```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
# Reading CSV file
path="/content/drive/MyDrive/Lectures_ML/Modified RAFM dataset .csv"
# Using CSV instruction to read a txt file

df = pd.read_csv(path)
# Reading Excel file
df_xls = pd.read_excel("/content/drive/MyDrive/Lectures_ML/Modified RAFM dataset.xlsx")
```

```
/usr/local/lib/python3.7/dist-packages/openpyxl/worksheet/_reader.py:312: UserWarning: Unknown extension is not suppc
  warn(msg)
```

```
# For csv file
print(df.head(1))

# for excel file
print(df_xls.head(3))
```

```
   Ref.No.     C    Cr     W    Si     V  ...    B   TT  Tt(min)     YS     US    TE
0        1  0.12  8.73  2.09  0.25  0.25  ...  0.0  750       60  549.0  659.0  12.3

[1 rows x 15 columns]
  Unnamed: 0 Unnamed: 1 Unnamed: 2  ... Unnamed: 12 Unnamed: 13 Unnamed: 14
0    Ref.No.          C         Cr  ...          YS          US          TE
1          1       0.12       8.73  ...         549         659        12.3
2          1        0.1       8.72  ...         544         652        12.3

[3 rows x 15 columns]
```

```
#  Reading TXT file
dftxt= pd.read_csv("/content/drive/MyDrive/Lectures_ML/example_1.txt", delimiter='\t')
print(dftxt)
```

```
        #                    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

[800 rows x 12 columns]
```

```
# Read columns
print(df.columns)

# Read a specific column
#df['Ref.No.']
print(df['Ref.No.'][0:5])
# or it can be printed as:
#df.Cr
```

```
Index(['Ref.No.', 'C', 'Cr', 'W', 'Si', 'V', 'Ta', 'Ti', 'N', 'B', 'TT',
       'Tt(min)', 'YS', 'US', 'TE'],
      dtype='object')
0    1
1    1
2    1
3    2
4    2
Name: Ref.No., dtype: int64
```

```
# Read several columns
# print(df[['Ref.No.', 'C']])
print(df[['Ref.No.', 'C']][0:5])
```

```
   Ref.No.      C
0        1  0.120
1        1  0.100
2        1  0.092
3        2  0.100
4        2  0.100
```

```
# Read a row
print(df.iloc[0:4])
```

```
   Ref.No.      C    Cr     W    Si  ...   TT  Tt(min)     YS     US    TE
0        1  0.120  8.73  2.09  0.25  ...  750       60  549.0  659.0  12.3
1        1  0.100  8.72  2.09  0.23  ...  750       60  544.0  652.0  12.3
2        1  0.092  8.32  0.00  0.15  ...  760       60  539.0  630.0  13.3
3        2  0.100  9.30  0.93  0.11  ...  650      120  674.0  783.0  13.0

[4 rows x 15 columns]
```

```
# Read a specific location
print(df.iloc[2,1])
# Using a loop For
# for index, row in df.iterrows():
  # print(index, row)
  # print(index, row['C'])
```

```
0.092
```

```
df.loc[df['C']== 0.100]
```

	Ref.No.	C	Cr	W	Si	V	Ta	Ti	N	B	TT	Tt(min)	YS	US	TE
1	1	0.1	8.72	2.09	0.230	0.230	0.070	0.0000	0.000	0.0	750	60	544.0	652.0	12.3
3	2	0.1	9.30	0.93	0.110	0.220	0.094	0.0000	0.002	0.0	650	120	674.0	783.0	13.0
4	2	0.1	9.30	0.93	0.110	0.220	0.094	0.0000	0.002	0.0	750	120	538.0	657.0	15.0
5	2	0.1	9.30	0.95	0.130	0.230	0.000	0.0560	0.002	0.0	650	120	573.0	698.0	16.0
6	2	0.1	9.30	0.95	0.130	0.230	0.000	0.0560	0.002	0.0	750	120	464.0	605.0	23.0
7	3	0.1	9.30	2.22	0.120	0.052	0.000	0.0056	0.430	0.0	780	90	560.0	700.0	22.0
10	5	0.1	8.96	1.10	0.086	0.210	0.074	0.0000	0.000	0.0	760	90	500.0	640.0	30.0
51	11	0.1	9.00	1.10	0.250	0.300	0.140	0.0000	0.060	0.0	750	30	652.0	NaN	20.0
52	11	0.1	8.60	1.30	0.100	0.200	0.100	0.1500	0.003	0.0	750	30	787.0	NaN	14.3

Sorting/Describing Data

```
df.describe()
```

	Ref.No.	C	Cr	W	Si	V	Ta	Ti	N	B	TT	T
count	60.000000	60.000000	60.000000	60.000000	60.000000	60.000000	60.000000	60.000000	60.000000	60.000000	60.000000	60

```
# Sorting values given a pre defined name in a column
#print(df.sort_values('Cr'))
# print(df.sort_values('Cr', ascending=False))
print(df.sort_values(['TT', 'W'], ascending=[1,1]))
```

```
    31        6  0.120  8.95  2.01  0.230  ...  700       60  792.0  923.0   7.60
    33        6  0.110  8.90  2.01  0.210  ...  700       60  824.0  956.0   6.60
    35        6  0.094  8.38  2.02  0.210  ...  700       60  810.0  938.0   7.50
    13        6  0.110  4.67  2.11  0.200  ...  700       60  805.0  910.0   7.30
    15        6  0.120  4.65  2.14  0.190  ...  700       60  769.0  888.0   7.50
    21        6  0.110  4.61  2.87  0.210  ...  700       60  651.0  837.0   7.50
    27        6  0.120  6.98  2.97  0.190  ...  700       60  780.0  912.0   8.10
    23        6  0.110  4.61  2.99  0.210  ...  700       60  621.0  819.0   7.90
    17        6  0.096  4.97  3.00  0.120  ...  700       60  718.0  830.0   7.60
    19        6  0.110  4.63  3.01  0.220  ...  700       60  816.0  928.0   7.50
    42        8  0.110  8.55  1.48  0.000  ...  740       90  563.0  671.0  22.00
    41        8  0.120  8.50  1.50  0.000  ...  740       90  557.0  679.0  21.60
    4         2  0.100  9.30  0.93  0.110  ...  750      120  538.0  657.0  15.00
    6         2  0.100  9.30  0.95  0.130  ...  750      120  464.0  605.0  23.00
    48       10  0.049  8.80  0.98  0.036  ...  750      120  592.0  695.0  19.90
    47       10  0.049  8.82  0.99  0.038  ...  750      120  544.0  657.0  24.10
    46       10  0.045  8.83  1.02  0.034  ...  750      120  637.0  728.0  21.10
    59       12  0.100  8.32  1.03  0.101  ...  750       30  733.0  821.0  15.50
    51       11  0.100  9.00  1.10  0.250  ...  750       30  652.0    NaN  20.00
    55       12  0.110  8.94  1.20  0.109  ...  750       30  740.0  845.0  15.70
    56       12  0.096  8.54  1.28  0.101  ...  750       30  720.0  821.0  15.00
    57       12  0.110  8.44  1.28  0.097  ...  750       30  696.7  780.7  14.00
    58       12  0.103  8.51  1.29  0.091  ...  750       30  673.3  760.0  14.30
    52       11  0.100  8.60  1.30  0.100  ...  750       30  787.0    NaN  14.30
    53       11  0.100  8.60  1.30  0.150  ...  750       30  749.0    NaN  17.10

    49       10  0.048  8.81  1.96  0.035  ...  750      120  515.0  640.0  26.90
    54       12  0.055  9.20  1.97  0.120  ...  750       30  514.0  629.0  28.00
    30        6  0.120  7.02  1.98  0.190  ...  750       60  590.0  731.0   9.40
    12        6  0.120  5.04  2.01  0.230  ...  750       60  558.0  742.0  10.00
    26        6  0.120  7.01  2.01  0.190  ...  750       60  583.0  713.0  10.40
    32        6  0.120  8.95  2.01  0.230  ...  750       60  596.0  737.0   9.10
```

```
34         6  0.110  8.90  2.01  0.210  ...  750        60  646.0  787.0   7.90
44         9  0.120  8.79  2.01  0.220  ...  750       120  538.0  679.0  23.00
36         6  0.094  8.38  2.02  0.210  ...  750        60  655.0  771.0   8.50
45         9  0.120  8.84  2.05  0.770  ...  750       120  561.0  730.0  23.00
43         9  0.110  8.86  2.07  0.050  ...  750       120  527.0  660.0  23.00
50        10  0.071  8.87  2.08  0.400  ...  750       120  519.0  668.8  27.90
0          1  0.120  8.73  2.09  0.250  ...  750        60  549.0  659.0  12.30
1          1  0.100  8.72  2.09  0.230  ...  750        60  544.0  652.0  12.30
14         6  0.110  4.67  2.11  0.200  ...  750        60  627.0  770.0   8.20
16         6  0.120  4.65  2.14  0.190  ...  750        60  597.0  746.0   9.30
22         6  0.110  4.61  2.87  0.210  ...  750        60  585.0  758.0  10.70
28         6  0.120  6.98  2.97  0.190  ...  750        60  597.0  756.0  11.10
24         6  0.110  4.61  2.99  0.210  ...  750        60  582.0  758.0  10.70
18         6  0.096  4.97  3.00  0.120  ...  750        60  561.0  706.0   9.50
20         6  0.110  4.63  3.01  0.220  ...  750        60  558.0  733.0   9.90
2          1  0.092  8.32  0.00  0.150  ...  760        60  539.0  630.0  13.30
9          4  0.110  8.70  1.00  0.000  ...  760        90  500.0  620.0  27.00
10         5  0.100  8.96  1.10  0.086  ...  760        90  500.0  640.0  30.00
37         7  0.080  9.04  1.00  0.090  ...  763        90  557.0  688.0  17.00
38         7  0.093  9.07  1.01  0.090  ...  763        90  492.0  650.0  17.00
39         7  0.126  9.03  1.39  0.060  ...  763        90  508.0  658.0  17.20
40         7  0.120  8.99  2.06  0.060  ...  763        90  521.0  676.0  17.40
7          3  0.100  9.30  2.22  0.120  ...  780        90  560.0  700.0  22.00
8          3  0.096  9.04  2.41  0.043  ...  780        90  520.0  690.0  22.00

[60 rows x 15 columns]
```

Making changes to the data

```
# df['Total'] = df['C'] + df['Cr']
# print(df['Total'])


df
```

28	6	0.120	6.98	2.97	0.190	0.240	0.050	0.0000	0.0140	0.00000	750	60	597.0	756.0	11.10
29	6	0.120	7.02	1.98	0.190	0.240	0.050	0.0000	0.0130	0.00400	700	60	744.0	868.0	7.50
30	6	0.120	7.02	1.98	0.190	0.240	0.050	0.0000	0.0130	0.00400	750	60	590.0	731.0	9.40
31	6	0.120	8.95	2.01	0.230	0.240	0.000	0.0000	0.0290	0.00000	700	60	792.0	923.0	7.60
32	6	0.120	8.95	2.01	0.230	0.240	0.000	0.0000	0.0290	0.00000	750	60	596.0	737.0	9.10
33	6	0.110	8.90	2.01	0.210	0.230	0.060	0.0000	0.0170	0.00000	700	60	824.0	956.0	6.60
34	6	0.110	8.90	2.01	0.210	0.230	0.060	0.0000	0.0170	0.00000	750	60	646.0	787.0	7.90
35	6	0.094	8.38	2.02	0.210	0.230	0.060	0.0000	0.0140	0.00500	700	60	810.0	938.0	7.50
36	6	0.094	8.38	2.02	0.210	0.230	0.060	0.0000	0.0140	0.00500	750	60	655.0	771.0	8.50
37	7	0.080	9.04	1.00	0.090	0.220	0.060	0.0000	0.0226	0.00050	763	90	557.0	688.0	17.00
38	7	0.093	9.07	1.01	0.090	0.220	0.060	0.0000	0.0200	0.00050	763	90	492.0	650.0	17.00
39	7	0.126	9.03	1.39	0.060	0.240	0.060	0.0000	0.0300	0.00000	763	90	508.0	658.0	17.20
40	7	0.120	8.99	2.06	0.060	0.240	0.060	0.0000	0.0200	0.00000	763	90	521.0	676.0	17.40
41	8	0.120	8.50	1.50	0.000	0.250	0.100	0.0000	0.0067	0.00000	740	90	557.0	679.0	21.60
42	8	0.110	8.55	1.48	0.000	0.250	0.100	0.0000	0.0250	0.00000	740	90	563.0	671.0	22.00
43	9	0.110	8.86	2.07	0.050	0.240	0.110	0.0000	0.0000	0.00000	750	120	527.0	660.0	23.00
44	9	0.120	8.79	2.01	0.220	0.240	0.110	0.0000	0.0000	0.00000	750	120	538.0	679.0	23.00
45	9	0.120	8.84	2.05	0.770	0.240	0.110	0.0000	0.0000	0.00000	750	120	561.0	730.0	23.00
46	10	0.045	8.83	1.02	0.034	0.280	0.180	0.0000	0.0170	0.00640	750	120	637.0	728.0	21.10
47	10	0.049	8.82	0.99	0.038	0.280	0.170	0.0000	0.0240	0.00000	750	120	544.0	657.0	24.10
48	10	0.049	8.80	0.98	0.036	0.280	0.190	0.0000	0.0370	0.00650	750	120	592.0	695.0	19.90
49	10	0.048	8.81	1.96	0.035	0.300	0.200	0.0000	0.0610	0.00039	750	120	515.0	640.0	26.90
50	10	0.071	8.87	2.08	0.400	0.300	0.130	0.0000	0.0770	0.00000	750	120	519.0	668.8	27.90

51	11	0.100	9.00	1.10	0.250	0.300	0.140	0.0000	0.0600	0.00000	750	30	652.0	NaN	20.00
52	11	0.100	8.60	1.30	0.100	0.200	0.100	0.1500	0.0030	0.00000	750	30	787.0	NaN	14.30
53	11	0.100	8.60	1.30	0.150	0.100	0.100	0.1500	0.0040	0.00000	750	30	749.0	NaN	17.10
54	12	0.055	9.20	1.97	0.120	0.220	0.200	0.0000	0.0400	0.00000	750	30	514.0	629.0	28.00
55	12	0.110	8.94	1.20	0.109	0.198	0.078	0.0220	0.0034	0.00000	750	30	740.0	845.0	15.70
56	12	0.096	8.54	1.28	0.101	0.210	0.100	0.1400	0.0030	0.00000	750	30	720.0	821.0	15.00
57	12	0.110	8.44	1.28	0.097	0.190	0.290	0.1380	0.0030	0.00000	750	30	696.7	780.7	14.00
58	12	0.103	8.51	1.29	0.091	0.203	0.550	0.1450	0.0030	0.00000	750	30	673.3	760.0	14.30
59	12	0.100	8.32	1.03	0.101	0.170	0.085	0.1520	0.0030	0.00000	750	30	733.0	821.0	15.50

```
# Dropping a column once has been created
# df.drop(columns=['Total'])
#   Adding predefined columns using their location
df['Total'] = df.iloc[:,0:2].sum(axis=1)


# df['Total'] # the end parameter (column) is exclusive
# df
```

Reordering the columns in the table

[ ] ↳ *3 cells hidden*

Filtering over **data**

[ ] ↳ *5 cells hidden*

Filtered by name or word

[ ] ↳ *1 cell hidden*

Going further!

[ ] ↳ *5 cells hidden*

Conditional Changes

[ ] ↳ *1 cell hidden*

What about conditional change using more than two columns?

[ ] ↳ *2 cells hidden*

Aggregate some Statistics

```
# Lets read again the file since we have messed it up
```

```
df_xls = pd.read_excel("/content/drive/MyDrive/Lectures_ML/data_Example.xlsx")
```

```
/usr/local/lib/python3.7/dist-packages/openpyxl/worksheet/_reader.py:312: UserWarning: Unknown extension is not suppo
  warn(msg)
```

```
df_xls.groupby(['Type 1']).mean()
```

Type 1	#	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
Bug	334.492754	56.884058	70.971014	70.724638	53.869565	64.797101	61.681159	3.217391	0.000000

```
df_xls.groupby(['Type 1']).mean().sort_values('Defense', ascending=False)
```

	#	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
Type 1									
Fairy	449.529412	74.117647	61.529412	65.705882	78.529412	84.705882	48.588235	4.117647	0.058824

```
# df_xls.groupby(['Type 1']).sum()
# df_xls.groupby(['Type 1']).count()
df_xls['count'] = 1
df_xls
# Just to show the column with the count
df_xls.groupby(['Type 1']).count()['count']
```

```
Type 1
Bug          69
Dark         31
Dragon       32
Electric     44
Fairy        17
Fighting     27
Fire         52
Flying        4
Ghost        32
Grass        70
Ground       32
Ice          24
Normal       98
Poison       28
Psychic      57
Rock         44
Steel        27
Water       112
Name: count, dtype: int64
```

```
# Multiple parameters
df_xls.groupby(['Type 1', 'Type 2']).count()['count']
```

```
Type 1  Type 2
Bug     Electric     2
        Fighting     2
        Fire         2
```

```
              Flying      14
              Ghost        1
                         ..
     Water    Ice          3
              Poison       3
              Psychic      5
              Rock         4
              Steel        1
     Name: count, Length: 136, dtype: int64
```

```
#Working with large amounts of Data
#  Panda allows you to work with big data
df_xls.to_csv('/content/drive/MyDrive/Lectures_ML/modified.csv')
for df_1 in pd.read_csv('/content/drive/MyDrive/Lectures_ML/modified.csv', chunksize=5):
  print("Chunk size")
  print(df_1)
```

```
     768         768  698     Amaura      Rock  ...    46           6      False      1
     769         769  699    Aurorus      Rock  ...    58           6      False      1

     [5 rows x 14 columns]
     Chunk size
          Unnamed: 0    #      Name    Type 1  ... Speed  Generation  Legendary  count
     770         770  700   Sylveon     Fairy  ...    60           6      False      1
     771         771  701  Hawlucha  Fighting  ...   118           6      False      1
     772         772  702   Dedenne  Electric  ...   101           6      False      1
     773         773  703   Carbink      Rock  ...    50           6      False      1
     774         774  704     Goomy    Dragon  ...    40           6      False      1

     [5 rows x 14 columns]
     Chunk size
          Unnamed: 0    #       Name  Type 1  ... Speed  Generation  Legendary  count
     775         775  705    Sliggoo  Dragon  ...    60           6      False      1
     776         776  706     Goodra  Dragon  ...    80           6      False      1
     777         777  707     Klefki   Steel  ...    75           6      False      1
     778         778  708   Phantump   Ghost  ...    38           6      False      1
     779         779  709  Trevenant   Ghost  ...    56           6      False      1

     [5 rows x 14 columns]
     Chunk size
```

```
     Unnamed: 0    #                    Name  ... Generation Legendary  count
780         780  710  PumpkabooAverage Size  ...          6     False      1
781         781  710    PumpkabooSmall Size  ...          6     False      1
782         782  710    PumpkabooLarge Size  ...          6     False      1
783         783  710    PumpkabooSuper Size  ...          6     False      1
784         784  711  GourgeistAverage Size  ...          6     False      1

[5 rows x 14 columns]
Chunk size
     Unnamed: 0    #                    Name  ... Generation Legendary  count
785         785  711  GourgeistSmall Size  ...          6     False      1
786         786  711  GourgeistLarge Size  ...          6     False      1
787         787  711  GourgeistSuper Size  ...          6     False      1
788         788  712             Bergmite  ...          6     False      1
789         789  713              Avalugg  ...          6     False      1

[5 rows x 14 columns]
Chunk size
     Unnamed: 0    #              Name  ... Generation Legendary  count
790         790  714            Noibat  ...          6     False      1
791         791  715           Noivern  ...          6     False      1
792         792  716           Xerneas  ...          6      True      1
793         793  717           Yveltal  ...          6      True      1
794         794  718  Zygarde50% Forme  ...          6      True      1

[5 rows x 14 columns]
Chunk size
     Unnamed: 0    #                    Name  ... Generation Legendary  count
795         795  719                 Diancie  ...          6      True      1
796         796  719  DiancieMega Diancie     ...          6      True      1
797         797  720  HoopaHoopa Confined     ...          6      True      1
798         798  720   HoopaHoopa Unbound     ...          6      True      1
799         799  721              Volcanion  ...          6      True      1

[5 rows x 14 columns]
```

```
new_df = pd.DataFrame(columns=df.columns)
for df_1 in pd.read_csv('/content/drive/MyDrive/Lectures_ML/modified.csv', chunksize=5):
  results = df_1.groupby(['Type 1']).count()
```

```
  new_df = pd.concat([new_df, results])
new_df
```

	Ref.No.	C	Cr	Total	W	Si	V	Ta	Ti	N	B	TT	Tt(min)	YS	US	Unnamed: 0	#	Name	Typ
Fire	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	1.0	1.0	0
Grass	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	4.0	4.0	4.0	4
Fire	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	4.0	4.0	4.0	3
Water	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	1.0	1.0	0
Bug	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.0	2.0	2.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
Fairy	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	1.0	1.0	0
Flying	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.0	2.0	2.0	2
Fire	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	1.0	1.0	1
Psychic	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.0	2.0	2.0	2
Rock	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.0	2.0	2.0	2

433 rows × 28 columns