Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Data Mining

Lab - 3

Name: Smit Maru

Enrollment No: 23010101161

1) First, you need to read the titanic dataset from local disk and display first five records

In [1]:
```
import pandas as pd
```

In [2]:
```
import numpy as np
```

In [3]:
```
titanic = pd.read_csv('titanic.csv')
titanic.head(5)
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

2) Identify Nominal, Ordinal, Binary and Numeric attributes from data sets and display all values.

In [4]:
```
Nominal = ['Name','Sex','Embarked']
Binary = ['Survived','SibSp']
Ordinal = ['Pclass']
Numeric = ['Age','Parch','Fare']
```

3) Identify symmetric and asymmetric binary attributes from data sets and display all values.

In [6]:
```
print("Count : ")
print(titanic['Servived'].value_count())
```

```
Count :
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3805, in Index.get_loc(self, key)
   3804 try:
-> 3805     return self._engine.get_loc(casted_key)
   3806 except KeyError as err:

File index.pyx:167, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:196, in pandas._libs.index.IndexEngine.get_loc()

File pandas\\_libs\\hashtable_class_helper.pxi:7081, in pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas\\_libs\\hashtable_class_helper.pxi:7089, in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Servived'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[6], line 2
      1 print("Count : ")
----> 2 print(titanic['Servived'])

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:4102, in DataFrame.__getitem__(self, key)
   4100 if self.columns.nlevels > 1:
   4101     return self._getitem_multilevel(key)
-> 4102 indexer = self.columns.get_loc(key)
   4103 if is_integer(indexer):
   4104     indexer = [indexer]

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3812, in Index.get_loc(self, key)
   3807     if isinstance(casted_key, slice) or (
   3808         isinstance(casted_key, abc.Iterable)
   3809         and any(isinstance(x, slice) for x in casted_key)
   3810     ):
   3811         raise InvalidIndexError(key)
-> 3812     raise KeyError(key) from err
   3813 except TypeError:
   3814     # If we have a listlike key, _check_indexing_error will raise
   3815     #  InvalidIndexError. Otherwise we fall through and re-raise
   3816     #  the TypeError.
   3817     self._check_indexing_error(key)

KeyError: 'Servived'
```

4) For each quantitative attribute, calculate its average, standard deviation, minimum, mode, range and maximum values.

In [7]:
```
Quantative = ['PassengerId','Survived','Pclass','Age','SibSp','Parch','Fare']

for col in Quantative:
    print(':::',col,':::')
    print('Mean:',titanic[col].mean())
    print('SD:',titanic[col].std())
    print('Min:',titanic[col].min())
    print('Max:',titanic[col].max())
    print('Mod:',titanic[col].mode()[0])
    print('Range:',titanic[col].max() - titanic[col].min())
```

```
::: PassengerId :::
Mean: 446.0
SD: 257.3538420152301
Min: 1
Max: 891
Mod: 1
Range: 890
::: Survived :::
Mean: 0.3838383838383838
SD: 0.4865924542648585
Min: 0
Max: 1
Mod: 0
Range: 1
::: Pclass :::
Mean: 2.308641975308642
SD: 0.8360712409770513
Min: 1
Max: 3
Mod: 3
Range: 2
::: Age :::
Mean: 29.69911764705882
SD: 14.526497332334044
Min: 0.42
Max: 80.0
Mod: 24.0
Range: 79.58
::: SibSp :::
Mean: 0.5230078563411896
SD: 1.1027434322934275
Min: 0
Max: 8
Mod: 0
Range: 8
::: Parch :::
Mean: 0.38159371492704824
SD: 0.8060572211299559
Min: 0
Max: 6
Mod: 0
Range: 6
::: Fare :::
Mean: 32.204207968574636
SD: 49.693428597180905
Min: 0.0
Max: 512.3292
Mod: 8.05
Range: 512.3292
```

6) For the qualitative attribute (class), count the frequency for each of its distinct values.

In [9]:
```
print("Passenger Class frequency : ")
print(titanic['Pclass'].value_counts())
```

```
Passenger Class frequency :
Pclass
3    491
1    216
2    184
Name: count, dtype: int64
```

7) It is also possible to display the summary for all the attributes simultaneously in a table using the describe() function. If an attribute is quantitative, it will display its mean, standard deviation and various quantiles (including minimum, median, and maximum) values. If an attribute is qualitative, it will display its number of unique values and the top (most frequent) values.

In [10]:
```
print("Numeric Summary")
print(titanic.describe())
print("Catagorial Summary")
print(titanic.describe(include=['object']))
```

```
Numeric Summary
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
Catagorial Summary
                           Name   Sex  Ticket    Cabin Embarked
count                       891   891     891      204      889
unique                      891     2     681      147        3
top     Braund, Mr. Owen Harris  male  347082  B96 B98        S
freq                          1   577       7        4      644
```

8) For multivariate statistics, you can compute the covariance and correlation between pairs of attributes.

In [11]:
```
print(titanic.cov(numeric_only=True))
```

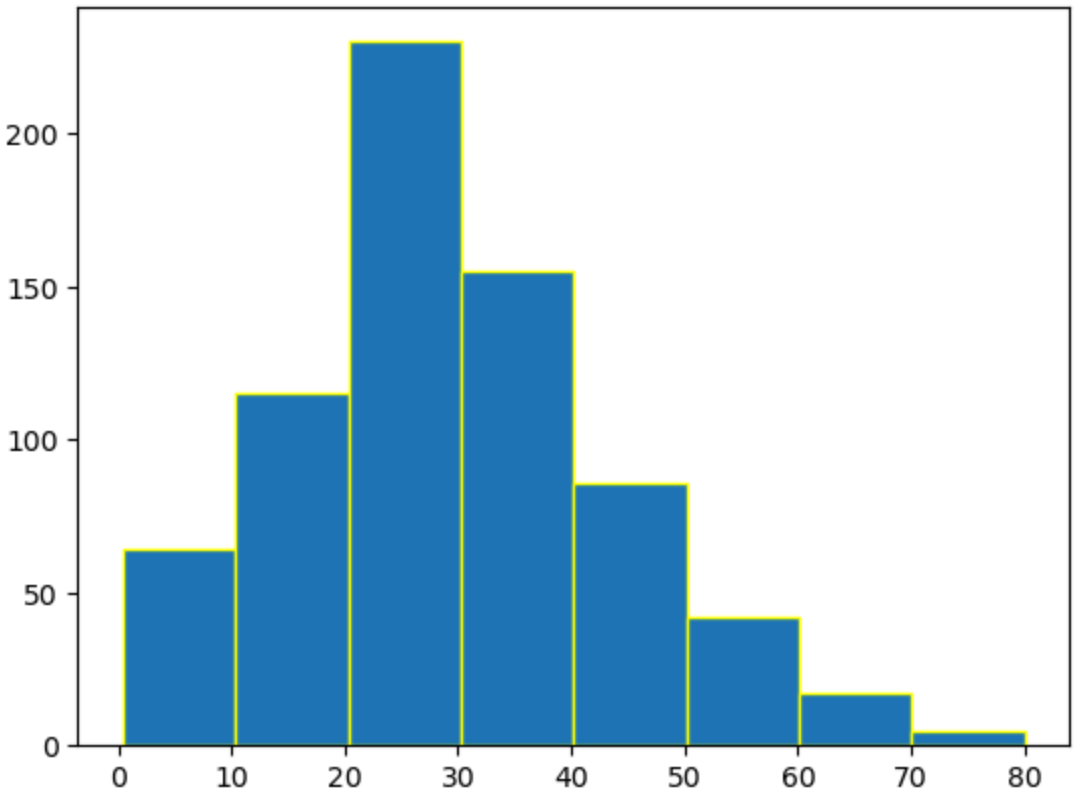
```
             PassengerId  Survived     Pclass         Age      SibSp  \
PassengerId  66231.000000 -0.626966  -7.561798  138.696504 -16.325843
Survived        -0.626966  0.236772  -0.137703   -0.551296  -0.018954
Pclass          -7.561798 -0.137703   0.699015   -4.496004   0.076599
Age            138.696504 -0.551296  -4.496004  211.019125  -4.163334
SibSp          -16.325843 -0.018954   0.076599   -4.163334   1.216043
Parch           -0.342697  0.032017   0.012429   -2.344191   0.368739
Fare           161.883369  6.221787 -22.830196   73.849030   8.748734

                Parch         Fare
PassengerId -0.342697   161.883369
Survived     0.032017     6.221787
Pclass       0.012429   -22.830196
Age         -2.344191    73.849030
SibSp        0.368739     8.748734
Parch        0.649728     8.661052
Fare         8.661052  2469.436846
```

In [12]:
```
print(titanic.corr(numeric_only=True))
```

```
             PassengerId  Survived    Pclass       Age     SibSp     Parch  \
PassengerId     1.000000 -0.005007 -0.035144  0.036847 -0.057527 -0.001652
Survived       -0.005007  1.000000 -0.338481 -0.077221 -0.035322  0.081629
Pclass         -0.035144 -0.338481  1.000000 -0.369226  0.083081  0.018443
Age             0.036847 -0.077221 -0.369226  1.000000 -0.308247 -0.189119
SibSp          -0.057527 -0.035322  0.083081 -0.308247  1.000000  0.414838
Parch          -0.001652  0.081629  0.018443 -0.189119  0.414838  1.000000
Fare            0.012658  0.257307 -0.549500  0.096067  0.159651  0.216225

                 Fare
PassengerId  0.012658
Survived     0.257307
Pclass      -0.549500
Age          0.096067
SibSp        0.159651
Parch        0.216225
Fare         1.000000
```

9) Display the histogram for Age attribute by discretizing it into 8 separate bins and counting the frequency for each bin.

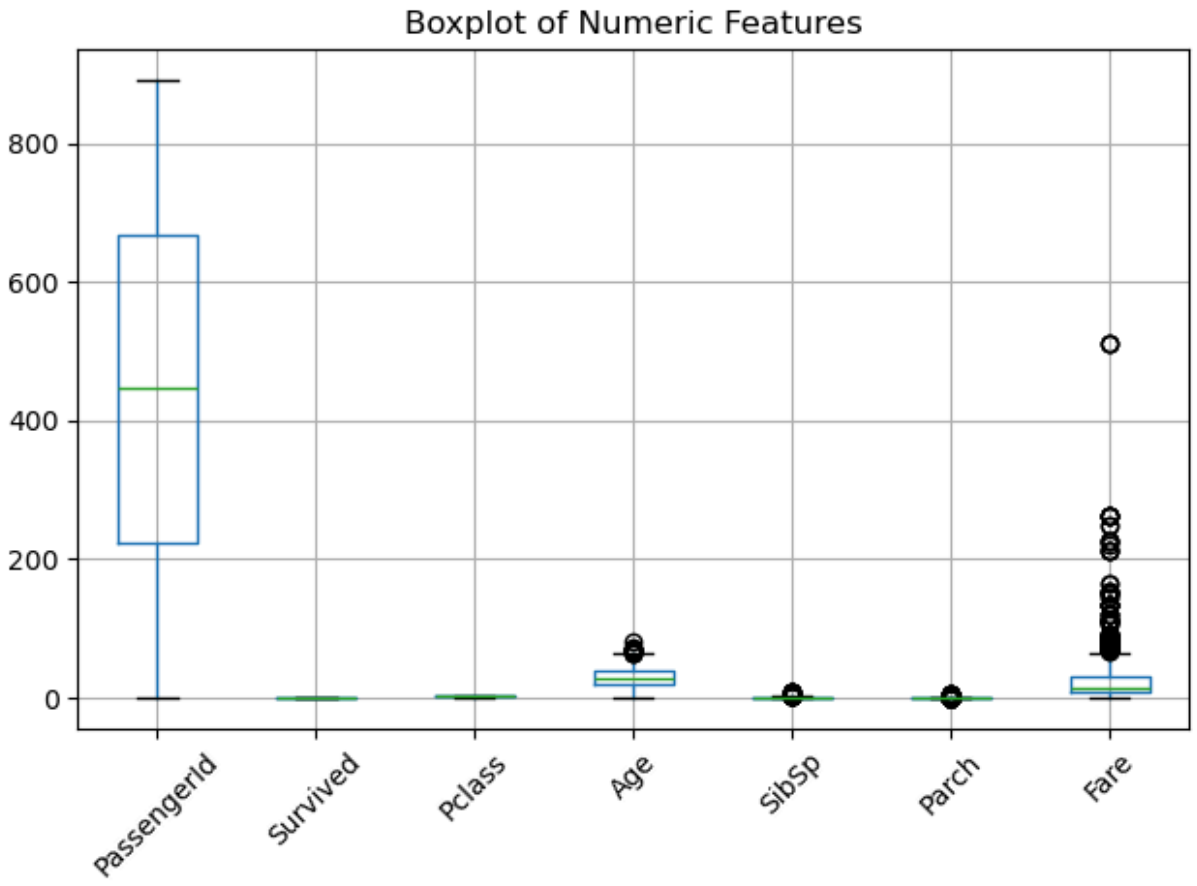
In [13]:
```
import matplotlib.pyplot as plt
plt.hist(titanic['Age'],bins=8,edgecolor='yellow')
plt.show()
```



In [ ]:

10) A boxplot can also be used to show the distribution of values for each attribute.

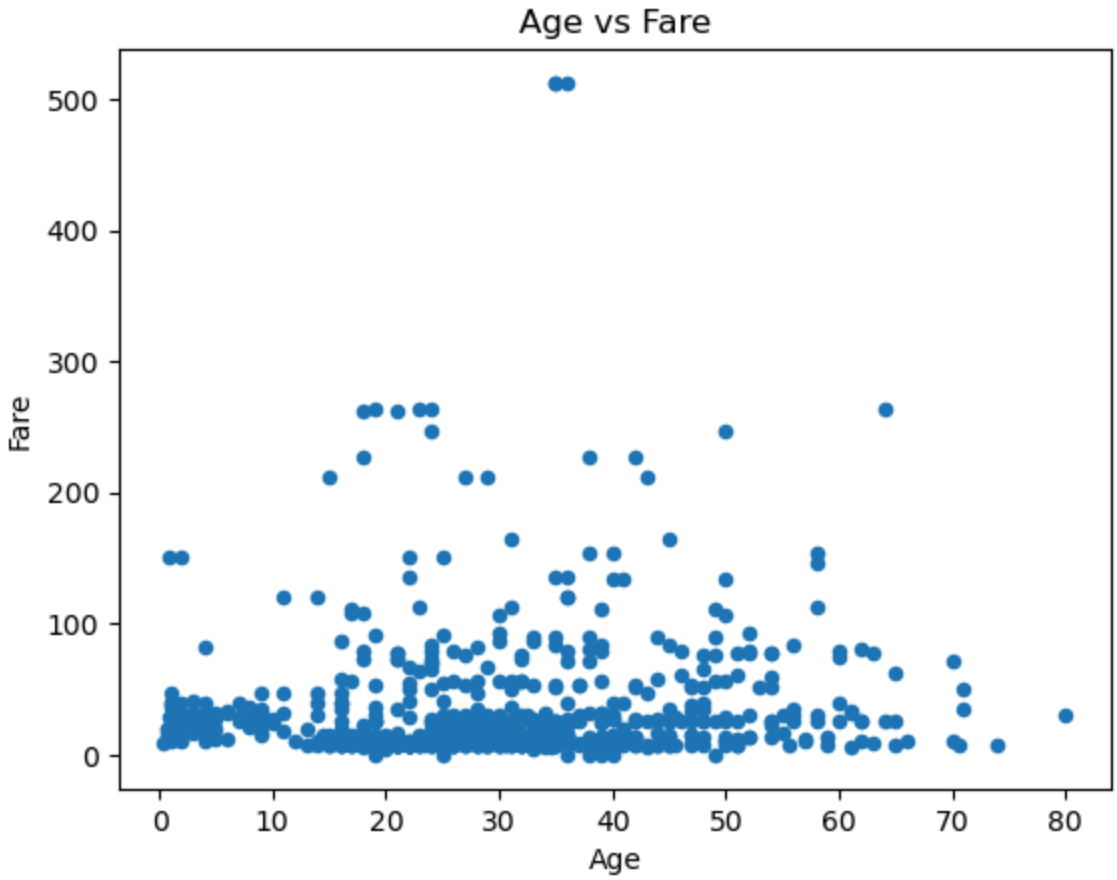
In [16]:
```
titanic.boxplot()
plt.title("Boxplot of Numeric Features")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

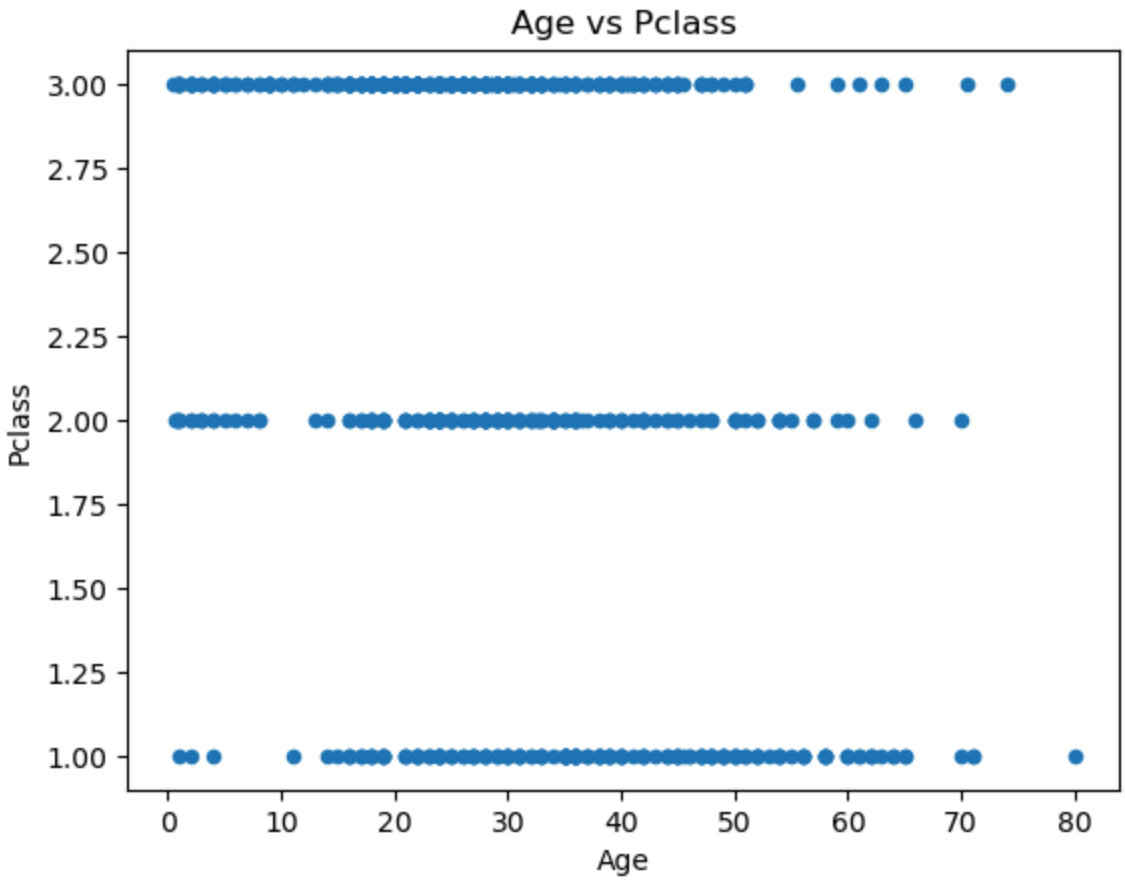


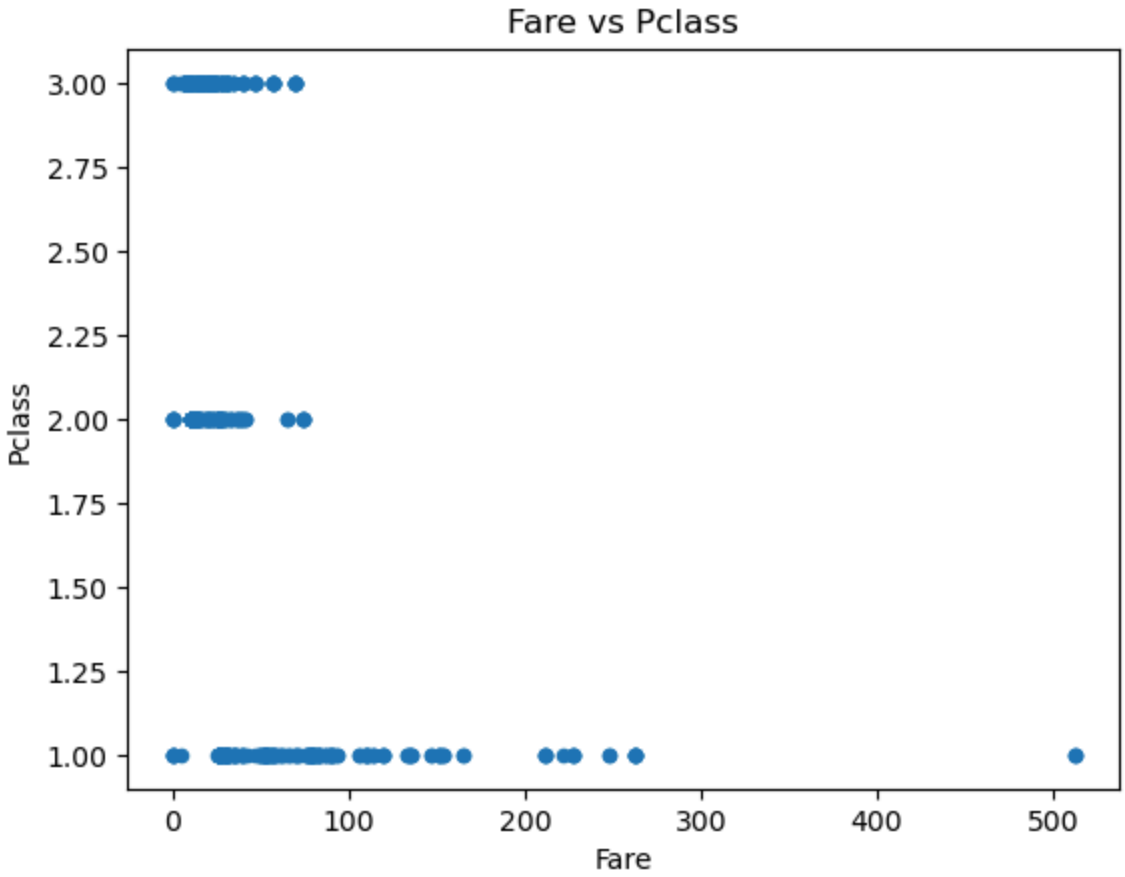
11) Display scatter plot for any 5 pair of attributes , we can use a scatter plot to visualize their joint distribution.

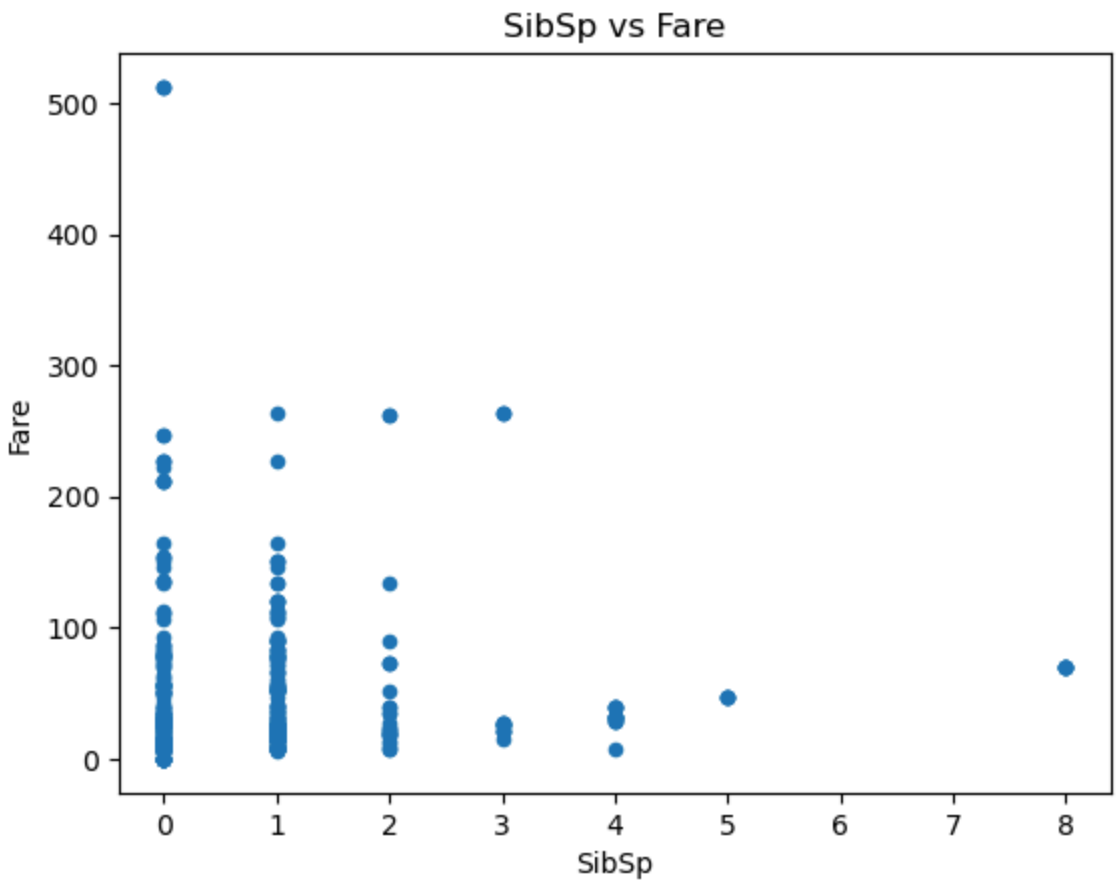
In [17]:
```
titanic.plot.scatter(x='Age', y='Fare', title='Age vs Fare')

titanic.plot.scatter(x='Age', y='Pclass', title='Age vs Pclass')

titanic.plot.scatter(x='Fare', y='Pclass', title='Fare vs Pclass')

titanic.plot.scatter(x='SibSp', y='Fare', title='SibSp vs Fare')

titanic.plot.scatter(x='Parch', y='Age', title='Parch vs Age')
```

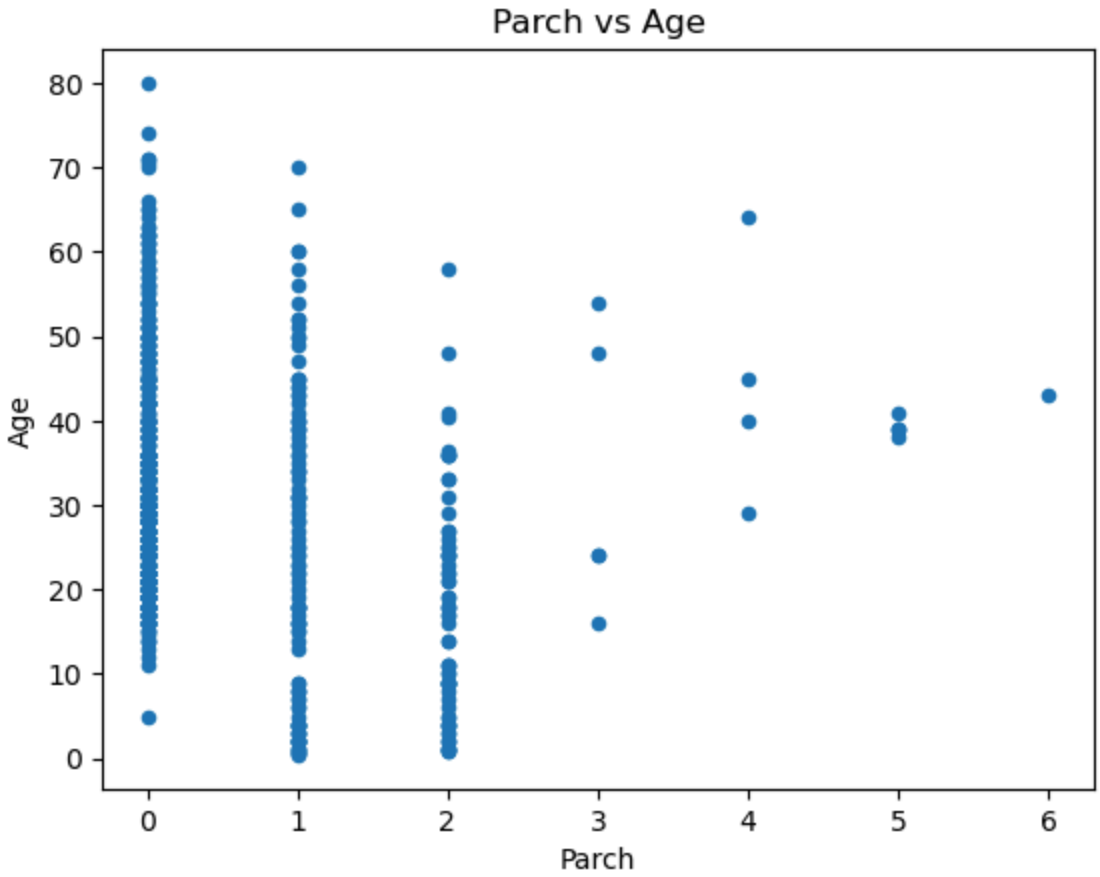
Out[17]:
```
<Axes: title={'center': 'Parch vs Age'}, xlabel='Parch', ylabel='Age'>
```


Age vs Fare
Fare
Age
0
100
200
300
400
500
0
10
20
30
40
50
60
70
80


Age vs Pclass
Pclass
Age
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
0
10
20
30
40
50
60
70
80


Fare vs Pclass
Pclass
Fare
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0
100
200
300
400
500


SibSp vs Fare
Fare
SibSp
500
400
300
200
100
0
0
1
2
3
4
5
6
7
8


Parch vs Age
Age
Parch
0
10
20
30
40
50
60
70
80
0
1
2
3
4
5
6