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:
a. Identify and count missing values in a dataframe.
b. Drop the column having more than 5 null values.
c. Identify the row label having maximum of the sum of all values in a row and drop that row.
d. Sort the dataframe on the basis of the first column.
e. Remove all duplicates from the first column.
f. Find the correlation between first and second column and covariance between second and third column.
g. Detect the outliers and remove the rows having outliers.
h. Discretize second column and create 5 bins

In [264]:

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

Create a dataframe having at least 3 columns and 50 rows to store numeric data generated using a random function

data=np.random.randint(1,1000,250,int).reshape(50,5) data

In [265]:

```
df=pd.DataFrame(data,columns=['a','b','c','d','e'])
df2 = pd.DataFrame({df.columns[0]:[-1500, -90, -400],
                    df.columns[1]:[-1035, 2010, 2113],
                    df.columns[2]:[10, 2001, 2134],
                    df.columns[3]:[-210, -1114, 3134],
                    df.columns[4]:[1900, -314, 1034]})
df=pd.concat([df,df2],ignore_index=True)
df.T
```

Out[265]:

	0	1	2	3	4	5	6	7	8	9	...	43	44	45	46	47	48	49	
a	717	58	858	165	105	674	538	865	491	803	...	470	433	711	619	41	373	921	-1
b	296	665	459	610	83	697	849	513	411	683	...	338	252	479	229	709	583	558	-1
c	307	738	851	598	458	662	712	39	836	91	...	100	777	966	68	305	445	518	
d	794	258	432	603	259	361	418	828	365	286	...	479	225	587	91	790	354	999	-
e	119	854	660	751	949	971	641	636	899	506	...	454	689	917	211	991	812	973	1

5 rows × 53 columns

Replace 10% of the values by null values whose index positions are generated using random function

In [ ]:

```
```

In [266]:

```
for col in df.columns:
    df.loc[df.sample(n=5).index,col]=np.nan
```

A-> Identify and count missing values in a dataframe.

In [267]:

```
df.isna().sum()
```

Out[267]:

```
a    5
b    5
c    5
d    5
e    5
dtype: int64
```

B-> Drop the column having more than 5 null values.

In [268]:

```
df=df.drop('e',axis=1)
df
```

Out[268]:

	a	b	c	d
0	717.0	296.0	307.0	794.0
1	58.0	665.0	738.0	258.0
2	858.0	459.0	851.0	432.0
3	165.0	610.0	598.0	603.0
4	105.0	83.0	458.0	259.0
5	674.0	697.0	662.0	361.0
6	538.0	849.0	712.0	418.0
7	865.0	NaN	39.0	828.0
8	491.0	NaN	836.0	365.0
9	803.0	683.0	91.0	286.0
10	961.0	908.0	758.0	416.0
11	812.0	510.0	450.0	880.0
12	816.0	776.0	438.0	33.0
13	664.0	234.0	56.0	175.0
14	496.0	436.0	NaN	94.0
15	641.0	490.0	861.0	36.0
16	710.0	554.0	151.0	224.0
17	82.0	45.0	158.0	740.0
18	221.0	724.0	NaN	253.0
19	35.0	369.0	379.0	407.0
20	631.0	229.0	NaN	NaN
21	10.0	758.0	682.0	873.0
22	NaN	984.0	260.0	447.0
23	NaN	493.0	311.0	737.0
24	236.0	197.0	757.0	NaN
25	835.0	227.0	486.0	638.0
26	398.0	853.0	763.0	396.0
27	129.0	NaN	646.0	453.0
28	959.0	611.0	927.0	NaN
29	801.0	127.0	853.0	744.0
30	694.0	384.0	222.0	166.0
31	189.0	449.0	344.0	195.0
32	723.0	780.0	966.0	417.0
33	323.0	483.0	920.0	994.0
34	529.0	681.0	983.0	200.0
35	5.0	NaN	158.0	NaN

	a	b	c	d
36	NaN	909.0	126.0	460.0
37	499.0	552.0	212.0	226.0
38	177.0	442.0	768.0	805.0
39	96.0	682.0	950.0	474.0
40	753.0	908.0	637.0	621.0
41	NaN	128.0	416.0	956.0
42	456.0	NaN	935.0	450.0
43	470.0	338.0	100.0	479.0
44	433.0	252.0	777.0	225.0
45	711.0	479.0	966.0	587.0
46	619.0	229.0	68.0	91.0
47	41.0	709.0	NaN	790.0
48	373.0	583.0	NaN	354.0
49	921.0	558.0	518.0	NaN
50	-1500.0	-1035.0	10.0	-210.0
51	-90.0	2010.0	2001.0	-1114.0
52	NaN	2113.0	2134.0	3134.0

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

In [269]:

```
df.sum()
```

Out[269]:

```
a    21133.0
b    26501.0
c    28439.0
d    22450.0
dtype: float64
```

In [270]:

```
max_sum_col=df.columns[df.sum().argmax()]
max_sum_col
```

Out[270]:

```
'c'
```

In [271]:

```
df=df.drop(max_sum_col,axis=1)
df
```

Out[271]:

	a	b	d
0	717.0	296.0	794.0
1	58.0	665.0	258.0
2	858.0	459.0	432.0
3	165.0	610.0	603.0
4	105.0	83.0	259.0
5	674.0	697.0	361.0
6	538.0	849.0	418.0
7	865.0	NaN	828.0
8	491.0	NaN	365.0
9	803.0	683.0	286.0
10	961.0	908.0	416.0
11	812.0	510.0	880.0
12	816.0	776.0	33.0
13	664.0	234.0	175.0
14	496.0	436.0	94.0
15	641.0	490.0	36.0
16	710.0	554.0	224.0
17	82.0	45.0	740.0
18	221.0	724.0	253.0
19	35.0	369.0	407.0
20	631.0	229.0	NaN
21	10.0	758.0	873.0
22	NaN	984.0	447.0
23	NaN	493.0	737.0
24	236.0	197.0	NaN
25	835.0	227.0	638.0
26	398.0	853.0	396.0
27	129.0	NaN	453.0
28	959.0	611.0	NaN
29	801.0	127.0	744.0
30	694.0	384.0	166.0
31	189.0	449.0	195.0
32	723.0	780.0	417.0
33	323.0	483.0	994.0
34	529.0	681.0	200.0
35	5.0	NaN	NaN

	a	b	d
36	NaN	909.0	460.0
37	499.0	552.0	226.0
38	177.0	442.0	805.0
39	96.0	682.0	474.0
40	753.0	908.0	621.0
41	NaN	128.0	956.0
42	456.0	NaN	450.0
43	470.0	338.0	479.0
44	433.0	252.0	225.0
45	711.0	479.0	587.0
46	619.0	229.0	91.0
47	41.0	709.0	790.0
48	373.0	583.0	354.0
49	921.0	558.0	NaN
50	-1500.0	-1035.0	-210.0
51	-90.0	2010.0	-1114.0
52	NaN	2113.0	3134.0

D-> Sort the dataframe on the basis of the first column.

In [272]:

```
df.sort_values(by=df.columns[0])
```

Out[272]:

	a	b	d
50	-1500.0	-1035.0	-210.0
51	-90.0	2010.0	-1114.0
35	5.0	NaN	NaN
21	10.0	758.0	873.0
19	35.0	369.0	407.0
47	41.0	709.0	790.0
1	58.0	665.0	258.0
17	82.0	45.0	740.0
39	96.0	682.0	474.0
4	105.0	83.0	259.0
27	129.0	NaN	453.0
3	165.0	610.0	603.0
38	177.0	442.0	805.0
31	189.0	449.0	195.0
18	221.0	724.0	253.0
24	236.0	197.0	NaN
33	323.0	483.0	994.0
48	373.0	583.0	354.0
26	398.0	853.0	396.0
44	433.0	252.0	225.0
42	456.0	NaN	450.0
43	470.0	338.0	479.0
8	491.0	NaN	365.0
14	496.0	436.0	94.0
37	499.0	552.0	226.0
34	529.0	681.0	200.0
6	538.0	849.0	418.0
46	619.0	229.0	91.0
20	631.0	229.0	NaN
15	641.0	490.0	36.0
13	664.0	234.0	175.0
5	674.0	697.0	361.0
30	694.0	384.0	166.0
16	710.0	554.0	224.0
45	711.0	479.0	587.0
0	717.0	296.0	794.0

	a	b	d
32	723.0	780.0	417.0
40	753.0	908.0	621.0
29	801.0	127.0	744.0
9	803.0	683.0	286.0
11	812.0	510.0	880.0
12	816.0	776.0	33.0
25	835.0	227.0	638.0
2	858.0	459.0	432.0
7	865.0	NaN	828.0
49	921.0	558.0	NaN
28	959.0	611.0	NaN
10	961.0	908.0	416.0
22	NaN	984.0	447.0
23	NaN	493.0	737.0
36	NaN	909.0	460.0
41	NaN	128.0	956.0
52	NaN	2113.0	3134.0

E-> Remove all duplicates from the first column.

In [273]:

```
df.drop_duplicates(subset=['a'], keep='first', inplace=True)
df.duplicated('a').sum()
```

Out[273]:

```
0
```

F-> Find the correlation between first and second column and covariance between second and third column.

In [274]:

```
print("correlation b/w first n second column")
df[df.columns[0]].corr(df[df.columns[1]])
```

```
correlation b/w first n second column
```

Out[274]:

```
0.3414722023293018
```

In [275]:

```
print("Covariance b/w second n third column")
df[df.columns[1]].cov(df[df.columns[2]])
```

```
Covariance b/w second n third column
```

Out[275]:

```
-35030.11602564102
```

G-> Detect the outliers and remove the rows having outliers

In [ ]:

```

```

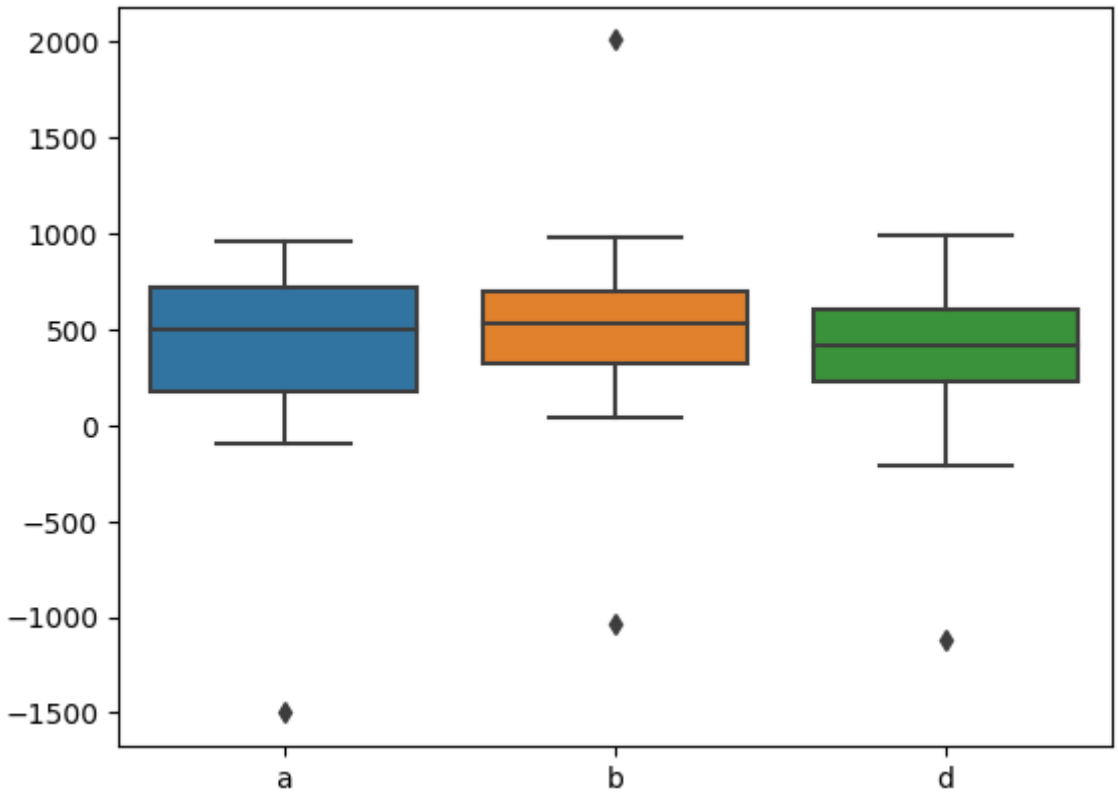
Outliers

In [276]:

```
sns.boxplot(df)
```

Out[276]:

```
<Axes: >
```



In [287]:

```
df.describe().T
```

Out[287]:

	count	mean	std	min	25%	50%	75%	max
a	48.0	440.270833	417.524760	-1500.0	174.00	497.5	718.5	961.0
b	44.0	519.500000	404.353029	-1035.0	327.50	531.0	700.0	2010.0
d	44.0	390.068182	353.499278	-1114.0	224.75	411.5	607.5	994.0

In [293]:

```
df=df[(df<1000) & (df>-500)]
```

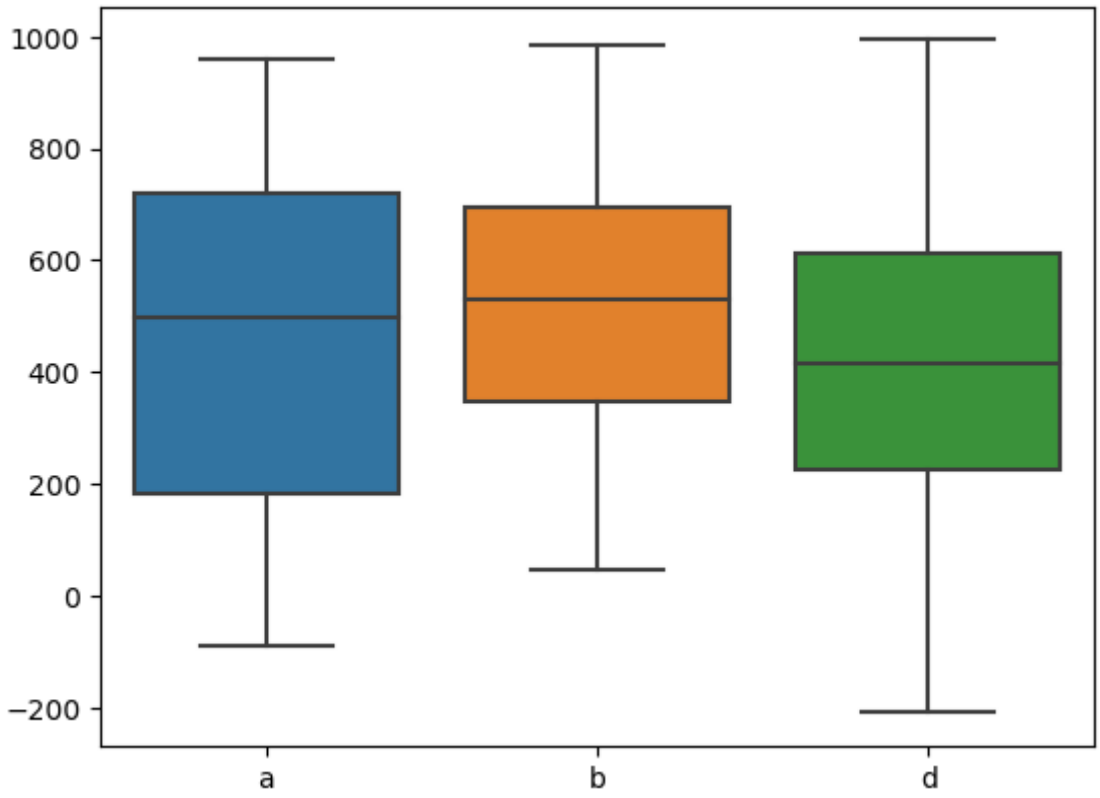
Outliers removed

In [294]:

```
sns.boxplot(df)
```

Out[294]:

```
<Axes: >
```



H-> Discretize second column and create 5 bins

In [304]:

```
# right = True, by default
pd.cut(df.b, bins=5, right=0).head()
```

Out[304]:

```
0    [232.8, 420.6)
1    [608.4, 796.2)
2    [420.6, 608.4)
3    [608.4, 796.2)
4     [45.0, 232.8)
Name: b, dtype: category
Categories (5, interval[float64, left]): [[45.0, 232.8) < [232.8, 420.
6) < [420.6, 608.4) < [608.4, 796.2) < [796.2, 984.939)]
```

In [ ]:

```
sns.barplot()
```