MSAS Tutorial Sequence -Module Four-

WRITTEN BY GARRETT FOLBE

BACK AT IT

Goals of Today

1. Understand what Git is
   1. Implement a system to keep your work uniform
2. Catch up from last time
   1. Pandas overview
   2. Reading from excel
3. Learn to manipulate DataFrames
   1. Select specific rows
   2. Select specific columns
4. Get some feedback!

Git is useful, but hard ☹

- What is Git?
  - Git is a software used for version control
  - Used in some shape or form in every tech company
- What is GitHub?
  - A company that provides hosting for software development
  - Users can create code repositories that can be used by others
  - Allows for collaboration and version control
  - Think about it like Google Drive without the real-time collaboration
- What is GitBash?
  - Windows users only (although not exclusive)
  - A terminal that allows you to run linux commands (useful for git, especially)

Why do we use GitHub?

- I am very familiar with it
- Allows me to make my work accessible to the public
- File storage is very easy
- Good skill for everyone to have!

Let’s make our lives easier…

1. Delete all files you have already cloned and/or created related to this tutorial sequence
2. Open up your terminal
3. Run the command: cd ~/Desktop
4. Run the command: git clone https://github.com/gfolbe318/MSAS_tutorials.git
5. NEVER modify the folder you just cloned
6. For each week from now on, COPY the current module (entire folder) we are working on
7. Paste it somewhere you’d like to store your local files. This new copy can be modified

Review from last time

Pandas

Welcome to REAL Data Science

Libraries in Python

- A python library is a collection of functions that allows you to perform specific actions with your code
- More often than not, a library is designed for a specific feature. There are libraries to…
  - Move and modify files on your computer
  - Create graphical interfaces for your code
  - Webscrape (more on that in the next module)
  - Perform data science operations ☺
  - And much more!
- Libraries can be imported with the following syntax:
  - import library (imports entire library)
  - import library as lib (imports entire library as an alias, usually used to not clutter up code)
  - from library, import function a, b (imports functions a and b from the library, saves memory)
- To access functions of your imported library, you will need to use the dot operator
  - We will see this throughout the module sequence

What is Pandas?

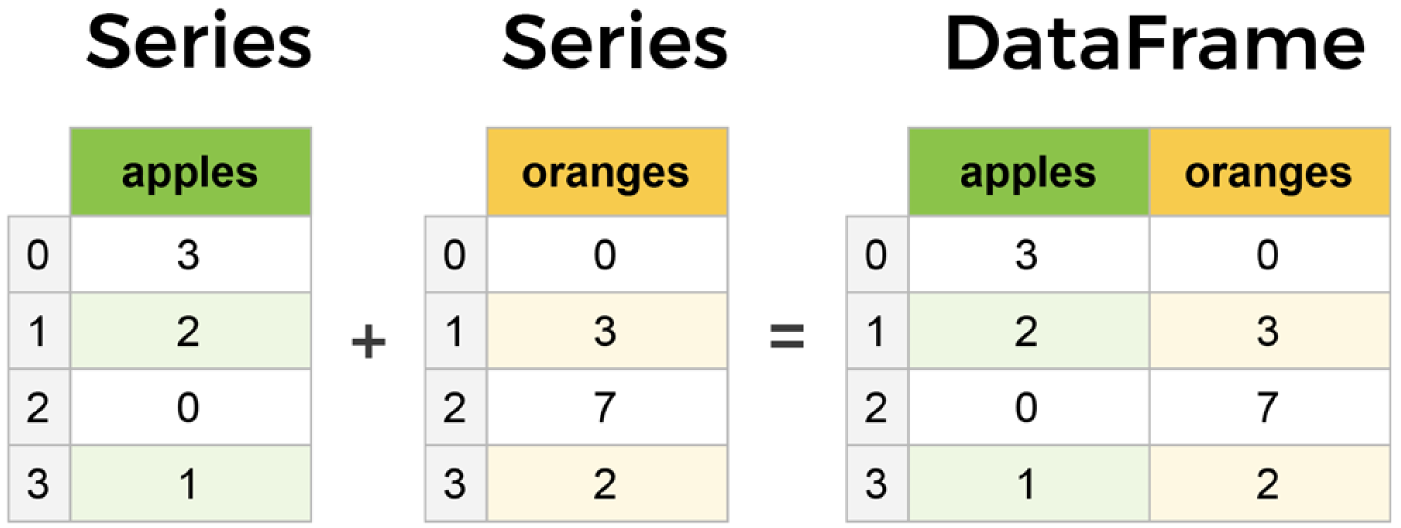
- From their documentation:

“pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.”

- Very powerful data science library
- The basis of the rest of this module, as well as the next three!
- Import pandas at the top of your jupyter notebok
  - import pandas as pd

Series vs DataFrames

- The primary two components of pandas are Series and DataFrames
- Series is a column in your table, and a DataFrame is a multidimensional table comprised of a collection of Series



Series

	apples
0	3
1	2
2	0
3	1

+

Series

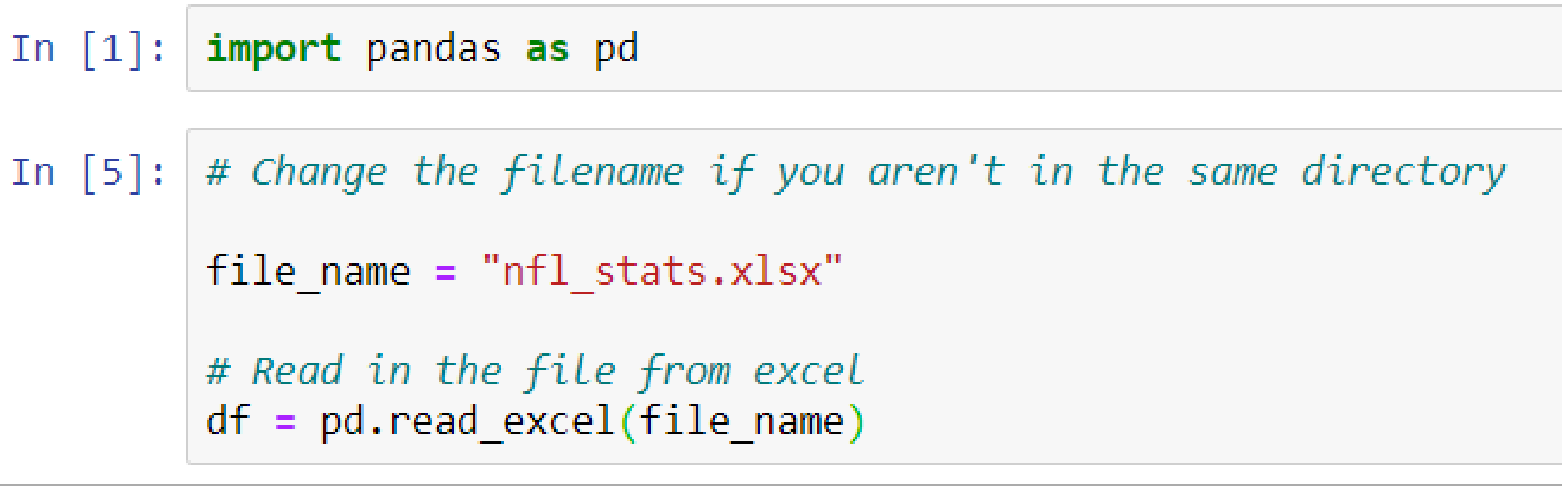
	oranges
0	0
1	3
2	7
3	2

=

DataFrame

	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2

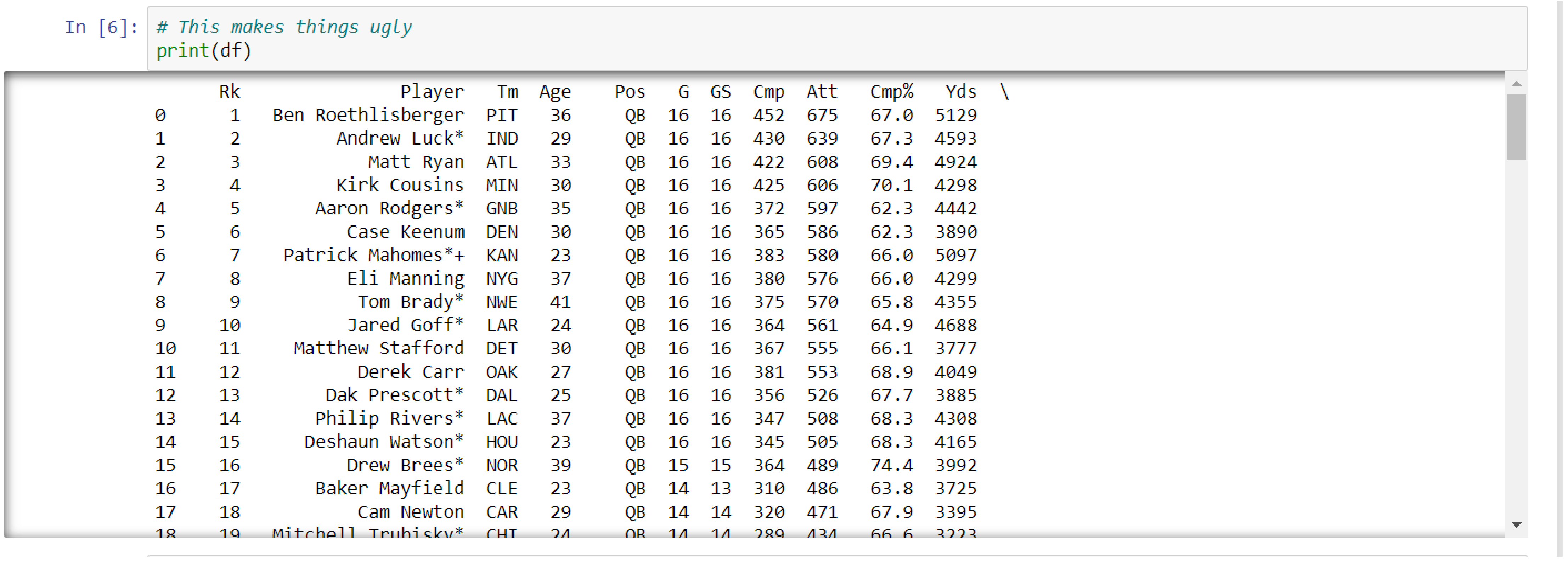
Reading in Files



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

```
In [5]: # Change the filename if you aren't in the same directory

        file_name = "nfl_stats.xlsx"

        # Read in the file from excel
        df = pd.read_excel(file_name)
```

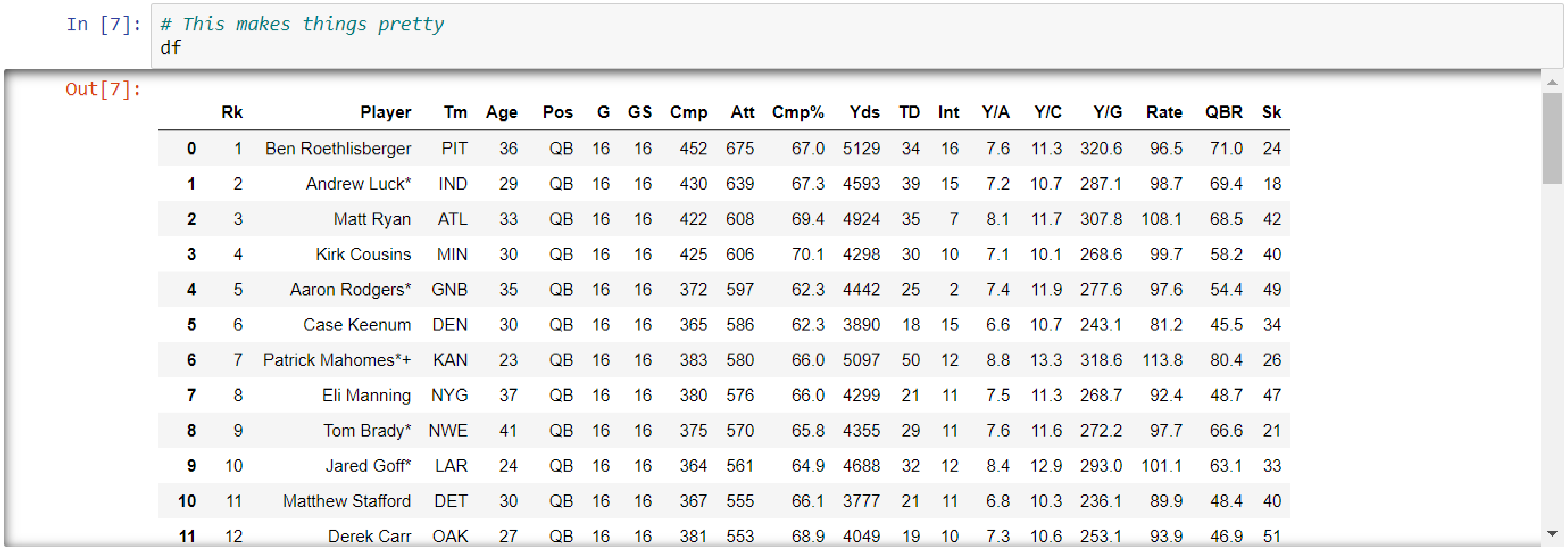
Printing a DataFrame



```
In [6]: # This makes things ugly
        print(df)
```

```
        Rk              Player   Tm  Age   Pos   G  GS  Cmp  Att  Cmp%   Yds  \
0        1   Ben Roethlisberger  PIT   36    QB  16  16  452  675  67.0  5129
1        2        Andrew Luck*   IND   29    QB  16  16  430  639  67.3  4593
2        3           Matt Ryan   ATL   33    QB  16  16  422  608  69.4  4924
3        4        Kirk Cousins   MIN   30    QB  16  16  425  606  70.1  4298
4        5      Aaron Rodgers*   GNB   35    QB  16  16  372  597  62.3  4442
5        6         Case Keenum   DEN   30    QB  16  16  365  586  62.3  3890
6        7    Patrick Mahomes*+  KAN   23    QB  16  16  383  580  66.0  5097
7        8         Eli Manning   NYG   37    QB  16  16  380  576  66.0  4299
8        9          Tom Brady*   NWE   41    QB  16  16  375  570  65.8  4355
9       10         Jared Goff*   LAR   24    QB  16  16  364  561  64.9  4688
10      11     Matthew Stafford  DET   30    QB  16  16  367  555  66.1  3777
11      12          Derek Carr   OAK   27    QB  16  16  381  553  68.9  4049
12      13        Dak Prescott*  DAL   25    QB  16  16  356  526  67.7  3885
13      14       Philip Rivers*  LAC   37    QB  16  16  347  508  68.3  4308
14      15      Deshaun Watson*  HOU   23    QB  16  16  345  505  68.3  4165
15      16         Drew Brees*   NOR   39    QB  15  15  364  489  74.4  3992
16      17       Baker Mayfield  CLE   23    QB  14  13  310  486  63.8  3725
17      18          Cam Newton   CAR   29    QB  14  14  320  471  67.9  3395
18      19   Mitchell Trubisky*  CHI   24    QB  14  14  289  434  66.6  3223
```

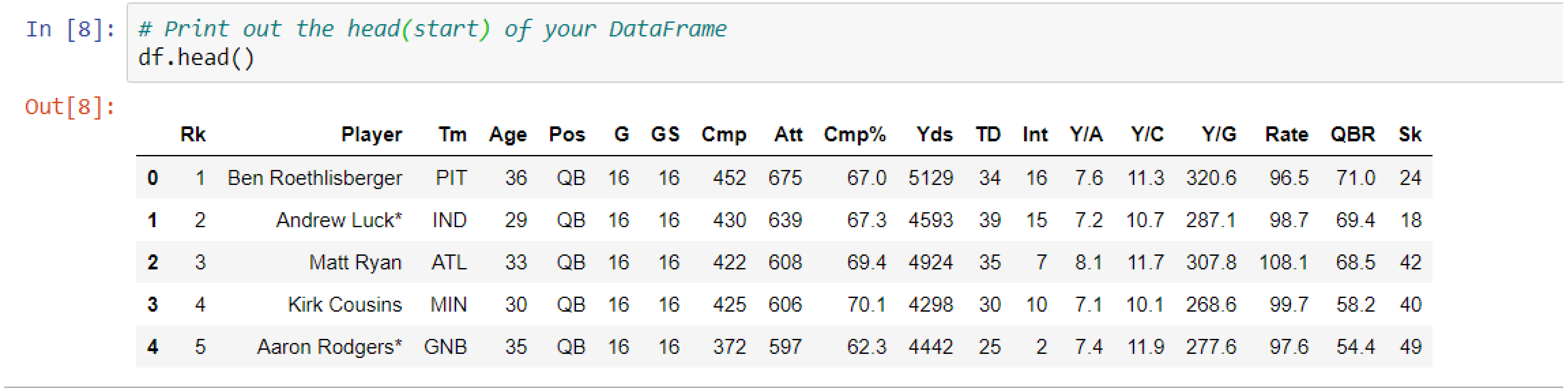
Displaying a DataFrame



```
In [7]: # This makes things pretty
        df
```

```
Out[7]:
      Rk              Player   Tm  Age  Pos   G  GS  Cmp  Att  Cmp%   Yds  TD  Int  Y/A   Y/C    Y/G   Rate   QBR  Sk
  0    1   Ben Roethlisberger  PIT   36   QB  16  16  452  675  67.0  5129  34   16  7.6  11.3  320.6   96.5  71.0  24
  1    2        Andrew Luck*   IND   29   QB  16  16  430  639  67.3  4593  39   15  7.2  10.7  287.1   98.7  69.4  18
  2    3          Matt Ryan    ATL   33   QB  16  16  422  608  69.4  4924  35    7  8.1  11.7  307.8  108.1  68.5  42
  3    4        Kirk Cousins   MIN   30   QB  16  16  425  606  70.1  4298  30   10  7.1  10.1  268.6   99.7  58.2  40
  4    5      Aaron Rodgers*   GNB   35   QB  16  16  372  597  62.3  4442  25    2  7.4  11.9  277.6   97.6  54.4  49
  5    6         Case Keenum   DEN   30   QB  16  16  365  586  62.3  3890  18   15  6.6  10.7  243.1   81.2  45.5  34
  6    7   Patrick Mahomes*+   KAN   23   QB  16  16  383  580  66.0  5097  50   12  8.8  13.3  318.6  113.8  80.4  26
  7    8         Eli Manning   NYG   37   QB  16  16  380  576  66.0  4299  21   11  7.5  11.3  268.7   92.4  48.7  47
  8    9         Tom Brady*    NWE   41   QB  16  16  375  570  65.8  4355  29   11  7.6  11.6  272.2   97.7  66.6  21
  9   10         Jared Goff*   LAR   24   QB  16  16  364  561  64.9  4688  32   12  8.4  12.9  293.0  101.1  63.1  33
 10   11     Matthew Stafford  DET   30   QB  16  16  367  555  66.1  3777  21   11  6.8  10.3  236.1   89.9  48.4  40
 11   12          Derek Carr   OAK   27   QB  16  16  381  553  68.9  4049  19   10  7.3  10.6  253.1   93.9  46.9  51
```

Displaying the head of a DataFrame



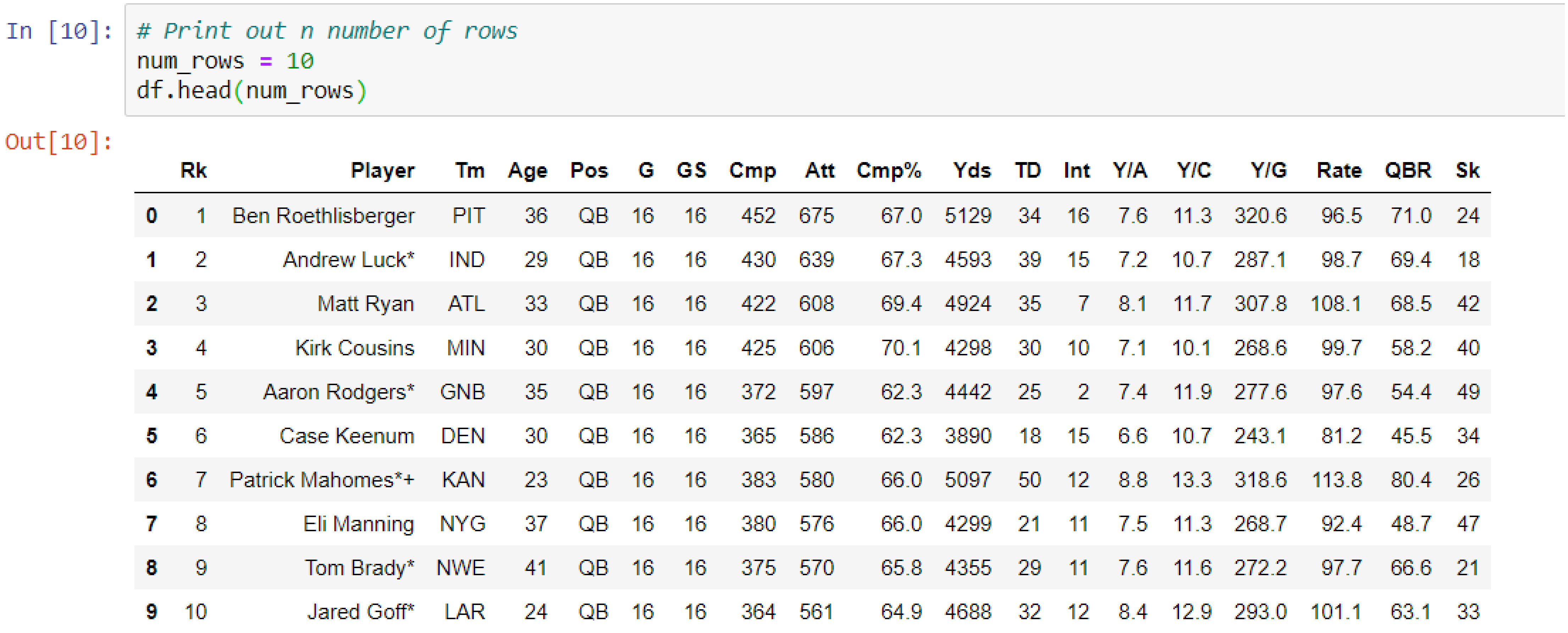
In [8]:

```
# Print out the head(start) of your DataFrame
df.head()
```

Out[8]:

	Rk	Player	Tm	Age	Pos	G	GS	Cmp	Att	Cmp%	Yds	TD	Int	Y/A	Y/C	Y/G	Rate	QBR	Sk
0	1	Ben Roethlisberger	PIT	36	QB	16	16	452	675	67.0	5129	34	16	7.6	11.3	320.6	96.5	71.0	24
1	2	Andrew Luck*	IND	29	QB	16	16	430	639	67.3	4593	39	15	7.2	10.7	287.1	98.7	69.4	18
2	3	Matt Ryan	ATL	33	QB	16	16	422	608	69.4	4924	35	7	8.1	11.7	307.8	108.1	68.5	42
3	4	Kirk Cousins	MIN	30	QB	16	16	425	606	70.1	4298	30	10	7.1	10.1	268.6	99.7	58.2	40
4	5	Aaron Rodgers*	GNB	35	QB	16	16	372	597	62.3	4442	25	2	7.4	11.9	277.6	97.6	54.4	49

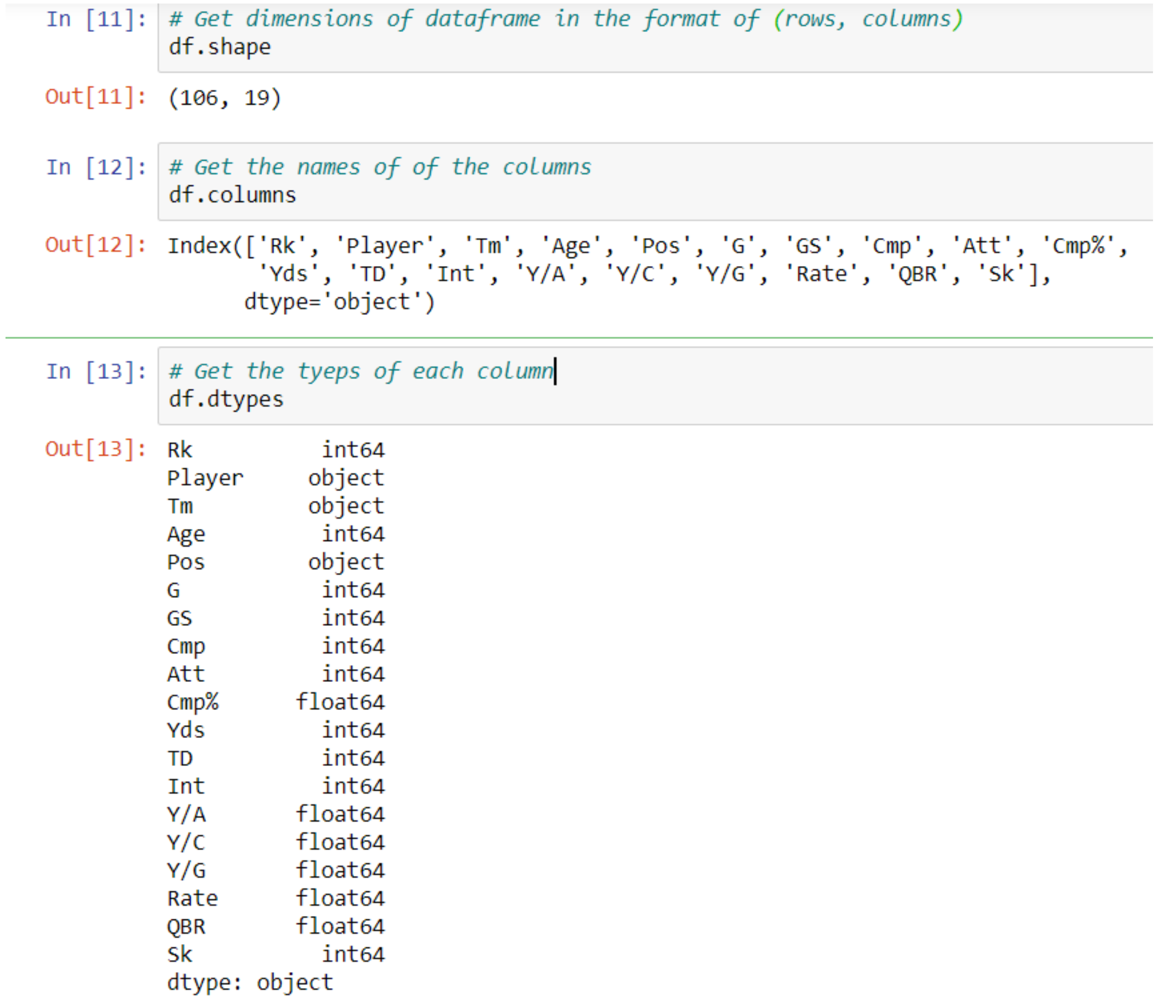
Display a Specific Number of Rows



In [10]:
```python
# Print out n number of rows
num_rows = 10
df.head(num_rows)
```

Out[10]:

	Rk	Player	Tm	Age	Pos	G	GS	Cmp	Att	Cmp%	Yds	TD	Int	Y/A	Y/C	Y/G	Rate	QBR	Sk
0	1	Ben Roethlisberger	PIT	36	QB	16	16	452	675	67.0	5129	34	16	7.6	11.3	320.6	96.5	71.0	24
1	2	Andrew Luck*	IND	29	QB	16	16	430	639	67.3	4593	39	15	7.2	10.7	287.1	98.7	69.4	18
2	3	Matt Ryan	ATL	33	QB	16	16	422	608	69.4	4924	35	7	8.1	11.7	307.8	108.1	68.5	42
3	4	Kirk Cousins	MIN	30	QB	16	16	425	606	70.1	4298	30	10	7.1	10.1	268.6	99.7	58.2	40
4	5	Aaron Rodgers*	GNB	35	QB	16	16	372	597	62.3	4442	25	2	7.4	11.9	277.6	97.6	54.4	49
5	6	Case Keenum	DEN	30	QB	16	16	365	586	62.3	3890	18	15	6.6	10.7	243.1	81.2	45.5	34
6	7	Patrick Mahomes*+	KAN	23	QB	16	16	383	580	66.0	5097	50	12	8.8	13.3	318.6	113.8	80.4	26
7	8	Eli Manning	NYG	37	QB	16	16	380	576	66.0	4299	21	11	7.5	11.3	268.7	92.4	48.7	47
8	9	Tom Brady*	NWE	41	QB	16	16	375	570	65.8	4355	29	11	7.6	11.6	272.2	97.7	66.6	21
9	10	Jared Goff*	LAR	24	QB	16	16	364	561	64.9	4688	32	12	8.4	12.9	293.0	101.1	63.1	33



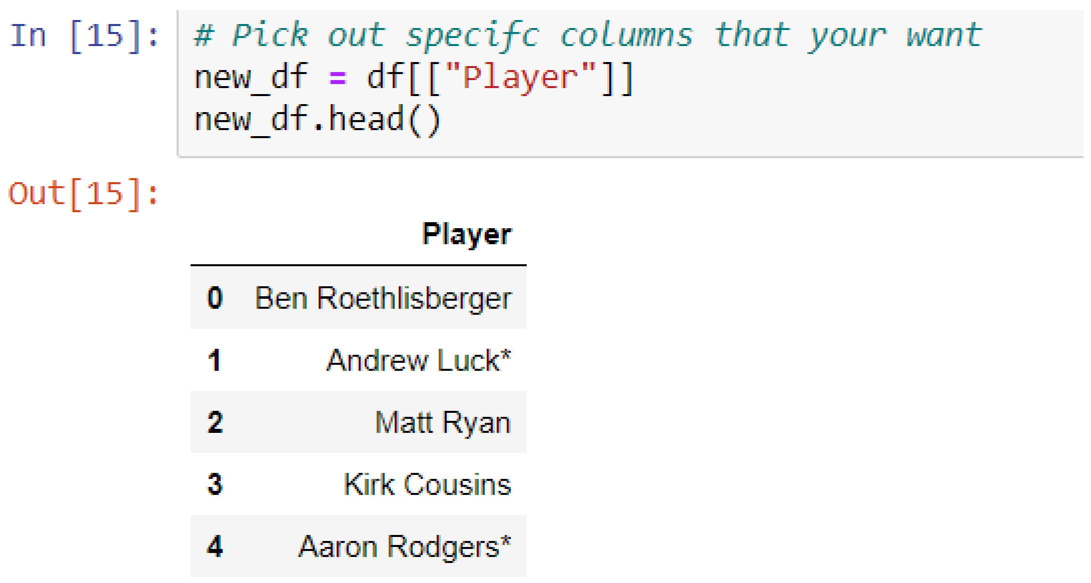
```
In [11]: # Get dimensions of dataframe in the format of (rows, columns)
         df.shape

Out[11]: (106, 19)

In [12]: # Get the names of of the columns
         df.columns

Out[12]: Index(['Rk', 'Player', 'Tm', 'Age', 'Pos', 'G', 'GS', 'Cmp', 'Att', 'Cmp%',
                'Yds', 'TD', 'Int', 'Y/A', 'Y/C', 'Y/G', 'Rate', 'QBR', 'Sk'],
               dtype='object')

In [13]: # Get the tyeps of each column
         df.dtypes

Out[13]: Rk          int64
         Player     object
         Tm         object
         Age         int64
         Pos        object
         G           int64
         GS          int64
         Cmp         int64
         Att         int64
         Cmp%      float64
         Yds         int64
         TD          int64
         Int         int64
         Y/A       float64
         Y/C       float64
         Y/G       float64
         Rate      float64
         QBR       float64
         Sk          int64
         dtype: object
```

Get Information about the DataFrame

Let’s dive a little deeper

- Extracting columns
- Extracting rows
- Modifying the table
  - Sorting values, deleting columns, renaming the index
- Combining DataFrames

Extract Certain Columns

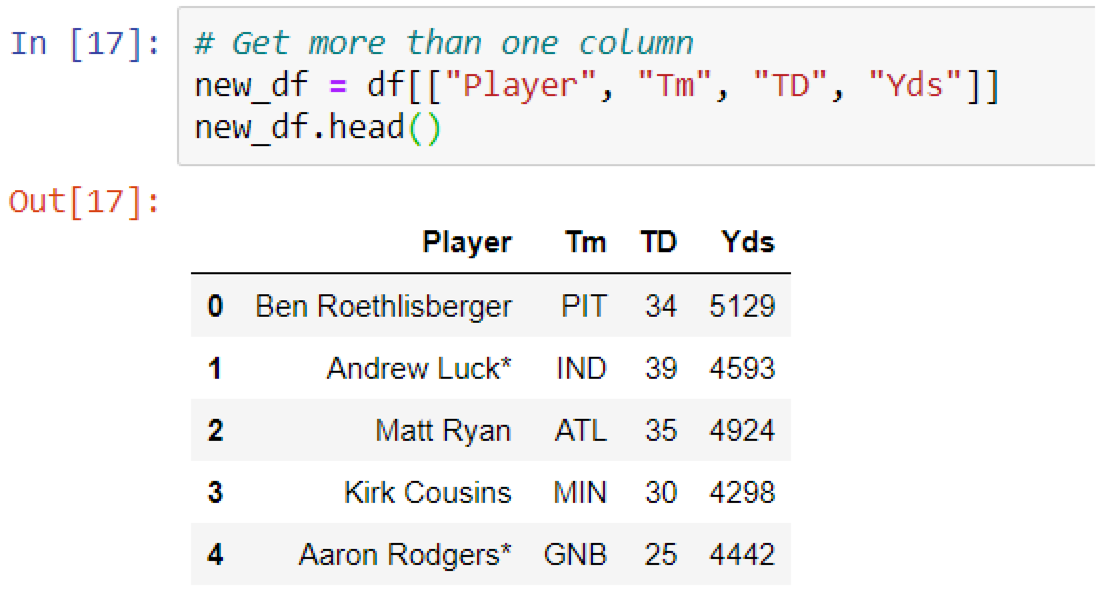


In [15]:

```
# Pick out specifc columns that your want
new_df = df[["Player"]]
new_df.head()
```

Out[15]:

	Player
0	Ben Roethlisberger
1	Andrew Luck*
2	Matt Ryan
3	Kirk Cousins
4	Aaron Rodgers*



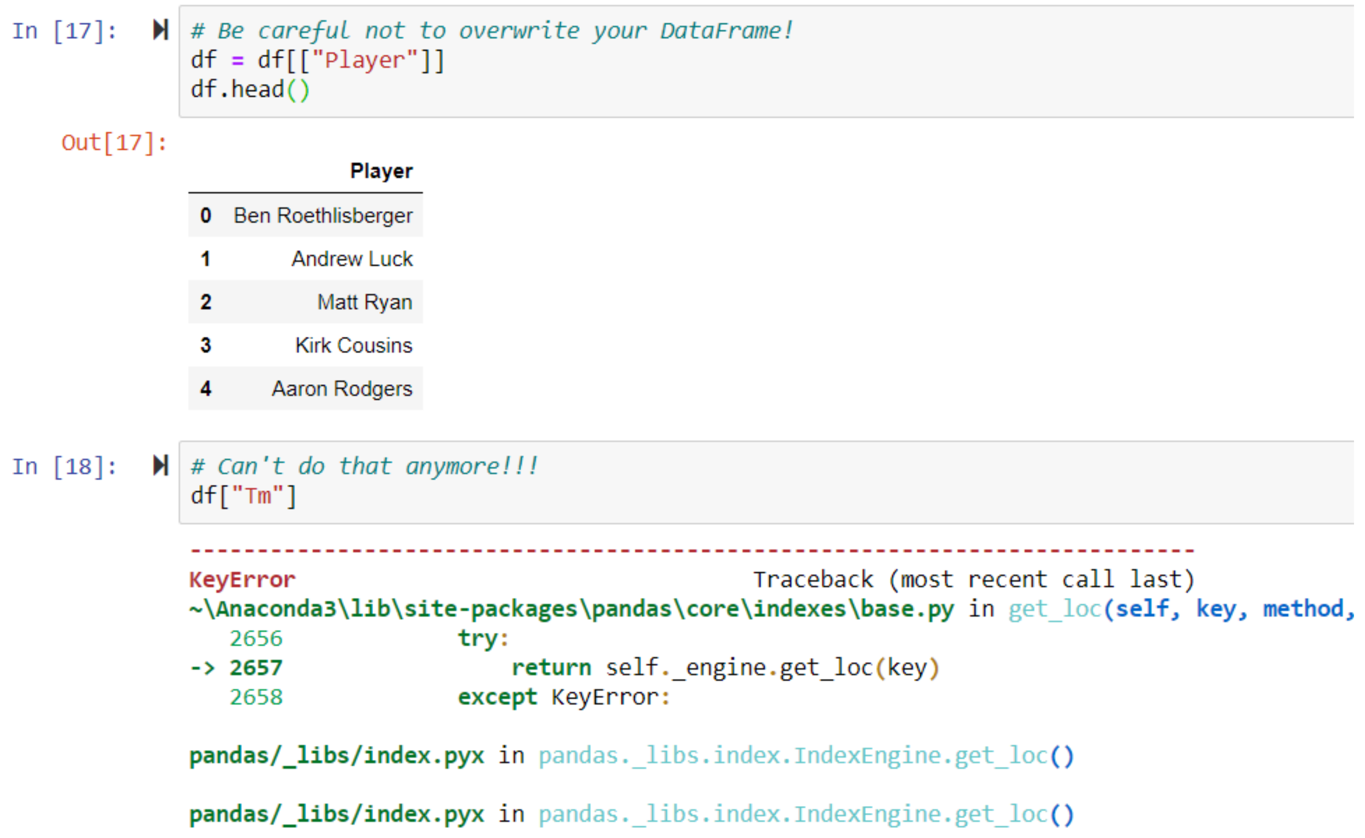
In [17]:

```
# Get more than one column
new_df = df[["Player", "Tm", "TD", "Yds"]]
new_df.head()
```

Out[17]:

	Player	Tm	TD	Yds
0	Ben Roethlisberger	PIT	34	5129
1	Andrew Luck*	IND	39	4593
2	Matt Ryan	ATL	35	4924
3	Kirk Cousins	MIN	30	4298
4	Aaron Rodgers*	GNB	25	4442

Be careful to not overwrite your DataFrame!



```
In [17]:  # Be careful not to overwrite your DataFrame!
          df = df[["Player"]]
          df.head()
```

Out[17]:

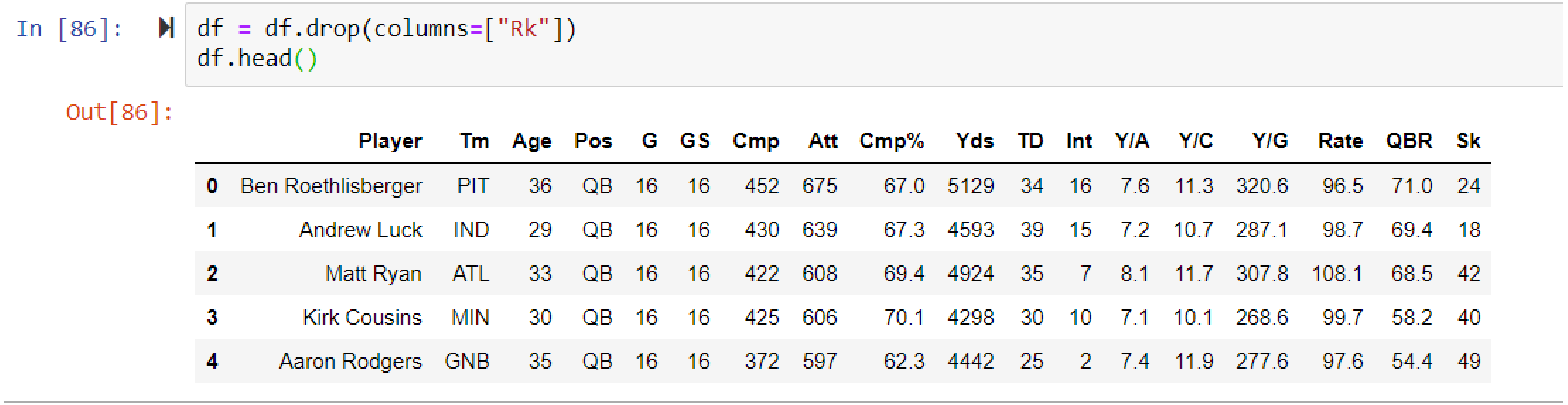
	Player
0	Ben Roethlisberger
1	Andrew Luck
2	Matt Ryan
3	Kirk Cousins
4	Aaron Rodgers

```
In [18]:  # Can't do that anymore!!!
          df["Tm"]

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method,
   2656             try:
-> 2657                 return self._engine.get_loc(key)
   2658             except KeyError:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()
```

How do we undo this?

Re-read the table into the DataFrame (re-run the cell where it is declared)

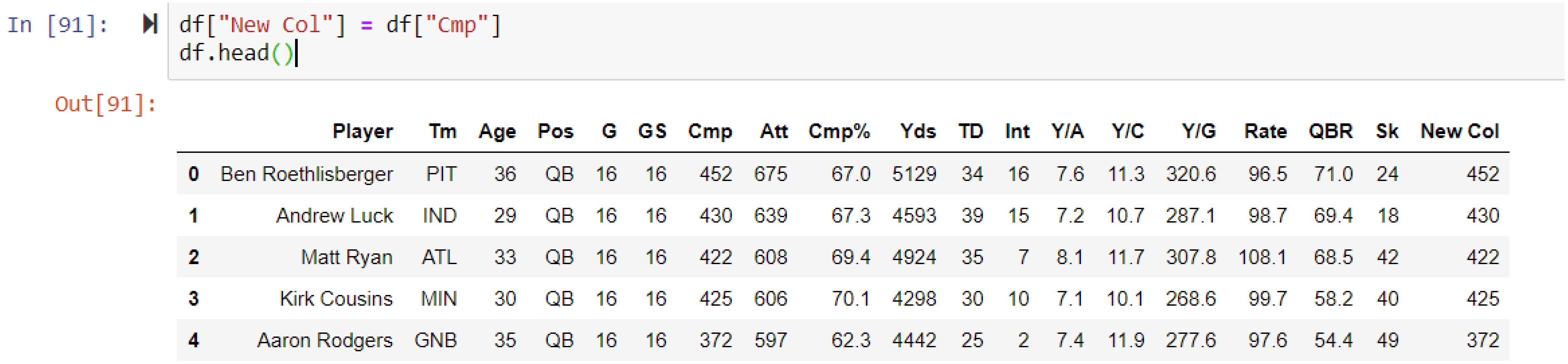
Delete a column



```
In [86]:  df = df.drop(columns=["Rk"])
          df.head()
```

```
Out[86]:
             Player   Tm  Age  Pos   G  GS  Cmp  Att  Cmp%   Yds  TD  Int  Y/A   Y/C    Y/G   Rate   QBR  Sk
0  Ben Roethlisberger  PIT   36   QB  16  16  452  675  67.0  5129  34   16  7.6  11.3  320.6   96.5  71.0  24
1        Andrew Luck   IND   29   QB  16  16  430  639  67.3  4593  39   15  7.2  10.7  287.1   98.7  69.4  18
2          Matt Ryan   ATL   33   QB  16  16  422  608  69.4  4924  35    7  8.1  11.7  307.8  108.1  68.5  42
3       Kirk Cousins   MIN   30   QB  16  16  425  606  70.1  4298  30   10  7.1  10.1  268.6   99.7  58.2  40
4     Aaron Rodgers   GNB   35   QB  16  16  372  597  62.3  4442  25    2  7.4  11.9  277.6   97.6  54.4  49
```

Create a new column!

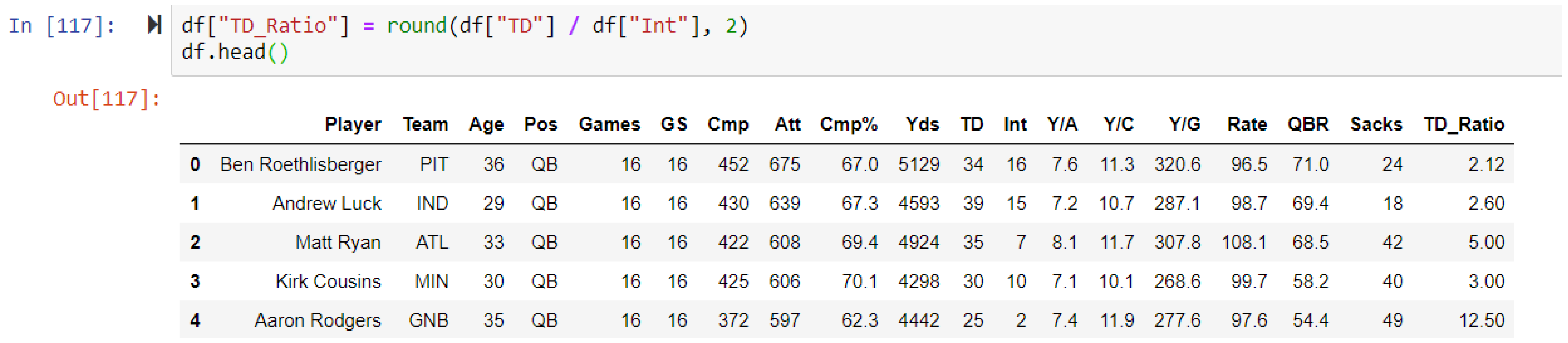


```
In [91]:  df["New Col"] = df["Cmp"]
           df.head()
```

Out[91]:

	Player	Tm	Age	Pos	G	GS	Cmp	Att	Cmp%	Yds	TD	Int	Y/A	Y/C	Y/G	Rate	QBR	Sk	New Col
0	Ben Roethlisberger	PIT	36	QB	16	16	452	675	67.0	5129	34	16	7.6	11.3	320.6	96.5	71.0	24	452
1	Andrew Luck	IND	29	QB	16	16	430	639	67.3	4593	39	15	7.2	10.7	287.1	98.7	69.4	18	430
2	Matt Ryan	ATL	33	QB	16	16	422	608	69.4	4924	35	7	8.1	11.7	307.8	108.1	68.5	42	422
3	Kirk Cousins	MIN	30	QB	16	16	425	606	70.1	4298	30	10	7.1	10.1	268.6	99.7	58.2	40	425
4	Aaron Rodgers	GNB	35	QB	16	16	372	597	62.3	4442	25	2	7.4	11.9	277.6	97.6	54.4	49	372

Create a new column using an operation

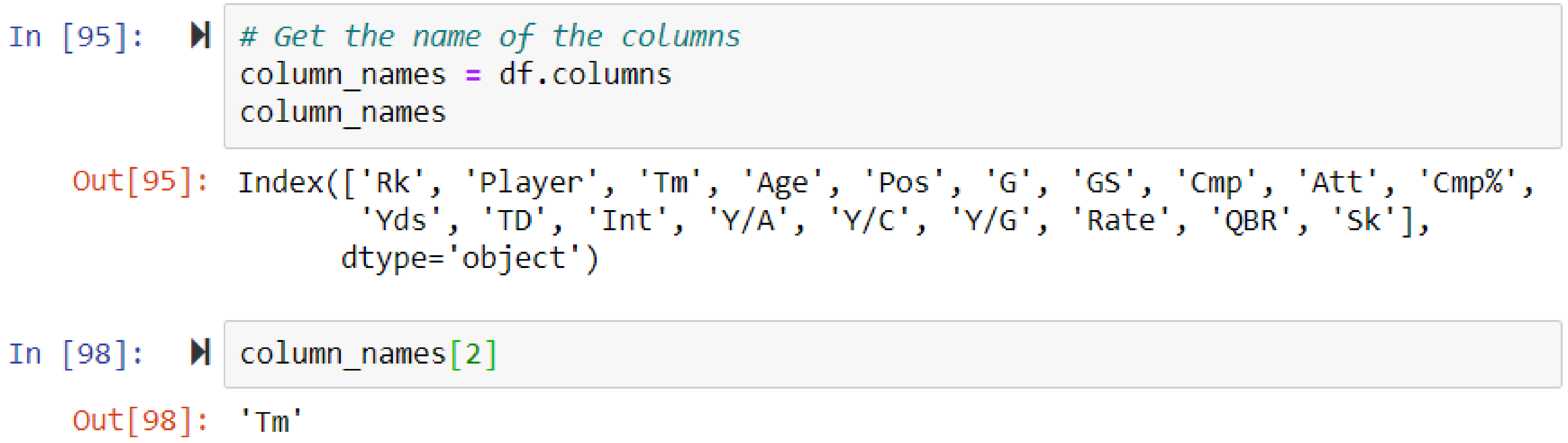


```
In [117]:  df["TD_Ratio"] = round(df["TD"] / df["Int"], 2)
           df.head()
```

Out[117]:

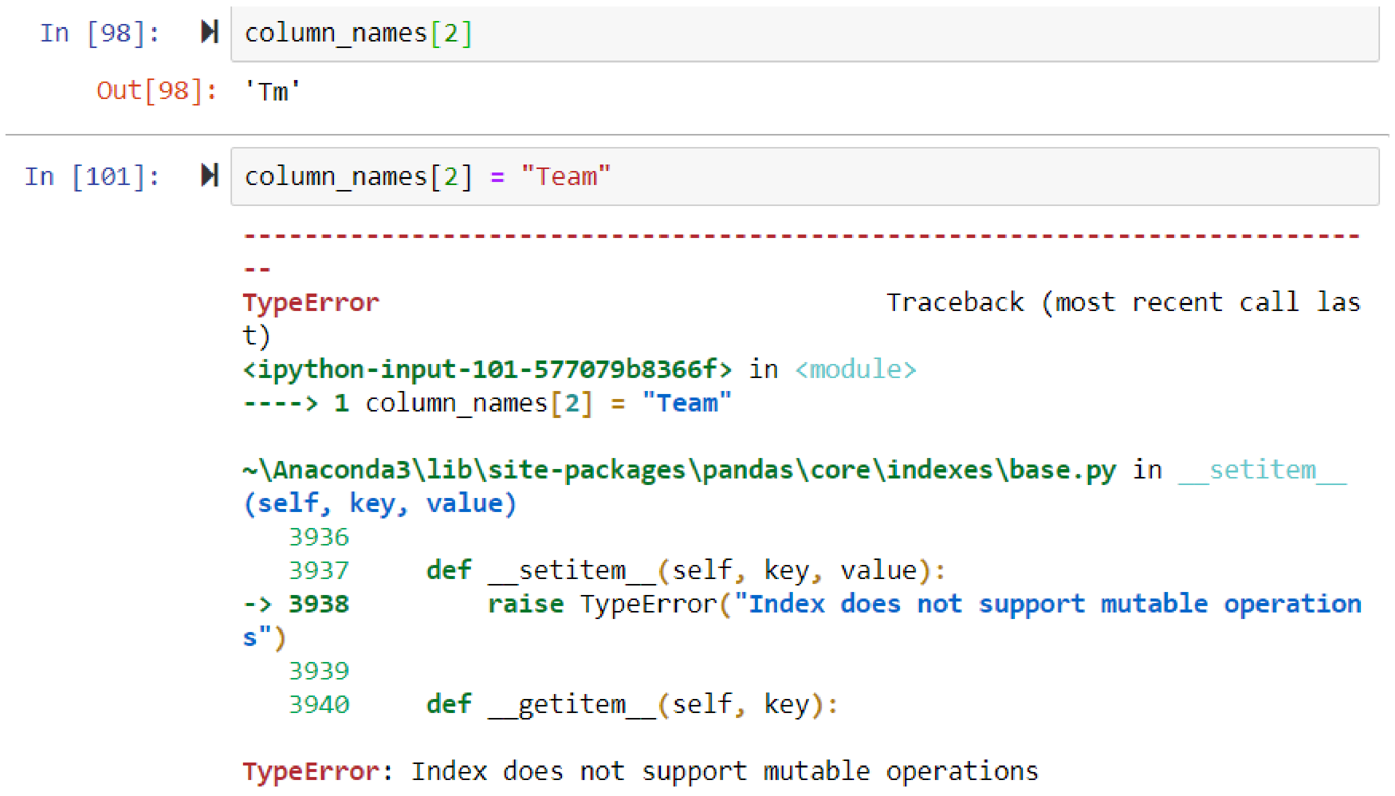
	Player	Team	Age	Pos	Games	GS	Cmp	Att	Cmp%	Yds	TD	Int	Y/A	Y/C	Y/G	Rate	QBR	Sacks	TD_Ratio
0	Ben Roethlisberger	PIT	36	QB	16	16	452	675	67.0	5129	34	16	7.6	11.3	320.6	96.5	71.0	24	2.12
1	Andrew Luck	IND	29	QB	16	16	430	639	67.3	4593	39	15	7.2	10.7	287.1	98.7	69.4	18	2.60
2	Matt Ryan	ATL	33	QB	16	16	422	608	69.4	4924	35	7	8.1	11.7	307.8	108.1	68.5	42	5.00
3	Kirk Cousins	MIN	30	QB	16	16	425	606	70.1	4298	30	10	7.1	10.1	268.6	99.7	58.2	40	3.00
4	Aaron Rodgers	GNB	35	QB	16	16	372	597	62.3	4442	25	2	7.4	11.9	277.6	97.6	54.4	49	12.50

Changing column names



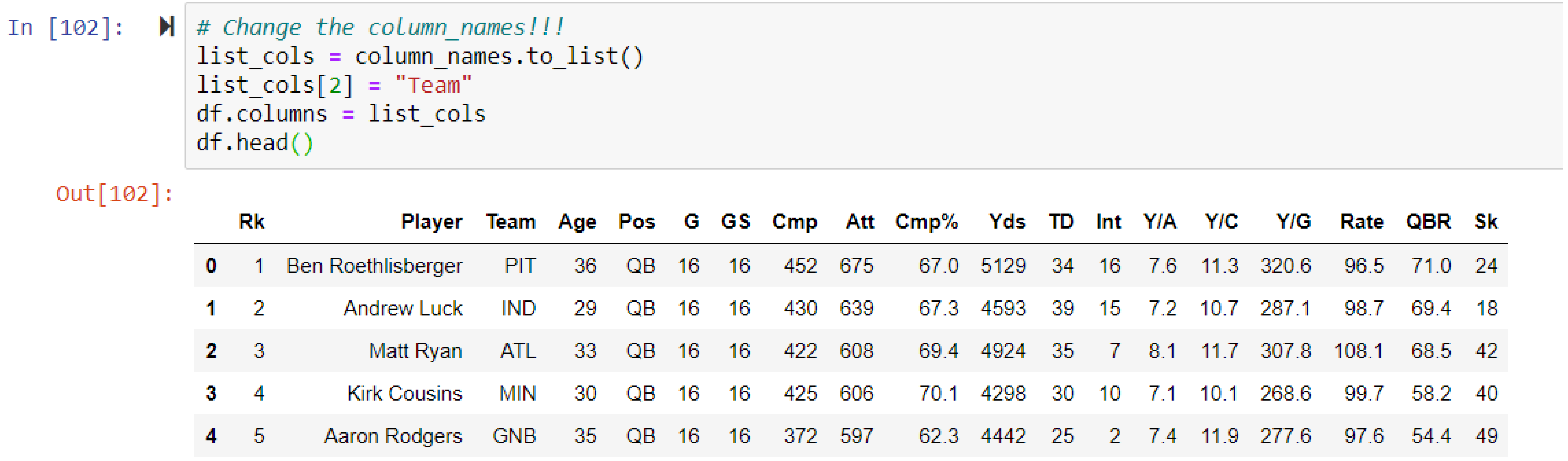
```
In [95]:  # Get the name of the columns
          column_names = df.columns
          column_names

Out[95]:  Index(['Rk', 'Player', 'Tm', 'Age', 'Pos', 'G', 'GS', 'Cmp', 'Att', 'Cmp%',
                 'Yds', 'TD', 'Int', 'Y/A', 'Y/C', 'Y/G', 'Rate', 'QBR', 'Sk'],
                dtype='object')

In [98]:  column_names[2]

Out[98]:  'Tm'
```

Not as easy as you'd think…



```
In [98]:   column_names[2]

Out[98]:   'Tm'

In [101]:  column_names[2] = "Team"

           ---------------------------------------------------------------------------
           --
           TypeError                                 Traceback (most recent call las
           t)
           <ipython-input-101-577079b8366f> in <module>
           ----> 1 column_names[2] = "Team"

           ~\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in __setitem__
           (self, key, value)
              3936
              3937     def __setitem__(self, key, value):
           -> 3938         raise TypeError("Index does not support mutable operation
           s")
              3939
              3940     def __getitem__(self, key):

           TypeError: Index does not support mutable operations
```

Way #1 – Beginner



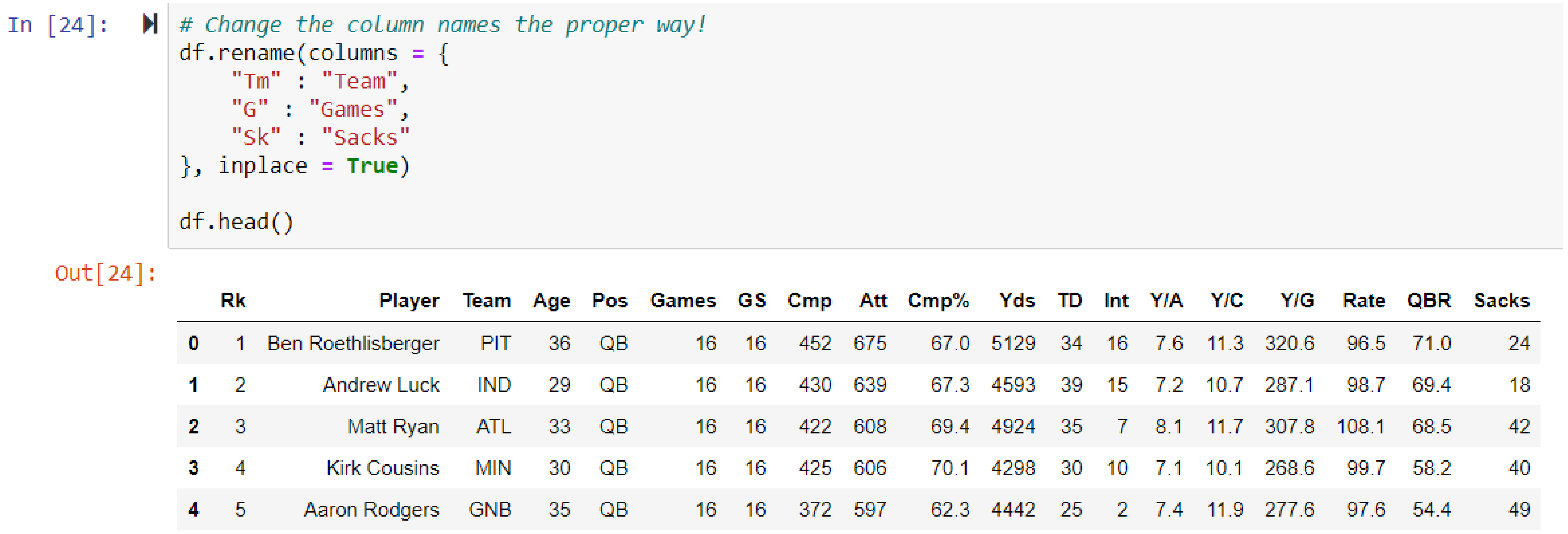
In [102]:

```
# Change the column_names!!!
list_cols = column_names.to_list()
list_cols[2] = "Team"
df.columns = list_cols
df.head()
```

Out[102]:

	Rk	Player	Team	Age	Pos	G	GS	Cmp	Att	Cmp%	Yds	TD	Int	Y/A	Y/C	Y/G	Rate	QBR	Sk
0	1	Ben Roethlisberger	PIT	36	QB	16	16	452	675	67.0	5129	34	16	7.6	11.3	320.6	96.5	71.0	24
1	2	Andrew Luck	IND	29	QB	16	16	430	639	67.3	4593	39	15	7.2	10.7	287.1	98.7	69.4	18
2	3	Matt Ryan	ATL	33	QB	16	16	422	608	69.4	4924	35	7	8.1	11.7	307.8	108.1	68.5	42
3	4	Kirk Cousins	MIN	30	QB	16	16	425	606	70.1	4298	30	10	7.1	10.1	268.6	99.7	58.2	40
4	5	Aaron Rodgers	GNB	35	QB	16	16	372	597	62.3	4442	25	2	7.4	11.9	277.6	97.6	54.4	49

Way #2 – Advanced



In [24]:

```
# Change the column names the proper way!
df.rename(columns = {
    "Tm" : "Team",
    "G" : "Games",
    "Sk" : "Sacks"
}, inplace = True)

df.head()
```

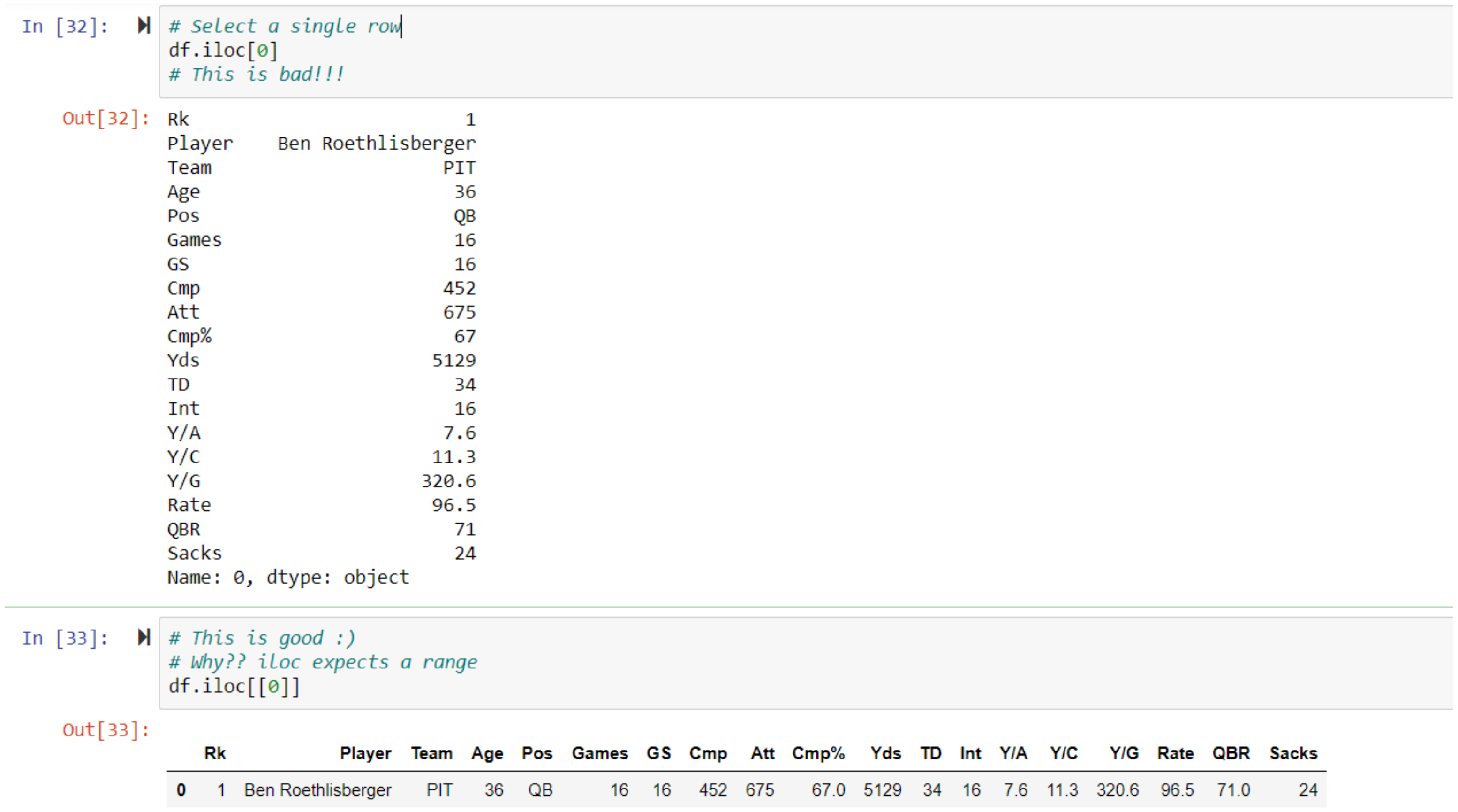
Out[24]:

	Rk	Player	Team	Age	Pos	Games	GS	Cmp	Att	Cmp%	Yds	TD	Int	Y/A	Y/C	Y/G	Rate	QBR	Sacks
0	1	Ben Roethlisberger	PIT	36	QB	16	16	452	675	67.0	5129	34	16	7.6	11.3	320.6	96.5	71.0	24
1	2	Andrew Luck	IND	29	QB	16	16	430	639	67.3	4593	39	15	7.2	10.7	287.1	98.7	69.4	18
2	3	Matt Ryan	ATL	33	QB	16	16	422	608	69.4	4924	35	7	8.1	11.7	307.8	108.1	68.5	42
3	4	Kirk Cousins	MIN	30	QB	16	16	425	606	70.1	4298	30	10	7.1	10.1	268.6	99.7	58.2	40
4	5	Aaron Rodgers	GNB	35	QB	16	16	372	597	62.3	4442	25	2	7.4	11.9	277.6	97.6	54.4	49

Selecting rows

- Now we'll look at getting data by rows.
- For rows, we have two options:
  - .loc - locates by name
  - .iloc- locates by numerical index

Selecting a single row using iloc

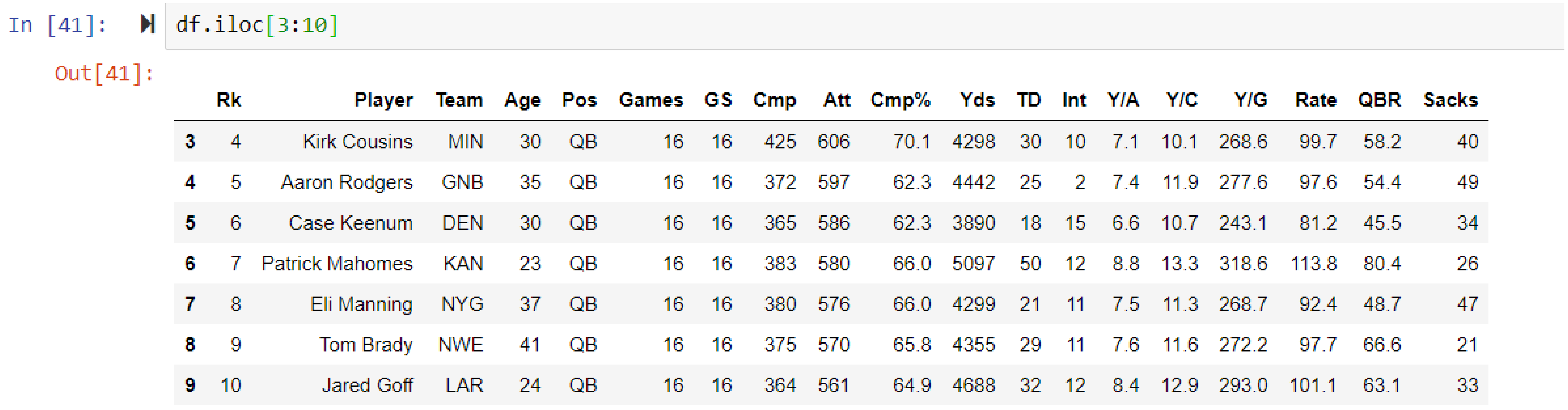


```
In [32]:  # Select a single row
          df.iloc[0]
          # This is bad!!!

Out[32]:  Rk                           1
          Player     Ben Roethlisberger
          Team                      PIT
          Age                        36
          Pos                        QB
          Games                      16
          GS                         16
          Cmp                       452
          Att                       675
          Cmp%                       67
          Yds                      5129
          TD                         34
          Int                        16
          Y/A                       7.6
          Y/C                      11.3
          Y/G                     320.6
          Rate                     96.5
          QBR                        71
          Sacks                      24
          Name: 0, dtype: object
```

```
In [33]:  # This is good :)
          # Why?? iloc expects a range
          df.iloc[[0]]

Out[33]:
```

	Rk	Player	Team	Age	Pos	Games	GS	Cmp	Att	Cmp%	Yds	TD	Int	Y/A	Y/C	Y/G	Rate	QBR	Sacks
0	1	Ben Roethlisberger	PIT	36	QB	16	16	452	675	67.0	5129	34	16	7.6	11.3	320.6	96.5	71.0	24

Selecting a range of rows using iloc



```
In [41]:  df.iloc[3:10]
```

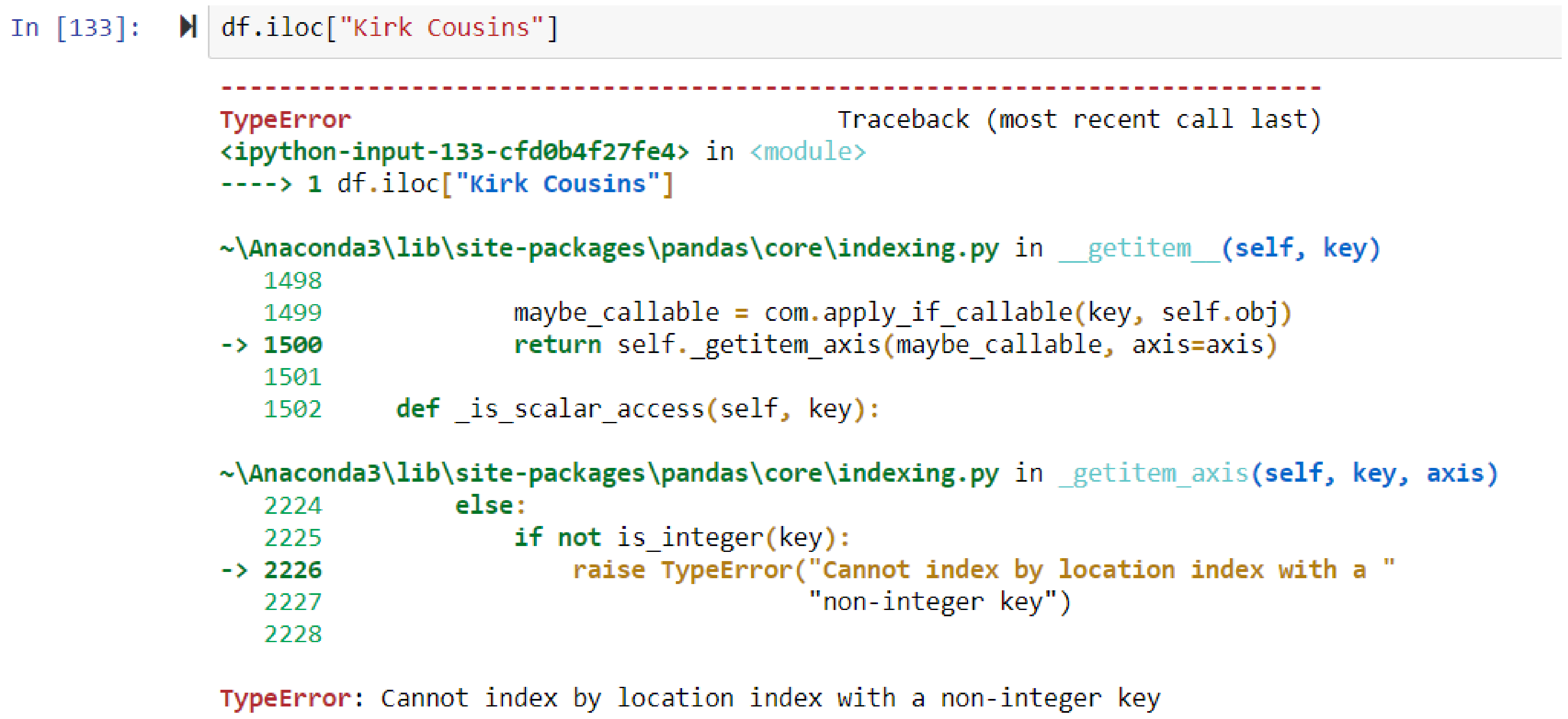
```
Out[41]:
```

	Rk	Player	Team	Age	Pos	Games	GS	Cmp	Att	Cmp%	Yds	TD	Int	Y/A	Y/C	Y/G	Rate	QBR	Sacks
3	4	Kirk Cousins	MIN	30	QB	16	16	425	606	70.1	4298	30	10	7.1	10.1	268.6	99.7	58.2	40
4	5	Aaron Rodgers	GNB	35	QB	16	16	372	597	62.3	4442	25	2	7.4	11.9	277.6	97.6	54.4	49
5	6	Case Keenum	DEN	30	QB	16	16	365	586	62.3	3890	18	15	6.6	10.7	243.1	81.2	45.5	34
6	7	Patrick Mahomes	KAN	23	QB	16	16	383	580	66.0	5097	50	12	8.8	13.3	318.6	113.8	80.4	26
7	8	Eli Manning	NYG	37	QB	16	16	380	576	66.0	4299	21	11	7.5	11.3	268.7	92.4	48.7	47
8	9	Tom Brady	NWE	41	QB	16	16	375	570	65.8	4355	29	11	7.6	11.6	272.2	97.7	66.6	21
9	10	Jared Goff	LAR	24	QB	16	16	364	561	64.9	4688	32	12	8.4	12.9	293.0	101.1	63.1	33

What is the “index” column?

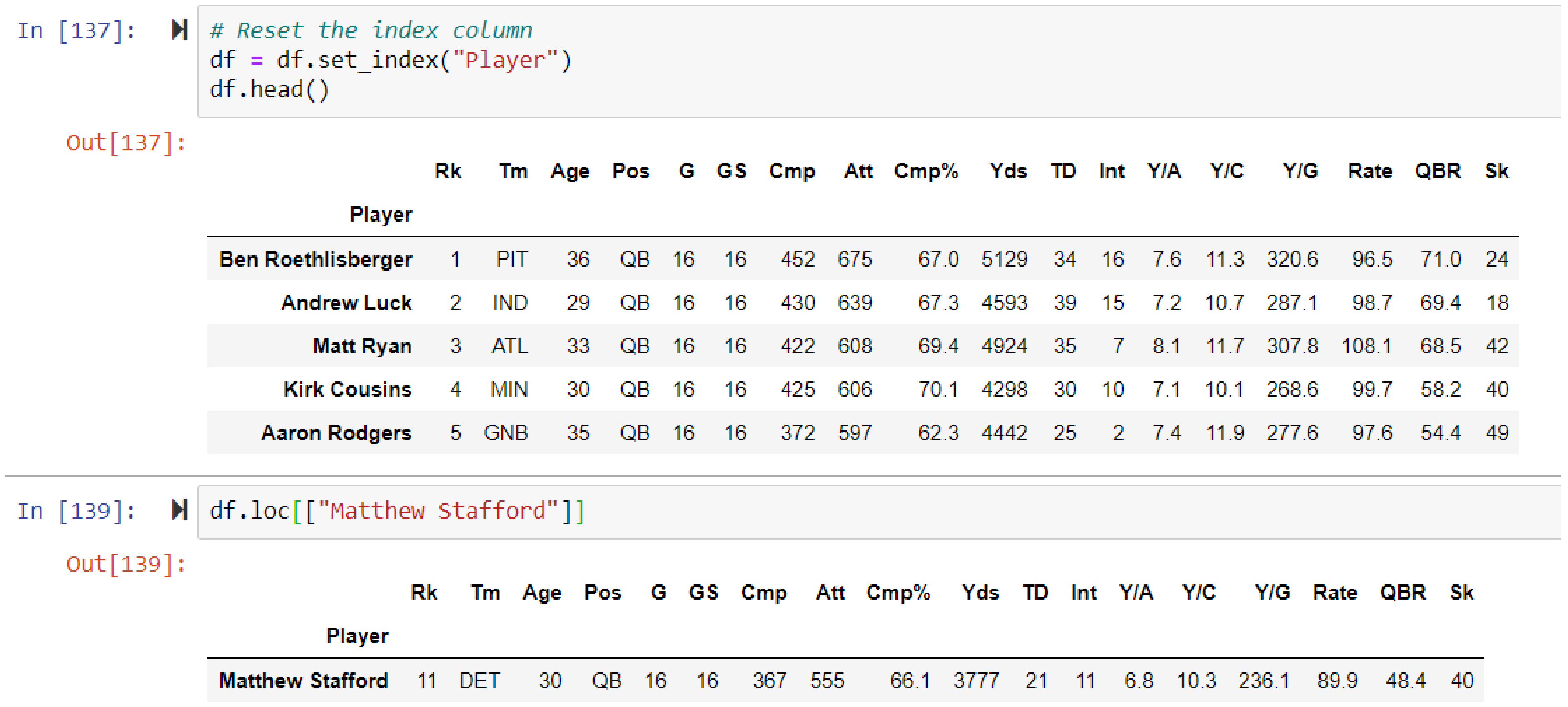
- The index column is how you access specific rows
- The default will always be an increasing index unless it is set
- Useful for ranges and determining order
    - Not much else
- Why should we change the index??
    - Next slide

Example of a good time to change the index



```
In [133]:    df.iloc["Kirk Cousins"]

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-133-cfd0b4f27fe4> in <module>
----> 1 df.iloc["Kirk Cousins"]

~\Anaconda3\lib\site-packages\pandas\core\indexing.py in __getitem__(self, key)
   1498 
   1499             maybe_callable = com.apply_if_callable(key, self.obj)
-> 1500             return self._getitem_axis(maybe_callable, axis=axis)
   1501 
   1502     def _is_scalar_access(self, key):

~\Anaconda3\lib\site-packages\pandas\core\indexing.py in _getitem_axis(self, key, axis)
   2224         else:
   2225             if not is_integer(key):
-> 2226                 raise TypeError("Cannot index by location index with a "
   2227                                 "non-integer key")
   2228 

TypeError: Cannot index by location index with a non-integer key
```

How to change the index



In [137]:

```python
# Reset the index column
df = df.set_index("Player")
df.head()
```

Out[137]:

Player	Rk	Tm	Age	Pos	G	GS	Cmp	Att	Cmp%	Yds	TD	Int	Y/A	Y/C	Y/G	Rate	QBR	Sk
Ben Roethlisberger	1	PIT	36	QB	16	16	452	675	67.0	5129	34	16	7.6	11.3	320.6	96.5	71.0	24
Andrew Luck	2	IND	29	QB	16	16	430	639	67.3	4593	39	15	7.2	10.7	287.1	98.7	69.4	18
Matt Ryan	3	ATL	33	QB	16	16	422	608	69.4	4924	35	7	8.1	11.7	307.8	108.1	68.5	42
Kirk Cousins	4	MIN	30	QB	16	16	425	606	70.1	4298	30	10	7.1	10.1	268.6	99.7	58.2	40
Aaron Rodgers	5	GNB	35	QB	16	16	372	597	62.3	4442	25	2	7.4	11.9	277.6	97.6	54.4	49

In [139]:

```python
df.loc[["Matthew Stafford"]]
```

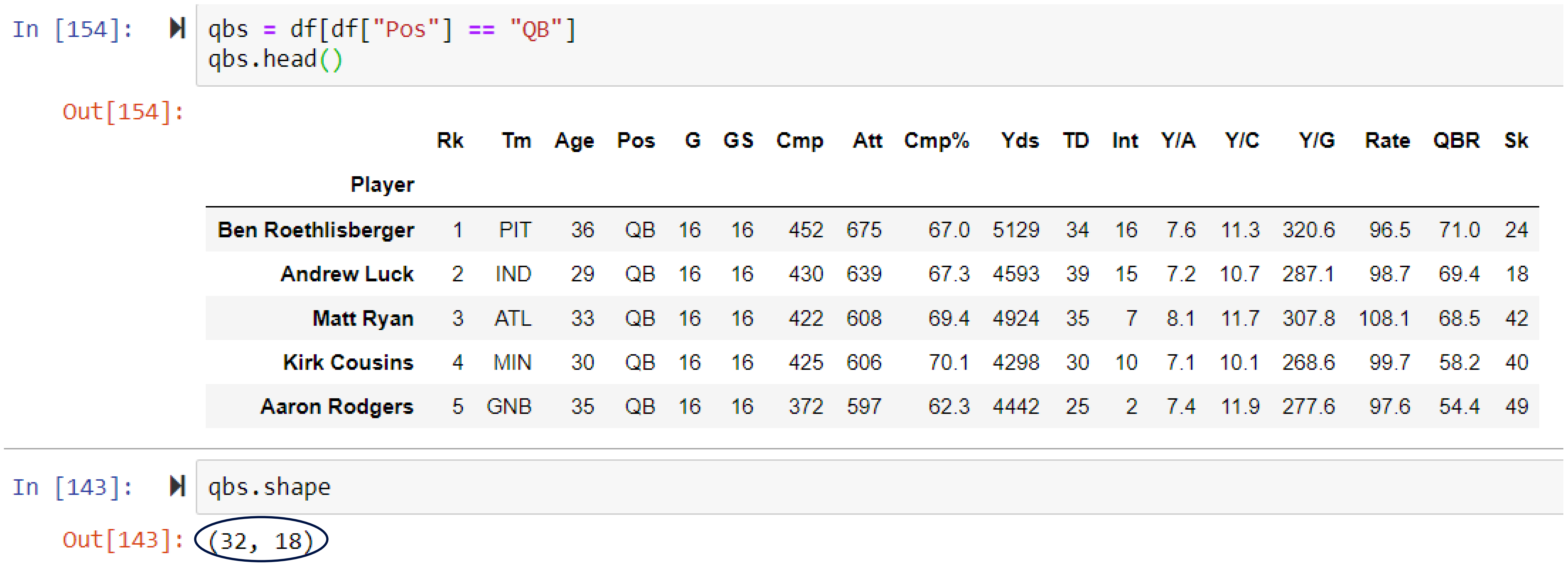
Out[139]:

Player	Rk	Tm	Age	Pos	G	GS	Cmp	Att	Cmp%	Yds	TD	Int	Y/A	Y/C	Y/G	Rate	QBR	Sk
Matthew Stafford	11	DET	30	QB	16	16	367	555	66.1	3777	21	11	6.8	10.3	236.1	89.9	48.4	40

Selecting rows based on conditions

- Basic operation that is extremely powerful
- Allows you to view data based on criterion you want to view
- Why is filtering useful?

Filtering by one qualification



In [154]:
```python
qbs = df[df["Pos"] == "QB"]
qbs.head()
```

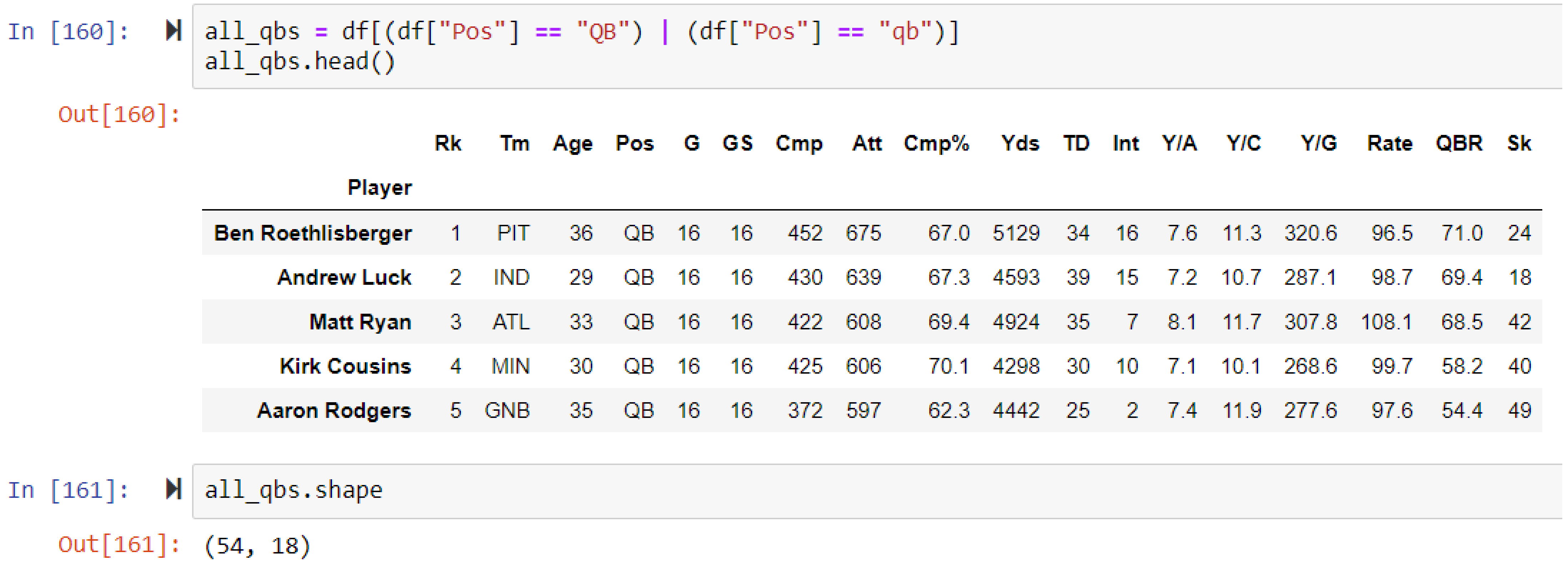
Out[154]:

	Rk	Tm	Age	Pos	G	GS	Cmp	Att	Cmp%	Yds	TD	Int	Y/A	Y/C	Y/G	Rate	QBR	Sk
Player																		
Ben Roethlisberger	1	PIT	36	QB	16	16	452	675	67.0	5129	34	16	7.6	11.3	320.6	96.5	71.0	24
Andrew Luck	2	IND	29	QB	16	16	430	639	67.3	4593	39	15	7.2	10.7	287.1	98.7	69.4	18
Matt Ryan	3	ATL	33	QB	16	16	422	608	69.4	4924	35	7	8.1	11.7	307.8	108.1	68.5	42
Kirk Cousins	4	MIN	30	QB	16	16	425	606	70.1	4298	30	10	7.1	10.1	268.6	99.7	58.2	40
Aaron Rodgers	5	GNB	35	QB	16	16	372	597	62.3	4442	25	2	7.4	11.9	277.6	97.6	54.4	49

In [143]:
```python
qbs.shape
```

Out[143]:
```
(32, 18)
```

Filtering by multiple conditions



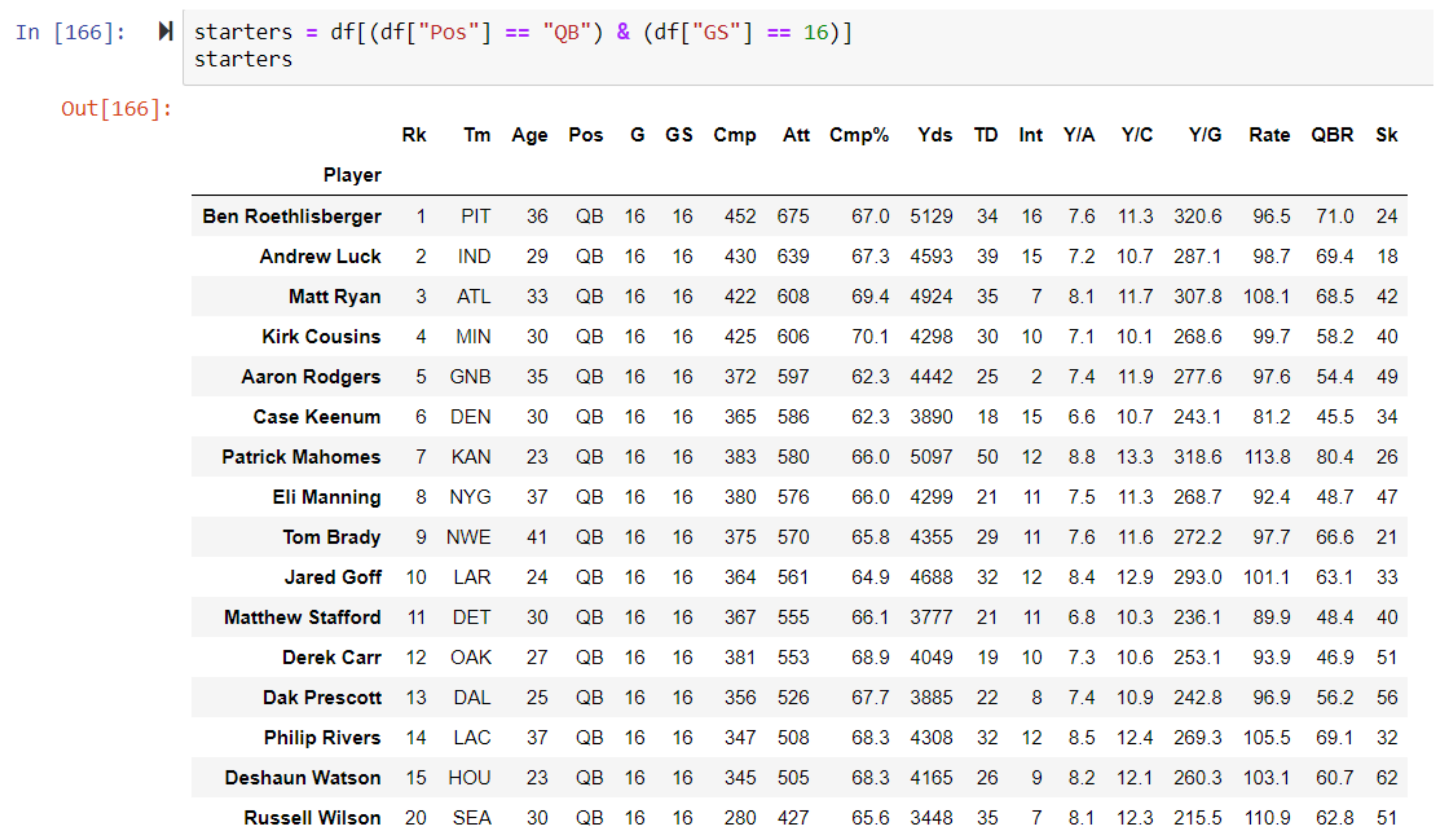
```
In [160]:  all_qbs = df[(df["Pos"] == "QB") | (df["Pos"] == "qb")]
           all_qbs.head()
```

Out[160]:

	Rk	Tm	Age	Pos	G	GS	Cmp	Att	Cmp%	Yds	TD	Int	Y/A	Y/C	Y/G	Rate	QBR	Sk
Player																		
Ben Roethlisberger	1	PIT	36	QB	16	16	452	675	67.0	5129	34	16	7.6	11.3	320.6	96.5	71.0	24
Andrew Luck	2	IND	29	QB	16	16	430	639	67.3	4593	39	15	7.2	10.7	287.1	98.7	69.4	18
Matt Ryan	3	ATL	33	QB	16	16	422	608	69.4	4924	35	7	8.1	11.7	307.8	108.1	68.5	42
Kirk Cousins	4	MIN	30	QB	16	16	425	606	70.1	4298	30	10	7.1	10.1	268.6	99.7	58.2	40
Aaron Rodgers	5	GNB	35	QB	16	16	372	597	62.3	4442	25	2	7.4	11.9	277.6	97.6	54.4	49

```
In [161]:  all_qbs.shape

Out[161]:  (54, 18)
```

Additional Example – What does this show?

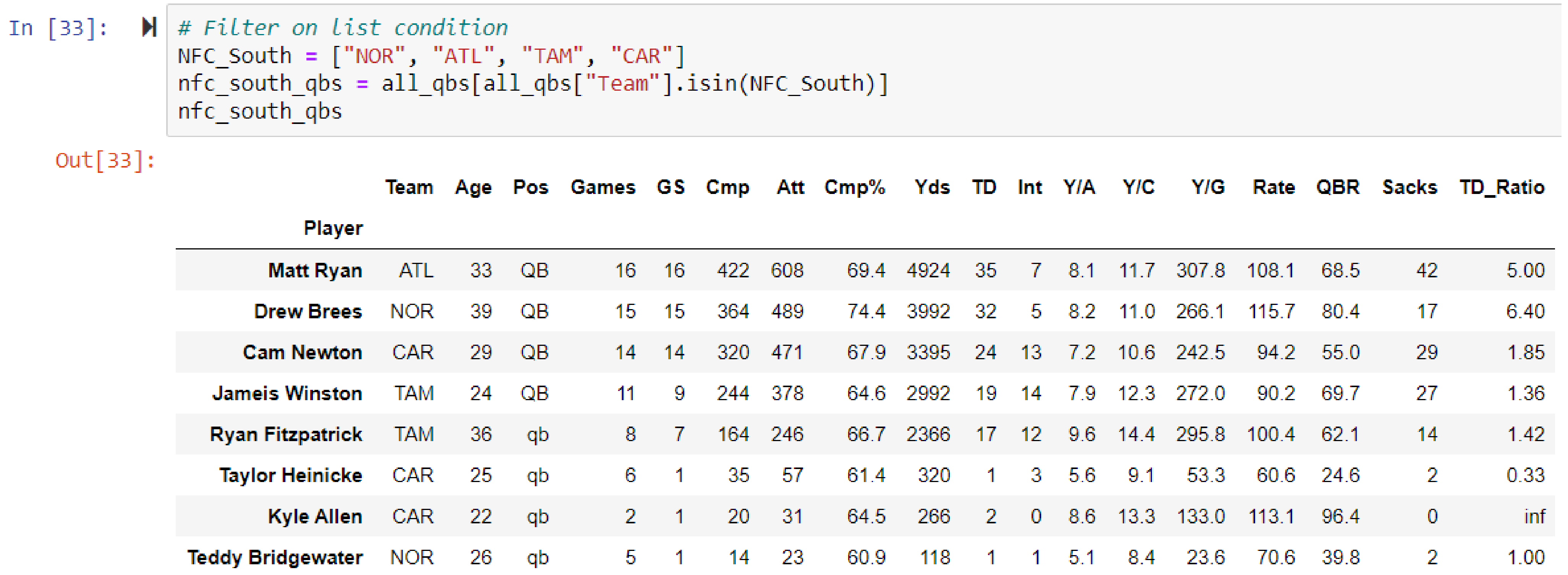


```
In [166]:  starters = df[(df["Pos"] == "QB") & (df["GS"] == 16)]
           starters
```

Out[166]:

Player	Rk	Tm	Age	Pos	G	GS	Cmp	Att	Cmp%	Yds	TD	Int	Y/A	Y/C	Y/G	Rate	QBR	Sk
Ben Roethlisberger	1	PIT	36	QB	16	16	452	675	67.0	5129	34	16	7.6	11.3	320.6	96.5	71.0	24
Andrew Luck	2	IND	29	QB	16	16	430	639	67.3	4593	39	15	7.2	10.7	287.1	98.7	69.4	18
Matt Ryan	3	ATL	33	QB	16	16	422	608	69.4	4924	35	7	8.1	11.7	307.8	108.1	68.5	42
Kirk Cousins	4	MIN	30	QB	16	16	425	606	70.1	4298	30	10	7.1	10.1	268.6	99.7	58.2	40
Aaron Rodgers	5	GNB	35	QB	16	16	372	597	62.3	4442	25	2	7.4	11.9	277.6	97.6	54.4	49
Case Keenum	6	DEN	30	QB	16	16	365	586	62.3	3890	18	15	6.6	10.7	243.1	81.2	45.5	34
Patrick Mahomes	7	KAN	23	QB	16	16	383	580	66.0	5097	50	12	8.8	13.3	318.6	113.8	80.4	26
Eli Manning	8	NYG	37	QB	16	16	380	576	66.0	4299	21	11	7.5	11.3	268.7	92.4	48.7	47
Tom Brady	9	NWE	41	QB	16	16	375	570	65.8	4355	29	11	7.6	11.6	272.2	97.7	66.6	21
Jared Goff	10	LAR	24	QB	16	16	364	561	64.9	4688	32	12	8.4	12.9	293.0	101.1	63.1	33
Matthew Stafford	11	DET	30	QB	16	16	367	555	66.1	3777	21	11	6.8	10.3	236.1	89.9	48.4	40
Derek Carr	12	OAK	27	QB	16	16	381	553	68.9	4049	19	10	7.3	10.6	253.1	93.9	46.9	51
Dak Prescott	13	DAL	25	QB	16	16	356	526	67.7	3885	22	8	7.4	10.9	242.8	96.9	56.2	56
Philip Rivers	14	LAC	37	QB	16	16	347	508	68.3	4308	32	12	8.5	12.4	269.3	105.5	69.1	32
Deshaun Watson	15	HOU	23	QB	16	16	345	505	68.3	4165	26	9	8.2	12.1	260.3	103.1	60.7	62
Russell Wilson	20	SEA	30	QB	16	16	280	427	65.6	3448	35	7	8.1	12.3	215.5	110.9	62.8	51

Check if element is in a list



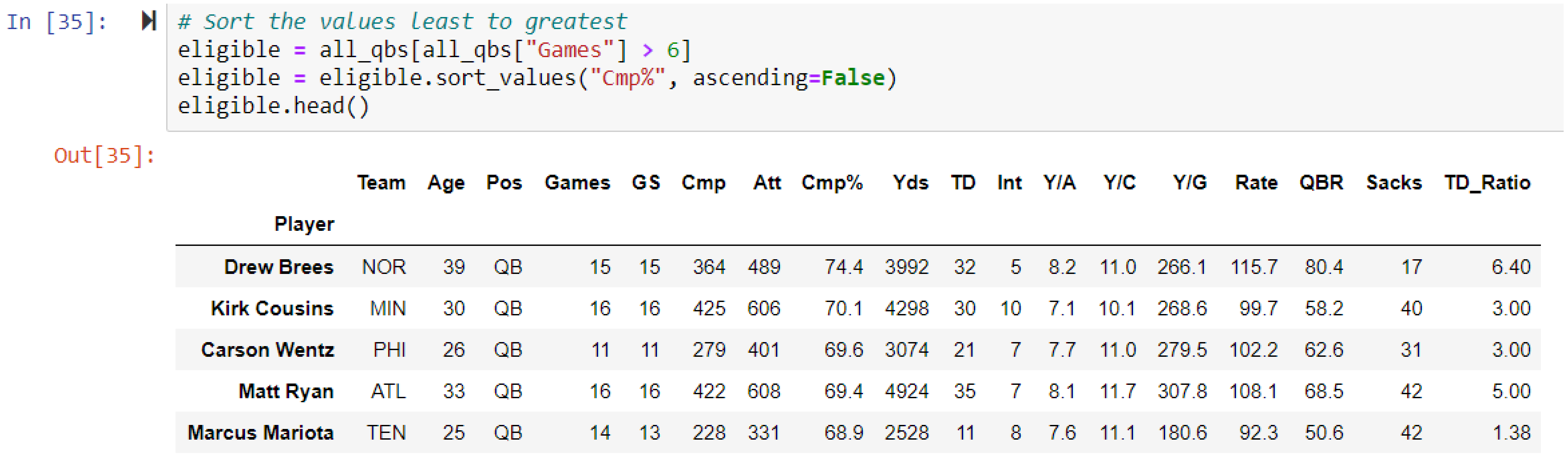
In [33]:

```python
# Filter on list condition
NFC_South = ["NOR", "ATL", "TAM", "CAR"]
nfc_south_qbs = all_qbs[all_qbs["Team"].isin(NFC_South)]
nfc_south_qbs
```

Out[33]:

	Team	Age	Pos	Games	GS	Cmp	Att	Cmp%	Yds	TD	Int	Y/A	Y/C	Y/G	Rate	QBR	Sacks	TD_Ratio
Player																		
Matt Ryan	ATL	33	QB	16	16	422	608	69.4	4924	35	7	8.1	11.7	307.8	108.1	68.5	42	5.00
Drew Brees	NOR	39	QB	15	15	364	489	74.4	3992	32	5	8.2	11.0	266.1	115.7	80.4	17	6.40
Cam Newton	CAR	29	QB	14	14	320	471	67.9	3395	24	13	7.2	10.6	242.5	94.2	55.0	29	1.85
Jameis Winston	TAM	24	QB	11	9	244	378	64.6	2992	19	14	7.9	12.3	272.0	90.2	69.7	27	1.36
Ryan Fitzpatrick	TAM	36	qb	8	7	164	246	66.7	2366	17	12	9.6	14.4	295.8	100.4	62.1	14	1.42
Taylor Heinicke	CAR	25	qb	6	1	35	57	61.4	320	1	3	5.6	9.1	53.3	60.6	24.6	2	0.33
Kyle Allen	CAR	22	qb	2	1	20	31	64.5	266	2	0	8.6	13.3	133.0	113.1	96.4	0	inf
Teddy Bridgewater	NOR	26	qb	5	1	14	23	60.9	118	1	1	5.1	8.4	23.6	70.6	39.8	2	1.00

Sort Values



In [35]:

```
# Sort the values least to greatest
eligible = all_qbs[all_qbs["Games"] > 6]
eligible = eligible.sort_values("Cmp%", ascending=False)
eligible.head()
```

Out[35]:

	Team	Age	Pos	Games	GS	Cmp	Att	Cmp%	Yds	TD	Int	Y/A	Y/C	Y/G	Rate	QBR	Sacks	TD_Ratio
Player																		
Drew Brees	NOR	39	QB	15	15	364	489	74.4	3992	32	5	8.2	11.0	266.1	115.7	80.4	17	6.40
Kirk Cousins	MIN	30	QB	16	16	425	606	70.1	4298	30	10	7.1	10.1	268.6	99.7	58.2	40	3.00
Carson Wentz	PHI	26	QB	11	11	279	401	69.6	3074	21	7	7.7	11.0	279.5	102.2	62.6	31	3.00
Matt Ryan	ATL	33	QB	16	16	422	608	69.4	4924	35	7	8.1	11.7	307.8	108.1	68.5	42	5.00
Marcus Mariota	TEN	25	QB	14	13	228	331	68.9	2528	11	8	7.6	11.1	180.6	92.3	50.6	42	1.38

Tips for more efficient data manipulation

- Never modify your original DataFrame in the middle of your notebook
  - Perform all cleaning operations in the beginning
  - Allows you to keep working linearly
- Store modified tables in placeholder tables
  - Adding or deleting rows
  - Conditional filters
- Use helpful names