21-07-2023

Day 3 Assignment

In [2]:

```
import pandas as pd
import numpy as np
```

1. Create any Series and print the output

In [4]:

```
a=pd.Series([10,20,30,40,50])
a
```

Out[4]:

```
0    10
1    20
2    30
3    40
4    50
dtype: int64
```

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

In [44]:

```
df=pd.DataFrame(np.random.rand(10,5))
df[4][1]="NaN"
df[0][3]="NaN"
df[2][5]="NaN"
df[2][7]="NaN"
df[4][9]="NaN"
df[0][0]="A"
df
```

Out[44]:

	0	1	2	3	4
0	A	0.186408	0.935429	0.060907	0.277167
1	0.917805	0.674018	0.974844	0.676015	NaN
2	0.873594	0.001803	0.608771	0.802191	0.054605
3	NaN	0.164431	0.348860	0.194949	0.565721
4	0.648503	0.560420	0.438621	0.857265	0.351626
5	0.692569	0.465682	NaN	0.617035	0.262419
6	0.194507	0.334514	0.297704	0.646909	0.162016
7	0.210085	0.154150	NaN	0.994842	0.799139
8	0.38131	0.796511	0.458933	0.262119	0.950761
9	0.066521	0.638104	0.875809	0.495601	NaN

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

In [45]:

```
df.head(7)#First 7 rows
```

Out[45]:

	0	1	2	3	4
0	A	0.186408	0.935429	0.060907	0.277167
1	0.917805	0.674018	0.974844	0.676015	NaN
2	0.873594	0.001803	0.608771	0.802191	0.054605
3	NaN	0.164431	0.348860	0.194949	0.565721
4	0.648503	0.560420	0.438621	0.857265	0.351626
5	0.692569	0.465682	NaN	0.617035	0.262419
6	0.194507	0.334514	0.297704	0.646909	0.162016

In [46]:

```
df.tail(6)#Last 6 rows
```

Out[46]:

	0	1	2	3	4
4	0.648503	0.560420	0.438621	0.857265	0.351626
5	0.692569	0.465682	NaN	0.617035	0.262419
6	0.194507	0.334514	0.297704	0.646909	0.162016
7	0.210085	0.154150	NaN	0.994842	0.799139
8	0.38131	0.796511	0.458933	0.262119	0.950761
9	0.066521	0.638104	0.875809	0.495601	NaN

4. Fill with a constant value and print the output

In [47]:

```
df.fillna(value="7")
```

Out[47]:

	0	1	2	3	4
0	A	0.186408	0.935429	0.060907	0.277167
1	0.917805	0.674018	0.974844	0.676015	7
2	0.873594	0.001803	0.608771	0.802191	0.054605
3	7	0.164431	0.34886	0.194949	0.565721
4	0.648503	0.560420	0.438621	0.857265	0.351626
5	0.692569	0.465682	7	0.617035	0.262419
6	0.194507	0.334514	0.297704	0.646909	0.162016
7	0.210085	0.154150	7	0.994842	0.799139
8	0.38131	0.796511	0.458933	0.262119	0.950761
9	0.066521	0.638104	0.875809	0.495601	7

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

In [48]:

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

Out[48]:

	1	3
0	0.186408	0.060907
1	0.674018	0.676015
2	0.001803	0.802191
3	0.164431	0.194949
4	0.560420	0.857265
5	0.465682	0.617035
6	0.334514	0.646909
7	0.154150	0.994842
8	0.796511	0.262119
9	0.638104	0.495601

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

In [49]:

```
df.dropna()
```

Out[49]:

	0	1	2	3	4
0	A	0.186408	0.935429	0.060907	0.277167
2	0.873594	0.001803	0.608771	0.802191	0.054605
4	0.648503	0.560420	0.438621	0.857265	0.351626
6	0.194507	0.334514	0.297704	0.646909	0.162016
8	0.38131	0.796511	0.458933	0.262119	0.950761

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

In [50]:

```
df.isna()
```

Out[50]:

	0	1	2	3	4
0	False	False	False	False	False
1	False	False	False	False	True
2	False	False	False	False	False
3	True	False	False	False	False
4	False	False	False	False	False
5	False	False	True	False	False
6	False	False	False	False	False
7	False	False	True	False	False
8	False	False	False	False	False
9	False	False	False	False	True

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

In [51]:

```
df[df[2]<0.5]
```

Out[51]:

	0	1	2	3	4
3	NaN	0.164431	0.348860	0.194949	0.565721
4	0.648503	0.560420	0.438621	0.857265	0.351626
6	0.194507	0.334514	0.297704	0.646909	0.162016
8	0.38131	0.796511	0.458933	0.262119	0.950761

In [52]:

```
df[df[3]<0.5]
```

Out[52]:

	0	1	2	3	4
0	A	0.186408	0.935429	0.060907	0.277167
3	NaN	0.164431	0.348860	0.194949	0.565721
8	0.38131	0.796511	0.458933	0.262119	0.950761
9	0.066521	0.638104	0.875809	0.495601	NaN

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

In [53]:

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

Out[53]:

	0	1	2	3	4
1	0.917805	0.674018	0.974844	0.676015	NaN
2	0.873594	0.001803	0.608771	0.802191	0.054605
3	NaN	0.164431	0.348860	0.194949	0.565721

In [56]:

```
df.iloc[0:3]
```

Out[56]:

	0	1	2	3	4
0	A	0.186408	0.935429	0.060907	0.277167
1	0.917805	0.674018	0.974844	0.676015	NaN
2	0.873594	0.001803	0.608771	0.802191	0.054605

10. Display the statistical summary of data

In [31]:

```
df.describe()
```

Out[31]:

	0	1	2	3	4
count	9.000000	10.000000	8.000000	10.000000	8.000000
mean	0.654959	0.549762	0.467264	0.403009	0.455408
std	0.284247	0.261409	0.293252	0.285864	0.251728
min	0.184197	0.174682	0.053532	0.054238	0.141331
25%	0.445140	0.335574	0.266651	0.229928	0.260953
50%	0.702687	0.589471	0.468421	0.335048	0.403969
75%	0.925751	0.755991	0.632449	0.519013	0.685223
max	0.991522	0.938201	0.980470	0.928611	0.779407