# SANTHOSH GOPI B

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

## 1. Create any Series and print the output

In [2]:
```
aa =pd.Series([1,2,3,4,5])
aa
```

Out[2]:
```
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

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

In [3]:
```
df = pd.DataFrame(np.random.randn(10,5))
df
```

Out[3]:

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| **0** | -0.013839 | -0.525329 | 0.208437 | -0.673406 | 0.471206 |
| **1** | -0.291022 | -1.576789 | 1.849250 | -0.661912 | -0.745451 |
| **2** | -0.581851 | 1.436894 | 0.274813 | 0.125788 | -0.772750 |
| **3** | -1.119815 | -1.117697 | -0.328377 | 2.831505 | -0.263338 |
| **4** | 0.831094 | -0.393743 | -0.826535 | 0.495945 | 1.598672 |
| **5** | 1.648602 | -0.520315 | 0.287161 | 0.864881 | -1.023753 |
| **6** | 0.231970 | 1.684926 | 0.411361 | 1.116783 | 2.474340 |
| **7** | 0.073293 | -1.793142 | -0.498066 | 0.520871 | -1.648530 |
| **8** | -0.488041 | -1.134718 | 0.430956 | -0.355409 | 0.637969 |
| **9** | 0.145443 | 0.297327 | 0.021506 | -0.158999 | -0.167570 |

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

In [4]:
```
df.head(7)
```

Out[4]:

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| **0** | -0.013839 | -0.525329 | 0.208437 | -0.673406 | 0.471206 |

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| **1** | -0.291022 | -1.576789 | 1.849250 | -0.661912 | -0.745451 |
| **2** | -0.581851 | 1.436894 | 0.274813 | 0.125788 | -0.772750 |
| **3** | -1.119815 | -1.117697 | -0.328377 | 2.831505 | -0.263338 |
| **4** | 0.831094 | -0.393743 | -0.826535 | 0.495945 | 1.598672 |
| **5** | 1.648602 | -0.520315 | 0.287161 | 0.864881 | -1.023753 |
| **6** | 0.231970 | 1.684926 | 0.411361 | 1.116783 | 2.474340 |

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

Out[5]:

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| **4** | 0.831094 | -0.393743 | -0.826535 | 0.495945 | 1.598672 |
| **5** | 1.648602 | -0.520315 | 0.287161 | 0.864881 | -1.023753 |
| **6** | 0.231970 | 1.684926 | 0.411361 | 1.116783 | 2.474340 |
| **7** | 0.073293 | -1.793142 | -0.498066 | 0.520871 | -1.648530 |
| **8** | -0.488041 | -1.134718 | 0.430956 | -0.355409 | 0.637969 |
| **9** | 0.145443 | 0.297327 | 0.021506 | -0.158999 | -0.167570 |

# 4. Fill with a constant value and print the output

In [31]:
```
df1=pd.DataFrame(
{
    "A":[1,2,3,4,5,6,7,8,9,10],
    "B":[7,8,9,np.nan,np.nan,1,2,3,6,7],
    "C":[2,3,4,5,np.nan,np.nan,np.nan,8,9,10],
    "D":[3,4,6,7,8,np.nan,np.nan,np.nan,3,4],

})
np.isnan(df1)
```

Out[31]:

| | A | B | C | D |
|---|---|---|---|---|
| **0** | False | False | False | False |
| **1** | False | False | False | False |
| **2** | False | False | False | False |
| **3** | False | True | False | False |
| **4** | False | True | True | False |
| **5** | False | False | True | True |
| **6** | False | False | True | True |
| **7** | False | False | False | True |

| | A | B | C | D |
|---|---|---|---|---|
| **8** | False | False | False | False |
| **9** | False | False | False | False |

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

In [13]:
```
df.columns
df.drop
```

Out[13]:
```
<bound method DataFrame.drop of           0         1         2         3         4
0 -0.013839 -0.525329  0.208437 -0.673406  0.471206
1 -0.291022 -1.576789  1.849250 -0.661912 -0.745451
2 -0.581851  1.436894  0.274813  0.125788 -0.772750
3 -1.119815 -1.117697 -0.328377  2.831505 -0.263338
4  0.831094 -0.393743 -0.826535  0.495945  1.598672
5  1.648602 -0.520315  0.287161  0.864881 -1.023753
6  0.231970  1.684926  0.411361  1.116783  2.474340
7  0.073293 -1.793142 -0.498066  0.520871 -1.648530
8 -0.488041 -1.134718  0.430956 -0.355409  0.637969
9  0.145443  0.297327  0.021506 -0.158999 -0.167570>
```

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

In [18]:
```
df.index
df.drop
```

Out[18]:
```
<bound method DataFrame.drop of           0         1         2         3         4
0 -0.013839 -0.525329  0.208437 -0.673406  0.471206
1 -0.291022 -1.576789  1.849250 -0.661912 -0.745451
2 -0.581851  1.436894  0.274813  0.125788 -0.772750
3 -1.119815 -1.117697 -0.328377  2.831505 -0.263338
4  0.831094 -0.393743 -0.826535  0.495945  1.598672
5  1.648602 -0.520315  0.287161  0.864881 -1.023753
6  0.231970  1.684926  0.411361  1.116783  2.474340
7  0.073293 -1.793142 -0.498066  0.520871 -1.648530
8 -0.488041 -1.134718  0.430956 -0.355409  0.637969
9  0.145443  0.297327  0.021506 -0.158999 -0.167570>
```

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

In [19]:
```
df.isna
```

Out[19]:
```
<bound method DataFrame.isna of           0         1         2         3         4
0 -0.013839 -0.525329  0.208437 -0.673406  0.471206
1 -0.291022 -1.576789  1.849250 -0.661912 -0.745451
2 -0.581851  1.436894  0.274813  0.125788 -0.772750
3 -1.119815 -1.117697 -0.328377  2.831505 -0.263338
4  0.831094 -0.393743 -0.826535  0.495945  1.598672
5  1.648602 -0.520315  0.287161  0.864881 -1.023753
```

```
6  0.231970  1.684926  0.411361  1.116783  2.474340
7  0.073293 -1.793142 -0.498066  0.520871 -1.648530
8 -0.488041 -1.134718  0.430956 -0.355409  0.637969
9  0.145443  0.297327  0.021506 -0.158999 -0.167570>
```

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

In [37]:
```
df1=pd.DataFrame(
{
    "A":[1,2,3,4,5,6,7,8,9,10],
    "B":[7,8,9,8,5,1,2,3,6,7],
    "C":[2,3,4,5,6,7,8,9,7,10],
    "D":[3,4,6,7,8,9,6,5,3,4],

})
df1[df1["A"]>7]
```

Out[37]:

| | A | B | C | D |
|---|---|---|---|---|
| **7** | 8 | 3 | 9 | 5 |
| **8** | 9 | 6 | 7 | 3 |
| **9** | 10 | 7 | 10 | 4 |

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

In [39]:
```
df1=pd.DataFrame(
{
    "A":[1,2,3,4,5,6,7,8,9,10],
    "B":[7,8,9,8,5,1,2,3,6,7],
    "C":[2,3,4,5,6,7,8,9,7,10],
    "D":[3,4,6,7,8,9,6,5,3,4],
})
```

In [42]:
```
df1.loc[2:5]
```

Out[42]:

| | A | B | C | D |
|---|---|---|---|---|
| **2** | 3 | 9 | 4 | 6 |
| **3** | 4 | 8 | 5 | 7 |
| **4** | 5 | 5 | 6 | 8 |
| **5** | 6 | 1 | 7 | 9 |

In [43]:
```
df1.iloc[2:5]
```

Out[43]:

| | A | B | C | D |
|---|---|---|---|---|
| **2** | 3 | 9 | 4 | 6 |
| **3** | 4 | 8 | 5 | 7 |
| **4** | 5 | 5 | 6 | 8 |

In [48]:
```
df1.index
```

Out[48]:
```
RangeIndex(start=0, stop=10, step=1)
```

In [47]:
```
df1.columns
```

Out[47]:
```
Index(['A', 'B', 'C', 'D'], dtype='object')
```

# 10. Display the statistical summary of data

In [49]:
```
df1=pd.DataFrame(
{
    "A":[1,2,3,4,5,6,7,8,9,10],
    "B":[7,8,9,8,5,1,2,3,6,7],
    "C":[2,3,4,5,6,7,8,9,7,10],
    "D":[3,4,6,7,8,9,6,5,3,4],
})
df1.describe
```

Out[49]:
```
<bound method NDFrame.describe of     A  B   C  D
0   1  7   2  3
1   2  8   3  4
2   3  9   4  6
3   4  8   5  7
4   5  5   6  8
5   6  1   7  9
6   7  2   8  6
7   8  3   9  5
8   9  6   7  3
9  10  7  10  4>
```

In [ ]: