Day 3 Task

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

1. Create any Series and print the output

In [2]:
```
a=pd.Series([20,40,60,80,100])
print(a)
```

```
0     20
1     40
2     60
3     80
4    100
dtype: int64
```

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

In [14]:
```
df=pd.DataFrame(np.random.rand(10,5))
df[3][2]=np.nan
df[2][4]=np.nan
df[0][8]=np.nan
df
```

Out[14]:

	0	1	2	3	4
0	0.420702	0.745455	0.705493	0.245064	0.657963
1	0.227202	0.588925	0.021333	0.576791	0.551916
2	0.996535	0.663107	0.243137	NaN	0.596514
3	0.530324	0.996106	0.735149	0.235099	0.999253
4	0.018840	0.703509	NaN	0.653305	0.809451
5	0.434705	0.640684	0.051865	0.220355	0.760144
6	0.524341	0.387746	0.377957	0.370733	0.871537
7	0.913333	0.240910	0.229992	0.546014	0.125003
8	NaN	0.325384	0.105023	0.319750	0.592201
9	0.249601	0.110806	0.444489	0.661664	0.925504

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

In [15]:
```
df.head(7)
```

Out[15]:

	0	1	2	3	4
0	0.420702	0.745455	0.705493	0.245064	0.657963
1	0.227202	0.588925	0.021333	0.576791	0.551916
2	0.996535	0.663107	0.243137	NaN	0.596514
3	0.530324	0.996106	0.735149	0.235099	0.999253
4	0.018840	0.703509	NaN	0.653305	0.809451
5	0.434705	0.640684	0.051865	0.220355	0.760144
6	0.524341	0.387746	0.377957	0.370733	0.871537

In [16]:
```
df.tail(6)
```

Out[16]:

	0	1	2	3	4
4	0.018840	0.703509	NaN	0.653305	0.809451
5	0.434705	0.640684	0.051865	0.220355	0.760144
6	0.524341	0.387746	0.377957	0.370733	0.871537
7	0.913333	0.240910	0.229992	0.546014	0.125003
8	NaN	0.325384	0.105023	0.319750	0.592201
9	0.249601	0.110806	0.444489	0.661664	0.925504

4. Fill with a constant value and print the output

In [17]:
```
df.fillna(value="10")
```

Out[17]:

	0	1	2	3	4
0	0.420702	0.745455	0.705493	0.245064	0.657963
1	0.227202	0.588925	0.021333	0.576791	0.551916
2	0.996535	0.663107	0.243137	10	0.596514
3	0.530324	0.996106	0.735149	0.235099	0.999253
4	0.01884	0.703509	10	0.653305	0.809451
5	0.434705	0.640684	0.051865	0.220355	0.760144
6	0.524341	0.387746	0.377957	0.370733	0.871537
7	0.913333	0.240910	0.229992	0.546014	0.125003
8	10	0.325384	0.105023	0.31975	0.592201
9	0.249601	0.110806	0.444489	0.661664	0.925504

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

In [18]:
```
df.dropna(axis=1)
```

Out[18]:

	1	4
0	0.745455	0.657963
1	0.588925	0.551916
2	0.663107	0.596514
3	0.996106	0.999253
4	0.703509	0.809451
5	0.640684	0.760144
6	0.387746	0.871537
7	0.240910	0.125003
8	0.325384	0.592201
9	0.110806	0.925504

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

In [19]:
```
df.dropna()
```

Out[19]:

	0	1	2	3	4
0	0.420702	0.745455	0.705493	0.245064	0.657963
1	0.227202	0.588925	0.021333	0.576791	0.551916
3	0.530324	0.996106	0.735149	0.235099	0.999253
5	0.434705	0.640684	0.051865	0.220355	0.760144
6	0.524341	0.387746	0.377957	0.370733	0.871537
7	0.913333	0.240910	0.229992	0.546014	0.125003
9	0.249601	0.110806	0.444489	0.661664	0.925504

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

In [20]:
```
df.isna()
```

Out[20]:

	0	1	2	3	4
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	True	False
3	False	False	False	False	False
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	True	False	False	False	False
9	False	False	False	False	False

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

In [26]:
```
df=df[df>0.5]
df
```

Out[26]:

	0	1	2	3	4
0	NaN	0.745455	0.705493	NaN	0.657963
1	NaN	0.588925	NaN	0.576791	0.551916
2	0.996535	0.663107	NaN	NaN	0.596514
3	0.530324	0.996106	0.735149	NaN	0.999253
4	NaN	0.703509	NaN	0.653305	0.809451
5	NaN	0.640684	NaN	NaN	0.760144
6	0.524341	NaN	NaN	NaN	0.871537
7	0.913333	NaN	NaN	0.546014	NaN
8	NaN	NaN	NaN	NaN	0.592201
9	NaN	NaN	NaN	0.661664	0.925504

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

In [28]:
```
df.loc[1:3]
```

Out[28]:

	0	1	2	3	4
1	NaN	0.588925	NaN	0.576791	0.551916
2	0.996535	0.663107	NaN	NaN	0.596514
3	0.530324	0.996106	0.735149	NaN	0.999253

In [29]:
```
df.iloc[2:6]
```

Out[29]:

	0	1	2	3	4
2	0.996535	0.663107	NaN	NaN	0.596514
3	0.530324	0.996106	0.735149	NaN	0.999253
4	NaN	0.703509	NaN	0.653305	0.809451
5	NaN	0.640684	NaN	NaN	0.760144

10. Display the statistical summary of data

In [30]:
```
df.describe()
```

Out[30]:

	0	1	2	3	4
count	4.000000	6.000000	2.000000	4.000000	9.000000
mean	0.741133	0.722964	0.720321	0.609443	0.751609
std	0.249213	0.144090	0.020970	0.056980	0.161044
min	0.524341	0.588925	0.705493	0.546014	0.551916
25%	0.528828	0.646290	0.712907	0.569097	0.596514
50%	0.721828	0.683308	0.720321	0.615048	0.760144
75%	0.934133	0.734969	0.727735	0.655395	0.871537
max	0.996535	0.996106	0.735149	0.661664	0.999253