GANPAT UNIVERSITY

U. V. Patel College of Engineering

Data Science / Machine Learning

B.Tech Semester: VI

Computer Engineering/ Information Technology

Enrolment No: 19012012009 **Name: Priyank Bhavsar**

Aim:

1. Basic Understanding of Data Science and frequently useful libraries.

1. Pandas:-

Pandas is an open-source Python package that provides high-performance, easy-to-use data structures and data analysis tools for the labeled data in Python programming language. Pandas stand for Python Data Analysis Library.

2. NumPy:-

One of the most fundamental packages in Python, NumPy is a general-purpose array-processing package. It provides high-performance multidimensional array objects and tools to work with the arrays. NumPy is an efficient container of generic multi-dimensional data.

3. CSV:-

The csv module implements classes to read and write tabular data in CSV format. It allows programmers to say, “write this data in the format preferred by Excel,” or “read data from this file which was generated by Excel,” without knowing the precise details of the CSV format used by Excel.

4. Matplotlib

This is undoubtedly my favorite and a quintessential Python library. You can create stories with the data visualized with Matplotlib. Another library from the SciPy Stack, Matplotlib plots 2D figures.

5. Beautifulsoup4

So when you read the official documentation on Seaborn, it is defined as the data visualization library based on Matplotlib that provides a high-level interface for drawing attractive and informative statistical graphics. Putting it simply, seaborn is an extension of Matplotlib with advanced features.

2. Perform Basic data analysis and merge – sort operations over dataset zoo.csv.

- import pandas as pd
  dataset = pd.read_csv('zoo.csv') #read the data
  dataset #to print the data

dataset - DataFrame

Index	animal	uniq_id	vater_neec
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

- dataset.head() #to print first five records

```
In [4]: dataset.head()
Out[4]:
     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
```

- dataset.info()

```
In [5]: dataset.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22 entries, 0 to 21
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   animal      22 non-null     object
 1   uniq_id     22 non-null     int64
 2   water_need  22 non-null     int64
dtypes: int64(2), object(1)
memory usage: 656.0+ bytes
```

- dataset.describe()

```
In [6]: dataset.describe()
Out[6]:
            uniq_id  water_need
count    22.000000   22.000000
mean   1011.500000  347.727273
std       6.493587  147.549243
min    1001.000000   80.000000
25%    1006.250000  232.500000
50%    1011.500000  325.000000
75%    1016.750000  427.500000
max    1022.000000  600.000000
```

- dataset.sample(7) #to fetch any random data from the file

```
In [7]: print(dataset.sample(7))
      animal  uniq_id  water_need
4      tiger     1005         320
1   elephant     1002         600
7      tiger     1008         310
21  kangaroo     1022         410
10     zebra     1011         240
0   elephant     1001         500
12     zebra     1013         220
```

- **Creating a Dataframe**

```
dataset1 = pd.DataFrame([['elephant',"vegitables"],
                         ['tiger',"mear"],
                         ['cat',"milk"],
                         ['kangaroo',"vegitables"],
                         ['zebra',"vegitables"]],
                         columns=['animal','food'])
```

dataset1 - DataFrame

Index	animal	food
0	elephant	vegitables
1	tiger	mear
2	cat	milk
3	kangaroo	vegitables
4	zebra	vegitables

- Join Operations:-

  - dataset.merge(dataset1)

```
In [9]:
   ...: dataset.merge(dataset1)
Out[9]:
      animal  uniq_id  water_need        food
0   elephant     1001         500  vegitables
1   elephant     1002         600  vegitables
2   elephant     1003         550  vegitables
3      tiger     1004         300        mear
4      tiger     1005         320        mear
5      tiger     1006         330        mear
6      tiger     1007         290        mear
7      tiger     1008         310        mear
8      zebra     1009         200  vegitables
9      zebra     1010         220  vegitables
10     zebra     1011         240  vegitables
11     zebra     1012         230  vegitables
12     zebra     1013         220  vegitables
13     zebra     1014         100  vegitables
14     zebra     1015          80  vegitables
15  kangaroo     1020         410  vegitables
16  kangaroo     1021         430  vegitables
17  kangaroo     1022         410  vegitables
```

- dataset.merge(dataset1,how='outer')

```
In [10]: dataset.merge(dataset1,how='outer')
Out[10]:
      animal  uniq_id  water_need        food
0   elephant   1001.0       500.0  vegitables
1   elephant   1002.0       600.0  vegitables
2   elephant   1003.0       550.0  vegitables
3      tiger   1004.0       300.0        mear
4      tiger   1005.0       320.0        mear
5      tiger   1006.0       330.0        mear
6      tiger   1007.0       290.0        mear
7      tiger   1008.0       310.0        mear
8      zebra   1009.0       200.0  vegitables
9      zebra   1010.0       220.0  vegitables
10     zebra   1011.0       240.0  vegitables
11     zebra   1012.0       230.0  vegitables
12     zebra   1013.0       220.0  vegitables
13     zebra   1014.0       100.0  vegitables
14     zebra   1015.0        80.0  vegitables
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  vegitables
20  kangaroo   1021.0       430.0  vegitables
21  kangaroo   1022.0       410.0  vegitables
22       cat      NaN         NaN        milk
```

- dataset.merge(dataset1,how='inner')

```
In [11]: dataset.merge(dataset1,how='inner')
Out[11]:
      animal  uniq_id  water_need        food
0   elephant     1001         500  vegitables
1   elephant     1002         600  vegitables
2   elephant     1003         550  vegitables
3      tiger     1004         300        mear
4      tiger     1005         320        mear
5      tiger     1006         330        mear
6      tiger     1007         290        mear
7      tiger     1008         310        mear
8      zebra     1009         200  vegitables
9      zebra     1010         220  vegitables
10     zebra     1011         240  vegitables
11     zebra     1012         230  vegitables
12     zebra     1013         220  vegitables
13     zebra     1014         100  vegitables
14     zebra     1015          80  vegitables
15  kangaroo     1020         410  vegitables
16  kangaroo     1021         430  vegitables
17  kangaroo     1022         410  vegitables
```

- dataset.merge(dataset1, how='outer').fillna('MISSING')

```
In [12]: dataset.merge(dataset1, how='outer').fillna('empty')
Out[12]:
      animal uniq_id water_need        food
0   elephant    1001        500  vegitables
1   elephant    1002        600  vegitables
2   elephant    1003        550  vegitables
3      tiger    1004        300        mear
4      tiger    1005        320        mear
5      tiger    1006        330        mear
6      tiger    1007        290        mear
7      tiger    1008        310        mear
8      zebra    1009        200  vegitables
9      zebra    1010        220  vegitables
10     zebra    1011        240  vegitables
11     zebra    1012        230  vegitables
12     zebra    1013        220  vegitables
13     zebra    1014        100  vegitables
14     zebra    1015         80  vegitables
15      lion    1016        420       empty
16      lion    1017        600       empty
17      lion    1018        500       empty
18      lion    1019        390       empty
19  kangaroo    1020        410  vegitables
20  kangaroo    1021        430  vegitables
21  kangaroo    1022        410  vegitables
22       cat   empty      empty        milk
```

- **Sorting Data**
  - dataset1.sort_values('food')

```
In [13]: dataset1.sort_values('food')
Out[13]:
     animal        food
1     tiger        mear
2       cat        milk
0  elephant  vegitables
3  kangaroo  vegitables
4     zebra  vegitables
```

  - dataset=dataset.sort_values('water_need',ascending=False)

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

  - dataset.sort_values('water_need',ascending=False,inplace=True)
    #inplsace is used to save the changes

```
Out[19]:
      animal  uniq_id  water_need
1   elephant     1002         600
16      lion     1017         600
2   elephant     1003         550
0   elephant     1001         500
17      lion     1018         500
20  kangaroo     1021         430
15      lion     1016         420
19  kangaroo     1020         410
21  kangaroo     1022         410
18      lion     1019         390
5      tiger     1006         330
4      tiger     1005         320
```

```
7      tiger     1008         310
3      tiger     1004         300
6      tiger     1007         290
10     zebra     1011         240
11     zebra     1012         230
9      zebra     1010         220
12     zebra     1013         220
8      zebra     1009         200
13     zebra     1014         100
14     zebra     1015          80
```

- **Missing Values:-**

  - dataset2=dataset.merge(dataset1,how='left').fillna('Empty')

```
In [21]: dataset2
Out[21]:
      animal  uniq_id  water_need        food
0   elephant     1002         600  vegitables
1       lion     1017         600       Empty
2   elephant     1003         550  vegitables
3   elephant     1001         500  vegitables
4       lion     1018         500       Empty
5   kangaroo     1021         430  vegitables
6       lion     1016         420       Empty
7   kangaroo     1020         410  vegitables
8   kangaroo     1022         410  vegitables
9       lion     1019         390       Empty
10     tiger     1006         330        mear
11     tiger     1005         320        mear
12     tiger     1008         310        mear
13     tiger     1004         300        mear
14     tiger     1007         290        mear
15     zebra     1011         240  vegitables
16     zebra     1012         230  vegitables
17     zebra     1010         220  vegitables
18     zebra     1013         220  vegitables
19     zebra     1009         200  vegitables
20     zebra     1014         100  vegitables
21     zebra     1015          80  vegitables
```

- **Inplace Parameter:-**
  - dataset.water_need

```
In [22]: dataset.water_need
Out[22]:
1     600
16    600
2     550
0     500
17    500
20    430
15    420
19    410
21    410
18    390
5     330
```

```
4     320
7     310
3     300
6     290
10    240
11    230
9     220
12    220
8     200
13    100
14     80
Name: water_need, dtype: int64
```

- dataset1[['animal','food']]

```
In [26]: dataset1[['animal','food']]
Out[26]:
     animal        food
0  elephant  vegitables
1     tiger        mear
2       cat        milk
3  kangaroo  vegitables
4     zebra  vegitables
```

- **Aggregation:-**

  - dataset[['animal']].count()

```
In [27]: dataset[['animal']].count()
Out[27]:
animal    22
dtype: int64
```

  - dataset[['animal','water_need']].max()

```
In [28]: dataset[['animal','water_need']].max()
Out[28]:
animal        zebra
water_need      600
dtype: object
```

  - dataset[['animal','water_need']].min()

```
In [29]: dataset[['animal','water_need']].min()
Out[29]:
animal        elephant
water_need          80
dtype: object
```

  - dataset[['animal','water_need']].median()

```
In [32]: dataset[['animal','water_need']].median()
Out[32]:
water_need    325.0
dtype: float64
```

- dataset[['water_need']].sum()

```
In [30]: dataset[['water_need']].sum()
Out[30]:
water_need    7650
dtype: int64
```

- dataset[['animal','water_need']].mode()

```
In [33]: dataset[['animal','water_need']].mode()
Out[33]:
  animal  water_need
0  zebra         220
1    NaN         410
2    NaN         500
3    NaN         600
```

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

- import pandas as pd
  df = pd.read_excel('Covid.xlsx')
  df

```
    S. No.                        Name of State / UT  ...  Recovered  Deaths
0      1.0                                Maharashtra  ...       3800     779
1      2.0                                    Gujarat  ...       2091     472
2      3.0                                      Delhi  ...       2020      68
3      4.0                                  Tamil Nadu  ...       1824      44
4      5.0                                  Rajasthan  ...       2176     107
5      6.0                             Madhya Pradesh  ...       1480     211
6      7.0                              Uttar Pradesh  ...       1499      74
7      8.0                             Andhra Pradesh  ...        887      44
8      9.0                                West Bengal  ...        372     171
9     10.0                                     Punjab  ...        157      31
10    11.0                                  Telangana  ...        751      30
11    12.0                            Jammu and Kashmir  ...        368       9
12    13.0                                  Karnataka  ...        386      30
13    14.0                                    Haryana  ...        290       9
14    15.0                                      Bihar  ...        318       5
15    16.0                                     Kerala  ...        485       4
16    17.0                                     Odisha  ...         68       3
17    18.0                                  Chandigarh  ...         24       2
18    19.0                                   Jharkhand  ...         78       3
19    20.0                                     Tripura  ...          2       0
20    21.0                                Uttarakhand  ...         46       1
21    22.0                                      Assam  ...         35       1
22    23.0                                Chhattisgarh  ...         43       0
23    24.0                             Himachal Pradesh  ...         35       3
24    25.0                                      Ladakh  ...         18       0
25    26.0                 Andaman and Nicobar Islands  ...         33       0
26    27.0                                   Meghalaya  ...         10       1
27    28.0                                  Puducherry  ...          8       0
28    29.0                                         Goa  ...          7       0
29    30.0                                     Manipur  ...          2       0
30    31.0                                     Mizoram  ...          1       0
31    32.0                             Arunachal Pradesh  ...          1       0
32    33.0  Dadra and Nagar Haveli and Daman and Diu  ...          0       0
33     NaN                                       Total  ...      19315    2102
```

- df.columns[1:3]

```
In [36]: df.columns[1:3]
Out[36]: Index(['Name of State / UT', 'TotalConfirmedCases'], dtype='object')
```

- columns = df.columns
  columns

```
Out[37]:
Index(['S. No.', 'Name of State / UT', 'TotalConfirmedCases', 'Active',
       'Recovered', 'Deaths'],
      dtype='object')
```

- df['Active']>500

```
In [38]: df['Active']>500
Out[38]:
0      True
1      True
2      True
3      True
4      True
5      True
6      True
7      True
8      True
9      True
10    False
11    False
12    False
13    False
14    False
15    False
16    False
17    False
18    False
19    False
20    False
21    False
22    False
23    False
24    False
25    False
26    False
27    False
28    False
29    False
30    False
31    False
32    False
```

- df.loc[df['Active']>500]

```
In [39]: df.loc[df['Active']>500]
Out[39]:
    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
33     NaN              Total                62916   41495      19315    2102
```

- new=df.loc[df['Active']>500,['Name of State / UT','Active','Deaths']]

```
In [41]: new
Out[41]:
   Name of State / UT  Active  Deaths
0         Maharashtra   15649     779
1             Gujarat    5234     472
2               Delhi    4454      68
3          Tamil Nadu    4667      44
4           Rajasthan    1458     107
5      Madhya Pradesh    1766     211
6       Uttar Pradesh    1800      74
7      Andhra Pradesh     999      44
8         West Bengal    1243     171
9              Punjab    1574      31
33              Total   41495    2102
```

- new.sort_values('Active',ascending=False,inplace=True)

```
In [43]: new
Out[43]:
   Name of State / UT  Active  Deaths
33              Total   41495    2102
0         Maharashtra   15649     779
1             Gujarat    5234     472
3          Tamil Nadu    4667      44
2               Delhi    4454      68
6       Uttar Pradesh    1800      74
5      Madhya Pradesh    1766     211
9              Punjab    1574      31
4           Rajasthan    1458     107
8         West Bengal    1243     171
7      Andhra Pradesh     999      44
```

- **Highest Value:-**

  - df.sort_values('Active',ascending=False,inplace=True)
    df.head()
    df.nlargest(5,'Deaths')

```
Out[44]:
    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
```

- **Lowest Value:-**

  - df.sort_values('Active',inplace=True)
    df.head()
    df.nsmallest(5,'Deaths')

```
Out[45]:
    S. No.           Name of State / UT  ...  Recovered  Deaths
31    32.0            Arunachal Pradesh  ...          1       0
25    26.0  Andaman and Nicobar Islands  ...         33       0
30    31.0                     Mizoram  ...          1       0
29    30.0                     Manipur  ...          2       0
28    29.0                         Goa  ...          7       0
```

- df.iloc[(df['Active']>500).values & (df['Deaths']>500).values,[1,3,5]]

```
Out[46]:
   Name of State / UT  Active  Deaths
0         Maharashtra   15649     779
33              Total   41495    2102
```

- df.loc[(df['Deaths']>500) & (df['Active']>500),['Name of State /UT','Active','Deaths']]

```
Out[47]:
   Name of State / UT  Active  Deaths
0         Maharashtra   15649     779
33              Total   41495    2102
```

- df.sample(frac=0.4)

Random data

```
Out[48]:
    S. No. Name of State / UT  TotalConfirmedCases  Active  Recovered  Deaths
23    24.0   Himachal Pradesh                   52      11         35       3
17    18.0         Chandigarh                  169     143         24       2
9     10.0             Punjab                 1762    1574        157      31
15    16.0             Kerala                  506      17        485       4
19    20.0            Tripura                  135     133          2       0
24    25.0             Ladakh                   42      24         18       0
20    21.0        Uttarakhand                   67      20         46       1
30    31.0            Mizoram                    1       0          1       0
7      8.0     Andhra Pradesh                 1930     999        887      44
29    30.0            Manipur                    2       0          2       0
27    28.0         Puducherry                   10       2          8       0
31    32.0  Arunachal Pradesh                    1       0          1       0
26    27.0          Meghalaya                   13       2         10       1
14    15.0              Bihar                  629     306        318       5
```

- df.info()

```
In [52]: df.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
```

- df.shape

```
In [53]: df.shape
Out[53]: (34, 6)
```

- df.describe()

```
In [54]: df.describe()
Out[54]:
         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
```

- print('total confirmed cases = ',df['Active'].sum())

```
In [55]: print('total confirmed cases = ',df['Active'].sum())
total confirmed cases =  82990
```

- print('total deaths = ',df['Deaths'].sum())

```
In [56]: print('total deaths = ',df['Deaths'].sum())
total deaths =  4204
```

- df['TotalCases']=df['Active']+df['Recovered']+df['Deaths']

#add Column

```
Out[60]:
    S. No.                      Name of State / UT  Active  Recovered  Deaths
0      1.0                             Maharashtra   15649       3800     779
1      2.0                                 Gujarat    5234       2091     472
2      3.0                                   Delhi    4454       2020      68
3      4.0                               Tamil Nadu    4667       1824      44
4      5.0                                Rajasthan    1458       2176     107
5      6.0                           Madhya Pradesh    1766       1480     211
6      7.0                            Uttar Pradesh    1800       1499      74
7      8.0                           Andhra Pradesh     999        887      44
8      9.0                              West Bengal    1243        372     171
9     10.0                                   Punjab    1574        157      31
10    11.0                                Telangana     382        751      30
11    12.0                        Jammu and Kashmir     459        368       9
12    13.0                                Karnataka     377        386      30
13    14.0                                  Haryana     376        290       9
14    15.0                                    Bihar     306        318       5
15    16.0                                   Kerala      17        485       4
16    17.0                                   Odisha     281         68       3
17    18.0                                Chandigarh     143         24       2
18    19.0                                 Jharkhand      75         78       3
19    20.0                                   Tripura     133          2       0
20    21.0                                Uttarakhand      20         46       1
21    22.0                                      Assam      26         35       1
22    23.0                                Chhattisgarh      16         43       0
23    24.0                           Himachal Pradesh      11         35       3
24    25.0                                     Ladakh      24         18       0
25    26.0                Andaman and Nicobar Islands       0         33       0
26    27.0                                  Meghalaya       2         10       1
27    28.0                                 Puducherry       2          8       0
28    29.0                                        Goa       0          7       0
29    30.0                                     Manipur       0          2       0
30    31.0                                     Mizoram       0          1       0
```

- df.drop(labels='TotalCases',axis = 1,inplace=True)

#Delete Column

Out[64]:

	S. No.	Name of State / UT	Active	Recovered	Deaths
0	1.0	Maharashtra	15649	3800	779
1	2.0	Gujarat	5234	2091	472
2	3.0	Delhi	4454	2020	68
3	4.0	Tamil Nadu	4667	1824	44
4	5.0	Rajasthan	1458	2176	107
5	6.0	Madhya Pradesh	1766	1480	211
6	7.0	Uttar Pradesh	1800	1499	74
7	8.0	Andhra Pradesh	999	887	44
8	9.0	West Bengal	1243	372	171
9	10.0	Punjab	1574	157	31
10	11.0	Telangana	382	751	30
11	12.0	Jammu and Kashmir	459	368	9
12	13.0	Karnataka	377	386	30
13	14.0	Haryana	376	290	9
14	15.0	Bihar	306	318	5
15	16.0	Kerala	17	485	4
16	17.0	Odisha	281	68	3
17	18.0	Chandigarh	143	24	2
18	19.0	Jharkhand	75	78	3
19	20.0	Tripura	133	2	0
20	21.0	Uttarakhand	20	46	1
21	22.0	Assam	26	35	1
22	23.0	Chhattisgarh	16	43	0
23	24.0	Himachal Pradesh	11	35	3
24	25.0	Ladakh	24	18	0
25	26.0	Andaman and Nicobar Islands	0	33	0
26	27.0	Meghalaya	2	10	1
27	28.0	Puducherry	2	8	0

- **Copying data to new variable and placing nan to Location 1,2.**

  - import numpy as np
  - new_df = df[:]
  - new_df.iloc[1,2] = np.NaN

new_df - DataFrame

Index	S. No.	Name of State / UT	Active	Recoverec	Deaths
0	1	Maharashtra	15649	3800	779
1	2	Gujarat	nan	2091	472
2	3	Delhi	4454	2020	68
3	4	Tamil Nadu	4667	1824	44
4	5	Rajasthan	1458	2176	107
5	6	Madhya Pradesh	1766	1480	211
6	7	Uttar Pradesh	1800	1499	74
7	8	Andhra Pradesh	999	887	44
8	9	West Bengal	1243	372	171
9	10	Punjab	1574	157	31

Format Resize Background color Column min/max Save and Close Close

- new_df.info()

```
In [69]: new_df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34 entries, 0 to 33
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   S. No.              33 non-null     float64
 1   Name of State / UT  34 non-null     object
 2   Active              33 non-null     float64
 3   Recovered           34 non-null     int64
 4   Deaths              34 non-null     int64
dtypes: float64(2), int64(2), object(1)
memory usage: 1.5+ KB
```

- new_df = new_df.dropna()

  # Drop row when NaN are there

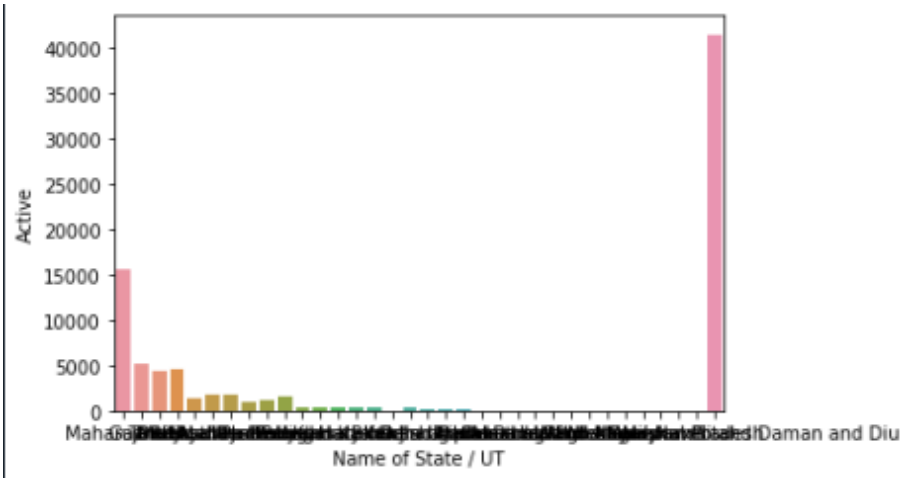
```
In [72]: new_df.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 32 entries, 0 to 32
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   S. No.              32 non-null     float64
 1   Name of State / UT  32 non-null     object
 2   Active              32 non-null     float64
 3   Recovered           32 non-null     int64
 4   Deaths              32 non-null     int64
dtypes: float64(2), int64(2), object(1)
memory usage: 1.5+ KB
```

- **Matplotlib & seaborn –**

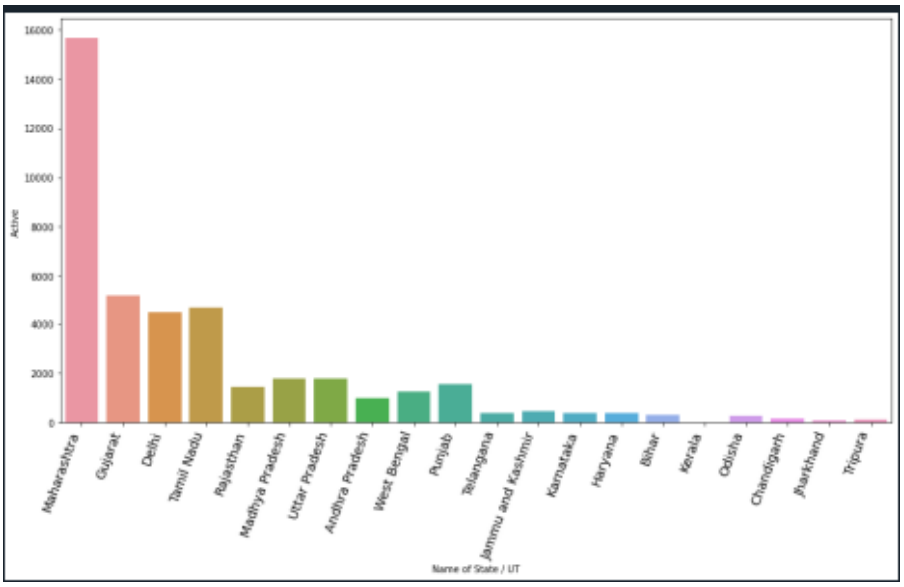
  - import matplotlib.pyplot as plt
    import seaborn as sns
    df.reset_index(drop=True, inplace=True)
    df

```
Out[73]:
    S. No.                      Name of State / UT  Active  Recovered  Deaths
0      1.0                             Maharashtra   15649       3800     779
1      2.0                                 Gujarat    5234       2091     472
2      3.0                                   Delhi    4454       2020      68
3      4.0                               Tamil Nadu    4667       1824      44
4      5.0                                Rajasthan    1458       2176     107
5      6.0                           Madhya Pradesh    1766       1480     211
6      7.0                            Uttar Pradesh    1800       1499      74
7      8.0                           Andhra Pradesh     999        887      44
8      9.0                              West Bengal    1243        372     171
9     10.0                                   Punjab    1574        157      31
10    11.0                                Telangana     382        751      30
11    12.0                          Jammu and Kashmir     459        368       9
12    13.0                                Karnataka     377        386      30
13    14.0                                  Haryana     376        290       9
14    15.0                                    Bihar     306        318       5
15    16.0                                   Kerala      17        485       4
```

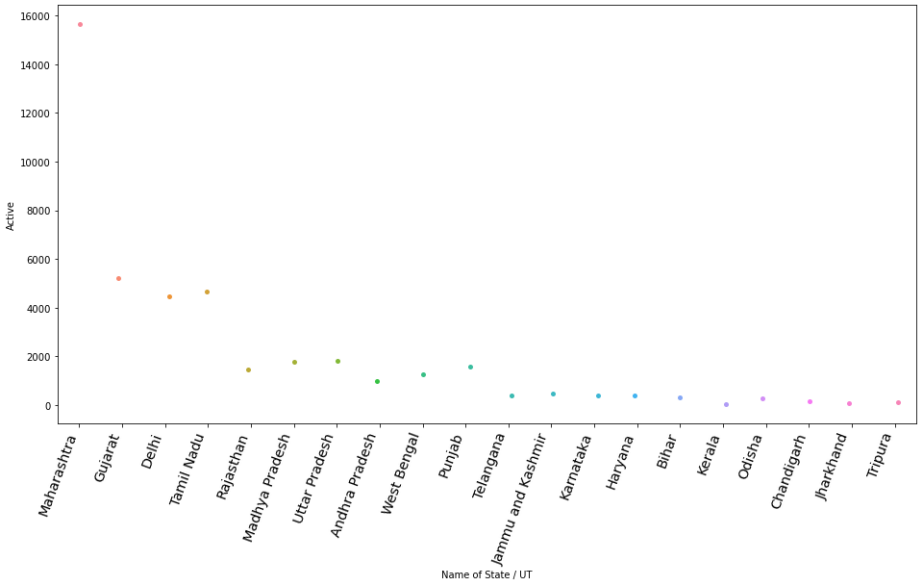
- sns.barplot(x="Name of State / UT", y="Active", data=df)
  plt.xticks(rotation=70, horizontalalignment='right', fontweight='light', fontsize='small')
  df = df[:20]
  df



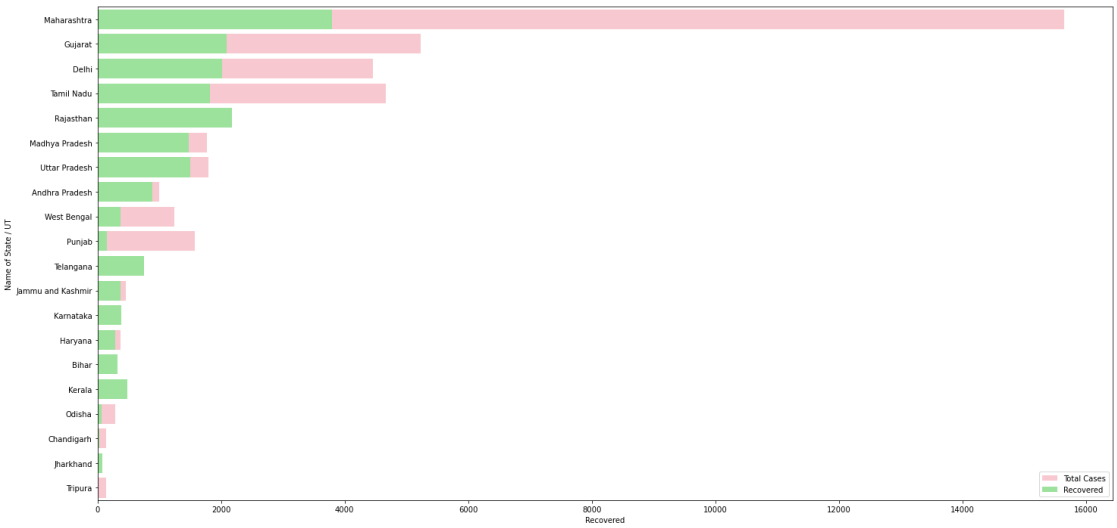
- plt.figure(figsize=(16,8))
  sns.barplot(x="Name of State / UT", y="TotalCases", data=df)
  plt.xticks(rotation=70, horizontalalignment='right', fontweight='light', fontsize='x-large')
  plt.show()



- df.columns
  plt.figure(figsize=(16,8))
  sns.stripplot(x="Name of State / UT", y="Active", data=df)
  plt.xticks(rotation=70, horizontalalignment='right', fontweight='light', fontsize='x-large')
  plt.show()



- plt.figure(figsize=(24,12))
  sns.barplot(x=df["TotalCases"], y=df["Name of State / UT"], color="pink", label="Total Cases")
  sns.barplot(x=df["Recovered"], y=df["Name of State / UT"], color="lightgreen", label="Recovered")
  plt.legend()



Exercise with pokemon dataset

- import pandas as pd
  df = pd.read_excel('pokemon_data.xlsx')
  df

```
In [3]: import pandas as pd
   ...: df = pd.read_excel('pokemon_data.xlsx')
   ...: df
Out[3]:
       #                   Name   Type 1  ... Speed  Generation  Legendary
0      1              Bulbasaur    Grass  ...    45           1      False
1      2                Ivysaur    Grass  ...    60           1      False
2      3               Venusaur    Grass  ...    80           1      False
3      3  VenusaurMega Venusaur    Grass  ...    80           1      False
4      4             Charmander     Fire  ...    65           1      False
..   ...                    ...      ...  ...   ...         ...        ...
795  719                Diancie     Rock  ...    50           6       True
796  719    DiancieMega Diancie     Rock  ...   110           6       True
797  720    HoopaHoopa Confined  Psychic  ...    70           6       True
798  720     HoopaHoopa Unbound  Psychic  ...    80           6       True
799  721              Volcanion     Fire  ...    70           6       True
```

- df.describe()

```
In [4]: df.describe()
Out[4]:
                #          HP      Attack  ...     Sp. Def       Speed  Generation
count  800.000000  800.000000  800.000000  ...  800.000000  800.000000   800.00000
mean   362.813750   69.258750   79.001250  ...   71.902500   68.277500     3.32375
std    208.343798   25.534669   32.457366  ...   27.828916   29.060474     1.66129
min      1.000000    1.000000    5.000000  ...   20.000000    5.000000     1.00000
25%    184.750000   50.000000   55.000000  ...   50.000000   45.000000     2.00000
50%    364.500000   65.000000   75.000000  ...   70.000000   65.000000     3.00000
75%    539.250000   80.000000  100.000000  ...   90.000000   90.000000     5.00000
max    721.000000  255.000000  190.000000  ...  230.000000  180.000000     6.00000
```

- df.info()

```
In [6]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 800 entries, 0 to 799
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   #           800 non-null    int64
 1   Name        800 non-null    object
 2   Type 1      800 non-null    object
 3   Type 2      414 non-null    object
 4   HP          800 non-null    int64
 5   Attack      800 non-null    int64
 6   Defense     800 non-null    int64
 7   Sp. Atk     800 non-null    int64
 8   Sp. Def     800 non-null    int64
 9   Speed       800 non-null    int64
 10  Generation  800 non-null    int64
 11  Legendary   800 non-null    bool
dtypes: bool(1), int64(8), object(3)
memory usage: 69.7+ KB
```

- df.shape

```
In [5]: df.shape
Out[5]: (800, 12)
```

- df.sort_values('HP',ascending=False)
-

```
In [7]: df.sort_values('HP',ascending=False)
Out[7]:
       #       Name   Type 1 Type 2  ...  Sp. Def  Speed  Generation  Legendary
261  242    Blissey   Normal    NaN  ...      135     55           2      False
121  113    Chansey   Normal    NaN  ...      105     50           1      False
217  202  Wobbuffet  Psychic    NaN  ...       58     33           2      False
351  321    Wailord    Water    NaN  ...       45     60           3      False
655  594  Alomomola    Water    NaN  ...       45     65           5      False
..   ...        ...      ...    ...  ...      ...    ...         ...        ...
139  129   Magikarp    Water    NaN  ...       20     80           1      False
381  349     Feebas    Water    NaN  ...       55     80           3      False
388  355    Duskull    Ghost    NaN  ...       90     25           3      False
55    50    Diglett   Ground    NaN  ...       45     95           1      False
316  292   Shedinja      Bug  Ghost  ...       30     40           3      False
```

- df.sort_values('Speed',ascending=False)

```
In [11]: df.sort_values('Speed',ascending=False)
Out[11]:
       #                       Name   Type 1  ... Speed  Generation  Legendary
431  386          DeoxysSpeed Forme  Psychic  ...   180           3       True
315  291                    Ninjask      Bug  ...   160           3      False
428  386         DeoxysNormal Forme  Psychic  ...   150           3       True
154  142  AerodactylMega Aerodactyl     Rock  ...   150           1      False
71    65      AlakazamMega Alakazam  Psychic  ...   150           1      False
..   ...                        ...      ...  ...   ...         ...        ...
658  597                   Ferroseed    Grass  ...    10           5      False
486  438                      Bonsly     Rock  ...    10           4      False
359  328                    Trapinch   Ground  ...    10           3      False
230  213                     Shuckle      Bug  ...     5           2      False
495  446                    Munchlax   Normal  ...     5           4      False
```

- df.Name

```
In [12]: df.Name
Out[12]:
0                  Bulbasaur
1                    Ivysaur
2                   Venusaur
3      VenusaurMega Venusaur
4                 Charmander
               ...
795                   Diancie
796      DiancieMega Diancie
797      HoopaHoopa Confined
798       HoopaHoopa Unbound
799                 Volcanion
Name: Name, Length: 800, dtype: object
```

- df.columns[1:3]
  new_df= df[df.columns[1:3]]
  new_df.head()

```
Out[13]:
                    Name Type 1
0              Bulbasaur  Grass
1                Ivysaur  Grass
2               Venusaur  Grass
3  VenusaurMega Venusaur  Grass
4             Charmander   Fire
```

- df[['Speed']].count()

```
In [14]: df[['Speed']].count()
Out[14]:
Speed    800
dtype: int64
```

- df[['Speed']].sum()

```
In [15]: df[['Speed']].sum()
Out[15]:
Speed    54622
dtype: int64
```

- df.loc[df['Speed']<50]

```
In [21]: df.loc[df['Speed']<50]
Out[21]:
       #                 Name Type 1  ... Speed  Generation  Legendary
0      1            Bulbasaur  Grass  ...    45           1      False
9      7             Squirtle  Water  ...    43           1      False
13    10             Caterpie    Bug  ...    45           1      False
14    11              Metapod    Bug  ...    30           1      False
17    14               Kakuna    Bug  ...    35           1      False
..   ...                  ...    ...  ...   ...         ...        ...
778  708             Phantump  Ghost  ...    38           6      False
782  710  PumpkabooLarge Size  Ghost  ...    46           6      False
783  710  PumpkabooSuper Size  Ghost  ...    41           6      False
788  712             Bergmite    Ice  ...    28           6      False
789  713              Avalugg    Ice  ...    28           6      False
```

- ghost = df.loc[df['Type 1'] == 'Ghost']
  ghost_HP = ghost.sort_values('HP',ascending=False)
  ghost_HP.head()

```
In [8]: ghost = df.loc[df['Type 1'] == 'Ghost']
   ...: ghost_HP = ghost.sort_values('HP',ascending=False)
   ...: ghost_HP.head()
Out[8]:
       #                   Name Type 1  ... Speed  Generation  Legendary
545  487   GiratinaOrigin Forme  Ghost  ...    90           4       True
544  487  GiratinaAltered Forme  Ghost  ...    90           4       True
473  426               Drifblim  Ghost  ...    80           4      False
472  425               Drifloon  Ghost  ...    70           4      False
779  709              Trevenant  Ghost  ...    56           6      False

[5 rows x 12 columns]
```

- df[['Speed']].max()

```
In [23]: df[['Speed']].max()
Out[23]:
Speed    180
dtype: int64
```

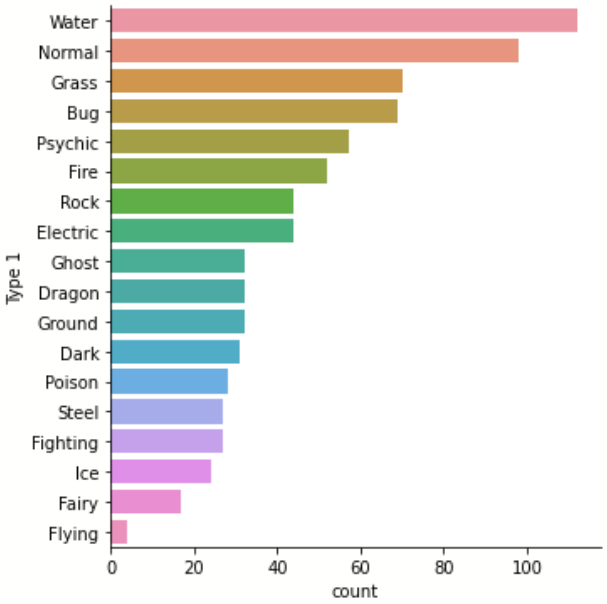
- df[['Speed']].min()

```
In [24]: df[['Speed']].min()
Out[24]:
Speed    5
dtype: int64
```

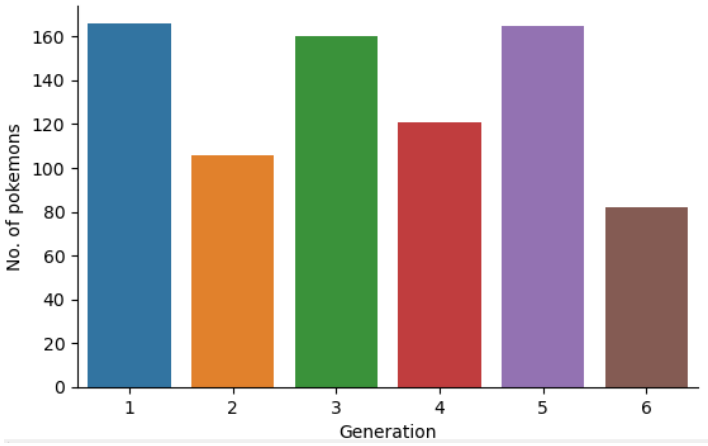
- new_df = df[10:16]
  new_df
  new_df.shape

```
In [22]: new_df = df[10:16]
    ...: new_df
    ...: new_df.shape
Out[22]: (6, 12)
```

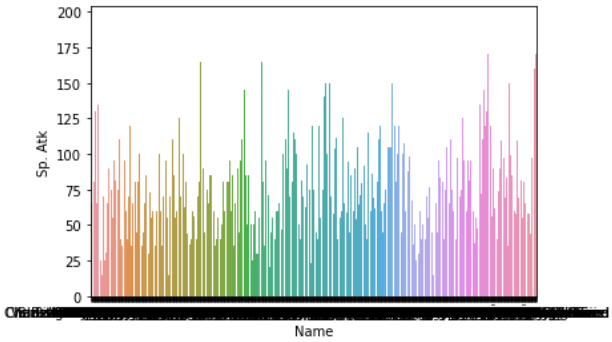
- sns.factorplot(y='Type 1',data=df,kind='count',order=df['Type 1'].value_counts().index)



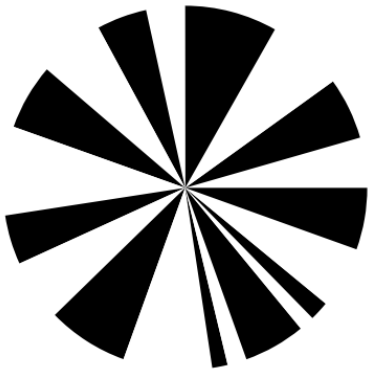
- # plot data no. of pokemon generations...
  sns.catplot(x="Generation",kind='count',data=df)
  plt.xlabel('Generation')
  plt.ylabel('No. of pokemons')
  plt.show()



- df= pd.read_excel('pokemon_data.xlsx')
  df.reset_index(drop=True,inplace=True)
  sns.barplot(x="Name",y="Sp. Atk",data=df)
  df = df[:20]
  df



- plt.figure(figsize=(16,8))
  plt.pie(x="Sp. Def",colors="101010",data=df)
  plt.show()



PIE CHART

- lab = df['Type 1'].unique()
  # explode pie where the Type 1 Column name is equal to Ghost or Normal.
  default= 0
  size = len(lab)
  exp = [default] * size # creating a int with having value of length of size.
  for i in range(size):
    if lab[i] == 'Ghost':
      exp[i] = 0.2
  exp = tuple(exp) # converting to convert
  plt.pie(x=df['Type 1'].value_counts(),explode=exp,labels=lab,shadow='True',startangle=0,autopct='%1.1f%%',radius=3,textprops={"fontsize":15})
  plt.show()