CHAPTER – 7 - Class Starts at 1:05 PM (PKT)

MANIPULATING DATA FRAMES

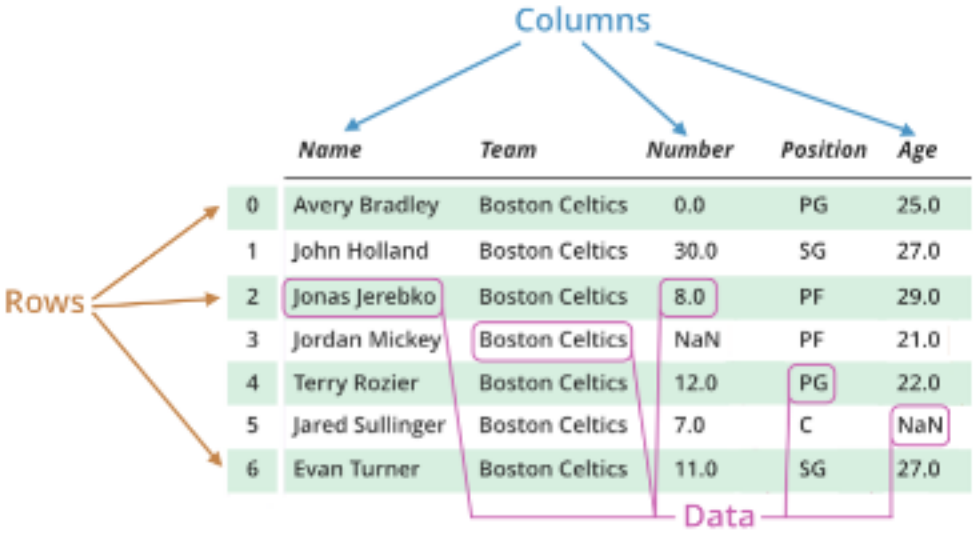
DATAFRAME - RECAP

Pandas DataFrame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns).

A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns.

Pandas DataFrame consists of three principal components, the data, rows, and columns.

Syntax: pd.DataFrame(“Dictionary Name”)



	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

MANIPULATING DATAFRAMES

Regardless of the original data source, once you have data loaded into a DataFrame, you gain the ability to manipulate your data. For instance, you can:

<u>Select rows using</u>

- Logical criteria
- head() and tail()
- iloc()

Select columns

Handle missing data with dropna() and fillna()

Make new columns

<u>Reshape a DataFrame</u>

- sort
- drop columns
- melt() and pivot()

Combine datasets with merge()

Group data

DATAFRAME STRUCTURE

The dataframe.shape() function gives dimensions of the array.

The dataframe.describe() function computes a summary of statistics pertaining to the DataFrame columns.

Pandas dataframe.info() function is used to get a concise summary of the dataframe.

Pandas dataframe.columns function is used to get a columns names of the dataframe.

TASK BACKGROUND

To explore this, we'll use a data frame consisting of the salaries and personal statistics of major league baseball players.

```
#accessing MLB players data
import pandas as pd
players_df = pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/players.csv")
```

TASK

What is the shape of the MLB dataframe players_df?

For this task use the following link to access MLB Dataframe

https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/players.csv

ANSWER

```
# Import pandas
import pandas as pd
# Read the file into a DataFrame: df
players_df = pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/players.csv")
# Print the shape of df
players_df.shape
```

TASK

Use the describe() function on dataframe players_df?

ANSWER

```
#describe players_df
players_df.describe()
```

TASK

Use the .info() function on dataframe players_df?

ANSWER

```
#info of players_df
 players_df.info()
```

TASK

Use the .columns to view columns names of dataframe players_df?

ANSWER

#columns names of dataframe

players_df. columns

MISSING VALUES - NULLS

By a null value, pandas means a value that is missing. Null values are represented in pandas as NaN.
A null value does not necessarily mean that the number is zero.
It could be missing due to recording error, its not being applicable, a sampling bias, etc.

The .isnull() method returns a Boolean result indicating whether each value in a DataFrame is missing.
True equates to a missing value.

TASK

Find which values are null in dataframe players_df?

ANSWER

```
# Import pandas
import pandas as pd
# Read the file into a DataFrame: df
players_df = pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/players.csv")
#null values in df
players_df.isnull()
```

DROPPING NULL VALUES

Immediately we can see that there are quite a few missing values in the deathyear field. It makes sense that there are missing values here; if a player is still living, then their death year is indeed unknown.

Making this connection shows the importance of understanding how data is collected; the data itself may not provide the answers.

Should we want to exclude all records with a missing value, we can call the .dropna() method. By default, this will drop any row which includes a missing value, in any field.

DROPPING NULL VALUES - EXAMPLE

Let’s create a filtered Dataframe by removing nulls players_df_filtered

# Import pandas

import pandas as pd

# Read the file into a DataFrame: df

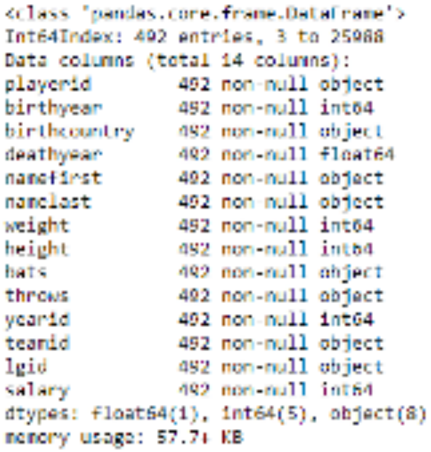
players_df = pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/players.csv")

#filtered dataframe

players_df_filtered = players_df.dropna()

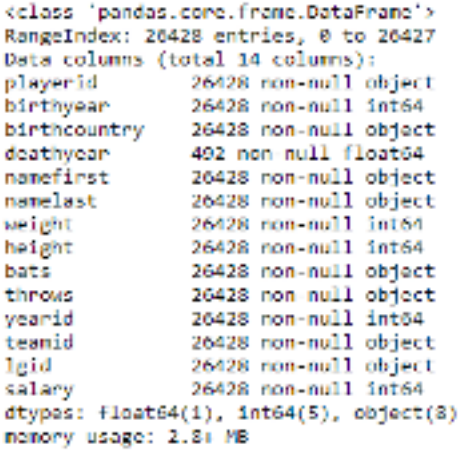
players_df_filtered.info()

#Out



```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 492 entries, 3 to 25988
Data columns (total 14 columns):
playerid        492 non-null object
birthyear       492 non-null int64
birthcountry    492 non-null object
deathyear       492 non-null float64
namefirst       492 non-null object
namelast        492 non-null object
weight          492 non-null int64
height          492 non-null int64
bats            492 non-null object
throws          492 non-null object
yearid          492 non-null int64
teamid          492 non-null object
lgid            492 non-null object
salary          492 non-null int64
dtypes: float64(1), int64(5), object(8)
memory usage: 57.7+ KB
```

Filtered Dataframe



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26428 entries, 0 to 26427
Data columns (total 14 columns):
playerid        26428 non-null object
birthyear       26428 non-null int64
birthcountry    26428 non-null object
deathyear       492 non-null float64
namefirst       26428 non-null object
namelast        26428 non-null object
weight          26428 non-null int64
height          26428 non-null int64
bats            26428 non-null object
throws          26428 non-null object
yearid          26428 non-null int64
teamid          26428 non-null object
lgid            26428 non-null object
salary          26428 non-null int64
dtypes: float64(1), int64(5), object(8)
memory usage: 2.8+ MB
```

Unfiltered Dataframe

FILLING VALUES

Our DataFrame now contains only 492 rows -- all rows where the player's death year is unknown have been dropped.

Let's say that instead of dropping these rows, we want to fill in the missing values with something other than NaN. We can do so with the fillna() method.

Syntax: df.fillna('Data You Want To Include')

This can be useful in making a DataFrame more legible, or in assigning a special value or character to nulls.

TASK

Fill the null values in the deathyear column to “Still alive” for dataframe players_df?

For this task use the following link to access MLB Dataframe

https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/players.csv

ANSWER

```
# Import pandas
import pandas as pd
# Read the file into a DataFrame: df
players_df = pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/players.csv")
#filling null values in df
players_df.fillna('Still alive')
```

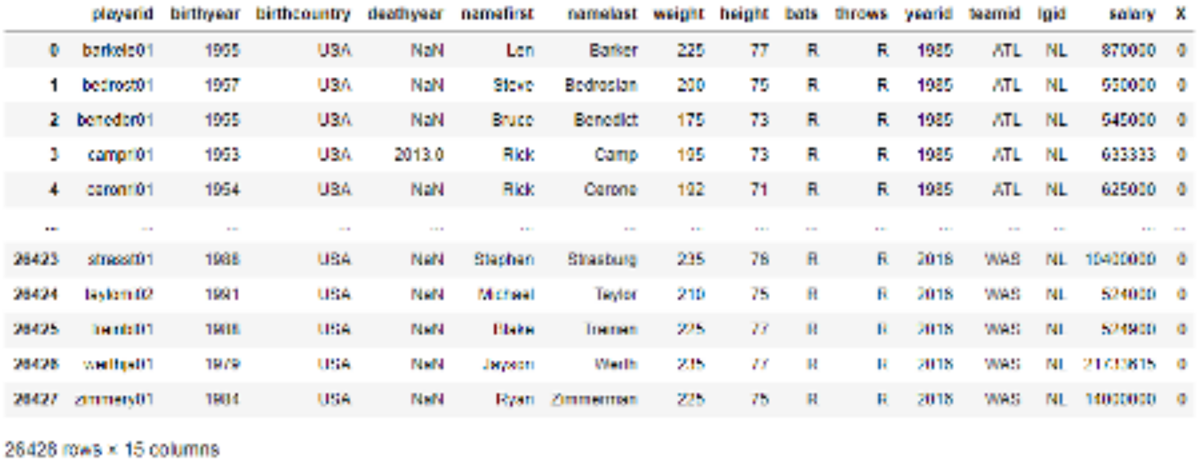
ADDING NEW COLUMNS TO A DATAFRAME

Sometimes, you'll need to add new columns by deriving values from calculations involving other columns.

To create a column of values calculated from the values in other columns use the assign() method of the DataFrame.

For example, lets add a column X with 0's as value to our dataframe

```
# Import pandas
import pandas as pd
# Read the file into a DataFrame: df
players_df = pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/players.csv")
#adding new column X
players_df['X'] = 0
players_df
```



```
        playerid  birthyear  birthcountry  deathyear  namefirst   namelast  weight  height  bats  throws  yearid  teamid  lgid    salary  X
    0   barkele01      1955           USA        NaN        Len     Barker     225      77     R       R    1985     ATL    NL    870000  0
    1   bedrost01      1957           USA        NaN      Steve  Bedrosian     200      75     R       R    1985     ATL    NL    550000  0
    2   benedbr01      1955           USA        NaN      Bruce   Benedict     175      73     R       R    1985     ATL    NL    545000  0
    3    campri01      1953           USA     2013.0       Rick       Camp     195      73     R       R    1985     ATL    NL    633333  0
    4   caronri01      1954           USA        NaN       Rick     Cerone     192      71     R       R    1985     ATL    NL    625000  0
  ...         ...       ...           ...        ...        ...        ...     ...     ...   ...     ...     ...     ...   ...       ...  ...
26423   strasst01      1988           USA        NaN    Stephen   Strasburg     235      76     R       R    2016     WAS    NL  10400000  0
26424   taylomi02      1991           USA        NaN    Michael     Taylor     210      75     R       R    2016     WAS    NL    524000  0
26425   treinbl01      1988           USA        NaN      Blake     Treinen     225      77     R       R    2016     WAS    NL    524900  0
26426   werthja01      1979           USA        NaN     Jayson      Werth     235      77     R       R    2016     WAS    NL  21733615  0
26427   zimmery01      1984           USA        NaN       Ryan  Zimmerman     225      75     R       R    2016     WAS    NL  14000000  0

26428 rows × 15 columns
```

TASK

We have data on the weight and height of all players on a team, we could compute a new column for body mass index (BMI) by using a formula given below. Use the weight & height columns to create a new column BMI in your MLB dataframe.

$$bmi = \frac{weight * 703}{height^2}$$

For this task use the following link to access MLB Dataframe

https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/players.csv

ANSWER

```
#adding a new column BMI
players_df = players_df.assign(bmi = (703 * players_df['weight']) / (players_df['height']**2))
players_df
```

DELETING COLUMNS FROM A DATAFRAME

Pandas provide data analysts a way to delete and filter data frame using .drop() method.

Rows or columns can be removed using index label or column name using this method.

Syntax:DataFrame.drop(labels=None, axis=0)

Parameters

labels: String or list of strings referring row or column name.

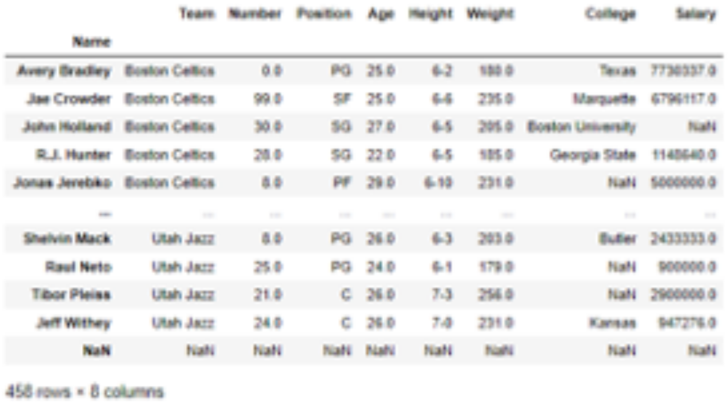
axis: int or string value, 0 ‘index’ for Rows and 1 ‘columns’ for Columns.

DELETING COLUMNS FROM A DATAFRAME - EXAMPLE

NBA Dataframe: Dropping columns with column name. In this code, columns are dropped using column names. Axis parameter is kept as 1 since 1 refers to columns.

```
# importing pandas module
import pandas as pd
data=pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/nba.csv",
 index_col ="Name" )
# dropping passed columns
dropcolum=data.drop(["Team", "Weight"], axis = 1)
# display
dropcolum
```

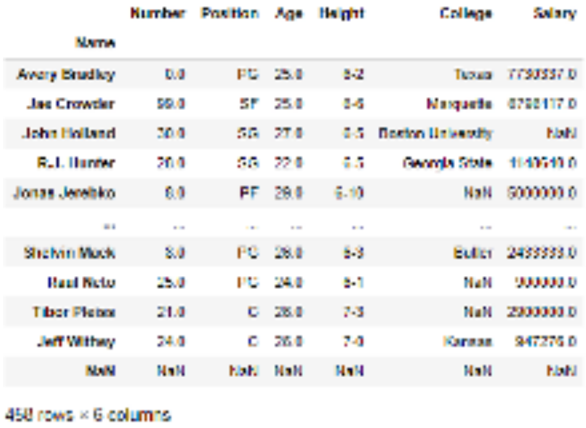
Data Frame Before Dropping Columns



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

458 rows × 8 columns

Data Frame After Dropping Columns



Name	Number	Position	Age	Height	College	Salary
Avery Bradley	0.0	PG	25.0	6-2	Texas	7730337.0
Jae Crowder	99.0	SF	25.0	6-6	Marquette	6796117.0
John Holland	30.0	SG	27.0	6-5	Boston University	NaN
R.J. Hunter	28.0	SG	22.0	6-5	Georgia State	1148640.0
Jonas Jerebko	8.0	PF	29.0	6-10	NaN	5000000.0
...	...	...	...	...	...	...
Shelvin Mack	8.0	PG	26.0	6-3	Butler	2433333.0
Raul Neto	25.0	PG	24.0	6-1	NaN	900000.0
Tibor Pleiss	21.0	C	26.0	7-3	NaN	2900000.0
Jeff Withey	24.0	C	26.0	7-0	Kansas	947276.0
NaN	NaN	NaN	NaN	NaN	NaN	NaN

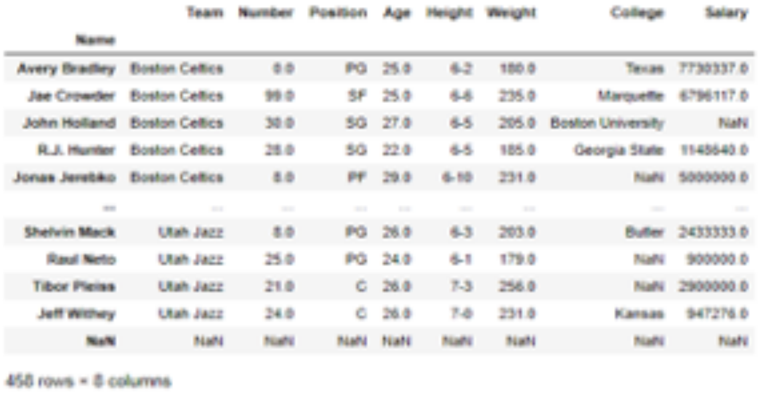
458 rows × 6 columns

DELETING ROWS FROM A DATAFRAME - EXAMPLE

Using the NBA Dataframe, a list of index labels is passed and the rows corresponding to those labels are dropped using .drop() method.

```
# importing pandas module

import pandas as pd

# making data frame from csv file

data=pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/nba.csv",

index_col ="Name" )

# dropping passed values

data.drop(["Avery Bradley", "John Holland", "R.J. Hunter"],axis=0)

# display

data
```

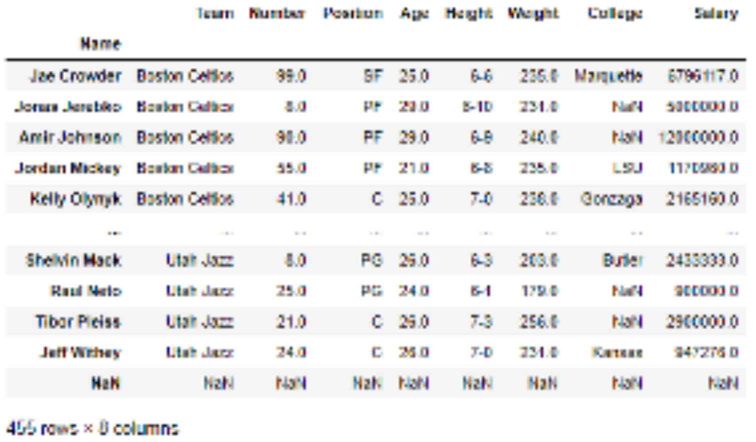
Data Frame before Dropping Rows



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

458 rows × 8 columns

Data Frame After Dropping Rows



Name	Team	Number	Position	Age	Height	Weight	College	Salary
Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	6796117.0
Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0
Amir Johnson	Boston Celtics	90.0	PF	29.0	6-9	240.0	NaN	12000000.0
Jordan Mickey	Boston Celtics	55.0	PF	21.0	6-8	235.0	LSU	1170960.0
Kelly Olynyk	Boston Celtics	41.0	C	25.0	7-0	238.0	Gonzaga	2165160.0
...	...	...	...	...	...	...	...	...
Shelvin Mack	Utah Jazz	8.0	PG	26.0	6-3	203.0	Butler	2433333.0
Raul Neto	Utah Jazz	25.0	PG	24.0	6-1	179.0	NaN	900000.0
Tibor Pleiss	Utah Jazz	21.0	C	26.0	7-3	256.0	NaN	2900000.0
Jeff Withey	Utah Jazz	24.0	C	26.0	7-0	231.0	Kansas	947276.0
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

455 rows × 8 columns

TASK

Delete the column lgid and store the new DataFrame players_df_changed. Print the column list before and after deleting column lgid.

ANSWER

```
#deleting columns
players_df_changed= players_df.drop(['X'], axis=1)
#printing column list after deletion
print(players_df_changed.columns)
#printing column list before deletion
print(players_df.columns)
```

HEAD & TAIL

Python is a great language for doing data analysis, primarily because of the fantastic ecosystem of data-centric Python packages.

You’re already familiar with the Pandas package and how it makes importing and analyzing data easier.

head() method is used to return top n rows (5 by default) of a data frame.

tail() method is used to return bottom n rows(5 by default) of a data frame.

TASK

Print the first 5 rows of the MLB Dataframe.

ANSWER

```
#selecting first 5 rows
firstRows=players_df.head()
firstRows
```

TASK

Print the last 5 rows of the MLB Dataframe.

ANSWER

```
#selecting last 5 rows
lastRows=players_df.tail()
lastRows
```

SELECTING DATA

One thing you'll need to do constantly when exploring data in Pandas is selecting a subset of rows based on some criterion.

Suppose, for instance, that you need to see the data for just a single year? Or maybe you need to select some rows by numbers?

Selecting by Number

We have already seen that you can get the first n rows or the last n rows of a DataFrame using head(n) and tail(n) respectively.

iloc may take inputs in a number of different ways:

an integer

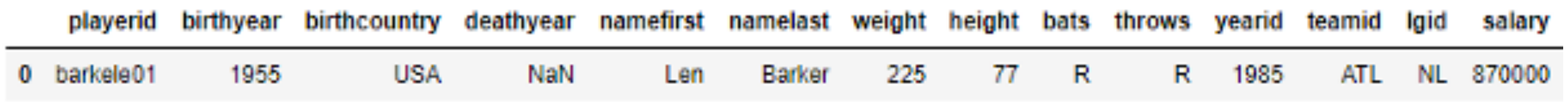
a list of of integers

a slice object

a boolean array

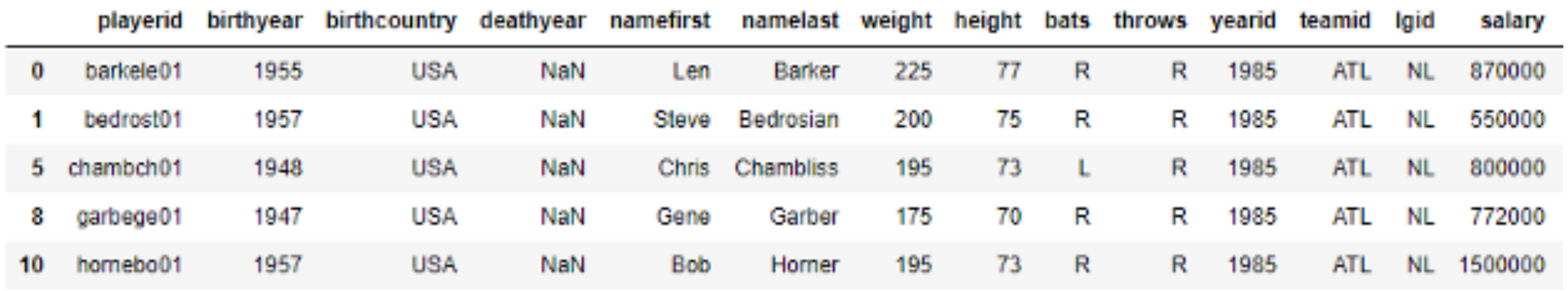
SELECTING ROWS - EXAMPLES

```
# select the first row
players_df.iloc[[0]]
```



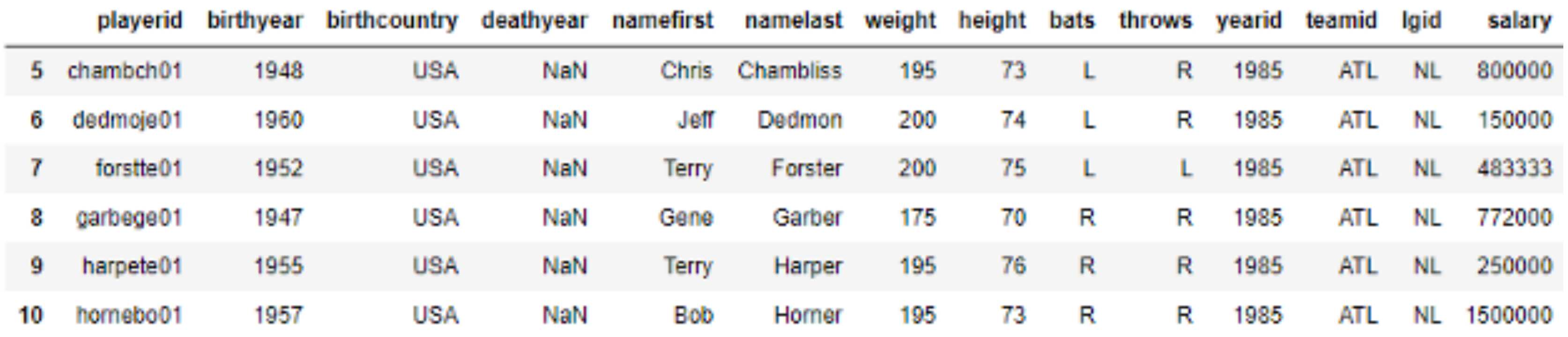
	playerid	birthyear	birthcountry	deathyear	namefirst	namelast	weight	height	bats	throws	yearid	teamid	lgid	salary
0	barkele01	1955	USA	NaN	Len	Barker	225	77	R	R	1985	ATL	NL	870000

```
#selecting more than one row
players_df.iloc[[0, 1, 5, 8, 10]]
```



	playerid	birthyear	birthcountry	deathyear	namefirst	namelast	weight	height	bats	throws	yearid	teamid	lgid	salary
0	barkele01	1955	USA	NaN	Len	Barker	225	77	R	R	1985	ATL	NL	870000
1	bedrost01	1957	USA	NaN	Steve	Bedrosian	200	75	R	R	1985	ATL	NL	550000
5	chambch01	1948	USA	NaN	Chris	Chambliss	195	73	L	R	1985	ATL	NL	800000
8	garbege01	1947	USA	NaN	Gene	Garber	175	70	R	R	1985	ATL	NL	772000
10	hornebo01	1957	USA	NaN	Bob	Horner	195	73	R	R	1985	ATL	NL	1500000

```
# Select rows 5 to 10
players_df.iloc[5:11]
```



	playerid	birthyear	birthcountry	deathyear	namefirst	namelast	weight	height	bats	throws	yearid	teamid	lgid	salary
5	chambch01	1948	USA	NaN	Chris	Chambliss	195	73	L	R	1985	ATL	NL	800000
6	dedmoje01	1960	USA	NaN	Jeff	Dedmon	200	74	L	R	1985	ATL	NL	150000
7	forstte01	1952	USA	NaN	Terry	Forster	200	75	L	L	1985	ATL	NL	483333
8	garbege01	1947	USA	NaN	Gene	Garber	175	70	R	R	1985	ATL	NL	772000
9	harpete01	1955	USA	NaN	Terry	Harper	195	76	R	R	1985	ATL	NL	250000
10	hornebo01	1957	USA	NaN	Bob	Horner	195	73	R	R	1985	ATL	NL	1500000

TASK

Select row number 2, 4, 8, 16, 32, 64, 128, 264 and print the result.

ANSWER

```
#selecting more than one row
players_df.iloc[[2, 4, 8, 16, 32, 64, 128, 264]]
```

TASK

Select rows 150 to 201 and print the result.

ANSWER

```
#select rows 150 to 200
players_df.iloc[150:201]
```

TASK

Select rows 10 to 15 and columns 1 to 4 and print the result.

ANSWER

```
# Select rows 10 to 15 and columns 1 to 4
players_df.iloc[10:16,1:5]
```

SORTING

Sorting data is one of the most important task when you’re working with data. To sort a DataFrame, we use sort_values().

It's possible to sort by the values in one or more columns and to sort either in ascending order or descending order.

Parameters

by: either the name of a single column or a list of names of columns

axis: either 0 to sort rows or 1 to sort columns, default 0.

ascending: True, or False for descending, defaults to True

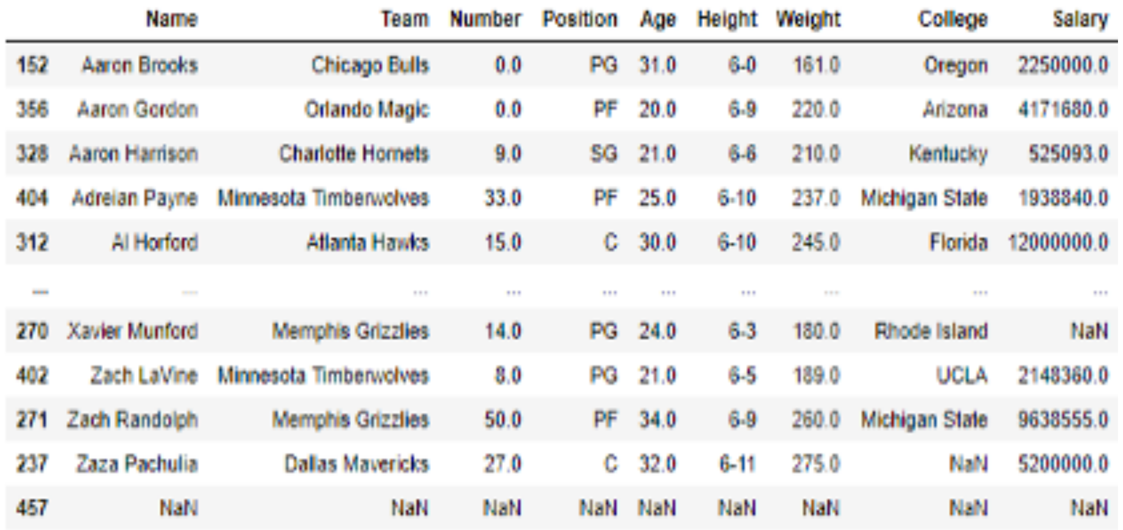
inplace: True to sort inplace and modify the DataFrame, False to create a new DataFrame, defaults to False

SORTING - EXAMPLE

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.

```
# importing pandas package

import pandas as pd

# making data frame from csv file

data = pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/nba.csv")

# sorting data frame by name

data.sort_values("Name", axis = 0, ascending =  True, inplace = True)

# display

data
```

After Sorting the Name Column



	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

TASK

Sort the players by weight from lowest value to highest value (ascending)

ANSWER

```
# sorting data frame by name
players_df.sort_values("weight", axis = 0, ascending =  True, inplace = True)
# display
players_df
```

TASK

Sort the players by salary from highest value to lowest value (descending)

ANSWER

```
# sorting data frame by name
players_df.sort_values("salary", axis = 0, ascending =  False, inplace = True)
# display
players_df
```

TASK

Sort the players by weight & height from lowest value to highest value (ascending)

ANSWER

```
# sorting data frame by name
players_df.sort_values(by=['weight', 'height'], axis = 0, ascending =  True, inplace = True)
# display
players_df
```

GROUPING

Pandas dataframe.groupby() function is used to split the data into groups based on some criteria.

Pandas dataframe can be split on any axes.

The definition of grouping is to provide a mapping of labels to group names.

Syntax: DataFrame.groupby(by=None, axis=0)

Parameters

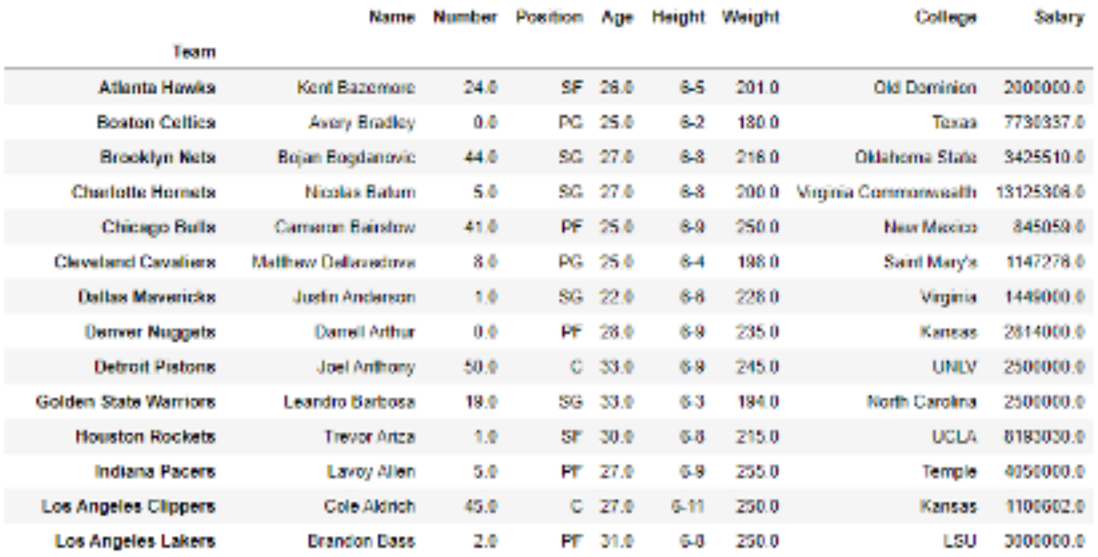
by : mapping, function, str, or iterable

axis : int, default 0

GROUPING - EXAMPLE

Example: Using groupby() function to group the data based on the "Team"

```
# importing pandas package
import pandas as pd
# making data frame from csv file
data = pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/nba.csv",index_col='Team')
# applying groupby() function to  group the data on team value.
group=data.groupby(['Team'])
# Let's print the first entries in all the groups formed.
group.first()
```

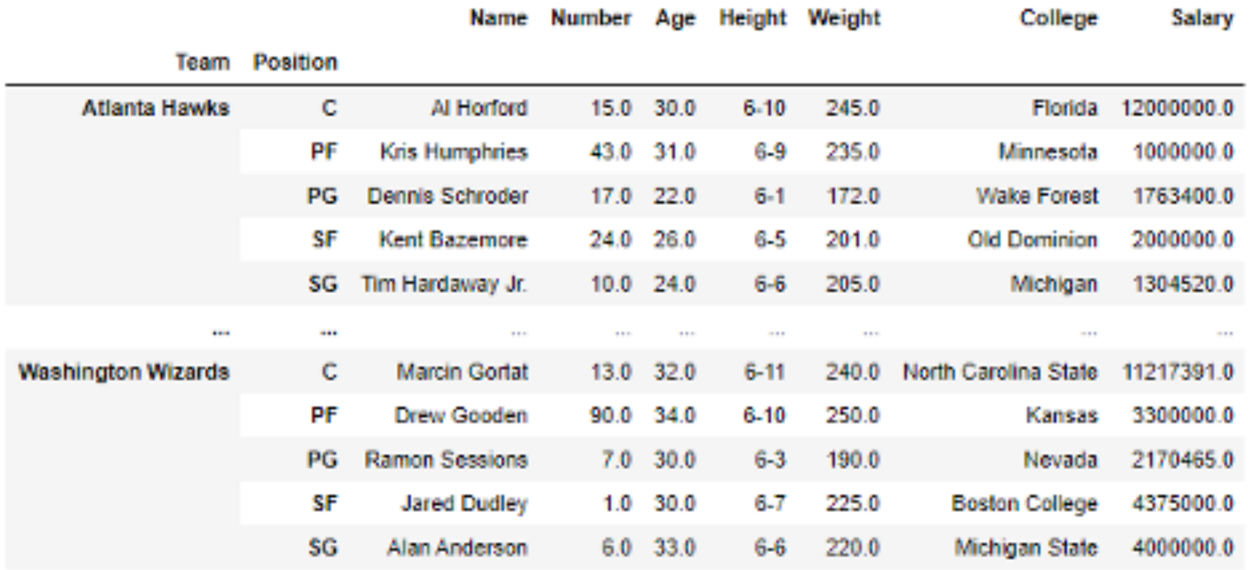


	Name	Number	Position	Age	Height	Weight	College	Salary
Team								
Atlanta Hawks	Kent Bazemore	24.0	SF	26.0	6-5	201.0	Old Dominion	2000000.0
Boston Celtics	Avery Bradley	0.0	PG	25.0	6-2	180.0	Texas	7730337.0
Brooklyn Nets	Bojan Bogdanovic	44.0	SG	27.0	6-8	216.0	Oklahoma State	3425510.0
Charlotte Hornets	Nicolas Batum	5.0	SG	27.0	6-8	200.0	Virginia Commonwealth	13125306.0
Chicago Bulls	Cameron Bairstow	41.0	PF	25.0	6-9	250.0	New Mexico	845059.0
Cleveland Cavaliers	Matthew Dellavedova	8.0	PG	25.0	6-4	198.0	Saint Mary's	1147276.0
Dallas Mavericks	Justin Anderson	1.0	SG	22.0	6-6	228.0	Virginia	1449000.0
Denver Nuggets	Darrell Arthur	0.0	PF	28.0	6-9	235.0	Kansas	2814000.0
Detroit Pistons	Joel Anthony	50.0	C	33.0	6-9	245.0	UNLV	2500000.0
Golden State Warriors	Leandro Barbosa	19.0	SG	33.0	6-3	194.0	North Carolina	2500000.0
Houston Rockets	Trevor Ariza	1.0	SF	30.0	6-8	215.0	UCLA	8193030.0
Indiana Pacers	Lavoy Allen	5.0	PF	27.0	6-9	255.0	Temple	4050000.0
Los Angeles Clippers	Cole Aldrich	45.0	C	27.0	6-11	250.0	Kansas	1100602.0
Los Angeles Lakers	Brandon Bass	2.0	PF	31.0	6-8	250.0	LSU	3000000.0

GROUPING - EXAMPLE

Example: Use groupby() function to form a group based on more than one category Teams & Positions.

```
# importing pandas package
import pandas as pd
# making data frame from csv file
data = pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/nba.csv",index_col='Team')
# applying groupby() function to  group the data on team value.
group=data.groupby(['Team','Position'])
# Let's print the first entries in all the groups formed.
group.first()
```



		Name	Number	Age	Height	Weight	College	Salary
Team	Position							
Atlanta Hawks	C	Al Horford	15.0	30.0	6-10	245.0	Florida	12000000.0
	PF	Kris Humphries	43.0	31.0	6-9	235.0	Minnesota	1000000.0
	PG	Dennis Schroder	17.0	22.0	6-1	172.0	Wake Forest	1763400.0
	SF	Kent Bazemore	24.0	26.0	6-5	201.0	Old Dominion	2000000.0
	SG	Tim Hardaway Jr.	10.0	24.0	6-6	205.0	Michigan	1304520.0
...	...	...	...	...	...	...	...	...
Washington Wizards	C	Marcin Gortat	13.0	32.0	6-11	240.0	North Carolina State	11217391.0
	PF	Drew Gooden	90.0	34.0	6-10	250.0	Kansas	3300000.0
	PG	Ramon Sessions	7.0	30.0	6-3	190.0	Nevada	2170465.0
	SF	Jared Dudley	1.0	30.0	6-7	225.0	Boston College	4375000.0
	SG	Alan Anderson	6.0	33.0	6-6	220.0	Michigan State	4000000.0

TASK

Group the players dataframe based on yearid and apply .mean() function.

print the results.

ANSWER

```
# group data frame by yearid
groupYear=players_df. groupby(['yearid']).mean()
# display
groupYear
```

TASK

Group the players dataframe based on teamid & bats & apply the describe() function on salary and print the results.

ANSWER

```
# group data frame by teamid & bats
groupTeam=players_df. groupby(['teamid', 'bats'])['salary'].describe()
# display
groupTeam
```

TASK

From the players dataframe what is the number of players in each team of Major League Baseball Team.

ANSWER

```
#count number of player
number_player=players_df.groupby(['teamid'])['playerid'].count()
number_player
```

TASK

From the players dataframe find how many players have played for (teamid) ATL.

ANSWER

```
players=players_df[players_df['teamid']=='ATL']
#count number of player in ATL
player_count=players.groupby(['teamid'])['playerid'].count()
player_count
```

TASK

From the players dataframe what is the average salary for MLB Teams.

ANSWER

#find average salary of MLB teams

players_df.groupby (['teamid'])['salary'].mean()

TASK

From the players dataframe what is the average salary after year (yearid) 2000 for MLB Teams.

ANSWER

```
players=players_df[players_df['yearid']>2000]
averageYearSalary=players.groupby(['teamid'])['salary'].mean()
averageYearSalary
```

RESHAPING THE DATAFRAME

Pandas uses various methods to reshape the dataframe and series. Let’s see about the some of that reshaping method.

pivot() function produces pivot table based on 3 columns of the DataFrame

melt() function is useful to massage a DataFrame into a format where one or more columns are identifier variables

RESHAPING THE DATAFRAME – PIVOT()

pandas.pivot(index, columns, values) function produces pivot table based on 3 columns of the DataFrame. It uses unique values from index / columns and fills with values.

Parameters

index[ndarray] : Labels to use to make new frame's index

columns[ndarray] : Labels to use to make new frame's columns

values[ndarray] : Values to use for populating new frame's values

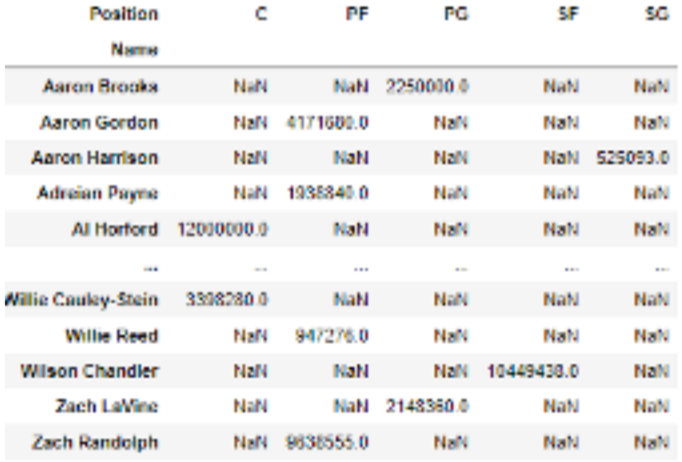
RESHAPING THE DATAFRAME – PIVOT() EXAMPLE

The data that we have consists of a list of NBA players and their Positions in the team.

<u>To create a pivot table</u>

1. We will use players name to specify the index, that is, which column will be used to identify each row of the DataFrame.
2. Then, we use Position to specify which column values are going to become the column names of the table.
3. Finally, we Salary to specify which column will provide the values that go in the table cells.

```
Non_duplicate = data.drop_duplicates(['Name', 'Position'])
povitTable= Non_duplicate.pivot(index='Name', columns='Position', values= Salary')
PovitTable
```



Position	C	PF	PG	SF	SG
Name					
Aaron Brooks	NaN	NaN	2250000.0	NaN	NaN
Aaron Gordon	NaN	4171680.0	NaN	NaN	NaN
Aaron Harrison	NaN	NaN	NaN	NaN	525093.0
Adreian Payne	NaN	1938840.0	NaN	NaN	NaN
Al Horford	12000000.0	NaN	NaN	NaN	NaN
...	...	...	...	...	...
Willie Cauley-Stein	3398280.0	NaN	NaN	NaN	NaN
Willie Reed	NaN	947276.0	NaN	NaN	NaN
Wilson Chandler	NaN	NaN	NaN	10449438.0	NaN
Zach LaVine	NaN	NaN	2148360.0	NaN	NaN
Zach Randolph	NaN	9638555.0	NaN	NaN	NaN

<u>Important consideration</u>

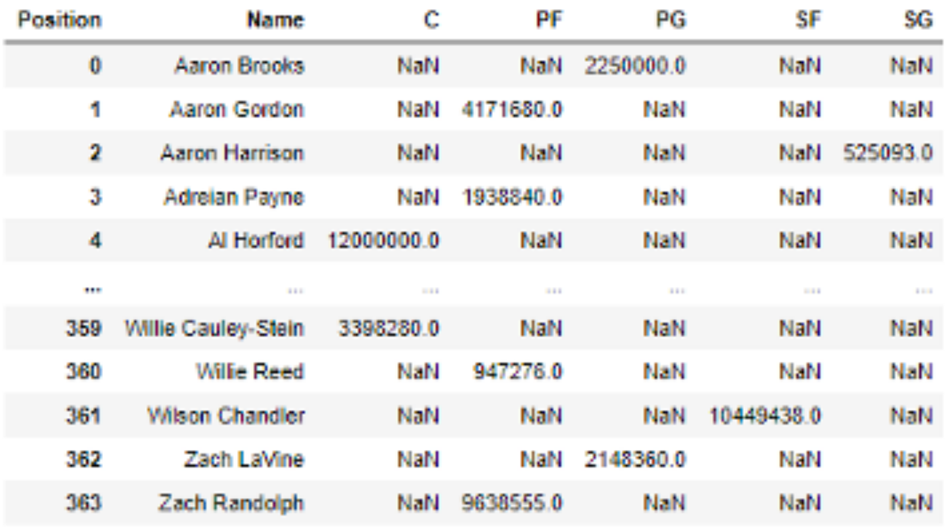
If duplicate values exist in dataframe pivot method return an error. To solve this problem remove duplicate values using .drop_duplicates() before apply pivot method.

RESHAPING THE DATAFRAME – PIVOT() EXAMPLE

You may notice that the DataFrame now has Name placed on top of Position

To pivot the data so that it appears in a more familiar tabular format, use the .reset_index() method after you pivot() the DataFrame:

```
Non_duplicate = data.drop_duplicates(['Name', 'Position'])
pivotTable=Non_duplicate.pivot(index='Name',columns='Position',values=Salary'). reset_index()
pivotTable
```



Position	Name	C	PF	PG	SF	SG
0	Aaron Brooks	NaN	NaN	2250000.0	NaN	NaN
1	Aaron Gordon	NaN	4171680.0	NaN	NaN	NaN
2	Aaron Harrison	NaN	NaN	NaN	NaN	525093.0
3	Adreian Payne	NaN	1938840.0	NaN	NaN	NaN
4	Al Horford	12000000.0	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...
359	Willie Cauley-Stein	3398280.0	NaN	NaN	NaN	NaN
360	Willie Reed	NaN	947276.0	NaN	NaN	NaN
361	Wilson Chandler	NaN	NaN	NaN	10449438.0	NaN
362	Zach LaVine	NaN	NaN	2148360.0	NaN	NaN
363	Zach Randolph	NaN	9638555.0	NaN	NaN	NaN

TASK

Create a pivot table years_pivot on MLB players_df dataframe. For this task use columns='yearid', values='salary', aggfunction = ‘mean’.

Answer

```
pivottable=pd.pivot_table(players_df,values='salary',index='teamid',columns='yearid',aggfunc='mean')
pivottable
```

TASK

Create a Pivot Table which provides the Number of Players by Team per Year.

ANSWER

```
# Import pandas
import pandas as pd
# Read the file into a DataFrame: df
players_df = pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/players.csv")
#getting rid of duplicates
years = players_df.drop_duplicates(['namefirst', 'yearid'])
#creating the pivot table
years_pivot = years.pivot(index='namefirst', columns='yearid', values='salary')
#print
years_pivot
```

TASK

Aapply .reset_index() method on years_pivot pivot table created on last task also save on five_years_pivot and print the result.

ANSWER

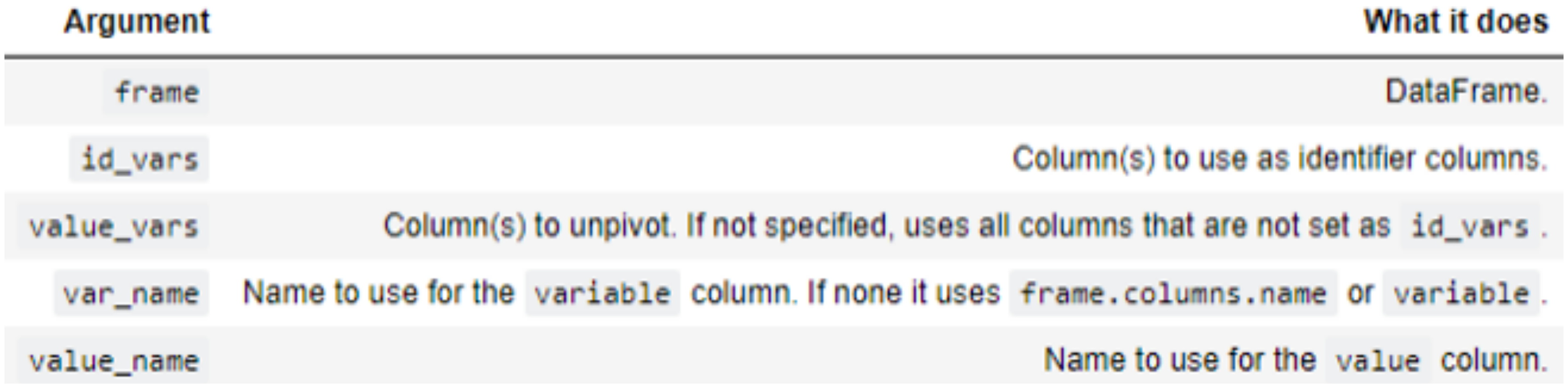
```
#apply reset_index()
five_years_pivot=years_pivot.reset_index()
```

RESHAPING THE DATAFRAME – MELT()

With the melt() function we can re-shape a dataframe.

Syntax: pandas.melt(frame, id_vars=None, value_vars=None, var_name=None, value_name='value')

Here's what each argument does, according to pandas documentation



Argument	What it does
`frame`	DataFrame.
`id_vars`	Column(s) to use as identifier columns.
`value_vars`	Column(s) to unpivot. If not specified, uses all columns that are not set as `id_vars` .
`var_name`	Name to use for the `variable` column. If none it uses `frame.columns.name` or `variable` .
`value_name`	Name to use for the `value` column.

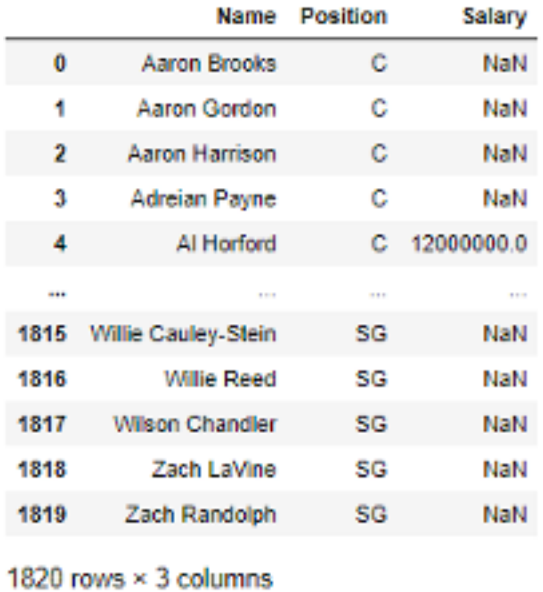
RESHAPING THE DATAFRAME – MELT()

Using melt(), we can "un-pivot" the above PovitTable dataframe

Rather than having five columns of position indicating salary information, we just want one column called position and another called salary.

That means that Name is our id_vars, and the rest are our value_vars. We'll set var_name to Position and value_name to Salary.

```
pd.melt(frame=povittest,id_vars=['Name'],var_name='Position',value_name='Salary')
```



	Name	Position	Salary
0	Aaron Brooks	C	NaN
1	Aaron Gordon	C	NaN
2	Aaron Harrison	C	NaN
3	Adreian Payne	C	NaN
4	Al Horford	C	12000000.0
...	...	...	...
1815	Willie Cauley-Stein	SG	NaN
1816	Willie Reed	SG	NaN
1817	Wilson Chandler	SG	NaN
1818	Zach LaVine	SG	NaN
1819	Zach Randolph	SG	NaN

1820 rows × 3 columns

TASK

Using melt(), how can we "un-pivot" the years_pivot you created in the last task?

Rather than having five columns for 2011-2015 indicating salary information, we just want one column called year and another called salary.

ANSWER

```
# create the melt table form pivot table
year_melt=pd.melt(frame=five_years_pivot,id_vars=['teamid'],var_name='yearid',value_name='salary')
year_melt
```

MERGING DATAFRAMES

Joining two or more tables is one of the most common data preparation tasks, and there are many ways to do it. In Pandas joins are modeled based on those in SQL.

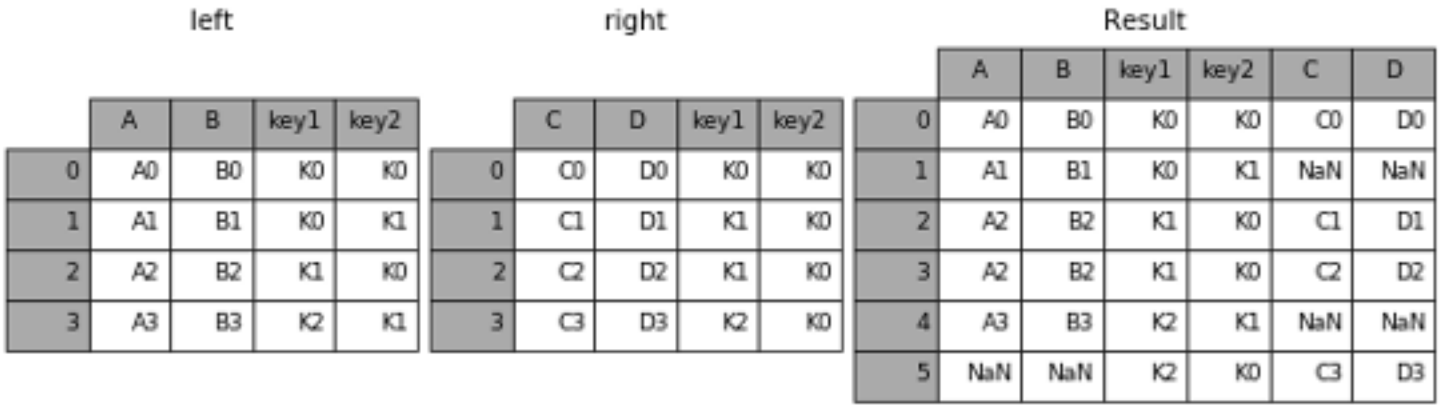
Pandas provides a single function, merge(), as the entry point for all standard database join operations between DataFrame objects.

Syntax: pd.merge(left, right)

Parameters

left − A DataFrame object.

right − Another DataFrame object.



left

	A	B	key1	key2
0	A0	B0	K0	K0
1	A1	B1	K0	K1
2	A2	B2	K1	K0
3	A3	B3	K2	K1

right

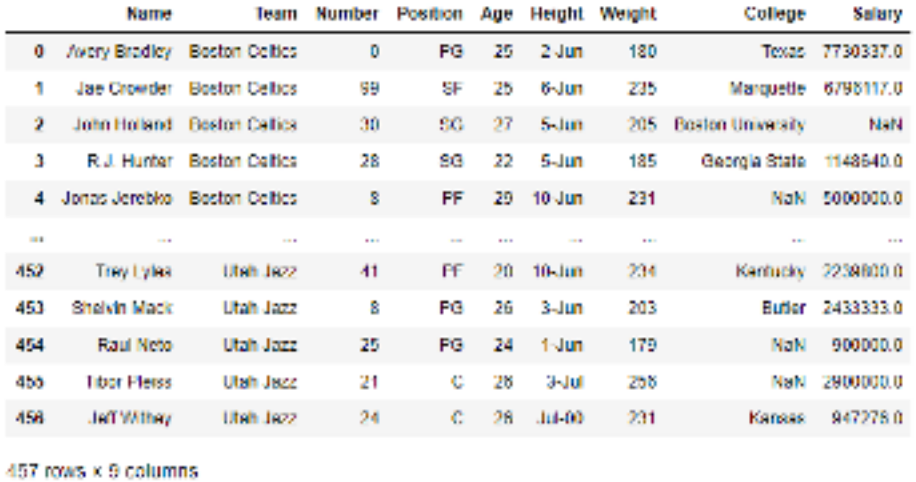
	C	D	key1	key2
0	C0	D0	K0	K0
1	C1	D1	K1	K0
2	C2	D2	K1	K0
3	C3	D3	K2	K0

Result

	A	B	key1	key2	C	D
0	A0	B0	K0	K0	C0	D0
1	A1	B1	K0	K1	NaN	NaN
2	A2	B2	K1	K0	C1	D1
3	A2	B2	K1	K0	C2	D2
4	A3	B3	K2	K1	NaN	NaN
5	NaN	NaN	K2	K0	C3	D3

MERGING DATAFRAMES - EXAMPLE

We have two datasets for NBA Players with their info and NBA Players and their salaries. Let’s merge these datasets in one dataframe.



	Name	Team	Number	Position	Age	Height	Weight	College	Salary
0	Avery Bradley	Boston Celtics	0	PG	25	2-Jun	180	Texas	7730337.0
1	Jae Crowder	Boston Celtics	99	SF	25	6-Jun	235	Marquette	6796117.0
2	John Holland	Boston Celtics	30	SG	27	5-Jun	205	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28	SG	22	5-Jun	185	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8	PF	29	10-Jun	231	NaN	5000000.0
...	...	...	...	...	...	...	...	...	...
452	Trey Lyles	Utah Jazz	41	PF	20	10-Jun	234	Kentucky	2239800.0
453	Shelvin Mack	Utah Jazz	8	PG	26	3-Jun	203	Butler	2433333.0
454	Raul Neto	Utah Jazz	25	PG	24	1-Jun	179	NaN	900000.0
455	Tibor Pleiss	Utah Jazz	21	C	26	3-Jul	256	NaN	2900000.0
456	Jeff Withey	Utah Jazz	24	C	26	Jul-00	231	Kansas	947276.0

457 rows × 9 columns

```
#import numpy as np
import numpy as np
#import pandas as pd
import pandas as pd
#access players data from the csv file.
salaries=pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/nbaPayersData.csv")
players=pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/nabPlayersSalaryData.csv")
#marge to dataset in one dataframe
merged = pd.merge(salaries, players)
#print mered
merged
```

You have two datasets with MLB players info and salary, merge these two together

Use the following datasets in this task:

salaries =
pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/salaries.csv")

people =
pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/people.csv")

ANSWER

```
#import numpy as np
import numpy as np
#import pandas as pd
import pandas as pd
#access payers data from the csv file.
salaries=pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/salaries.csv")
people=pd.read_csv("https://raw.githubusercontent.com/Masadn/PythonCourse/master/dataset/people.csv")
#marge to dataset in one dataframe
merged = pd.merge(salaries, people)
#print mered
merged
```

TASK

Select the players where their weight is between 210 to 230 and print the result.

ANSWER

```
players_df.loc[(players_df['weight'].between(210,230))]
```

TASK

Select the players who joined the league between (yearid) 2011 to 2016.

ANSWER

```
players_df.loc[(players_df['yearid'].between(2011,2016))]
```

TASK

Select players who have a bmi between 22 to 26, and print the result.

ANSWER

```
players_df.loc[players_df['bmi'].between(22,26)]
```

TASK

Select the players where the height is greater then 50 and their weight is greater then 200, print the result.

ANSWER

```
players_df.loc[(players_df['height'] > 50) & (players_df['weight'] >200)]
```

TASK

Select the players who were born between (birthyear) 1955 to 1988 and their weight is between 210 to 230, print the result.

ANSWER

```
players_df.loc[(players_df['birthyear'].between(1955,1988)) &
(players_df['weight'] .between(210,230))]
```

TASK

Select the players who joined the league after (yearid) 2007 or were born after (BirthYear) 1990.

ANSWER

```
players_df.loc[(players_df['yearid'] < 2007) | (players_df['birthyear'] > 1990)]
```

TASK

Select players where their bmi is between 20 to 27 and who were born between (birthyear) 1957 to 2000 print the result.

ANSWER

```
players_df.loc[players_df['bmi'].between(20,27) &
players_df['birthyear'].between(1957,2000)]
```

TASK

Select players where their Height is between 70 to 77 or birth country is USA or Venezuela & print the result.

ANSWER

```
players_df.loc[(players_df['bmi'].between(20,27)) | (players_df['birthcountry'] == 'USA')| (players_df['birthcountry'] == 'Venezuela')]
```

Q&A

THANK YOU