Introducing the dataset

Data Set Information: A dataset containing the information and other features diamonds.

Number of Attributes: 10

Feature Information: A data frame with rows and variables:

```
import numpy as np
import pandas as pd
import os
```

Loading the data

```
import pandas as pd
from google.colab import files
```

```
uploaded=files.upload()
```

Choose Files IRIS.csv

- **IRIS.csv**(application/vnd.ms-excel) - 3861 bytes, last modified: 2/11/2021 - 100% done

```
Saving IRIS.csv to IRIS.csv
```

```
uploaded
```

```
{'IRIS.csv': b'\xef\xbb\xbfsepal_length,sepal_width,petal_length,petal_width,species\n5
```

```
data = pd.read_csv('IRIS.csv')
data.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
data.info()
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

```
import io
iris = pd.read_csv(io.StringIO(uploaded['IRIS.csv'].decode('utf-8')))
print(iris)
```

```
     sepal_length  sepal_width  petal_length  petal_width    species
0             5.1          3.5           1.4          0.2     setosa
1             4.9          3.0           1.4          0.2     setosa
2             4.7          3.2           1.3          0.2     setosa
3             4.6          3.1           1.5          0.2     setosa
4             5.0          3.6           1.4          0.2     setosa
..            ...          ...           ...          ...        ...
145           6.7          3.0           5.2          2.3  virginica
146           6.3          2.5           5.0          1.9  virginica
147           6.5          3.0           5.2          2.0  virginica
148           6.2          3.4           5.4          2.3  virginica
149           5.9          3.0           5.1          1.8  virginica

[150 rows x 5 columns]
```

Untuk melihat ukuran data csv (baris, kolom), pada data ini terdapat 150 baris 5 kolom dari dataset

```
iris.shape
```

```
(150, 5)
```

Menampilkan data beberapa baris yang awal

```
iris.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa

Displaying the number of rows randomly.

In sample() function, it will also display the rows according to arguments given, but it will display the rows randomly.

```
data.sample(10)
```

	sepal_length	sepal_width	petal_length	petal_width	species
101	5.8	2.7	5.1	1.9	virginica
141	6.9	3.1	5.1	2.3	virginica
93	5.0	2.3	3.3	1.0	versicolor
137	6.4	3.1	5.5	1.8	virginica
52	6.9	3.1	4.9	1.5	versicolor
24	4.8	3.4	1.9	0.2	setosa
103	6.3	2.9	5.6	1.8	virginica
76	6.8	2.8	4.8	1.4	versicolor
131	7.9	3.8	6.4	2.0	virginica
118	7.7	2.6	6.9	2.3	virginica

Code: Displaying the number of columns and names of the columns.

The column() function prints all the columns of the dataset in a list form.

```
data.columns
```

```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

Display dataset (row, colomns)

```
data.shape
```

```
(150, 5)
```

Display the whole dataset using print 'nama file' ==> print data or print iris

```
print(data)
```

```
     sepal_length  sepal_width  petal_length  petal_width    species
0             5.1          3.5           1.4          0.2     setosa
1             4.9          3.0           1.4          0.2     setosa
2             4.7          3.2           1.3          0.2     setosa
3             4.6          3.1           1.5          0.2     setosa
4             5.0          3.6           1.4          0.2     setosa
..            ...          ...           ...          ...        ...
145           6.7          3.0           5.2          2.3  virginica
146           6.3          2.5           5.0          1.9  virginica
147           6.5          3.0           5.2          2.0  virginica
148           6.2          3.4           5.4          2.3  virginica
149           5.9          3.0           5.1          1.8  virginica

[150 rows x 5 columns]
```

```
print(iris)
```

```
     sepal_length  sepal_width  petal_length  petal_width    species
0             5.1          3.5           1.4          0.2     setosa
1             4.9          3.0           1.4          0.2     setosa
2             4.7          3.2           1.3          0.2     setosa
3             4.6          3.1           1.5          0.2     setosa
4             5.0          3.6           1.4          0.2     setosa
..            ...          ...           ...          ...        ...
145           6.7          3.0           5.2          2.3  virginica
146           6.3          2.5           5.0          1.9  virginica
147           6.5          3.0           5.2          2.0  virginica
148           6.2          3.4           5.4          2.3  virginica
149           5.9          3.0           5.1          1.8  virginica

[150 rows x 5 columns]
```

Slicing the rows.

Slicing means if you want to print or work upon a particular group of lines that is from 10th row to 20th row.

```
#data[start:end]
#start is inclusive whereas end is exclusive
print(data[10:21])
# it will print the rows from 10 to 20.

# you can also save it in a variable for further use in analysis
sliced_data=data[10:21]
print(sliced_data)
```

```
    sepal_length  sepal_width  petal_length  petal_width species
10           5.4          3.7           1.5          0.2  setosa
11           4.8          3.4           1.6          0.2  setosa
```

```
    12           4.8          3.0           1.4          0.1  setosa
    13           4.3          3.0           1.1          0.1  setosa
    14           5.8          4.0           1.2          0.2  setosa
    15           5.7          4.4           1.5          0.4  setosa
    16           5.4          3.9           1.3          0.4  setosa
    17           5.1          3.5           1.4          0.3  setosa
    18           5.7          3.8           1.7          0.3  setosa
    19           5.1          3.8           1.5          0.3  setosa
    20           5.4          3.4           1.7          0.2  setosa
        sepal_length  sepal_width  petal_length  petal_width species
    10           5.4          3.7           1.5          0.2  setosa
    11           4.8          3.4           1.6          0.2  setosa
    12           4.8          3.0           1.4          0.1  setosa
    13           4.3          3.0           1.1          0.1  setosa
    14           5.8          4.0           1.2          0.2  setosa
    15           5.7          4.4           1.5          0.4  setosa
    16           5.4          3.9           1.3          0.4  setosa
    17           5.1          3.5           1.4          0.3  setosa
    18           5.7          3.8           1.7          0.3  setosa
    19           5.1          3.8           1.5          0.3  setosa
    20           5.4          3.4           1.7          0.2  setosa
```

```
#here in the case of Iris dataset
#we will save it in a another variable named "specific_data"

specific_data1=data[["sepal_length","sepal_width","species"]]
#data[["column_name 2","column_name3","column_name6"]]

#now we will print the first 10 columns of the specific_data dataframe.
print(specific_data1.head(10))
```

```
       sepal_length  sepal_width species
    0           5.1          3.5  setosa
    1           4.9          3.0  setosa
    2           4.7          3.2  setosa
    3           4.6          3.1  setosa
    4           5.0          3.6  setosa
    5           5.4          3.9  setosa
    6           4.6          3.4  setosa
    7           5.0          3.4  setosa
    8           4.4          2.9  setosa
    9           4.9          3.1  setosa
```

Display spesific data use colomns 2,3 and 6 the which sliced data 10:21. It will print the rows from 10 to 20.

```
#here in the case of Iris dataset
#we will save it in a another variable named "specific_data"

specific_data2=data[["sepal_length","sepal_width","species"]]
#data[["column_name 2","column_name3","column_name6"]]
```

```
#now we will print the first 3 columns of the specific_data dataframe and the rows from 10 an
print(specific_data2[10:21])

    sepal_length  sepal_width species
10           5.4          3.7  setosa
11           4.8          3.4  setosa
12           4.8          3.0  setosa
13           4.3          3.0  setosa
14           5.8          4.0  setosa
15           5.7          4.4  setosa
16           5.4          3.9  setosa
17           5.1          3.5  setosa
18           5.7          3.8  setosa
19           5.1          3.8  setosa
20           5.4          3.4  setosa
```

Checking summary statistiscs for numerical columns

Menampilkan nilai baris 1 adalah jumlah data, baris 2 rata-rata, baris 3 standar deviasi, baris 4 nilai minimum

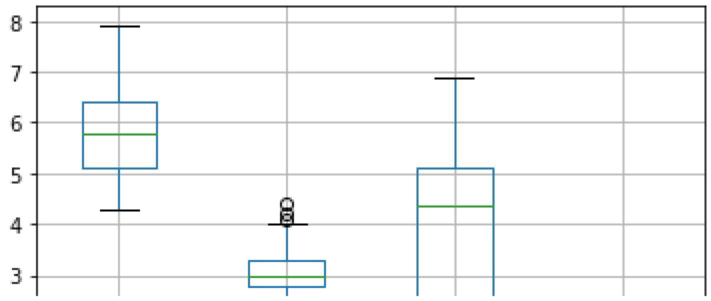
```
data.describe()
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Boxplot

```
iris.boxplot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7efca13c7cf8>
```



Filtering:Displaying the specific rows using “iloc” and “loc” functions.

1. The “loc” functions use the index name of the row to display the particular row of the dataset.
2. The “iloc” functions use the index integer of the row, which gives complete information about the row.

```
#1. Use iloc

data.iloc[5]
```

```
sepal_length       5.4
sepal_width        3.9
petal_length       1.7
petal_width        0.4
species         setosa
Name: 5, dtype: object
```

```
#2. Use loc which it will display records only with species "setosa".
data.loc[data["species"] == "setosa"]
```

16	5.4	3.9	1.3	0.4	setosa
17	5.1	3.5	1.4	0.3	setosa
18	5.7	3.8	1.7	0.3	setosa
19	5.1	3.8	1.5	0.3	setosa
20	5.4	3.4	1.7	0.2	setosa
21	5.1	3.7	1.5	0.4	setosa
22	4.6	3.6	1.0	0.2	setosa
23	5.1	3.3	1.7	0.5	setosa
24	4.8	3.4	1.9	0.2	setosa
25	5.0	3.0	1.6	0.2	setosa
26	5.0	3.4	1.6	0.4	setosa
27	5.2	3.5	1.5	0.2	setosa
28	5.2	3.4	1.4	0.2	setosa
29	4.7	3.2	1.6	0.2	setosa
30	4.8	3.1	1.6	0.2	setosa
31	5.4	3.4	1.5	0.4	setosa
32	5.2	4.1	1.5	0.1	setosa
33	5.5	4.2	1.4	0.2	setosa
34	4.9	3.1	1.5	0.1	setosa
35	5.0	3.2	1.2	0.2	setosa
36	5.5	3.5	1.3	0.2	setosa
37	4.9	3.1	1.5	0.1	setosa
38	4.4	3.0	1.3	0.2	setosa
39	5.1	3.4	1.5	0.2	setosa
40	5.0	3.5	1.3	0.3	setosa
41	4.5	2.3	1.3	0.3	setosa
42	4.4	3.2	1.3	0.2	setosa
43	5.0	3.5	1.6	0.6	setosa
44	5.1	3.8	1.9	0.4	setosa
45	4.8	3.0	1.4	0.3	setosa
46	5.1	3.8	1.6	0.2	setosa

47	4.6	3.2	1.4	0.2	setosa
48	5.3	3.7	1.5	0.2	setosa
49	5.0	3.3	1.4	0.2	setosa

Display specific row use value threshold. Example sepal_length have valued high tha 5 (sepal length >5). dataset display row the value

```
data.loc[data["sepal_length"]  > 5]
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
10	5.4	3.7	1.5	0.2	setosa
14	5.8	4.0	1.2	0.2	setosa
15	5.7	4.4	1.5	0.4	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

118 rows × 5 columns

Display specific row use spesific value threshold that are two condition. Example sepal_length have valued high than 5 (sepal length >5) and sepal_width > 2. dataset display row the value

```
data.loc[(data["sepal_length"]  > 5)|(data["sepal_width"] > 2)]
data_loc1=data.loc[(data["sepal_length"]  > 5)|(data["sepal_width"] > 2)]
display(data_loc1)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica

```
# cek this logical and compare function loc use 2 condition

data.loc[(data["sepal_length"]  > 5)|(data["petal_width"] < 2)]
data_loc2=data.loc[(data["sepal_length"]  > 5)|(data["petal_width"] < 2)]
display(data_loc2)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

```
# cek this logical and compare function loc use 2 condition

data.loc[(data["sepal_length"]  < 5)|(data["sepal_width"] > 2)]
data_loc3=data.loc[(data["sepal_length"]  < 5)|(data["sepal_width"] > 2)]
display(data_loc3)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

Counting the number of counts of unique values using “value_counts()”.

The value_counts() function, counts the number of times a particular instance or data has occurred

```
#In this dataset we will work on the Species column, it will count number of times a particul
data["species"].value_counts()
#it will display in descending order.
```

```
setosa        50
versicolor    50
virginica     50
Name: species, dtype: int64
```

```
#In this dataset we will work on the Species column, it will count number of times a particul
data["sepal_length"].value_counts()
#it will display in descending order.
```

```
5.0    10
6.3     9
5.1     9
6.7     8
5.7     8
5.5     7
5.8     7
6.4     7
6.0     6
4.9     6
6.1     6
5.4     6
5.6     6
6.5     5
4.8     5
```

```
7.7     4
6.9     4
5.2     4
6.2     4
4.6     4
7.2     3
6.8     3
4.4     3
5.9     3
6.6     2
4.7     2
7.6     1
7.4     1
4.3     1
7.9     1
7.3     1
7.0     1
4.5     1
5.3     1
7.1     1
Name: sepal_length, dtype: int64
```

```
#In this dataset we will work on the Species column, it will count number of times a particul
data["sepal_width"].value_counts()
#it will display in descending order.
```

```
3.0    26
2.8    14
3.2    13
3.4    12
3.1    12
2.9    10
2.7     9
2.5     8
3.5     6
3.8     6
3.3     6
2.6     5
2.3     4
3.6     3
2.4     3
2.2     3
3.7     3
3.9     2
4.2     1
4.1     1
4.4     1
2.0     1
4.0     1
Name: sepal_width, dtype: int64
```

```
#In this dataset we will work on the Species column, it will count number of times a particul
data["petal_length"].value_counts()
#it will display in descending order.
```

```
1.5    14
1.4    12
5.1     8
4.5     8
1.3     7
1.6     7
5.6     6
4.0     5
4.9     5
4.7     5
4.8     4
1.7     4
4.4     4
4.2     4
5.0     4
4.1     3
5.5     3
4.6     3
6.1     3
5.7     3
3.9     3
5.8     3
1.2     2
1.9     2
6.7     2
3.5     2
5.9     2
6.0     2
5.4     2
5.3     2
3.3     2
4.3     2
5.2     2
6.3     1
1.1     1
6.4     1
3.6     1
3.7     1
3.0     1
3.8     1
6.6     1
6.9     1
1.0     1
Name: petal_length, dtype: int64
```

```
#In this dataset we will work on the Species column, it will count number of times a particul
data["petal_width"].value_counts()
#it will display in descending order.
```

```
0.2    28
1.3    13
1.5    12
1.8    12
1.4     8
2.3     8
```

```
1.0     7
0.3     7
0.4     7
0.1     6
2.0     6
2.1     6
1.2     5
1.9     5
1.6     4
2.5     3
2.2     3
2.4     3
1.1     3
1.7     2
0.6     1
0.5     1
Name: petal_width, dtype: int64
```

Calculating sum, mean and mode of a particular column.

We can also calculate the sum, mean, median, min and max value mode of any integer columns as I have done in the following code.

```
# data["column_name"].sum()

sum_data = data["sepal_length"].sum()
mean_data = data["sepal_length"].mean()
median_data = data["sepal_length"].median()
min_data=data["sepal_length"].min()
max_data=data["sepal_length"].max()

print("Sum:",sum_data, "\nMean:", mean_data, "\nMedian:",median_data, "\nMinimum:",min_data,
```

```
Sum: 876.5
Mean: 5.843333333333335
Median: 5.8
Minimum: 4.3
Maximum: 7.9
```

```
# data["column_name"].sum()

sum_data = data["sepal_width"].sum()
mean_data = data["sepal_width"].mean()
median_data = data["sepal_width"].median()
min_data=data["sepal_width"].min()
max_data=data["sepal_width"].max()

print("Sum:",sum_data, "\nMean:", mean_data, "\nMedian:",median_data, "\nMinimum:",min_data,
```

```
Sum: 458.1
Mean: 3.0540000000000007
Median: 3.0
```

```
     Minimum: 2.0
     Maximum: 4.4
```

Adding a column to the dataset.

If want to add a new column in our dataset, as we are doing any calculations or extracting some information from the dataset, and if you want to save it a new column. This can be done by the following code by taking a case where we have added all integer values of all columns.

```
# For example, if we want to add a column let say "total_values",
# that means if you want to add all the integer value of that particular
# row and get total answer in the new column "total_values".
# first we will extract the columns which have integer values.
cols = data.columns

# it will print the list of column names.
print(cols)

# we will take that columns which have integer values.
cols = cols[1:5]

# we will save it in the new dataframe variable
data1 = data[cols]

# now adding new column "total_values" to dataframe data.
data["total_values"]=data1[cols].sum(axis=1)
# data_new=data["total_values"]
# here axis=1 means you are working in rows,
# whereas axis=0 means you are working in columns.

#print(data["total_values"])
```

```
     Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
            'species'],
           dtype='object')
```

Now we will highlight the maximum and minimum column-wise, row-wise, and the whole dataframe wise using Style.apply function. The Style.apply function passes each column or row of the dataframe depending upon the keyword argument axis. For column-wise use axis=0, row-wise use axis=1, and for the entire table at once use axis=None.

```
# we will here print only the top 10 rows of the dataset = data after "total_values",
# if you want to see the result of the whole dataset remove
#.head(10) from the below code

data.head(10).style.highlight_max(color='lightgreen', axis=0)
```

	sepal_length	sepal_width	petal_length	petal_width	species	total_values
0	5.100000	3.500000	1.400000	0.200000	setosa	5.100000
1	4.900000	3.000000	1.400000	0.200000	setosa	4.600000
2	4.700000	3.200000	1.300000	0.200000	setosa	4.700000
3	4.600000	3.100000	1.500000	0.200000	setosa	4.800000
4	5.000000	3.600000	1.400000	0.200000	setosa	5.200000
5	5.400000	3.900000	1.700000	0.400000	setosa	6.000000
6	4.600000	3.400000	1.400000	0.300000	setosa	5.100000
7	5.000000	3.400000	1.500000	0.200000	setosa	5.100000
8	4.400000	2.900000	1.400000	0.200000	setosa	4.500000

```
data.head(10).style.highlight_max(color='lightgreen', axis=1)
```

	sepal_length	sepal_width	petal_length	petal_width	species	total_values
0	5.100000	3.500000	1.400000	0.200000	setosa	5.100000
1	4.900000	3.000000	1.400000	0.200000	setosa	4.600000
2	4.700000	3.200000	1.300000	0.200000	setosa	4.700000
3	4.600000	3.100000	1.500000	0.200000	setosa	4.800000
4	5.000000	3.600000	1.400000	0.200000	setosa	5.200000
5	5.400000	3.900000	1.700000	0.400000	setosa	6.000000
6	4.600000	3.400000	1.400000	0.300000	setosa	5.100000
7	5.000000	3.400000	1.500000	0.200000	setosa	5.100000
8	4.400000	2.900000	1.400000	0.200000	setosa	4.500000
9	4.900000	3.100000	1.500000	0.100000	setosa	4.700000

```
data.head(10).style.highlight_max(color='lightgreen', axis=None)
```

	sepal_length	sepal_width	petal_length	petal_width	species	total_values
0	5.100000	3.500000	1.400000	0.200000	setosa	5.100000
1	4.900000	3.000000	1.400000	0.200000	setosa	4.600000
2	4.700000	3.200000	1.300000	0.200000	setosa	4.700000
3	4.600000	3.100000	1.500000	0.200000	setosa	4.800000
4	5.000000	3.600000	1.400000	0.200000	setosa	5.200000
5	5.400000	3.900000	1.700000	0.400000	setosa	6.000000
6	4.600000	3.400000	1.400000	0.300000	setosa	5.100000
7	5.000000	3.400000	1.500000	0.200000	setosa	5.100000
8	4.400000	2.900000	1.400000	0.200000	setosa	4.500000
9	4.900000	3.100000	1.500000	0.100000	setosa	4.700000

```
# we will here print only the top 10 rows of the dataset =iris ,
# if you want to see the result of the whole dataset remove
#.head(10) from the below code

iris.head(10).style.highlight_max(color='lightgreen', axis=0)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.100000	3.500000	1.400000	0.200000	setosa
1	4.900000	3.000000	1.400000	0.200000	setosa
2	4.700000	3.200000	1.300000	0.200000	setosa
3	4.600000	3.100000	1.500000	0.200000	setosa
4	5.000000	3.600000	1.400000	0.200000	setosa
5	5.400000	3.900000	1.700000	0.400000	setosa
6	4.600000	3.400000	1.400000	0.300000	setosa
7	5.000000	3.400000	1.500000	0.200000	setosa

```
iris.head(10).style.highlight_max(color='lightgreen', axis=1)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.100000	3.500000	1.400000	0.200000	setosa
1	4.900000	3.000000	1.400000	0.200000	setosa
2	4.700000	3.200000	1.300000	0.200000	setosa
3	4.600000	3.100000	1.500000	0.200000	setosa
4	5.000000	3.600000	1.400000	0.200000	setosa
5	5.400000	3.900000	1.700000	0.400000	setosa
6	4.600000	3.400000	1.400000	0.300000	setosa
7	5.000000	3.400000	1.500000	0.200000	setosa
8	4.400000	2.900000	1.400000	0.200000	setosa
9	4.900000	3.100000	1.500000	0.100000	setosa

```
iris.head(10).style.highlight_max(color='lightgreen', axis=None)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.100000	3.500000	1.400000	0.200000	setosa
1	4.900000	3.000000	1.400000	0.200000	setosa
2	4.700000	3.200000	1.300000	0.200000	setosa
3	4.600000	3.100000	1.500000	0.200000	setosa
4	5.000000	3.600000	1.400000	0.200000	setosa
5	5.400000	3.900000	1.700000	0.400000	setosa
6	4.600000	3.400000	1.400000	0.300000	setosa
7	5.000000	3.400000	1.500000	0.200000	setosa
8	4.400000	2.900000	1.400000	0.200000	setosa
9	4.900000	3.100000	1.500000	0.100000	setosa

Cleaning and detecting missing values

In this dataset, we will now try to find the missing values i.e NaN, which can occur due to several reasons.

```
data.isnull()
#if there is data is missing, it will display True else False.
```

	sepal_length	sepal_width	petal_length	petal_width	species	total_values
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
...	...	...	...	...	...	...
145	False	False	False	False	False	False
146	False	False	False	False	False	False
147	False	False	False	False	False	False
148	False	False	False	False	False	False
149	False	False	False	False	False	False

Summarizing the missing values.

We will display how many missing values are present in each column

```
data.sum =data.isnull()
print(data.sum)
```

```
     sepal_length  sepal_width  ...  species  total_values
0           False        False  ...    False         False
1           False        False  ...    False         False
2           False        False  ...    False         False
3           False        False  ...    False         False
4           False        False  ...    False         False
..            ...          ...  ...      ...           ...
145         False        False  ...    False         False
146         False        False  ...    False         False
147         False        False  ...    False         False
148         False        False  ...    False         False
149         False        False  ...    False         False

[150 rows x 6 columns]
```

Importing seaborn

The heatmap is a data visualisation technique which is used to analyse the dataset as colors in two dimensions. Basically it shows correlation between all numerical variables in the dataset. Heatmap is an attribute of the Seaborn library.

Pandas Dataframe Correlation: Pandas correlation is used to determine pairwise correlation of all the columns of the dataset. In datafram.corr(), the missing values are excluded and non-numeric columns are also ignored.

```
import seaborn as sns
```

```
data.corr(method='pearson')
```

	sepal_length	sepal_width	petal_length	petal_width	total_values
sepal_length	1.000000	-0.109369	0.871754	0.817954	0.893970
sepal_width	-0.109369	1.000000	-0.420516	-0.356544	-0.245361
petal_length	0.871754	-0.420516	1.000000	0.962757	0.979301
petal_width	0.817954	-0.356544	0.962757	1.000000	0.975264
total_values	0.893970	-0.245361	0.979301	0.975264	1.000000