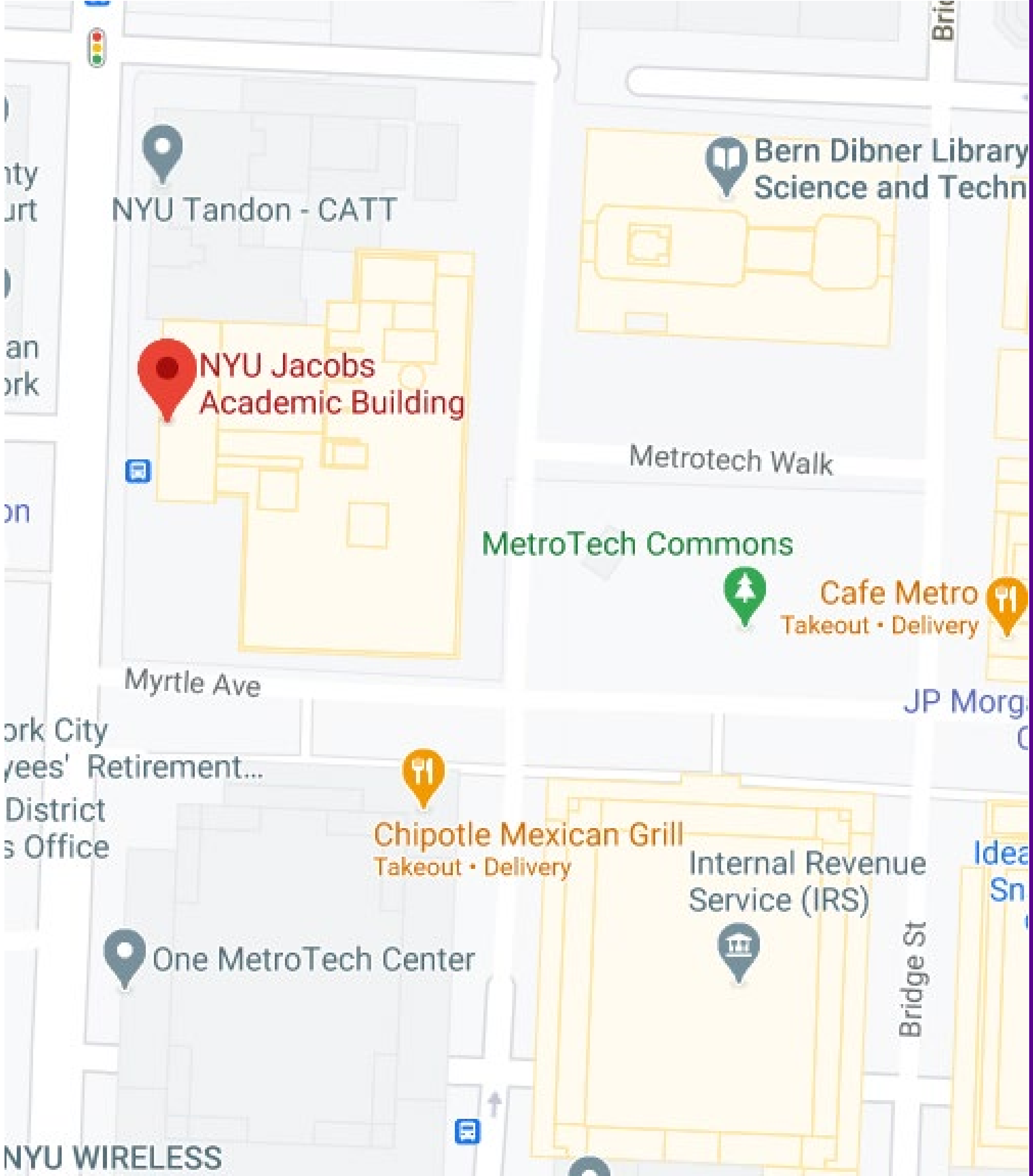
NYU | TANDON SCHOOL OF ENGINEERING

Lecture 5
Tables

Programming for Business Analytics
MG-GY 8401

Agenda

- Accessing Entries
- Filtering Records
- Joining Tables

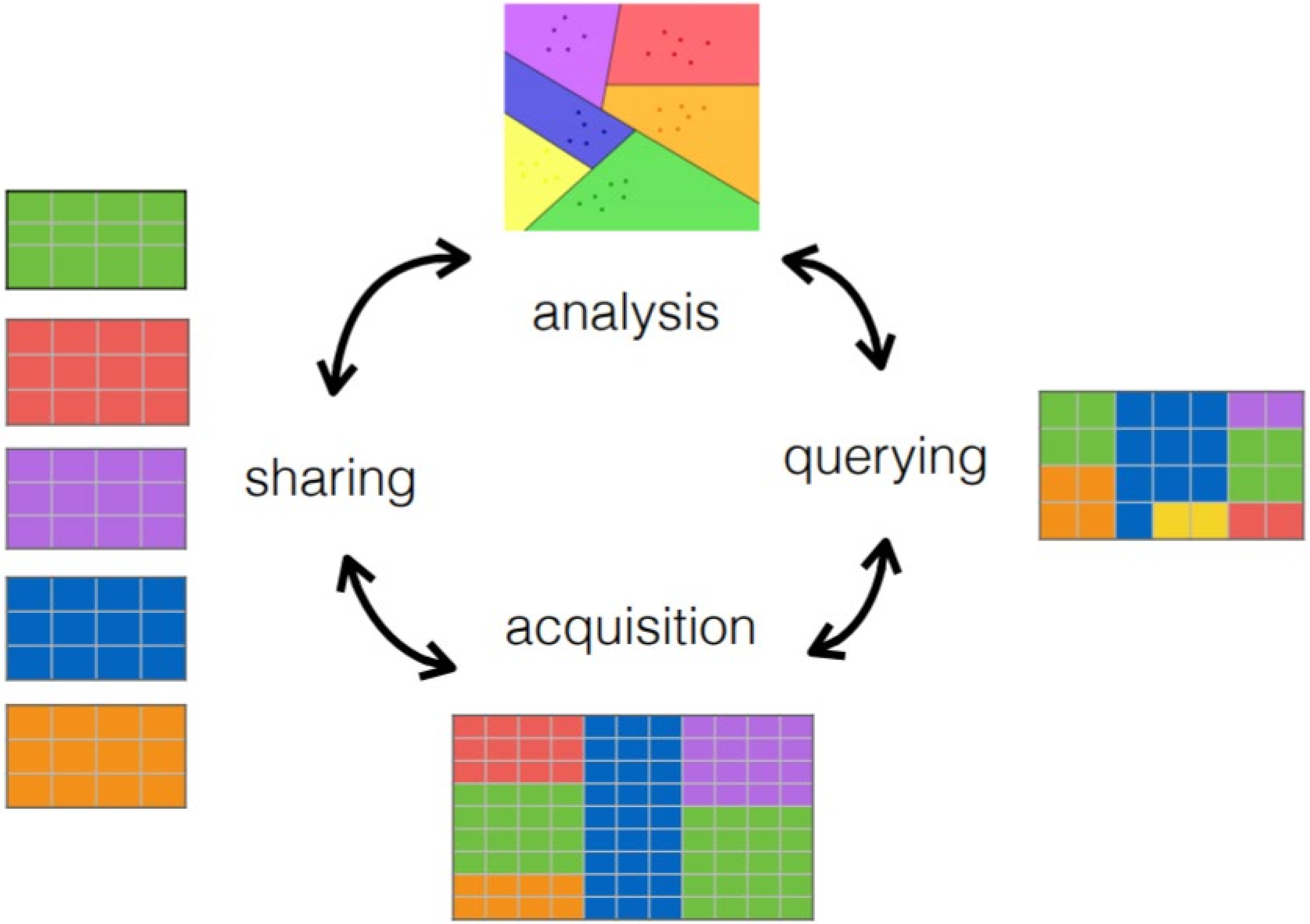


NYU TANDON SCHOOL OF ENGINEERING

Logistics

- Homework
  - Homework 5
  - Homework 4
  - Project

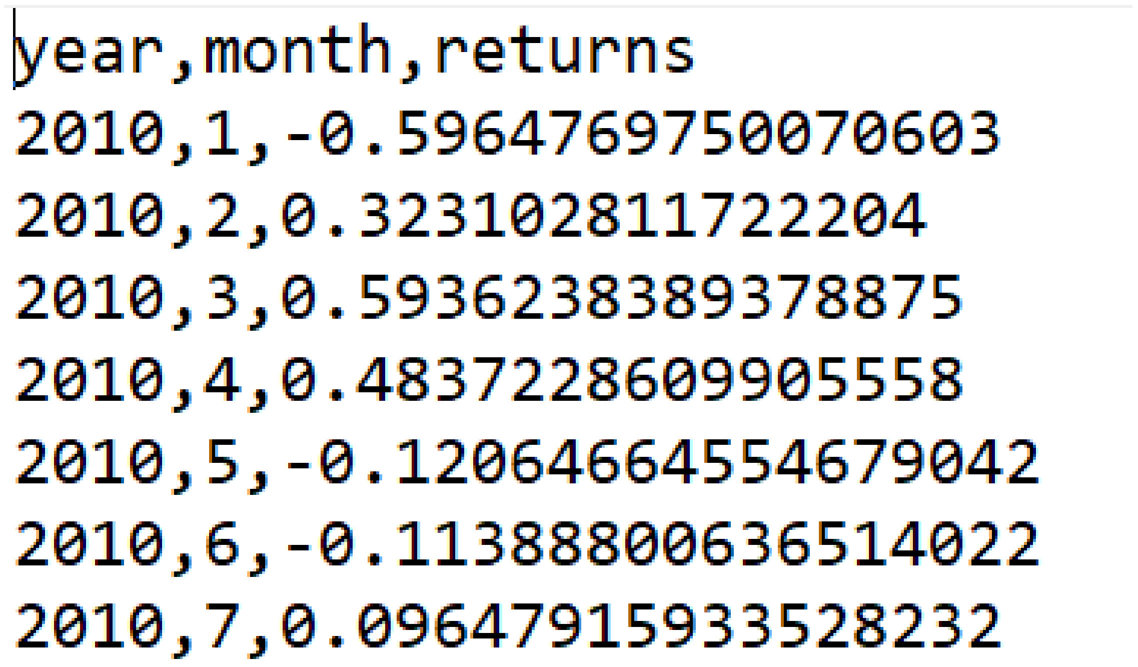
Case Study



File I/O

A comma separated value (csv) format can store tabular data

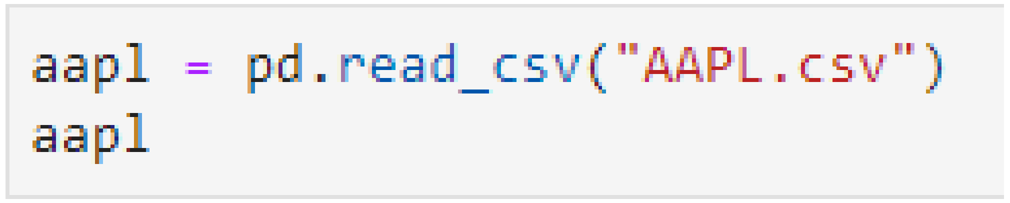
- We separate rows with different lines
- We separate columns with commas
- The first row indicates the headings of the columns



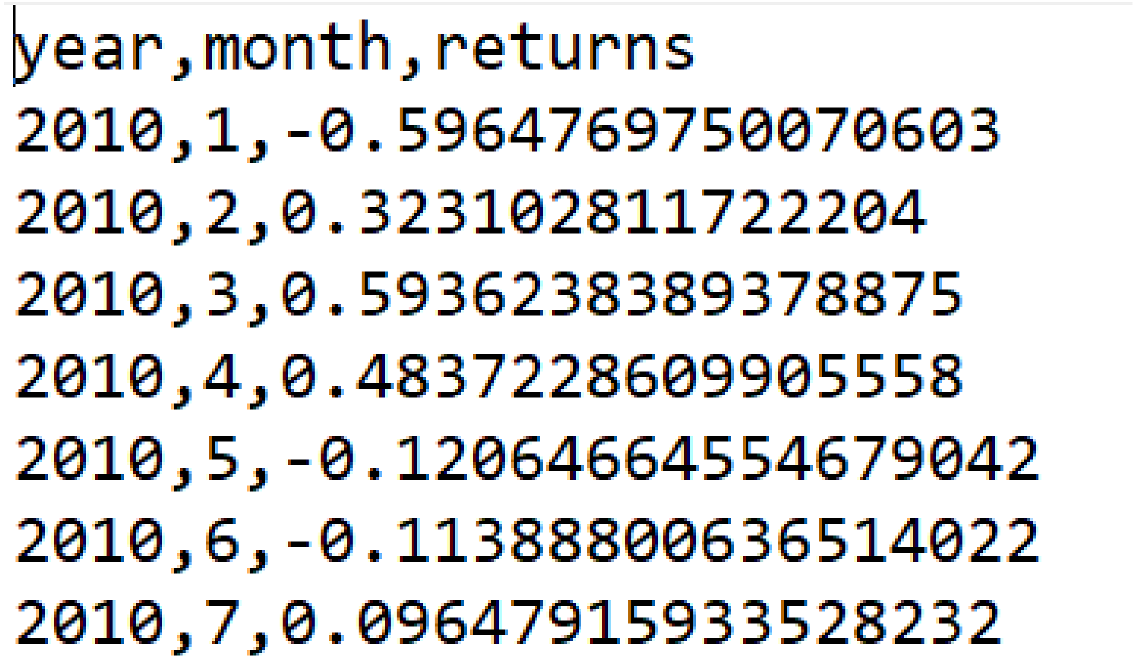
```
year,month,returns
2010,1,-0.596476975007060З
2010,2,0.32310281172220４
2010,3,0.593623838937887５
2010,4,0.483722860990555８
2010,5,-0.120646645467904２
2010,6,-0.113888006365140２２
2010,7,0.096479159352823２
```

File I/O

We can load the contents of a file in comma separated value (csv) format with read_csv

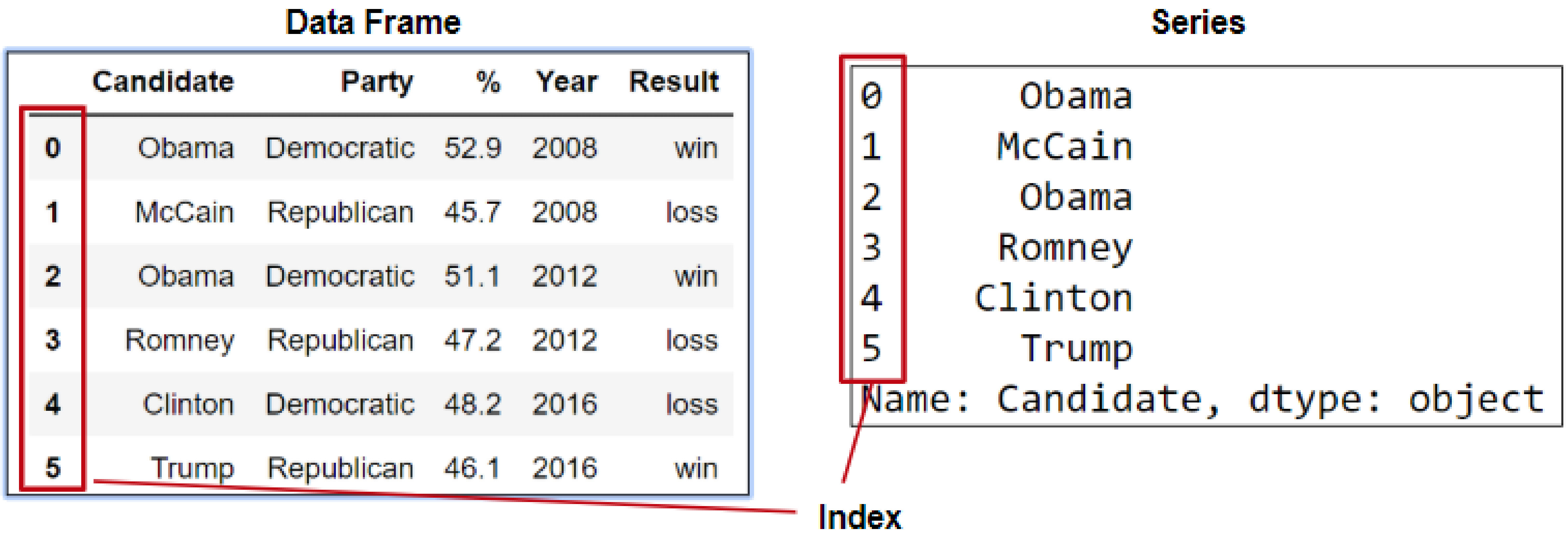


```
aapl = pd.read_csv("AAPL.csv")
aapl
```



```
year,month,returns
2010,1,-0.596476975007060З
2010,2,0.32310281172204
2010,3,0.593623838937887
2010,4,0.483722860990555
2010,5,-0.120646645467904
2010,6,-0.113888006365140
2010,7,0.096479159352823
```

Tables



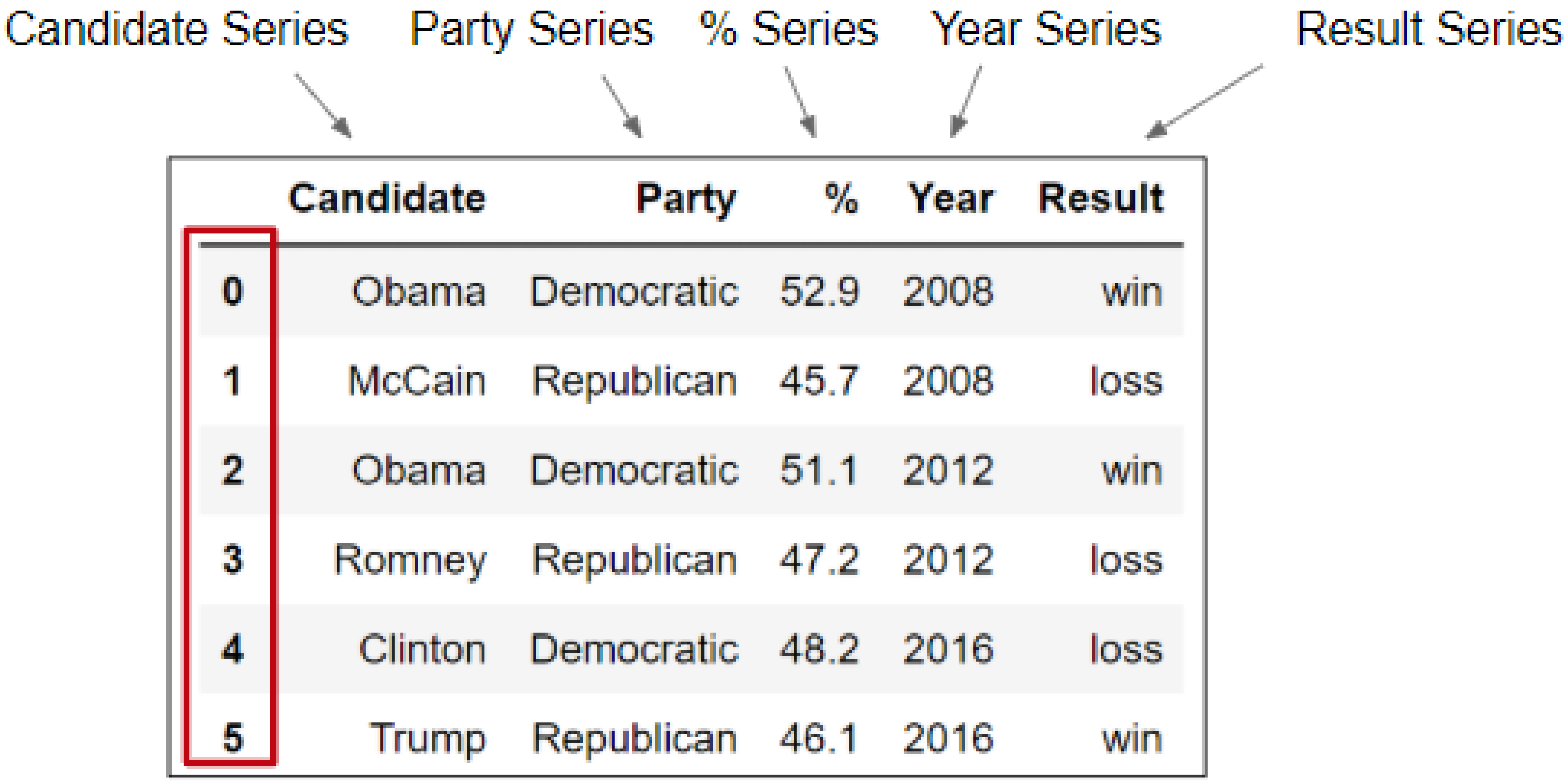
Data Frame

	Candidate	Party	%	Year	Result
0	Obama	Democratic	52.9	2008	win
1	McCain	Republican	45.7	2008	loss
2	Obama	Democratic	51.1	2012	win
3	Romney	Republican	47.2	2012	loss
4	Clinton	Democratic	48.2	2016	loss
5	Trump	Republican	46.1	2016	win

Series

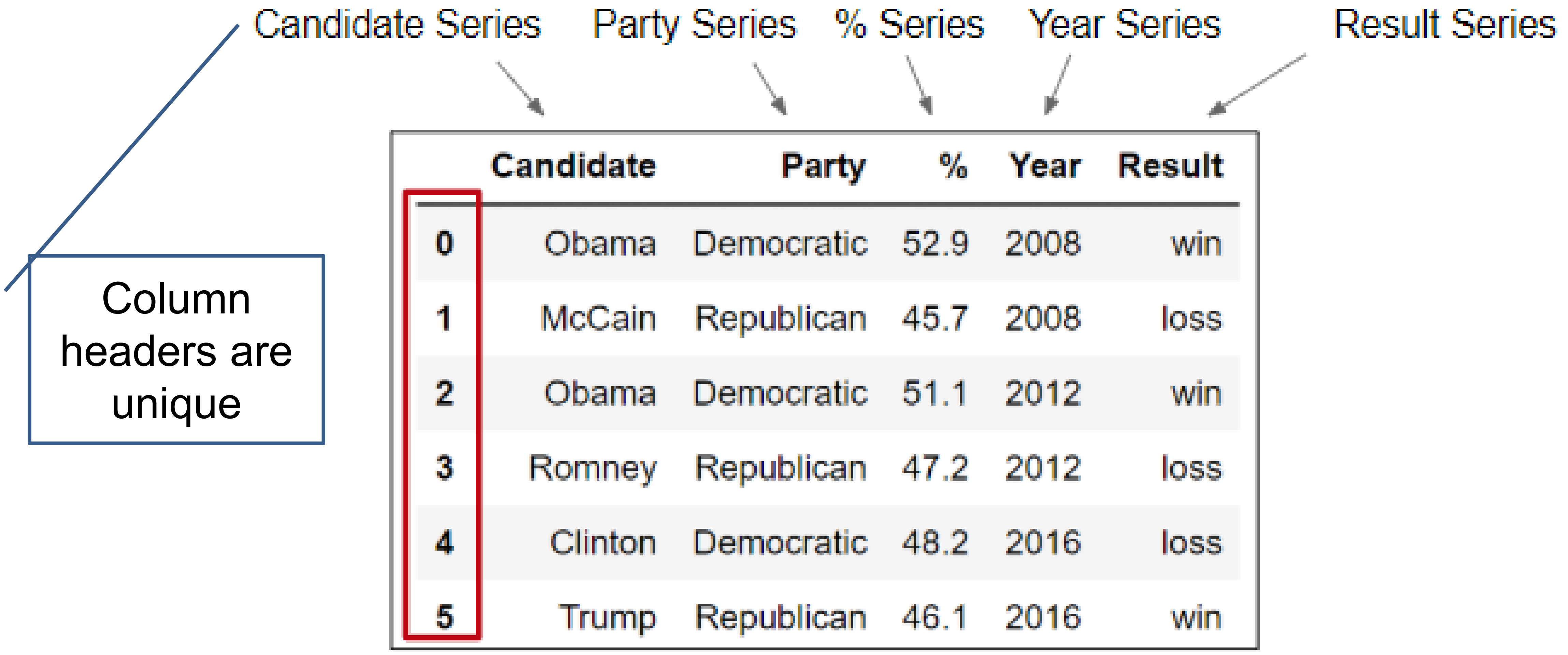
```
0        Obama
1       McCain
2        Obama
3       Romney
4      Clinton
5        Trump
Name: Candidate, dtype: object
```

Tables


Candidate Series
Party Series
% Series
Year Series
Result Series

	Candidate	Party	%	Year	Result
0	Obama	Democratic	52.9	2008	win
1	McCain	Republican	45.7	2008	loss
2	Obama	Democratic	51.1	2012	win
3	Romney	Republican	47.2	2012	loss
4	Clinton	Democratic	48.2	2016	loss
5	Trump	Republican	46.1	2016	win

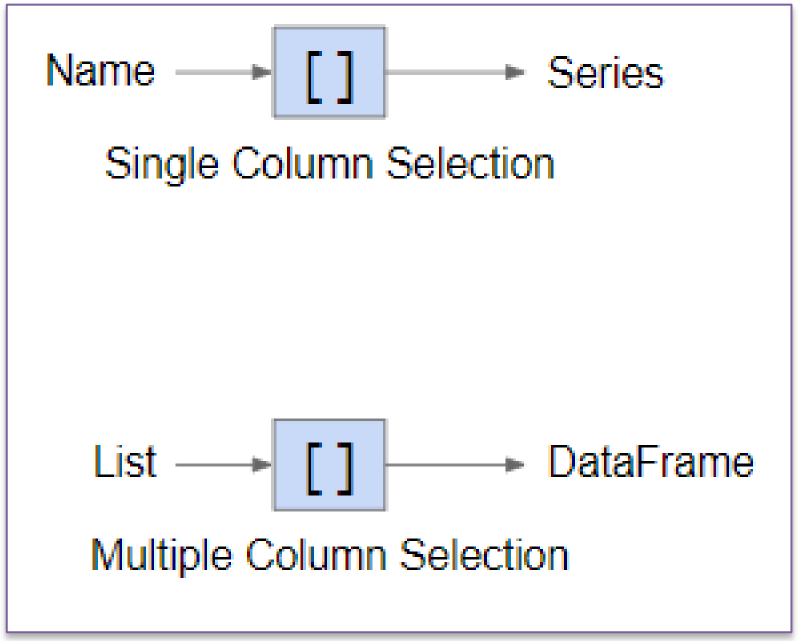
Tables



	Candidate	Party	%	Year	Result
0	Obama	Democratic	52.9	2008	win
1	McCain	Republican	45.7	2008	loss
2	Obama	Democratic	51.1	2012	win
3	Romney	Republican	47.2	2012	loss
4	Clinton	Democratic	48.2	2016	loss
5	Trump	Republican	46.1	2016	win

Access Columns

Use one pair of brackets to access a column from a table as a Series

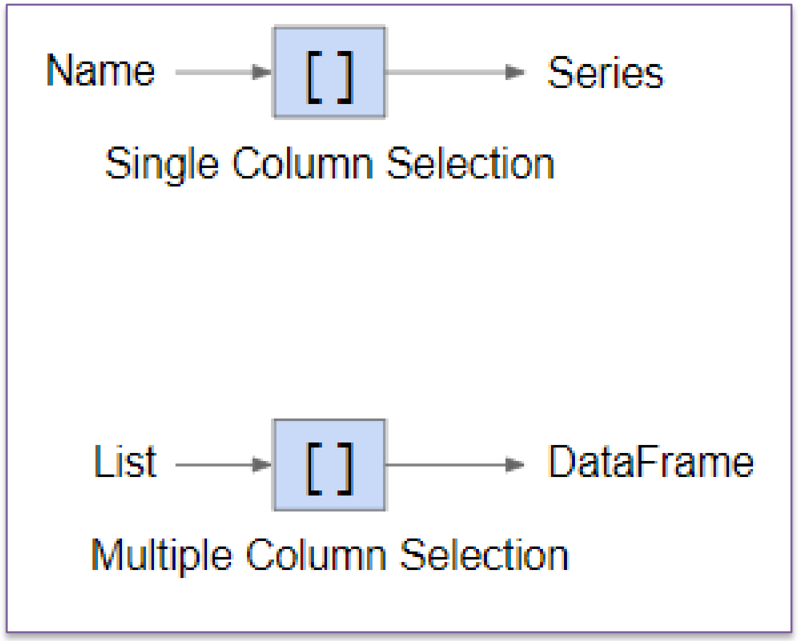


```
elections["Candidate"]
```

```
0          Andrew Jackson
1       John Quincy Adams
2          Andrew Jackson
3       John Quincy Adams
4          Andrew Jackson
              ...
173          Donald Trump
174          Evan McMullin
175          Gary Johnson
176       Hillary Clinton
177            Jill Stein
Name: Candidate, Length: 178, dtype: object
```

Access Columns

Use two pairs of nested brackets to access two or more columns from a table



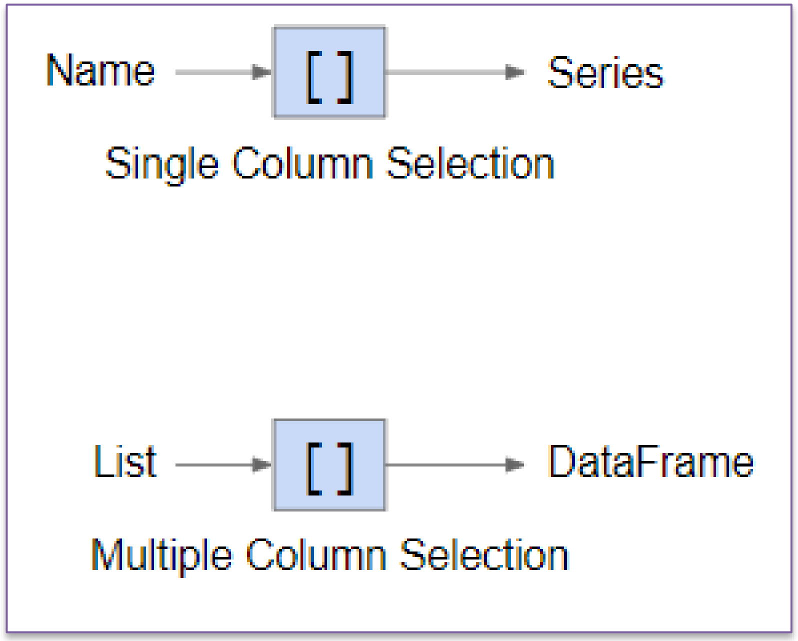
```
elections[["Candidate", "Party"]]
```

	Candidate	Party
0	Andrew Jackson	Democratic-Republican
1	John Quincy Adams	Democratic-Republican
2	Andrew Jackson	Democratic
3	John Quincy Adams	National Republican
4	Andrew Jackson	Democratic
...	...	...
173	Donald Trump	Republican
174	Evan McMullin	Independent
175	Gary Johnson	Libertarian
176	Hillary Clinton	Democratic
177	Jill Stein	Green

178 rows × 2 columns

Access Columns

Use two pairs of nested brackets to access a column from a table as a DataFrame



```
elections[["Candidate"]]
```

	Candidate
0	Andrew Jackson
1	John Quincy Adams
2	Andrew Jackson
3	John Quincy Adams
4	Andrew Jackson
...	...
173	Donald Trump
174	Evan McMullin
175	Gary Johnson
176	Hillary Clinton
177	Jill Stein

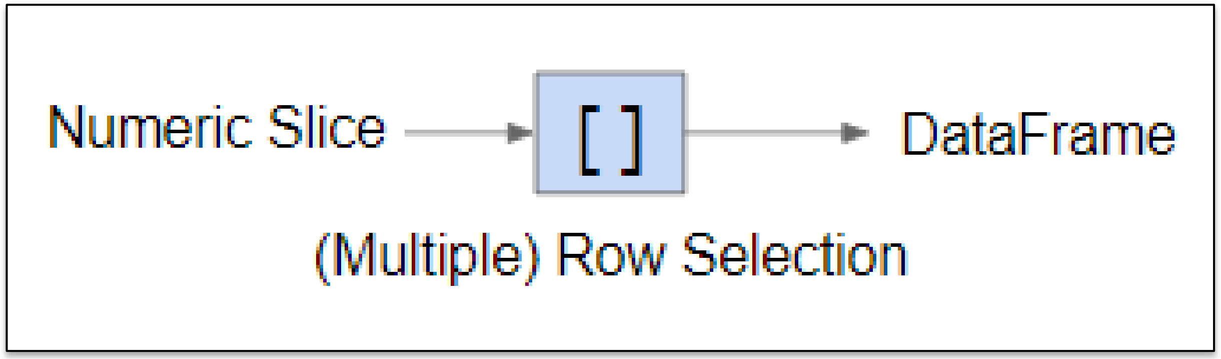
178 rows × 1 columns

Access Rows

```
elections[0:3]
```

	Candidate	Party	%	Result
Year				
1980	Reagan	Republican	50.7	win
1980	Carter	Democratic	41.0	loss
1980	Anderson	Independent	6.6	loss

Note that you must indicate adjacent rows with a numeric range such as 0:3 for 0,1,2



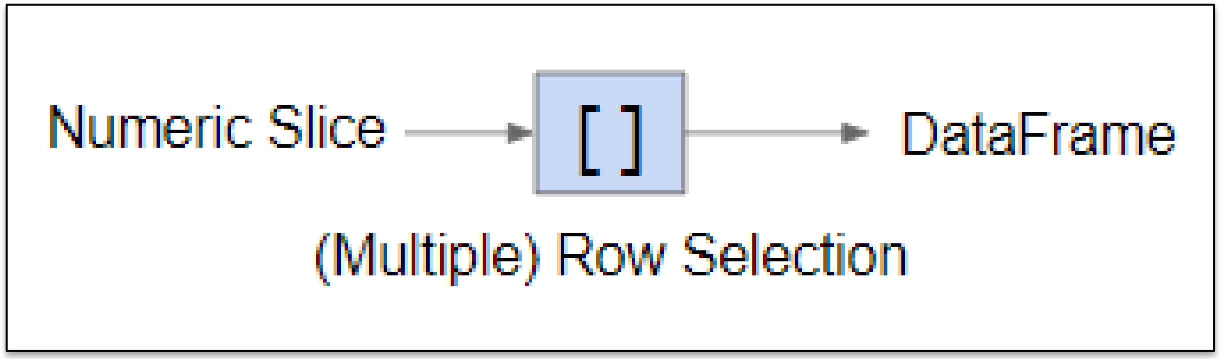
NYU TANDON SCHOOL OF ENGINEERING

Access Rows

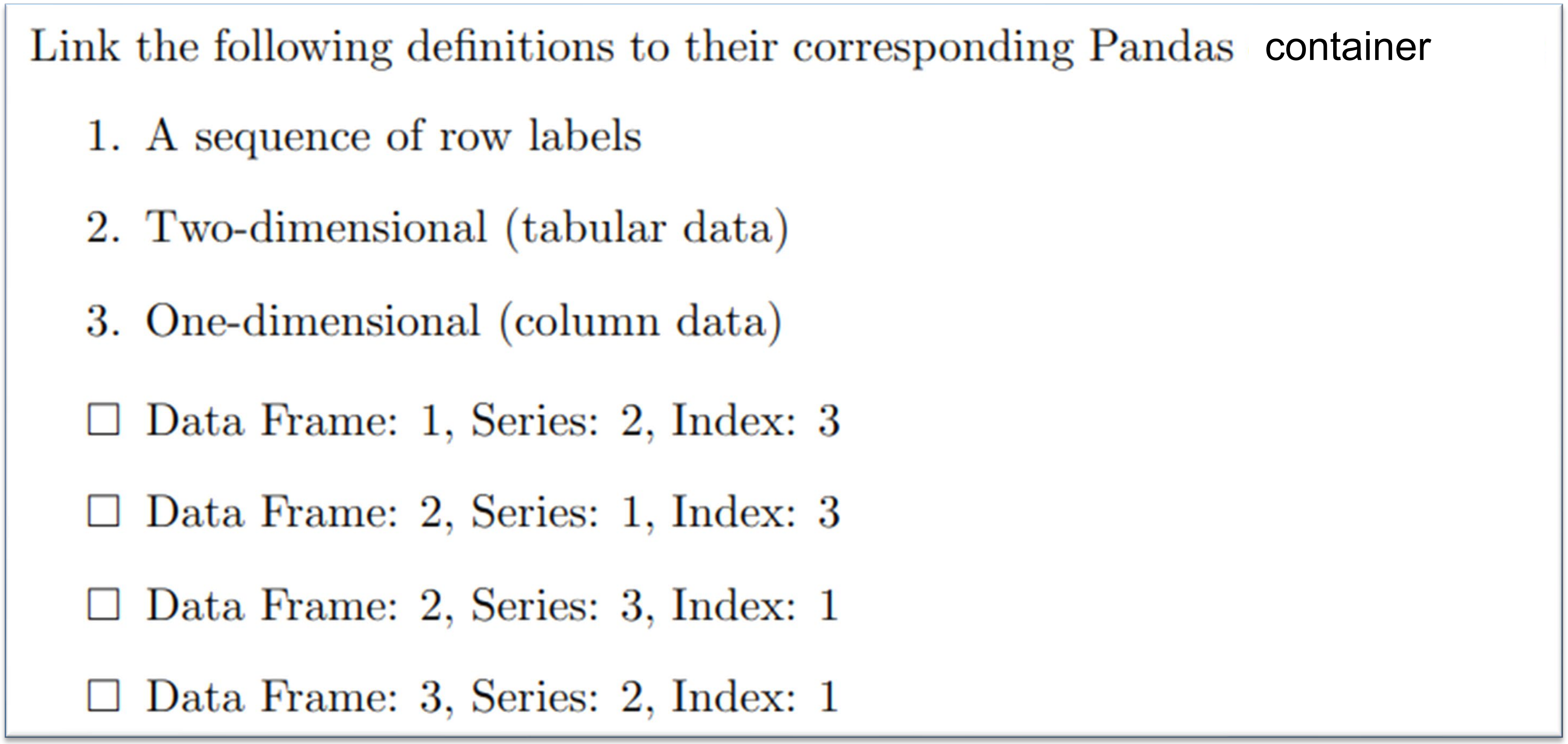
```
elections[0]
```

	Candidate	Party	%	Result
Year				
1980	Reagan	Republican	50.7	win
1980	Carter	Democratic	41.0	loss
1980	Anderson	Independent	6.6	loss

Note that you must indicate adjacent rows with a numeric range such as 0:3 for 0,1,2



Poll



Link the following definitions to their corresponding Pandas container

1. A sequence of row labels
2. Two-dimensional (tabular data)
3. One-dimensional (column data)

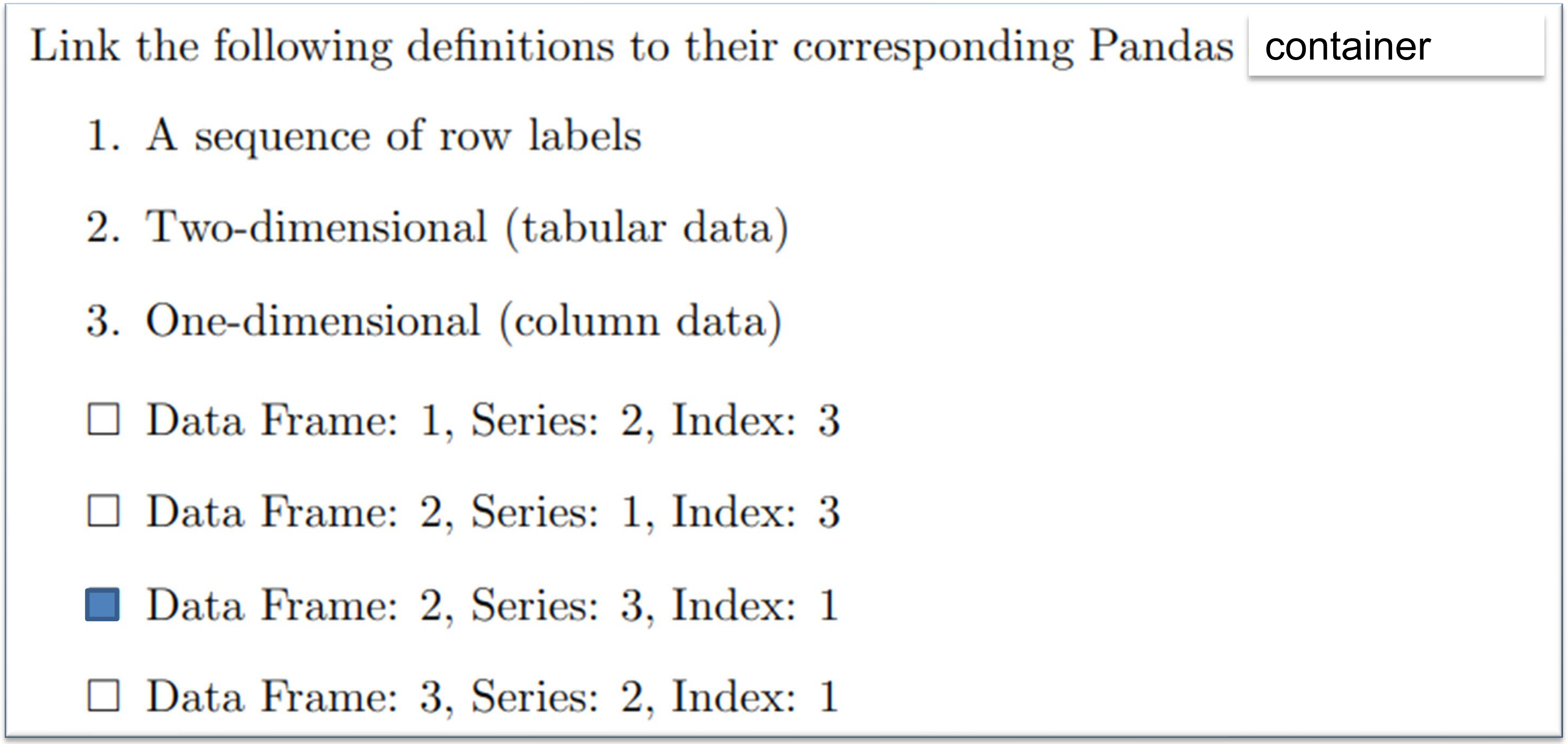
☐ Data Frame: 1, Series: 2, Index: 3

☐ Data Frame: 2, Series: 1, Index: 3

☐ Data Frame: 2, Series: 3, Index: 1

☐ Data Frame: 3, Series: 2, Index: 1

Poll



Link the following definitions to their corresponding Pandas container

1. A sequence of row labels
2. Two-dimensional (tabular data)
3. One-dimensional (column data)

☐ Data Frame: 1, Series: 2, Index: 3

☐ Data Frame: 2, Series: 1, Index: 3

[✓ Data Frame: 2, Series: 3, Index: 1]

☐ Data Frame: 3, Series: 2, Index: 1

Access Rows and Columns

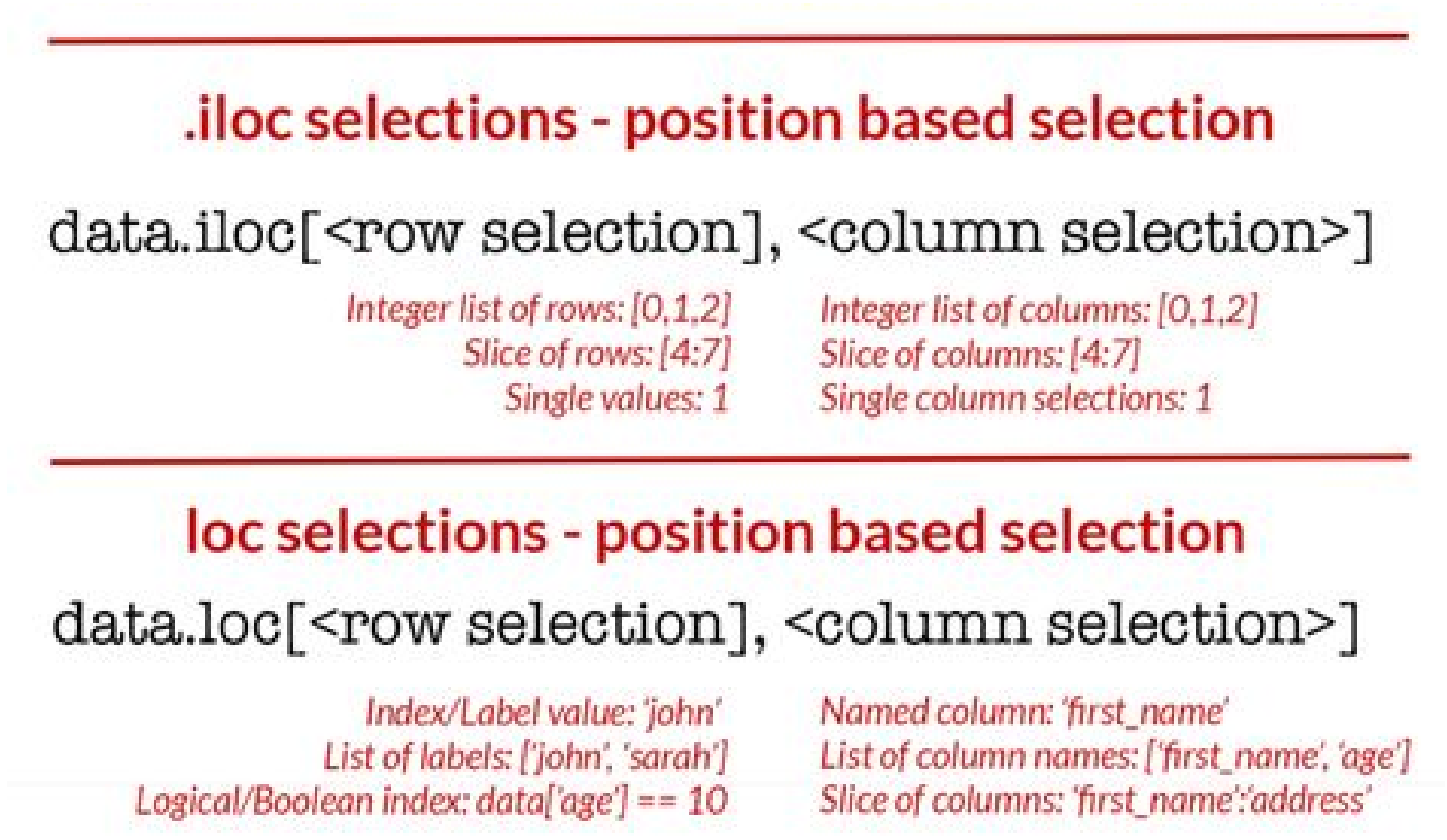
```
elections.loc[[0, 1, 2, 3, 4], ['Candidate','Party', 'Year']]
```

	Candidate	Party	Year
0	Reagan	Republican	1980
1	Carter	Democratic	1980
2	Anderson	Independent	1980
3	Reagan	Republican	1984
4	Mondale	Democratic	1984

Access Rows and Columns

```
elections.iloc[0:3, 0:3]
```

	Candidate	Party	%
0	Reagan	Republican	50.7
1	Carter	Democratic	41.0
2	Anderson	Independent	6.6



.iloc selections - position based selection

```
data.iloc[<row selection>, <column selection>]
```

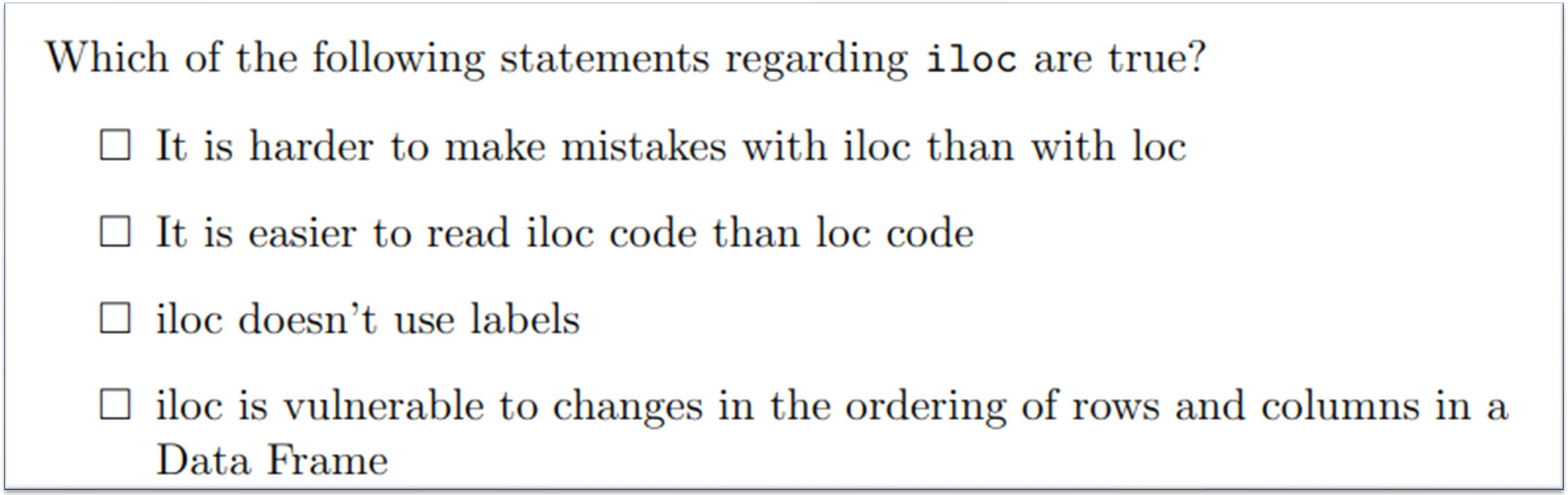
Integer list of rows: [0,1,2]	Integer list of columns: [0,1,2]
Slice of rows: [4:7]	Slice of columns: [4:7]
Single values: 1	Single column selections: 1

loc selections - position based selection

```
data.loc[<row selection>, <column selection>]
```

Index/Label value: 'john'	Named column: 'first_name'
List of labels: ['john', 'sarah']	List of column names: ['first_name', 'age']
Logical/Boolean index: data['age'] == 10	Slice of columns: 'first_name':'address'

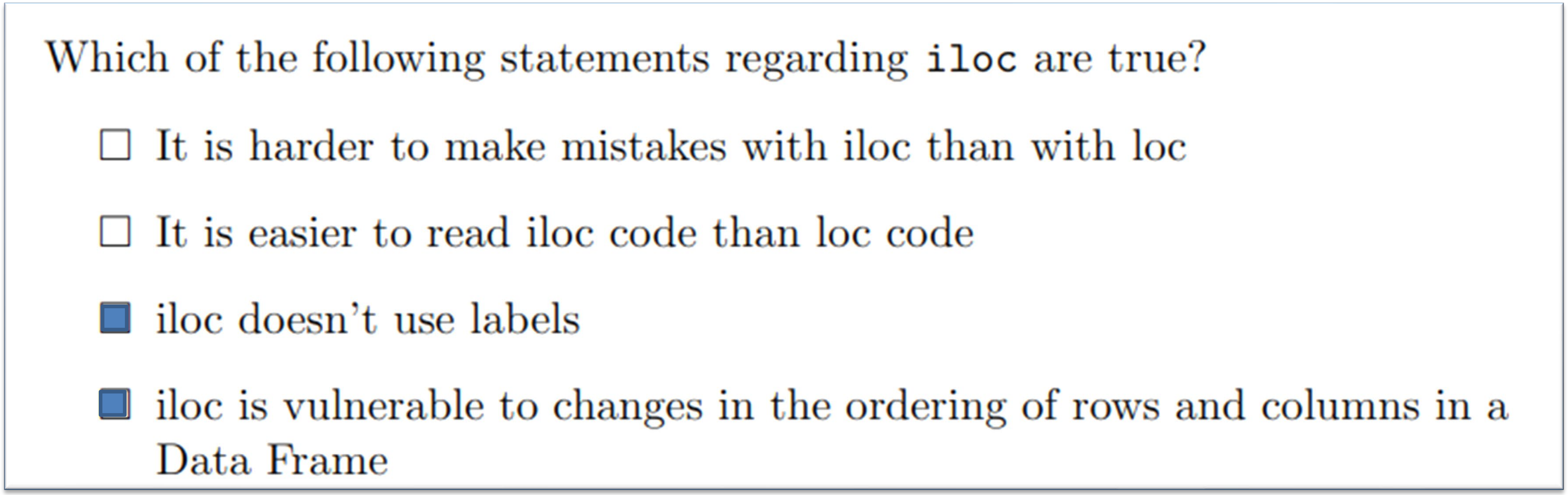
Breakout Room



Which of the following statements regarding `iloc` are true?

- ☐ It is harder to make mistakes with iloc than with loc
- ☐ It is easier to read iloc code than loc code
- ☐ iloc doesn't use labels
- ☐ iloc is vulnerable to changes in the ordering of rows and columns in a Data Frame

Breakout Room



Which of the following statements regarding `iloc` are true?

- ☐ It is harder to make mistakes with iloc than with loc
- ☐ It is easier to read iloc code than loc code
- [✓ iloc doesn't use labels]
- [✓ iloc is vulnerable to changes in the ordering of rows and columns in a Data Frame]

Filtering

```
elections[[False, False, False, False, False,
          False, False, True, False, False,
          True, False, False, False, True,
          False, False, False, False, False,
          False, False, True]]
```

	Candidate	Party	%	Year	Result
7	Clinton	Democratic	43.0	1992	win
10	Clinton	Democratic	49.2	1996	win
14	Bush	Republican	47.9	2000	win
22	Trump	Republican	46.1	2016	win

Filtering

```
elections[elections['Party'] == 'Independent']
```

	Candidate	Party	%	Year	Result
7	Clinton	Democratic	43.0	1992	win
10	Clinton	Democratic	49.2	1996	win
14	Bush	Republican	47.9	2000	win
22	Trump	Republican	46.1	2016	win

Filtering

You must use & for “and”, | for “or”, ~ for “not”

```
elections[(elections['Result'] == 'win')
          & (elections['%'] < 50)]
```

	Candidate	Party	%	Year	Result
7	Clinton	Democratic	43.0	1992	win
10	Clinton	Democratic	49.2	1996	win
14	Bush	Republican	47.9	2000	win
22	Trump	Republican	46.1	2016	win

Filtering

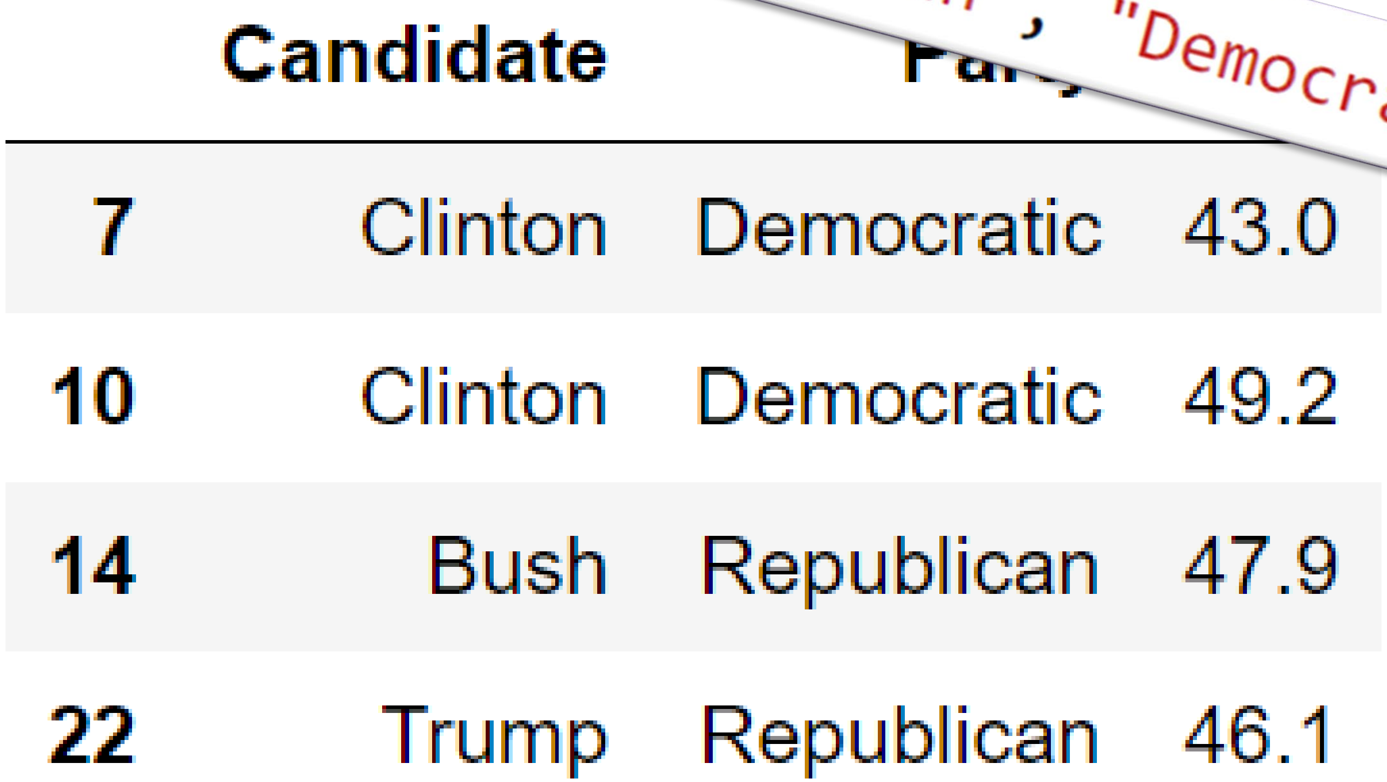
```
elections.loc[(elections['Result'] == 'win') & (elections['%'] < 50), 'Candidate':'%']
```

	Candidate	Party	%
7	Clinton	Democratic	43.0
10	Clinton	Democratic	49.2
14	Bush	Republican	47.9
22	Trump	Republican	46.1

Filtering

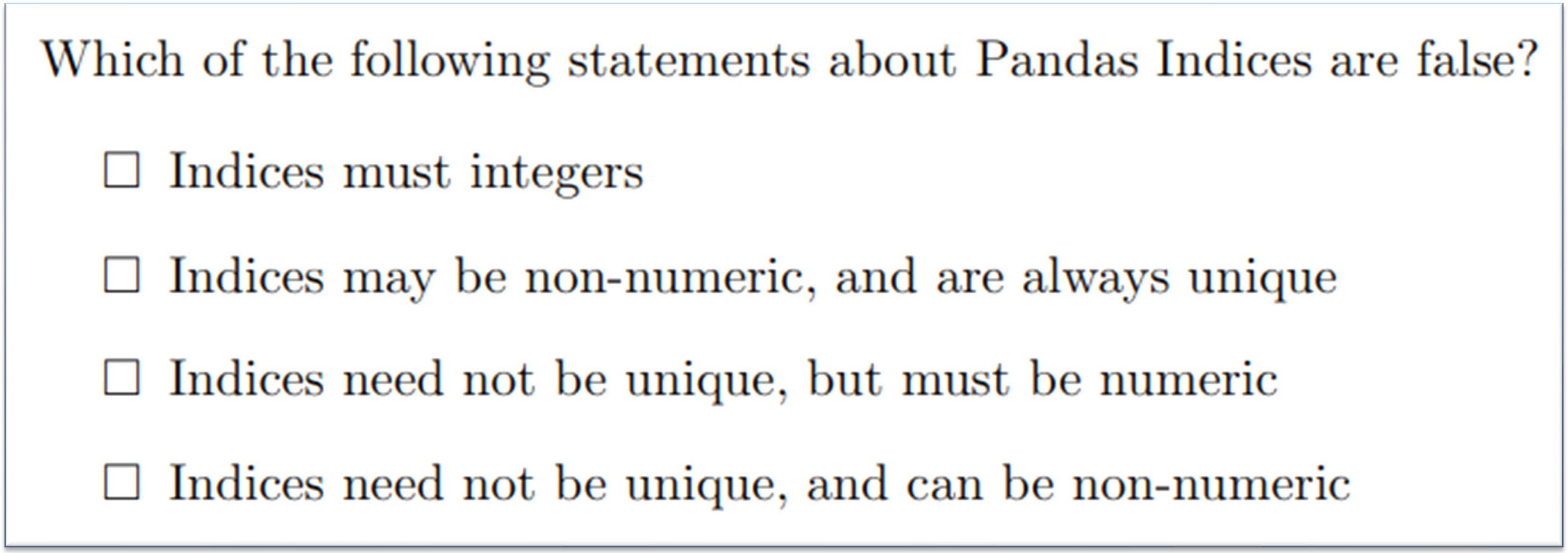
```
df[df["Party"].isin(["Republican", "Democratic"])]
```

```
elections.loc[(electio              n') & (elections['%'] < 50), 'Candidate':'%']
```



	Candidate	Par	
7	Clinton	Democratic	43.0
10	Clinton	Democratic	49.2
14	Bush	Republican	47.9
22	Trump	Republican	46.1

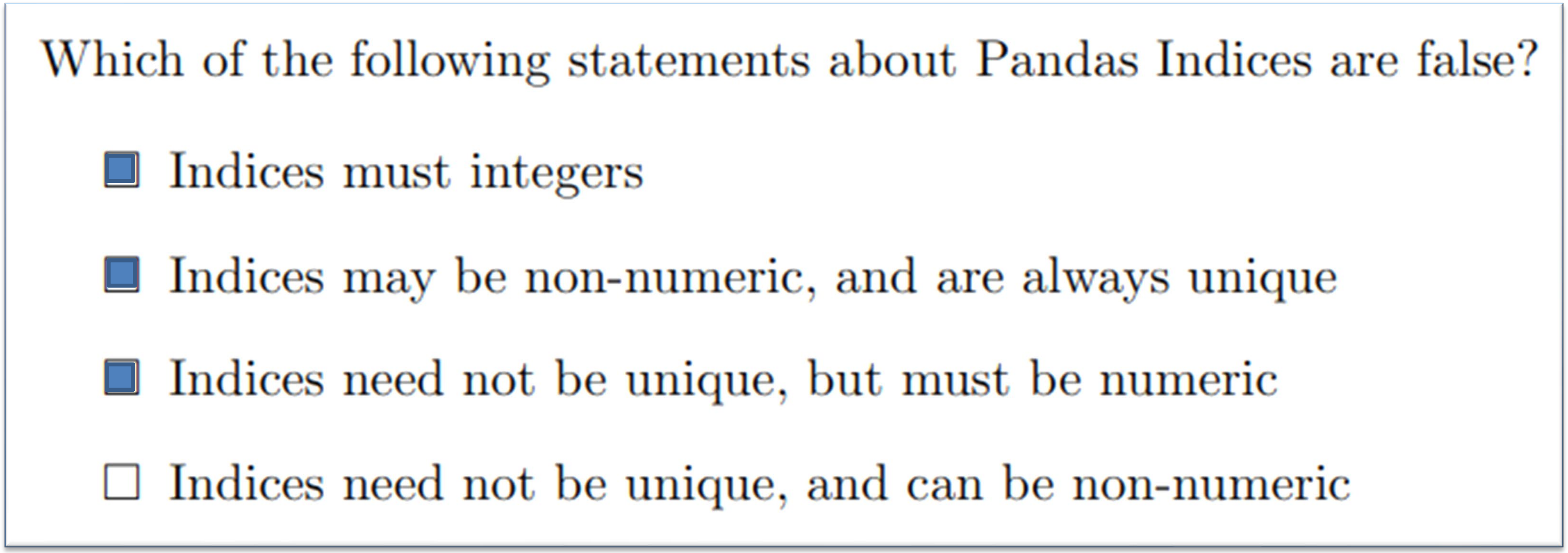
Poll



Which of the following statements about Pandas Indices are false?

- ☐ Indices must integers
- ☐ Indices may be non-numeric, and are always unique
- ☐ Indices need not be unique, but must be numeric
- ☐ Indices need not be unique, and can be non-numeric

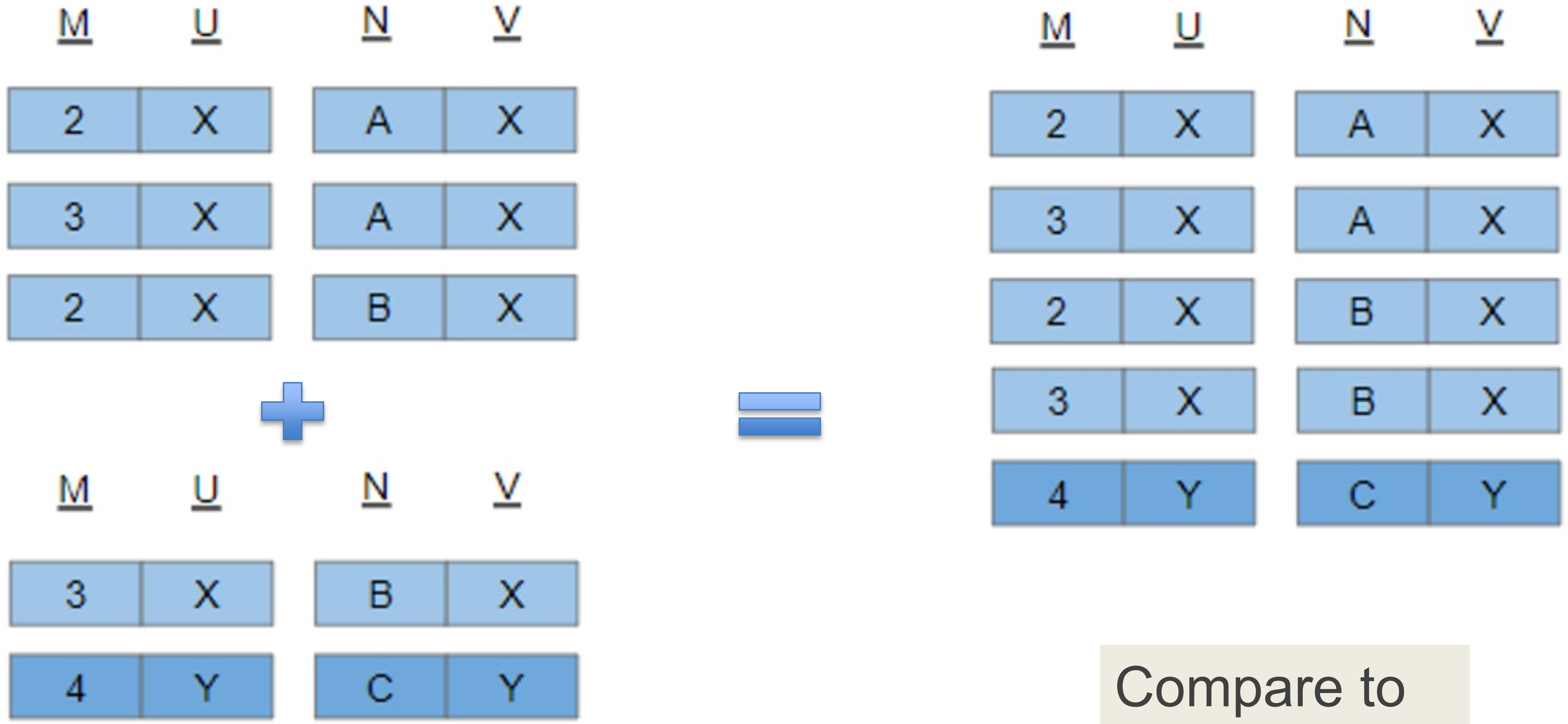
Poll



Which of the following statements about Pandas Indices are false?

- [✓ Indices must integers]
- [✓ Indices may be non-numeric, and are always unique]
- [✓ Indices need not be unique, but must be numeric]
- ☐ Indices need not be unique, and can be non-numeric

Concatenate



M	U	N	V
2	X	A	X
3	X	A	X
2	X	B	X

+

M	U	N	V
3	X	B	X
4	Y	C	Y

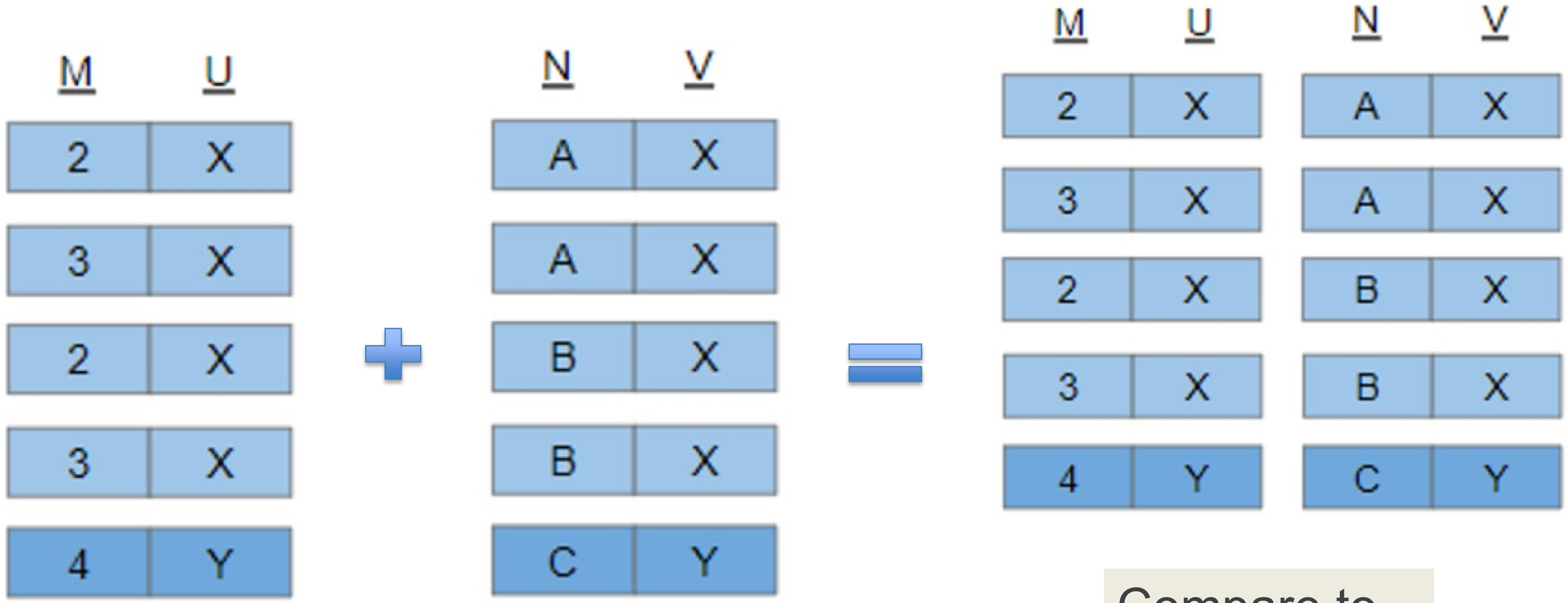
=

M	U	N	V
2	X	A	X
3	X	A	X
2	X	B	X
3	X	B	X
4	Y	C	Y

Compare to numpy.vstack

NYU TANDON SCHOOL OF ENGINEERING

Concatenate



M	U
2	X
3	X
2	X
3	X
4	Y

+

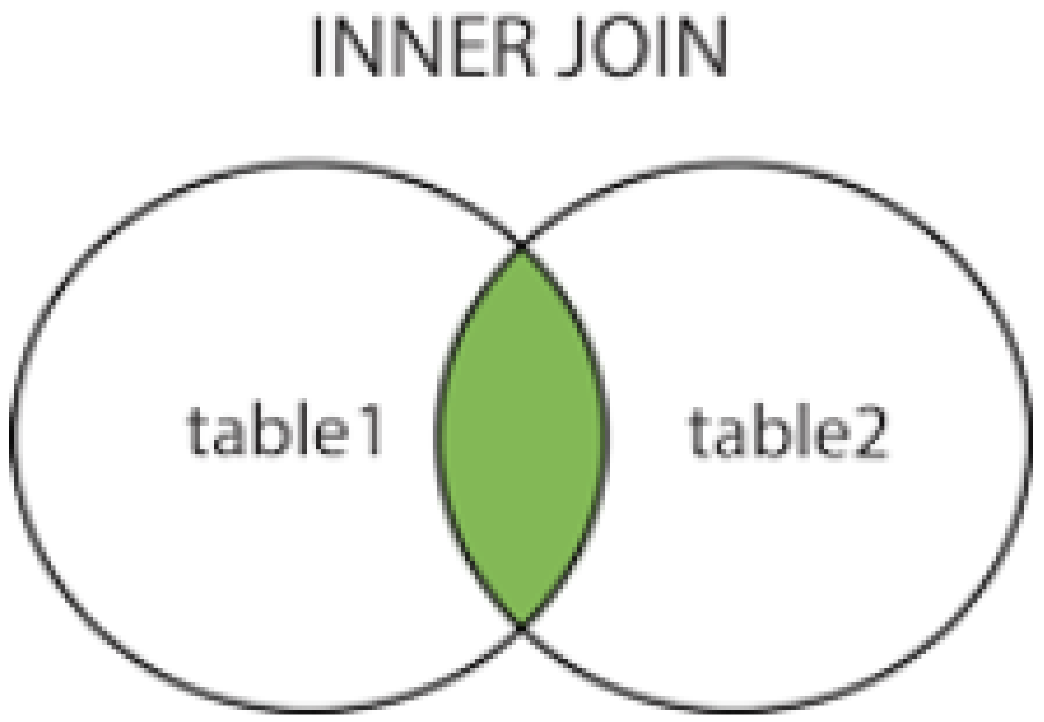
N	V
A	X
A	X
B	X
B	X
C	Y

=

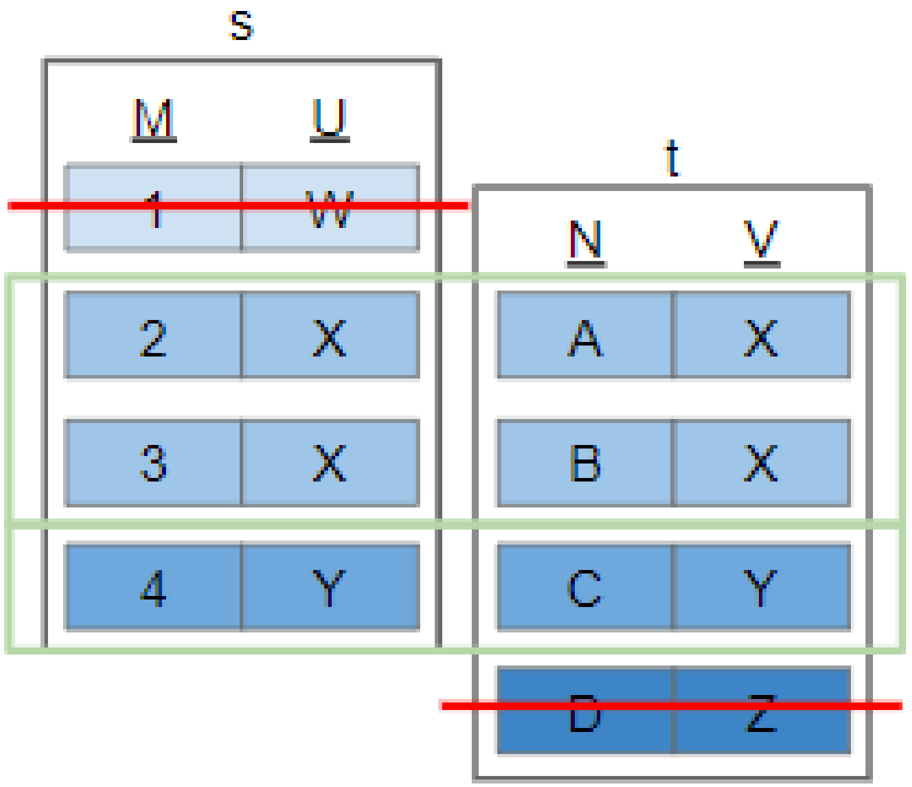
M	U	N	V
2	X	A	X
3	X	A	X
2	X	B	X
3	X	B	X
4	Y	C	Y

Compare to numpy.hstack

Joining Datasets

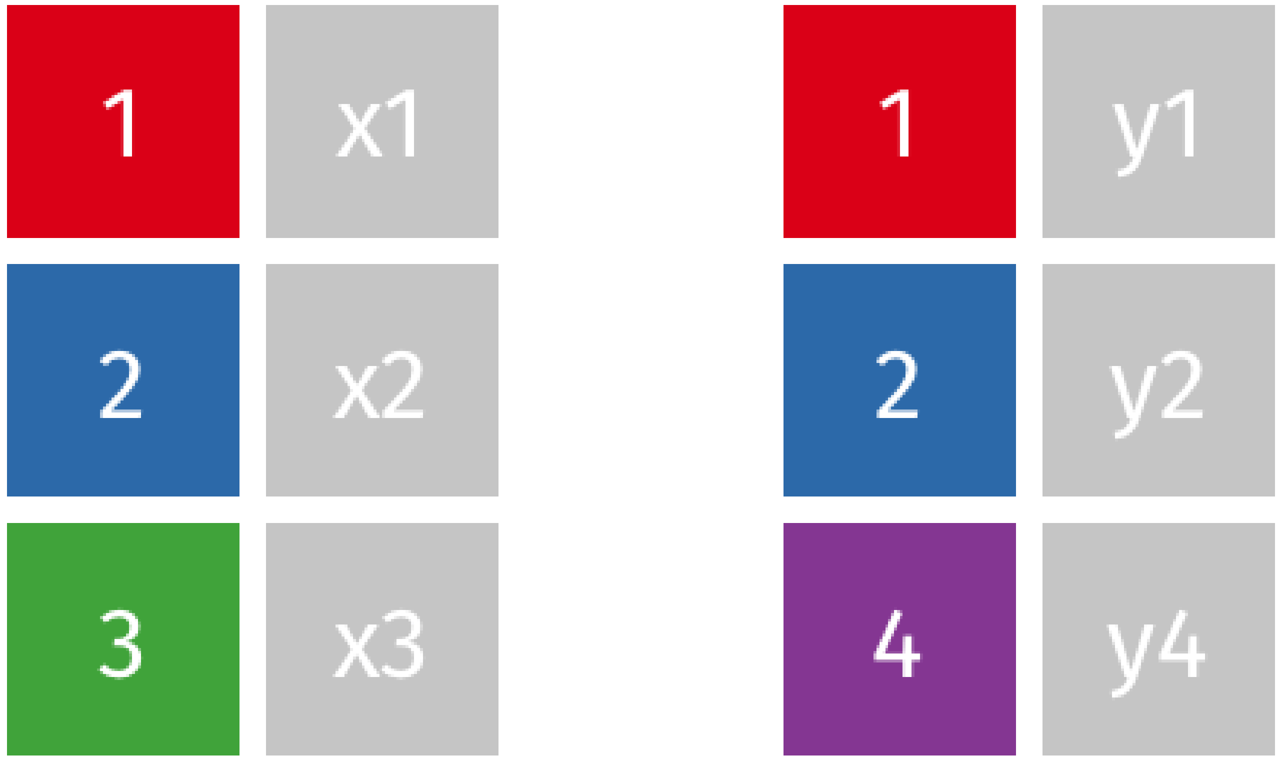


Inner Join

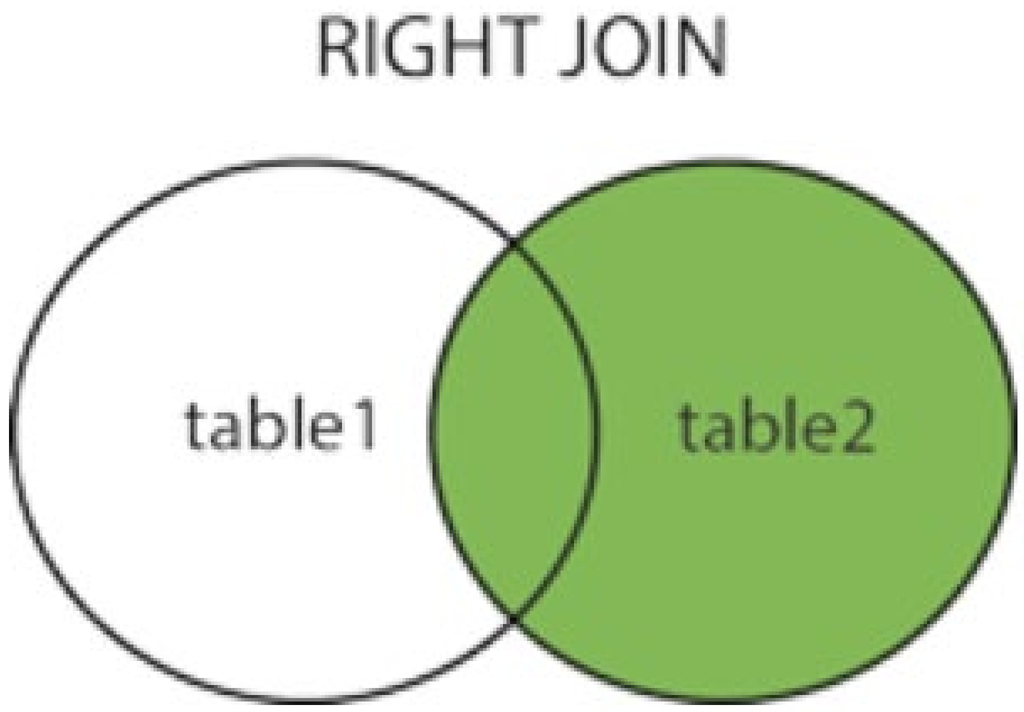


M	U	N	V
2	X	A	X
3	X	A	X
2	X	B	X
3	X	B	X
4	Y	C	Y

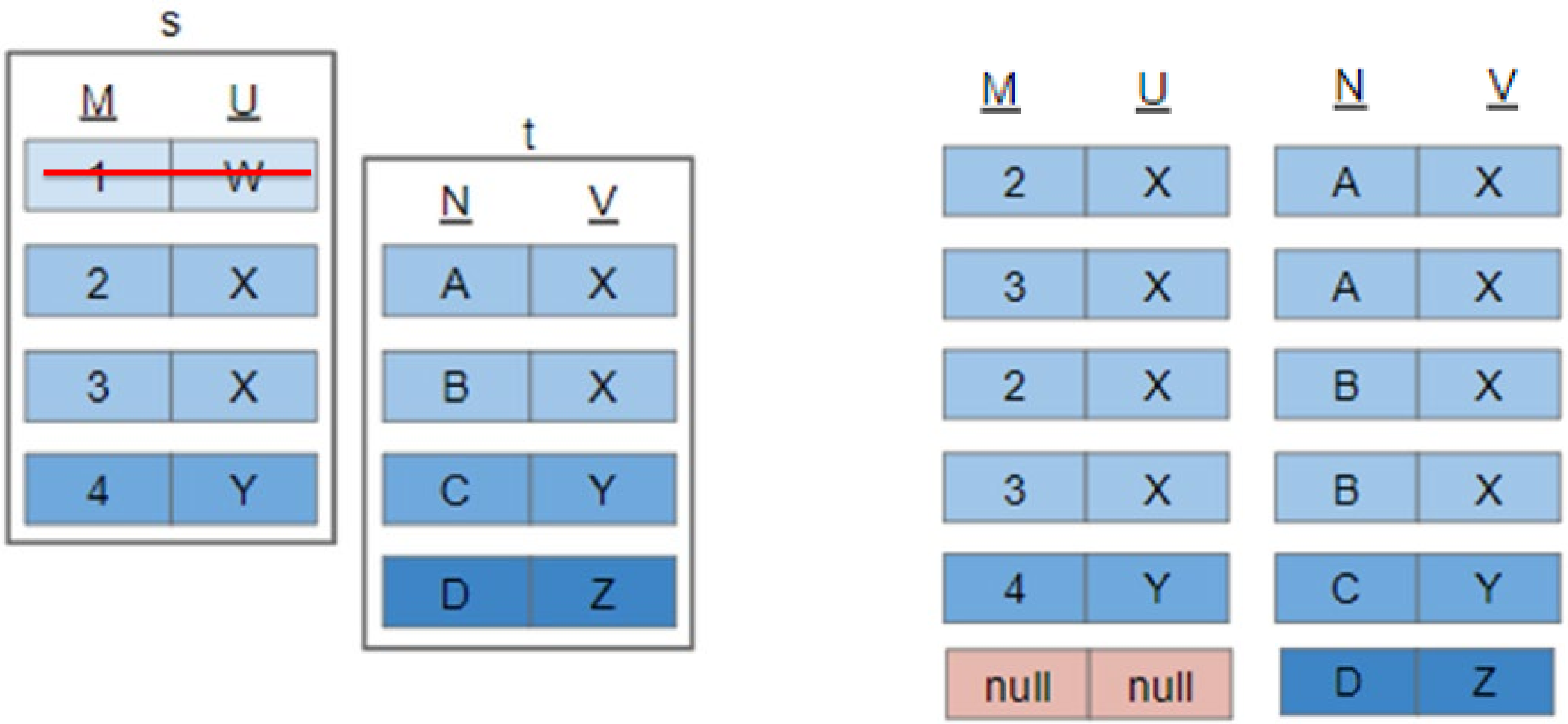
Inner Join



Joining Datasets



Right Join



s

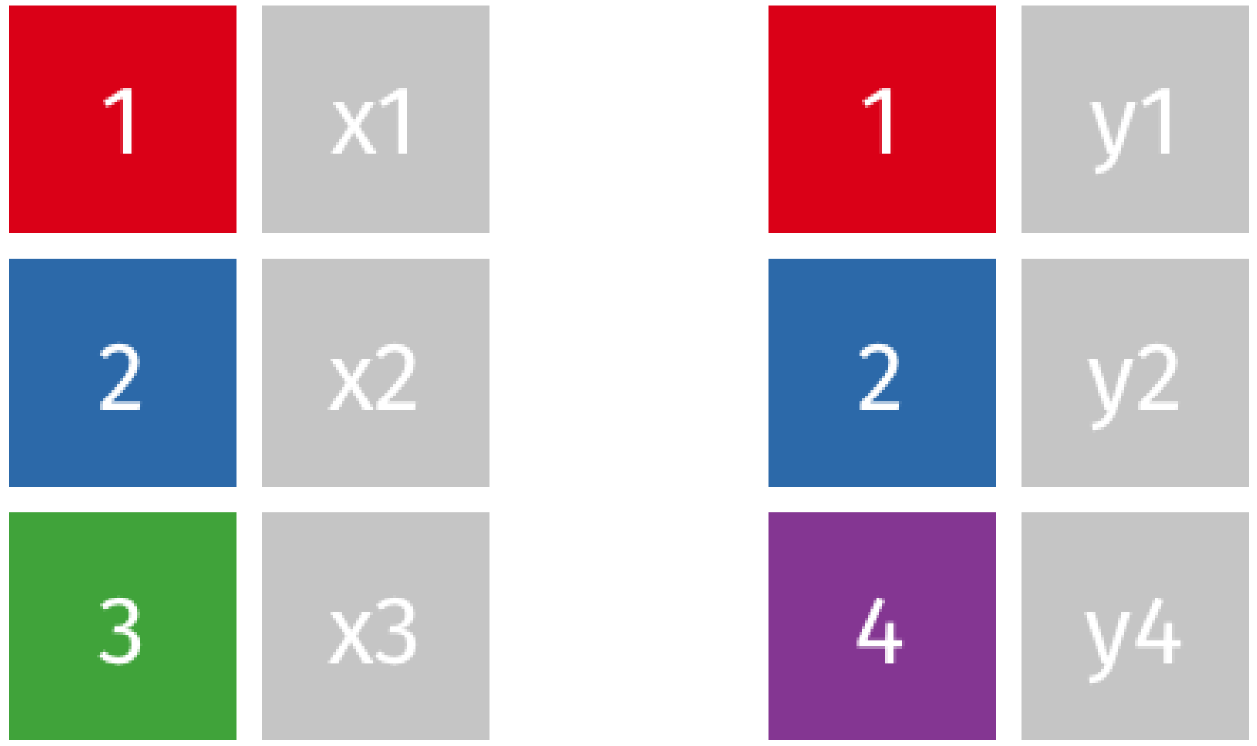
M	U
~~1~~	~~W~~
2	X
3	X
4	Y

t

N	V
A	X
B	X
C	Y
D	Z

M	U	N	V
2	X	A	X
3	X	A	X
2	X	B	X
3	X	B	X
4	Y	C	Y
null	null	D	Z

Right Join



Tables

- Accessing Entries
- Filtering Records
- Joining Datasets

NYU | TANDON SCHOOL OF ENGINEERING

References

- McKinney, Python for Data Analysis

(Chapter 11.1-11.4 + 7.1-7.2 + 8.2)

Questions

- Describe the learning objectives.
- Summarize the relevant take-aways.
- Ask about unclear information.