Participant

Title= "Matric Student"

Name= "Muhammad Bin Saqib Ali"

email = "muhammad.saqib8761@gmail.com"

whatsapp = "00923470159155"

Pandas:Tutorial (Day-11)

Install Libraries

In [ ]:
```
#pip install pandas
#pip install numpy
```

importing Libraries

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

Creating a Series of numerical values

In [ ]:
```
# Object Creation
s = pd.Series ([1,3,5,6,np.nan,7,8])
s
```

Out[ ]:
```
0    1.0
1    3.0
2    5.0
3    6.0
4    NaN
5    7.0
6    8.0
dtype: float64
```

Seting a date from specific range to specific period

In [ ]:
```
# CReating Date
date = pd.date_range("20220101", periods=40)
date
```

Out[ ]:
```
DatetimeIndex(['2022-01-01', '2022-01-02', '2022-01-03', '2022-01-04',
               '2022-01-05', '2022-01-06', '2022-01-07', '2022-01-08',
               '2022-01-09', '2022-01-10', '2022-01-11', '2022-01-12',
               '2022-01-13', '2022-01-14', '2022-01-15', '2022-01-16',
               '2022-01-17', '2022-01-18', '2022-01-19', '2022-01-20',
               '2022-01-21', '2022-01-22', '2022-01-23', '2022-01-24',
               '2022-01-25', '2022-01-26', '2022-01-27', '2022-01-28',
               '2022-01-29', '2022-01-30', '2022-01-31', '2022-02-01',
               '2022-02-02', '2022-02-03', '2022-02-04', '2022-02-05',
               '2022-02-06', '2022-02-07', '2022-02-08', '2022-02-09'],
              dtype='datetime64[ns]', freq='D')
```

Creating a **DataFrame** by **random Function**

(Index: it is a starting row of **DataFrame**)

In [ ]:
```
# DataFrame
df = pd.DataFrame (np.random.randn(40, 4), index=date, columns=list("ABCD"
df
```

Out[ ]:

	A	B	C	D
2022-01-01	0.861080	0.518295	1.594613	-0.206789
2022-01-02	0.769607	-2.067329	-1.066190	-0.842647
2022-01-03	-1.324839	-0.336920	0.039691	1.573520
2022-01-04	0.025783	0.105809	1.507726	-0.049515
2022-01-05	0.065086	1.435294	0.141095	-0.310362
2022-01-06	-0.582491	0.077244	-0.231723	-0.780222
2022-01-07	-0.359972	0.311403	0.614240	-1.282170
2022-01-08	-2.252607	-0.285497	0.239184	1.096644
2022-01-09	-0.296173	0.866654	1.751737	0.990381
2022-01-10	0.647686	-0.428401	0.163521	-0.070621
2022-01-11	-1.067293	-1.354190	-0.517117	1.038739
2022-01-12	0.096722	0.645447	0.066206	0.117101
2022-01-13	0.248857	0.027707	-1.691843	0.183612
2022-01-14	-1.043219	0.306165	2.456722	0.260523
2022-01-15	1.023181	-0.900553	1.121127	0.502756
2022-01-16	-0.081444	1.257659	-0.867107	0.141059
2022-01-17	-0.324649	1.207945	-1.136862	1.204363
2022-01-18	0.130920	1.023295	2.300363	0.205549
2022-01-19	1.883483	-0.728569	-1.602766	0.924299
2022-01-20	-1.383758	0.990634	-0.829055	0.937103

2022-01-21	-1.999956	-1.068840	-1.307210	-0.830623
2022-01-22	0.013961	-1.991207	-1.282140	0.789164
2022-01-23	0.705746	0.884127	0.360751	0.074831
2022-01-24	-0.515063	-0.736965	0.697488	1.052559
2022-01-25	-0.884928	0.909480	-0.485511	-1.193945
2022-01-26	0.372206	0.092654	0.284366	0.573087
2022-01-27	-1.674698	-2.501990	-1.232978	2.005746
2022-01-28	0.102757	0.347470	-1.253513	1.348674
2022-01-29	0.241497	-0.800766	-0.241110	-0.342790
2022-01-30	-1.112846	-0.137580	-1.251490	-0.421910
2022-01-31	0.214532	1.198623	0.694565	-0.272362
2022-02-01	0.071638	0.168366	0.693795	-0.214502
2022-02-02	-0.219538	-0.436297	0.513991	2.179210
2022-02-03	-0.048160	-0.057407	-0.133839	0.543031
2022-02-04	-0.906471	0.440047	-0.471795	-0.433594
2022-02-05	1.051856	-0.375380	0.584664	-0.859949
2022-02-06	0.009072	1.358497	-0.190492	0.962168
2022-02-07	0.660396	-0.578342	1.850189	-0.083516
2022-02-08	0.980370	-0.668554	-0.584787	-0.968210
2022-02-09	-0.152661	1.403989	-0.138938	-0.160203

Creating **DataFrame** by using **Dictionary**

In [ ]:

```
df2 = pd.DataFrame(
    {
        "A" : "1.0",
        "B" : pd.date_range ("20220113","20220116"),
        "C" : pd.Series(1, index= list(range(4)), dtype="float32"),
        "D" : np.array ([3] *4 , dtype="int32" ),
        "E" : pd.Categorical (["girl", "women", "girl", "women" ]),
        "F" : "females"

    }
)
df2
```

Out[ ]:

	A	B	C	D	E	F
0	1.0	2022-01-13	1.0	3	girl	females
1	1.0	2022-01-14	1.0	3	women	females
2	1.0	2022-01-15	1.0	3	girl	females
3	1.0	2022-01-16	1.0	3	women	females

This will show the **type** of columns in **DataFrame**

In [ ]:
```
df2.dtypes
```

Out[ ]:
```
A            object
B    datetime64[ns]
C           float32
D             int32
E          category
F            object
dtype: object
```

By this we can see **Top** Rows & Columns

In [ ]:
```
df.head(2)
```

Out[ ]:

	A	B	C	D
2022-01-01	0.861080	0.518295	1.594613	-0.206789
2022-01-02	0.769607	-2.067329	-1.066190	-0.842647

By this we can see **Bottom** Rows & Columns

In [ ]:
```
df.tail(2)
```

Out[ ]:

	A	B	C	D
2022-02-08	0.980370	-0.668554	-0.584787	-0.968210
2022-02-09	-0.152661	1.403989	-0.138938	-0.160203

By this we can see the index of our **DataFrame**

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

Out[ ]:
```
DatetimeIndex(['2022-01-01', '2022-01-02', '2022-01-03', '2022-01-04',
               '2022-01-05', '2022-01-06', '2022-01-07', '2022-01-08',
               '2022-01-09', '2022-01-10', '2022-01-11', '2022-01-12',
               '2022-01-13', '2022-01-14', '2022-01-15', '2022-01-16',
               '2022-01-17', '2022-01-18', '2022-01-19', '2022-01-20',
               '2022-01-21', '2022-01-22', '2022-01-23', '2022-01-24',
               '2022-01-25', '2022-01-26', '2022-01-27', '2022-01-28',
               '2022-01-29', '2022-01-30', '2022-01-31', '2022-02-01',
               '2022-02-02', '2022-02-03', '2022-02-04', '2022-02-05',
               '2022-02-06', '2022-02-07', '2022-02-08', '2022-02-09'],
              dtype='datetime64[ns]', freq='D')
```

In [ ]:
```
df2.index
```

Out[ ]:
```
Int64Index([0, 1, 2, 3], dtype='int64')
```

Conveting **DataFrame** into an **Array**

In [ ]:
```
df.to_numpy ()
```

Out[ ]:
```
array([[ 0.86107967,  0.51829544,  1.59461297, -0.20678922],
       [ 0.7696068 , -2.06732883, -1.06618973, -0.84264691],
       [-1.32483881, -0.33692034,  0.03969148,  1.57351975],
       [ 0.0257829 ,  0.10580867,  1.5077264 , -0.04951545],
       [ 0.06508557,  1.43529401,  0.14109457, -0.31036247],
       [-0.58249124,  0.07724422, -0.23172276, -0.78022242],
       [-0.35997238,  0.3114033 ,  0.61423966, -1.28216999],
       [-2.25260656, -0.28549688,  0.23918353,  1.09664372],
       [-0.29617314,  0.86665422,  1.75173727,  0.99038105],
       [ 0.64768584, -0.4284008 ,  0.1635206 , -0.0706211 ],
       [-1.06729259, -1.3541902 , -0.51711658,  1.03873898],
       [ 0.09672207,  0.64544726,  0.0662064 ,  0.11710055],
       [ 0.2488574 ,  0.02770687, -1.69184331,  0.18361196],
       [-1.04321853,  0.3061652 ,  2.45672212,  0.26052341],
       [ 1.02318127, -0.90055345,  1.121127  ,  0.50275647],
       [-0.08144424,  1.25765879, -0.86710714,  0.14105918],
       [-0.32464914,  1.20794525, -1.13686217,  1.20436341],
       [ 0.13091954,  1.0232954 ,  2.30036338,  0.20554856],
       [ 1.88348291, -0.72856916, -1.60276618,  0.92429942],
       [-1.38375804,  0.99063423, -0.82905529,  0.93710339],
       [-1.99995633, -1.06884008, -1.30721046, -0.83062338],
       [ 0.01396098, -1.99120749, -1.28213985,  0.78916384],
       [ 0.70574608,  0.88412723,  0.36075122,  0.07483055],
       [-0.51506288, -0.73696457,  0.69748782,  1.05255856],
       [-0.88492756,  0.90948034, -0.48551062, -1.19394548],
       [ 0.37220633,  0.09265425,  0.28436595,  0.57308742],
       [-1.67469796, -2.50199008, -1.23297814,  2.00574557],
       [ 0.1027571 ,  0.34747022, -1.25351272,  1.34867423],
       [ 0.24149742, -0.80076623, -0.24110982, -0.34279027],
       [-1.11284619, -0.13757957, -1.25148974, -0.4219101 ],
       [ 0.21453216,  1.1986233 ,  0.69456472, -0.2723619 ],
       [ 0.07163784,  0.16836615,  0.69379517, -0.21450217],
       [-0.21953797, -0.43629665,  0.51399081,  2.17920985],
       [-0.04815997, -0.05740662, -0.1338387 ,  0.54303061],
       [-0.90647085,  0.4400474 , -0.4717948 , -0.43359367],
       [ 1.05185625, -0.37537989,  0.58466438, -0.85994922],
       [ 0.00907181,  1.3584971 , -0.1904917 ,  0.96216799],
       [ 0.66039617, -0.57834217,  1.85018863, -0.08351566],
       [ 0.98036976, -0.66855394, -0.58478678, -0.96820978],
       [-0.15266065,  1.40398928, -0.13893847, -0.16020294]])
```

In [ ]:
```
df2.to_numpy
```

Out[ ]:
```
<bound method DataFrame.to_numpy of      A          B    C  D      E
F
0  1.0 2022-01-13  1.0  3   girl  females
1  1.0 2022-01-14  1.0  3  women  females
2  1.0 2022-01-15  1.0  3   girl  females
3  1.0 2022-01-16  1.0  3  women  females>
```

By **describe function** we can see our **DataFrame** Summary

In [ ]:
```
df.describe ()
```

Out[ ]:

	A	B	C	D
count	40.000000	40.000000	40.000000	40.000000
mean	-0.151358	0.003051	0.028989	0.234505
std	0.883063	0.980608	1.076268	0.855842
min	-2.252607	-2.501990	-1.691843	-1.282170
25%	-0.658100	-0.600895	-0.838568	-0.318469
50%	0.011516	0.084949	-0.047074	0.129080
75%	0.279695	0.871022	0.634129	0.943370
max	1.883483	1.435294	2.456722	2.179210

By this we can Transpose Data

In [ ]:
```
df2.T
```

Out[ ]:

	0	1	2	3
A	1.0	1.0	1.0	1.0
B	2022-01-13 00:00:00	2022-01-14 00:00:00	2022-01-15 00:00:00	2022-01-16 00:00:00
C	1.0	1.0	1.0	1.0
D	3	3	3	3
E	girl	women	girl	women
F	females	females	females	females

By this we can **Reverse** the columns

In [ ]:
```
df2.sort_index (axis=1, ascending=False) # axis 0= Vertical , axis 1= Hori
```

Out[ ]:

	F	E	D	C	B	A
0	females	girl	3	1.0	2022-01-13	1.0
1	females	women	3	1.0	2022-01-14	1.0
2	females	girl	3	1.0	2022-01-15	1.0
3	females	women	3	1.0	2022-01-16	1.0

By this we can **Reverse** the Rows

In [ ]:
```
df2.sort_index (axis=0, ascending=False) # axis 0= Vertical , axis 1= Hori
```

Out[ ]:

	A	B	C	D	E	F
3	1.0	2022-01-16	1.0	3	women	females
2	1.0	2022-01-15	1.0	3	girl	females
1	1.0	2022-01-14	1.0	3	women	females
0	1.0	2022-01-13	1.0	3	girl	females

By this we can set our Column Row in Asscending Order

In [ ]:
```
df.sort_values (by="B", ascending=False)
```

Out[ ]:

	A	B	C	D
2022-01-05	0.065086	1.435294	0.141095	-0.310362
2022-02-09	-0.152661	1.403989	-0.138938	-0.160203
2022-02-06	0.009072	1.358497	-0.190492	0.962168
2022-01-16	-0.081444	1.257659	-0.867107	0.141059
2022-01-17	-0.324649	1.207945	-1.136862	1.204363
2022-01-31	0.214532	1.198623	0.694565	-0.272362
2022-01-18	0.130920	1.023295	2.300363	0.205549
2022-01-20	-1.383758	0.990634	-0.829055	0.937103
2022-01-25	-0.884928	0.909480	-0.485511	-1.193945
2022-01-23	0.705746	0.884127	0.360751	0.074831
2022-01-09	-0.296173	0.866654	1.751737	0.990381
2022-01-12	0.096722	0.645447	0.066206	0.117101
2022-01-01	0.861080	0.518295	1.594613	-0.206789
2022-02-04	-0.906471	0.440047	-0.471795	-0.433594
2022-01-28	0.102757	0.347470	-1.253513	1.348674
2022-01-07	-0.359972	0.311403	0.614240	-1.282170
2022-01-14	-1.043219	0.306165	2.456722	0.260523
2022-02-01	0.071638	0.168366	0.693795	-0.214502
2022-01-04	0.025783	0.105809	1.507726	-0.049515

2022-01-26	0.372206	0.092654	0.284366	0.573087
2022-01-06	-0.582491	0.077244	-0.231723	-0.780222
2022-01-13	0.248857	0.027707	-1.691843	0.183612
2022-02-03	-0.048160	-0.057407	-0.133839	0.543031
2022-01-30	-1.112846	-0.137580	-1.251490	-0.421910
2022-01-08	-2.252607	-0.285497	0.239184	1.096644
2022-01-03	-1.324839	-0.336920	0.039691	1.573520
2022-02-05	1.051856	-0.375380	0.584664	-0.859949
2022-01-10	0.647686	-0.428401	0.163521	-0.070621
2022-02-02	-0.219538	-0.436297	0.513991	2.179210
2022-02-07	0.660396	-0.578342	1.850189	-0.083516
2022-02-08	0.980370	-0.668554	-0.584787	-0.968210
2022-01-19	1.883483	-0.728569	-1.602766	0.924299
2022-01-24	-0.515063	-0.736965	0.697488	1.052559
2022-01-29	0.241497	-0.800766	-0.241110	-0.342790
2022-01-15	1.023181	-0.900553	1.121127	0.502756
2022-01-21	-1.999956	-1.068840	-1.307210	-0.830623
2022-01-11	-1.067293	-1.354190	-0.517117	1.038739
2022-01-22	0.013961	-1.991207	-1.282140	0.789164
2022-01-02	0.769607	-2.067329	-1.066190	-0.842647
2022-01-27	-1.674698	-2.501990	-1.232978	2.005746

By this we can see a Data of specific Column

In [ ]:
```
df["A"]
```

Out[ ]:
```
2022-01-01    0.861080
2022-01-02    0.769607
2022-01-03   -1.324839
2022-01-04    0.025783
2022-01-05    0.065086
2022-01-06   -0.582491
2022-01-07   -0.359972
2022-01-08   -2.252607
2022-01-09   -0.296173
2022-01-10    0.647686
2022-01-11   -1.067293
2022-01-12    0.096722
2022-01-13    0.248857
2022-01-14   -1.043219
2022-01-15    1.023181
2022-01-16   -0.081444
2022-01-17   -0.324649
2022-01-18    0.130920
2022-01-19    1.883483
2022-01-20   -1.383758
2022-01-21   -1.999956
2022-01-22    0.013961
2022-01-23    0.705746
2022-01-24   -0.515063
2022-01-25   -0.884928
2022-01-26    0.372206
2022-01-27   -1.674698
2022-01-28    0.102757
2022-01-29    0.241497
2022-01-30   -1.112846
2022-01-31    0.214532
2022-02-01    0.071638
2022-02-02   -0.219538
2022-02-03   -0.048160
2022-02-04   -0.906471
2022-02-05    1.051856
2022-02-06    0.009072
2022-02-07    0.660396
2022-02-08    0.980370
2022-02-09   -0.152661
Freq: D, Name: A, dtype: float64
```

By this we can see a specific range of Rows

In [ ]:
```
df [0:2]
```

Out[ ]:

	A	B	C	D
2022-01-01	0.861080	0.518295	1.594613	-0.206789
2022-01-02	0.769607	-2.067329	-1.066190	-0.842647

By this we can see a Data of **specific** date

In [ ]:
```
df.loc [date[0]]
```

Out[ ]:
```
A    0.861080
B    0.518295
C    1.594613
D   -0.206789
Name: 2022-01-01 00:00:00, dtype: float64
```

By this we can see a Specific Data of particular Columns with **All Rows**

In [ ]:
```
df.loc [:,["A","B"]]
```

Out[ ]:

	A	B
2022-01-01	0.861080	0.518295
2022-01-02	0.769607	-2.067329
2022-01-03	-1.324839	-0.336920
2022-01-04	0.025783	0.105809
2022-01-05	0.065086	1.435294
2022-01-06	-0.582491	0.077244
2022-01-07	-0.359972	0.311403
2022-01-08	-2.252607	-0.285497
2022-01-09	-0.296173	0.866654
2022-01-10	0.647686	-0.428401
2022-01-11	-1.067293	-1.354190
2022-01-12	0.096722	0.645447
2022-01-13	0.248857	0.027707
2022-01-14	-1.043219	0.306165
2022-01-15	1.023181	-0.900553
2022-01-16	-0.081444	1.257659
2022-01-17	-0.324649	1.207945
2022-01-18	0.130920	1.023295
2022-01-19	1.883483	-0.728569
2022-01-20	-1.383758	0.990634
2022-01-21	-1.999956	-1.068840

2022-01-22	0.013961	-1.991207
2022-01-23	0.705746	0.884127
2022-01-24	-0.515063	-0.736965
2022-01-25	-0.884928	0.909480
2022-01-26	0.372206	0.092654
2022-01-27	-1.674698	-2.501990
2022-01-28	0.102757	0.347470
2022-01-29	0.241497	-0.800766
2022-01-30	-1.112846	-0.137580
2022-01-31	0.214532	1.198623
2022-02-01	0.071638	0.168366
2022-02-02	-0.219538	-0.436297
2022-02-03	-0.048160	-0.057407
2022-02-04	-0.906471	0.440047
2022-02-05	1.051856	-0.375380
2022-02-06	0.009072	1.358497
2022-02-07	0.660396	-0.578342
2022-02-08	0.980370	-0.668554
2022-02-09	-0.152661	1.403989

By this we can see a Specific Data of particular Columns with **specific** date

In [ ]:
```
df.loc ["20220105", ["A","B"]]
```

Out[ ]:
```
A    0.065086
B    1.435294
Name: 2022-01-05 00:00:00, dtype: float64
```

By this we can see a Specific Data of particular Columns with **2** specific date

In [ ]:
```
df.loc [["20220105","20220110"], ["A","B"]]
```

Out[ ]:

	A	B
2022-01-05	0.065086	1.435294
2022-01-10	0.647686	-0.428401

By this we can see a Specific Data of particular Columns with specific **Range** date

In [ ]:
```
df.loc ["20220105":"20220110", ["A","B"]]
```

Out[ ]:

	A	B
2022-01-05	0.065086	1.435294
2022-01-06	-0.582491	0.077244
2022-01-07	-0.359972	0.311403
2022-01-08	-2.252607	-0.285497
2022-01-09	-0.296173	0.866654
2022-01-10	0.647686	-0.428401

By this we can see a Specific Data of particular Columns with specific date

In [ ]:
```
df.at [date[0], "A"]
```

Out[ ]:
```
0.8610796656587548
```

By this we can set a range of **Rows** we have to see with **All** Columns by **Index** number

In [ ]:
```
df.iloc [4:10]
```

Out[ ]:

	A	B	C	D
2022-01-05	0.065086	1.435294	0.141095	-0.310362
2022-01-06	-0.582491	0.077244	-0.231723	-0.780222
2022-01-07	-0.359972	0.311403	0.614240	-1.282170
2022-01-08	-2.252607	-0.285497	0.239184	1.096644
2022-01-09	-0.296173	0.866654	1.751737	0.990381
2022-01-10	0.647686	-0.428401	0.163521	-0.070621

By this we can set a range of **Rows** and **Columns** by **Index** number

In [ ]:
```
df.iloc [3:10, 0:3]
```

Out[ ]:

	A	B	C
2022-01-04	0.025783	0.105809	1.507726
2022-01-05	0.065086	1.435294	0.141095
2022-01-06	-0.582491	0.077244	-0.231723
2022-01-07	-0.359972	0.311403	0.614240
2022-01-08	-2.252607	-0.285497	0.239184
2022-01-09	-0.296173	0.866654	1.751737
2022-01-10	0.647686	-0.428401	0.163521

By this we can set a range of **Rows** we have to see with **All** Columns

In [ ]:
```
df.iloc [3:10, :]
```

Out[ ]:

	A	B	C	D
2022-01-04	0.025783	0.105809	1.507726	-0.049515
2022-01-05	0.065086	1.435294	0.141095	-0.310362
2022-01-06	-0.582491	0.077244	-0.231723	-0.780222
2022-01-07	-0.359972	0.311403	0.614240	-1.282170
2022-01-08	-2.252607	-0.285497	0.239184	1.096644
2022-01-09	-0.296173	0.866654	1.751737	0.990381
2022-01-10	0.647686	-0.428401	0.163521	-0.070621

By this we can set a range of **Columns** we have to see with **All** Rows

In [ ]:
```
df.iloc [:, 0:2]
```

Out[ ]:

	A	B
2022-01-01	0.861080	0.518295
2022-01-02	0.769607	-2.067329
2022-01-03	-1.324839	-0.336920
2022-01-04	0.025783	0.105809
2022-01-05	0.065086	1.435294
2022-01-06	-0.582491	0.077244
2022-01-07	-0.359972	0.311403

2022-01-08	-2.252607	-0.285497
2022-01-09	-0.296173	0.866654
2022-01-10	0.647686	-0.428401
2022-01-11	-1.067293	-1.354190
2022-01-12	0.096722	0.645447
2022-01-13	0.248857	0.027707
2022-01-14	-1.043219	0.306165
2022-01-15	1.023181	-0.900553
2022-01-16	-0.081444	1.257659
2022-01-17	-0.324649	1.207945
2022-01-18	0.130920	1.023295
2022-01-19	1.883483	-0.728569
2022-01-20	-1.383758	0.990634
2022-01-21	-1.999956	-1.068840
2022-01-22	0.013961	-1.991207
2022-01-23	0.705746	0.884127
2022-01-24	-0.515063	-0.736965
2022-01-25	-0.884928	0.909480
2022-01-26	0.372206	0.092654
2022-01-27	-1.674698	-2.501990
2022-01-28	0.102757	0.347470
2022-01-29	0.241497	-0.800766
2022-01-30	-1.112846	-0.137580
2022-01-31	0.214532	1.198623
2022-02-01	0.071638	0.168366
2022-02-02	-0.219538	-0.436297
2022-02-03	-0.048160	-0.057407
2022-02-04	-0.906471	0.440047
2022-02-05	1.051856	-0.375380
2022-02-06	0.009072	1.358497
2022-02-07	0.660396	-0.578342
2022-02-08	0.980370	-0.668554
2022-02-09	-0.152661	1.403989

By this we can see a greater than value of our Rows Elements value With Specific Column

In [ ]:
```
df [df["A"]>1]
```

Out[ ]:

	A	B	C	D
2022-01-15	1.023181	-0.900553	1.121127	0.502756
2022-01-19	1.883483	-0.728569	-1.602766	0.924299
2022-02-05	1.051856	-0.375380	0.584664	-0.859949

By this we can see a greater than value of our Rows Elements value of whole Data

In [ ]:
```
df [df>0]
```

Out[ ]:

	A	B	C	D
2022-01-01	0.861080	0.518295	1.594613	NaN
2022-01-02	0.769607	NaN	NaN	NaN
2022-01-03	NaN	NaN	0.039691	1.573520
2022-01-04	0.025783	0.105809	1.507726	NaN
2022-01-05	0.065086	1.435294	0.141095	NaN
2022-01-06	NaN	0.077244	NaN	NaN
2022-01-07	NaN	0.311403	0.614240	NaN
2022-01-08	NaN	NaN	0.239184	1.096644
2022-01-09	NaN	0.866654	1.751737	0.990381
2022-01-10	0.647686	NaN	0.163521	NaN
2022-01-11	NaN	NaN	NaN	1.038739
2022-01-12	0.096722	0.645447	0.066206	0.117101
2022-01-13	0.248857	0.027707	NaN	0.183612
2022-01-14	NaN	0.306165	2.456722	0.260523
2022-01-15	1.023181	NaN	1.121127	0.502756
2022-01-16	NaN	1.257659	NaN	0.141059
2022-01-17	NaN	1.207945	NaN	1.204363
2022-01-18	0.130920	1.023295	2.300363	0.205549
2022-01-19	1.883483	NaN	NaN	0.924299
2022-01-20	NaN	0.990634	NaN	0.937103

2022-01-21	NaN	NaN	NaN	NaN
2022-01-22	0.013961	NaN	NaN	0.789164
2022-01-23	0.705746	0.884127	0.360751	0.074831
2022-01-24	NaN	NaN	0.697488	1.052559
2022-01-25	NaN	0.909480	NaN	NaN
2022-01-26	0.372206	0.092654	0.284366	0.573087
2022-01-27	NaN	NaN	NaN	2.005746
2022-01-28	0.102757	0.347470	NaN	1.348674
2022-01-29	0.241497	NaN	NaN	NaN
2022-01-30	NaN	NaN	NaN	NaN
2022-01-31	0.214532	1.198623	0.694565	NaN
2022-02-01	0.071638	0.168366	0.693795	NaN
2022-02-02	NaN	NaN	0.513991	2.179210
2022-02-03	NaN	NaN	NaN	0.543031
2022-02-04	NaN	0.440047	NaN	NaN
2022-02-05	1.051856	NaN	0.584664	NaN
2022-02-06	0.009072	1.358497	NaN	0.962168
2022-02-07	0.660396	NaN	1.850189	NaN
2022-02-08	0.980370	NaN	NaN	NaN
2022-02-09	NaN	1.403989	NaN	NaN

By this we can copy a **DataFrame**

In [ ]:
```
new = df.copy ()
```

By this we can add a column on a DataFrame

note: same number of rows in DataFrame required to create a new Column

In [ ]:
```
new ["E"]= ["one", "two", "three", "four", "five",
"one", "two", "three", "four", "five",
"one", "two", "three", "four", "five",
"one", "two", "three", "four", "five",
"one", "two", "three", "four", "five",
"one", "two", "three", "four", "five",
"one", "two", "three", "four", "five",]
```

In [ ]:
```
new
```

Out[ ]:

	A	B	C	D	E
2022-01-01	0.861080	0.518295	1.594613	-0.206789	one
2022-01-02	0.769607	-2.067329	-1.066190	-0.842647	two
2022-01-03	-1.324839	-0.336920	0.039691	1.573520	three
2022-01-04	0.025783	0.105809	1.507726	-0.049515	four
2022-01-05	0.065086	1.435294	0.141095	-0.310362	five
2022-01-06	-0.582491	0.077244	-0.231723	-0.780222	one
2022-01-07	-0.359972	0.311403	0.614240	-1.282170	two
2022-01-08	-2.252607	-0.285497	0.239184	1.096644	three
2022-01-09	-0.296173	0.866654	1.751737	0.990381	four
2022-01-10	0.647686	-0.428401	0.163521	-0.070621	five
2022-01-11	-1.067293	-1.354190	-0.517117	1.038739	one
2022-01-12	0.096722	0.645447	0.066206	0.117101	two
2022-01-13	0.248857	0.027707	-1.691843	0.183612	three
2022-01-14	-1.043219	0.306165	2.456722	0.260523	four
2022-01-15	1.023181	-0.900553	1.121127	0.502756	five
2022-01-16	-0.081444	1.257659	-0.867107	0.141059	one
2022-01-17	-0.324649	1.207945	-1.136862	1.204363	two
2022-01-18	0.130920	1.023295	2.300363	0.205549	three
2022-01-19	1.883483	-0.728569	-1.602766	0.924299	four
2022-01-20	-1.383758	0.990634	-0.829055	0.937103	five
2022-01-21	-1.999956	-1.068840	-1.307210	-0.830623	one
2022-01-22	0.013961	-1.991207	-1.282140	0.789164	two
2022-01-23	0.705746	0.884127	0.360751	0.074831	three
2022-01-24	-0.515063	-0.736965	0.697488	1.052559	four
2022-01-25	-0.884928	0.909480	-0.485511	-1.193945	five
2022-01-26	0.372206	0.092654	0.284366	0.573087	one
2022-01-27	-1.674698	-2.501990	-1.232978	2.005746	two
2022-01-28	0.102757	0.347470	-1.253513	1.348674	three
2022-01-29	0.241497	-0.800766	-0.241110	-0.342790	four
2022-01-30	-1.112846	-0.137580	-1.251490	-0.421910	five
2022-01-31	0.214532	1.198623	0.694565	-0.272362	one

2022-02-01	0.071638	0.168366	0.693795	-0.214502	two
2022-02-02	-0.219538	-0.436297	0.513991	2.179210	three
2022-02-03	-0.048160	-0.057407	-0.133839	0.543031	four
2022-02-04	-0.906471	0.440047	-0.471795	-0.433594	five
2022-02-05	1.051856	-0.375380	0.584664	-0.859949	one
2022-02-06	0.009072	1.358497	-0.190492	0.962168	two
2022-02-07	0.660396	-0.578342	1.850189	-0.083516	three
2022-02-08	0.980370	-0.668554	-0.584787	-0.968210	four
2022-02-09	-0.152661	1.403989	-0.138938	-0.160203	five

In [ ]: