Pandas

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])
A
```

Out[2]:
```
0    1
1    2
2    3
3    4
dtype: int64
```

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

In [3]:
```
df=pd.DataFrame({"A":pd.Series(range(5)),"B":pd.Series(5),"C":pd.Series(range
                 "F":pd.Series(range(24,29)),"G":pd.Series(range(6,11)),"F":pd
df
```

Out[3]:

	A	B	C	D	E	F	G	H	I	J
0	0	5.0	6	12	18	24	6	24	6	NaN
1	1	NaN	7	13	19	25	7	25	7	NaN
2	2	NaN	8	14	20	26	8	26	8	NaN
3	3	NaN	9	15	21	27	9	27	9	NaN
4	4	NaN	10	16	22	28	10	28	10	NaN

In [4]:
```
df.shape
```

Out[4]:
```
(5, 10)
```

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

In [5]:
```
df1=pd.concat([df,df])
ind=pd.Series(range(10))
df1=df1.set_index(ind)
print(df1.head(7))
df1.tail(6)
```

```
   A    B   C   D   E   F   G   H   I   J
0  0  5.0   6  12  18  24   6  24   6 NaN
1  1  NaN   7  13  19  25   7  25   7 NaN
2  2  NaN   8  14  20  26   8  26   8 NaN
3  3  NaN   9  15  21  27   9  27   9 NaN
4  4  NaN  10  16  22  28  10  28  10 NaN
5  0  5.0   6  12  18  24   6  24   6 NaN
6  1  NaN   7  13  19  25   7  25   7 NaN
```

Out[5]:

	A	B	C	D	E	F	G	H	I	J
4	4	NaN	10	16	22	28	10	28	10	NaN
5	0	5.0	6	12	18	24	6	24	6	NaN
6	1	NaN	7	13	19	25	7	25	7	NaN
7	2	NaN	8	14	20	26	8	26	8	NaN
8	3	NaN	9	15	21	27	9	27	9	NaN
9	4	NaN	10	16	22	28	10	28	10	NaN

4. Fill with a constant value and print the output

In [6]:
```
df2=df1.fillna(value=7)
df2
```

Out[6]:

	A	B	C	D	E	F	G	H	I	J
0	0	5.0	6	12	18	24	6	24	6	7.0
1	1	7.0	7	13	19	25	7	25	7	7.0
2	2	7.0	8	14	20	26	8	26	8	7.0
3	3	7.0	9	15	21	27	9	27	9	7.0
4	4	7.0	10	16	22	28	10	28	10	7.0
5	0	5.0	6	12	18	24	6	24	6	7.0
6	1	7.0	7	13	19	25	7	25	7	7.0
7	2	7.0	8	14	20	26	8	26	8	7.0
8	3	7.0	9	15	21	27	9	27	9	7.0
9	4	7.0	10	16	22	28	10	28	10	7.0

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

In [7]:
```
df3=df1.dropna(axis=1)
df3
```

Out[7]:

	A	C	D	E	F	G	H	I
0	0	6	12	18	24	6	24	6
1	1	7	13	19	25	7	25	7
2	2	8	14	20	26	8	26	8
3	3	9	15	21	27	9	27	9
4	4	10	16	22	28	10	28	10
5	0	6	12	18	24	6	24	6
6	1	7	13	19	25	7	25	7
7	2	8	14	20	26	8	26	8
8	3	9	15	21	27	9	27	9
9	4	10	16	22	28	10	28	10

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

In [8]:
```
df4=df1.dropna()
df4
```

Out[8]:

	A	B	C	D	E	F	G	H	I	J

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

In [9]:
```
df1.isna()
```

Out[9]:

	A	B	C	D	E	F	G	H	I	J
0	False	False	False	False	False	False	False	False	False	True
1	False	True	False	False	False	False	False	False	False	True
2	False	True	False	False	False	False	False	False	False	True
3	False	True	False	False	False	False	False	False	False	True
4	False	True	False	False	False	False	False	False	False	True
5	False	False	False	False	False	False	False	False	False	True
6	False	True	False	False	False	False	False	False	False	True
7	False	True	False	False	False	False	False	False	False	True
8	False	True	False	False	False	False	False	False	False	True
9	False	True	False	False	False	False	False	False	False	True

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

In [10]:
```
df5=df1[df1>8]
df5
```

Out[10]:

	A	B	C	D	E	F	G	H	I	J
0	NaN	NaN	NaN	12	18	24	NaN	24	NaN	NaN
1	NaN	NaN	NaN	13	19	25	NaN	25	NaN	NaN
2	NaN	NaN	NaN	14	20	26	NaN	26	NaN	NaN
3	NaN	NaN	9.0	15	21	27	9.0	27	9.0	NaN
4	NaN	NaN	10.0	16	22	28	10.0	28	10.0	NaN
5	NaN	NaN	NaN	12	18	24	NaN	24	NaN	NaN
6	NaN	NaN	NaN	13	19	25	NaN	25	NaN	NaN
7	NaN	NaN	NaN	14	20	26	NaN	26	NaN	NaN
8	NaN	NaN	9.0	15	21	27	9.0	27	9.0	NaN
9	NaN	NaN	10.0	16	22	28	10.0	28	10.0	NaN

In [11]:
```
df6=df1[df2==7]
df6
```

Out[11]:

	A	B	C	D	E	F	G	H	I	J
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	7.0	NaN	NaN	NaN	7.0	NaN	7.0	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	NaN	NaN	7.0	NaN	NaN	NaN	7.0	NaN	7.0	NaN
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

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

In [12]:
```
df.loc["A":"J"]
```

Out[12]:

	A	B	C	D	E	F	G	H	I	J

In [13]:
```
df1.loc[0:5]
```

Out[13]:

	A	B	C	D	E	F	G	H	I	J
0	0	5.0	6	12	18	24	6	24	6	NaN
1	1	NaN	7	13	19	25	7	25	7	NaN
2	2	NaN	8	14	20	26	8	26	8	NaN
3	3	NaN	9	15	21	27	9	27	9	NaN
4	4	NaN	10	16	22	28	10	28	10	NaN
5	0	5.0	6	12	18	24	6	24	6	NaN

In [14]:
```
df1.iloc[6:9]
```

Out[14]:

	A	B	C	D	E	F	G	H	I	J
6	1	NaN	7	13	19	25	7	25	7	NaN
7	2	NaN	8	14	20	26	8	26	8	NaN
8	3	NaN	9	15	21	27	9	27	9	NaN

10. Display the statistical summary of data

In [15]:
```
df1.describe()
```

Out[15]:

	A	B	C	D	E	F	G	H	
count	10.000000	2.0	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000
mean	2.000000	5.0	8.000000	14.000000	20.000000	26.000000	8.000000	26.000000	8.000
std	1.490712	0.0	1.490712	1.490712	1.490712	1.490712	1.490712	1.490712	1.490
min	0.000000	5.0	6.000000	12.000000	18.000000	24.000000	6.000000	24.000000	6.000
25%	1.000000	5.0	7.000000	13.000000	19.000000	25.000000	7.000000	25.000000	7.000
50%	2.000000	5.0	8.000000	14.000000	20.000000	26.000000	8.000000	26.000000	8.000
75%	3.000000	5.0	9.000000	15.000000	21.000000	27.000000	9.000000	27.000000	9.000
max	4.000000	5.0	10.000000	16.000000	22.000000	28.000000	10.000000	28.000000	10.000

MINI-PROJECT: Analyse the given dataset (refer to the link in whatsapp group description) and perform basic analysis using numpy and pandas

In [16]:
```
dat1=pd.read_csv("2015 - 2015.csv")
dat1
```

Out[16]:

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

158 rows × 12 columns

In [24]:
```
dat1.head()
```

Out[24]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Free
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.6
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.6
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.6
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.6
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.6

In [25]:
```
dat1.tail()
```

Out[25]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Free
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.42864	0.5
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910	0.4
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193	0.1
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.22396	0.1
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.28443	0.3

In [26]:
```
dat1.describe()
```

Out[26]:

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	(
count	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	
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	

In [28]:
```
dat1.shape
```

Out[28]: (158, 12)

In [29]:
```
dat1.size
```

Out[29]: 1896

In [30]:
```
dat1.isna()
```

Out[30]:

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

158 rows × 12 columns

In [37]:
```
dat1.dropna()
```

Out[37]:

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

158 rows × 12 columns

In [32]:
```
dat2=pd.read_csv("fiat500_VehicleSelection_Dataset - fiat500_VehicleSelection
dat2
```

Out[32]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.61155
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.2418
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.4
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.6346
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.4956
...	...	...	...	...	...	...	...	
1544	NaN	NaN	NaN	NaN	NaN	NaN	NaN	le
1545	NaN	NaN	NaN	NaN	NaN	NaN	NaN	cc
1546	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Null va
1547	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1548	NaN	NaN	NaN	NaN	NaN	NaN	NaN	se

1549 rows × 11 columns

In [33]:
```
dat2.head()
```

Out[33]:

	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

In [34]:
```
dat2.tail()
```

Out[34]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price	l
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	

In [35]:
```
dat2.describe()
```

Out[35]:

	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

In [36]:
```
dat2.shape
```

Out[36]: (1549, 11)

In [38]:
```
dat2.size
```

Out[38]: 17039

In [40]:
```
dat2.isna()
```

Out[40]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price	U
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

In [41]:
```
dat2.fillna(value=4)
```

Out[41]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	
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	4.0	4	4.0	4.0	4.0	4.0	4.000000	len
1545	4.0	4	4.0	4.0	4.0	4.0	4.000000	con
1546	4.0	4	4.0	4.0	4.0	4.0	4.000000	Null valu
1547	4.0	4	4.0	4.0	4.0	4.0	4.000000	f
1548	4.0	4	4.0	4.0	4.0	4.0	4.000000	sea

1549 rows × 11 columns

In [ ]: