PRACTICAL-1

AIM: Basic Understanding of Data Science and frequently useful libraries.

1. Perform basic data analysis and merge – sort operations over dataset zoo.csv

a. Read csv file using pandas

Code:

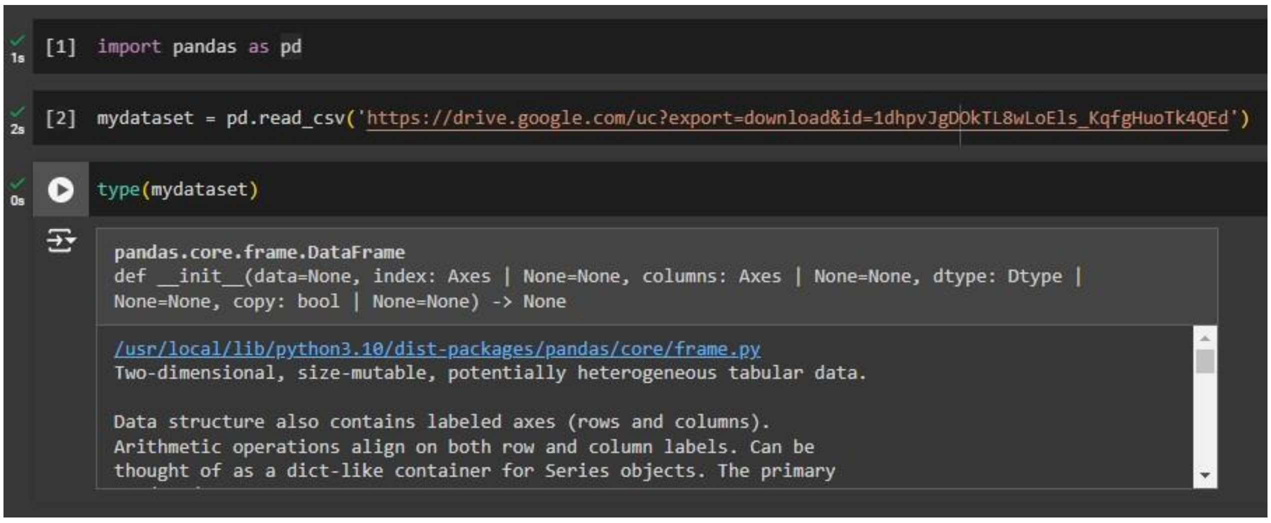
import pandas as pd

mydataset =

pd.read_csv('https://drive.google.com/uc?export=download&id=1dhpvJgDOkTL8wLoEls_
KqfgHuoTk4QEd')

type(mydataset)

Output:



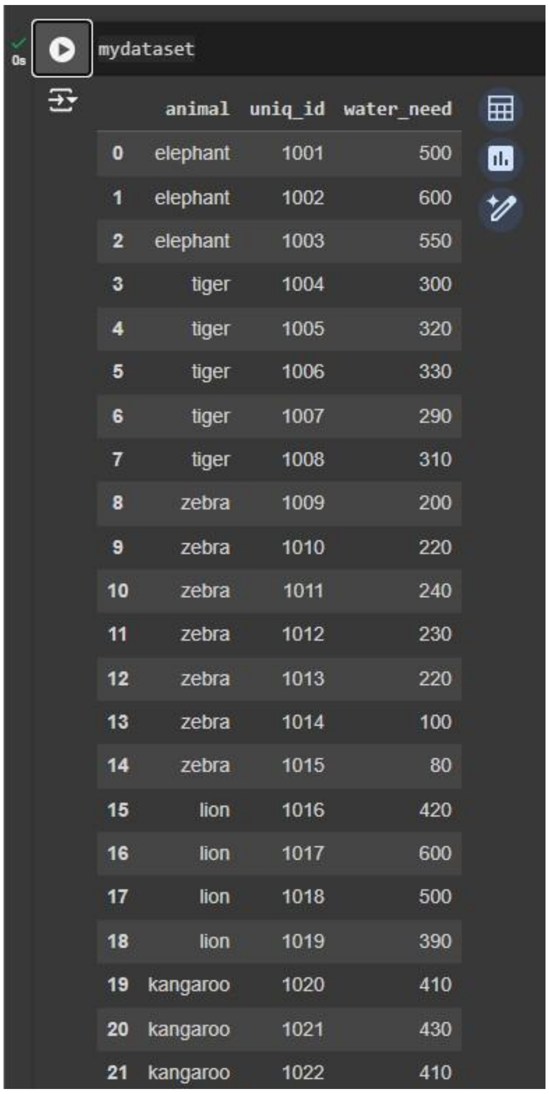
```
[1] import pandas as pd

[2] mydataset = pd.read_csv('https://drive.google.com/uc?export=download&id=1dhpvJgDOkTL8wLoEls_KqfgHuoTk4QEd')

type(mydataset)

pandas.core.frame.DataFrame
def __init__(data=None, index: Axes | None=None, columns: Axes | None=None, dtype: Dtype |
None=None, copy: bool | None=None) -> None

/usr/local/lib/python3.10/dist-packages/pandas/core/frame.py
Two-dimensional, size-mutable, potentially heterogeneous tabular data.

Data structure also contains labeled axes (rows and columns).
Arithmetic operations align on both row and column labels. Can be
thought of as a dict-like container for Series objects. The primary
```

b. Basic data analysis.

Code:

mydataset

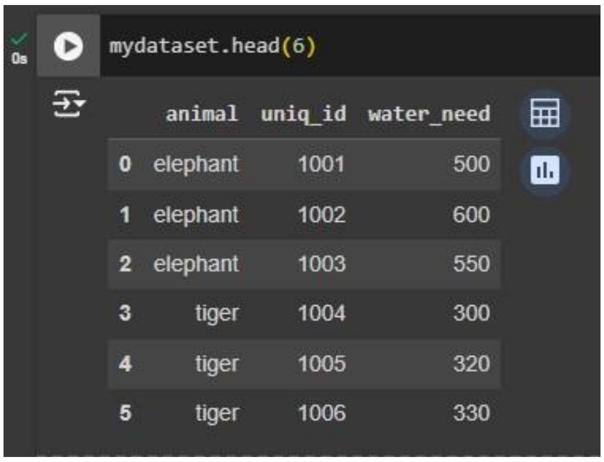
Output:



```
mydataset
```

	animal	uniq_id	water_need
0	elephant	1001	500
1	elephant	1002	600
2	elephant	1003	550
3	tiger	1004	300
4	tiger	1005	320
5	tiger	1006	330
6	tiger	1007	290
7	tiger	1008	310
8	zebra	1009	200
9	zebra	1010	220
10	zebra	1011	240
11	zebra	1012	230
12	zebra	1013	220
13	zebra	1014	100
14	zebra	1015	80
15	lion	1016	420
16	lion	1017	600
17	lion	1018	500
18	lion	1019	390
19	kangaroo	1020	410
20	kangaroo	1021	430
21	kangaroo	1022	410

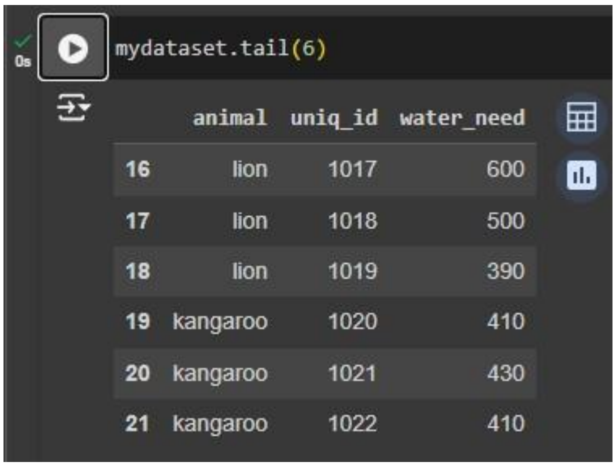
mydataset.head(6)



```
mydataset.head(6)
```

	animal	uniq_id	water_need
0	elephant	1001	500
1	elephant	1002	600
2	elephant	1003	550
3	tiger	1004	300
4	tiger	1005	320
5	tiger	1006	330

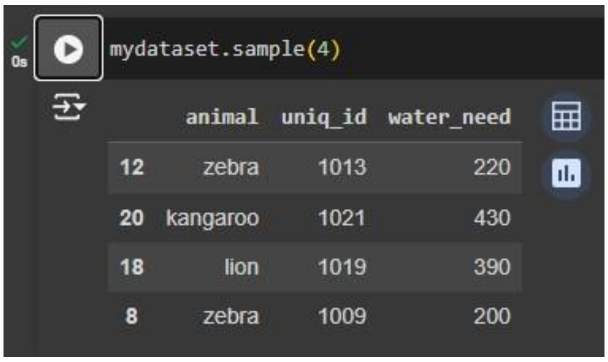
mydataset.tail(6)



```
mydataset.tail(6)
```

	animal	uniq_id	water_need
16	lion	1017	600
17	lion	1018	500
18	lion	1019	390
19	kangaroo	1020	410
20	kangaroo	1021	430
21	kangaroo	1022	410

mydataset.sample(4)



```
mydataset.sample(4)
```

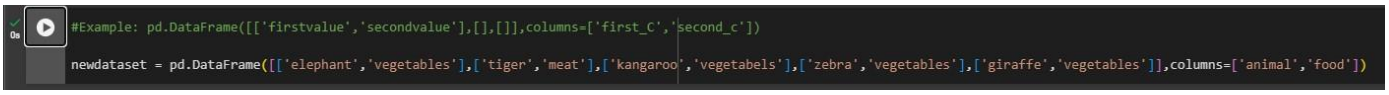
	animal	uniq_id	water_need
12	zebra	1013	220
20	kangaroo	1021	430
18	lion	1019	390
8	zebra	1009	200

c. Create new dataset and merge it with zoo dataset.

Code: newdataset =

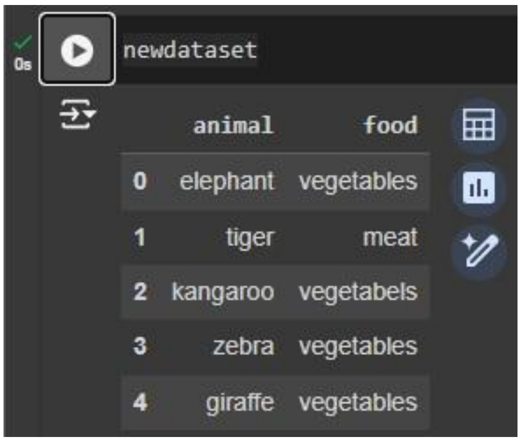
pd.DataFrame([['elephant','vegetables'],['tiger','meat'],['kangaroo','vegetabels'],['zebra','vegetables'],['giraffe','vegetables']],columns=['animal','food'])

Output:



```
#Example: pd.DataFrame([['firstvalue','secondvalue'],[],[]],columns=['first_C','second_c'])
newdataset = pd.DataFrame([['elephant','vegetables'],['tiger','meat'],['kangaroo','vegetabels'],['zebra','vegetables'],['giraffe','vegetables']],columns=['animal','food'])
```

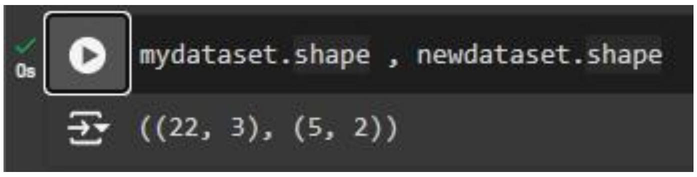
newdataset



```
newdataset
```

	animal	food
0	elephant	vegetables
1	tiger	meat
2	kangaroo	vegetabels
3	zebra	vegetables
4	giraffe	vegetables

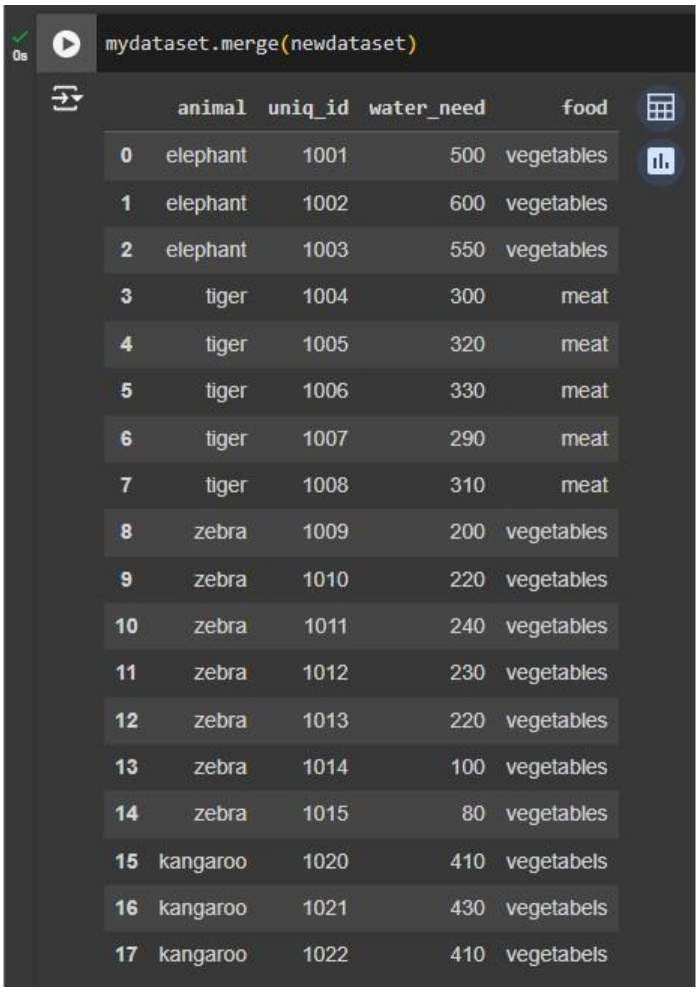
mydataset.shape , newdataset.shape



```
mydataset.shape , newdataset.shape
```

```
((22, 3), (5, 2))
```

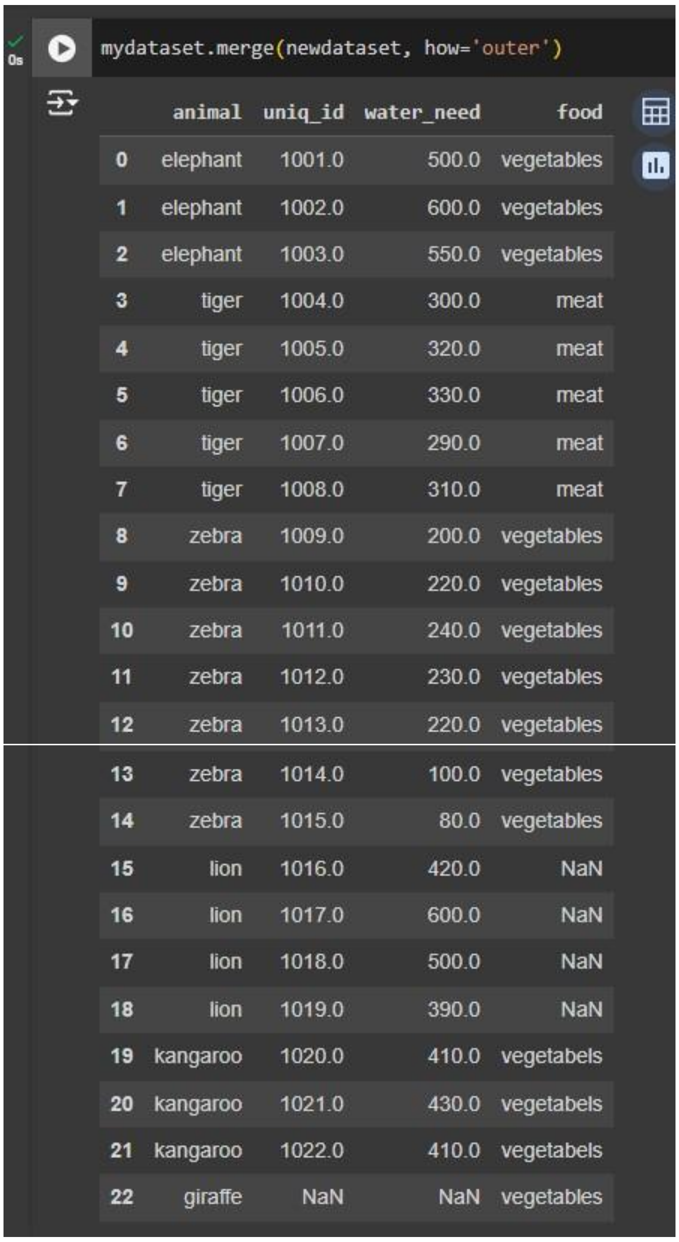
mydataset.merge(newdataset)



```
mydataset.merge(newdataset)
```

	animal	uniq_id	water_need	food
0	elephant	1001	500	vegetables
1	elephant	1002	600	vegetables
2	elephant	1003	550	vegetables
3	tiger	1004	300	meat
4	tiger	1005	320	meat
5	tiger	1006	330	meat
6	tiger	1007	290	meat
7	tiger	1008	310	meat
8	zebra	1009	200	vegetables
9	zebra	1010	220	vegetables
10	zebra	1011	240	vegetables
11	zebra	1012	230	vegetables
12	zebra	1013	220	vegetables
13	zebra	1014	100	vegetables
14	zebra	1015	80	vegetables
15	kangaroo	1020	410	vegetabels
16	kangaroo	1021	430	vegetabels
17	kangaroo	1022	410	vegetabels

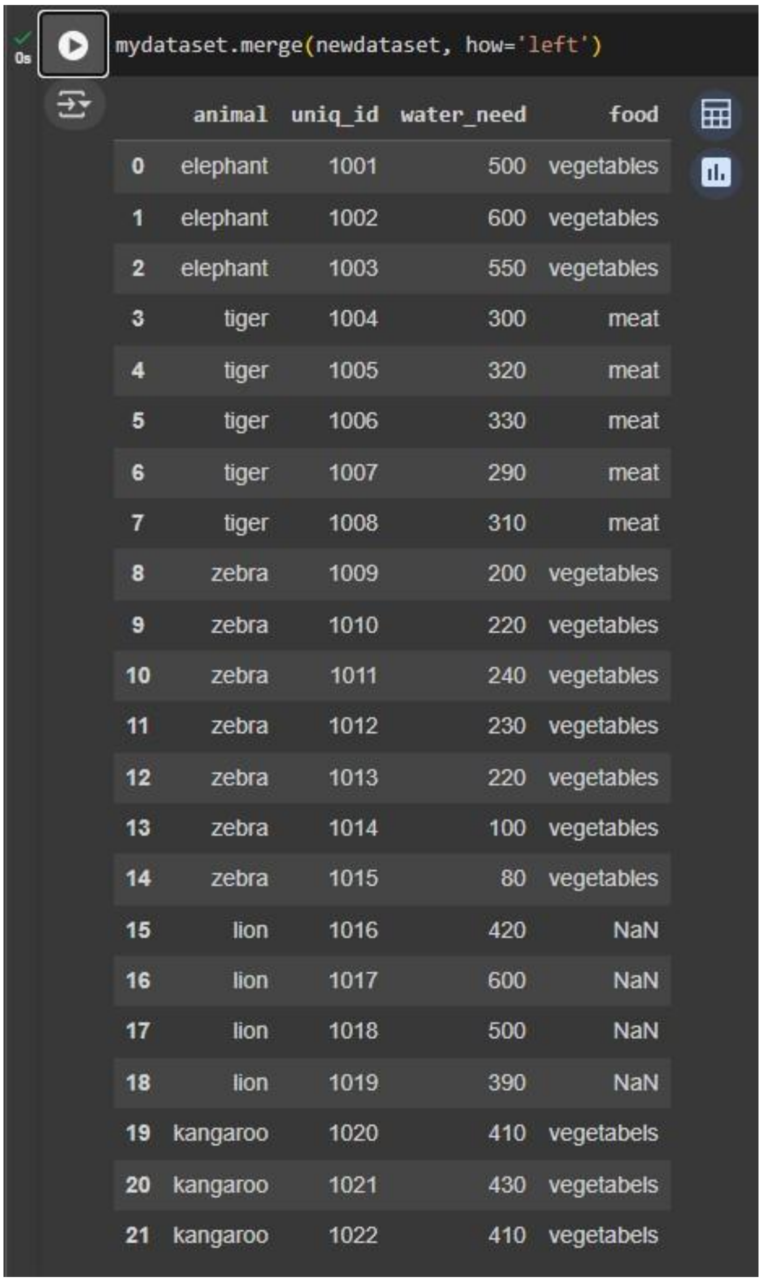
mydataset.merge(newdataset, how='outer')



```
mydataset.merge(newdataset, how='outer')
```

```
      animal  uniq_id  water_need        food
0   elephant   1001.0       500.0  vegetables
1   elephant   1002.0       600.0  vegetables
2   elephant   1003.0       550.0  vegetables
3      tiger   1004.0       300.0        meat
4      tiger   1005.0       320.0        meat
5      tiger   1006.0       330.0        meat
6      tiger   1007.0       290.0        meat
7      tiger   1008.0       310.0        meat
8      zebra   1009.0       200.0  vegetables
9      zebra   1010.0       220.0  vegetables
10     zebra   1011.0       240.0  vegetables
11     zebra   1012.0       230.0  vegetables
12     zebra   1013.0       220.0  vegetables
13     zebra   1014.0       100.0  vegetables
14     zebra   1015.0        80.0  vegetables
15      lion   1016.0       420.0         NaN
16      lion   1017.0       600.0         NaN
17      lion   1018.0       500.0         NaN
18      lion   1019.0       390.0         NaN
19  kangaroo   1020.0       410.0  vegetabels
20  kangaroo   1021.0       430.0  vegetabels
21  kangaroo   1022.0       410.0  vegetabels
22   giraffe      NaN         NaN  vegetables
```

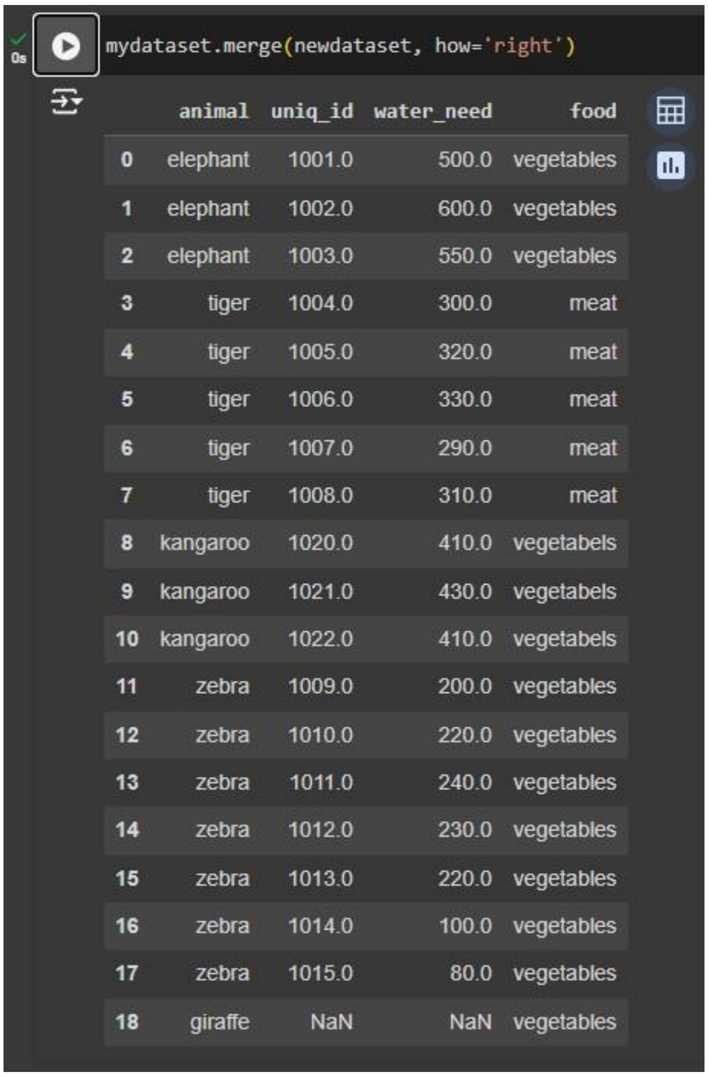
mydataset.merge(newdataset, how='left')



```
mydataset.merge(newdataset, how='left')
```

```
      animal  uniq_id  water_need        food
0   elephant     1001         500  vegetables
1   elephant     1002         600  vegetables
2   elephant     1003         550  vegetables
3      tiger     1004         300        meat
4      tiger     1005         320        meat
5      tiger     1006         330        meat
6      tiger     1007         290        meat
7      tiger     1008         310        meat
8      zebra     1009         200  vegetables
9      zebra     1010         220  vegetables
10     zebra     1011         240  vegetables
11     zebra     1012         230  vegetables
12     zebra     1013         220  vegetables
13     zebra     1014         100  vegetables
14     zebra     1015          80  vegetables
15      lion     1016         420         NaN
16      lion     1017         600         NaN
17      lion     1018         500         NaN
18      lion     1019         390         NaN
19  kangaroo     1020         410  vegetabels
20  kangaroo     1021         430  vegetabels
21  kangaroo     1022         410  vegetabels
```

mydataset.merge(newdataset, how='right')



```
mydataset.merge(newdataset, how='right')
```

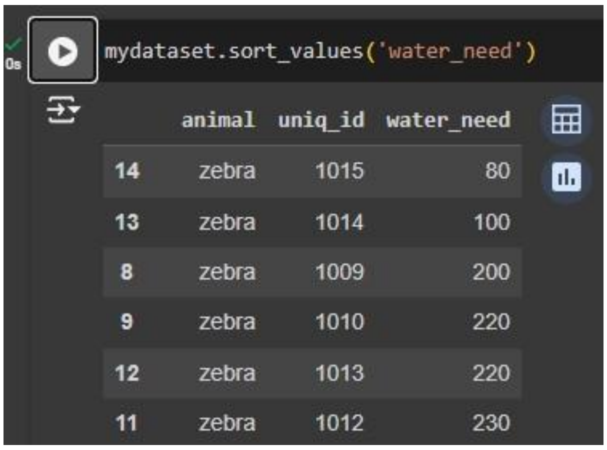
```
      animal  uniq_id  water_need        food
0   elephant   1001.0       500.0  vegetables
1   elephant   1002.0       600.0  vegetables
2   elephant   1003.0       550.0  vegetables
3      tiger   1004.0       300.0        meat
4      tiger   1005.0       320.0        meat
5      tiger   1006.0       330.0        meat
6      tiger   1007.0       290.0        meat
7      tiger   1008.0       310.0        meat
8   kangaroo   1020.0       410.0  vegetabels
9   kangaroo   1021.0       430.0  vegetabels
10  kangaroo   1022.0       410.0  vegetabels
11     zebra   1009.0       200.0  vegetables
12     zebra   1010.0       220.0  vegetables
13     zebra   1011.0       240.0  vegetables
14     zebra   1012.0       230.0  vegetables
15     zebra   1013.0       220.0  vegetables
16     zebra   1014.0       100.0  vegetables
17     zebra   1015.0        80.0  vegetables
18   giraffe      NaN         NaN  vegetables
```

d. Sort Dataset

Code:

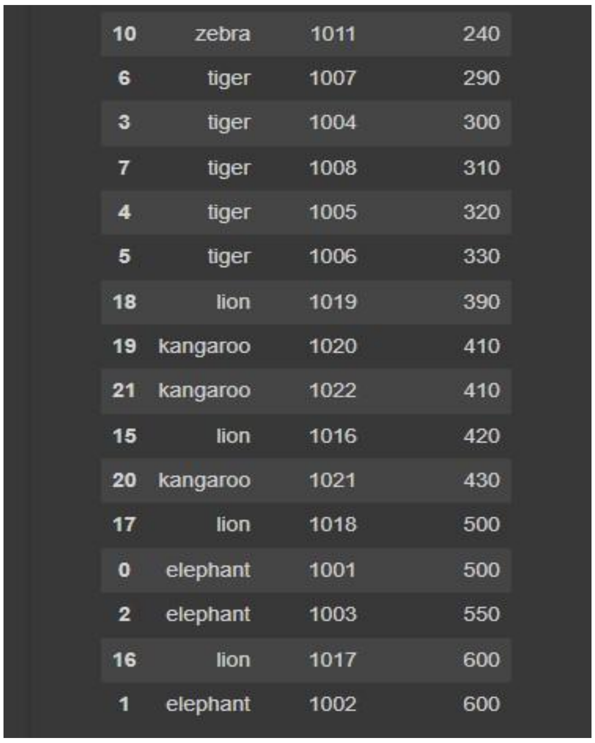
mydataset.sort_values('water_need')

Output:



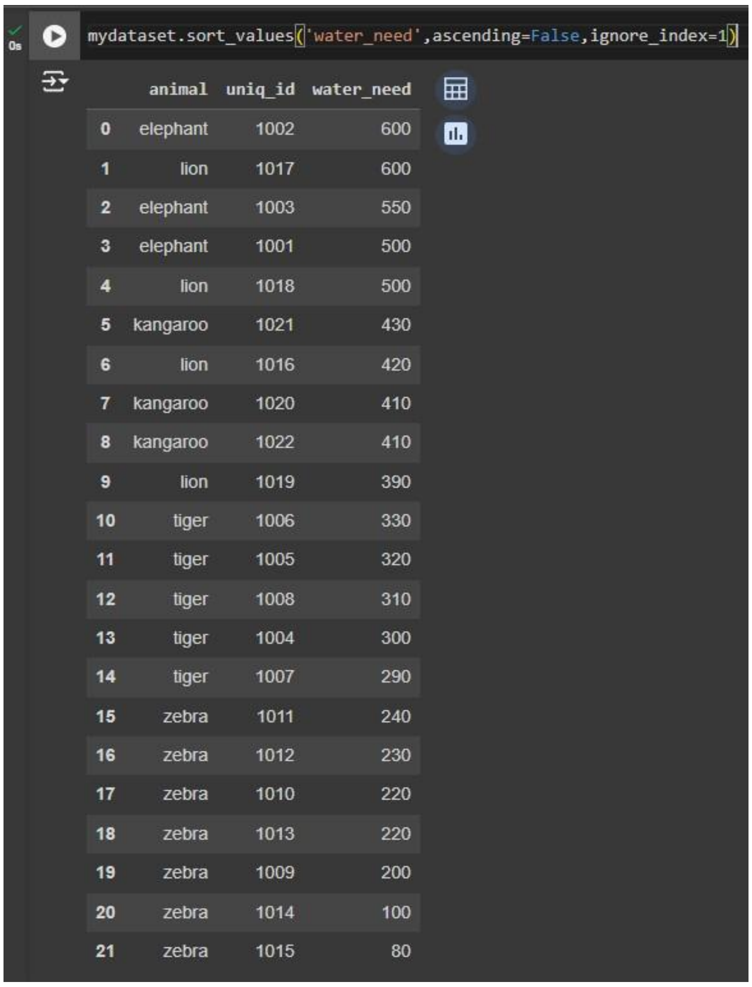
```
mydataset.sort_values('water_need')
```

```
    animal  uniq_id  water_need
14   zebra     1015          80
13   zebra     1014         100
8    zebra     1009         200
9    zebra     1010         220
12   zebra     1013         220
11   zebra     1012         230
```



```
10     zebra  1011  240
 6     tiger  1007  290
 3     tiger  1004  300
 7     tiger  1008  310
 4     tiger  1005  320
 5     tiger  1006  330
18      lion  1019  390
19  kangaroo  1020  410
21  kangaroo  1022  410
15      lion  1016  420
20  kangaroo  1021  430
17      lion  1018  500
 0  elephant  1001  500
 2  elephant  1003  550
16      lion  1017  600
 1  elephant  1002  600
```

mydataset.sort_values('water_need',ascending=False,ignore_index=1)

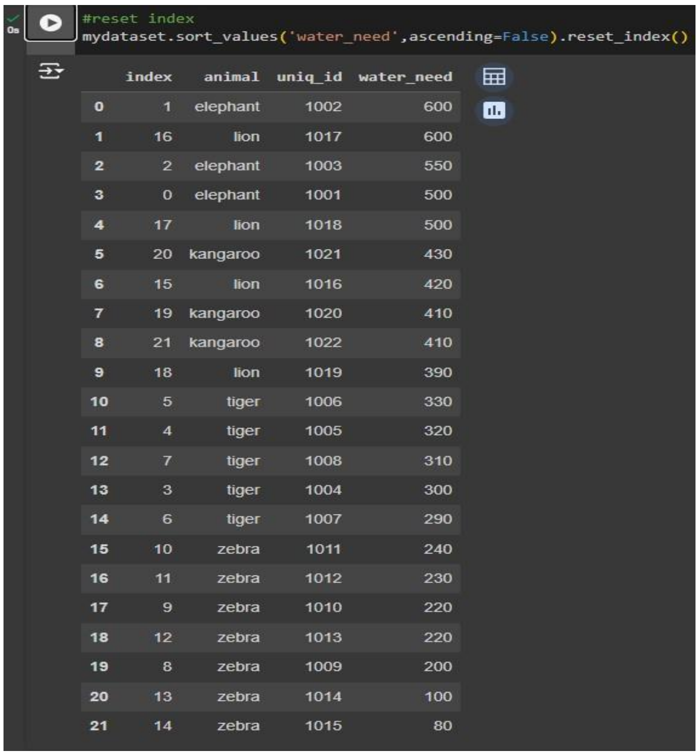


```
mydataset.sort_values('water_need',ascending=False,ignore_index=1)
```

```
      animal  uniq_id  water_need
 0  elephant     1002         600
 1      lion     1017         600
 2  elephant     1003         550
 3  elephant     1001         500
 4      lion     1018         500
 5  kangaroo     1021         430
 6      lion     1016         420
 7  kangaroo     1020         410
 8  kangaroo     1022         410
 9      lion     1019         390
10     tiger     1006         330
11     tiger     1005         320
12     tiger     1008         310
13     tiger     1004         300
14     tiger     1007         290
15     zebra     1011         240
16     zebra     1012         230
17     zebra     1010         220
18     zebra     1013         220
19     zebra     1009         200
20     zebra     1014         100
21     zebra     1015          80
```

#reset index

mydataset.sort_values('water_need',ascending=False).reset_index()



```
#reset index
mydataset.sort_values('water_need',ascending=False).reset_index()
```

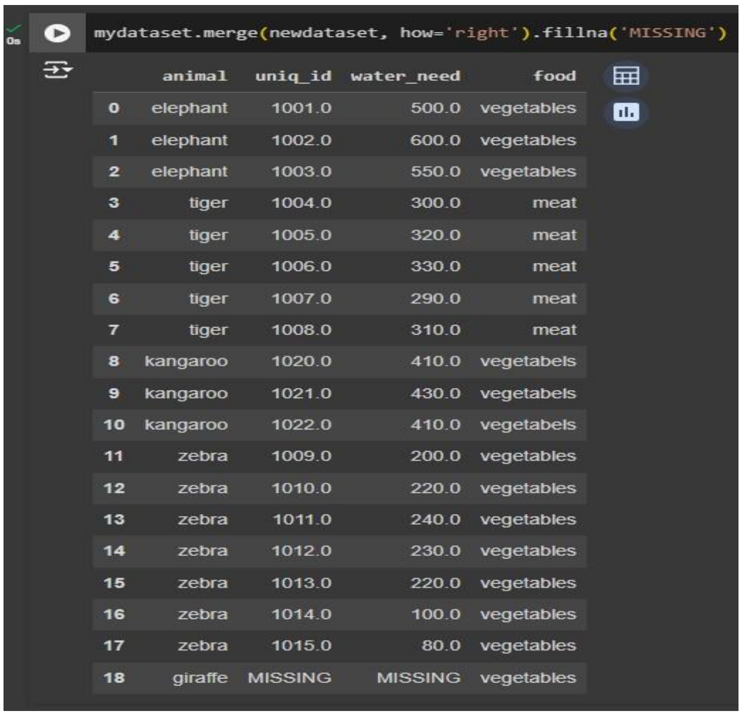
	index	animal	uniq_id	water_need
0	1	elephant	1002	600
1	16	lion	1017	600
2	2	elephant	1003	550
3	0	elephant	1001	500
4	17	lion	1018	500
5	20	kangaroo	1021	430
6	15	lion	1016	420
7	19	kangaroo	1020	410
8	21	kangaroo	1022	410
9	18	lion	1019	390
10	5	tiger	1006	330
11	4	tiger	1005	320
12	7	tiger	1008	310
13	3	tiger	1004	300
14	6	tiger	1007	290
15	10	zebra	1011	240
16	11	zebra	1012	230
17	9	zebra	1010	220
18	12	zebra	1013	220
19	8	zebra	1009	200
20	13	zebra	1014	100
21	14	zebra	1015	80

e. Replace Missing Values

Code:

mydataset.merge(newdataset, how='right').fillna('MISSING')

Output:



```
mydataset.merge(newdataset, how='right').fillna('MISSING')
```

	animal	uniq_id	water_need	food
0	elephant	1001.0	500.0	vegetables
1	elephant	1002.0	600.0	vegetables
2	elephant	1003.0	550.0	vegetables
3	tiger	1004.0	300.0	meat
4	tiger	1005.0	320.0	meat
5	tiger	1006.0	330.0	meat
6	tiger	1007.0	290.0	meat
7	tiger	1008.0	310.0	meat
8	kangaroo	1020.0	410.0	vegetabels
9	kangaroo	1021.0	430.0	vegetabels
10	kangaroo	1022.0	410.0	vegetabels
11	zebra	1009.0	200.0	vegetables
12	zebra	1010.0	220.0	vegetables
13	zebra	1011.0	240.0	vegetables
14	zebra	1012.0	230.0	vegetables
15	zebra	1013.0	220.0	vegetables
16	zebra	1014.0	100.0	vegetables
17	zebra	1015.0	80.0	vegetables
18	giraffe	MISSING	MISSING	vegetables

2. Perform data analysis and data visualization over dataset Covid cases in India.xlsx

a. Read excel file using pandas.

Code:

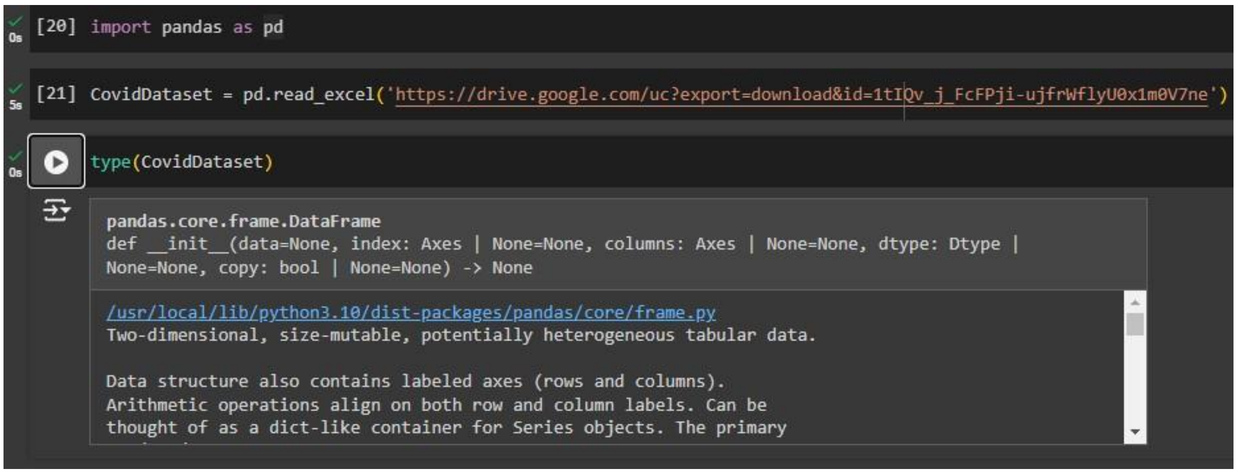
import pandas as pd

CovidDataset =

pd.read_excel('https://drive.google.com/uc?export=download&id=1tIQv_j_FcFPjiujfrWflyU0x1m0V7ne')

type(CovidDataset)

Output:

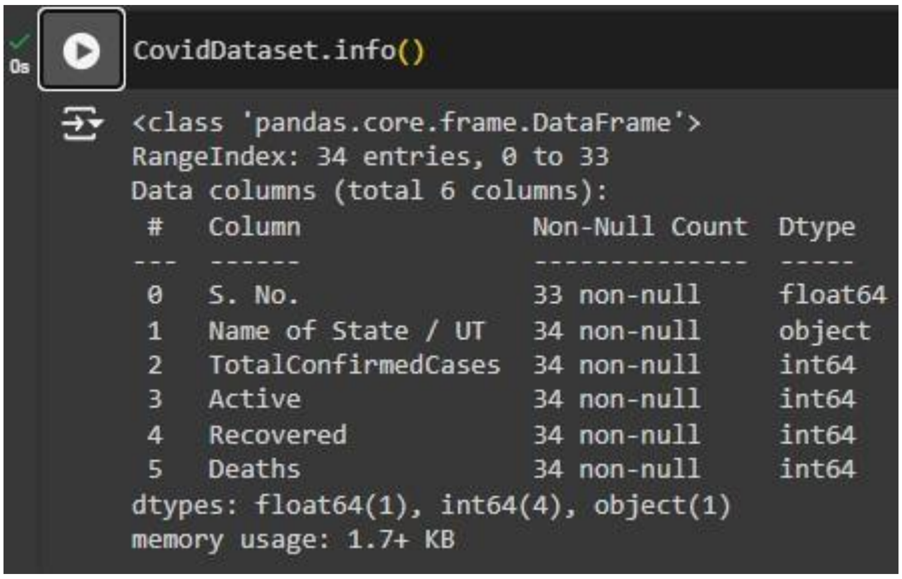


```
[20] import pandas as pd

[21] CovidDataset = pd.read_excel('https://drive.google.com/uc?export=download&id=1tIQv_j_FcFPji-ujfrWflyU0x1m0V7ne')

type(CovidDataset)

pandas.core.frame.DataFrame
def __init__(data=None, index: Axes | None=None, columns: Axes | None=None, dtype: Dtype |
None=None, copy: bool | None=None) -> None

/usr/local/lib/python3.10/dist-packages/pandas/core/frame.py
Two-dimensional, size-mutable, potentially heterogeneous tabular data.

Data structure also contains labeled axes (rows and columns).
Arithmetic operations align on both row and column labels. Can be
thought of as a dict-like container for Series objects. The primary
```

b. Basic data analysis
Code:

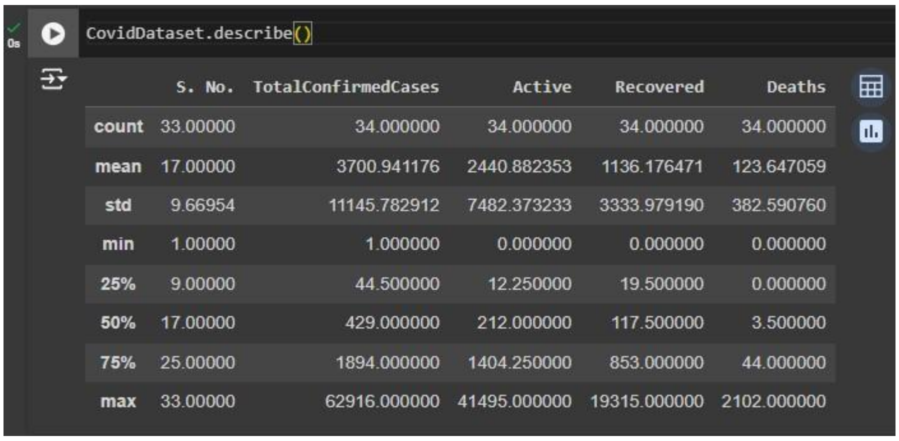
CovidDataset.info()

Output:



```
CovidDataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34 entries, 0 to 33
Data columns (total 6 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   S. No.                33 non-null     float64
 1   Name of State / UT    34 non-null     object
 2   TotalConfirmedCases   34 non-null     int64
 3   Active                34 non-null     int64
 4   Recovered             34 non-null     int64
 5   Deaths                34 non-null     int64
dtypes: float64(1), int64(4), object(1)
memory usage: 1.7+ KB
```

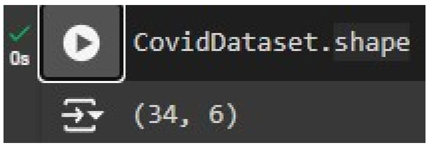
CovidDataset.describe()



```
CovidDataset.describe()
```

	S. No.	TotalConfirmedCases	Active	Recovered	Deaths
count	33.00000	34.000000	34.000000	34.000000	34.000000
mean	17.00000	3700.941176	2440.882353	1136.176471	123.647059
std	9.66954	11145.782912	7482.373233	3333.979190	382.590760
min	1.00000	1.000000	0.000000	0.000000	0.000000
25%	9.00000	44.500000	12.250000	19.500000	0.000000
50%	17.00000	429.000000	212.000000	117.500000	3.500000
75%	25.00000	1894.000000	1404.250000	853.000000	44.000000
max	33.00000	62916.000000	41495.000000	19315.000000	2102.000000

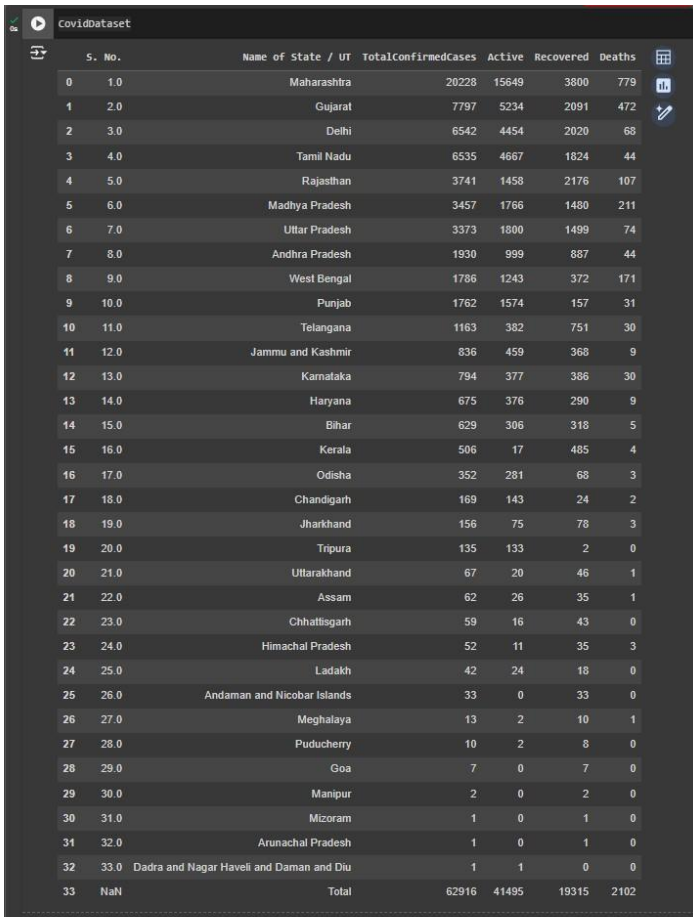
CovidDataset.shape



```
CovidDataset.shape
(34, 6)
```

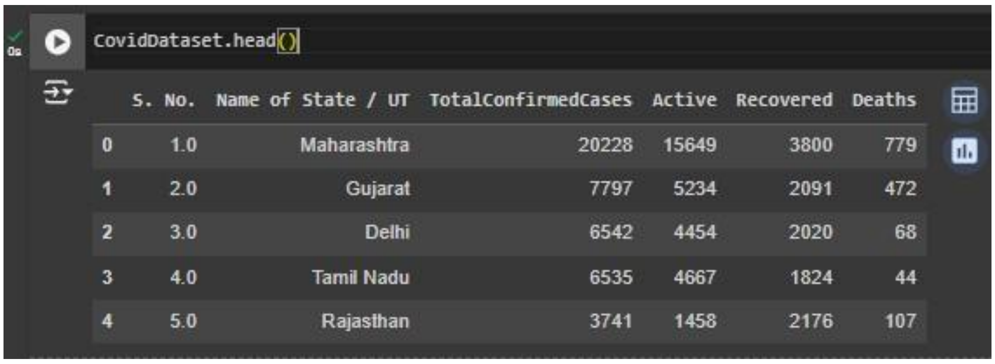
CovidDataset

CovidDataset.head()



```
CovidDataset
```

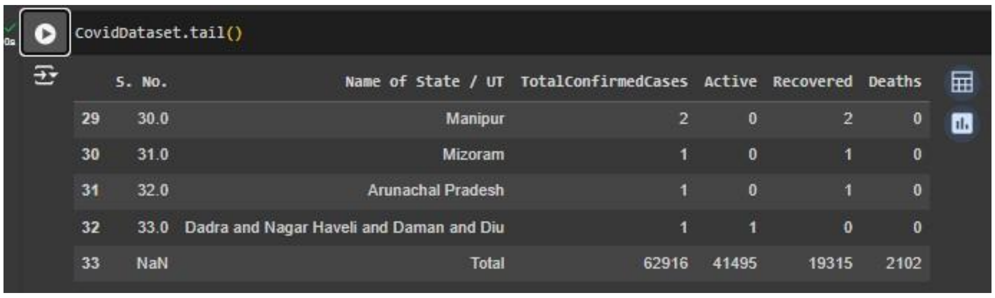
	S. No.	Name of State / UT	TotalConfirmedCases	Active	Recovered	Deaths
0	1.0	Maharashtra	20228	15649	3800	779
1	2.0	Gujarat	7797	5234	2091	472
2	3.0	Delhi	6542	4454	2020	68
3	4.0	Tamil Nadu	6535	4667	1824	44
4	5.0	Rajasthan	3741	1458	2176	107
5	6.0	Madhya Pradesh	3457	1766	1480	211
6	7.0	Uttar Pradesh	3373	1800	1499	74
7	8.0	Andhra Pradesh	1930	999	887	44
8	9.0	West Bengal	1786	1243	372	171
9	10.0	Punjab	1762	1574	157	31
10	11.0	Telangana	1163	382	751	30
11	12.0	Jammu and Kashmir	836	459	368	9
12	13.0	Karnataka	794	377	386	30
13	14.0	Haryana	675	376	290	9
14	15.0	Bihar	629	306	318	5
15	16.0	Kerala	506	17	485	4
16	17.0	Odisha	352	281	68	3
17	18.0	Chandigarh	169	143	24	2
18	19.0	Jharkhand	156	75	78	3
19	20.0	Tripura	135	133	2	0
20	21.0	Uttarakhand	67	20	46	1
21	22.0	Assam	62	26	35	1
22	23.0	Chhattisgarh	59	16	43	0
23	24.0	Himachal Pradesh	52	11	35	3
24	25.0	Ladakh	42	24	18	0
25	26.0	Andaman and Nicobar Islands	33	0	33	0
26	27.0	Meghalaya	13	2	10	1
27	28.0	Puducherry	10	2	8	0
28	29.0	Goa	7	0	7	0
29	30.0	Manipur	2	0	2	0
30	31.0	Mizoram	1	0	1	0
31	32.0	Arunachal Pradesh	1	0	1	0
32	33.0	Dadra and Nagar Haveli and Daman and Diu	1	1	0	0
33	NaN	Total	62916	41495	19315	2102



```
CovidDataset.head()
```

	S. No.	Name of State / UT	TotalConfirmedCases	Active	Recovered	Deaths
0	1.0	Maharashtra	20228	15649	3800	779
1	2.0	Gujarat	7797	5234	2091	472
2	3.0	Delhi	6542	4454	2020	68
3	4.0	Tamil Nadu	6535	4667	1824	44
4	5.0	Rajasthan	3741	1458	2176	107

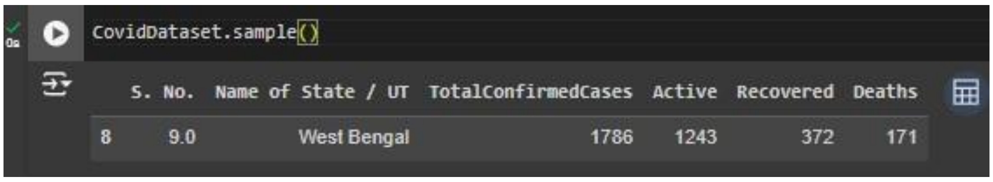
CovidDataset.tail()



```
CovidDataset.tail()
```

	S. No.	Name of State / UT	TotalConfirmedCases	Active	Recovered	Deaths
29	30.0	Manipur	2	0	2	0
30	31.0	Mizoram	1	0	1	0
31	32.0	Arunachal Pradesh	1	0	1	0
32	33.0	Dadra and Nagar Haveli and Daman and Diu	1	1	0	0
33	NaN	Total	62916	41495	19315	2102

CovidDataset.sample()



```
CovidDataset.sample()
```

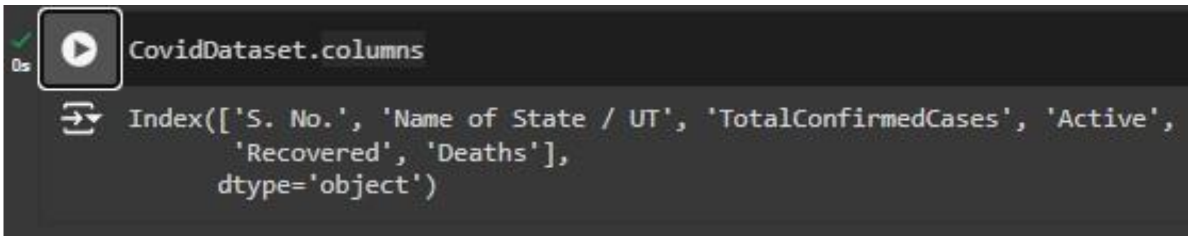
	S. No.	Name of State / UT	TotalConfirmedCases	Active	Recovered	Deaths
8	9.0	West Bengal	1786	1243	372	171

c. Display different columns

Code:

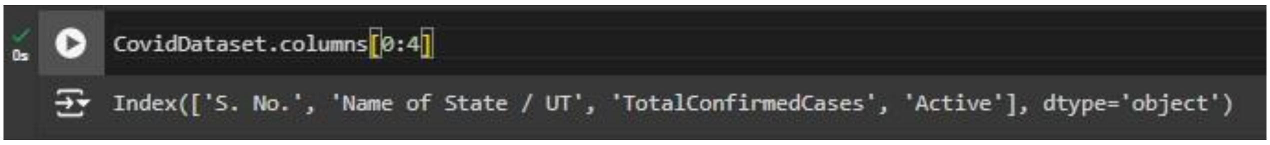
CovidDataset.columns

Output:



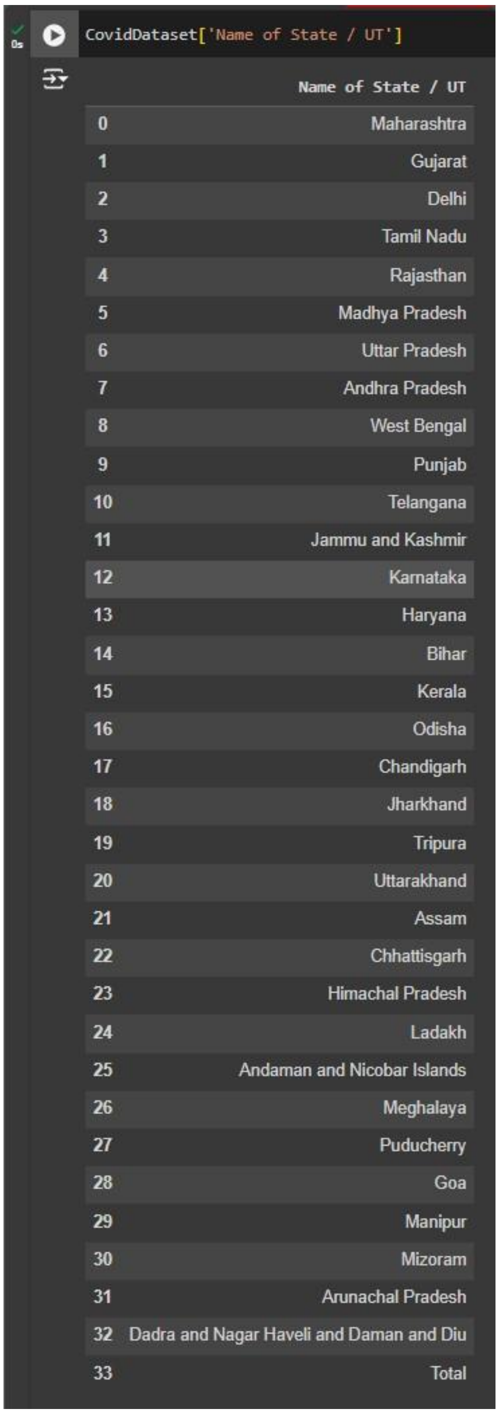
```
CovidDataset.columns

Index(['S. No.', 'Name of State / UT', 'TotalConfirmedCases', 'Active',
       'Recovered', 'Deaths'],
      dtype='object')
```

CovidDataset.columns[0:4]



```
CovidDataset.columns[0:4]

Index(['S. No.', 'Name of State / UT', 'TotalConfirmedCases', 'Active'], dtype='object')
```

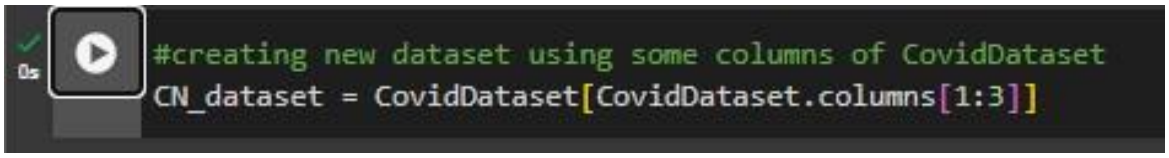
CovidDataset['Name of State / UT']



```
CovidDataset['Name of State / UT']

                          Name of State / UT
 0                               Maharashtra
 1                                   Gujarat
 2                                     Delhi
 3                                Tamil Nadu
 4                                 Rajasthan
 5                            Madhya Pradesh
 6                             Uttar Pradesh
 7                            Andhra Pradesh
 8                               West Bengal
 9                                    Punjab
10                                 Telangana
11                         Jammu and Kashmir
12                                 Karnataka
13                                   Haryana
14                                     Bihar
15                                    Kerala
16                                    Odisha
17                                Chandigarh
18                                 Jharkhand
19                                   Tripura
20                               Uttarakhand
21                                     Assam
22                              Chhattisgarh
23                          Himachal Pradesh
24                                    Ladakh
25               Andaman and Nicobar Islands
26                                 Meghalaya
27                                Puducherry
28                                       Goa
29                                   Manipur
30                                   Mizoram
31                         Arunachal Pradesh
32  Dadra and Nagar Haveli and Daman and Diu
33                                     Total
```

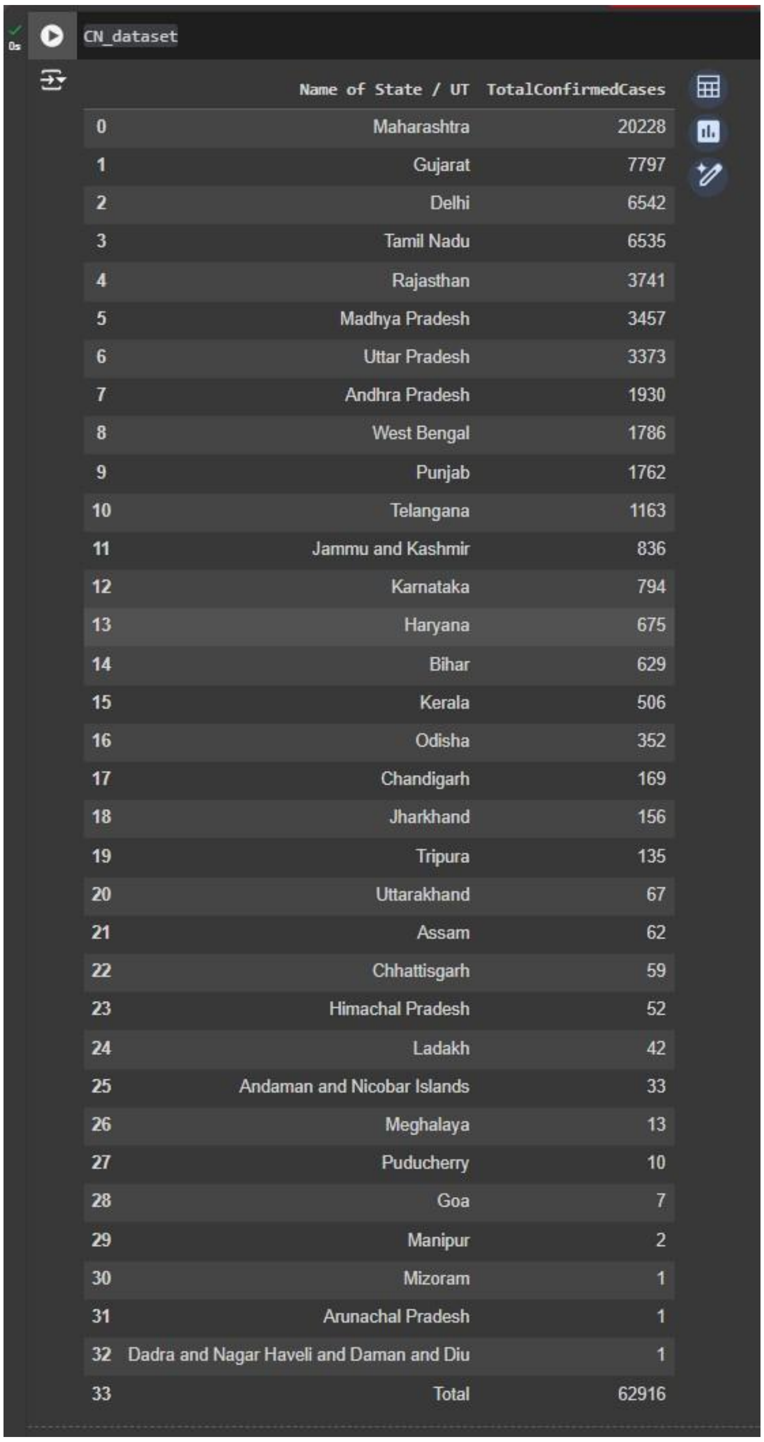
#creating new dataset using some columns of CovidDataset

CN_dataset = CovidDataset[CovidDataset.columns[1:3]]



```
#creating new dataset using some columns of CovidDataset
CN_dataset = CovidDataset[CovidDataset.columns[1:3]]
```

CN_dataset



```
CN_dataset
```

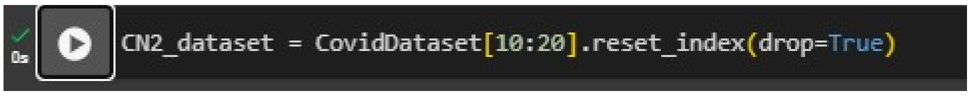
	Name of State / UT	TotalConfirmedCases
0	Maharashtra	20228
1	Gujarat	7797
2	Delhi	6542
3	Tamil Nadu	6535
4	Rajasthan	3741
5	Madhya Pradesh	3457
6	Uttar Pradesh	3373
7	Andhra Pradesh	1930
8	West Bengal	1786
9	Punjab	1762
10	Telangana	1163
11	Jammu and Kashmir	836
12	Karnataka	794
13	Haryana	675
14	Bihar	629
15	Kerala	506
16	Odisha	352
17	Chandigarh	169
18	Jharkhand	156
19	Tripura	135
20	Uttarakhand	67
21	Assam	62
22	Chhattisgarh	59
23	Himachal Pradesh	52
24	Ladakh	42
25	Andaman and Nicobar Islands	33
26	Meghalaya	13
27	Puducherry	10
28	Goa	7
29	Manipur	2
30	Mizoram	1
31	Arunachal Pradesh	1
32	Dadra and Nagar Haveli and Daman and Diu	1
33	Total	62916

d. Store different rows in new dataframe

Code:

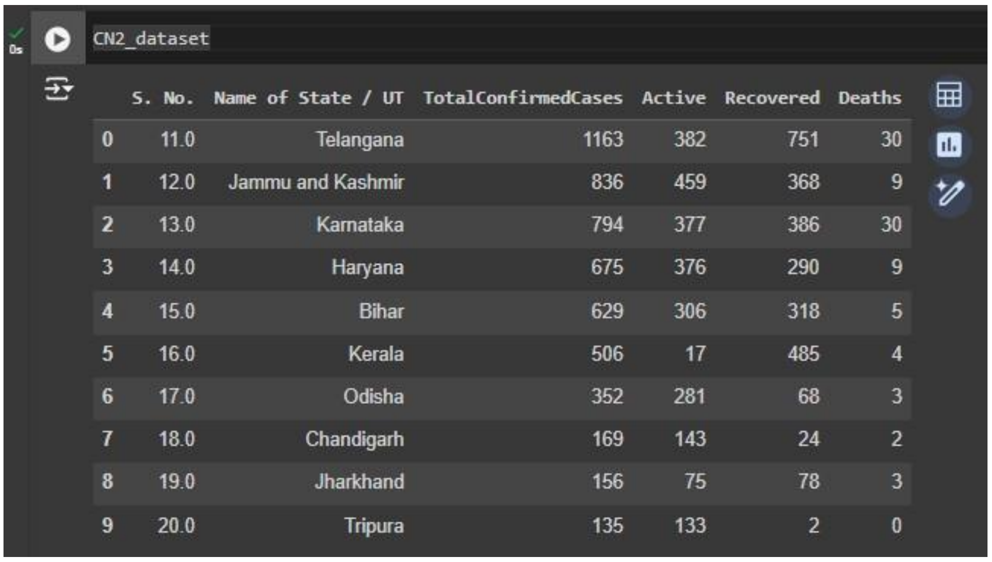
CN2_dataset = CovidDataset[10:20].reset_index(drop=True)

Output:



```
CN2_dataset = CovidDataset[10:20].reset_index(drop=True)
```

CN2_dataset



```
CN2_dataset
```

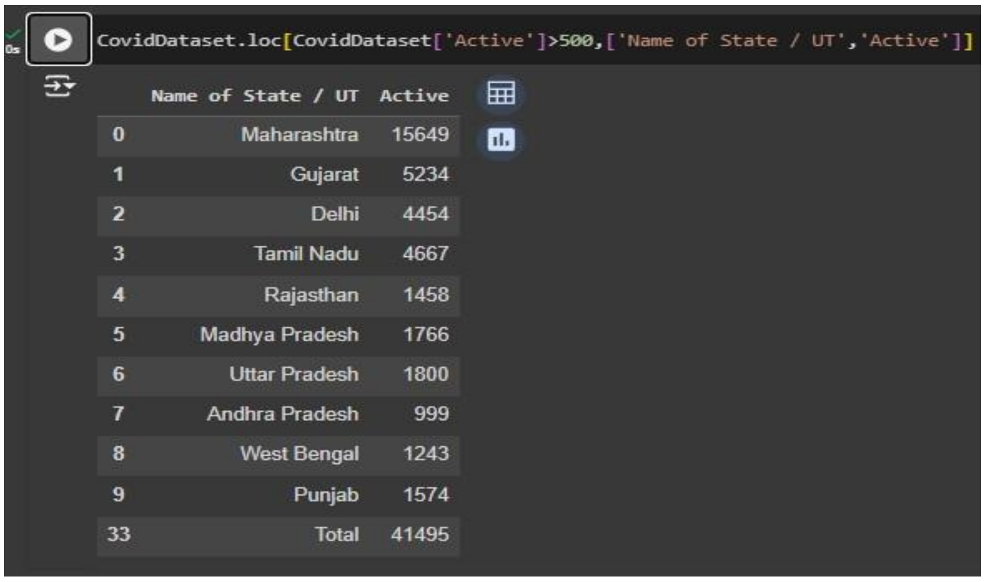
	S. No.	Name of State / UT	TotalConfirmedCases	Active	Recovered	Deaths
0	11.0	Telangana	1163	382	751	30
1	12.0	Jammu and Kashmir	836	459	368	9
2	13.0	Karnataka	794	377	386	30
3	14.0	Haryana	675	376	290	9
4	15.0	Bihar	629	306	318	5
5	16.0	Kerala	506	17	485	4
6	17.0	Odisha	352	281	68	3
7	18.0	Chandigarh	169	143	24	2
8	19.0	Jharkhand	156	75	78	3
9	20.0	Tripura	135	133	2	0

e. Identify state details with active cases > 500

Code:

CovidDataset.loc[CovidDataset['Active']>500,['Name of State / UT','Active']]

Output:



```
CovidDataset.loc[CovidDataset['Active']>500,['Name of State / UT','Active']]
```

	Name of State / UT	Active
0	Maharashtra	15649
1	Gujarat	5234
2	Delhi	4454
3	Tamil Nadu	4667
4	Rajasthan	1458
5	Madhya Pradesh	1766
6	Uttar Pradesh	1800
7	Andhra Pradesh	999
8	West Bengal	1243
9	Punjab	1574
33	Total	41495

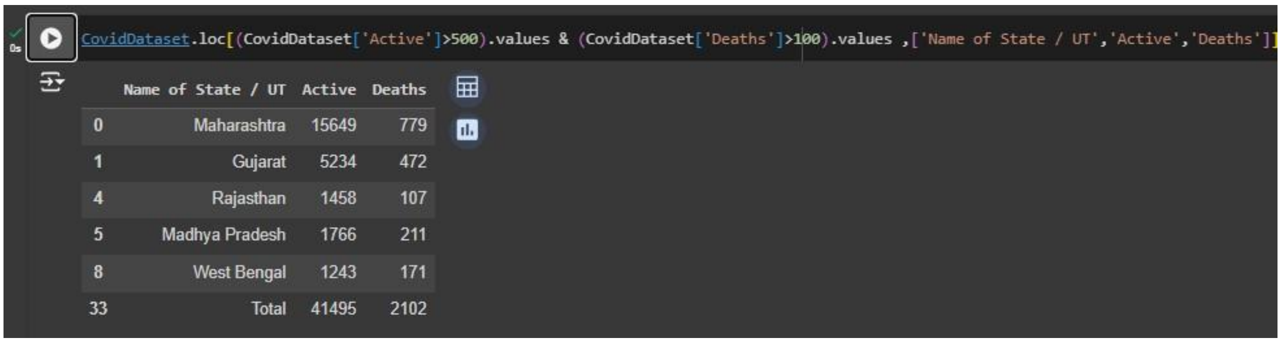
f. Print Columns: State, Active, Death with Active > 500 and death > 100

Code:

CovidDataset.loc[(CovidDataset['Active']>500).values &

(CovidDataset['Deaths']>100).values ,['Name of State / UT','Active','Deaths']]

Output:



```
CovidDataset.loc[(CovidDataset['Active']>500).values & (CovidDataset['Deaths']>100).values ,['Name of State / UT','Active','Deaths']]
```

	Name of State / UT	Active	Deaths
0	Maharashtra	15649	779
1	Gujarat	5234	472
4	Rajasthan	1458	107
5	Madhya Pradesh	1766	211
8	West Bengal	1243	171
33	Total	41495	2102

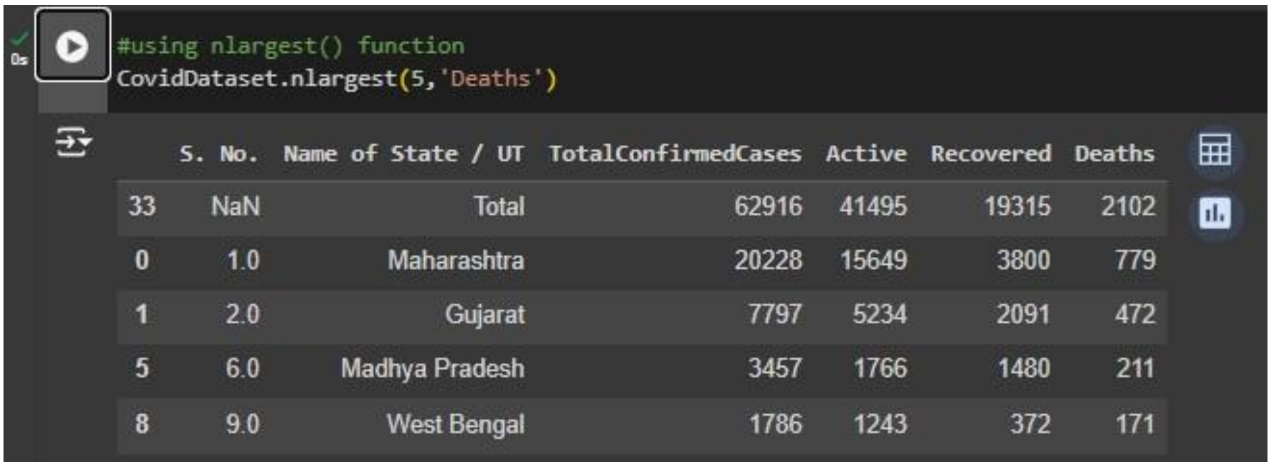
g. Display top 5 state details with heighest deaths

Code:

#using nlargest() function

CovidDataset.nlargest(5,'Deaths')

Output:

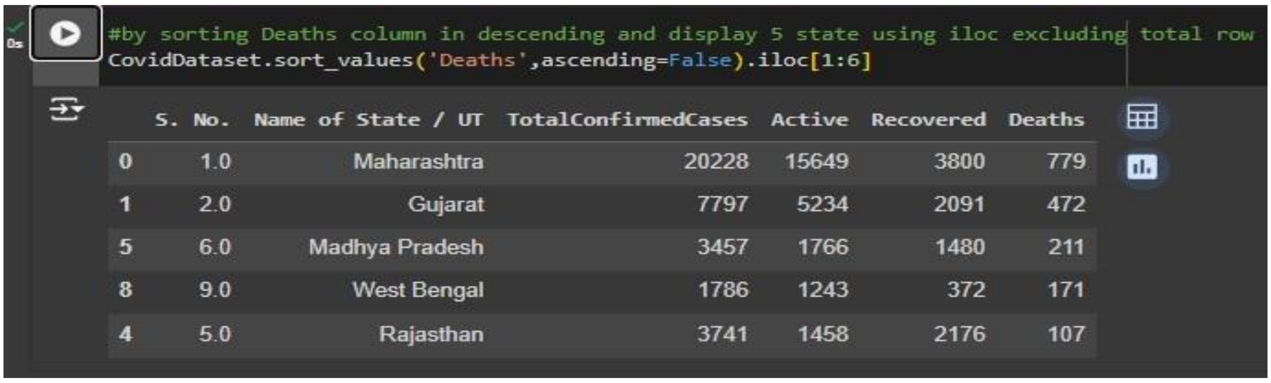


```
#using nlargest() function
CovidDataset.nlargest(5,'Deaths')
```

	S. No.	Name of State / UT	TotalConfirmedCases	Active	Recovered	Deaths
33	NaN	Total	62916	41495	19315	2102
0	1.0	Maharashtra	20228	15649	3800	779
1	2.0	Gujarat	7797	5234	2091	472
5	6.0	Madhya Pradesh	3457	1766	1480	211
8	9.0	West Bengal	1786	1243	372	171

#by sorting Deaths column in descending and display 5 state using iloc excluding total row

CovidDataset.sort_values('Deaths',ascending=False).iloc[1:6]



```
#by sorting Deaths column in descending and display 5 state using iloc excluding total row
CovidDataset.sort_values('Deaths',ascending=False).iloc[1:6]
```

	S. No.	Name of State / UT	TotalConfirmedCases	Active	Recovered	Deaths
0	1.0	Maharashtra	20228	15649	3800	779
1	2.0	Gujarat	7797	5234	2091	472
5	6.0	Madhya Pradesh	3457	1766	1480	211
8	9.0	West Bengal	1786	1243	372	171
4	5.0	Rajasthan	3741	1458	2176	107

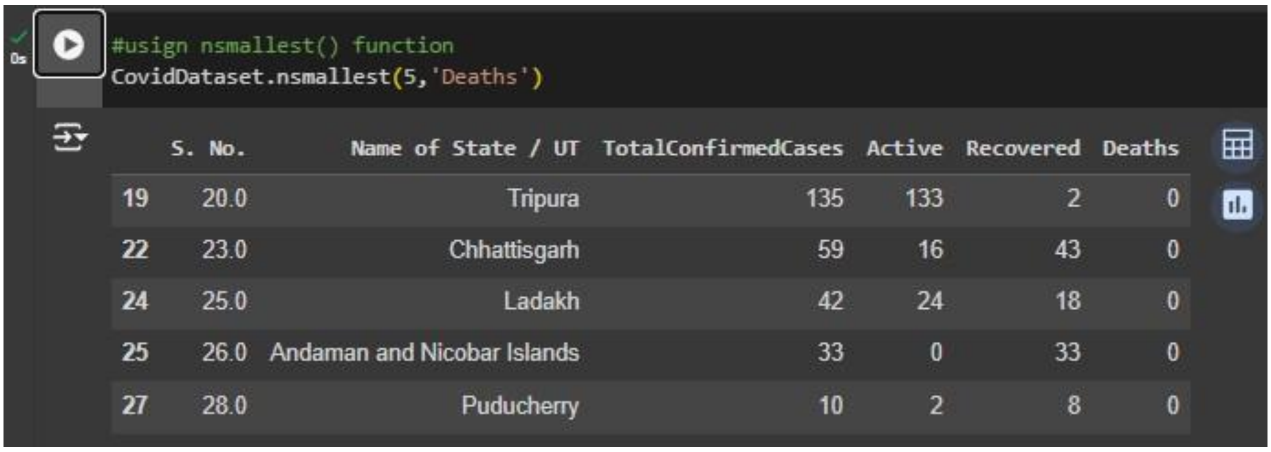
h. Display 5 states with least value of deaths

Code:

#usign nsmallest() function

CovidDataset.nsmallest(5,'Deaths')

Output:

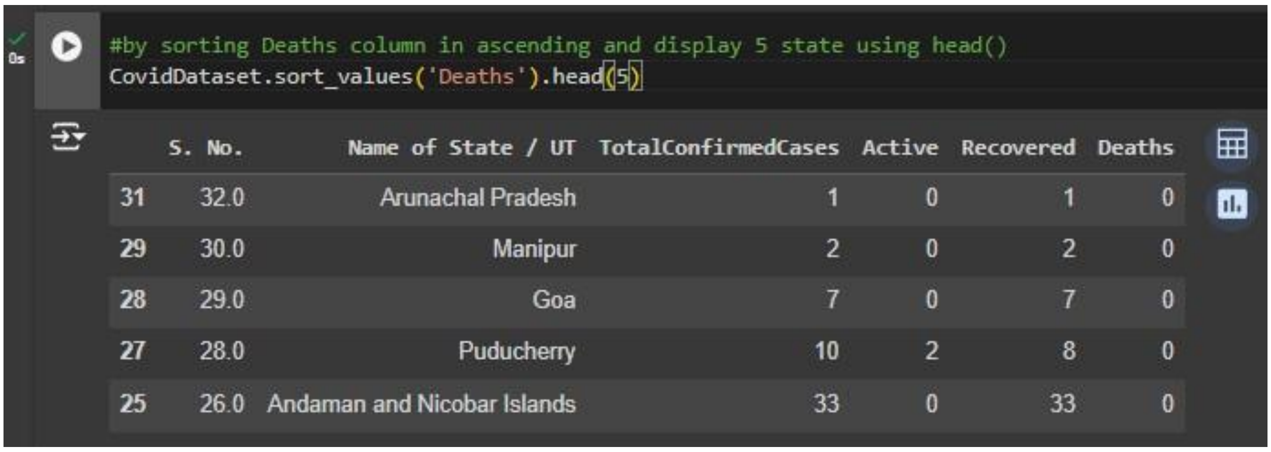


```
#usign nsmallest() function
CovidDataset.nsmallest(5,'Deaths')
```

	S. No.	Name of State / UT	TotalConfirmedCases	Active	Recovered	Deaths
19	20.0	Tripura	135	133	2	0
22	23.0	Chhattisgarh	59	16	43	0
24	25.0	Ladakh	42	24	18	0
25	26.0	Andaman and Nicobar Islands	33	0	33	0
27	28.0	Puducherry	10	2	8	0

#by sorting Deaths column in ascending and display 5 state using head()

CovidDataset.sort_values('Deaths').head(5)



```
#by sorting Deaths column in ascending and display 5 state using head()
CovidDataset.sort_values('Deaths').head(5)
```

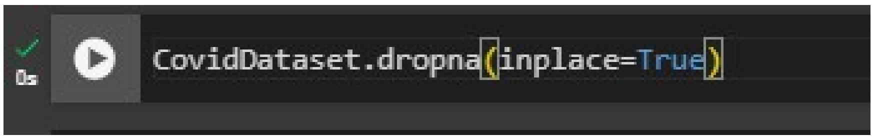
	S. No.	Name of State / UT	TotalConfirmedCases	Active	Recovered	Deaths
31	32.0	Arunachal Pradesh	1	0	1	0
29	30.0	Manipur	2	0	2	0
28	29.0	Goa	7	0	7	0
27	28.0	Puducherry	10	2	8	0
25	26.0	Andaman and Nicobar Islands	33	0	33	0

i. Drop rows with missing values

Code:

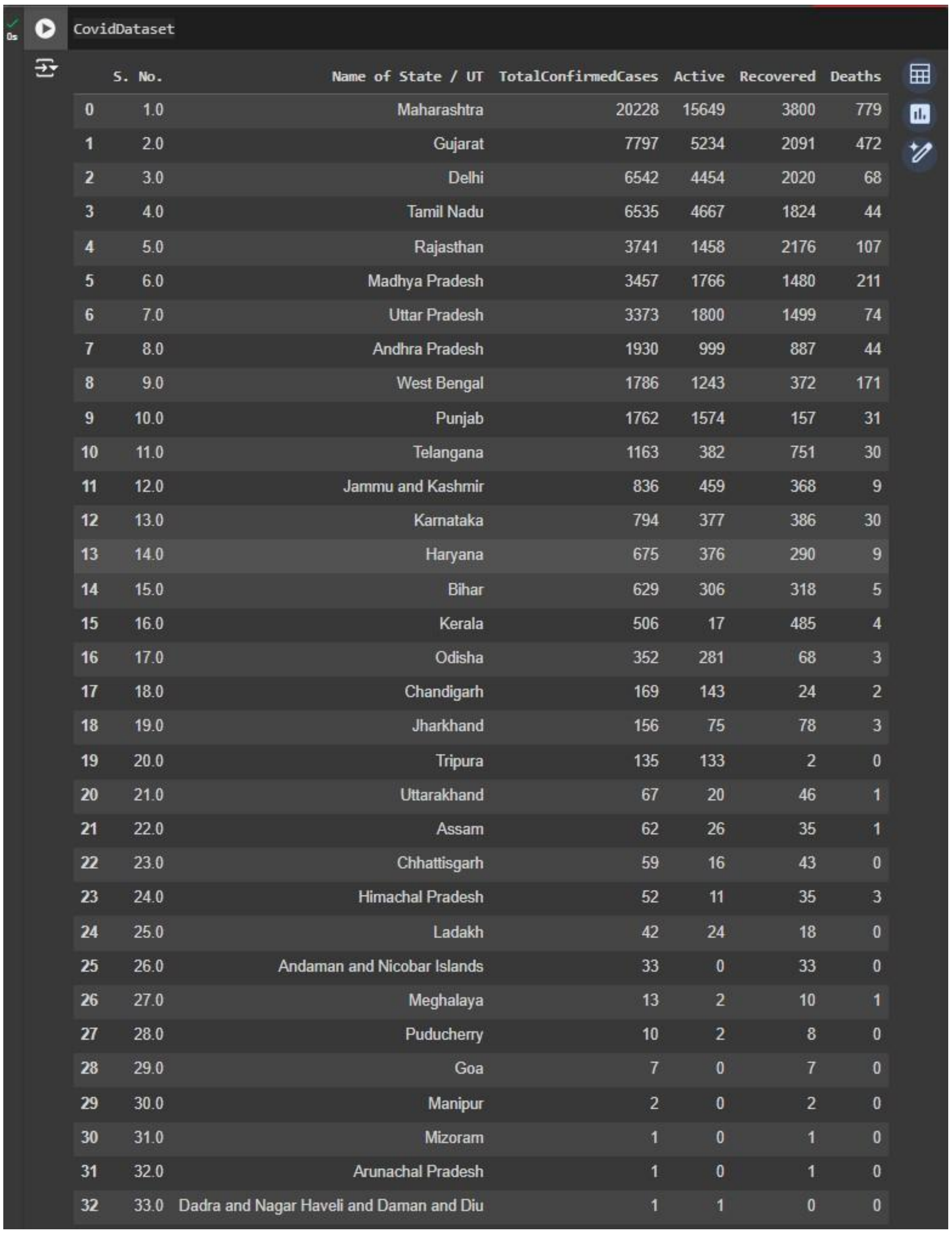
CovidDataset.dropna(inplace=True)

Output:



```
CovidDataset.dropna(inplace=True)
```

CovidDataset



```
CovidDataset
```

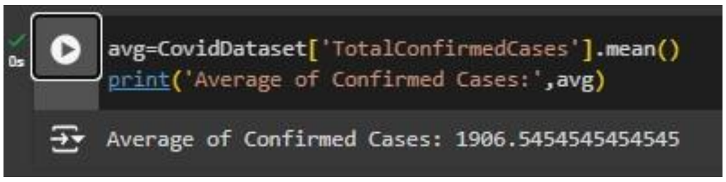
	S. No.	Name of State / UT	TotalConfirmedCases	Active	Recovered	Deaths
0	1.0	Maharashtra	20228	15649	3800	779
1	2.0	Gujarat	7797	5234	2091	472
2	3.0	Delhi	6542	4454	2020	68
3	4.0	Tamil Nadu	6535	4667	1824	44
4	5.0	Rajasthan	3741	1458	2176	107
5	6.0	Madhya Pradesh	3457	1766	1480	211
6	7.0	Uttar Pradesh	3373	1800	1499	74
7	8.0	Andhra Pradesh	1930	999	887	44
8	9.0	West Bengal	1786	1243	372	171
9	10.0	Punjab	1762	1574	157	31
10	11.0	Telangana	1163	382	751	30
11	12.0	Jammu and Kashmir	836	459	368	9
12	13.0	Karnataka	794	377	386	30
13	14.0	Haryana	675	376	290	9
14	15.0	Bihar	629	306	318	5
15	16.0	Kerala	506	17	485	4
16	17.0	Odisha	352	281	68	3
17	18.0	Chandigarh	169	143	24	2
18	19.0	Jharkhand	156	75	78	3
19	20.0	Tripura	135	133	2	0
20	21.0	Uttarakhand	67	20	46	1
21	22.0	Assam	62	26	35	1
22	23.0	Chhattisgarh	59	16	43	0
23	24.0	Himachal Pradesh	52	11	35	3
24	25.0	Ladakh	42	24	18	0
25	26.0	Andaman and Nicobar Islands	33	0	33	0
26	27.0	Meghalaya	13	2	10	1
27	28.0	Puducherry	10	2	8	0
28	29.0	Goa	7	0	7	0
29	30.0	Manipur	2	0	2	0
30	31.0	Mizoram	1	0	1	0
31	32.0	Arunachal Pradesh	1	0	1	0
32	33.0	Dadra and Nagar Haveli and Daman and Diu	1	1	0	0

j. Print Average number of Confirmed Cases

Code: avg=CovidDataset['TotalConfirmedCases'].mean()

print('Average of Confirmed Cases:',avg)

Output:



```
avg=CovidDataset['TotalConfirmedCases'].mean()
print('Average of Confirmed Cases:',avg)

Average of Confirmed Cases: 1906.5454545454545
```

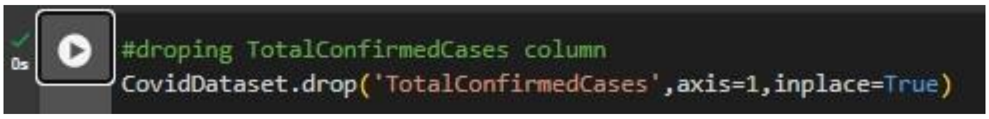
k. Create New Column Total Cases by adding active, recovered, and deaths

Code:

#droping TotalConfirmedCases column

CovidDataset.drop('TotalConfirmedCases',axis=1,inplace=True)

Output:

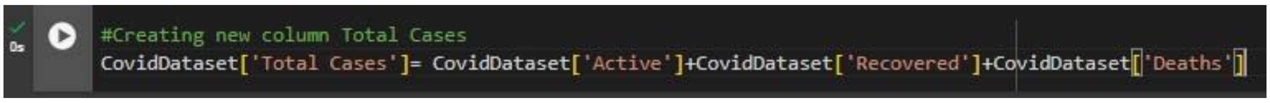


```
#droping TotalConfirmedCases column
CovidDataset.drop('TotalConfirmedCases',axis=1,inplace=True)
```

#Creating new column Total Cases

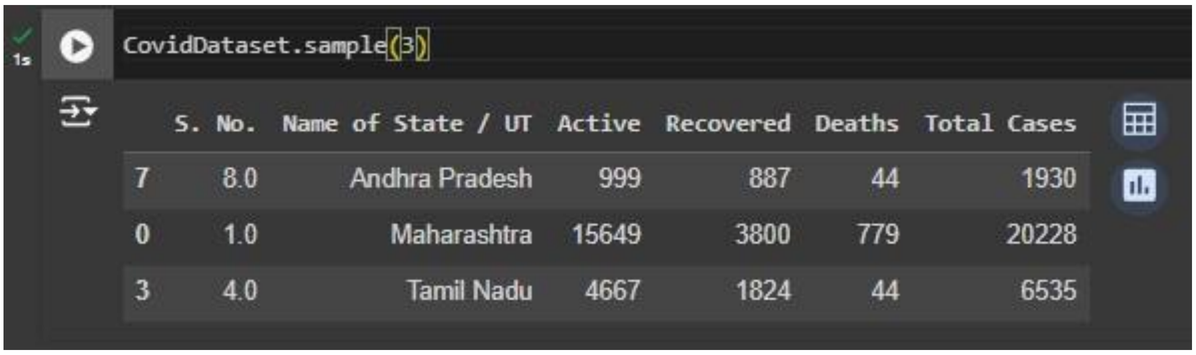
CovidDataset['Total Cases']=

CovidDataset['Active']+CovidDataset['Recovered']+CovidDataset['Deaths']



```
#Creating new column Total Cases
CovidDataset['Total Cases']= CovidDataset['Active']+CovidDataset['Recovered']+CovidDataset['Deaths']
```

CovidDataset.sample(3)



```
CovidDataset.sample(3)
```

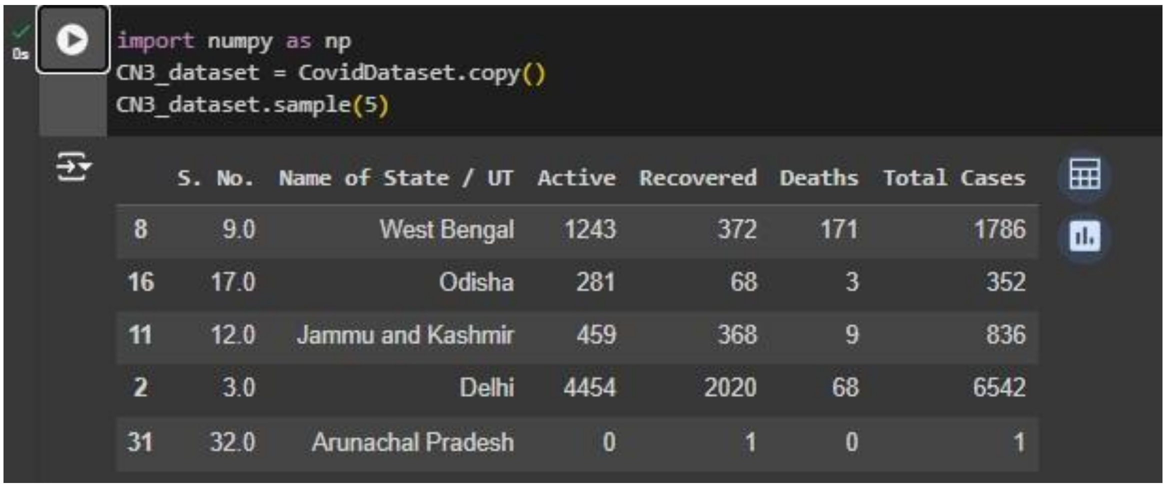
	S. No.	Name of State / UT	Active	Recovered	Deaths	Total Cases
7	8.0	Andhra Pradesh	999	887	44	1930
0	1.0	Maharashtra	15649	3800	779	20228
3	4.0	Tamil Nadu	4667	1824	44	6535

l. replace missing values with mean(), median(), and MOD

Code:

```
import numpy as np

CN3_dataset = CovidDataset.copy()

CN3_dataset.sample(5)
```

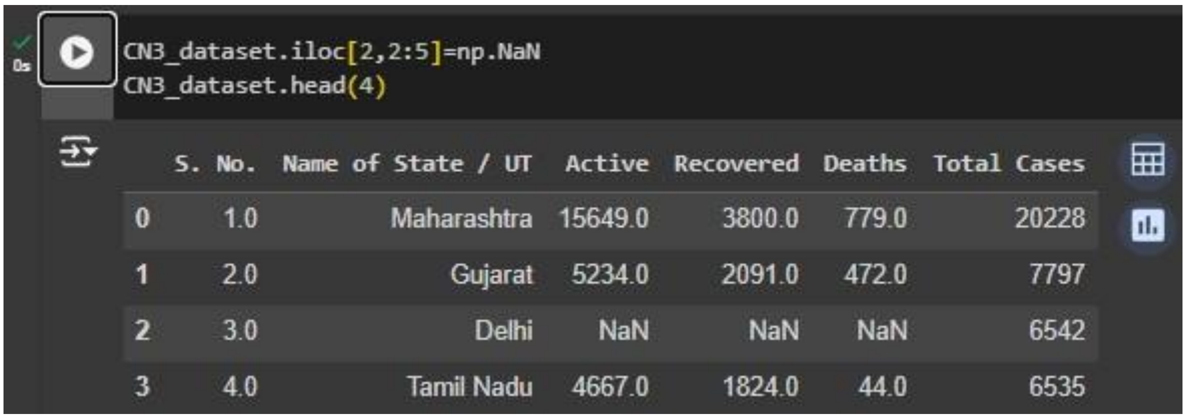
Output:



```
import numpy as np
CN3_dataset = CovidDataset.copy()
CN3_dataset.sample(5)
```

	S. No.	Name of State / UT	Active	Recovered	Deaths	Total Cases
8	9.0	West Bengal	1243	372	171	1786
16	17.0	Odisha	281	68	3	352
11	12.0	Jammu and Kashmir	459	368	9	836
2	3.0	Delhi	4454	2020	68	6542
31	32.0	Arunachal Pradesh	0	1	0	1

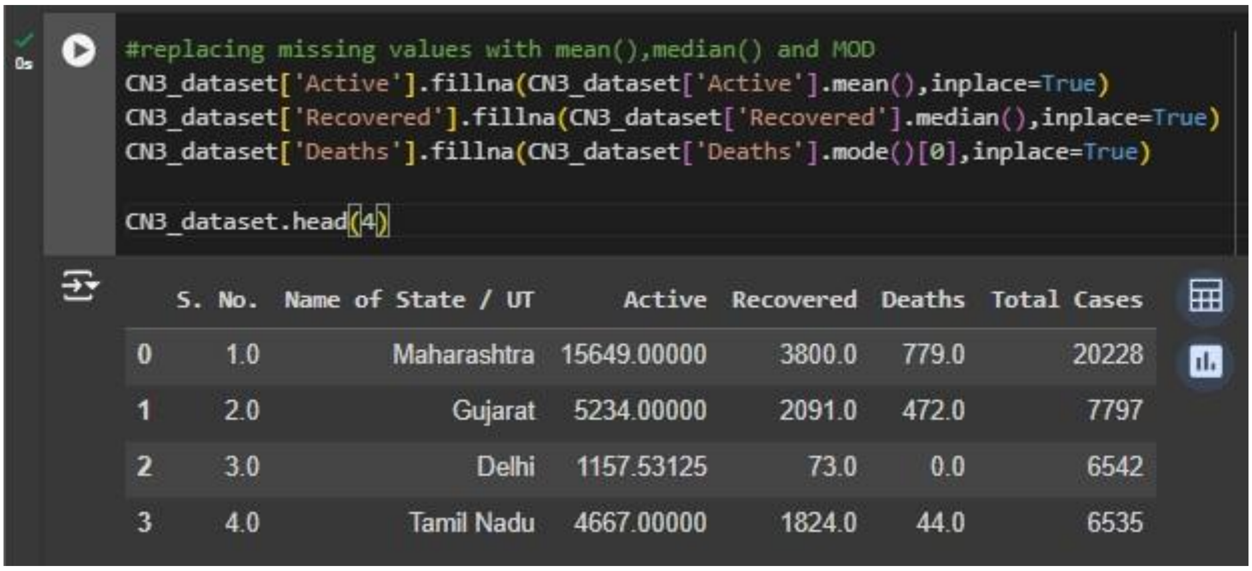
```
CN3_dataset.iloc[2,2:5]=np.NaN

CN3_dataset.head(4)
```



```
CN3_dataset.iloc[2,2:5]=np.NaN
CN3_dataset.head(4)
```

	S. No.	Name of State / UT	Active	Recovered	Deaths	Total Cases
0	1.0	Maharashtra	15649.0	3800.0	779.0	20228
1	2.0	Gujarat	5234.0	2091.0	472.0	7797
2	3.0	Delhi	NaN	NaN	NaN	6542
3	4.0	Tamil Nadu	4667.0	1824.0	44.0	6535

```
#replacing missing values with mean(),median() and MOD

CN3_dataset['Active'].fillna(CN3_dataset['Active'].mean(),inplace=True)

CN3_dataset['Recovered'].fillna(CN3_dataset['Recovered'].median(),inplace=True)

CN3_dataset['Deaths'].fillna(CN3_dataset['Deaths'].mode()[0],inplace=True)

CN3_dataset.head(4)
```



```
#replacing missing values with mean(),median() and MOD
CN3_dataset['Active'].fillna(CN3_dataset['Active'].mean(),inplace=True)
CN3_dataset['Recovered'].fillna(CN3_dataset['Recovered'].median(),inplace=True)
CN3_dataset['Deaths'].fillna(CN3_dataset['Deaths'].mode()[0],inplace=True)

CN3_dataset.head(4)
```

	S. No.	Name of State / UT	Active	Recovered	Deaths	Total Cases
0	1.0	Maharashtra	15649.00000	3800.0	779.0	20228
1	2.0	Gujarat	5234.00000	2091.0	472.0	7797
2	3.0	Delhi	1157.53125	73.0	0.0	6542
3	4.0	Tamil Nadu	4667.00000	1824.0	44.0	6535

m. Use matplotlib, seaborn library for Data Visualization

Code:

import matplotlib.pyplot as plt

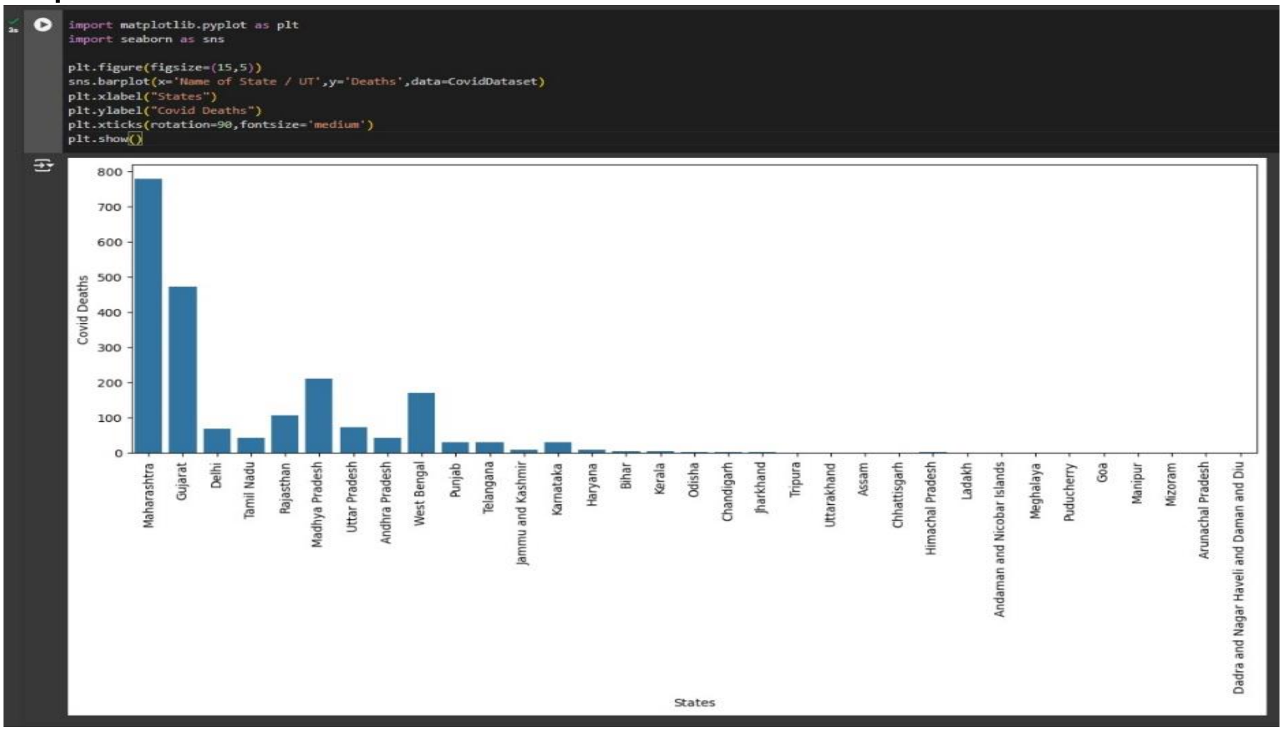
import seaborn as sns

plt.figure(figsize=(15,5))

sns.barplot(x='Name of State / UT',y='Deaths',data=CovidDataset)
plt.xlabel("States") plt.ylabel("Covid Deaths")

plt.xticks(rotation=90,fontsize='medium') plt.show()

Output:



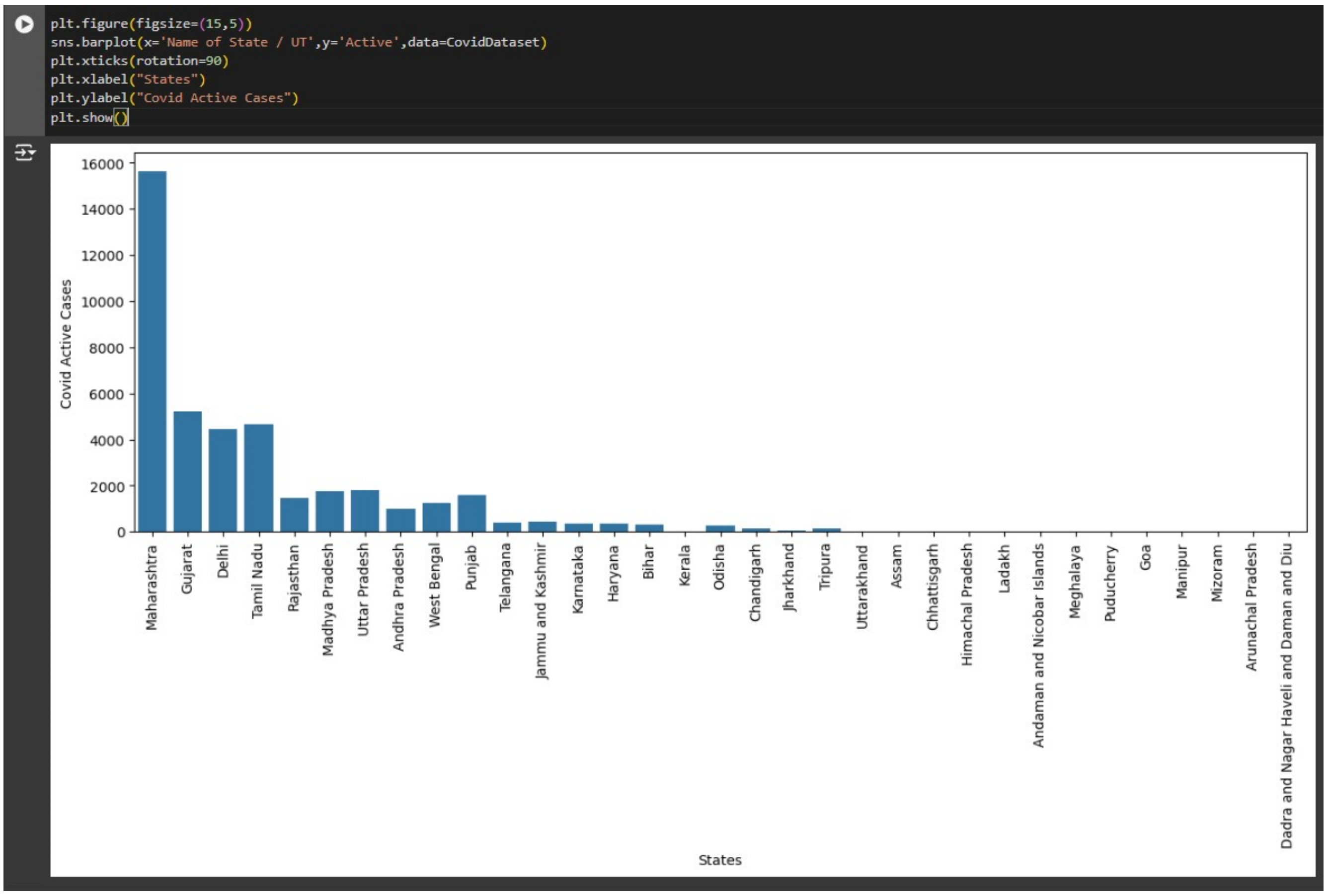
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(15,5))
sns.barplot(x='Name of State / UT',y='Deaths',data=CovidDataset)
plt.xlabel("States")
plt.ylabel("Covid Deaths")
plt.xticks(rotation=90,fontsize='medium')
plt.show()
```

n. Create graph with columns 'Name of State / UT' and y-axis 'Active'

Code:

```
plt.figure(figsize=(15,5))

sns.barplot(x='Name of State / UT',y='Active',data=CovidDataset)
plt.xticks(rotation=90) plt.xlabel("States")

plt.ylabel("Covid Active Cases") plt.show()
```

Output:



```
plt.figure(figsize=(15,5))
sns.barplot(x='Name of State / UT',y='Active',data=CovidDataset)
plt.xticks(rotation=90)
plt.xlabel("States")
plt.ylabel("Covid Active Cases")
plt.show()
```

Name : DHRUV SHERE
Enrollment No. : 23012022021
Batch : B-2