Heamnath N

20104028

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

1.Create any Series and print the output

In [2]:
```
a=pd.Series([1,2,3,4,5,6,7,8,9,10])
a
```

Out[2]:
```
0     1
1     2
2     3
3     4
4     5
5     6
6     7
7     8
8     9
9    10
dtype: int64
```

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

In [3]:
```
df=pd.DataFrame(
{
    "a":[1,2,3,np.nan,5,6,7,np.nan,9,10] ,
    "b":[np.nan,4,5,6,7,1,2,3,1,2] ,
    "c":[1,2,2,3,6,5,8,6,4,1] ,
    "d":[1,22,3,3,4,np.nan,np.nan,4,2,5] ,
    "e":[1,2,3,4,5,6,7,8,9,10]

}
)
df
```

Out[3]:

	a	b	c	d	e
0	1.0	NaN	1	1.0	1
1	2.0	4.0	2	22.0	2
2	3.0	5.0	2	3.0	3
3	NaN	6.0	3	3.0	4

	a	b	c	d	e
4	5.0	7.0	6	4.0	5
5	6.0	1.0	5	NaN	6
6	7.0	2.0	8	NaN	7
7	NaN	3.0	6	4.0	8
8	9.0	1.0	4	2.0	9

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

In [4]:
```
c=pd.DataFrame(
{
    "a":np.empty(20,dtype='int64'),
    "b":np.empty(20,dtype='int64'),
    "c":np.empty(20,dtype='int64'),
    "d":np.empty(20,dtype='int64')

})
c.head(7)
```

Out[4]:

	a	b	c	d
0	25895968444448860	25895968444448860	261993005153	22518393277644867
1	22518393277644867	22518393277644867	429496729712	32088589733920882
2	32088589733920882	32088589733920882	356482285614	27303364805853281
3	27303364805853281	27303364805853281	489626271845	18296268629540980
4	18296268629540980	18296268629540980	433791697001	31244147623002222
5	31244147623002222	31244147623002222	171798691955	14355640430624878
6	14355640430624878	14355640430624878	210453397595	27584998696288348

In [5]:
```
c.tail(6)
```

Out[5]:

	a	b	c	d
14	27866538097049692	27866417837965426	244813135916	32651591226294388
15	26740517931581541	32933053318627439	210453397548	12948291317530729
16	33496016156360815	29555370777182324	399431958576	15762817746468963
17	28429470870863986	31244117559083119	42949673001	30962698417537069
18	34058953221341298	31525394967625840	2464988757464449121	28147965828857951
19	0	12948256950583417	8319683848551211564	31525394963497014

4. Fill with a constant value and print the output

In [6]:
```
df=pd.DataFrame(
{
    "a":[1,2,3,np.nan,5,6,7,np.nan,9,10] ,
    "b":[np.nan,4,5,6,7,1,2,3,1,2] ,
    "c":[1,2,2,3,6,5,8,6,4,1] ,
    "d":[1,22,3,3,4,np.nan,np.nan,4,2,5] ,
    "e":[1,2,3,4,5,6,7,8,9,10]

}
)
df.fillna(value=0)
```

Out[6]:

	a	b	c	d	e
0	1.0	0.0	1	1.0	1
1	2.0	4.0	2	22.0	2
2	3.0	5.0	2	3.0	3
3	0.0	6.0	3	3.0	4
4	5.0	7.0	6	4.0	5
5	6.0	1.0	5	0.0	6
6	7.0	2.0	8	0.0	7
7	0.0	3.0	6	4.0	8
8	9.0	1.0	4	2.0	9
9	10.0	2.0	1	5.0	10

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

In [7]:
```
df=pd.DataFrame(
{
    "a":[1,2,3,np.nan,5,6,7,np.nan,9,10] ,
    "b":[np.nan,4,5,6,7,1,2,3,1,2] ,
    "c":[1,2,2,3,6,5,8,6,4,1] ,
    "d":[1,22,3,3,4,np.nan,np.nan,4,2,5] ,
    "e":[1,2,3,4,5,6,7,8,9,10]

}
)
df.isna()
```

Out[7]:

	a	b	c	d	e
0	False	True	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False

	a	b	c	d	e
3	True	False	False	False	False
4	False	False	False	False	False
5	False	False	False	True	False
6	False	False	False	True	False
7	True	False	False	False	False
8	False	False	False	False	False

In [8]:
```
df=pd.DataFrame(
{
    "a":[1,2,3,np.nan,5,6,7,np.nan,9,10] ,
    "b":[np.nan,4,5,6,7,1,2,3,1,2] ,
    "c":[1,2,2,3,6,5,8,6,4,1] ,
    "d":[1,22,3,3,4,np.nan,np.nan,4,2,5] ,
    "e":[1,2,3,4,5,6,7,8,9,10]

}
)
df.dropna(axis=1)
```

Out[8]:

	c	e
0	1	1
1	2	2
2	2	3
3	3	4
4	6	5
5	5	6
6	8	7
7	6	8
8	4	9
9	1	10

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

In [9]:
```
df=pd.DataFrame(
{
    "a":[1,2,3,np.nan,5,6,7,np.nan,9,10] ,
    "b":[np.nan,4,5,6,7,1,2,3,1,2] ,
    "c":[1,2,2,3,6,5,8,6,4,1] ,
    "d":[1,22,3,3,4,np.nan,np.nan,4,2,5] ,
    "e":[1,2,3,4,5,6,7,8,9,10]

}
)
df.dropna()
```

Out[9]:

	a	b	c	d	e
1	2.0	4.0	2	22.0	2
2	3.0	5.0	2	3.0	3
4	5.0	7.0	6	4.0	5
8	9.0	1.0	4	2.0	9
9	10.0	2.0	1	5.0	10

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

In [10]:
```
df=pd.DataFrame(
{
    "a":[1,2,3,np.nan,5,6,7,np.nan,9,10] ,
    "b":[np.nan,4,5,6,7,1,2,3,1,2] ,
    "c":[1,2,2,3,6,5,8,6,4,1] ,
    "d":[1,22,3,3,4,np.nan,np.nan,4,2,5] ,
    "e":[1,2,3,4,5,6,7,8,9,10]

}
)
df.isna()
```

Out[10]:

	a	b	c	d	e
0	False	True	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	True	False	False	False	False
4	False	False	False	False	False
5	False	False	False	True	False
6	False	False	False	True	False
7	True	False	False	False	False

	a	b	c	d	e
8	False	False	False	False	False

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

In [11]:
```
df=pd.DataFrame(
{
    "a":[1,2,3,4,5,6,7,8,9,10] ,
    "b":[9,4,5,6,7,1,2,3,1,2] ,
    "c":[1,2,2,3,6,5,8,6,4,1] ,
    "d":[1,22,3,3,4,5,6,4,2,5] ,
    "e":[1,2,3,4,5,6,7,8,9,10]

}
)
df[df["a"]>2]
```

Out[11]:

	a	b	c	d	e
2	3	5	2	3	3
3	4	6	3	3	4
4	5	7	6	4	5
5	6	1	5	5	6
6	7	2	8	6	7
7	8	3	6	4	8
8	9	1	4	2	9
9	10	2	1	5	10

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

In [12]:
```
df=pd.DataFrame(
{
    "a":[1,2,3,4,5,6,7,8,9,10] ,
    "b":[9,4,5,6,7,1,2,3,1,2] ,
    "c":[1,2,2,3,6,5,8,6,4,1] ,
    "d":[1,22,3,3,4,5,6,4,2,5] ,
    "e":[1,2,3,4,5,6,7,8,9,10]

}
)
df.loc[0:2]
```

Out[12]:

	a	b	c	d	e
0	1	9	1	1	1
1	2	4	2	22	2
2	3	5	2	3	3

In [13]:
```
df.iloc[0:5]
```

Out[13]:

	a	b	c	d	e
0	1	9	1	1	1
1	2	4	2	22	2
2	3	5	2	3	3
3	4	6	3	3	4
4	5	7	6	4	5

In [14]:
```
df.columns
```

Out[14]:
```
Index(['a', 'b', 'c', 'd', 'e'], dtype='object')
```

In [15]:
```
df.index
```

Out[15]:
```
RangeIndex(start=0, stop=10, step=1)
```

10. Display the statistical summary of data

In [16]:
```
df=pd.DataFrame(
{
    "a":[1,2,3,4,5,6,7,8,9,10] ,
    "b":[9,4,5,6,7,1,2,3,1,2] ,
    "c":[1,2,2,3,6,5,8,6,4,1] ,
    "d":[1,22,3,3,4,5,6,4,2,5] ,
    "e":[1,2,3,4,5,6,7,8,9,10]

}
)
df.describe()
```

Out[16]:

	a	b	c	d	e
count	10.00000	10.000000	10.000000	10.000000	10.00000
mean	5.50000	4.000000	3.800000	5.500000	5.50000
std	3.02765	2.708013	2.394438	5.986095	3.02765
min	1.00000	1.000000	1.000000	1.000000	1.00000

	a	b	c	d	e
25%	3.25000	2.000000	2.000000	3.000000	3.25000
50%	5.50000	3.500000	3.500000	4.000000	5.50000
75%	7.75000	5.750000	5.750000	5.000000	7.75000