3. Create a dataframe having at least 3 columns and 50 rows to store numeric data generated using a random function. Replace 10% of the values by null values whose index positions are generated using random function. Do the following:

In [6]:
```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randint(0,100,size=(50,3)), columns=list('123'))

for c in  df.sample(int(df.shape[0]*df.shape[1]*0.10)).index:
    df.loc[c,str(np.random.randint(1,4))]=np.nan
df
```

Out[6]:

	1	2	3
0	25.0	83.0	92.0
1	87.0	59.0	55.0
2	35.0	10.0	76.0
3	16.0	72.0	87.0
4	93.0	NaN	7.0
5	39.0	NaN	95.0
6	85.0	45.0	86.0
7	24.0	45.0	12.0
8	50.0	23.0	37.0
9	59.0	26.0	1.0
10	98.0	39.0	84.0
11	89.0	10.0	NaN
12	17.0	0.0	NaN
13	46.0	65.0	NaN
14	1.0	93.0	4.0
15	87.0	0.0	30.0
16	5.0	62.0	56.0
17	97.0	48.0	70.0
18	86.0	91.0	46.0
19	92.0	63.0	57.0
20	67.0	NaN	16.0
21	34.0	0.0	50.0
22	17.0	80.0	NaN
23	72.0	NaN	81.0
24	3.0	90.0	NaN
25	63.0	70.0	55.0
26	33.0	NaN	68.0
27	80.0	87.0	30.0
28	77.0	10.0	15.0
29	16.0	99.0	3.0
30	16.0	43.0	60.0
31	7.0	14.0	90.0
32	97.0	24.0	54.0
33	87.0	NaN	52.0
34	NaN	31.0	0.0
35	28.0	NaN	72.0
36	28.0	82.0	66.0
37	25.0	75.0	4.0
38	NaN	94.0	58.0

	1	2	3
39	32.0	30.0	77.0
40	8.0	2.0	30.0
41	10.0	63.0	15.0
42	76.0	51.0	88.0
43	4.0	21.0	26.0
44	1.0	0.0	15.0
45	31.0	10.0	56.0
46	67.0	93.0	56.0
47	40.0	13.0	46.0
48	58.0	68.0	77.0
49	NaN	53.0	56.0

a. Identify and count missing values in a dataframe.

In [7]:
```
print(df.isnull().sum().sum())
```

```
15
```

b. Drop the column having more than 5 null values.

In [8]:
```
for col in df.columns:
    print(col,df[col].isnull().sum())
df.dropna(axis = 1,thresh=(df.shape[0]-5)).head()
```

```
1 3
2 7
3 5
```

Out[8]:

	1	3
0	25.0	92.0
1	87.0	55.0
2	35.0	76.0
3	16.0	87.0
4	93.0	7.0

c. Identify the row label having maximum of the sum of all values in a row and drop that row.

In [10]:
```
sum=df.sum(axis=1)
print("SUM IS :\n",sum)
print("\nMAXIMUM SUM IS :",sum.max())
max_sum_row = df.sum(axis=1).idxmax()
print("\nRow Label having maximum sum is :" ,max_sum_row)

df = df.drop(max_sum_row ,axis =0)
print("\nDATA Frame AFTER REMOVING THE ROW HAVING MAXIMUM SUM VALUE")
df
```

```
SUM IS :
 0     200.0
1     201.0
2     121.0
3     175.0
4     100.0
5     134.0
6     216.0
7      81.0
8     110.0
9      86.0
10    221.0
11     99.0
12     17.0
13    111.0
14     98.0
15    117.0
16    123.0
17    215.0
19    212.0
20     83.0
21     84.0
22     97.0
23    153.0
24     93.0
25    188.0
26    101.0
27    197.0
28    102.0
29    118.0
30    119.0
31    111.0
32    175.0
33    139.0
34     31.0
35    100.0
36    176.0
37    104.0
38    152.0
39    139.0
40     40.0
41     88.0
42    215.0
43     51.0
44     16.0
45     97.0
46    216.0
47     99.0
48    203.0
49    109.0
dtype: float64

MAXIMUM SUM IS : 221.0

Row Label having maximum sum is : 10

DATA Frame AFTER REMOVING THE ROW HAVING MAXIMUM SUM VALUE
```

Out[10]:

	1	2	3
0	25.0	83.0	92.0
1	87.0	59.0	55.0
2	35.0	10.0	76.0
3	16.0	72.0	87.0
4	93.0	NaN	7.0
5	39.0	NaN	95.0
6	85.0	45.0	86.0
7	24.0	45.0	12.0
8	50.0	23.0	37.0
9	59.0	26.0	1.0
11	89.0	10.0	NaN
12	17.0	0.0	NaN
13	46.0	65.0	NaN
14	1.0	93.0	4.0
15	87.0	0.0	30.0
16	5.0	62.0	56.0
17	97.0	48.0	70.0
19	92.0	63.0	57.0
20	67.0	NaN	16.0
21	34.0	0.0	50.0
22	17.0	80.0	NaN
23	72.0	NaN	81.0
24	3.0	90.0	NaN
25	63.0	70.0	55.0
26	33.0	NaN	68.0
27	80.0	87.0	30.0
28	77.0	10.0	15.0
29	16.0	99.0	3.0
30	16.0	43.0	60.0
31	7.0	14.0	90.0
32	97.0	24.0	54.0
33	87.0	NaN	52.0
34	NaN	31.0	0.0
35	28.0	NaN	72.0
36	28.0	82.0	66.0
37	25.0	75.0	4.0
38	NaN	94.0	58.0
39	32.0	30.0	77.0
40	8.0	2.0	30.0

	1	2	3
41	10.0	63.0	15.0
42	76.0	51.0	88.0
43	4.0	21.0	26.0
44	1.0	0.0	15.0
45	31.0	10.0	56.0
46	67.0	93.0	56.0
47	40.0	13.0	46.0
48	58.0	68.0	77.0
49	NaN	53.0	56.0

d. Sort the dataframe on the basis of the first colum

In [13]:
```
sortdf=df.sort_values('1')
sortdf
```

Out[13]:

	1	2	3
44	1.0	0.0	15.0
14	1.0	93.0	4.0
24	3.0	90.0	NaN
43	4.0	21.0	26.0
16	5.0	62.0	56.0
31	7.0	14.0	90.0
40	8.0	2.0	30.0
41	10.0	63.0	15.0
3	16.0	72.0	87.0
30	16.0	43.0	60.0
29	16.0	99.0	3.0
12	17.0	0.0	NaN
22	17.0	80.0	NaN
7	24.0	45.0	12.0
37	25.0	75.0	4.0
0	25.0	83.0	92.0
36	28.0	82.0	66.0
35	28.0	NaN	72.0
45	31.0	10.0	56.0
39	32.0	30.0	77.0
26	33.0	NaN	68.0
21	34.0	0.0	50.0
2	35.0	10.0	76.0
5	39.0	NaN	95.0
47	40.0	13.0	46.0
13	46.0	65.0	NaN
8	50.0	23.0	37.0
48	58.0	68.0	77.0
9	59.0	26.0	1.0
25	63.0	70.0	55.0
20	67.0	NaN	16.0
46	67.0	93.0	56.0
23	72.0	NaN	81.0
42	76.0	51.0	88.0
28	77.0	10.0	15.0
27	80.0	87.0	30.0
6	85.0	45.0	86.0
15	87.0	0.0	30.0
33	87.0	NaN	52.0

	1	2	3
1	87.0	59.0	55.0
11	89.0	10.0	NaN
19	92.0	63.0	57.0
4	93.0	NaN	7.0
17	97.0	48.0	70.0
32	97.0	24.0	54.0
34	NaN	31.0	0.0
38	NaN	94.0	58.0
49	NaN	53.0	56.0

e. Remove all duplicates from the first column.

In [15]:
```
df =df.drop_duplicates(subset='1',keep = "first")
df
```

Out[15]:

	1	2	3
0	25.0	83.0	92.0
1	87.0	59.0	55.0
2	35.0	10.0	76.0
3	16.0	72.0	87.0
4	93.0	NaN	7.0
5	39.0	NaN	95.0
6	85.0	45.0	86.0
7	24.0	45.0	12.0
8	50.0	23.0	37.0
9	59.0	26.0	1.0
11	89.0	10.0	NaN
12	17.0	0.0	NaN
13	46.0	65.0	NaN
14	1.0	93.0	4.0
16	5.0	62.0	56.0
17	97.0	48.0	70.0
19	92.0	63.0	57.0
20	67.0	NaN	16.0
21	34.0	0.0	50.0
23	72.0	NaN	81.0
24	3.0	90.0	NaN
25	63.0	70.0	55.0
26	33.0	NaN	68.0
27	80.0	87.0	30.0
28	77.0	10.0	15.0
31	7.0	14.0	90.0
34	NaN	31.0	0.0
35	28.0	NaN	72.0
39	32.0	30.0	77.0
40	8.0	2.0	30.0
41	10.0	63.0	15.0
42	76.0	51.0	88.0
43	4.0	21.0	26.0
45	31.0	10.0	56.0
47	40.0	13.0	46.0
48	58.0	68.0	77.0

f. Find the correlation between first and second column and covariance between second and third column

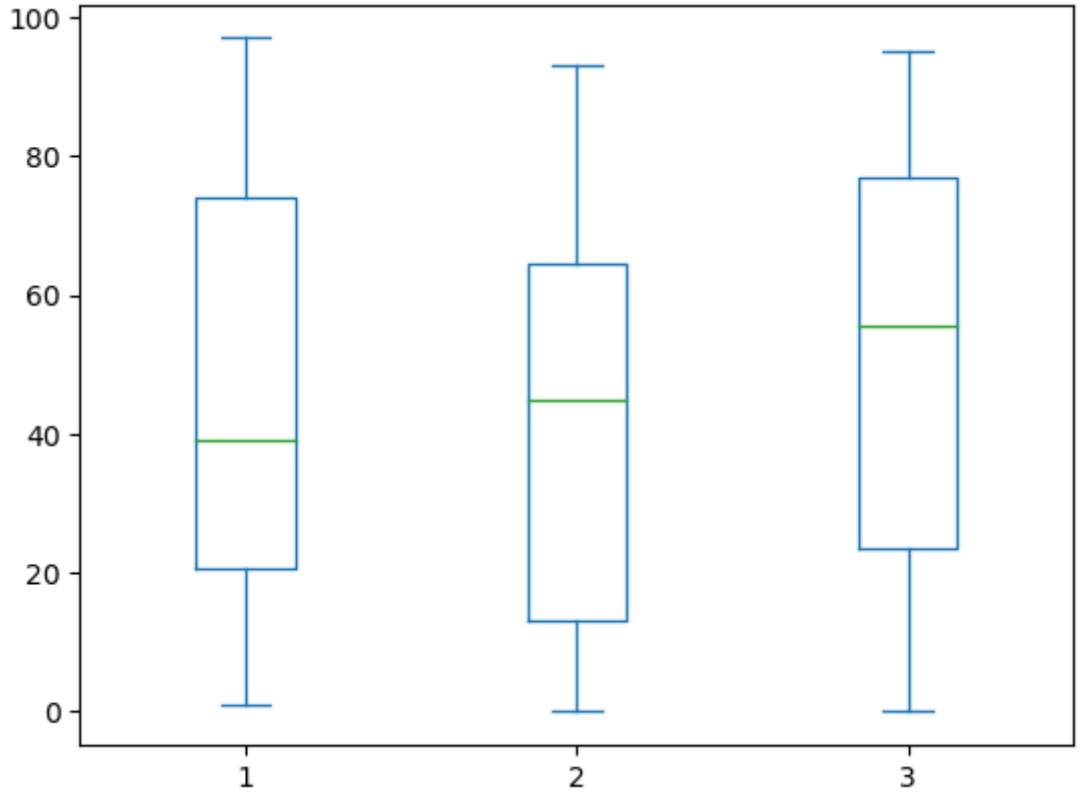
In [17]:
```
correlation = df['1'].corr(df['2'])
print("CORRELATION between column 1 and 2 : ", correlation)
covariance = df['2'].cov(df['3'])
print("COVARIANCE between column 2 and 3 :",covariance)
```

```
CORRELATION between column 1 and 2 :  0.025998863412796235
COVARIANCE between column 2 and 3 : 86.28923076923074
```

g. Detect the outliers and remove the rows having outliers.

In [19]:
```
df.plot.box()
```

Out[19]:
```
<AxesSubplot:>
```



h. Discretize second column and create 5 bins

In [20]:
```
df1 =  pd.cut(df['2'],bins=5).head()
df1
```

Out[20]:
```
0      (74.4, 93.0]
1      (55.8, 74.4]
2    (-0.093, 18.6]
3      (55.8, 74.4]
4               NaN
Name: 2, dtype: category
Categories (5, interval[float64, right]): [(-0.093, 18.6] < (18.6, 37.2] < (37.2, 55.8] < (5
5.8, 74.4] < (74.4, 93.0]]
```