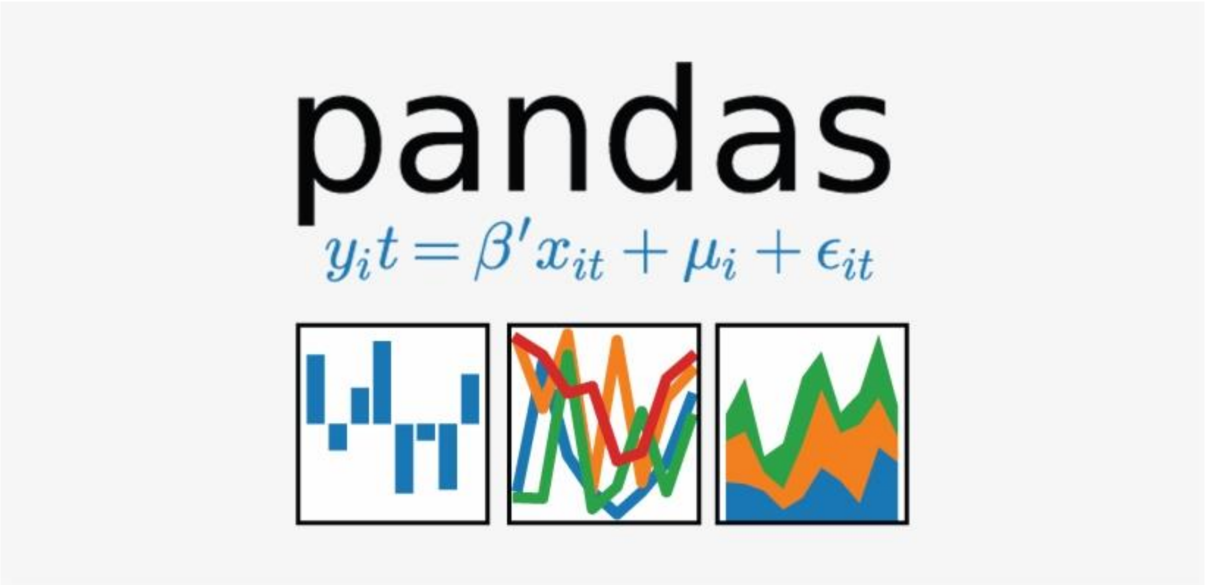
Common Pandas Methods



Author

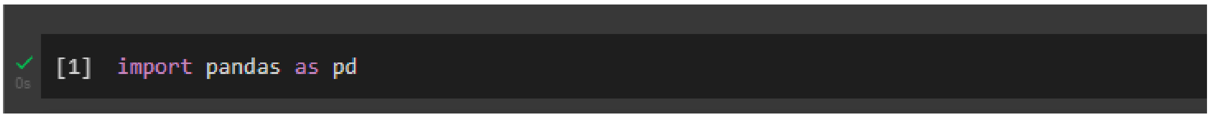
Uman Sheikh

Reading Csv File

Python library Pandas provide *read_csv* function for reading a csv file and convert in into data frame. This function also accepts some arguments which are discussed below.

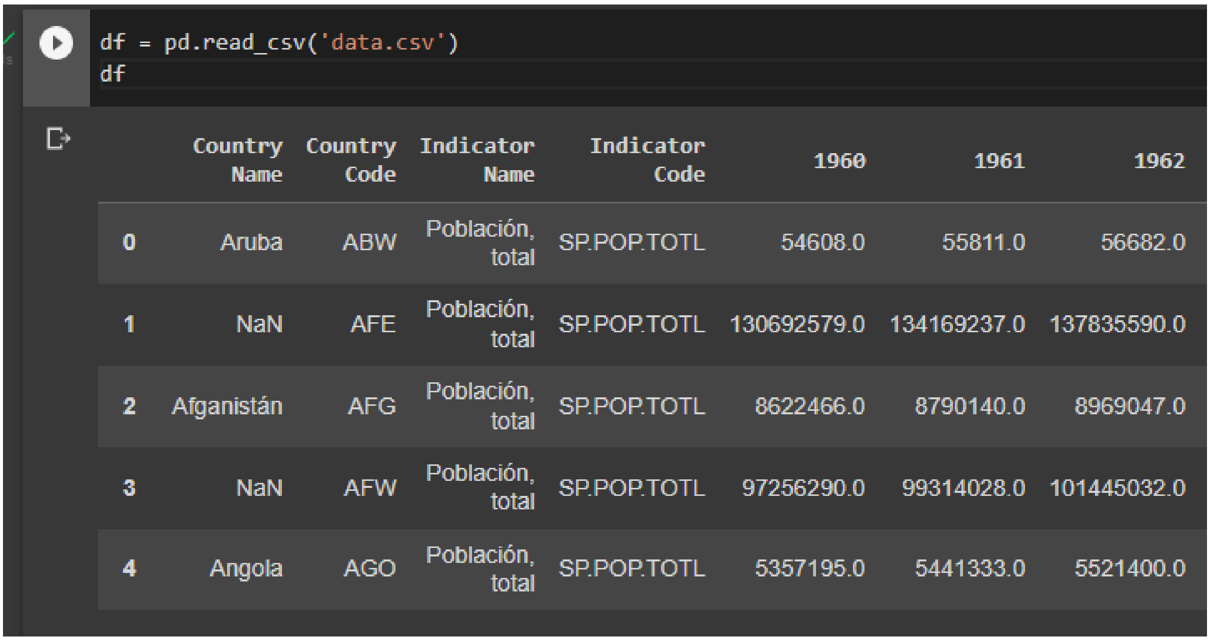
- **filepath_or_buffer**: It is the location of the file. It accepts any string path or URL of the file.
- **sep**: It stands for separator, default is "," as in CSV(comma separated values).
- **header**: It is used to name columns in your data. It accepts int and list of int.
- **usecols**: It is used to retrieve only selected columns from the CSV file.
- **nrows**: It is used to display selected numbers of rows from the dataset.
- **skiprows**: Skips passed rows in the new data frame.

Let's check all of the above methods with an example



```
[1]  import pandas as pd
```

Importing pandas as pd is a convention that many data scientist use so, we will continue using this convention.



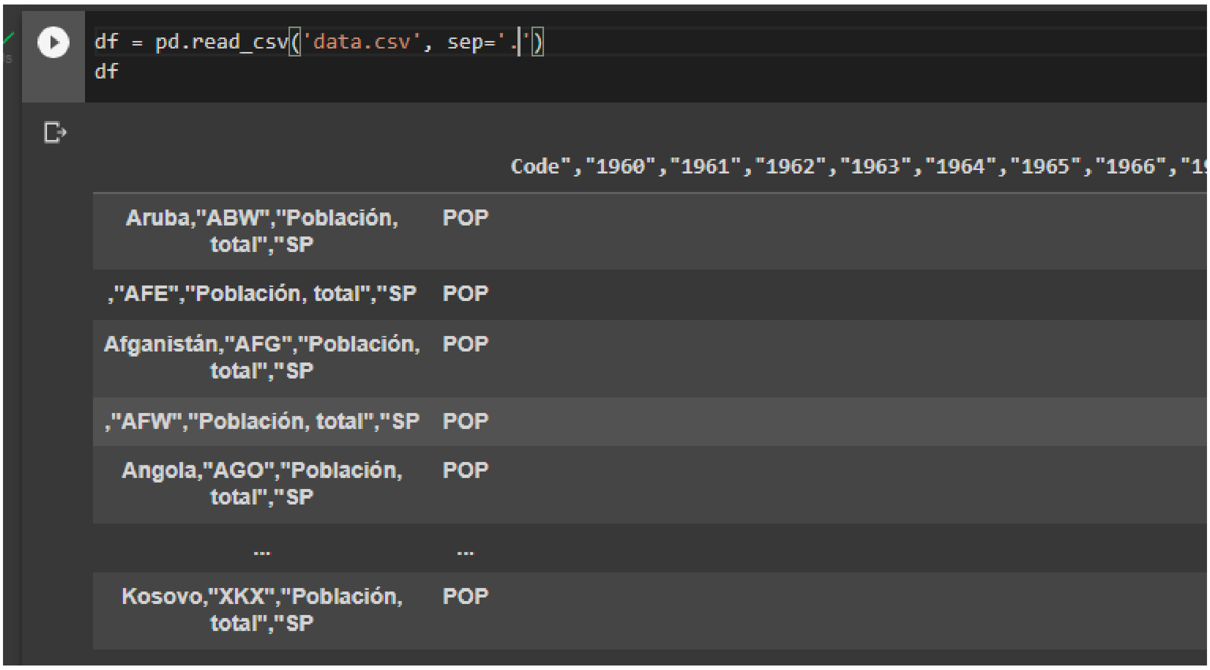
```
df = pd.read_csv('data.csv')
df
```

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962
0	Aruba	ABW	Población, total	SP.POP.TOTL	54608.0	55811.0	56682.0
1	NaN	AFE	Población, total	SP.POP.TOTL	130692579.0	134169237.0	137835590.0
2	Afganistán	AFG	Población, total	SP.POP.TOTL	8622466.0	8790140.0	8969047.0
3	NaN	AFW	Población, total	SP.POP.TOTL	97256290.0	99314028.0	101445032.0
4	Angola	AGO	Población, total	SP.POP.TOTL	5357195.0	5441333.0	5521400.0

We are using *read_csv* function, passing it file name which is *data.csv* in above example and then storing it in a df variable which stands for Data Frame. After that we can simply see our data in the notebook.

Separator

Now we will pass sep argument and then we will check the data frame.



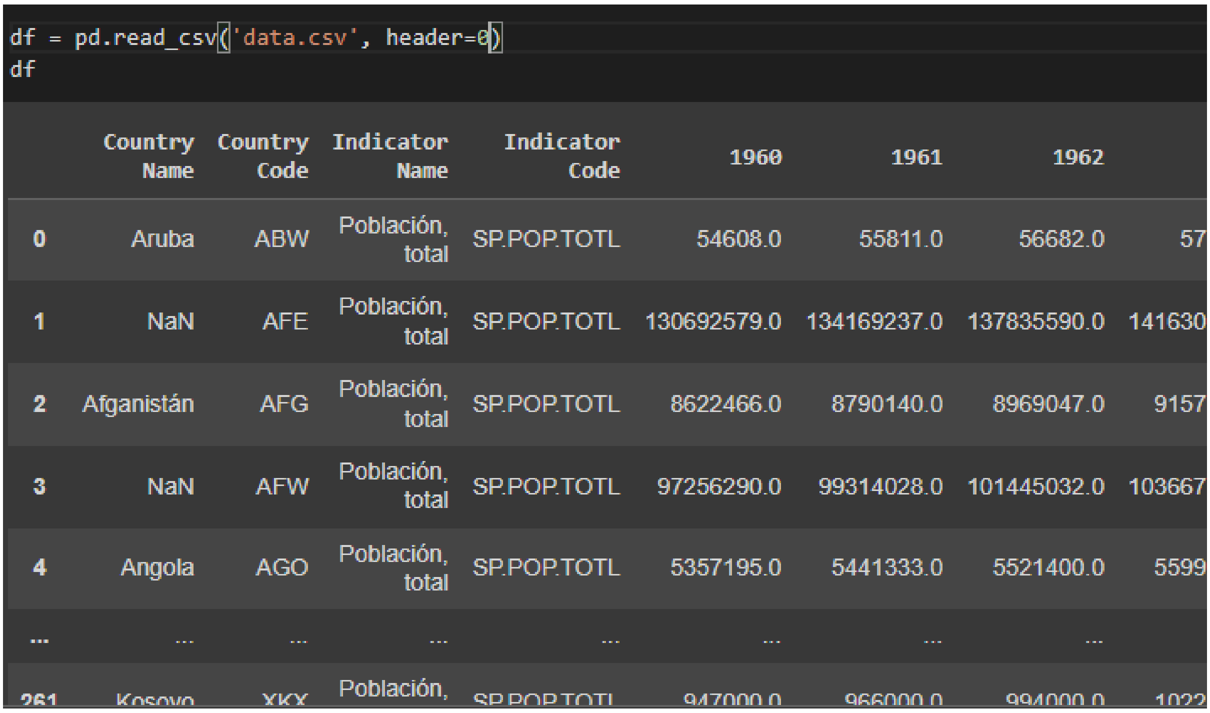
```
df = pd.read_csv('data.csv', sep='.')
df
```

	Code","1960","1961","1962","1963","1964","1965","1966","1
Aruba,"ABW","Población, total","SP	POP
,"AFE","Población, total","SP	POP
Afganistán,"AFG","Población, total","SP	POP
,"AFW","Población, total","SP	POP
Angola,"AGO","Población, total","SP	POP
...	...
Kosovo,"XKX","Población, total","SP	POP

By passing the sep='.' we can see a different data frame it is because the pandas have separated columns where it found '.'

Header

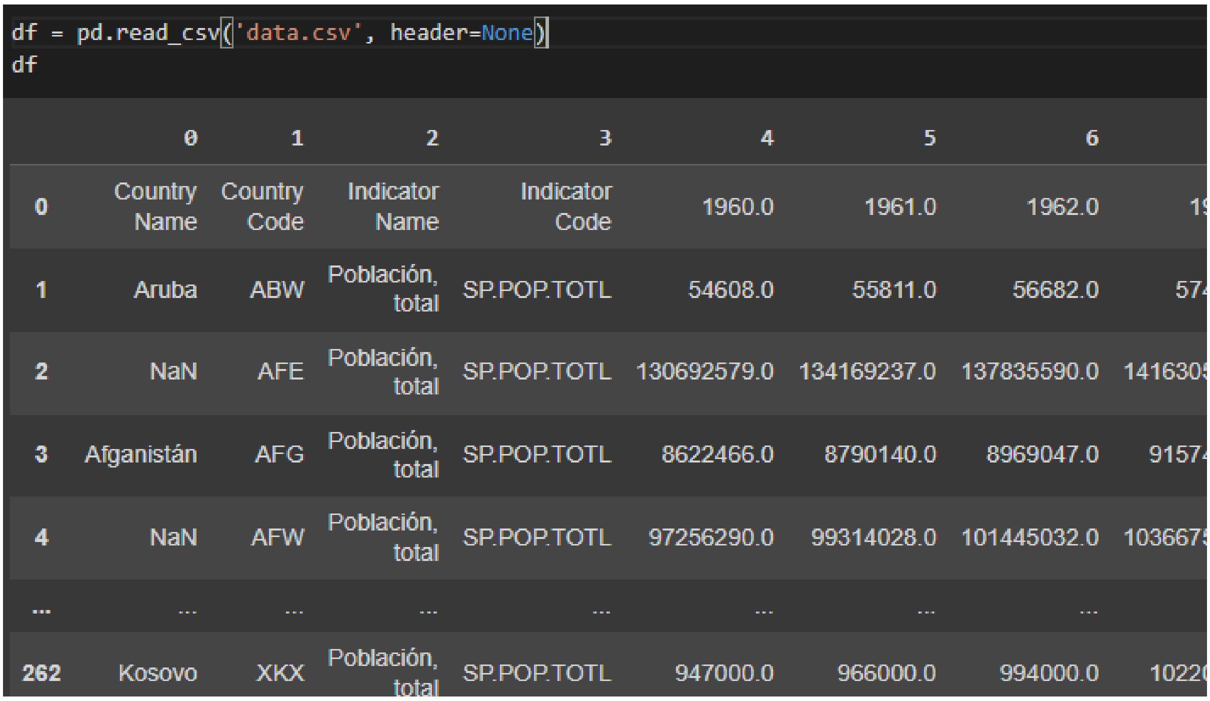
Let's see how we can use header argument.



```
df = pd.read_csv('data.csv', header=0)
df
```

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	
0	Aruba	ABW	Población, total	SP.POP.TOTL	54608.0	55811.0	56682.0	57
1	NaN	AFE	Población, total	SP.POP.TOTL	130692579.0	134169237.0	137835590.0	141630
2	Afganistán	AFG	Población, total	SP.POP.TOTL	8622466.0	8790140.0	8969047.0	9157
3	NaN	AFW	Población, total	SP.POP.TOTL	97256290.0	99314028.0	101445032.0	103667
4	Angola	AGO	Población, total	SP.POP.TOTL	5357195.0	5441333.0	5521400.0	5599
...	...	...	...	...	...	...	...	
261	Kosovo	XKX	Población,	SP.POP.TOTL	947000.0	966000.0	994000.0	1022

Header tells the function to use a certain row as columns. Default is 0. If you want the first row to be used as a normal row and not as a column header, give the value None to the 'header' argument.



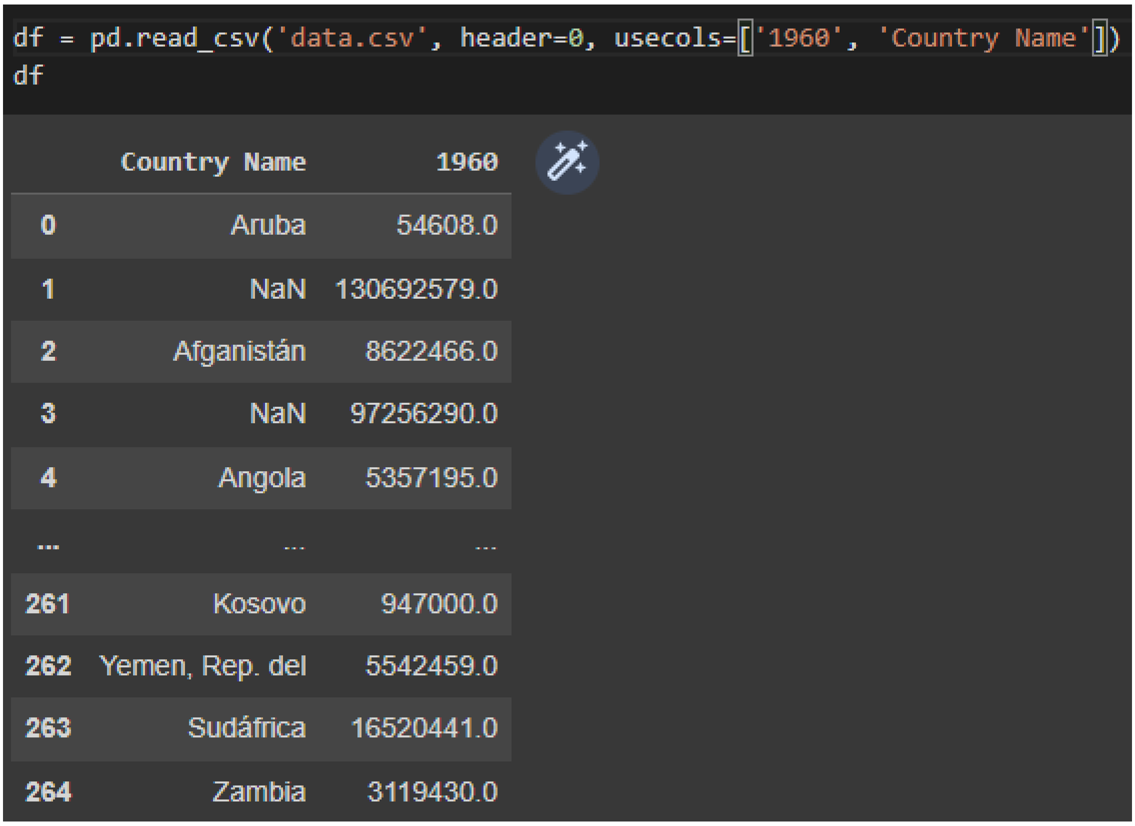
```
df = pd.read_csv('data.csv', header=None)
df
```

```
          0        1           2           3            4            5            6
0    Country  Country   Indicator   Indicator       1960.0       1961.0       1962.0      19
        Name     Code        Name        Code
1      Aruba      ABW   Población,  SP.POP.TOTL    54608.0      55811.0      56682.0      574
                            total
2        NaN      AFE   Población,  SP.POP.TOTL  130692579.0  134169237.0  137835590.0  1416305
                            total
3  Afganistán     AFG   Población,  SP.POP.TOTL   8622466.0    8790140.0    8969047.0    91574
                            total
4        NaN      AFW   Población,  SP.POP.TOTL  97256290.0   99314028.0  101445032.0  1036675
                            total
...      ...      ...         ...         ...          ...          ...          ...
262   Kosovo      XKX   Población,  SP.POP.TOTL   947000.0     966000.0     994000.0    10220
                            total
```

Above image shows that we are using first row as a normal row instead of column names.

Usecols

It is used to retrieve only specific columns.

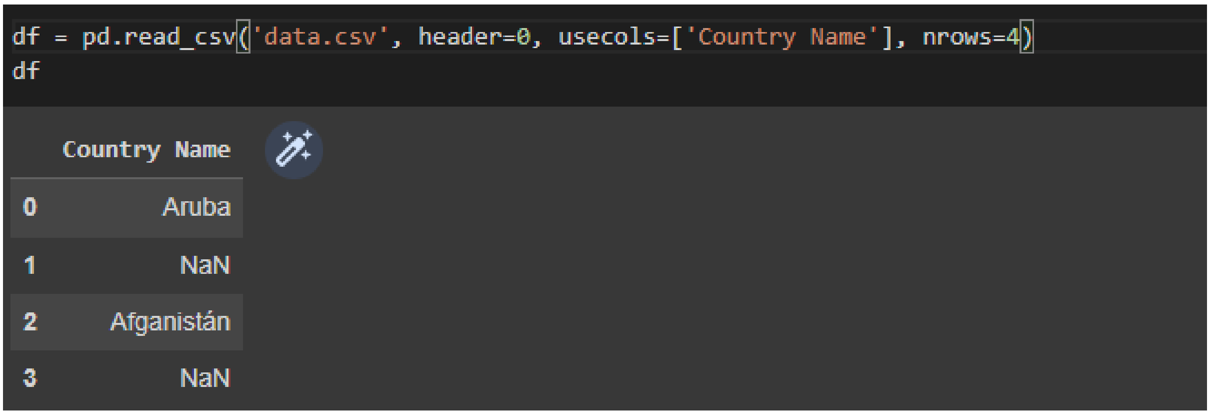


```
df = pd.read_csv('data.csv', header=0, usecols=['1960', 'Country Name'])
df
```

```
        Country Name         1960
0              Aruba      54608.0
1                NaN  130692579.0
2         Afganistán    8622466.0
3                NaN   97256290.0
4             Angola    5357195.0
...              ...          ...
261           Kosovo     947000.0
262  Yemen, Rep. del    5542459.0
263        Sudáfrica   16520441.0
264           Zambia    3119430.0
```

Nrows

It is used to select the specified rows only.

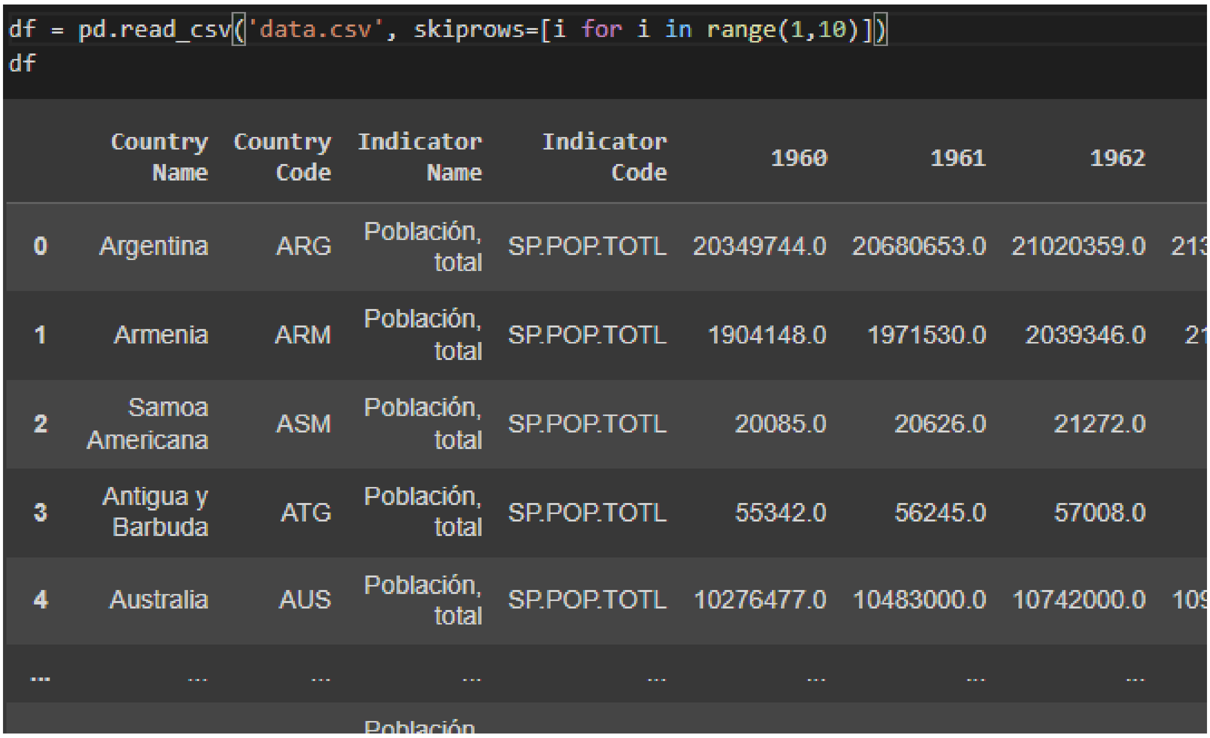


```
df = pd.read_csv('data.csv', header=0, usecols=['Country Name'], nrows=4)
df
```

	Country Name
0	Aruba
1	NaN
2	Afganistán
3	NaN

Skiprows

This argument is use to skip the given rows. Type given must be a list of integers.

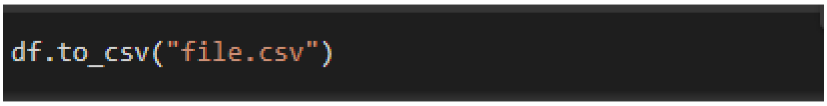


```
df = pd.read_csv('data.csv', skiprows=[i for i in range(1,10)])
df
```

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	
0	Argentina	ARG	Población, total	SP.POP.TOTL	20349744.0	20680653.0	21020359.0	213
1	Armenia	ARM	Población, total	SP.POP.TOTL	1904148.0	1971530.0	2039346.0	21
2	Samoa Americana	ASM	Población, total	SP.POP.TOTL	20085.0	20626.0	21272.0	
3	Antigua y Barbuda	ATG	Población, total	SP.POP.TOTL	55342.0	56245.0	57008.0	
4	Australia	AUS	Población, total	SP.POP.TOTL	10276477.0	10483000.0	10742000.0	109
...	...	...	...	...	...	...	...	
			Población					

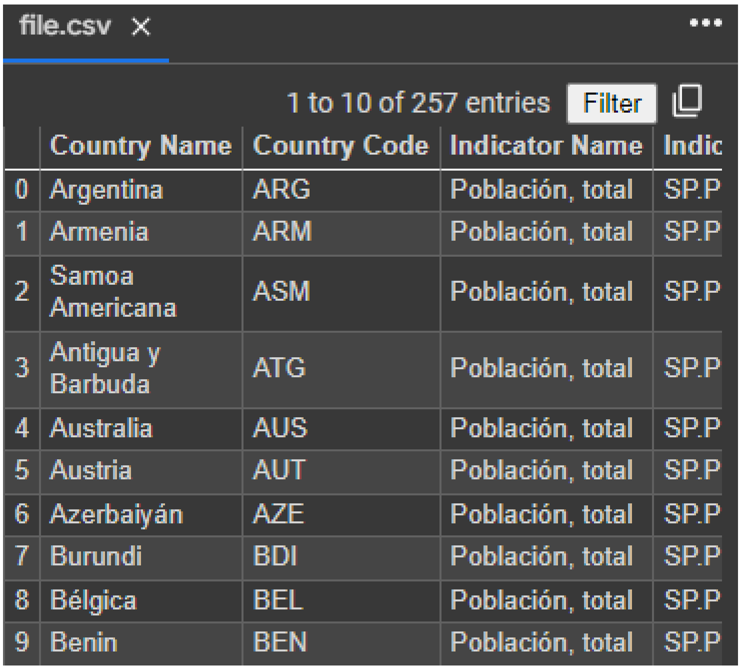
Writing Data to CSV files

We can write data to csv file using *to_csv* function. Following is an example that we can use to save our analysis to a csv file.



```
df.to_csv("file.csv")
```

The output of the above code is given blow.



1 to 10 of 257 entries

	Country Name	Country Code	Indicator Name	Indic
0	Argentina	ARG	Población, total	SP.P
1	Armenia	ARM	Población, total	SP.P
2	Samoa Americana	ASM	Población, total	SP.P
3	Antigua y Barbuda	ATG	Población, total	SP.P
4	Australia	AUS	Población, total	SP.P
5	Austria	AUT	Población, total	SP.P
6	Azerbaiyán	AZE	Población, total	SP.P
7	Burundi	BDI	Población, total	SP.P
8	Bélgica	BEL	Población, total	SP.P
9	Benin	BEN	Población, total	SP.P

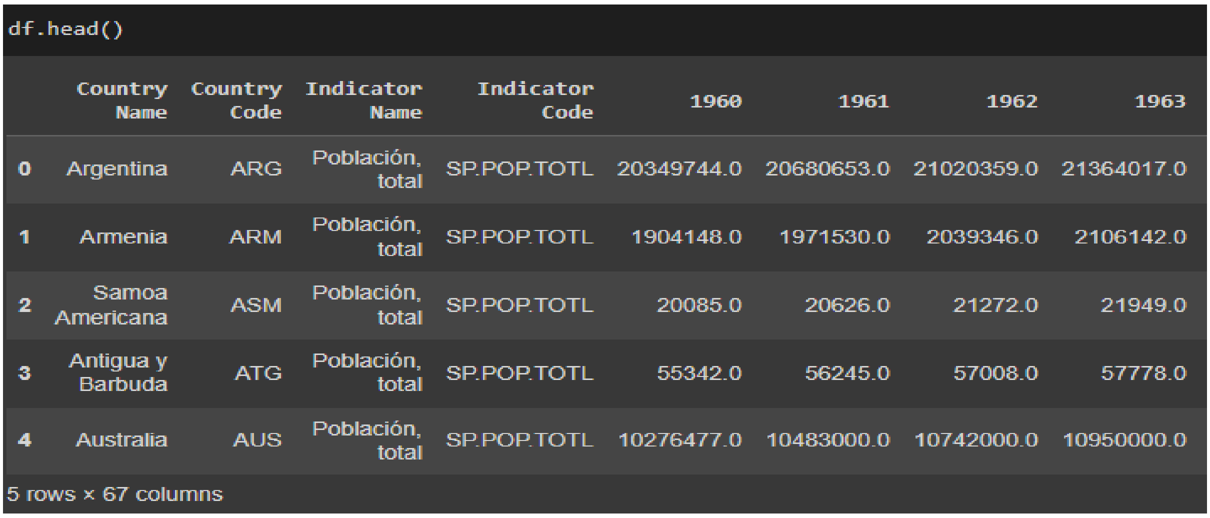
Data Inspection

Following are some functions which are used for data inspection. These functions help to understand given data more precisely. These are:

- **<u>head:</u>** Display the first 5 rows of a Data Frame.
- **<u>tail:</u>** Display the last 5 rows of a Data Frame.
- **<u>info:</u>** Display information about Data Frame, it also tells data types and memory usage.
- **<u>Describe:</u>** Display summary statistics of numerical columns in a Data Frame.

Head

It will display first 5 rows of data frame.



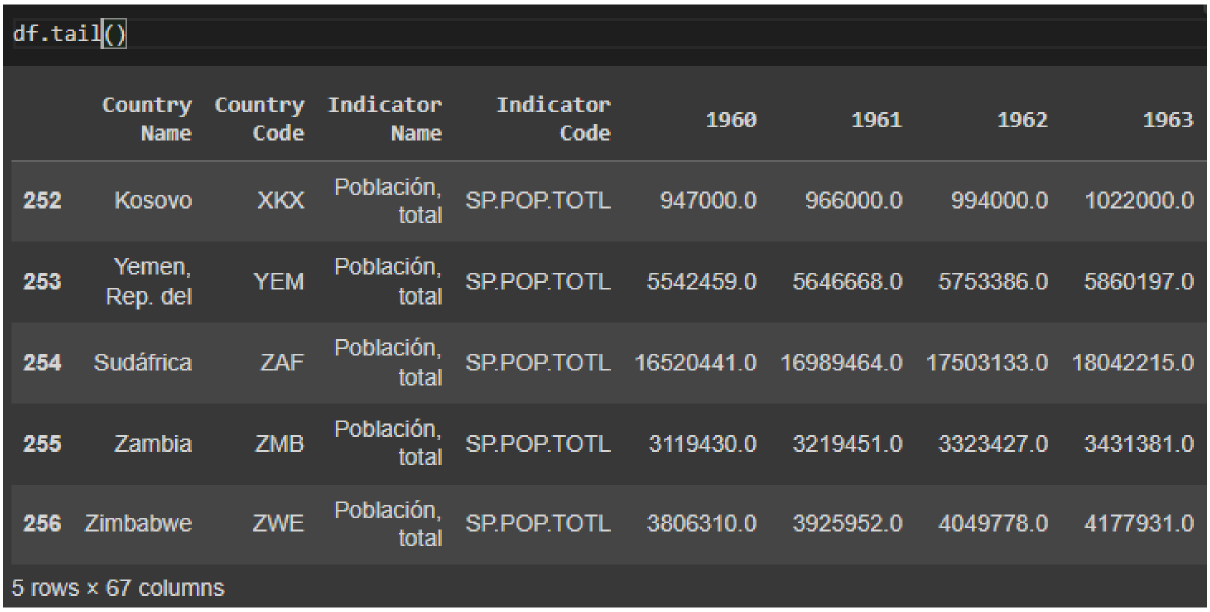
```
df.head()
```

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963
0	Argentina	ARG	Población, total	SP.POP.TOTL	20349744.0	20680653.0	21020359.0	21364017.0
1	Armenia	ARM	Población, total	SP.POP.TOTL	1904148.0	1971530.0	2039346.0	2106142.0
2	Samoa Americana	ASM	Población, total	SP.POP.TOTL	20085.0	20626.0	21272.0	21949.0
3	Antigua y Barbuda	ATG	Población, total	SP.POP.TOTL	55342.0	56245.0	57008.0	57778.0
4	Australia	AUS	Población, total	SP.POP.TOTL	10276477.0	10483000.0	10742000.0	10950000.0

5 rows × 67 columns

Tail

It will display last 5 rows of data frame.



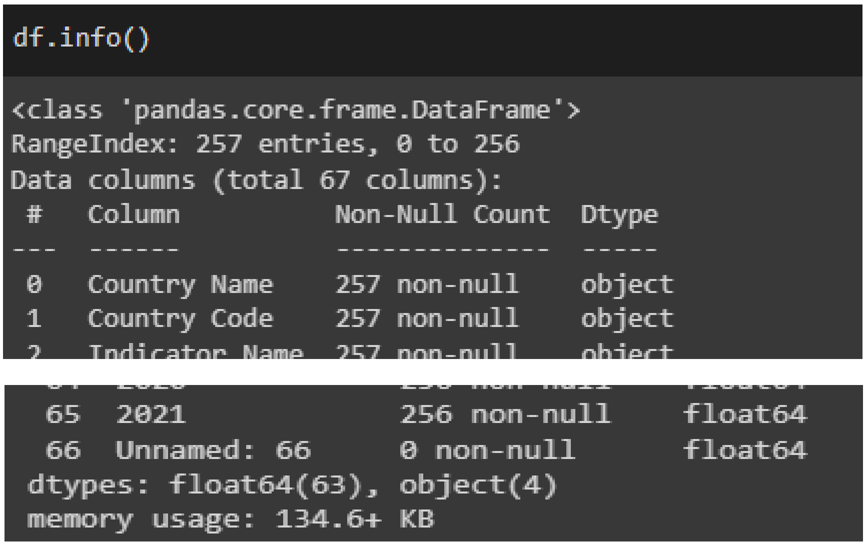
```
df.tail()
```

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963
252	Kosovo	XKX	Población, total	SP.POP.TOTL	947000.0	966000.0	994000.0	1022000.0
253	Yemen, Rep. del	YEM	Población, total	SP.POP.TOTL	5542459.0	5646668.0	5753386.0	5860197.0
254	Sudáfrica	ZAF	Población, total	SP.POP.TOTL	16520441.0	16989464.0	17503133.0	18042215.0
255	Zambia	ZMB	Población, total	SP.POP.TOTL	3119430.0	3219451.0	3323427.0	3431381.0
256	Zimbabwe	ZWE	Población, total	SP.POP.TOTL	3806310.0	3925952.0	4049778.0	4177931.0

5 rows × 67 columns

Info

Display information about Data Frame, it also tells data types and memory usage.



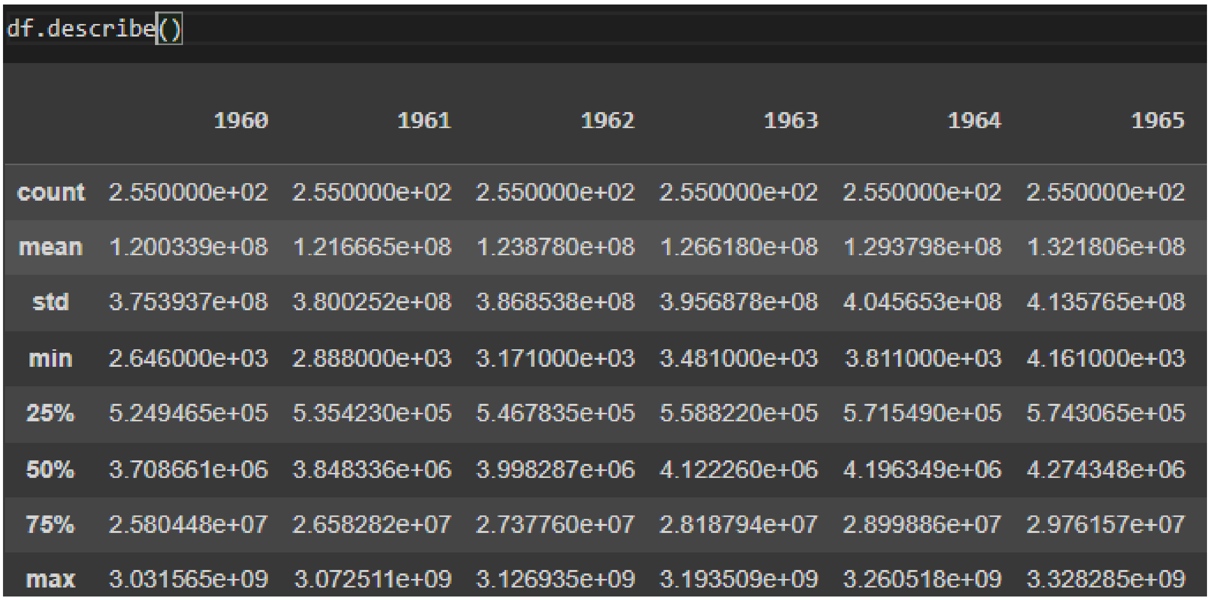
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 257 entries, 0 to 256
Data columns (total 67 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Country Name    257 non-null    object
 1   Country Code    257 non-null    object
 2   Indicator Name  257 non-null    object
```

```
 64  2020            256 non-null    float64
 65  2021            256 non-null    float64
 66  Unnamed: 66     0 non-null      float64
dtypes: float64(63), object(4)
memory usage: 134.6+ KB
```

Describe

Display summary statistics of numerical columns in a Data Frame.



```
df.describe()
```

	1960	1961	1962	1963	1964	1965
count	2.550000e+02	2.550000e+02	2.550000e+02	2.550000e+02	2.550000e+02	2.550000e+02
mean	1.200339e+08	1.216665e+08	1.238780e+08	1.266180e+08	1.293798e+08	1.321806e+08
std	3.753937e+08	3.800252e+08	3.868538e+08	3.956878e+08	4.045653e+08	4.135765e+08
min	2.646000e+03	2.888000e+03	3.171000e+03	3.481000e+03	3.811000e+03	4.161000e+03
25%	5.249465e+05	5.354230e+05	5.467835e+05	5.588220e+05	5.715490e+05	5.743065e+05
50%	3.708661e+06	3.848336e+06	3.998287e+06	4.122260e+06	4.196349e+06	4.274348e+06
75%	2.580448e+07	2.658282e+07	2.737760e+07	2.818794e+07	2.899886e+07	2.976157e+07
max	3.031565e+09	3.072511e+09	3.126935e+09	3.193509e+09	3.260518e+09	3.328285e+09

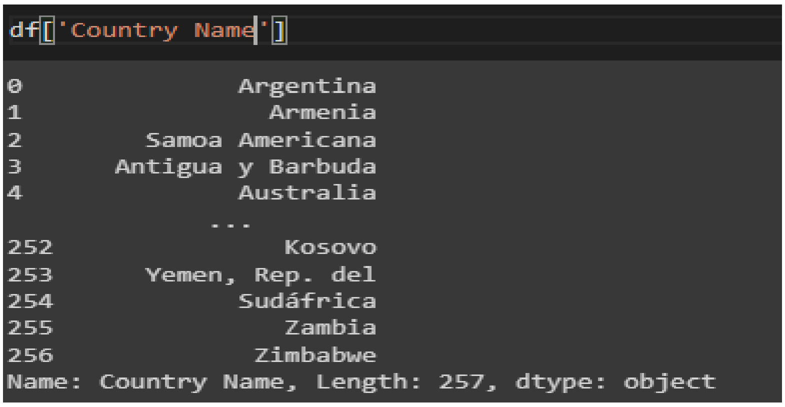
Data Selection

Data selection is an import step in data science. Using it we can performs calculations on specific data. Sometimes we are given a huge amount of data but only few are required for analysis. So this is a major and import step, instead of removing and recreating a dataset we just select and performs operations on required columns. Following are some methods used for selecting data from a data frame.

- **df[col]:** Select a single column by name as a *Series.*
- **df[[col1, col2]]:** Select multiple columns by name as a *Data Frame.*
- **df.loc[row, col]:** Select a single value by row and column name.
- **df.iloc[row, col]:** Select a single value by row and column *index*.

Df[col]

It will select a single column by name as a *Series*. Where *Series* is a data type in python.

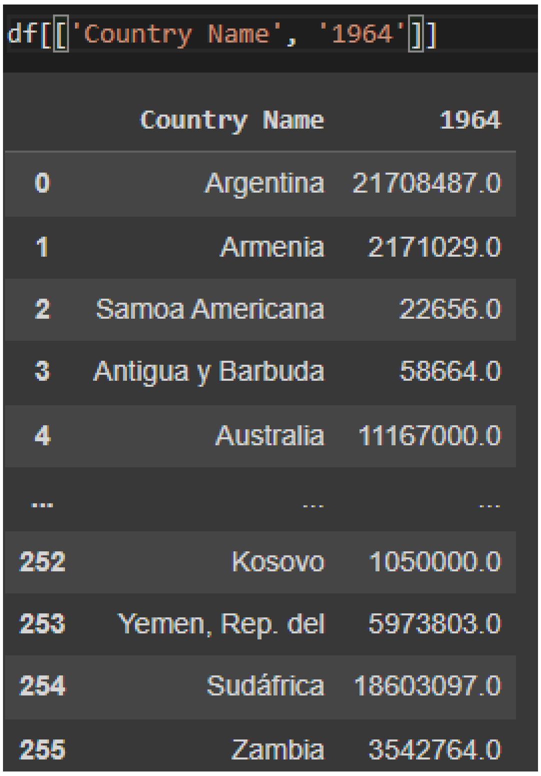


```
df['Country Name']

0              Argentina
1                Armenia
2        Samoa Americana
3      Antigua y Barbuda
4              Australia
             ...
252               Kosovo
253       Yemen, Rep. del
254            Sudáfrica
255               Zambia
256             Zimbabwe
Name: Country Name, Length: 257, dtype: object
```

Df[[col1, col2]]

Select multiple columns by name as a *Data Frame*.

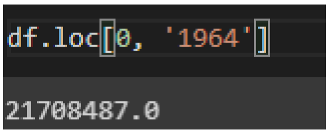


```
df[['Country Name', '1964']]
```

	Country Name	1964
0	Argentina	21708487.0
1	Armenia	2171029.0
2	Samoa Americana	22656.0
3	Antigua y Barbuda	58664.0
4	Australia	11167000.0
...	...	...
252	Kosovo	1050000.0
253	Yemen, Rep. del	5973803.0
254	Sudáfrica	18603097.0
255	Zambia	3542764.0

Df.loc[row, col]

It is used to select a single value by row and column name.

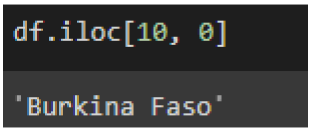


```
df.loc[0, '1964']
```

```
21708487.0
```

Df.iloc[row,col]

It is used to select a single value by row and column index.



```
df.iloc[10, 0]
```

```
'Burkina Faso'
```