2.3-Pandas Intermediate

KeytoDataScience.com

Table of Contents

1 Indexing and Selecting Data on Columns

Indexing in pandas means simply selecting particular rows and columns of data from a DataFrame. Indexing could mean selecting all the rows and some of the columns, some of the rows and all of the columns, or some of each of the rows and columns. Indexing can also be known as **Subset Selection**.

Since a dataframe is two-dimensional, we can perform **basic operations on rows/columns like:**

- selecting,
- deleting,
- adding,
- renaming

1.1 Column Selection

In Order to select a column in Pandas DataFrame, we can access the columns by calling them by their columns name.

In [1]:
```
# Import pandas package
```

```
import pandas as pd

# Define a dictionary containing employee data
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}
data
# Convert the dictionary into DataFrame
df = pd.DataFrame(data)

# select two columns
print(df[['Name', 'Qualification']])
```

```
     Name Qualification
0     Jai           Msc
1  Princi            MA
2  Gaurav           MCA
3    Anuj           Phd
```

1.2 Column Addition

In Order to add a column in Pandas DataFrame, we can declare a new list as a column and add to a existing Dataframe.

In [2]:

```
# Declare a list that is to be converted into a column
address = ['Delhi', 'Bangalore', 'Chennai', 'Patna']
score = [40,56,23,98]

# Using 'Address' as the column name
# and equating it to the list
df['Address'] = address
df['Score'] = score
# Observe the result
df
```

Out[2]:

	Name	Age	Qualification	Address	Score
0	Jai	27	Msc	Delhi	40
1	Princi	24	MA	Bangalore	56
2	Gaurav	22	MCA	Chennai	23
3	Anuj	32	Phd	Patna	98

1.3 Column Deletion

In Order to delete a column in Pandas DataFrame, we can use the drop() method. Columns is deleted by dropping columns with column names.

In [3]:

```
# dropping passed columns
df.drop(["Address", "Score"], axis = 1, inplace = True)

# display
print(df)
```

```
     Name  Age Qualification
0     Jai   27           Msc
1  Princi   24            MA
2  Gaurav   22           MCA
3    Anuj   32           Phd
```

1.4 Columns Names

While analyzing the real datasets which are often very huge in size, we might need to get the column names in order to perform some certain operations.

Let's discuss how to get column names in Pandas dataframe.

Method #1: Simply iterating over columns

In [4]:
```
# iterating the columns
for col in df.columns:
    print(col)
```

```
Name
Age
Qualification
```

Method #2: Using columns with dataframe object

In [5]:
```
# list(data) or
list(df.columns)
```

Out[5]:
```
['Name', 'Age', 'Qualification']
```

Method #3: Using sorted() method

Sorted() method will return the list of columns sorted in alphabetical order.

In [6]:
```
# using sorted() method
sorted(df)
```

Out[6]:
```
['Age', 'Name', 'Qualification']
```

↥ back to top

2 Indexing and Selecting Data on Rows

2.1 Row Selection:

Pandas provide a unique method to retrieve rows from a Data frame.

DataFrame.loc[] method is used to retrieve rows from Pandas DataFrame.

Rows can also be selected by passing integer location to an iloc[] function.

In [7]:
```
#index by name
```

```
df.set_index("Name",inplace=True)
df
# retrieving row by loc method
first = df.loc["Jai"]
second = df.loc["Anuj"]


print(first, "\n\n\n", second)
```

```
Age               27
Qualification    Msc
Name: Jai, dtype: object


 Age               32
Qualification    Phd
Name: Anuj, dtype: object
```

In [8]:

```
# Using iloc[] function
df.iloc[2,:]
```

Out[8]:

```
Age               22
Qualification    MCA
Name: Gaurav, dtype: object
```

2.2 Row Addition

In Order to add a Row in Pandas DataFrame, we can concat the old dataframe with new one.

In [9]:

```
# Define a dictionary containing employee data
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd'],
        'Address':['Delhi','Gurgaon','Haryana','Bihar'] }

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)
new_row = pd.DataFrame({'Name':'Navin', 'Age':33, 'Address':'Noida',
                        'Qualification':'Btech'},index =[0])
# simply concatenate both dataframes
df = pd.concat([new_row, df],sort=True).reset_index(drop = True)
df
```

Out[9]:

	Address	Age	Name	Qualification
0	Noida	33	Navin	Btech
1	Delhi	27	Jai	Msc
2	Gurgaon	24	Princi	MA
3	Haryana	22	Gaurav	MCA
4	Bihar	32	Anuj	Phd

2.3 Row Deletion

In Order to delete a row in Pandas DataFrame, we can use the drop() method. Rows is deleted by dropping Rows by index label.

In [10]:
```
# dropping passed values
df.drop([0], inplace = True)

# display
df
```

Out[10]:

	Address	Age	Name	Qualification
1	Delhi	27	Jai	Msc
2	Gurgaon	24	Princi	MA
3	Haryana	22	Gaurav	MCA
4	Bihar	32	Anuj	Phd

2.4 Row Names

How to get rows/index names in Pandas dataframe

In [11]:
```
# iterate the indices and print each one
for row in df.index:
    print(row, end= " ")
```

```
1 2 3 4
```

In [12]:
```
print(df)

# OR
print(list(df.index))

# OR
print(list(df.index.values))
```

```
   Address  Age    Name Qualification
1    Delhi   27     Jai           Msc
2  Gurgaon   24  Princi            MA
3  Haryana   22  Gaurav           MCA
4    Bihar   32    Anuj           Phd
[1, 2, 3, 4]
[1, 2, 3, 4]
```

↥ back to top

3 Indexing and Selecting Data on Rows and Columns

Pandas Dataframe type has two attributes called **'columns'** and **'index'** which can be used to change the **column names as well as the row indexes**.

In [13]:
```
# first import the libraries
import pandas as pd
```

```
# Create a dataFrame using dictionary
df=pd.DataFrame({"Name":['Tom','Nick','John','Peter'],
                 "Age":[15,26,17,28]})

# Creates a dataFrame with
# 2 columns and 4 rows
df
```

Out[13]:

	Name	Age
0	Tom	15
1	Nick	26
2	John	17
3	Peter	28

Method #1: Changing the **column name** and **row index** using **df.columns** and **df.index** attribute.

- In order to change the column names, we provide a Python list containing the names for column df.columns= ['First_col', 'Second_col', 'Third_col', ...].
- In order to change the row indexes, we also provide a python list to it df.index=['row1', 'row2', 'row3', ...].

In [14]:

```
# Change the column names
df.columns =['Col_1', 'Col_2']

# Change the row indexes
df.index = ['Row_1', 'Row_2', 'Row_3', 'Row_4']

# printing the data frame
df
```

Out[14]:

	Col_1	Col_2
Row_1	Tom	15
Row_2	Nick	26
Row_3	John	17
Row_4	Peter	28

Method #2: Using rename() function with dictionary to change a column name.

Let's change the first column name from "Col_1" to "Names" using rename() function

In [15]:

```
df = df.rename(columns = {"Col_1":"Names"})

df
```

Out[15]:

	Names	Col_2

	Names	Col_2
Row_1	Tom	15
Row_2	Nick	26
Row_3	John	17
Row_4	Peter	28

In [16]:
```
# We can change multiple column names by
# passing a dictionary of old names and
# new names, to the rename() function.
df = df.rename({"Names":"Student Name","Col_2":"Marks"}, axis='columns')

df
```

Out[16]:

	Student Name	Marks
Row_1	Tom	15
Row_2	Nick	26
Row_3	John	17
Row_4	Peter	28

Method #3: Using Lambda Function to rename the columns.

A lambda function is a small anonymous function which can take any number of arguments, but can only have one expression. Using the lambda function we can modify all of the column names at once. Let's add 'x' at the end of each column name using lambda function

In [17]:
```
df = df.rename(columns=lambda x: x+'_x')

# this will modify all the column names
df
```

Out[17]:

	Student Name_x	Marks_x
Row_1	Tom	15
Row_2	Nick	26
Row_3	John	17
Row_4	Peter	28

Method #4: Using list to rename the columns.

We can pass the list with updated columns names we want to change.

In [18]:
```
# this will modify the name of columns
df.columns = ['Name','Student_Marks']

df
```

Out[18]:

	Name	Student_Marks
Row_1	Tom	15
Row_2	Nick	26
Row_3	John	17
Row_4	Peter	28

Method #5: Using lambda function change the row index.

In [19]:

```
# Let's change the row index using the Lambda function
# To change the row indexes
df = pd.DataFrame({"A":['Tom','Nick','John','Peter'],
                   "B":[25,16,27,18]})
df
```

Out[19]:

	A	B
0	Tom	25
1	Nick	16
2	John	27
3	Peter	18

In [20]:

```
# this will increase the row index value by 10 for each row
df = df.rename(index = lambda x: x + 10)

df
```

Out[20]:

	A	B
10	Tom	25
11	Nick	16
12	John	27
13	Peter	18

Method #6: Change both Column and Row indexes

In [21]:

```
# increase all the row index label by value 5
# append a value 'x' at the end of each column name.

df = df.rename(index = lambda x: x + 5,
               columns = lambda x: x +'x')

df
```

Out[21]:

	Ax	Bx
15	Tom	25

	Ax	Bx
16	Nick	16
17	John	27
18	Peter	18

Method #7: Get **unique values** from a column in Pandas DataFrame

In [22]:
```
# create a dictionary with five fields each
data = {
    'A':['A1', 'A2', 'A3', 'A4', 'A5'],
    'B':['B1', 'B2', 'B3', 'B4', 'B4'],
    'C':['C1', 'C2', 'C3', 'C3', 'C3'],
    'D':['D1', 'D2', 'D2', 'D2', 'D2'],
    'E':['E1', 'E1', 'E1', 'E1', 'E1'] }

# Convert the dictionary into DataFrame
dframe = pd.DataFrame(data)

dframe
```

Out[22]:

	A	B	C	D	E
0	A1	B1	C1	D1	E1
1	A2	B2	C2	D2	E1
2	A3	B3	C3	D2	E1
3	A4	B4	C3	D2	E1
4	A5	B4	C3	D2	E1

In [23]:
```
# Get the unique values of 'B' column
dframe.B.unique()
```

Out[23]:
```
array(['B1', 'B2', 'B3', 'B4'], dtype=object)
```

In [24]:
```
# Get the unique values of 'E' column
dframe.E.unique()
```

Out[24]:
```
array(['E1'], dtype=object)
```

In [25]:
```
# Get the count unique values of 'B' column

dframe.B.nunique()
```

Out[25]:
```
4
```

In [26]:
```
# Get the count unique values of 'E' column

dframe.E.nunique()
```

Out[26]:
```
1
```

IMPORTANT :

Selecting Rows and Columns Based on Conditions in Python Pandas DataFrame - KeytoDataScience

↥ back to top

4 Iterating over rows and columns in Pandas DataFrame

In Pandas Dataframe we can **iterate an element in two ways:**

- Iterating over **rows**
- Iterating over **columns**

In [27]:
```
# dictionary of lists
dict = {'name':["aparna", "pankaj", "sudhir", "Geeku"],
        'degree': ["MBA", "BCA", "M.Tech", "MBA"],
        'score':[90, 40, 80, 98]}

# creating a dataframe from a dictionary
df = pd.DataFrame(dict)

df
```

Out[27]:

	name	degree	score
0	aparna	MBA	90
1	pankaj	BCA	40
2	sudhir	M.Tech	80
3	Geeku	MBA	98

Method #1: Iterate over rows using iterrows()

In [28]:
```
# iterating over rows using iterrows() function
for i, j in df.iterrows():
    print(i,"\n------\n",j)
    print("_____________________________")
```

```
0
------
 name      aparna
degree       MBA
score         90
Name: 0, dtype: object
_____________________________
1
------
 name      pankaj
degree       BCA
score         40
```

```
Name: 1, dtype: object
_____________________________
2
------
 name      sudhir
degree    M.Tech
score         80
Name: 2, dtype: object
_____________________________
3
------
 name      Geeku
degree      MBA
score        98
Name: 3, dtype: object
_____________________________
```

Method #2: Iterate over rows using itertuples()

In [29]:
```
# using a itertuples()
for i in df.itertuples():
    print(i)
```

```
Pandas(Index=0, name='aparna', degree='MBA', score=90)
Pandas(Index=1, name='pankaj', degree='BCA', score=40)
Pandas(Index=2, name='sudhir', degree='M.Tech', score=80)
Pandas(Index=3, name='Geeku', degree='MBA', score=98)
```

Method #3: Iterate over columns using list

Now we iterate through columns in order to iterate through columns we first create a list of dataframe columns and then iterate through list.

In [30]:
```
# creating a list of dataframe columns
columns = list(df)

for i in columns:

    # printing the third element of the column
    print (df[i][2])
```

```
sudhir
M.Tech
80
```

↥ back to top

5 Sorting using Pandas

In [31]:
```
# making data frame from csv file
data = pd.read_csv("nba.csv")

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

```
# display
data.head()
```

Out[31]:

	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

Example #1: Sorting by Name and Team

In the following example, A data frame is made from the csv file and the data frame is sorted in ascending order of Team and in every Team the Name is also sorted in Ascending order.

In [32]:

```
#making data frame from csv file
data=pd.read_csv("nba.csv")

#sorting data frame by Team and then By names
data.sort_values(by=["Team", "Name"], axis=0,
                 ascending=True, inplace=True)

#display
data
```

Out[32]:

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
312	Al Horford	Atlanta Hawks	15.0	C	30.0	6-10	245.0	Florida	12000000.0
318	Dennis Schroder	Atlanta Hawks	17.0	PG	22.0	6-1	172.0	NaN	1763400.0
323	Jeff Teague	Atlanta Hawks	0.0	PG	27.0	6-2	186.0	Wake Forest	8000000.0
309	Kent Bazemore	Atlanta Hawks	24.0	SF	26.0	6-5	201.0	Old Dominion	2000000.0
311	Kirk Hinrich	Atlanta Hawks	12.0	SG	35.0	6-4	190.0	Kansas	2854940.0
...	...	...	...	...	...	...	...	...	...
376	Markieff Morris	Washington Wizards	5.0	PF	26.0	6-10	245.0	Kansas	8000000.0
375	Nene Hilario	Washington Wizards	42.0	C	33.0	6-11	250.0	NaN	13000000.0
378	Otto Porter Jr.	Washington Wizards	22.0	SF	23.0	6-8	198.0	Georgetown	4662960.0
379	Ramon Sessions	Washington Wizards	7.0	PG	30.0	6-3	190.0	Nevada	2170465.0

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
457	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

458 rows × 9 columns

Example #2: Passing list to Ascending Parameter

As shown in the above example, a Data frame can be sorted with respect to multiple columns by passing a list to the 'by' Parameter.

We can also pass a list to the 'ascending' Parameter to tell pandas which column to sort how (ascending/descing). The index of Boolean in 'ascending' parameter should be same as the index of column name in 'by' Parameter.

In [33]:
```
#making data frame from csv file
data=pd.read_csv("nba.csv")

#sorting data frame by Team and then By names
data.sort_values(["Team", "Name"], axis=0,
                 ascending=[True,False], inplace=True)

#display
data
```

Out[33]:

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
322	Walter Tavares	Atlanta Hawks	22.0	C	24.0	7-3	260.0	NaN	1000000.0
310	Tim Hardaway Jr.	Atlanta Hawks	10.0	SG	24.0	6-6	205.0	Michigan	1304520.0
321	Tiago Splitter	Atlanta Hawks	11.0	C	31.0	6-11	245.0	NaN	9756250.0
320	Thabo Sefolosha	Atlanta Hawks	25.0	SF	32.0	6-7	220.0	NaN	4000000.0
315	Paul Millsap	Atlanta Hawks	4.0	PF	31.0	6-8	246.0	Louisiana Tech	18671659.0
...	...	...	...	...	...	...	...	...	...
380	Garrett Temple	Washington Wizards	17.0	SG	30.0	6-6	195.0	LSU	1100602.0
372	Drew Gooden	Washington Wizards	90.0	PF	34.0	6-10	250.0	Kansas	3300000.0
369	Bradley Beal	Washington Wizards	3.0	SG	22.0	6-5	207.0	Florida	5694674.0
368	Alan Anderson	Washington Wizards	6.0	SG	33.0	6-6	220.0	Michigan State	4000000.0
457	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

458 rows × 9 columns

Example #3: Changing position of Null values

In the give 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 Null values are kept at the top.

In [34]:
```
# making data frame from csv file
data = pd.read_csv("nba.csv")

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

data.head(15)
# display
```

Out[34]:

	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
270	Xavier Munford	Memphis Grizzlies	14.0	PG	24.0	6-3	180.0	Rhode Island	NaN
273	Alex Stepheson	Memphis Grizzlies	35.0	PF	28.0	6-10	270.0	USC	NaN
350	Briante Weber	Miami Heat	12.0	PG	23.0	6-2	165.0	Virginia Commonwealth	NaN
353	Dorell Wright	Miami Heat	11.0	SF	30.0	6-9	205.0	NaN	NaN
397	Axel Toupane	Denver Nuggets	6.0	SG	23.0	6-7	210.0	NaN	NaN
409	Greg Smith	Minnesota Timberwolves	4.0	PF	25.0	6-10	250.0	Fresno State	NaN
457	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
32	Thanasis Antetokounmpo	New York Knicks	43.0	SF	23.0	6-7	205.0	NaN	30888.0
291	Orlando Johnson	New Orleans Pelicans	0.0	SG	27.0	6-5	220.0	UC Santa Barbara	55722.0
130	Phil Pressey	Phoenix Suns	25.0	PG	25.0	5-11	175.0	Missouri	55722.0

↥ back to top

6 Revision - Basic operations on a Data Frame

In [35]:
```
# importing pandas module
import pandas as pd

# making data frame
df = pd.read_csv("https://media.geeksforgeeks.org/wp-content/uploads/nba.csv")

df.head(5)
```

Out[35]:

	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

Method #1: Fetch **n** smallest or largest values of a column

In [36]:
```
# five smallest values in column Salary
df.nsmallest(5, ['Salary'])
```

Out[36]:

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
32	Thanasis Antetokounmpo	New York Knicks	43.0	SF	23.0	6-7	205.0	NaN	30888.0
130	Phil Pressey	Phoenix Suns	25.0	PG	25.0	5-11	175.0	Missouri	55722.0
291	Orlando Johnson	New Orleans Pelicans	0.0	SG	27.0	6-5	220.0	UC Santa Barbara	55722.0
135	Alan Williams	Phoenix Suns	15.0	C	23.0	6-8	260.0	UC Santa Barbara	83397.0
175	Jordan McRae	Cleveland Cavaliers	12.0	SG	25.0	6-5	179.0	Tennessee	111196.0

In [37]:
```
# five largest values in column age
df.nlargest(5, ['Age'])
```

Out[37]:

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
298	Tim Duncan	San Antonio Spurs	21.0	C	40.0	6-11	250.0	Wake Forest	5250000.0

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
304	Andre Miller	San Antonio Spurs	24.0	PG	40.0	6-3	200.0	Utah	250750.0
400	Kevin Garnett	Minnesota Timberwolves	21.0	PF	40.0	6-11	240.0	NaN	8500000.0
102	Pablo Prigioni	Los Angeles Clippers	9.0	PG	39.0	6-3	185.0	NaN	947726.0
261	Vince Carter	Memphis Grizzlies	15.0	SG	39.0	6-6	220.0	North Carolina	4088019.0

In [38]:
```
# Ten largest values in column Weight
df.nlargest(10, ['Weight'])
```

Out[38]:

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
405	Nikola Pekovic	Minnesota Timberwolves	14.0	C	30.0	6-11	307.0	NaN	12100000.0
302	Boban Marjanovic	San Antonio Spurs	40.0	C	27.0	7-3	290.0	NaN	1200000.0
330	Al Jefferson	Charlotte Hornets	25.0	C	31.0	6-10	289.0	NaN	13500000.0
395	Jusuf Nurkic	Denver Nuggets	23.0	C	21.0	7-0	280.0	NaN	1842000.0
188	Andre Drummond	Detroit Pistons	0.0	C	22.0	6-11	279.0	Connecticut	3272091.0
41	Kevin Seraphin	New York Knicks	1.0	C	26.0	6-10	278.0	NaN	2814000.0
23	Brook Lopez	Brooklyn Nets	11.0	C	28.0	7-0	275.0	Stanford	19689000.0
56	Jahlil Okafor	Philadelphia 76ers	8.0	C	20.0	6-11	275.0	Duke	4582680.0
155	Cristiano Felicio	Chicago Bulls	6.0	PF	23.0	6-10	275.0	NaN	525093.0
176	Timofey Mozgov	Cleveland Cavaliers	20.0	C	29.0	7-1	275.0	NaN	4950000.0

Method #2: Apply uppercase to a column in Pandas dataframe

In [39]:
```
df['Name'] = df['Name'].str.upper()

df.head()
```

Out[39]:

	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

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
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

In [40]:
```
# removing null values to avoid errors
df.dropna(inplace = True)

# Applying upper() method on 'College' column
df['College'].apply(lambda x: x.upper()).head()
```

Out[40]:
```
0            TEXAS
1        MARQUETTE
3    GEORGIA STATE
6              LSU
7          GONZAGA
Name: College, dtype: object
```

Method #3: Selecting some rows and some columns:

In order to select two rows and three columns, we select a two rows which we want to select and three columns and put it in a separate list like this:

In [41]:
```
# making data frame from csv file. This is another way of reading a CSV file
data = pd.read_csv("nba.csv", index_col ="Name")

# retrieving two rows and three columns by loc method
first = data.loc[["Avery Bradley", "R.J. Hunter"],
                 ["Team", "Number", "Position"]]

print(first)
```

```
                      Team  Number Position
Name
Avery Bradley  Boston Celtics     0.0       PG
R.J. Hunter    Boston Celtics    28.0       SG
```

Method #4: Selecting rows and columns using **.loc**:

In order to select all of the rows and some columns, we use single colon [:] to select all of rows and list of some columns which we want to select like this:

In [42]:
```
# retrieving all rows and some columns by loc method
first = data.loc[:, ["Team", "Number", "Position"]]
first
```

Out[42]:

	Team	Number	Position

Name	Team	Number	Position
Name			
Avery Bradley	Boston Celtics	0.0	PG
Jae Crowder	Boston Celtics	99.0	SF
John Holland	Boston Celtics	30.0	SG
R.J. Hunter	Boston Celtics	28.0	SG
Jonas Jerebko	Boston Celtics	8.0	PF
...	...	...	...
Shelvin Mack	Utah Jazz	8.0	PG
Raul Neto	Utah Jazz	25.0	PG
Tibor Pleiss	Utah Jazz	21.0	C
Jeff Withey	Utah Jazz	24.0	C
NaN	NaN	NaN	NaN

458 rows × 3 columns

Method #5: Selecting rows and columns using **.iloc**:

In [43]:
```
# retrieving rows by iloc method
row2 = data.iloc[3]
row2
```

Out[43]:
```
Team         Boston Celtics
Number                 28.0
Position                 SG
Age                    22.0
Height                  6-5
Weight                185.0
College       Georgia State
Salary           1148640.0
Name: R.J. Hunter, dtype: object
```

In [44]:
```
# retrieving two rows and two columns by iloc method
row2 = data.iloc [[3, 4], [1, 2]]

print(row2)
```

```
               Number Position
Name
R.J. Hunter      28.0       SG
Jonas Jerebko     8.0       PF
```

In [45]:
```
# retrieving all rows and some columns by iloc method
row2 = data.iloc [:, [1, 2]]
row2
```

Out[45]:

Name	Number	Position
Avery Bradley	0.0	PG
Jae Crowder	99.0	SF
John Holland	30.0	SG
R.J. Hunter	28.0	SG
Jonas Jerebko	8.0	PF
...	...	...
Shelvin Mack	8.0	PG
Raul Neto	25.0	PG
Tibor Pleiss	21.0	C
Jeff Withey	24.0	C
NaN	NaN	NaN

458 rows × 2 columns

↥ back to top

(Optional): This Pandas exercise will help the learners to get a better understanding of data analysis problems. This practice page consists of a huge set of Pandas programs like Pandas Dataframe/series, handling Rows/Columns, grouping and all sort of frequently encounterd problems.

Pandas Practice Excercises Questions & Solutions

Great Job!
