# SUMESH R - 20104169

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])
a
```

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]:
```
data = pd.DataFrame({
    "a": [1,2,3,4,5,6,7,8,9,10],
    "b": [6,7,8,9,0,np.nan,np.nan,1,2,3],
    "c": [1,4,6,8,4,2,np.nan,np.nan,np.nan,2],
    "d": [3,5,6,np.nan,4,1,7,8,np.nan,np.nan],
    "e": [5,np.nan,3,4,5,np.nan,np.nan,np.nan,2,1]
})
data
```

Out[3]:

| | a | b | c | d | e |
|---|---|---|---|---|---|
| **0** | 1 | 6.0 | 1.0 | 3.0 | 5.0 |
| **1** | 2 | 7.0 | 4.0 | 5.0 | NaN |
| **2** | 3 | 8.0 | 6.0 | 6.0 | 3.0 |
| **3** | 4 | 9.0 | 8.0 | NaN | 4.0 |
| **4** | 5 | 0.0 | 4.0 | 4.0 | 5.0 |
| **5** | 6 | NaN | 2.0 | 1.0 | NaN |
| **6** | 7 | NaN | NaN | 7.0 | NaN |
| **7** | 8 | 1.0 | NaN | 8.0 | NaN |
| **8** | 9 | 2.0 | NaN | NaN | 2.0 |
| **9** | 10 | 3.0 | 2.0 | NaN | 1.0 |

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

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

In [5]:
```
data.head(7)
```

Out[5]:

| | a | b | c | d |
|---|---|---|---|---|
| **0** | 4617315517961601024 | 25895968444448860 | 22518393277644867 | 25895968444448860 |
| **1** | 0 | 22518393277644867 | 32088589733920882 | 22518393277644867 |
| **2** | 9221120237041090560 | 32088589733920882 | 27303364805853281 | 32088589733920882 |
| **3** | 0 | 27303364805853281 | 18296268629540980 | 27303364805853281 |
| **4** | 4613937818241073152 | 18296268629540980 | 31244147623002222 | 18296268629540980 |
| **5** | 0 | 31244147623002222 | 14355640430624878 | 31244147623002222 |
| **6** | 4616189618054758400 | 14355640430624878 | 27584998696288348 | 14355640430624878 |

In [6]:
```
data.tail(6)
```

Out[6]:

| | a | b | c | d |
|---|---|---|---|---|
| **14** | 9221120237041090560 | 33777431003005033 | 32370111954354288 | 25896191785042025 |
| **15** | 0 | 32370094774747252 | 12948342857400421 | 30962724187078762 |
| **16** | 4611686018427387904 | 32370094775402601 | 15762817746468963 | 14355704855068718 |
| **17** | 0 | 29555280582738012 | 30962698417537069 | 29555383658676280 |
| **18** | 4607182418800017408 | 26740621010600046 | 28147965828857951 | 30681189079515246 |
| **19** | 0 | 34058953221341279 | 31525394963497014 | 12948072270528612 |

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

In [7]:
```
d = pd.DataFrame({
    "a": [1,2,3,4,5,6,7,8,9,10],
    "b": [6,7,8,9,0,np.nan,np.nan,1,2,3],
    "c": [1,4,6,8,4,2,np.nan,np.nan,np.nan,2],
    "d": [3,5,6,np.nan,4,1,7,8,np.nan,np.nan],
```

```
    "e": [5,np.nan,3,4,5,np.nan,np.nan,np.nan,2,1]
})
np.isnan(d)
```

Out[7]:

| | a | b | c | d | e |
|---|---|---|---|---|---|
| **0** | False | False | False | False | False |
| **1** | False | False | False | False | True |
| **2** | False | False | False | False | False |
| **3** | False | False | False | True | False |
| **4** | False | False | False | False | False |
| **5** | False | True | False | False | True |
| **6** | False | True | True | False | True |
| **7** | False | False | True | False | True |
| **8** | False | False | True | True | False |
| **9** | False | False | False | True | False |

In [8]:
```
d.fillna(5)
```

Out[8]:

| | a | b | c | d | e |
|---|---|---|---|---|---|
| **0** | 1 | 6.0 | 1.0 | 3.0 | 5.0 |
| **1** | 2 | 7.0 | 4.0 | 5.0 | 5.0 |
| **2** | 3 | 8.0 | 6.0 | 6.0 | 3.0 |
| **3** | 4 | 9.0 | 8.0 | 5.0 | 4.0 |
| **4** | 5 | 0.0 | 4.0 | 4.0 | 5.0 |
| **5** | 6 | 5.0 | 2.0 | 1.0 | 5.0 |
| **6** | 7 | 5.0 | 5.0 | 7.0 | 5.0 |
| **7** | 8 | 1.0 | 5.0 | 8.0 | 5.0 |
| **8** | 9 | 2.0 | 5.0 | 5.0 | 2.0 |
| **9** | 10 | 3.0 | 2.0 | 5.0 | 1.0 |

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

In [9]:
```
d = pd.DataFrame({
    "a": [1,2,3,4,5,6,7,8,9,10],
    "b": [6,7,8,9,0,np.nan,np.nan,1,2,3],
    "c": [1,4,6,8,4,2,np.nan,np.nan,np.nan,2],
    "d": [3,5,6,np.nan,4,1,7,8,np.nan,np.nan],
    "e": [5,np.nan,3,4,5,np.nan,np.nan,np.nan,2,1]
```

```
})
np.isnan(d)
```

Out[9]:

| | a | b | c | d | e |
|---|---|---|---|---|---|
| **0** | False | False | False | False | False |
| **1** | False | False | False | False | True |
| **2** | False | False | False | False | False |
| **3** | False | False | False | True | False |
| **4** | False | False | False | False | False |
| **5** | False | True | False | False | True |
| **6** | False | True | True | False | True |
| **7** | False | False | True | False | True |
| **8** | False | False | True | True | False |
| **9** | False | False | False | True | False |

In [10]:

```
d.dropna(axis="columns")
```

Out[10]:

| | a |
|---|---|
| **0** | 1 |
| **1** | 2 |
| **2** | 3 |
| **3** | 4 |
| **4** | 5 |
| **5** | 6 |
| **6** | 7 |
| **7** | 8 |
| **8** | 9 |
| **9** | 10 |

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

In [11]:

```
d = pd.DataFrame({
    "a": [1,2,3,4,5,6,7,8,9,10],
    "b": [6,7,8,9,0,np.nan,np.nan,1,2,3],
    "c": [1,4,6,8,4,2,np.nan,np.nan,np.nan,2],
    "d": [3,5,6,np.nan,4,1,7,8,np.nan,np.nan],
    "e": [5,np.nan,3,4,5,np.nan,np.nan,np.nan,2,1]
})
np.isnan(d)
```

Out[11]:

| | a | b | c | d | e |
|---|---|---|---|---|---|
| **0** | False | False | False | False | False |
| **1** | False | False | False | False | True |
| **2** | False | False | False | False | False |
| **3** | False | False | False | True | False |
| **4** | False | False | False | False | False |
| **5** | False | True | False | False | True |
| **6** | False | True | True | False | True |
| **7** | False | False | True | False | True |
| **8** | False | False | True | True | False |
| **9** | False | False | False | True | False |

In [12]:
```
d.dropna()
```

Out[12]:

| | a | b | c | d | e |
|---|---|---|---|---|---|
| **0** | 1 | 6.0 | 1.0 | 3.0 | 5.0 |
| **2** | 3 | 8.0 | 6.0 | 6.0 | 3.0 |
| **4** | 5 | 0.0 | 4.0 | 4.0 | 5.0 |

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

In [13]:
```
d = pd.DataFrame({
    "a": [1,2,3,4,5,6,7,8,9,10],
    "b": [6,7,8,9,0,np.nan,np.nan,1,2,3],
    "c": [1,4,6,8,4,2,np.nan,np.nan,np.nan,2],
    "d": [3,5,6,np.nan,4,1,7,8,np.nan,np.nan],
    "e": [5,np.nan,3,4,5,np.nan,np.nan,np.nan,2,1]
})
```

In [14]:
```
d.isna()
```

Out[14]:

| | a | b | c | d | e |
|---|---|---|---|---|---|
| **0** | False | False | False | False | False |
| **1** | False | False | False | False | True |
| **2** | False | False | False | False | False |
| **3** | False | False | False | True | False |
| **4** | False | False | False | False | False |

| | a | b | c | d | e |
|---|---|---|---|---|---|
| **5** | False | True | False | False | True |
| **6** | False | True | True | False | True |
| **7** | False | False | True | False | True |
| **8** | False | False | True | True | False |
| **9** | False | False | False | True | False |

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

In [15]:
```
d = pd.DataFrame({
    "a": [1,2,3,4,5,6,7,8,9,10],
    "b": [6,7,8,9,0,4,5,1,2,3],
    "c": [1,4,6,8,4,2,3,5,7,2],
    "d": [6,7,8,9,0,4,5,1,2,3],
    "e": [1,2,3,4,5,6,7,8,9,10]
})
d[d["a"]>7]
```

Out[15]:

| | a | b | c | d | e |
|---|---|---|---|---|---|
| **7** | 8 | 1 | 5 | 1 | 8 |
| **8** | 9 | 2 | 7 | 2 | 9 |
| **9** | 10 | 3 | 2 | 3 | 10 |

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

In [16]:
```
d = pd.DataFrame({
    "a": [1,2,3,4,5,6,7,8,9,10],
    "b": [6,7,8,9,0,4,5,1,2,3],
    "c": [1,4,6,8,4,2,3,5,7,2],
    "d": [6,7,8,9,0,4,5,1,2,3],
    "e": [1,2,3,4,5,6,7,8,9,10]
})
```

In [17]:
```
d.loc[2:5]
```

Out[17]:

| | a | b | c | d | e |
|---|---|---|---|---|---|
| **2** | 3 | 8 | 6 | 8 | 3 |
| **3** | 4 | 9 | 8 | 9 | 4 |
| **4** | 5 | 0 | 4 | 0 | 5 |

| | a | b | c | d | e |
|---|---|---|---|---|---|
| **5** | 6 | 4 | 2 | 4 | 6 |

In [18]:
```
d.iloc[2:5]
```

Out[18]:

| | a | b | c | d | e |
|---|---|---|---|---|---|
| **2** | 3 | 8 | 6 | 8 | 3 |
| **3** | 4 | 9 | 8 | 9 | 4 |
| **4** | 5 | 0 | 4 | 0 | 5 |

In [19]:
```
d.columns
```

Out[19]:
```
Index(['a', 'b', 'c', 'd', 'e'], dtype='object')
```

In [20]:
```
d.index
```

Out[20]:
```
RangeIndex(start=0, stop=10, step=1)
```

# 10. Display the statistical summary of data

In [21]:
```
d = pd.DataFrame({
    "a": [1,2,3,4,5,6,7,8,9,10],
    "b": [6,7,8,9,0,4,5,1,2,3],
    "c": [1,4,6,8,4,2,3,5,7,2],
    "d": [6,7,8,9,0,4,5,1,2,3],
    "e": [1,2,3,4,5,6,7,8,9,10]
})
d.describe()
```

Out[21]:

| | a | b | c | d | e |
|---|---|---|---|---|---|
| **count** | 10.00000 | 10.00000 | 10.000000 | 10.00000 | 10.00000 |
| **mean** | 5.50000 | 4.50000 | 4.200000 | 4.50000 | 5.50000 |
| **std** | 3.02765 | 3.02765 | 2.299758 | 3.02765 | 3.02765 |
| **min** | 1.00000 | 0.00000 | 1.000000 | 0.00000 | 1.00000 |
| **25%** | 3.25000 | 2.25000 | 2.250000 | 2.25000 | 3.25000 |
| **50%** | 5.50000 | 4.50000 | 4.000000 | 4.50000 | 5.50000 |
| **75%** | 7.75000 | 6.75000 | 5.750000 | 6.75000 | 7.75000 |
| **max** | 10.00000 | 9.00000 | 8.000000 | 9.00000 | 10.00000 |