Day-3

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

1. Create any Series and print the output

In [2]:
```
data=[1,2,3,4,5,6,7,8,9,]
s=pd.DataFrame(data)
s
```

Out[2]:
```
   0
0  1
1  2
2  3
3  4
4  5
5  6
6  7
7  8
8  9
```

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

In [3]:

```
data1={'A': [1,2,3,4,5,6,7,8,9,10],
       'B': [11,12,13,14,15,16,17,18,19,20],
       'C': [21,22,23,np.nan,25,26,27,28,29,30],
       'D': [31,32,33,34,35,np.nan,37,38,39,40],
       'E': [41,42,np.nan,44,45,46,47,48,49,50]
}
df=pd.DataFrame(data1)
df
```

Out[3]:

	A	B	C	D	E
0	1	11	21.0	31.0	41.0
1	2	12	22.0	32.0	42.0
2	3	13	23.0	33.0	NaN
3	4	14	NaN	34.0	44.0
4	5	15	25.0	35.0	45.0
5	6	16	26.0	NaN	46.0
6	7	17	27.0	37.0	47.0
7	8	18	28.0	38.0	48.0
8	9	19	29.0	39.0	49.0
9	10	20	30.0	40.0	50.0

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

In [4]:

```
df.head(7)
```

Out[4]:

	A	B	C	D	E
0	1	11	21.0	31.0	41.0
1	2	12	22.0	32.0	42.0
2	3	13	23.0	33.0	NaN
3	4	14	NaN	34.0	44.0
4	5	15	25.0	35.0	45.0
5	6	16	26.0	NaN	46.0
6	7	17	27.0	37.0	47.0

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

Out[5]:

	A	B	C	D	E
4	5	15	25.0	35.0	45.0
5	6	16	26.0	NaN	46.0
6	7	17	27.0	37.0	47.0
7	8	18	28.0	38.0	48.0
8	9	19	29.0	39.0	49.0
9	10	20	30.0	40.0	50.0

4. Fill with a constant value and print the output

In [6]:
```
df1=df
df1.fillna(value=25)
```

Out[6]:

	A	B	C	D	E
0	1	11	21.0	31.0	41.0
1	2	12	22.0	32.0	42.0
2	3	13	23.0	33.0	25.0
3	4	14	25.0	34.0	44.0
4	5	15	25.0	35.0	45.0
5	6	16	26.0	25.0	46.0
6	7	17	27.0	37.0	47.0
7	8	18	28.0	38.0	48.0
8	9	19	29.0	39.0	49.0
9	10	20	30.0	40.0	50.0

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

In [7]:
```
df2=df
df2.dropna(axis=1)
```

Out[7]:
```
    A   B
0   1  11
1   2  12
2   3  13
3   4  14
4   5  15
5   6  16
6   7  17
7   8  18
8   9  19
9  10  20
```

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

In [8]:
```
df2=df
df2.dropna()
```

Out[8]:
```
    A   B     C     D     E
0   1  11  21.0  31.0  41.0
1   2  12  22.0  32.0  42.0
4   5  15  25.0  35.0  45.0
6   7  17  27.0  37.0  47.0
7   8  18  28.0  38.0  48.0
8   9  19  29.0  39.0  49.0
9  10  20  30.0  40.0  50.0
```

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

In [9]:

```
df.isnull()
```

Out[9]:

	A	B	C	D	E
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	True
3	False	False	True	False	False
4	False	False	False	False	False
5	False	False	False	True	False
6	False	False	False	False	False
7	False	False	False	False	False
8	False	False	False	False	False
9	False	False	False	False	False

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

In [10]:

```
df3=df[df>20]
df3
```

Out[10]:

	A	B	C	D	E
0	NaN	NaN	21.0	31.0	41.0
1	NaN	NaN	22.0	32.0	42.0
2	NaN	NaN	23.0	33.0	NaN
3	NaN	NaN	NaN	34.0	44.0
4	NaN	NaN	25.0	35.0	45.0
5	NaN	NaN	26.0	NaN	46.0
6	NaN	NaN	27.0	37.0	47.0
7	NaN	NaN	28.0	38.0	48.0
8	NaN	NaN	29.0	39.0	49.0
9	NaN	NaN	30.0	40.0	50.0

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

In [11]:
```
df.loc[0:5]
```

Out[11]:
```
   A   B     C     D     E
0  1  11  21.0  31.0  41.0
1  2  12  22.0  32.0  42.0
2  3  13  23.0  33.0   NaN
3  4  14   NaN  34.0  44.0
4  5  15  25.0  35.0  45.0
5  6  16  26.0   NaN  46.0
```

In [12]:
```
df.iloc[3:7]
```

Out[12]:
```
   A   B     C     D     E
3  4  14   NaN  34.0  44.0
4  5  15  25.0  35.0  45.0
5  6  16  26.0   NaN  46.0
6  7  17  27.0  37.0  47.0
```

10. Display the statistical summary of data

In [13]:
```
df.describe()
```

Out[13]:
```
              A         B          C          D          E
count  10.00000  10.00000   9.000000   9.000000   9.000000
 mean   5.50000  15.50000  25.666667  35.444444  45.777778
  std   3.02765   3.02765   3.162278   3.205897   3.073181
  min   1.00000  11.00000  21.000000  31.000000  41.000000
  25%   3.25000  13.25000  23.000000  33.000000  44.000000
  50%   5.50000  15.50000  26.000000  35.000000  46.000000
  75%   7.75000  17.75000  28.000000  38.000000  48.000000
  max  10.00000  20.00000  30.000000  40.000000  50.000000
```

MINI-PROJECT

Dataset-1

a)Import library

In [14]:
```python
import pandas as pd
```

b)Import dataset

In [15]:
```python
df4=pd.read_csv(r"C:\Users\user\Downloads\dataset.csv")
df4
```

Out[15]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	l
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.6115598
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.241889
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.417
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.634609
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.495650
...	...	...	...	...	...	...	...	
1544	NaN	NaN	NaN	NaN	NaN	NaN	NaN	leng
1545	NaN	NaN	NaN	NaN	NaN	NaN	NaN	con
1546	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Null valu
1547	NaN	NaN	NaN	NaN	NaN	NaN	NaN	fi
1548	NaN	NaN	NaN	NaN	NaN	NaN	NaN	sear

1549 rows × 11 columns

c)head

In [16]:
```python
df4.head(10)
```

Out[16]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.611559868
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.24188995
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.41784
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.63460922
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.49565029
5	6.0	pop	74.0	3623.0	70225.0	1.0	45.000702	7.68227005
6	7.0	lounge	51.0	731.0	11600.0	1.0	44.907242	8.611559868
7	8.0	lounge	51.0	1521.0	49076.0	1.0	41.903221	12.49565029
8	9.0	sport	73.0	4049.0	76000.0	1.0	45.548000	11.54946995
9	10.0	sport	51.0	3653.0	89000.0	1.0	45.438301	10.99170017

d)tail

In [17]:
```
df4.tail(10)
```

Out[17]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
1539	NaN	NaN	NaN	NaN	NaN	NaN	NaN	avg	8576.003901
1540	NaN	NaN	NaN	NaN	NaN	NaN	NaN	count	1538
1541	NaN	NaN	NaN	NaN	NaN	NaN	NaN	countif	377
1542	NaN	NaN	NaN	NaN	NaN	NaN	NaN	sumif	4017825
1543	NaN	NaN	NaN	NaN	NaN	NaN	NaN	counta (not empty)	1538
1544	NaN	NaN	NaN	NaN	NaN	NaN	NaN	length	5
1545	NaN	NaN	NaN	NaN	NaN	NaN	NaN	concat	lonprice
1546	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Null values	NO
1547	NaN	NaN	NaN	NaN	NaN	NaN	NaN	find	1
1548	NaN	NaN	NaN	NaN	NaN	NaN	NaN	search	1

e)describe

In [18]:
```
df4.describe()
```

Out[18]:

	ID	engine_power	age_in_days	km	previous_owners	lat
count	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000
mean	769.500000	51.904421	1650.980494	53396.011704	1.123537	43.541361
std	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.133518
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.855839
25%	385.250000	51.000000	670.000000	20006.250000	1.000000	41.802990
50%	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.394096
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.467960
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612

f)shape

In [19]:
```
df4.shape
```

Out[19]:
```
(1549, 11)
```

g)size

```
In [20]: df4.size
```

```
Out[20]: 17039
```

h)find missing values

```
In [21]: df4.isnull()
```

```
Out[21]:
```

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price	Un
0	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	
1544	True	True	True	True	True	True	True	False	False	
1545	True	True	True	True	True	True	True	False	False	
1546	True	True	True	True	True	True	True	False	False	
1547	True	True	True	True	True	True	True	False	False	
1548	True	True	True	True	True	True	True	False	False	

1549 rows × 11 columns

i)fill/drop

In [22]:
```
df5=df4
df5.fillna(value=0)
```

Out[22]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lo
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.61155986
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.2418899
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.4178
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.6346092
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.4956502
...	...	...	...	...	...	...	...	.
1544	0.0	0	0.0	0.0	0.0	0.0	0.000000	lengt
1545	0.0	0	0.0	0.0	0.0	0.0	0.000000	conca
1546	0.0	0	0.0	0.0	0.0	0.0	0.000000	Null value
1547	0.0	0	0.0	0.0	0.0	0.0	0.000000	fin
1548	0.0	0	0.0	0.0	0.0	0.0	0.000000	searc

1549 rows × 11 columns

In [23]:
```
df6=df4
df6.dropna(axis=1)
```

Out[23]:

	lon	price
0	8.611559868	8900
1	12.24188995	8800
2	11.41784	4200
3	17.63460922	6000
4	12.49565029	5700
...	...	...
1544	length	5
1545	concat	lonprice
1546	Null values	NO
1547	find	1
1548	search	1

1549 rows × 2 columns

Dataset-2

a)Import library

In [24]:
```
import pandas as pd
```

b)Import dataset

In [25]:
```
dff=pd.read_csv(r"C:\Users\user\Downloads\dataset2.csv")
dff
```

Out[25]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Fre
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.
...	...	...	...	...	...	...	...	...	
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.42864	0.
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910	0.
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193	0.
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.22396	0.
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.28443	0.

158 rows × 12 columns

c)head

In [26]:
```
dff.head(10)
```

Out[26]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freec
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63
5	Finland	Western Europe	6	7.406	0.03140	1.29025	1.31826	0.88911	0.64
6	Netherlands	Western Europe	7	7.378	0.02799	1.32944	1.28017	0.89284	0.61
7	Sweden	Western Europe	8	7.364	0.03157	1.33171	1.28907	0.91087	0.65
8	New Zealand	Australia and New Zealand	9	7.286	0.03371	1.25018	1.31967	0.90837	0.63
9	Australia	Australia and New Zealand	10	7.284	0.04083	1.33358	1.30923	0.93156	0.65

d)tail

In [27]:
```
dff.tail(10)
```

Out[27]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Fre
148	Chad	Sub-Saharan Africa	149	3.667	0.03830	0.34193	0.76062	0.15010	0
149	Guinea	Sub-Saharan Africa	150	3.656	0.03590	0.17417	0.46475	0.24009	0
150	Ivory Coast	Sub-Saharan Africa	151	3.655	0.05141	0.46534	0.77115	0.15185	0
151	Burkina Faso	Sub-Saharan Africa	152	3.587	0.04324	0.25812	0.85188	0.27125	0
152	Afghanistan	Southern Asia	153	3.575	0.03084	0.31982	0.30285	0.30335	0
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.42864	0
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910	0
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193	0
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.22396	0
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.28443	0

e)describe

In [28]:
```
dff.describe()
```

Out[28]:

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	(Go C
count	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	1
mean	79.493671	5.375734	0.047885	0.846137	0.991046	0.630259	0.428615	
std	45.754363	1.145010	0.017146	0.403121	0.272369	0.247078	0.150693	
min	1.000000	2.839000	0.018480	0.000000	0.000000	0.000000	0.000000	
25%	40.250000	4.526000	0.037268	0.545808	0.856823	0.439185	0.328330	
50%	79.500000	5.232500	0.043940	0.910245	1.029510	0.696705	0.435515	
75%	118.750000	6.243750	0.052300	1.158448	1.214405	0.811013	0.549092	
max	158.000000	7.587000	0.136930	1.690420	1.402230	1.025250	0.669730	

f)shape

In [29]:
```
dff.shape
```

Out[29]:
```
(158, 12)
```

g)size

In [30]:
```
dff.size
```

Out[30]:
```
1896
```

h)find missing values

In [31]:
```
dff.isnull()
```

Out[31]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedon
0	False	False	False	False	False	False	False	False	Fals
1	False	False	False	False	False	False	False	False	Fals
2	False	False	False	False	False	False	False	False	Fals
3	False	False	False	False	False	False	False	False	Fals
4	False	False	False	False	False	False	False	False	Fals
...	...	...	...	...	...	...	...	...	..
153	False	False	False	False	False	False	False	False	Fals
154	False	False	False	False	False	False	False	False	Fals
155	False	False	False	False	False	False	False	False	Fals
156	False	False	False	False	False	False	False	False	Fals
157	False	False	False	False	False	False	False	False	Fals

158 rows × 12 columns

i)fill/drop

In [32]:

```
dff1=dff
dff1.fillna(value=0)
```

Out[32]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Fre
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.
...	...	...	...	...	...	...	...	...	
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.42864	0.
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910	0.
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193	0.
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.22396	0.
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.28443	0.

158 rows × 12 columns

In [34]:
```
dff2=dff
dff2.dropna()
```

Out[34]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Fre
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.
...	...	...	...	...	...	...	...	...	
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.42864	0.
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910	0.
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193	0.
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.22396	0.
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.28443	0.

158 rows × 12 columns