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

1.Create any Series and print the output

In [19]:
```
d = pd.Series([1,2,3,4,5,6,7,8])
d
```

Out[19]:
```
0    1
1    2
2    3
3    4
4    5
5    6
6    7
7    8
dtype: int64
```

2.Create any dataframe of 10x5 with few nan values and print the output

In [25]:
```
a = pd.DataFrame({
    "A":[1,2,3,4,5,6,7,8,9,10],
    "B":[9,8,7,6,5,4,3,2,1,0],
    "C":[1,3,2,4,np.nan,10,7,5,8,3],
    "D":[45,67,np.nan,546,23,576,879,123,567,124],
    "E":[546,867,np.nan,np.nan,567,879,432,435,897,546]
})
a
```

Out[25]:

	A	B	C	D	E
0	1	9	1.0	45.0	546.0
1	2	8	3.0	67.0	867.0
2	3	7	2.0	NaN	NaN
3	4	6	4.0	546.0	NaN
4	5	5	NaN	23.0	567.0
5	6	4	10.0	576.0	879.0
6	7	3	7.0	879.0	432.0
7	8	2	5.0	123.0	435.0
8	9	1	8.0	567.0	897.0
9	10	0	3.0	124.0	546.0

In [23]:
```
data= pd.DataFrame([1,2,3,4,np.nan])
data
```

Out[23]:

	0
0	1.0
1	2.0
2	3.0
3	4.0
4	NaN

3.Display top 7 and last 6 rows and print the output

In [21]:
```
d.head(7)
```

Out[21]:
```
2    3
3    4
4    5
5    6
6    7
7    8
dtype: int64
```

In [22]:
```
d.tail(6)
```

Out[22]:
```
2    3
3    4
4    5
5    6
6    7
7    8
dtype: int64
```

4. Fill with a constant value and print the output

In [26]:
```
a.fillna(1)
```

Out[26]:

	A	B	C	D	E
0	1	9	1.0	45.0	546.0
1	2	8	3.0	67.0	867.0
2	3	7	2.0	1.0	1.0

	A	B	C	D	E
3	4	6	4.0	546.0	1.0
4	5	5	1.0	23.0	567.0
5	6	4	10.0	576.0	879.0
6	7	3	7.0	879.0	432.0
7	8	2	5.0	123.0	435.0
8	9	1	8.0	567.0	897.0

5.Drop the column with missing values and print the output

In [36]:
```
b = pd.DataFrame({
    "A":[1,2,3,4,5,6,7,8,9,10],
    "B":[9,8,7,6,5,4,3,2,1,0],
    "C":[1,3,2,4,np.nan,10,7,5,8,3],
    "D":[45,67,np.nan,546,23,576,879,123,567,124],
    "E":[546,867,np.nan,np.nan,567,879,432,435,897,546]
})
b.dropna(axis=0)
```

Out[36]:

	A	B	C	D	E
0	1	9	1.0	45.0	546.0
1	2	8	3.0	67.0	867.0
5	6	4	10.0	576.0	879.0
6	7	3	7.0	879.0	432.0
7	8	2	5.0	123.0	435.0
8	9	1	8.0	567.0	897.0
9	10	0	3.0	124.0	546.0

6. Drop the row with missing values and print the output

In [38]:
```
b.dropna()
```

Out[38]:

	A	B	C	D	E
0	1	9	1.0	45.0	546.0
1	2	8	3.0	67.0	867.0
5	6	4	10.0	576.0	879.0

	A	B	C	D	E
6	7	3	7.0	879.0	432.0
7	8	2	5.0	123.0	435.0
8	9	1	8.0	567.0	897.0

7.To check the presence of missing values in your dataframe

In [39]:
```
np.isnan(b)
```

Out[39]:

	A	B	C	D	E
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	True	True
3	False	False	False	False	True
4	False	False	True	False	False
5	False	False	False	False	False
6	False	False	False	False	False
7	False	False	False	False	False
8	False	False	False	False	False
9	False	False	False	False	False

8.Use operators and check the condition and print the output

In [43]:
```
b[b["A"]>=5]
```

Out[43]:

	A	B	C	D	E
4	5	5	NaN	23.0	567.0
5	6	4	10.0	576.0	879.0
6	7	3	7.0	879.0	432.0
7	8	2	5.0	123.0	435.0
8	9	1	8.0	567.0	897.0
9	10	0	3.0	124.0	546.0

9. Display your output using loc and iloc, row and column heading

In [59]:
```
d = pd.DataFrame({
    "A":[1,2,3,4,5,6,7,8,9,10],
    "B":[9,"sree",7,6,5,4,3,2,1,0],
    "C":[1,3,2,4,np.nan,10,7,5,8,3],
    "D":[45,67,np.nan,546,23,576,879,123,567,124],
    "E":[546,867,np.nan,np.nan,567,879,432,435,897,546]
})
d.loc["A":"C"]
d.columns
```

Out[59]:
```
Index(['A', 'B', 'C', 'D', 'E'], dtype='object')
```

In [60]:
```
d.index
```

Out[60]:
```
RangeIndex(start=0, stop=10, step=1)
```

In [61]:
```
d.iloc[0:4]
```

Out[61]:

	A	B	C	D	E
0	1	9	1.0	45.0	546.0
1	2	sree	3.0	67.0	867.0
2	3	7	2.0	NaN	NaN
3	4	6	4.0	546.0	NaN

10. Display the statistical summary of data

In [63]:
```
d.describe()
```

Out[63]:

	A	C	D	E
count	10.00000	9.000000	9.000000	8.000000
mean	5.50000	4.777778	327.777778	646.125000
std	3.02765	2.990726	315.163758	200.958018
min	1.00000	1.000000	23.000000	432.000000
25%	3.25000	3.000000	67.000000	518.250000
50%	5.50000	4.000000	124.000000	556.500000
75%	7.75000	7.000000	567.000000	870.000000

A	C	D	E

In [ ]: