Data management using Pandas

Data management is a crucial component to statistical analysis and data science work.

This notebook will show you how to import, view, undertand, and manage your data using the Pandas data processing library, i.e., the notebook will demonstrates how to read a dataset into Python, and obtain a basic understanding of its content.

Note that **Python** by itself is a general-purpose programming language and does not provide high-level data processing capabilities. The **Pandas** library was developed to meet this need. **Pandas** is the most popular Python library for data manipulation, and we will use it extensively in this course. **Pandas** provides high-performance, easy-to-use data structures and data analysis tools.

The main data structure that **Pandas** works with is called a **Data Frame**. This is a two-dimensional table of data in which the rows typically represent cases and the columns represent variables (e.g. data used in this tutorial). Pandas also has a one-dimensional data structure called a **Series** that we will encounter when accesing a single column of a Data Frame.

Pandas has a variety of functions named `read_xxx` for reading data in different formats. Right now we will focus on reading `csv` files, which stands for comma-separated values. However the other file formats include `excel`, `json`, and `sql`.

There are many other options to `read_csv` that are very useful. For example, you would use the option `sep='\t'` instead of the default `sep=','` if the fields of your data file are delimited by tabs instead of commas. See here for the full documentation for `read_csv`.

Acknowledgments

- The dataset used in this tutorial is from https://www.coursera.org/ from the course "Understanding and Visualizing Data with Python" by University of Michigan

Importing libraries

```
# Import the packages that we will be using
import pandas as pd
```

Importing data

```
# Define where you are running the code: colab or local
RunInColab          = False     # (False: no  | True: yes)

# If running in colab:
if RunInColab:
    # Mount your google drive in google colab
    from google.colab import drive
```

```
    drive.mount('/content/drive')

    # Find location
    #!pwd
    #!ls
    #!ls "/content/drive/My Drive/Colab
Notebooks/MachineLearningWithPython/"

    # Define path del proyecto
    Ruta            = "/content/drive/My Drive/Colab
Notebooks/MachineLearningWithPython/"

else:
    # Define path del proyecto
    Ruta            = "datasets/cartwheel/cartwheel.csv"

# url string that hosts our .csv file

# Read the .csv file and store it as a pandas Data Frame
df  = pd.read_csv(Ruta)
```

If we want to print the information about th output object type we would simply type the following: type(df)

```
type(df)
```

```
pandas.core.frame.DataFrame
```

Exploring the content of the data set

Use the `shape` method to determine the numbers of rows and columns in a data frame. This can be used to confirm that we have actually obtained the data the we are expecting.

Based on what we see below, the data set being read here has N_r rows, corresponding to N_r observations, and N_c columns, corresponding to N_c variables in this particular data file.

```
df.shape
```

```
(52, 12)
```

```
df.shape [0]
```

```
52
```

```
df.shape [1]
```

```
12
```

If we want to show the entire data frame we would simply write the following:

```
df
```

	ID	Age	Gender	GenderGroup	Glasses	GlassesGroup	Height	Wingspan
0	1	56.0	F	1	Y	1	62.00	61.0
1	2	26.0	F	1	Y	1	62.00	60.0
2	3	33.0	F	1	Y	1	66.00	64.0
3	4	39.0	F	1	N	0	64.00	63.0
4	5	27.0	M	2	N	0	73.00	75.0
5	6	24.0	M	2	N	0	75.00	71.0
6	7	28.0	M	2	N	0	75.00	76.0
7	8	22.0	F	1	N	0	65.00	62.0
8	9	29.0	M	2	Y	1	74.00	73.0
9	10	33.0	F	1	Y	1	63.00	60.0
10	11	30.0	M	2	Y	1	69.50	66.0
11	12	28.0	F	1	Y	1	62.75	58.0
12	13	25.0	F	1	Y	1	65.00	64.5
13	14	23.0	F	1	N	0	61.50	57.5
14	15	31.0	M	2	Y	1	73.00	74.0
15	16	26.0	M	2	Y	1	71.00	72.0
16	17	26.0	F	1	N	0	61.50	59.5
17	18	27.0	M	2	N	0	66.00	66.0
18	19	23.0	M	2	Y	1	70.00	69.0
19	20	24.0	F	1	Y	1	68.00	66.0
20	21	23.0	M	2	Y	1	69.00	67.0
21	22	29.0	M	2	N	0	71.00	70.0
22	23	25.0	M	2	N	0	70.00	68.0
23	24	26.0	M	2	N	0	69.00	71.0

24 25 23.0 F 1 Y 1 65.00
63.0
25 26 28.0 M 2 N 0 75.00
76.0
26 27 24.0 M 2 N 0 78.40
71.0
27 28 25.0 M 2 Y 1 76.00
73.0
28 29 32.0 F 1 Y 1 63.00
60.0
29 30 38.0 F 1 Y 1 61.50
61.0
30 31 27.0 F 1 Y 1 62.00
60.0
31 32 33.0 F 1 Y 1 65.30
64.0
32 33 38.0 F 1 N 0 64.00
63.0
33 34 27.0 M 2 N 0 77.00
75.0
34 35 24.0 F 1 N 0 67.80
62.0
35 36 27.0 M 2 N 0 68.00
66.0
36 37 25.0 F 1 Y 1 65.00
64.5
37 38 26.0 F 1 N 0 61.50
59.5
38 39 31.0 M 2 Y 1 73.00
74.0
39 40 30.0 M 2 Y 1 69.50
66.0
40 41 23.0 F 1 N 0 70.40
71.0
41 42 26.0 M 2 Y 1 73.50
72.0
42 43 28.0 F 1 Y 1 72.50
72.0
43 44 26.0 F 1 Y 1 72.00
72.0
44 45 30.0 F 1 Y 1 66.00
64.0
45 46 39.0 F 1 N 0 64.00
63.0
46 47 27.0 M 2 N 0 78.00
75.0
47 48 24.0 M 2 N 0 79.50
75.0
48 49 28.0 M 2 N 0 77.80
76.0

```
49  50  30.0      F            1       N             0   74.60
NaN
50  51   NaN      M            2       N             0   71.00
70.0
51  52  27.0      M            2       N             0     NaN
71.5
```

	CWDistance	Complete	CompleteGroup	Score
0	79	Y	1.0	7
1	70	Y	1.0	8
2	85	Y	1.0	7
3	87	Y	1.0	10
4	72	N	0.0	4
5	81	N	0.0	3
6	107	Y	1.0	10
7	98	Y	1.0	9
8	106	N	0.0	5
9	65	Y	1.0	8
10	96	Y	1.0	6
11	79	Y	1.0	10
12	92	Y	1.0	6
13	66	Y	1.0	4
14	72	Y	1.0	9
15	115	Y	1.0	6
16	90	N	0.0	10
17	74	Y	1.0	5
18	64	Y	1.0	3
19	85	Y	1.0	8
20	66	N	0.0	2
21	101	Y	1.0	8
22	82	Y	1.0	4
23	63	Y	1.0	5
24	67	N	0.0	3
25	111	Y	1.0	10
26	92	Y	1.0	7
27	107	Y	1.0	8
28	75	Y	1.0	8
29	78	Y	1.0	7
30	72	Y	1.0	8
31	91	Y	1.0	7
32	86	Y	1.0	10
33	100	Y	1.0	8
34	98	Y	1.0	9
35	74	Y	1.0	5
36	92	Y	1.0	6
37	90	Y	1.0	9
38	72	Y	1.0	9
39	96	Y	1.0	6
40	66	Y	1.0	4
41	115	Y	1.0	6

```
42          81        Y            1.0     10
43          92        Y            1.0      8
44          85        Y            1.0      7
45          87        Y            1.0     10
46          72        N            0.0      7
47          82        N            0.0      8
48          99        Y            1.0      9
49          71        Y            1.0      9
50         101        Y            NaN      8
51         103        Y            1.0     10
```

As you can see, we have a 2-Dimensional object where each row is an independent observation and each coloum is a variable.

Now, use the the `head()` function to show the first 5 rows of our data frame

```
df.head()
```

```
   ID   Age Gender  GenderGroup Glasses  GlassesGroup  Height
Wingspan  \
0   1  56.0      F            1       Y             1    62.0
61.0
1   2  26.0      F            1       Y             1    62.0
60.0
2   3  33.0      F            1       Y             1    66.0
64.0
3   4  39.0      F            1       N             0    64.0
63.0
4   5  27.0      M            2       N             0    73.0
75.0

   CWDistance Complete  CompleteGroup  Score
0          79        Y            1.0      7
1          70        Y            1.0      8
2          85        Y            1.0      7
3          87        Y            1.0     10
4          72        N            0.0      4
```

Also, you can use the the `tail()` function to show the last 5 rows of our data frame

```
df.tail()
```

```
    ID   Age Gender  GenderGroup Glasses  GlassesGroup  Height
Wingspan  \
47  48  24.0      M            2       N             0    79.5
75.0
48  49  28.0      M            2       N             0    77.8
76.0
49  50  30.0      F            1       N             0    74.6
NaN
50  51   NaN      M            2       N             0    71.0
```

```
70.0
51  52  27.0      M            2       N             0     NaN
71.5

    CWDistance Complete  CompleteGroup  Score
47          82        N            0.0      8
48          99        Y            1.0      9
49          71        Y            1.0      9
50         101        Y            NaN      8
51         103        Y            1.0     10
```

The columns in a Pandas data frame have names, to see the names, use the `columns` method:

To gather more information regarding the data, we can view the column names with the following function:

```
df.columns
```

```
Index(['ID', 'Age', 'Gender', 'GenderGroup', 'Glasses',
'GlassesGroup',
       'Height', 'Wingspan', 'CWDistance', 'Complete',
'CompleteGroup',
       'Score'],
      dtype='object')
```

Be aware that every variable in a Pandas data frame has a data type. There are many different data types, but most commonly you will encounter floating point values (real numbers), integers, strings (text), and date/time values. When Pandas reads a text/csv file, it guesses the data types based on what it sees in the first few rows of the data file. Usually it selects an appropriate type, but occasionally it does not. To confirm that the data types are consistent with what the variables represent, inspect the `dtypes` attribute of the data frame.

```
df.dtypes
```

```
ID                int64
Age             float64
Gender           object
GenderGroup       int64
Glasses          object
GlassesGroup      int64
Height          float64
Wingspan        float64
CWDistance        int64
Complete         object
CompleteGroup   float64
Score             int64
dtype: object
```

Summary statistics, which include things like the mean, min, and max of the data, can be useful to get a feel for how large some of the variables are and what variables may be the most important.

```
# Summary statistics for the quantitative variables
df.info()
df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52 entries, 0 to 51
Data columns (total 12 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   ID             52 non-null     int64
 1   Age            51 non-null     float64
 2   Gender         52 non-null     object
 3   GenderGroup    52 non-null     int64
 4   Glasses        52 non-null     object
 5   GlassesGroup   52 non-null     int64
 6   Height         51 non-null     float64
 7   Wingspan       51 non-null     float64
 8   CWDistance     52 non-null     int64
 9   Complete       52 non-null     object
 10  CompleteGroup  51 non-null     float64
 11  Score          52 non-null     int64
dtypes: float64(4), int64(5), object(3)
memory usage: 5.0+ KB
```

```
              ID        Age  GenderGroup  GlassesGroup     Height
Wingspan  \
count  52.000000  51.000000    52.000000     52.000000  51.000000
51.000000
mean   26.500000  28.411765     1.500000      0.500000  68.971569
67.313725
std    15.154757   5.755611     0.504878      0.504878   5.303812
5.624021
min     1.000000  22.000000     1.000000      0.000000  61.500000
57.500000
25%    13.750000  25.000000     1.000000      0.000000  64.500000
63.000000
50%    26.500000  27.000000     1.500000      0.500000  69.000000
66.000000
75%    39.250000  30.000000     2.000000      1.000000  73.000000
72.000000
max    52.000000  56.000000     2.000000      1.000000  79.500000
76.000000

       CWDistance  CompleteGroup      Score
count   52.000000      51.000000  52.000000
```

```
mean    85.576923       0.843137   7.173077
std     14.353173       0.367290   2.211566
min     63.000000       0.000000   2.000000
25%     72.000000       1.000000   6.000000
50%     85.000000       1.000000   8.000000
75%     96.500000       1.000000   9.000000
max    115.000000       1.000000  10.000000
```

```
# Drop observations with NaN values
df.Age.dropna().describe()
df.Wingspan.dropna().describe()
```

```
count    51.000000
mean     67.313725
std       5.624021
min      57.500000
25%      63.000000
50%      66.000000
75%      72.000000
max      76.000000
Name: Wingspan, dtype: float64
```

It is also possible to get statistics on the entire data frame or a column as follows

- `df.mean()` Returns the mean of all columns
- `df.corr()` Returns the correlation between columns in a data frame
- `df.count()` Returns the number of non-null values in each data frame column
- `df.max()` Returns the highest value in each column
- `df.min()` Returns the lowest value in each column
- `df.median()` Returns the median of each column
- `df.std()` Returns the standard deviation of each column

```
df.std()
```

```
C:\Users\vanec\AppData\Local\Temp\ipykernel_28148\3249793613.py:1:
FutureWarning: The default value of numeric_only in DataFrame.std is
deprecated. In a future version, it will default to False. In
addition, specifying 'numeric_only=None' is deprecated. Select only
valid columns or specify the value of numeric_only to silence this
warning.
  df.std()
```

```
ID              15.154757
Age              5.755611
GenderGroup      0.504878
GlassesGroup     0.504878
Height           5.303812
Wingspan         5.624021
CWDistance      14.353173
CompleteGroup    0.367290
```

```
Score             2.211566
dtype: float64
```

How to write a data frame to a File

To save a file with your data simply use the `to_csv` attribute

Examples:

- df.to_csv('myDataFrame.csv')
- df.to_csv('myDataFrame.csv', sep='\t')

```
df.to_csv('cartwheel.csv', sep='\t')
```

Rename columns

To change the name of a colum use the `rename` attribute

Example:

df = df.rename(columns={"Age": "Edad"})

df.head()

```
df = df.rename(columns={'CWDistance':'CartwheelDistance'})

# Back to the original name
df = df.rename(columns={'CartwheelDistance':'CWDistance'})
```

Selection of colums

As discussed above, a Pandas data frame is a rectangular data table, in which the rows represent observations or samples and the columns represent variables. One common manipulation of a data frame is to extract the data for one case or for one variable. There are several ways to do this, as shown below.

To extract all the values for one column (variable), use one of the following alternatives.

```
a = df.Age
b = df["Age"]
c = df.loc[:, "Age"]
d = df.iloc[:, 1]

print(d)

df[["Gender", "GenderGroup"]]
```

0	56.0
1	26.0
2	33.0
3	39.0
4	27.0
5	24.0
6	28.0
7	22.0
8	29.0
9	33.0
10	30.0
11	28.0
12	25.0
13	23.0
14	31.0
15	26.0
16	26.0
17	27.0
18	23.0
19	24.0
20	23.0
21	29.0
22	25.0
23	26.0
24	23.0
25	28.0
26	24.0
27	25.0
28	32.0
29	38.0
30	27.0
31	33.0
32	38.0
33	27.0
34	24.0
35	27.0
36	25.0
37	26.0
38	31.0
39	30.0
40	23.0
41	26.0
42	28.0
43	26.0
44	30.0
45	39.0
46	27.0
47	24.0
48	28.0
49	30.0

```
50     NaN
51    27.0
Name: Age, dtype: float64
```

	Gender	GenderGroup
0	F	1
1	F	1
2	F	1
3	F	1
4	M	2
5	M	2
6	M	2
7	F	1
8	M	2
9	F	1
10	M	2
11	F	1
12	F	1
13	F	1
14	M	2
15	M	2
16	F	1
17	M	2
18	M	2
19	F	1
20	M	2
21	M	2
22	M	2
23	M	2
24	F	1
25	M	2
26	M	2
27	M	2
28	F	1
29	F	1
30	F	1
31	F	1
32	F	1
33	M	2
34	F	1
35	M	2
36	F	1
37	F	1
38	M	2
39	M	2
40	F	1
41	M	2
42	F	1
43	F	1
44	F	1

```
45      F             1
46      M             2
47      M             2
48      M             2
49      F             1
50      M             2
51      M             2
```

Slicing a data set

As discussed above, a Pandas data frame is a rectangular data table, in which the rows represent cases and the columns represent variables. One common manipulation of a data frame is to extract the data for one observation or for one variable. There are several ways to do this, as shown below.

Lets say we would like to splice our data frame and select only specific portions of our data. There are three different ways of doing so.

1. .loc()
2. .iloc()
3. .ix()

We will cover the .loc() and .iloc() splicing functions.

The attibute **.loc()** uses labels/column names, in specific, it takes two single/list/range operator separated by ',', the first one indicates the rows and the second one indicates columns.

```
# Return all observations of CWDistance
df.loc[:,"CWDistance"]

# Return a subset of observations of CWDistance
df.loc[:9, "CWDistance"]

# Select all rows for multiple columns, ["Gender", "GenderGroup"]
df.loc[:,["Gender", "GenderGroup"]]

# Select multiple columns, ["Gender", "GenderGroup"]me
keep = ['Gender', 'GenderGroup']
df_gender = df[keep]

# Select few rows for multiple columns, ["CWDistance", "Height",
"Wingspan"]
df.loc[4:9, ["CWDistance", "Height", "Wingspan"]]

# Select range of rows for all columns
df.loc[10:15,:]
```

```
    ID   Age Gender  GenderGroup Glasses  GlassesGroup  Height
Wingspan  \
10  11  30.0      M            2       Y             1   69.50
66.0
11  12  28.0      F            1       Y             1   62.75
58.0
12  13  25.0      F            1       Y             1   65.00
64.5
13  14  23.0      F            1       N             0   61.50
57.5
14  15  31.0      M            2       Y             1   73.00
74.0
15  16  26.0      M            2       Y             1   71.00
72.0

    CWDistance Complete  CompleteGroup  Score
10          96        Y            1.0      6
11          79        Y            1.0     10
12          92        Y            1.0      6
13          66        Y            1.0      4
14          72        Y            1.0      9
15         115        Y            1.0      6
```

The attribute **iloc()** is an integer based slicing.

```
# Todos los renglones de las primeras 4 columnas#
df.iloc[:, :4]

# Los primeros 4 renglones de todas las columnas
df.iloc[:4, :]

# Tdos los renglones de las columnas 3 a 7
df.iloc[:, 3:7]

# Los renglones 4 a 8 de las columnas 2 a 4
df.iloc[4:8, 2:4]

# This is incorrect:
#df.iloc[1:5, ["Gender", "GenderGroup"]]
```

```
  Gender  GenderGroup
4      M            2
5      M            2
6      M            2
7      F            1
```

Get unique existing values

List unique values in the one of the columns

df.Gender.unique()

```
# List unique values in the df['Gender'] column

df.Gender.unique()
```

```
array(['F', 'M'], dtype=object)
```

```
# Lets explore df["GenderGroup] as well
df.GenderGroup.unique()
```

```
array([1, 2], dtype=int64)
```

Filter, Sort and Groupby

With **Filter** you can use different conditions to filter columns. For example, df[df[year] > 1984] would give you only the column year is greater than 1984. You can use & (and) or | (or) to add different conditions to your filtering. This is also called boolean filtering.

df[df["Height"] >= 70]

```
df[df["Height"] >= 70]
```

```
    ID   Age Gender  GenderGroup Glasses  GlassesGroup  Height
Wingspan  \
4    5  27.0      M            2       N             0    73.0
75.0
5    6  24.0      M            2       N             0    75.0
71.0
6    7  28.0      M            2       N             0    75.0
76.0
8    9  29.0      M            2       Y             1    74.0
73.0
14  15  31.0      M            2       Y             1    73.0
74.0
15  16  26.0      M            2       Y             1    71.0
72.0
18  19  23.0      M            2       Y             1    70.0
69.0
21  22  29.0      M            2       N             0    71.0
70.0
22  23  25.0      M            2       N             0    70.0
68.0
25  26  28.0      M            2       N             0    75.0
76.0
26  27  24.0      M            2       N             0    78.4
71.0
27  28  25.0      M            2       Y             1    76.0
73.0
33  34  27.0      M            2       N             0    77.0
```

```
75.0
38  39  31.0      M            2       Y            1    73.0
74.0
40  41  23.0      F            1       N            0    70.4
71.0
41  42  26.0      M            2       Y            1    73.5
72.0
42  43  28.0      F            1       Y            1    72.5
72.0
43  44  26.0      F            1       Y            1    72.0
72.0
46  47  27.0      M            2       N            0    78.0
75.0
47  48  24.0      M            2       N            0    79.5
75.0
48  49  28.0      M            2       N            0    77.8
76.0
49  50  30.0      F            1       N            0    74.6
NaN
50  51   NaN      M            2       N            0    71.0
70.0
```

	CWDistance	Complete	CompleteGroup	Score
4	72	N	0.0	4
5	81	N	0.0	3
6	107	Y	1.0	10
8	106	N	0.0	5
14	72	Y	1.0	9
15	115	Y	1.0	6
18	64	Y	1.0	3
21	101	Y	1.0	8
22	82	Y	1.0	4
25	111	Y	1.0	10
26	92	Y	1.0	7
27	107	Y	1.0	8
33	100	Y	1.0	8
38	72	Y	1.0	9
40	66	Y	1.0	4
41	115	Y	1.0	6
42	81	Y	1.0	10
43	92	Y	1.0	8
46	72	N	0.0	7
47	82	N	0.0	8
48	99	Y	1.0	9
49	71	Y	1.0	9
50	101	Y	NaN	8

With **Sort** is possible to sort values in a certain column in an ascending order using `df.sort_values("ColumnName")` or in descending order using `df.sort_values(ColumnName, ascending=False)`.

Furthermore, it's possible to sort values by Column1Name in ascending order then Column2Name in descending order by using `df.sort_values([Column1Name,Column2Name],ascending=[True,False])`

df.sort_values("Height")

#df.sort_values("Height",ascending=False)

```
df.sort_values("Height",ascending=False)
```

```
    ID   Age Gender  GenderGroup Glasses  GlassesGroup  Height
Wingspan  \
47  48  24.0      M            2       N             0   79.50
75.0
26  27  24.0      M            2       N             0   78.40
71.0
46  47  27.0      M            2       N             0   78.00
75.0
48  49  28.0      M            2       N             0   77.80
76.0
33  34  27.0      M            2       N             0   77.00
75.0
27  28  25.0      M            2       Y             1   76.00
73.0
5    6  24.0      M            2       N             0   75.00
71.0
6    7  28.0      M            2       N             0   75.00
76.0
25  26  28.0      M            2       N             0   75.00
76.0
49  50  30.0      F            1       N             0   74.60
NaN
8    9  29.0      M            2       Y             1   74.00
73.0
41  42  26.0      M            2       Y             1   73.50
72.0
4    5  27.0      M            2       N             0   73.00
75.0
38  39  31.0      M            2       Y             1   73.00
74.0
14  15  31.0      M            2       Y             1   73.00
74.0
42  43  28.0      F            1       Y             1   72.50
72.0
43  44  26.0      F            1       Y             1   72.00
72.0
21  22  29.0      M            2       N             0   71.00
70.0
50  51   NaN      M            2       N             0   71.00
70.0
15  16  26.0      M            2       Y             1   71.00
```

72.0
40 41 23.0 F 1 N 0 70.40
71.0
18 19 23.0 M 2 Y 1 70.00
69.0
22 23 25.0 M 2 N 0 70.00
68.0
10 11 30.0 M 2 Y 1 69.50
66.0
39 40 30.0 M 2 Y 1 69.50
66.0
23 24 26.0 M 2 N 0 69.00
71.0
20 21 23.0 M 2 Y 1 69.00
67.0
19 20 24.0 F 1 Y 1 68.00
66.0
35 36 27.0 M 2 N 0 68.00
66.0
34 35 24.0 F 1 N 0 67.80
62.0
2   3 33.0 F 1 Y 1 66.00
64.0
17 18 27.0 M 2 N 0 66.00
66.0
44 45 30.0 F 1 Y 1 66.00
64.0
31 32 33.0 F 1 Y 1 65.30
64.0
36 37 25.0 F 1 Y 1 65.00
64.5
7   8 22.0 F 1 N 0 65.00
62.0
12 13 25.0 F 1 Y 1 65.00
64.5
24 25 23.0 F 1 Y 1 65.00
63.0
45 46 39.0 F 1 N 0 64.00
63.0
32 33 38.0 F 1 N 0 64.00
63.0
3   4 39.0 F 1 N 0 64.00
63.0
28 29 32.0 F 1 Y 1 63.00
60.0
9   10 33.0 F 1 Y 1 63.00
60.0
11 12 28.0 F 1 Y 1 62.75
58.0
1   2 26.0 F 1 Y 1 62.00

```
60.0
0    1  56.0      F            1       Y            1   62.00
61.0
30  31  27.0      F            1       Y            1   62.00
60.0
13  14  23.0      F            1       N            0   61.50
57.5
37  38  26.0      F            1       N            0   61.50
59.5
16  17  26.0      F            1       N            0   61.50
59.5
29  30  38.0      F            1       Y            1   61.50
61.0
51  52  27.0      M            2       N            0     NaN
71.5
```

	CWDistance	Complete	CompleteGroup	Score
47	82	N	0.0	8
26	92	Y	1.0	7
46	72	N	0.0	7
48	99	Y	1.0	9
33	100	Y	1.0	8
27	107	Y	1.0	8
5	81	N	0.0	3
6	107	Y	1.0	10
25	111	Y	1.0	10
49	71	Y	1.0	9
8	106	N	0.0	5
41	115	Y	1.0	6
4	72	N	0.0	4
38	72	Y	1.0	9
14	72	Y	1.0	9
42	81	Y	1.0	10
43	92	Y	1.0	8
21	101	Y	1.0	8
50	101	Y	NaN	8
15	115	Y	1.0	6
40	66	Y	1.0	4
18	64	Y	1.0	3
22	82	Y	1.0	4
10	96	Y	1.0	6
39	96	Y	1.0	6
23	63	Y	1.0	5
20	66	N	0.0	2
19	85	Y	1.0	8
35	74	Y	1.0	5
34	98	Y	1.0	9
2	85	Y	1.0	7
17	74	Y	1.0	5
44	85	Y	1.0	7

```
31          91        Y             1.0      7
36          92        Y             1.0      6
7           98        Y             1.0      9
12          92        Y             1.0      6
24          67        N             0.0      3
45          87        Y             1.0     10
32          86        Y             1.0     10
3           87        Y             1.0     10
28          75        Y             1.0      8
9           65        Y             1.0      8
11          79        Y             1.0     10
1           70        Y             1.0      8
0           79        Y             1.0      7
30          72        Y             1.0      8
13          66        Y             1.0      4
37          90        Y             1.0      9
16          90        N             0.0     10
29          78        Y             1.0      7
51         103        Y             1.0     10
```

The attribute **Groupby** involves splitting the data into groups based on some criteria, applying a function to each group independently and combining the results into a data structure. df.groupby(col) returns a groupby object for values from one column while df.groupby([col1,col2]) returns a groupby object for values from multiple columns.

df.groupby(['Gender'])

```
df.groupby(['Gender'])
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at
0x000001F635301450>
```

Size of each group

df.groupby(['Gender']).size()

df.groupby(['Gender','GenderGroup']).size()

```
df.groupby(['Gender']).size()
```

```
df.groupby(['Gender','GenderGroup']).size()
```

```
Gender  GenderGroup
F       1              26
M       2              26
dtype: int64
```

This output indicates that we have two types of combinations.

- Case 1: Gender = F & Gender Group = 1
- Case 2: Gender = M & GenderGroup = 2.

This validates our initial assumption that these two fields essentially portray the same information.

Data Cleaning: handle with missing data

Before getting started to work with your data, it's a good practice to observe it thoroughly to identify missing values and handle them accordingly.

When reading a dataset using Pandas, there is a set of values including 'NA', 'NULL', and 'NaN' that are taken by default to represent a missing value. The full list of default missing value codes is in the '`read_csv`' documentation here. This document also explains how to change the way that '`read_csv`' decides whether a variable's value is missing.

Pandas has functions called `isnull` and `notnull` that can be used to identify where the missing and non-missing values are located in a data frame.

Below we use these functions to count the number of missing and non-missing values in each variable of the datasetr.

```
df.isnull()
df.notnull()
```

```
      ID    Age  Gender  GenderGroup  Glasses  GlassesGroup  Height
Wingspan  \
0   True   True    True         True     True          True    True
True
1   True   True    True         True     True          True    True
True
2   True   True    True         True     True          True    True
True
3   True   True    True         True     True          True    True
True
4   True   True    True         True     True          True    True
True
5   True   True    True         True     True          True    True
True
6   True   True    True         True     True          True    True
True
7   True   True    True         True     True          True    True
True
8   True   True    True         True     True          True    True
True
9   True   True    True         True     True          True    True
True
10  True   True    True         True     True          True    True
True
11  True   True    True         True     True          True    True
True
```

12	True	True	True	True	True	True	True	True
13	True	True	True	True	True	True	True	True
14	True	True	True	True	True	True	True	True
15	True	True	True	True	True	True	True	True
16	True	True	True	True	True	True	True	True
17	True	True	True	True	True	True	True	True
18	True	True	True	True	True	True	True	True
19	True	True	True	True	True	True	True	True
20	True	True	True	True	True	True	True	True
21	True	True	True	True	True	True	True	True
22	True	True	True	True	True	True	True	True
23	True	True	True	True	True	True	True	True
24	True	True	True	True	True	True	True	True
25	True	True	True	True	True	True	True	True
26	True	True	True	True	True	True	True	True
27	True	True	True	True	True	True	True	True
28	True	True	True	True	True	True	True	True
29	True	True	True	True	True	True	True	True
30	True	True	True	True	True	True	True	True
31	True	True	True	True	True	True	True	True
32	True	True	True	True	True	True	True	True
33	True	True	True	True	True	True	True	True
34	True	True	True	True	True	True	True	True
35	True	True	True	True	True	True	True	True
36	True	True	True	True	True	True	True	True

```
37  True   True    True         True     True          True    True
True
38  True   True    True         True     True          True    True
True
39  True   True    True         True     True          True    True
True
40  True   True    True         True     True          True    True
True
41  True   True    True         True     True          True    True
True
42  True   True    True         True     True          True    True
True
43  True   True    True         True     True          True    True
True
44  True   True    True         True     True          True    True
True
45  True   True    True         True     True          True    True
True
46  True   True    True         True     True          True    True
True
47  True   True    True         True     True          True    True
True
48  True   True    True         True     True          True    True
True
49  True   True    True         True     True          True    True
False
50  True  False    True         True     True          True    True
True
51  True   True    True         True     True          True   False
True

    CWDistance  Complete  CompleteGroup  Score
0         True      True           True   True
1         True      True           True   True
2         True      True           True   True
3         True      True           True   True
4         True      True           True   True
5         True      True           True   True
6         True      True           True   True
7         True      True           True   True
8         True      True           True   True
9         True      True           True   True
10        True      True           True   True
11        True      True           True   True
12        True      True           True   True
13        True      True           True   True
14        True      True           True   True
15        True      True           True   True
16        True      True           True   True
17        True      True           True   True
```

```
18        True      True           True   True
19        True      True           True   True
20        True      True           True   True
21        True      True           True   True
22        True      True           True   True
23        True      True           True   True
24        True      True           True   True
25        True      True           True   True
26        True      True           True   True
27        True      True           True   True
28        True      True           True   True
29        True      True           True   True
30        True      True           True   True
31        True      True           True   True
32        True      True           True   True
33        True      True           True   True
34        True      True           True   True
35        True      True           True   True
36        True      True           True   True
37        True      True           True   True
38        True      True           True   True
39        True      True           True   True
40        True      True           True   True
41        True      True           True   True
42        True      True           True   True
43        True      True           True   True
44        True      True           True   True
45        True      True           True   True
46        True      True           True   True
47        True      True           True   True
48        True      True           True   True
49        True      True           True   True
50        True      True          False   True
51        True      True           True   True
```

Unfortunately, our output indicates that some of our columns contain missing values so we are no able to continue on doing analysis with those colums

```
df.notnull().sum()
```

```
ID               52
Age              51
Gender           52
GenderGroup      52
Glasses          52
GlassesGroup     52
Height           51
Wingspan         51
CWDistance       52
Complete         52
CompleteGroup    51
```

```
Score            52
dtype: int64
```

```
df.isnull().sum()
```

```
ID               0
Age              1
Gender           0
GenderGroup      0
Glasses          0
GlassesGroup     0
Height           1
Wingspan         1
CWDistance       0
Complete         0
CompleteGroup    1
Score            0
dtype: int64
```

Now we use these functions to count the number of missing and non-missing values in a single variable in the dataset

print(df.Height.notnull().sum())

print(pd.isnull(df.Height).sum())

```
print( df.Height.notnull().sum() )

print( pd.isnull(df.Height).sum() )
```

```
51
1
```

```
# Extract all non-missing values of one of the columns into a new
variable
x = df.Age.dropna().describe()
x.describe()
```

```
count     8.000000
mean     30.645922
std      16.044470
min       5.755611
25%      24.250000
50%      27.705882
75%      35.250000
max      56.000000
Name: Age, dtype: float64
```

Add and eliminate columns

In some cases it is useful to create or eiminate new columns

```
df.head()
```

```
   ID   Age Gender  GenderGroup Glasses  GlassesGroup  Height
Wingspan  \
0   1  56.0      F            1       Y             1    62.0
61.0
1   2  26.0      F            1       Y             1    62.0
60.0
2   3  33.0      F            1       Y             1    66.0
64.0
3   4  39.0      F            1       N             0    64.0
63.0
4   5  27.0      M            2       N             0    73.0
75.0

   CWDistance Complete  CompleteGroup  Score
0          79        Y            1.0      7
1          70        Y            1.0      8
2          85        Y            1.0      7
3          87        Y            1.0     10
4          72        N            0.0      4
```

```
# Add a new column with new data
# Create a column data
NewColumnData = df.Wingspan/df.CWDistance

# Insert that column in the data frame
df.insert(12, "ColumnInserted", NewColumnData, True)

df.head()
```

```
   ID   Age Gender  GenderGroup Glasses  GlassesGroup  Height
Wingspan  \
0   1  56.0      F            1       Y             1    62.0
61.0
1   2  26.0      F            1       Y             1    62.0
60.0
2   3  33.0      F            1       Y             1    66.0
64.0
3   4  39.0      F            1       N             0    64.0
63.0
4   5  27.0      M            2       N             0    73.0
75.0

   CWDistance Complete  CompleteGroup  Score  ColumnInserted
0          79        Y            1.0      7        0.772152
1          70        Y            1.0      8        0.857143
2          85        Y            1.0      7        0.752941
3          87        Y            1.0     10        0.724138
4          72        N            0.0      4        1.041667
```

```
# # Eliminate inserted column
df.drop("ColumnInserted", axis=1, inplace = True)
#df.drop(columns=['ColumnInserted'], inplace = True)
# # Remove three columns as index base
#df.drop(df.columns[[12]], axis = 1, inplace = True)
#
df.head()
```

```
   ID   Age Gender  GenderGroup Glasses  GlassesGroup  Height
Wingspan  \
0   1  56.0      F            1       Y             1    62.0
61.0
1   2  26.0      F            1       Y             1    62.0
60.0
2   3  33.0      F            1       Y             1    66.0
64.0
3   4  39.0      F            1       N             0    64.0
63.0
4   5  27.0      M            2       N             0    73.0
75.0

   CWDistance Complete  CompleteGroup  Score
0          79        Y            1.0      7
1          70        Y            1.0      8
2          85        Y            1.0      7
3          87        Y            1.0     10
4          72        N            0.0      4
```

```
# # Add new column derived from existing columns
#
# # The new column is a function of another column
df["AgeInMonths"] = df["Age"] * 12
#
df.head()
```

```
   ID   Age Gender  GenderGroup Glasses  GlassesGroup  Height
Wingspan  \
0   1  56.0      F            1       Y             1    62.0
61.0
1   2  26.0      F            1       Y             1    62.0
60.0
2   3  33.0      F            1       Y             1    66.0
64.0
3   4  39.0      F            1       N             0    64.0
63.0
4   5  27.0      M            2       N             0    73.0
75.0

   CWDistance Complete  CompleteGroup  Score  AgeInMonths
0          79        Y            1.0      7        672.0
1          70        Y            1.0      8        312.0
```

```
2          85        Y            1.0      7        396.0
3          87        Y            1.0     10        468.0
4          72        N            0.0      4        324.0
```

```
# # Eliminate inserted column
df.drop("AgeInMonths", axis=1, inplace = True)
#
df.head()
```

```
   ID   Age Gender  GenderGroup Glasses  GlassesGroup  Height
Wingspan  \
0   1  56.0      F            1       Y             1    62.0
61.0
1   2  26.0      F            1       Y             1    62.0
60.0
2   3  33.0      F            1       Y             1    66.0
64.0
3   4  39.0      F            1       N             0    64.0
63.0
4   5  27.0      M            2       N             0    73.0
75.0

   CWDistance Complete  CompleteGroup  Score
0          79        Y            1.0      7
1          70        Y            1.0      8
2          85        Y            1.0      7
3          87        Y            1.0     10
4          72        N            0.0      4
```

```
# Add a new column with text labels reflecting the code's meaning

df["GenderGroupNew"] = df.GenderGroup.replace({1: "Female", 2:
"Male"})

# Show the first 5 rows of the created data frame
df.head()
```

```
   ID   Age Gender  GenderGroup Glasses  GlassesGroup  Height
Wingspan  \
0   1  56.0      F            1       Y             1    62.0
61.0
1   2  26.0      F            1       Y             1    62.0
60.0
2   3  33.0      F            1       Y             1    66.0
64.0
3   4  39.0      F            1       N             0    64.0
63.0
4   5  27.0      M            2       N             0    73.0
75.0

   CWDistance Complete  CompleteGroup  Score GenderGroupNew
```

```
0          79        Y            1.0      7         Female
1          70        Y            1.0      8         Female
2          85        Y            1.0      7         Female
3          87        Y            1.0     10         Female
4          72        N            0.0      4           Male
```

```
## Eliminate inserted column
df.drop("GenderGroupNew", axis=1, inplace = True)
#df.drop(['GenderGroupNew'],vaxis='columns',vinplace=True)
df.head()
```

```
   ID   Age Gender  GenderGroup Glasses  GlassesGroup  Height
Wingspan  \
0   1  56.0      F            1       Y             1    62.0
61.0
1   2  26.0      F            1       Y             1    62.0
60.0
2   3  33.0      F            1       Y             1    66.0
64.0
3   4  39.0      F            1       N             0    64.0
63.0
4   5  27.0      M            2       N             0    73.0
75.0

   CWDistance Complete  CompleteGroup  Score
0          79        Y            1.0      7
1          70        Y            1.0      8
2          85        Y            1.0      7
3          87        Y            1.0     10
4          72        N            0.0      4
```

```
## Add a new column with strata based on these cut points
#
## Create a column data
NewColumnData = df.Age/df.Age
#
## Insert that column in the data frame
df.insert(1, "ColumnStrata", NewColumnData, True)
#
df["ColumnStrata"] = pd.cut(df.Height, [60., 63., 66., 69., 72., 75.,
78.])
#
## Show the first 5 rows of the created data frame
df.head()
```

```
   ID  ColumnStrata   Age Gender  GenderGroup Glasses  GlassesGroup
Height  \
0   1  (60.0, 63.0]  56.0      F            1       Y             1
62.0
```

```
1   2  (60.0, 63.0]  26.0      F            1       Y             1
62.0
2   3  (63.0, 66.0]  33.0      F            1       Y             1
66.0
3   4  (63.0, 66.0]  39.0      F            1       N             0
64.0
4   5  (72.0, 75.0]  27.0      M            2       N             0
73.0
```

```
   Wingspan  CWDistance Complete  CompleteGroup  Score
0      61.0          79        Y            1.0      7
1      60.0          70        Y            1.0      8
2      64.0          85        Y            1.0      7
3      63.0          87        Y            1.0     10
4      75.0          72        N            0.0      4
```

```
## Eliminate inserted column
df.drop("ColumnStrata", axis=1, inplace = True)
#
df.head()
```

```
   ID   Age Gender  GenderGroup Glasses  GlassesGroup  Height
Wingspan  \
0   1  56.0      F            1       Y             1    62.0
61.0
1   2  26.0      F            1       Y             1    62.0
60.0
2   3  33.0      F            1       Y             1    66.0
64.0
3   4  39.0      F            1       N             0    64.0
63.0
4   5  27.0      M            2       N             0    73.0
75.0
```

```
   CWDistance Complete  CompleteGroup  Score
0          79        Y            1.0      7
1          70        Y            1.0      8
2          85        Y            1.0      7
3          87        Y            1.0     10
4          72        N            0.0      4
```

```
# Drop several "unused" columns
#vars = ["ID", "GenderGroup", "GlassesGroup", "CompleteGroup"]
#df.drop(vars, axis=1, inplace = True)
df.head()
```

```
   ID   Age Gender  GenderGroup Glasses  GlassesGroup  Height
Wingspan  \
0   1  56.0      F            1       Y             1    62.0
```

```
61.0
1   2  26.0      F            1       Y             1    62.0
60.0
2   3  33.0      F            1       Y             1    66.0
64.0
3   4  39.0      F            1       N             0    64.0
63.0
4   5  27.0      M            2       N             0    73.0
75.0
```

	CWDistance	Complete	CompleteGroup	Score
0	79	Y	1.0	7
1	70	Y	1.0	8
2	85	Y	1.0	7
3	87	Y	1.0	10
4	72	N	0.0	4

Add and eliminate rows

In some cases it is requiered to add new observations (rows) to the data set

```
# Print tail

df.loc[len(df.index)] = [26, 24, 'F', 1, 'Y', 1, 66, 'NaN', 68, 'N',
0, 3]
#
df.tail()
```

```
---------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
c:\Users\vanec\Desktop\6to semestre\Analitica\TC1002S\
NotebooksStudents\A01634064\A2_DataManagment_Cartwheel_EMPTY.ipynb
Celda 74 in 1
----> <a href='vscode-notebook-cell:/c%3A/Users/vanec/Desktop/6to
%20semestre/Analitica/TC1002S/NotebooksStudents/A01634064/
A2_DataManagment_Cartwheel_EMPTY.ipynb#Y133sZmlsZQ%3D%3D?line=0'>1</a>
df.loc[len(df.index)] = [26, 24, 'F', 1, 'Y', 1, 66, 'NaN', 68, 'N',
0, 3]
      <a href='vscode-notebook-cell:/c%3A/Users/vanec/Desktop/6to
%20semestre/Analitica/TC1002S/NotebooksStudents/A01634064/
A2_DataManagment_Cartwheel_EMPTY.ipynb#Y133sZmlsZQ%3D%3D?line=1'>2</a>
#
      <a href='vscode-notebook-cell:/c%3A/Users/vanec/Desktop/6to
%20semestre/Analitica/TC1002S/NotebooksStudents/A01634064/
A2_DataManagment_Cartwheel_EMPTY.ipynb#Y133sZmlsZQ%3D%3D?line=2'>3</a>
```

```
df.tail()

File c:\Users\vanec\AppData\Local\Programs\Python\Python310\lib\site-
packages\pandas\core\indexing.py:818, in
_LocationIndexer.__setitem__(self, key, value)
    815 self._has_valid_setitem_indexer(key)
    817 iloc = self if self.name == "iloc" else self.obj.iloc
--> 818 iloc._setitem_with_indexer(indexer, value, self.name)

File c:\Users\vanec\AppData\Local\Programs\Python\Python310\lib\site-
packages\pandas\core\indexing.py:1785, in
_iLocIndexer._setitem_with_indexer(self, indexer, value, name)
   1782     indexer, missing = convert_missing_indexer(indexer)
   1784     if missing:
-> 1785         self._setitem_with_indexer_missing(indexer, value)
   1786         return
   1788 if name == "loc":
   1789     # must come after setting of missing

File c:\Users\vanec\AppData\Local\Programs\Python\Python310\lib\site-
packages\pandas\core\indexing.py:2160, in
_iLocIndexer._setitem_with_indexer_missing(self, indexer, value)
   2157     if is_list_like_indexer(value):
   2158         # must have conforming columns
   2159         if len(value) != len(self.obj.columns):
-> 2160             raise ValueError("cannot set a row with mismatched
columns")
   2162     value = Series(value, index=self.obj.columns,
name=indexer)
   2164 if not len(self.obj):
   2165     # We will ignore the existing dtypes instead of using
   2166     #  internals.concat logic

ValueError: cannot set a row with mismatched columns
```

```
## Eliminate inserted row
#df.drop([28], inplace = True )
#
#df.tail()
```

Cleaning your data: drop out unused columns and/or drop out rows with any missing values

```
# Drop unused columns
#vars = ["ID", "GenderGroup", "GlassesGroup", "CompleteGroup"]
#df.drop(vars, axis=1, inplace = True)
```

```
#vars = ["Age", "Gender", "Glasses", "Height", "Wingspan",
"CWDistance", "Complete", "Score"]
#df = df[vars]

# Drop rows with any missing values
#df = df.dropna()

# Drop unused columns and drop rows with any missing values
vars = ["Age", "Gender", "Glasses", "Height", "Wingspan",
"CWDistance", "Complete", "Score"]
df2 = df[vars].dropna()

df.head()
```

	Age	Gender	Glasses	Height	Wingspan	CWDistance	Complete	Score
0	56.0	F	Y	62.0	61.0	79	Y	7
1	26.0	F	Y	62.0	60.0	70	Y	8
2	33.0	F	Y	66.0	64.0	85	Y	7
3	39.0	F	N	64.0	63.0	87	Y	10
4	27.0	M	N	73.0	75.0	72	N	4

Final remarks

- The understanding of your dataset is essential
  - Number of observations
  - Variables
  - Data types: numerical or categorial
  - What are my variables of interest
- There are several ways to do the same thing
- Cleaning your dataset (dropping out rows with any missing values) is a good practice
- The **Pandas** library provides fancy, high-performance, easy-to-use data structures and data analysis tools

Activity: work with the iris dataset

Repeat this tutorial with the iris data set and respond to the following inquiries

1. Calculate the statistical summary for each quantitative variables. Explain the results
   - Identify the name of each column
   - Identify the type of each column
   - Minimum, maximum, mean, average, median, standar deviation

2. Are there missing data? If so, create a new dataset containing only the rows with the non-missing data
3. Create a new dataset containing only the petal width and length and the type of Flower
4. Create a new dataset containing only the setal width and length and the type of Flower
5. Create a new dataset containing the setal width and length and the type of Flower encoded as a categorical numerical column

```
dfIris= pd.read_csv("datasets/iris/iris.csv")
```

```
dfIris.info()
dfIris.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149 entries, 0 to 148
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   5.1          149 non-null    float64
 1   3.5          149 non-null    float64
 2   1.4          149 non-null    float64
 3   0.2          149 non-null    float64
 4   Iris-setosa  149 non-null    object 
dtypes: float64(4), object(1)
memory usage: 5.9+ KB
```

```
              5.1         3.5         1.4         0.2
count  149.000000  149.000000  149.000000  149.000000
mean     5.848322    3.054362    3.773826    1.206040
std      0.828594    0.435810    1.760543    0.760354
min      4.300000    2.000000    1.000000    0.100000
25%      5.100000    2.800000    1.600000    0.300000
50%      5.800000    3.000000    4.400000    1.300000
75%      6.400000    3.300000    5.100000    1.800000
max      7.900000    4.400000    6.900000    2.500000
```

```
dfIris.dropna()
```

```
     5.1  3.5  1.4  0.2     Iris-setosa
0    4.9  3.0  1.4  0.2     Iris-setosa
1    4.7  3.2  1.3  0.2     Iris-setosa
2    4.6  3.1  1.5  0.2     Iris-setosa
3    5.0  3.6  1.4  0.2     Iris-setosa
4    5.4  3.9  1.7  0.4     Iris-setosa
..   ...  ...  ...  ...             ...
144  6.7  3.0  5.2  2.3  Iris-virginica
145  6.3  2.5  5.0  1.9  Iris-virginica
```

```
146  6.5  3.0  5.2  2.0  Iris-virginica
147  6.2  3.4  5.4  2.3  Iris-virginica
148  5.9  3.0  5.1  1.8  Iris-virginica

[149 rows x 5 columns]
```

```
# Add a header row to the data frame "PetalWidth", "PetalLength",
"SepalWidth", "SepalLength", "Species"
dfIris.columns = ["PetalWidth", "PetalLength", "SepalWidth",
"SepalLength", "Species"]
dfIris.head()
```

```
   PetalWidth  PetalLength  SepalWidth  SepalLength      Species
0         4.9          3.0         1.4          0.2  Iris-setosa
1         4.7          3.2         1.3          0.2  Iris-setosa
2         4.6          3.1         1.5          0.2  Iris-setosa
3         5.0          3.6         1.4          0.2  Iris-setosa
4         5.4          3.9         1.7          0.4  Iris-setosa
```

```
# Create a new dataset containing only the petal width and length and
the type of Flower
dfIris2 = dfIris[["PetalWidth", "PetalLength", "Species"]]
dfIris2
```

```
     PetalWidth  PetalLength         Species
0           4.9          3.0     Iris-setosa
1           4.7          3.2     Iris-setosa
2           4.6          3.1     Iris-setosa
3           5.0          3.6     Iris-setosa
4           5.4          3.9     Iris-setosa
..          ...          ...             ...
144         6.7          3.0  Iris-virginica
145         6.3          2.5  Iris-virginica
146         6.5          3.0  Iris-virginica
147         6.2          3.4  Iris-virginica
148         5.9          3.0  Iris-virginica

[149 rows x 3 columns]
```

```
#Create a new dataset containing only the setal width and length and
the type of Flower
dfIris3 = dfIris[["SepalWidth", "SepalLength", "Species"]]
dfIris3
```

```
     SepalWidth  SepalLength         Species
0           1.4          0.2     Iris-setosa
1           1.3          0.2     Iris-setosa
2           1.5          0.2     Iris-setosa
3           1.4          0.2     Iris-setosa
4           1.7          0.4     Iris-setosa
..          ...          ...             ...
```

```
144          5.2           2.3  Iris-virginica
145          5.0           1.9  Iris-virginica
146          5.2           2.0  Iris-virginica
147          5.4           2.3  Iris-virginica
148          5.1           1.8  Iris-virginica

[149 rows x 3 columns]
```

```
# Create a new dataset containing the setal width and length and the
type of Flower encoded as a categorical numerical column
dfIris["SpeciesNew"] = dfIris.Species.replace({"Iris-setosa":1 ,
"Iris-virginica": 2 , "Iris-versicolor": 3})

dfIris4 = dfIris[["SepalWidth", "SepalLength", "SpeciesNew"]]
dfIris4
```

```
     SepalWidth  SepalLength  SpeciesNew
0            1.4          0.2           1
1            1.3          0.2           1
2            1.5          0.2           1
3            1.4          0.2           1
4            1.7          0.4           1
..           ...          ...         ...
144          5.2          2.3           2
145          5.0          1.9           2
146          5.2          2.0           2
147          5.4          2.3           2
148          5.1          1.8           2

[149 rows x 3 columns]
```