II] Reading and Writing Data in Text Format

> In python, the pandas module allows us to load Dataset from external files and work on them. The dataset can be in different types of files.

Reading a CSV File into the DataFrame

File path Header Column delimiter Index column Select columns Omit rows Missing values Alter values Compress and decompress files

What is .csv files?

> A **comma-separated values (CSV)** file is a plaintext file with a .csv extension that holds tabular data.

> This is one of the **most popular file formats for storing large amounts of data.** Each row of the CSV file represents a single table row.

> The values in the same row are by default separated with commas, but you could change the separator to a semicolon, tab, space, or some other character.

> It is a simple text file, following a few formatting conventions. However, it is the most common, simple, and easiest method to store tabular data. This format arranges tables by following a specific structure divided into rows and columns. It is these rows and columns that contain your data.

A new line terminates each row to start the next row. Similarly, a delimiter (seperator), usually a comma, separates columns within each row.

For example:

We might have a table that looks like this:

City	State	Capital	Population
Philadelphia	Pennsylvania	No	1.581 Million
Sacramento	California	Yes	0.5 Million
New York	New York	No	8.623 Million
Austin	Texas	Yes	0.95 Million
Miami	Florida	No	0.463 Million

If we were to convert it into the CSV format, it'd look like this:

City,State,Capital,Population

Philadelphia,Pennsylvania,No,1.581 Million

Sacramento,California,Yes,0.5 Million

New York,New York,No,8.623 Million

Austin,Texas,Yes,0.95 Million

Miami,Florida,No,0.463 Million

Although the name (Comma-Separated Values) inherently uses a comma as the delimiter, you can use other delimiters (separators) as well, such as the semicolon (;). Each row of the table is a new line of the CSV file and it's a very compact and concise way to represent tabular data.

How to read a csv file()?

> Pandas provides a read_csv() to read any csv file. Here are a few other functions:
>
> - read_excel()
> - read_json()
> - read_html()
> - read_sql()

Now, let's take a look at the read_csv() function.

Using read_csv()

There are three parameters we can pass to the read_csv() function.

Syntax:

> **data=pandas.read_csv('filename.txt', sep=' ', header=None, names=["Column1", "Column2"])**

Parameters:

1. filename.txt: As the name suggests it is the name of the text file from which we want to read data.

2. sep: It is a separator field. In the text file, we use the space character(' ') as the separator.

3. header: This is an optional field. By default, it will take the first line of the text file as a header. If we use header=None then it will create the header.

4. names: We can assign column names while importing the text file by using the names argument.

Creating a Dataframe using read_csv()

```
# importing pandas as pd
import pandas as pd

# Creating the dataframe
df = pd.read_csv("/content/drive/MyDrive/nba.csv")

# Print the dataframe
df
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
...	...	...	...	...	...	...	...	...	...
453	Shelvin Mack	Utah Jazz	8.0	PG	26.0	6-3	203.0	Butler	2433333.0
454	Raul Neto	Utah Jazz	25.0	PG	24.0	6-1	179.0	NaN	900000.0
455	Tibor Pleiss	Utah Jazz	21.0	C	26.0	7-3	256.0	NaN	2900000.0
456	Jeff Withey	Utah Jazz	24.0	C	26.0	7-0	231.0	Kansas	947276.0
457	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

458 rows × 9 columns

Write operation to write a DataFrame into a CSV File.

Series and DataFrame objects have methods that enable writing data and labels to the clipboard or files.

Pandas provide .to_csv() to write a DataFrame into a csv file:

Syntax: DataFrame.to_csv()

```
df.to_csv("/content/drive/MyDrive/W.csv")
```

There are other functions for writing, including:

- .to_excel()
- .to_json()
- .to_html()
- .to_sql()

I] Sorting Pandas Data Frame

The two major sort functions are for:

- Sorting by the values of the selected columns
- Sorting by the labels of the DataFrame

Therefore, Sorting in Pandas can be done in the following two ways:

i) Sorting By Index / Labels

ii) Sorting By Value

A) Sort by index

sort_index()

- This function **sorts objects by labels** along the given axis.

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

Syntax:

> **DataFrame.sort_index(axis, ascending, inplace, kind, na_position)**

Parameters :

axis : index, columns to direct sorting

ascending : True or False to Sort ascending or descending

inplace : if True, perform operation in-place

kind : {‘quicksort’, ‘mergesort’, ‘heapsort’}, default ‘quicksort’.

Note: Choice of sorting algorithm. See also ndarray.np.sort for more information. mergesort is the only stable algorithm. For DataFrames, this option is only applied when sorting on a single column or label.

na_position : [{'first', 'last'}, default 'last'] If 'first' puts NaNs at the beginning, 'last' puts NaNs at the end.

Create a Dataframe using .csv file

```
# importing pandas as pd
import pandas as pd

# Creating the dataframe
df = pd.read_csv("/content/drive/MyDrive/nba.csv")

# Print the dataframe
df
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
...	...	...	...	...	...	...	...	...	...
453	Shelvin Mack	Utah Jazz	8.0	PG	26.0	6-3	203.0	Butler	2433333.0
454	Raul Neto	Utah Jazz	25.0	PG	24.0	6-1	179.0	NaN	900000.0
455	Tibor Pleiss	Utah Jazz	21.0	C	26.0	7-3	256.0	NaN	2900000.0
456	Jeff Withey	Utah Jazz	24.0	C	26.0	7-0	231.0	Kansas	947276.0
457	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

458 rows × 9 columns

Let's use the dataframe.sort_index() function to sort the dataframe based on the index labels

Example 1: Use sort_index() function to sort the dataframe based on the rows.

```
# sort by index labels
df.sort_index(axis = 0)
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0

Example 2: Use sort_index() function to sort the dataframe based on the column.

By default, sorting happen on index labels, Use axis=1 to change this and sort on columns by name in pandas DataFrame.

```
# importing pandas as pd
import pandas as pd

# Creating the dataframe
df = pd.read_csv("/content/drive/MyDrive/nba.csv")

# sorting based on column labels
df.sort_index(axis = 1)
```

	Age	College	Height	Name	Number	Position	Salary	Team	Weight
0	25.0	Texas	6-2	Avery Bradley	0.0	PG	7730337.0	Boston Celtics	180.0
1	25.0	Marquette	6-6	Jae Crowder	99.0	SF	6796117.0	Boston Celtics	235.0
2	27.0	Boston University	6-5	John Holland	30.0	SG	NaN	Boston Celtics	205.0
3	22.0	Georgia State	6-5	R.J. Hunter	28.0	SG	1148640.0	Boston Celtics	185.0
4	29.0	NaN	6-10	Jonas Jerebko	8.0	PF	5000000.0	Boston Celtics	231.0
...	...	...	...	...	...	...	...	...	...
453	26.0	Butler	6-3	Shelvin Mack	8.0	PG	2433333.0	Utah Jazz	203.0
454	24.0	NaN	6-1	Raul Neto	25.0	PG	900000.0	Utah Jazz	179.0
455	26.0	NaN	7-3	Tibor Pleiss	21.0	C	2900000.0	Utah Jazz	256.0
456	26.0	Kansas	7-0	Jeff Withey	24.0	C	947276.0	Utah Jazz	231.0
457	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

458 rows × 9 columns

```
# sort by index labels
df.sort_index(kind='mergesort')
```

```
df.sort_index(na_position='first')
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
...	...	...	...	...	...	...	...	...	...
453	Shelvin Mack	Utah Jazz	8.0	PG	26.0	6-3	203.0	Butler	2433333.0
454	Raul Neto	Utah Jazz	25.0	PG	24.0	6-1	179.0	NaN	900000.0
455	Tibor Pleiss	Utah Jazz	21.0	C	26.0	7-3	256.0	NaN	2900000.0
456	Jeff Withey	Utah Jazz	24.0	C	26.0	7-0	231.0	Kansas	947276.0
457	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

458 rows × 9 columns

To sort pandas DataFrame column values by descending order, use ascending=False. You can also specify different sorting orders for each label.

```
# sort by descending order
df.sort_index(ascending=False)
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
457	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
456	Jeff Withey	Utah Jazz	24.0	C	26.0	7-0	231.0	Kansas	947276.0
455	Tibor Pleiss	Utah Jazz	21.0	C	26.0	7-3	256.0	NaN	2900000.0
454	Raul Neto	Utah Jazz	25.0	PG	24.0	6-1	179.0	NaN	900000.0
453	Shelvin Mack	Utah Jazz	8.0	PG	26.0	6-3	203.0	Butler	2433333.0
...	...	...	...	...	...	...	...	...	...
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0

458 rows × 9 columns

B) Sort By Value

sort_values()

- It is used to sort by the values along either axis.
- It accepts a 'by' argument which will use the column name of the DataFrame with which the values are to be sorted.

Syntax:

> **DataFrame.sort_value(by, axis, ascending, inplace, kind, na_position)**

Every parameter has some default values except the 'by' parameter.

Parameters:

by: Single/List of column names to sort Data Frame by.

axis: 0 or 'index' for rows and **1 or 'columns' for Column.**

ascending: Boolean value which sorts Data frame in ascending order, if True.

inplace: Boolean value. Makes the changes in original dataframe itself, if True.

kind: String which can have three inputs('quicksort', 'mergesort' or 'heapsort') of algorithm used to sort data frame.

na_position: Takes two string input 'last' or 'first' to set position of Null values. Default is 'last'.

Create a DataFrame

```
# importing pandas package
import pandas as pd

# making data frame from csv file
data = pd.read_csv("/content/drive/MyDrive/nba.csv")

# display
data
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
...	...	...	...	...	...	...	...	...	...
453	Shelvin Mack	Utah Jazz	8.0	PG	26.0	6-3	203.0	Butler	2433333.0
454	Raul Neto	Utah Jazz	25.0	PG	24.0	6-1	179.0	NaN	900000.0
455	Tibor Pleiss	Utah Jazz	21.0	C	26.0	7-3	256.0	NaN	2900000.0
456	Jeff Withey	Utah Jazz	24.0	C	26.0	7-0	231.0	Kansas	947276.0
457	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

458 rows × 9 columns

Example 1: Sorting by Column Name

In the following example, A data frame is made from the csv file and the data frame is sorted in ascending order of Names of Players.

```
# sorting data frame by name

data.sort_values("Name", axis = 0, ascending = True,
                 inplace = True,
                 na_position ='last')
```

```
# display
data
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
152	Aaron Brooks	Chicago Bulls	0.0	PG	31.0	6-0	161.0	Oregon	2250000.0
356	Aaron Gordon	Orlando Magic	0.0	PF	20.0	6-9	220.0	Arizona	4171680.0
328	Aaron Harrison	Charlotte Hornets	9.0	SG	21.0	6-6	210.0	Kentucky	525093.0
404	Adreian Payne	Minnesota Timberwolves	33.0	PF	25.0	6-10	237.0	Michigan State	1938840.0
312	Al Horford	Atlanta Hawks	15.0	C	30.0	6-10	245.0	Florida	12000000.0
...	...	...	...	...	...	...	...	...	...
270	Xavier Munford	Memphis Grizzlies	14.0	PG	24.0	6-3	180.0	Rhode Island	NaN
402	Zach LaVine	Minnesota Timberwolves	8.0	PG	21.0	6-5	189.0	UCLA	2148360.0
271	Zach Randolph	Memphis Grizzlies	50.0	PF	34.0	6-9	260.0	Michigan State	9638555.0
237	Zaza Pachulia	Dallas Mavericks	27.0	C	32.0	6-11	275.0	NaN	5200000.0
457	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

458 rows × 9 columns

.....OR

by parameter also supports a list of labels, use this to sort DataFrame by multiple columns,

pandas sort_index() function by default sort DataFrame rows by index in ascending order. This by default returns a new DataFrame after sorting. Use inplace=True to update on existing DataFrame in place and returns a None.

```
# sorting data frame by name

data.sort_values(by = "Name", axis = 0, ascending = True, inplace = True, na_position ='last')

# display
data
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
152	Aaron Brooks	Chicago Bulls	0.0	PG	31.0	6-0	161.0	Oregon	2250000.0
356	Aaron Gordon	Orlando Magic	0.0	PF	20.0	6-9	220.0	Arizona	4171680.0
328	Aaron Harrison	Charlotte Hornets	9.0	SG	21.0	6-6	210.0	Kentucky	525093.0
404	Adreian Payne	Minnesota Timberwolves	33.0	PF	25.0	6-10	237.0	Michigan State	1938840.0
312	Al Horford	Atlanta Hawks	15.0	C	30.0	6-10	245.0	Florida	12000000.0
...	...	...	...	...	...	...	...	...	...
270	Xavier Munford	Memphis Grizzlies	14.0	PG	24.0	6-3	180.0	Rhode Island	NaN
402	Zach LaVine	Minnesota Timberwolves	8.0	PG	21.0	6-5	189.0	UCLA	2148360.0
271	Zach Randolph	Memphis Grizzlies	50.0	PF	34.0	6-9	260.0	Michigan State	9638555.0
237	Zaza Pachulia	Dallas Mavericks	27.0	C	32.0	6-11	275.0	NaN	5200000.0
457	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

458 rows × 9 columns

.....OR

```
#Write the result in new csv file

df.to_csv("/content/drive/MyDrive/WW.csv")
```

```
# sorting data frame by name

data.sort_values(by = ["Name","Team"], axis = 0, ascending = True, inplace = True, na_position ='last')

# display
data
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
152	Aaron Brooks	Chicago Bulls	0.0	PG	31.0	6-0	161.0	Oregon	2250000.0
356	Aaron Gordon	Orlando Magic	0.0	PF	20.0	6-9	220.0	Arizona	4171680.0
328	Aaron Harrison	Charlotte Hornets	9.0	SG	21.0	6-6	210.0	Kentucky	525093.0
404	Adreian Payne	Minnesota Timberwolves	33.0	PF	25.0	6-10	237.0	Michigan State	1938840.0
312	Al Horford	Atlanta Hawks	15.0	C	30.0	6-10	245.0	Florida	12000000.0
...	...	...	...	...	...	...	...	...	...
270	Xavier Munford	Memphis Grizzlies	14.0	PG	24.0	6-3	180.0	Rhode Island	NaN
402	Zach LaVine	Minnesota Timberwolves	8.0	PG	21.0	6-5	189.0	UCLA	2148360.0
271	Zach Randolph	Memphis Grizzlies	50.0	PF	34.0	6-9	260.0	Michigan State	9638555.0
237	Zaza Pachulia	Dallas Mavericks	27.0	C	32.0	6-11	275.0	NaN	5200000.0
457	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

458 rows × 9 columns

Example 2: What is 'na_position' parameter?

Changing position of NaN (Not a Number = missing values)

In the given data, there are many null values in different columns which are put in the last by default.

In this example, the Data Frame is sorted with respect to Salary column and NaN values are kept at the top.

By default, NaN on values are pushed at the bottom of the DataFrame, you can push it at the beginning by using na_position='first' parameter

```
# importing pandas package
import pandas as pd

# making data frame from csv file
data = pd.read_csv("/content/drive/MyDrive/nba.csv")

# sorting data frame by name
data.sort_values("Salary", axis = 0, ascending = True, inplace = True, na_position ='first')

data
# display
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
46	Elton Brand	Philadelphia 76ers	42.0	PF	37.0	6-9	254.0	Duke	NaN
171	Dahntay Jones	Cleveland Cavaliers	30.0	SG	35.0	6-6	225.0	Duke	NaN
264	Jordan Farmar	Memphis Grizzlies	4.0	PG	29.0	6-2	180.0	UCLA	NaN
269	Ray McCallum	Memphis Grizzlies	5.0	PG	24.0	6-3	190.0	Detroit	NaN
...	...	...	...	...	...	...	...	...	...
339	Chris Bosh	Miami Heat	1.0	PF	32.0	6-11	235.0	Georgia Tech	22192730.0
251	Dwight Howard	Houston Rockets	12.0	C	30.0	6-11	265.0	NaN	22359364.0
33	Carmelo Anthony	New York Knicks	7.0	SF	32.0	6-8	240.0	Syracuse	22875000.0

Observe, col1 values are sorted and the respective col2 value and row index will alter along with col1. Thus, they look unsorted.

> **'by' argument takes a list of column values.**

```
# Read Text Files with Pandas using read_csv()

# importing pandas
import pandas as pd

# read text file into pandas DataFrame
df = pd.read_csv("/content/drive/MyDrive/nba.csv")

print(df)
```

```
                Name            Team  Number Position   Age Height  Weight  \
    0    Avery Bradley  Boston Celtics     0.0       PG  25.0    6-2   180.0
    1      Jae Crowder  Boston Celtics    99.0       SF  25.0    6-6   235.0
    2     John Holland  Boston Celtics    30.0       SG  27.0    6-5   205.0
    3      R.J. Hunter  Boston Celtics    28.0       SG  22.0    6-5   185.0
    4    Jonas Jerebko  Boston Celtics     8.0       PF  29.0   6-10   231.0
    ..             ...             ...     ...      ...   ...    ...     ...
    453   Shelvin Mack       Utah Jazz     8.0       PG  26.0    6-3   203.0
    454      Raul Neto       Utah Jazz    25.0       PG  24.0    6-1   179.0
    455   Tibor Pleiss       Utah Jazz    21.0        C  26.0    7-3   256.0
    456    Jeff Withey       Utah Jazz    24.0        C  26.0    7-0   231.0
    457            NaN             NaN     NaN      NaN   NaN    NaN     NaN

                 College     Salary
    0              Texas  7730337.0
    1          Marquette  6796117.0
    2    Boston University        NaN
    3        Georgia State  1148640.0
    4                  NaN  5000000.0
    ..                 ...        ...
    453             Butler  2433333.0
    454                NaN   900000.0
    455                NaN  2900000.0
    456             Kansas   947276.0
    457                NaN        NaN

    [458 rows x 9 columns]
```

```
# sorting data frame by name

df.sort_values(by = "Name", axis = 0, ascending = True, inplace = True, na_position ='last')

# display
df
```

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
152	Aaron Brooks	Chicago Bulls	0.0	PG	31.0	6-0	161.0	Oregon	2250000.0
356	Aaron Gordon	Orlando Magic	0.0	PF	20.0	6-9	220.0	Arizona	4171680.0
328	Aaron Harrison	Charlotte Hornets	9.0	SG	21.0	6-6	210.0	Kentucky	525093.0
404	Adreian Payne	Minnesota Timberwolves	33.0	PF	25.0	6-10	237.0	Michigan State	1938840.0
312	Al Horford	Atlanta Hawks	15.0	C	30.0	6-10	245.0	Florida	12000000.0
...	...	...	...	...	...	...	...	...	...
270	Xavier Munford	Memphis Grizzlies	14.0	PG	24.0	6-3	180.0	Rhode Island	NaN
402	Zach LaVine	Minnesota Timberwolves	8.0	PG	21.0	6-5	189.0	UCLA	2148360.0
271	Zach Randolph	Memphis Grizzlies	50.0	PF	34.0	6-9	260.0	Michigan State	9638555.0
237	Zaza Pachulia	Dallas Mavericks	27.0	C	32.0	6-11	275.0	NaN	5200000.0
457	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Sorting Example:

Reset Index after sorting

Sometimes you may need to set the new index on the sorting result, you can do this while sorting by using ignore_index=True or by calling pandas.DataFrame.reset_index() on sorted DataFrame.

```
# Sort and ignore index
df2 = df.sort_values(by='Age', ignore_index=True)
print(df2)
```

```
                  Name                    Team  Number Position   Age Height  \
0         Rashad Vaughn         Milwaukee Bucks    20.0       SG  19.0    6-6
1          Devin Booker            Phoenix Suns     1.0       SG  19.0    6-6
2    Kristaps Porzingis         New York Knicks     6.0       PF  20.0    7-3
3            Tyus Jones  Minnesota Timberwolves     1.0       PG  20.0    6-2
4       Cliff Alexander  Portland Trail Blazers    34.0       PF  20.0    6-8
..                  ...                     ...     ...      ...   ...    ...
453      Pablo Prigioni    Los Angeles Clippers     9.0       PG  39.0    6-3
454          Tim Duncan       San Antonio Spurs    21.0        C  40.0   6-11
455       Kevin Garnett  Minnesota Timberwolves    21.0       PF  40.0   6-11
456        Andre Miller       San Antonio Spurs    24.0       PG  40.0    6-3
457                 NaN                     NaN     NaN      NaN   NaN    NaN

     Weight      College     Salary
0     202.0         UNLV  1733040.0
1     206.0     Kentucky  2127840.0
2     240.0          NaN  4131720.0
3     195.0         Duke  1282080.0
4     240.0       Kansas   525093.0
..      ...          ...        ...
453   185.0          NaN   947726.0
454   250.0  Wake Forest  5250000.0
455   240.0          NaN  8500000.0
456   200.0         Utah   250750.0
457     NaN          NaN        NaN

[458 rows x 9 columns]
```

Example 1: Sorting the Data frame in Ascending order

```
# importing pandas library
```

```
import pandas as pd

# creating and initializing a nested list
age_list = [['Afghanistan', 1952, 8425333, 'Asia'],
            ['Australia', 1957, 9712569, 'Oceania'],
            ['Brazil', 1962, 76039390, 'Americas'],
            ['China', 1957, 637408000, 'Asia'],
            ['France', 1957, 44310863, 'Europe'],
            ['India', 1952, 3.72e+08, 'Asia'],
            ['United States', 1957, 171984000, 'Americas']]

# creating a pandas dataframe
df = pd.DataFrame(age_list, columns=['Country', 'Year', 'Population', 'Continent'], ascending = True)

df
```

	Country	Year	Population	Continent
0	Afghanistan	1952	8425333.0	Asia
1	Australia	1957	9712569.0	Oceania
2	Brazil	1962	76039390.0	Americas
3	China	1957	637408000.0	Asia
4	France	1957	44310863.0	Europe
5	India	1952	372000000.0	Asia
6	United States	1957	171984000.0	Americas

Example 2: Sorting the Data frame in Descending order

By passing False to ascending, you reverse the sort order. Now your DataFrame is sorted in descending order

```
# Sorting Pandas Dataframe in Descending Order

# importing pandas library
import pandas as pd

# Initializing the nested list with Data set
age_list = [['Afghanistan', 1952, 8425333, 'Asia'],
            ['Australia', 1957, 9712569, 'Oceania'],
            ['Brazil', 1962, 76039390, 'Americas'],
            ['China', 1957, 637408000, 'Asia'],
            ['France', 1957, 44310863, 'Europe'],
            ['India', 1952, 3.72e+08, 'Asia'],
            ['United States', 1957, 171984000, 'Americas']]

# creating a pandas dataframe
df = pd.DataFrame(age_list, columns=['Country', 'Year', 'Population', 'Continent'])

# Sorting by column "Population"
df.sort_values(by=['Population'], ascending=False)
```

Country Year Population Continent

Example 3: Sorting Pandas Data frame by putting missing values first

```
# Sorting Pandas Data frame by putting
# missing values first

# importing pandas library
import pandas as pd

# Initializing the nested list with Data set
age_list = [['Afghanistan', 1952, 8425333, 'Asia'],
            ['Australia', 1957, 9712569, 'Oceania'],
            ['Brazil', 1962, 76039390, 'Americas'],
            ['China', 1957, 637408000, 'Asia'],
            ['France', 1957, 44310863, 'Europe'],
            ['India', 1952, 3.72e+08, 'Asia'],
            ['United States', 1957, 0, 'Americas']]

# creating a pandas dataframe
df = pd.DataFrame(age_list, columns=['Country', 'Year', 'Population', 'Continent'])

# Sorting by column "Population" # by putting missing values first
df.sort_values(by=['Population'], na_position='first')
```

	Country	Year	Population	Continent
6	United States	1957	0.0	Americas
0	Afghanistan	1952	8425333.0	Asia
1	Australia	1957	9712569.0	Oceania
4	France	1957	44310863.0	Europe
2	Brazil	1962	76039390.0	Americas
5	India	1952	372000000.0	Asia
3	China	1957	637408000.0	Asia

Example 4: Sorting Data frames by multiple columns

```
#  Sorting Pandas Dataframe based on
# the Values of Multiple Columns

# importing pandas library
import pandas as pd

# Initializing the nested list with data set
age_list = [['Afghanistan', 1952, 8425333, 'Asia'],
            ['Australia', 1957, 9712569, 'Oceania'],
            ['Brazil', 1962, 76039390, 'Americas'],
            ['China', 1957, 637408000, 'Asia'],
            ['France', 1957, 44310863, 'Europe'],
            ['India', 1952, 3.72e+08, 'Asia'],
            ['United States', 1957, 171984000, 'Americas']]

# creating a pandas dataframe
df = pd.DataFrame(age_list, columns=['Country', 'Year', 'Population', 'Continent'])


# Sorting by columns "Country" and then "Continent"
df.sort_values(by=['Country', 'Continent'])
```

	Country	Year	Population	Continent
0	Afghanistan	1952	8425333.0	Asia
1	Australia	1957	9712569.0	Oceania
2	Brazil	1962	76039390.0	Americas
3	China	1957	637408000.0	Asia
4	France	1957	44310863.0	Europe
5	India	1952	372000000.0	Asia

Additional Points:

Common questions for sorting can be:

1: Sort by one column's values

2: Sort by one column's values in descending order

3: Sort by multiple column values

4: Sort by multiple column values with a different sort order

5: Sort, but put missing values first

6: Sort the original dataframe (inplace = True)

7: Sort in copy of the dataframe (inplace = False)