# Pandas - Analyzing DataFrames

## Viewing the Data

One of the most used method for getting a quick overview of the DataFrame, is the head() method.

The head() method returns the headers and a specified number of rows, starting from the top.

### Example

Get a quick overview by printing the first 10 rows of the DataFrame:

import pandas as pd  
  
df = pd.read\_csv('data.csv')  
  
print(df.head(10))

****Note:**** if the number of rows is not specified, the head() method will return the top 5 rows.

### Example

Print the first 5 rows of the DataFrame:

import pandas as pd  
  
df = pd.read\_csv('data.csv')  
  
print(df.head())

There is also a tail() method for viewing the last rows of the DataFrame.

The tail() method returns the headers and a specified number of rows, starting from the bottom.

### Example

Print the last 5 rows of the DataFrame:

print(df.tail())

## Info About the Data

The DataFrames object has a method called info(), that gives you more information about the data set.

### Example

Print information about the data:

print(df.info())

## Result

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 169 entries, 0 to 168

Data columns (total 4 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Duration 169 non-null int64

1 Pulse 169 non-null int64

2 Maxpulse 169 non-null int64

3 Calories 164 non-null float64

dtypes: float64(1), int64(3)

memory usage: 5.4 KB

None

## Result Explained

The result tells us there are 169 rows and 4 columns:

RangeIndex: 169 entries, 0 to 168

Data columns (total 4 columns):

And the name of each column, with the data type:

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Duration 169 non-null int64

1 Pulse 169 non-null int64

2 Maxpulse 169 non-null int64

3 Calories 164 non-null float64

## Null Values

The info() method also tells us how many Non-Null values there are present in each column, and in our data set it seems like there are 164 of 169 Non-Null values in the "Calories" column.

Which means that there are 5 rows with no value at all, in the "Calories" column, for whatever reason.

Empty values, or Null values, can be bad when analyzing data, and you should consider removing rows with empty values. This is a step towards what is called cleaning data, and you will learn more about that in the next chapters.

# Pandas - Cleaning Data

## Data Cleaning

Data cleaning means fixing bad data in your data set.

Bad data could be:

* Empty cells
* Data in wrong format
* Wrong data
* Duplicates

In this tutorial you will learn how to deal with all of them.

## Our Data Set

In the next chapters we will use this data set:

Duration Date Pulse Maxpulse Calories

0 60 '2020/12/01' 110 130 409.1

1 60 '2020/12/02' 117 145 479.0

2 60 '2020/12/03' 103 135 340.0

3 45 '2020/12/04' 109 175 282.4

4 45 '2020/12/05' 117 148 406.0

5 60 '2020/12/06' 102 127 300.0

6 60 '2020/12/07' 110 136 374.0

7 450 '2020/12/08' 104 134 253.3

8 30 '2020/12/09' 109 133 195.1

9 60 '2020/12/10' 98 124 269.0

10 60 '2020/12/11' 103 147 329.3

11 60 '2020/12/12' 100 120 250.7

12 60 '2020/12/12' 100 120 250.7

13 60 '2020/12/13' 106 128 345.3

14 60 '2020/12/14' 104 132 379.3

15 60 '2020/12/15' 98 123 275.0

16 60 '2020/12/16' 98 120 215.2

17 60 '2020/12/17' 100 120 300.0

18 45 '2020/12/18' 90 112 NaN

19 60 '2020/12/19' 103 123 323.0

20 45 '2020/12/20' 97 125 243.0

21 60 '2020/12/21' 108 131 364.2

22 45 NaN 100 119 282.0

23 60 '2020/12/23' 130 101 300.0

24 45 '2020/12/24' 105 132 246.0

25 60 '2020/12/25' 102 126 334.5

26 60 2020/12/26 100 120 250.0

27 60 '2020/12/27' 92 118 241.0

28 60 '2020/12/28' 103 132 NaN

29 60 '2020/12/29' 100 132 280.0

30 60 '2020/12/30' 102 129 380.3

31 60 '2020/12/31' 92 115 243.0

The data set contains some empty cells ("Date" in row 22, and "Calories" in row 18 and 28).

The data set contains wrong format ("Date" in row 26).

The data set contains wrong data ("Duration" in row 7).

The data set contains duplicates (row 11 and 12).

# Pandas - Cleaning Empty Cells

## Empty Cells

Empty cells can potentially give you a wrong result when you analyze data.

## Remove Rows

One way to deal with empty cells is to remove rows that contain empty cells.

This is usually OK, since data sets can be very big, and removing a few rows will not have a big impact on the result.

### Example

Return a new Data Frame with no empty cells:

import pandas as pd  
  
df = pd.read\_csv('data.csv')  
  
new\_df = df.dropna()

****Note:**** By default, the dropna() method returns a new DataFrame, and will not change the original.

If you want to change the original DataFrame, use the inplace = True argument:

### Example

Remove all rows with NULL values:

import pandas as pd  
  
df = pd.read\_csv('data.csv')  
  
df.dropna(inplace = True)  
  
print(df.to\_string())

****Note:**** Now, the dropna(inplace = True) will NOT return a new DataFrame, but it will remove all rows containg NULL values from the original DataFrame.

## Replace Empty Values

Another way of dealing with empty cells is to insert a new value instead.

This way you do not have to delete entire rows just because of some empty cells.

The fillna() method allows us to replace empty cells with a value:

### Example

Replace NULL values with the number 130:

import pandas as pd  
  
df = pd.read\_csv('data.csv')  
  
df.fillna(130, inplace = True)

### Replace Only For Specified Columns

The example above replaces all empty cells in the whole Data Frame.

To only replace empty values for one column, specify the column name for the DataFrame:

### Example

Replace NULL values in the "Calories" columns with the number 130:

import pandas as pd  
  
df = pd.read\_csv('data.csv')  
  
df["Calories"].fillna(130, inplace = True)

## Replace Using Mean, Median, or Mode

A common way to replace empty cells, is to calculate the mean, median or mode value of the column.

Pandas uses the mean() median() and mode() methods to calculate the respective values for a specified column:

### Example

Calculate the MEAN, and replace any empty values with it:

import pandas as pd  
  
df = pd.read\_csv('data.csv')  
  
x = df["Calories"].mean()  
  
df["Calories"].fillna(x, inplace = True)

****Mean**** = the average value (the sum of all values divided by number of values).

### Example

Calculate the MEDIAN, and replace any empty values with it:

import pandas as pd  
  
df = pd.read\_csv('data.csv')  
  
x = df["Calories"].median()  
  
df["Calories"].fillna(x, inplace = True)

****Median**** = the value in the middle, after you have sorted all values ascending.

### Example

Calculate the MODE, and replace any empty values with it:

import pandas as pd  
  
df = pd.read\_csv('data.csv')  
  
x = df["Calories"].mode()[0]  
  
df["Calories"].fillna(x, inplace = True)

****Mode**** = the value that appears most frequently.

# Pandas - Cleaning Data of Wrong Format

## Data of Wrong Format

Cells with data of wrong format can make it difficult, or even impossible, to analyze data.

To fix it, you have two options: remove the rows, or convert all cells in the columns into the same format.

## Convert Into a Correct Format

In our Data Frame, we have two cells with the wrong format. Check out row 22 and 26, the 'Date' column should be a string that represents a date:

Duration Date Pulse Maxpulse Calories

0 60 '2020/12/01' 110 130 409.1

1 60 '2020/12/02' 117 145 479.0

2 60 '2020/12/03' 103 135 340.0

3 45 '2020/12/04' 109 175 282.4

4 45 '2020/12/05' 117 148 406.0

5 60 '2020/12/06' 102 127 300.0

6 60 '2020/12/07' 110 136 374.0

7 450 '2020/12/08' 104 134 253.3

8 30 '2020/12/09' 109 133 195.1

9 60 '2020/12/10' 98 124 269.0

10 60 '2020/12/11' 103 147 329.3

11 60 '2020/12/12' 100 120 250.7

12 60 '2020/12/12' 100 120 250.7

13 60 '2020/12/13' 106 128 345.3

14 60 '2020/12/14' 104 132 379.3

15 60 '2020/12/15' 98 123 275.0

16 60 '2020/12/16' 98 120 215.2

17 60 '2020/12/17' 100 120 300.0

18 45 '2020/12/18' 90 112 NaN

19 60 '2020/12/19' 103 123 323.0

20 45 '2020/12/20' 97 125 243.0

21 60 '2020/12/21' 108 131 364.2

22 45 NaN 100 119 282.0

23 60 '2020/12/23' 130 101 300.0

24 45 '2020/12/24' 105 132 246.0

25 60 '2020/12/25' 102 126 334.5

26 60 20201226 100 120 250.0

27 60 '2020/12/27' 92 118 241.0

28 60 '2020/12/28' 103 132 NaN

29 60 '2020/12/29' 100 132 280.0

30 60 '2020/12/30' 102 129 380.3

31 60 '2020/12/31' 92 115 243.0

Let's try to convert all cells in the 'Date' column into dates.

Pandas has a to\_datetime() method for this:

### Example

Convert to date:

import pandas as pd  
  
df = pd.read\_csv('data.csv')  
  
df['Date'] = pd.to\_datetime(df['Date'])  
  
print(df.to\_string())

Result:

Duration Date Pulse Maxpulse Calories

0 60 '2020/12/01' 110 130 409.1

1 60 '2020/12/02' 117 145 479.0

2 60 '2020/12/03' 103 135 340.0

3 45 '2020/12/04' 109 175 282.4

4 45 '2020/12/05' 117 148 406.0

5 60 '2020/12/06' 102 127 300.0

6 60 '2020/12/07' 110 136 374.0

7 450 '2020/12/08' 104 134 253.3

8 30 '2020/12/09' 109 133 195.1

9 60 '2020/12/10' 98 124 269.0

10 60 '2020/12/11' 103 147 329.3

11 60 '2020/12/12' 100 120 250.7

12 60 '2020/12/12' 100 120 250.7

13 60 '2020/12/13' 106 128 345.3

14 60 '2020/12/14' 104 132 379.3

15 60 '2020/12/15' 98 123 275.0

16 60 '2020/12/16' 98 120 215.2

17 60 '2020/12/17' 100 120 300.0

18 45 '2020/12/18' 90 112 NaN

19 60 '2020/12/19' 103 123 323.0

20 45 '2020/12/20' 97 125 243.0

21 60 '2020/12/21' 108 131 364.2

22 45 NaT 100 119 282.0

23 60 '2020/12/23' 130 101 300.0

24 45 '2020/12/24' 105 132 246.0

25 60 '2020/12/25' 102 126 334.5

26 60 '2020/12/26' 100 120 250.0

27 60 '2020/12/27' 92 118 241.0

28 60 '2020/12/28' 103 132 NaN

29 60 '2020/12/29' 100 132 280.0

30 60 '2020/12/30' 102 129 380.3

31 60 '2020/12/31' 92 115 243.0

As you can see from the result, the date in row 26 was fixed, but the empty date in row 22 got a NaT (Not a Time) value, in other words an empty value. One way to deal with empty values is simply removing the entire row.

## Removing Rows

The result from the converting in the example above gave us a NaT value, which can be handled as a NULL value, and we can remove the row by using the dropna() method.

### Example

Remove rows with a NULL value in the "Date" column:

import pandas as pd

df = pd.read\_csv('data.csv')

df['Date'] = pd.to\_datetime(df['Date'])

df.dropna(subset=['Date'], inplace = True)We just launched  
W3Schools videos

# Pandas - Fixing Wrong Data

## Wrong Data

"Wrong data" does not have to be "empty cells" or "wrong format", it can just be wrong, like if someone registered "199" instead of "1.99".

Sometimes you can spot wrong data by looking at the data set, because you have an expectation of what it should be.

If you take a look at our data set, you can see that in row 7, the duration is 450, but for all the other rows the duration is between 30 and 60.

It doesn't have to be wrong, but taking in consideration that this is the data set of someone's workout sessions, we conclude with the fact that this person did not work out in 450 minutes.

Duration Date Pulse Maxpulse Calories

0 60 '2020/12/01' 110 130 409.1

1 60 '2020/12/02' 117 145 479.0

2 60 '2020/12/03' 103 135 340.0

3 45 '2020/12/04' 109 175 282.4

4 45 '2020/12/05' 117 148 406.0

5 60 '2020/12/06' 102 127 300.0

6 60 '2020/12/07' 110 136 374.0

7 450 '2020/12/08' 104 134 253.3

8 30 '2020/12/09' 109 133 195.1

9 60 '2020/12/10' 98 124 269.0

10 60 '2020/12/11' 103 147 329.3

11 60 '2020/12/12' 100 120 250.7

12 60 '2020/12/12' 100 120 250.7

13 60 '2020/12/13' 106 128 345.3

14 60 '2020/12/14' 104 132 379.3

15 60 '2020/12/15' 98 123 275.0

16 60 '2020/12/16' 98 120 215.2

17 60 '2020/12/17' 100 120 300.0

18 45 '2020/12/18' 90 112 NaN

19 60 '2020/12/19' 103 123 323.0

20 45 '2020/12/20' 97 125 243.0

21 60 '2020/12/21' 108 131 364.2

22 45 NaN 100 119 282.0

23 60 '2020/12/23' 130 101 300.0

24 45 '2020/12/24' 105 132 246.0

25 60 '2020/12/25' 102 126 334.5

26 60 20201226 100 120 250.0

27 60 '2020/12/27' 92 118 241.0

28 60 '2020/12/28' 103 132 NaN

29 60 '2020/12/29' 100 132 280.0

30 60 '2020/12/30' 102 129 380.3

31 60 '2020/12/31' 92 115 243.0

How can we fix wrong values, like the one for "Duration" in row 7?

## Replacing Values

One way to fix wrong values is to replace them with something else.

In our example, it is most likely a typo, and the value should be "45" instead of "450", and we could just insert "45" in row 7:

### Example

Set "Duration" = 45 in row 7:

df.loc[7, 'Duration'] = 45

For small data sets you might be able to replace the wrong data one by one, but not for big data sets.

To replace wrong data for larger data sets you can create some rules, e.g. set some boundaries for legal values, and replace any values that are outside of the boundaries.

### Example

Loop through all values in the "Duration" column.

If the value is higher than 120, set it to 120:

for x in df.index:  
  if df.loc[x, "Duration"] > 120:  
    df.loc[x, "Duration"] = 120

## Removing Rows

Another way of handling wrong data is to remove the rows that contains wrong data.

This way you do not have to find out what to replace them with, and there is a good chance you do not need them to do your analyses.

### Example

Delete rows where "Duration" is higher than 120:

import pandas as pd

df = pd.read\_csv('data.csv')

for x in df.index:

if df.loc[x, "Duration"] > 120:

df.drop(x, inplace = True)

#remember to include the 'inplace = True' argument to make the changes in the original DataFrame object instead of returning a copy

print(df.to\_string())

# Pandas - Removing Duplicates

## Discovering Duplicates

Duplicate rows are rows that have been registered more than one time.

Duration Date Pulse Maxpulse Calories

0 60 '2020/12/01' 110 130 409.1

1 60 '2020/12/02' 117 145 479.0

2 60 '2020/12/03' 103 135 340.0

3 45 '2020/12/04' 109 175 282.4

4 45 '2020/12/05' 117 148 406.0

5 60 '2020/12/06' 102 127 300.0

6 60 '2020/12/07' 110 136 374.0

7 450 '2020/12/08' 104 134 253.3

8 30 '2020/12/09' 109 133 195.1

9 60 '2020/12/10' 98 124 269.0

10 60 '2020/12/11' 103 147 329.3

11 60 '2020/12/12' 100 120 250.7

12 60 '2020/12/12' 100 120 250.7

13 60 '2020/12/13' 106 128 345.3

14 60 '2020/12/14' 104 132 379.3

15 60 '2020/12/15' 98 123 275.0

16 60 '2020/12/16' 98 120 215.2

17 60 '2020/12/17' 100 120 300.0

18 45 '2020/12/18' 90 112 NaN

19 60 '2020/12/19' 103 123 323.0

20 45 '2020/12/20' 97 125 243.0

21 60 '2020/12/21' 108 131 364.2

22 45 NaN 100 119 282.0

23 60 '2020/12/23' 130 101 300.0

24 45 '2020/12/24' 105 132 246.0

25 60 '2020/12/25' 102 126 334.5

26 60 20201226 100 120 250.0

27 60 '2020/12/27' 92 118 241.0

28 60 '2020/12/28' 103 132 NaN

29 60 '2020/12/29' 100 132 280.0

30 60 '2020/12/30' 102 129 380.3

31 60 '2020/12/31' 92 115 243.0

By taking a look at our test data set, we can assume that row 11 and 12 are duplicates.

To discover duplicates, we can use the duplicated() method.

The duplicated() method returns a Boolean values for each row:

### Example

Returns True for every row that is a duplicate, othwerwise False:

print(df.duplicated())

## Removing Duplicates

To remove duplicates, use the drop\_duplicates() method.

### Example

Remove all duplicates:

df.drop\_duplicates(inplace = True)

****Remember:**** The (inplace = True) will make sure that the method does NOT return a new DataFrame, but it will remove all duplicates from the original DataFrame.