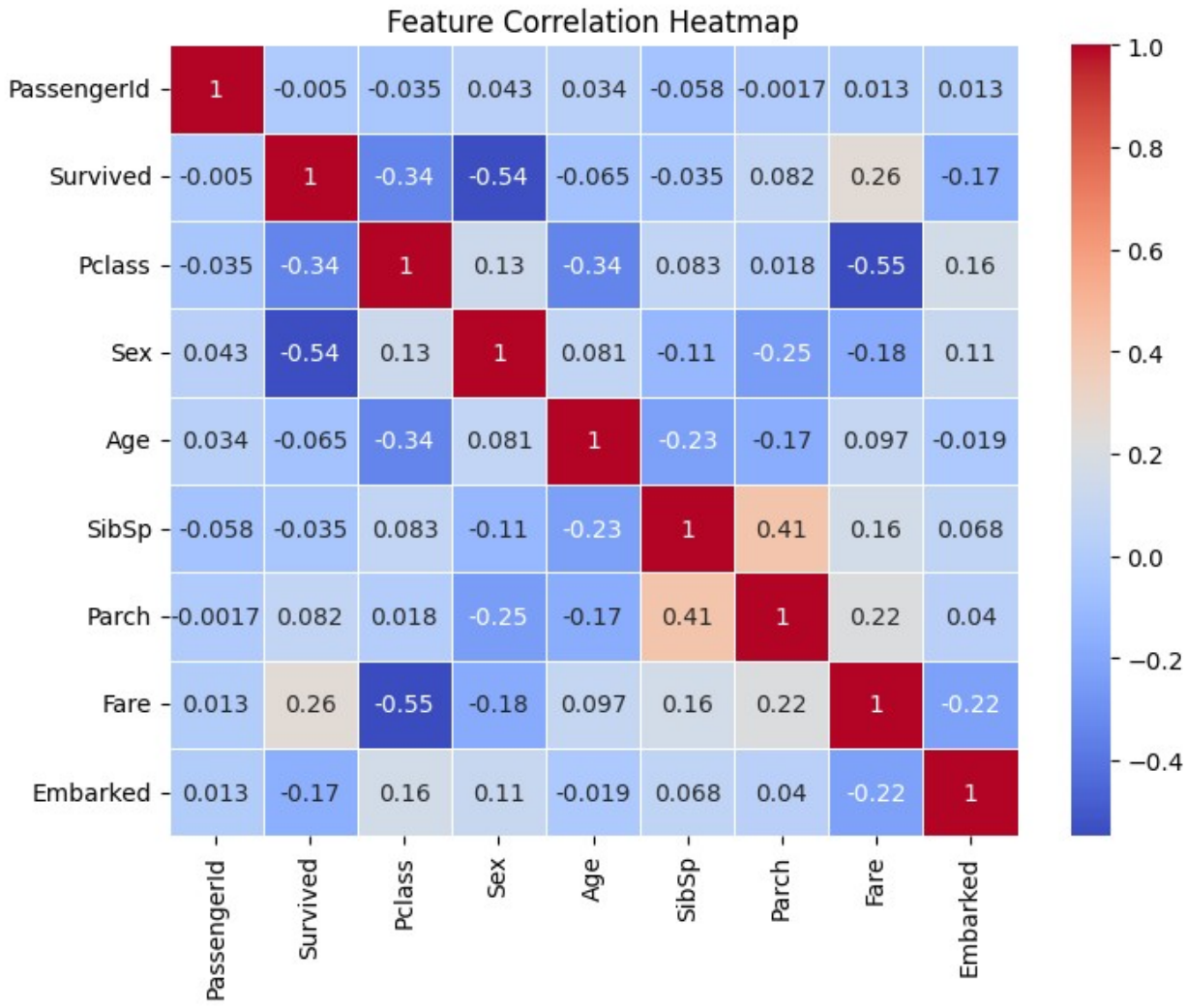
```
<ipython-input-28-9652e788c7c3>:2: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df["Age"].fillna(df["Age"].median(), inplace=True)
<ipython-input-28-9652e788c7c3>:5: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df["Embarked"].fillna(df["Embarked"].mode()[0], inplace=True)
```

```
# Compute correlation
corr = df_numeric.corr()

# Plot heatmap
plt.figure(figsize=(8,6))
sns.heatmap(corr, annot=True, cmap="coolwarm", linewidths=0.5)
plt.title("Feature Correlation Heatmap")
plt.show()
```

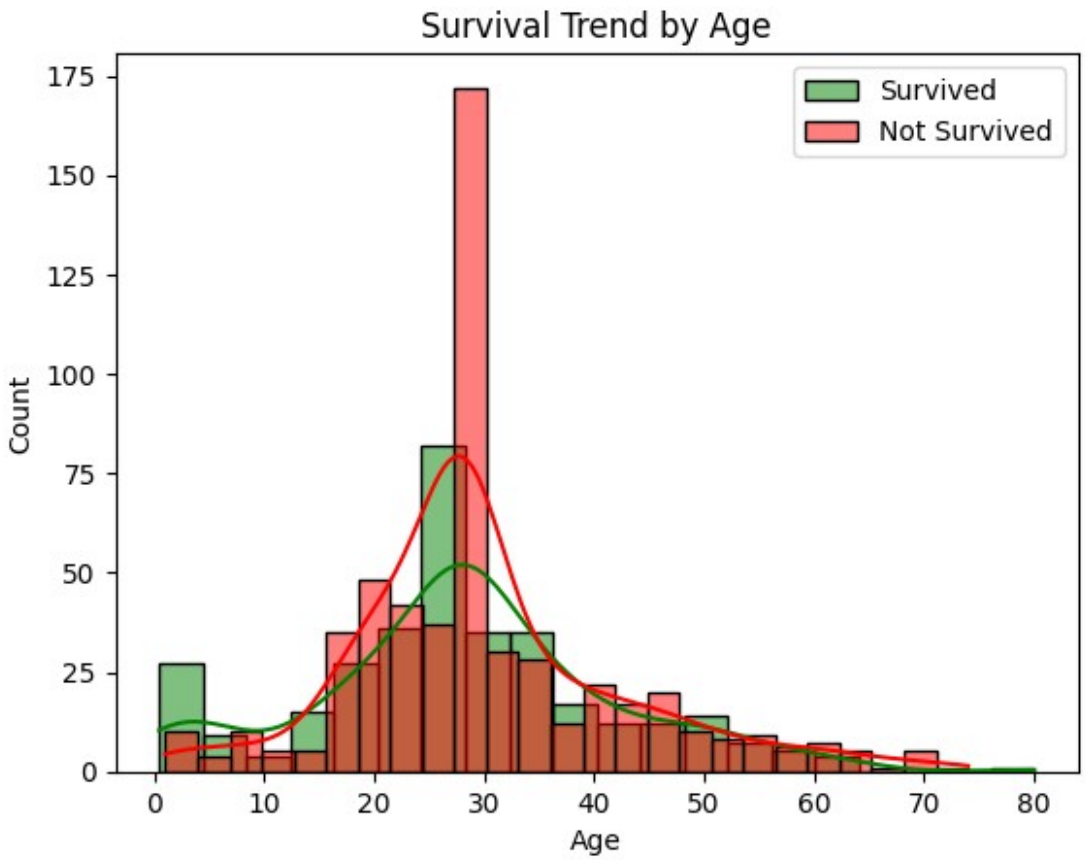
```
---------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
<ipython-input-29-bfff1129b7d0> in <cell line: 0>()
      1 # Compute correlation
----> 2 corr = df_numeric.corr()
      3
      4 # Plot heatmap
      5 plt.figure(figsize=(8,6))
```

```
NameError: name 'df_numeric' is not defined
```

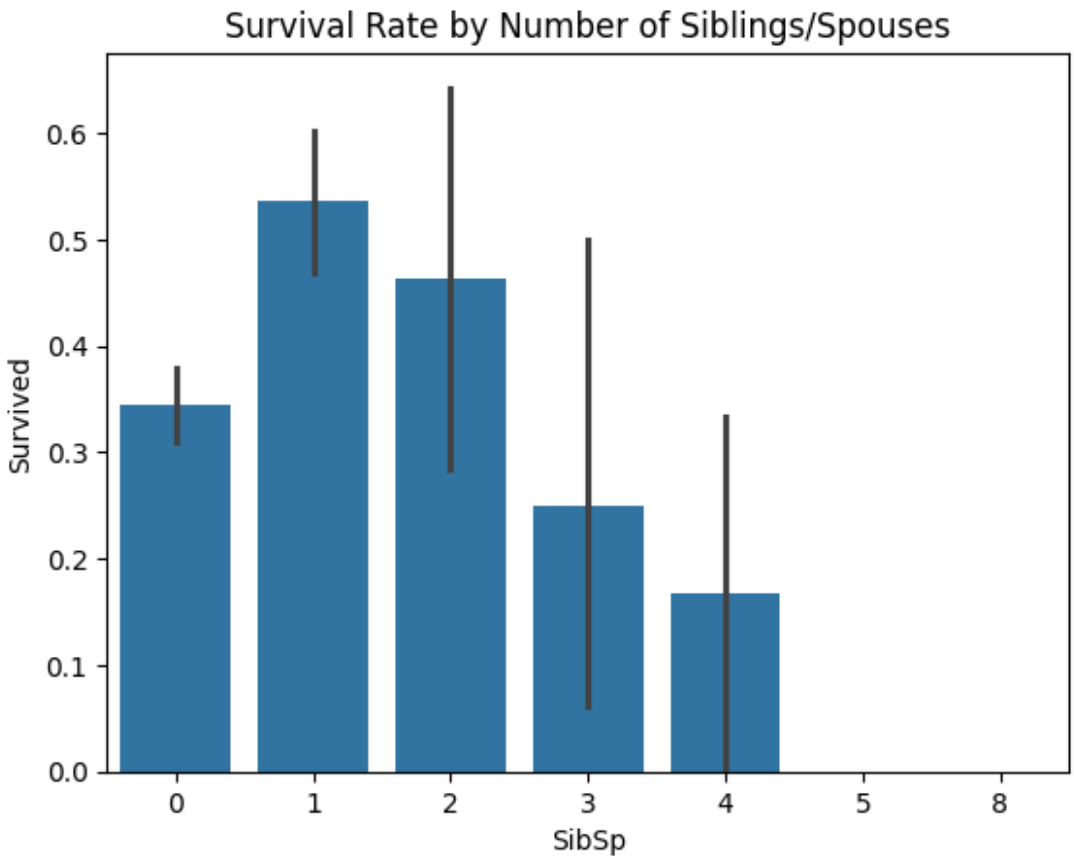
```
# Convert 'Sex' column: male -> 1, female -> 0
from sklearn.preprocessing import LabelEncoder

label_enc = LabelEncoder()
df["Sex"] = label_enc.fit_transform(df["Sex"])  # Male = 1, Female = 0

# Convert 'Embarked' column: S=2, C=0, Q=1
df["Embarked"] = label_enc.fit_transform(df["Embarked"])

# Drop non-numeric columns like 'Name' and 'Ticket'
df_numeric = df.drop(columns=["Name", "Ticket"])  # Now df_numeric
contains only numbers

# Compute correlation
corr = df_numeric.corr()

# Plot heatmap
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8,6))
sns.heatmap(corr, annot=True, cmap="coolwarm", linewidths=0.5)
plt.title("Feature Correlation Heatmap")
plt.show()
```

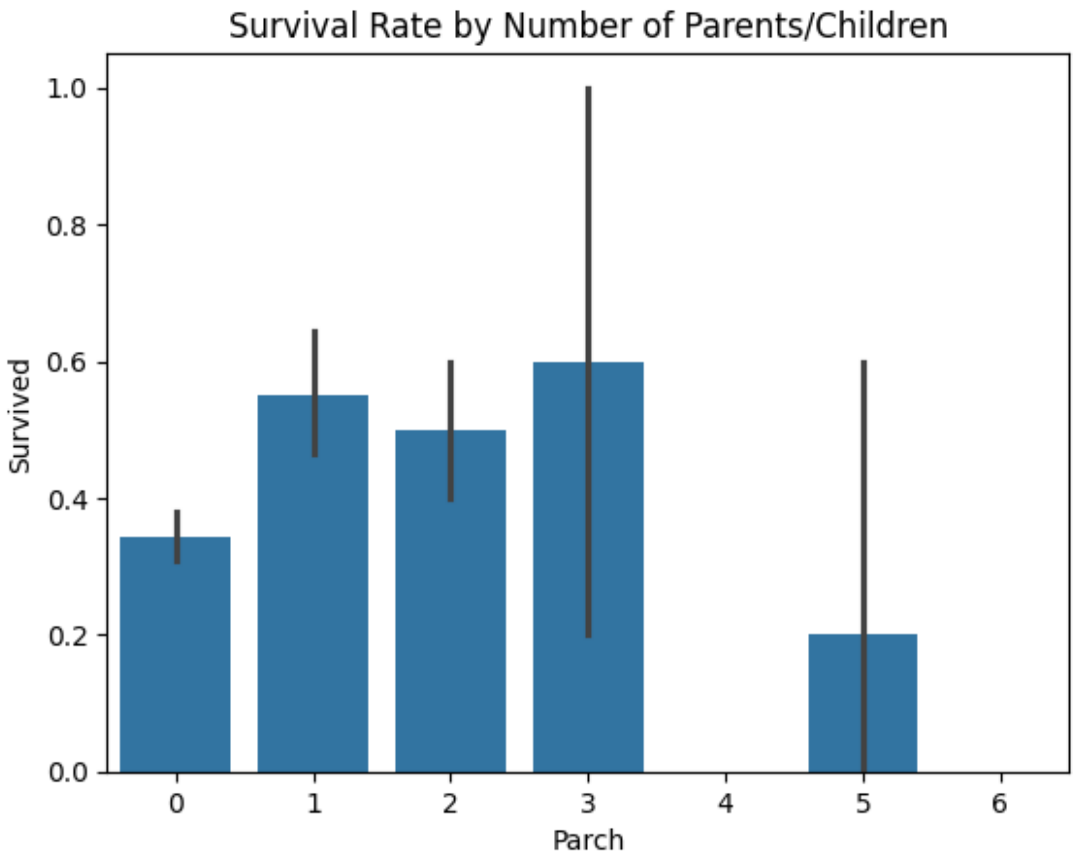



```
sns.histplot(df[df["Survived"] == 1]["Age"], color="green",
label="Survived", kde=True)
sns.histplot(df[df["Survived"] == 0]["Age"], color="red", label="Not
Survived", kde=True)
plt.legend()
plt.title("Survival Trend by Age")
plt.show()
```




```
sns.barplot(x="SibSp", y="Survived", data=df)
plt.title("Survival Rate by Number of Siblings/Spouses")
plt.show()

sns.barplot(x="Parch", y="Survived", data=df)
plt.title("Survival Rate by Number of Parents/Children")
plt.show()
```


Survival Rate by Number of Siblings/Spouses
Survived
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
1
2
3
4
5
8
SibSp


Survival Rate by Number of Parents/Children
Survived
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
6
Parch