Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Roll No - 130

Enrollment No - 24010101694

Lab - 3

Karan Sonagara

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

In [2]:
```
import pandas as pd
```

In [4]:
```
import numpy as ny
```

In [10]:
```
df = pd.read_csv("titanic.csv")
df.head(5)
```

Out[10]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0

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

In [25]:
```
nominal = ["Name","Cabin","Embarked","Ticket","Sex"]
ordinal = ["Pclass"]
numeric = ["Age","Fare","SibSp","Parch"]
binary = ["Sex","Survived"]

print("Nominal",nominal)
print("Ordinal",ordinal)
print("Numeric",numeric)
print("Binary",binary)
```

```
Nominal ['Name', 'Cabin', 'Embarked', 'Ticket', 'Sex']
Ordinal ['Pclass']
Numeric ['Age', 'Fare', 'SibSp', 'Parch']
Binary ['Sex', 'Survived']
```

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

In [47]:
```
print("Survived values (Asymmetric Binary):")
print(df['Survived'].value_counts())

print("Sex Values (Symmetric Value):")
print(df['Sex'])
```

```
Survived values (Asymmetric Binary):
Survived
0    549
1    342
Name: count, dtype: int64
```

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

In [65]:
```
arr = ["PassengerId","Survived","Pclass","SibSp","Parch","Fare"]
for i in arr:
    print(':::',i,':::',sep="|")
    print("   Mean",df[i].mean())
    print("   Min",df[i].min())
    print("   Mode",df[i].mode())
    print("   Max",df[i].max())
    print("   Standard Deviation",df[i].std())
    print("   Ranging",df[i].max()-1)
```

```
:::|PassengerId|:::
   Mean 446.0
   Min 1
   Mode 0        1
1        2
2        3
3        4
4        5
      ...
886    887
887    888
888    889
889    890
890    891
Name: PassengerId, Length: 891, dtype: int64
   Max 891
   Standard Deviation 257.3538420152301
   Ranging 890
:::|Survived|:::
   Mean 0.3838383838383838
   Min 0
   Mode 0    0
Name: Survived, dtype: int64
   Max 1
   Standard Deviation 0.4865924542648585
   Ranging 0
:::|Pclass|:::
   Mean 2.308641975308642
   Min 1
   Mode 0    3
Name: Pclass, dtype: int64
   Max 3
   Standard Deviation 0.8360712409770513
   Ranging 2
:::|SibSp|:::
   Mean 0.5230078563411896
   Min 0
   Mode 0    0
Name: SibSp, dtype: int64
   Max 8
   Standard Deviation 1.1027434322934275
   Ranging 7
:::|Parch|:::
   Mean 0.38159371492704824
   Min 0
   Mode 0    0
Name: Parch, dtype: int64
   Max 6
   Standard Deviation 0.8060572211299559
   Ranging 5
:::|Fare|:::
   Mean 32.204207968574636
   Min 0.0
   Mode 0    8.05
Name: Fare, dtype: float64
   Max 512.3292
   Standard Deviation 49.693428597180905
   Ranging 511.3292
```

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

In [70]:
```
df["Pclass"].value_counts()
```

Out[70]:
```
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 [92]:
```
df.describe(include=["object"])
```

Out[92]:

	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

In [94]:
```
df.describe()
```

Out[94]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329

In [88]:
```
df.describe(include="all")
```

Out[88]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	
count	891.000000	891.000000	891.000000	891	891	714.000000	891.000000	89
unique	NaN	NaN	NaN	891	2	NaN	NaN	
top	NaN	NaN	NaN	Braund, Mr. Owen Harris	male	NaN	NaN	
freq	NaN	NaN	NaN	1	577	NaN	NaN	
mean	446.000000	0.383838	2.308642	NaN	NaN	29.699118	0.523008	
std	257.353842	0.486592	0.836071	NaN	NaN	14.526497	1.102743	
min	1.000000	0.000000	1.000000	NaN	NaN	0.420000	0.000000	
25%	223.500000	0.000000	2.000000	NaN	NaN	20.125000	0.000000	
50%	446.000000	0.000000	3.000000	NaN	NaN	28.000000	0.000000	
75%	668.500000	1.000000	3.000000	NaN	NaN	38.000000	1.000000	
max	891.000000	1.000000	3.000000	NaN	NaN	80.000000	8.000000	

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

In [105…

```
print("\nCorrelation Matrix:\n")
df.corr(numeric_only=True)
```

```
Correlation Matrix:
```

Out[105…

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fa
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.01265
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.25730
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.54950
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.09606
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.15965
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.21622
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.00000

In [109…

```
print(" Covariance Matrix:\n")
df.cov(numeric_only=True)
```

```
Covariance Matrix:
```

Out[109…

	PassengerId	Survived	Pclass	Age	SibSp	Parch	
PassengerId	66231.000000	-0.626966	-7.561798	138.696504	-16.325843	-0.342697	16
Survived	-0.626966	0.236772	-0.137703	-0.551296	-0.018954	0.032017	
Pclass	-7.561798	-0.137703	0.699015	-4.496004	0.076599	0.012429	-2
Age	138.696504	-0.551296	-4.496004	211.019125	-4.163334	-2.344191	7
SibSp	-16.325843	-0.018954	0.076599	-4.163334	1.216043	0.368739	
Parch	-0.342697	0.032017	0.012429	-2.344191	0.368739	0.649728	
Fare	161.883369	6.221787	-22.830196	73.849030	8.748734	8.661052	246

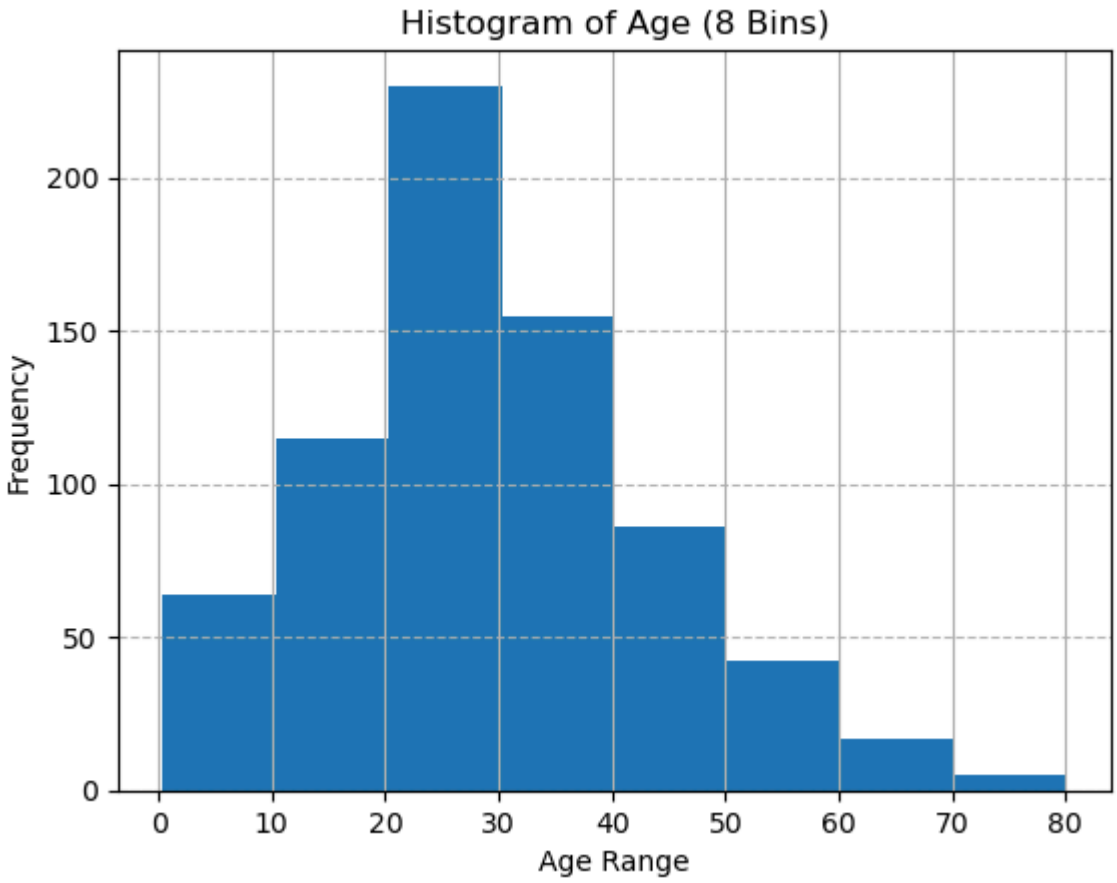
9) Display the histogram for Age attribute by discretizing it into 8 separate bins and counting the frequency for each bin.

In [166…

```
import matplotlib.pyplot as plt
```

In [143…

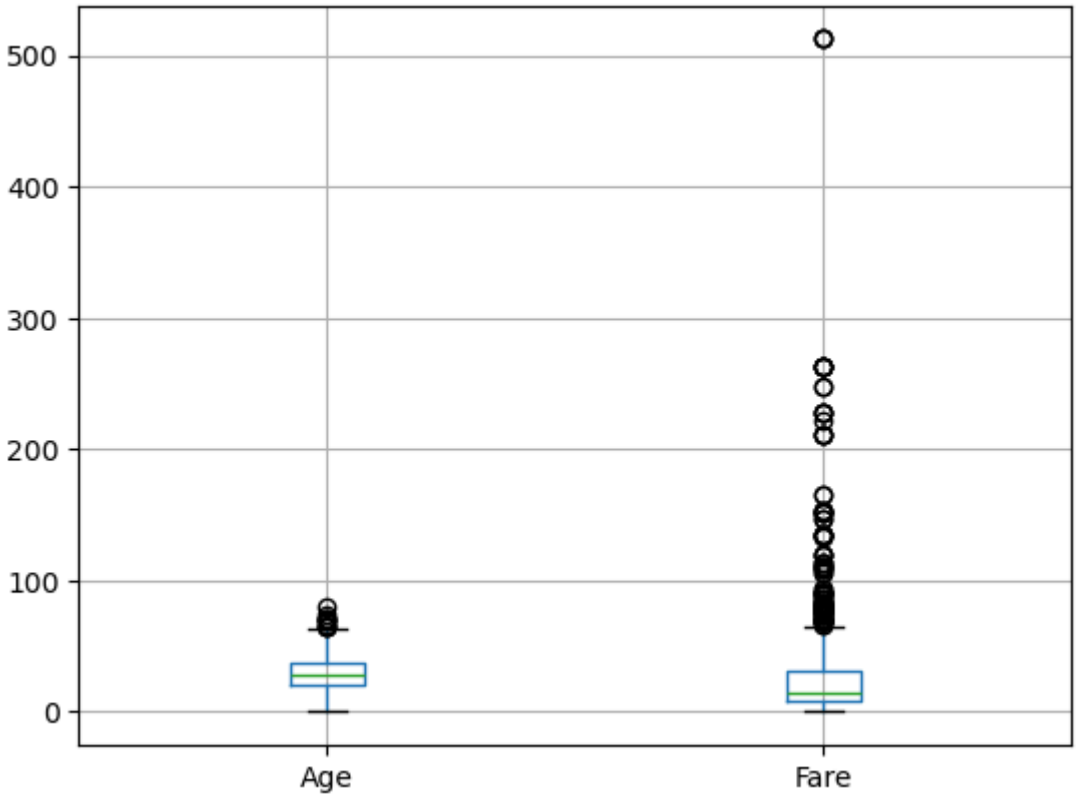
```
df['Age'].dropna().hist(bins=8)
plt.title('Histogram of Age (8 Bins)')
plt.xlabel('Age Range')
plt.ylabel('Frequency')
plt.grid(axis='y', linestyle='--', alpha=0.9)
plt.show()
```



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

In [176…
```
df.boxplot(['Age','Fare'])
```

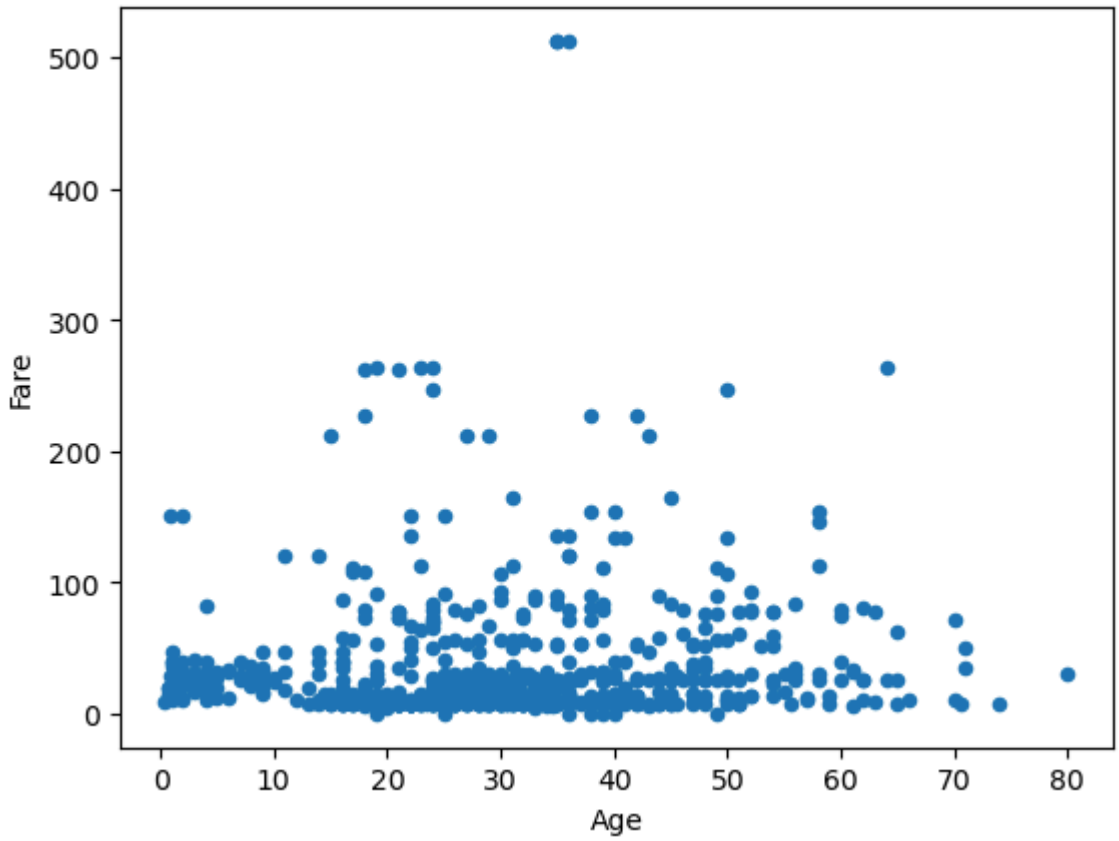
Out[176… <Axes: >

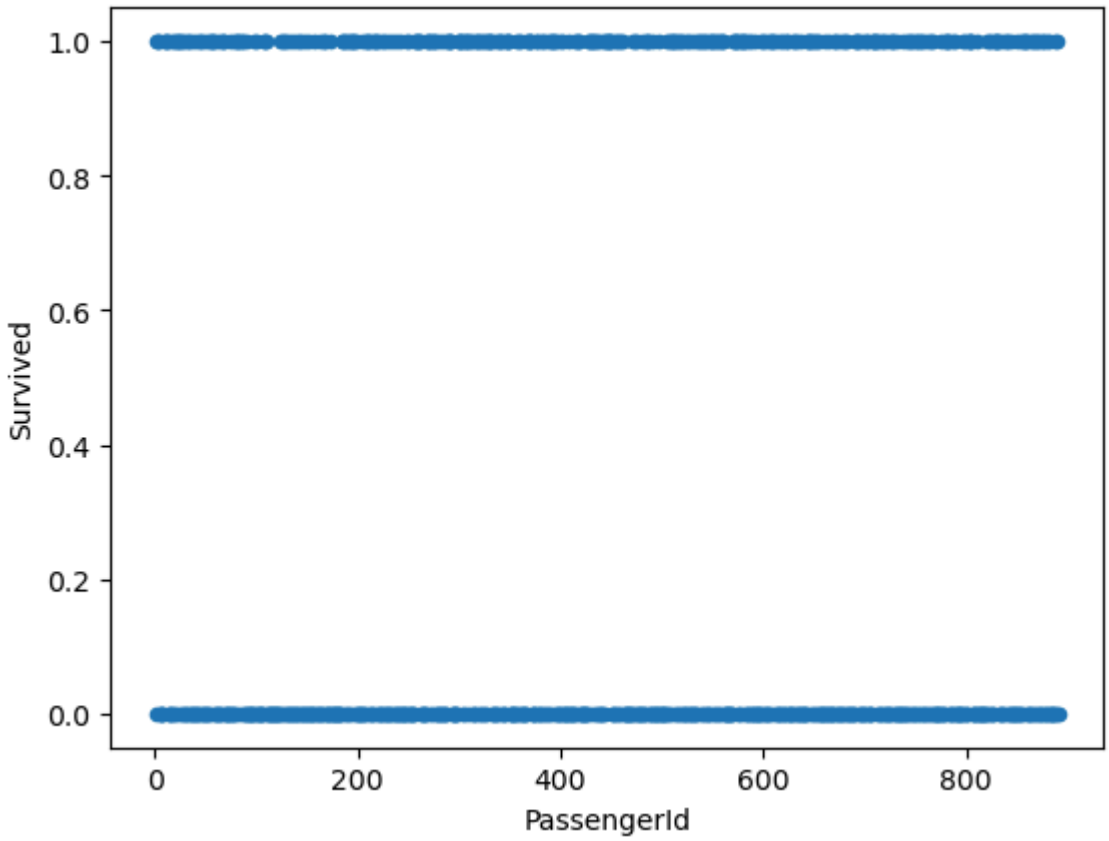


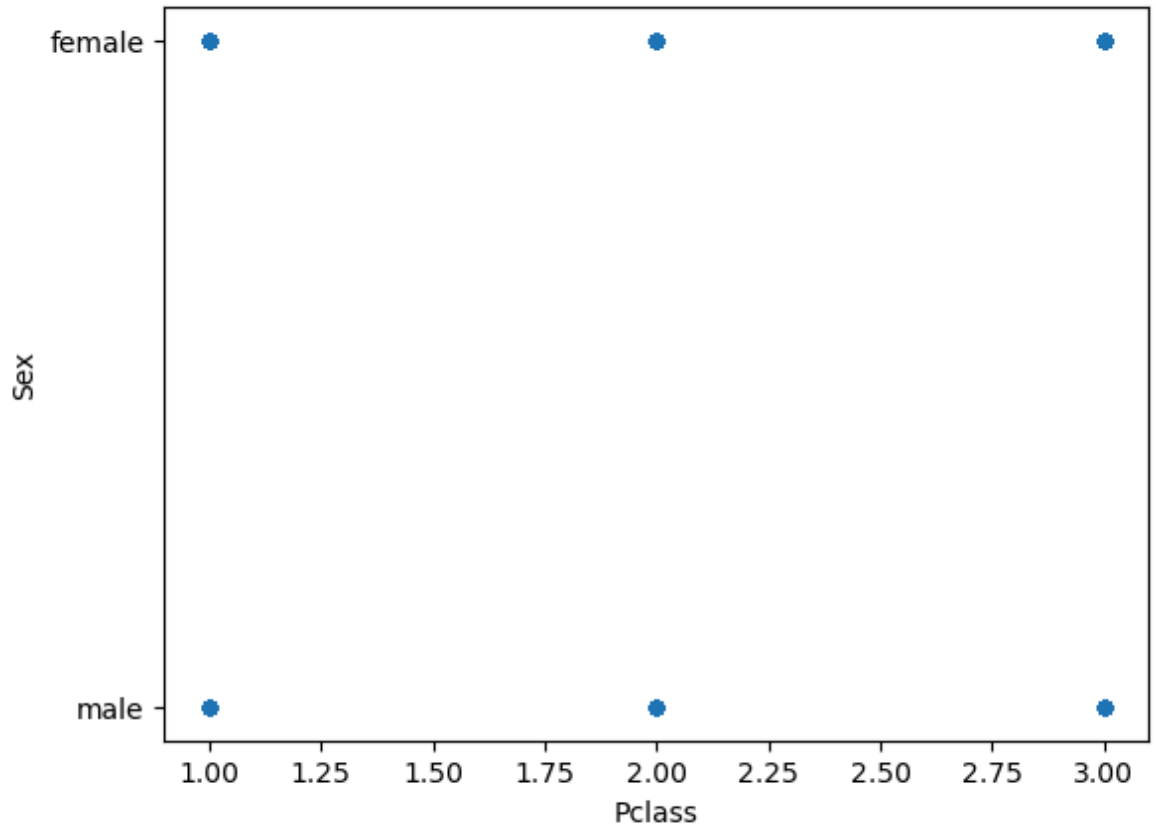
11) Display scatter plot for any 5 pair of attributes , we can use a scatter plot to visualize their joint distribution.

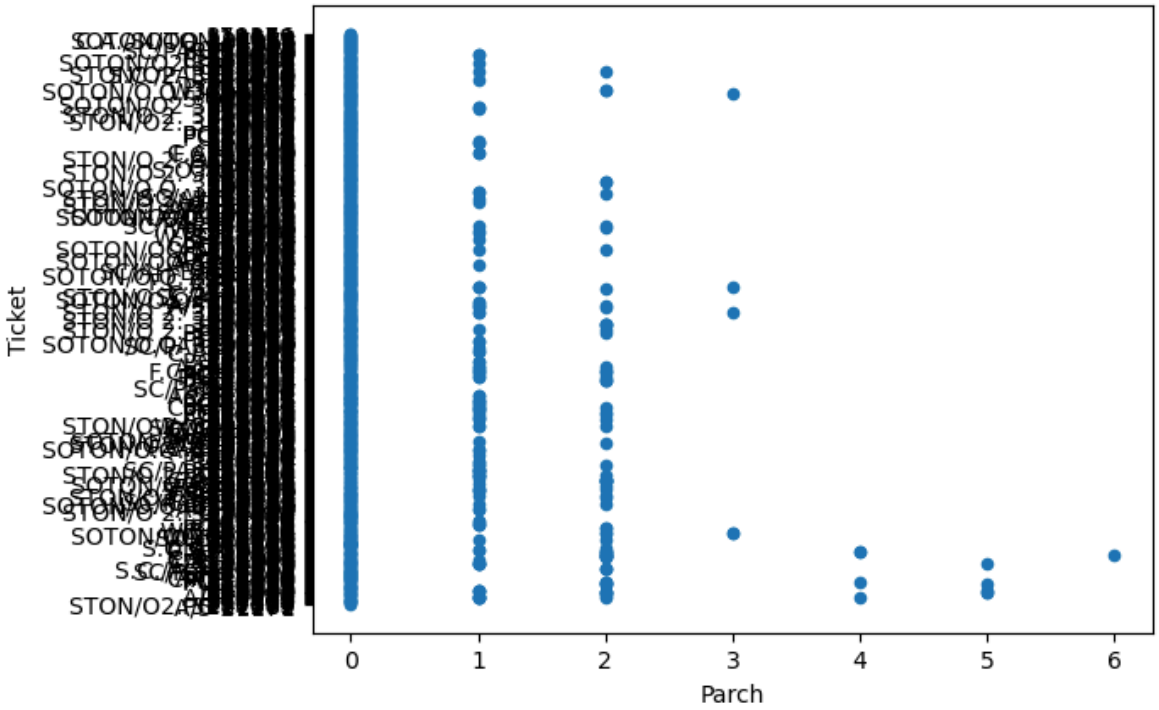
In [190…
```
df.plot.scatter(x="Age",y="Fare")
df.plot.scatter(x="PassengerId",y="Survived")
df.plot.scatter(x="Pclass",y="Sex")
df.plot.scatter(x="Parch",y="Ticket")
```

Out[190… <Axes: xlabel='Parch', ylabel='Ticket'>


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


Survived
PassengerId
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800


female
male
Sex
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
Pclass


Ticket
0
1
2
3
4
5
6
Parch

In [ ]: