CODEUP DATA SCIENCE - 2021

PANDAS

Germain, meet pandas; pandas, meet Germain!

FAITH KANE

LESSON OBJECTIVES

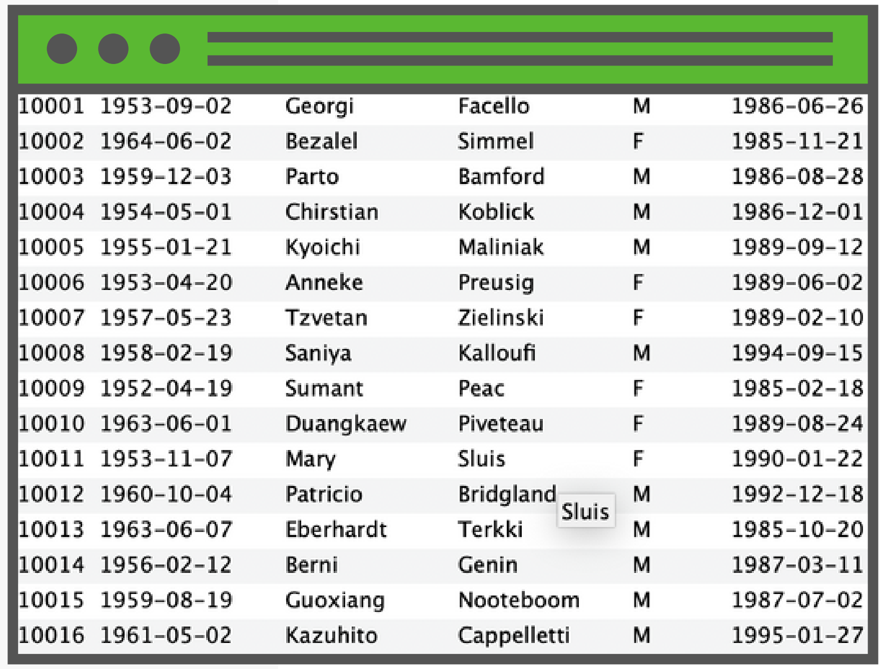
BY THE END OF THIS LESSON, YOU WILL UNDERSTAND...

- what the pandas library is.
- what a pandas DataFrame is.
- how to select/create a pandas Series.
- the components of a pandas Series and how to access its methods and attributes.
- the joy of vectorized operations.
- how to plot a Series really quickly.

PANDAS OVERVIEW...

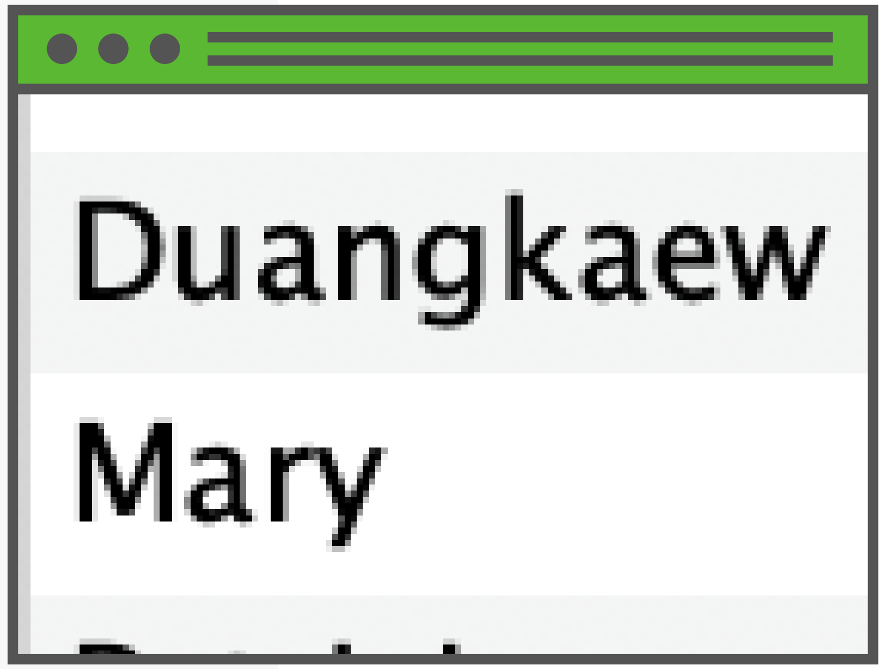
- open source python library
- built on NumPy and Matplotlib
- quickly acquire data from various sources (csv and json files, databases, etc.)
- structured data stored in DataFrames (tables)
- Series (columns) handle any data type
    - homogenous data type in each Series
- LOTS of vectorized functions
- LOTS of built-in attributes and methods to access properties and behaviors quickly

WHAT YOU ALREADY KNOW...



10001	1953-09-02	Georgi	Facello	M	1986-06-26
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
10003	1959-12-03	Parto	Bamford	M	1986-08-28
10004	1954-05-01	Chirstian	Koblick	M	1986-12-01
10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
10006	1953-04-20	Anneke	Preusig	F	1989-06-02
10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10
10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15
10009	1952-04-19	Sumant	Peac	F	1985-02-18
10010	1963-06-01	Duangkaew	Piveteau	F	1989-08-24
10011	1953-11-07	Mary	Sluis	F	1990-01-22
10012	1960-10-04	Patricio	Bridgland	M	1992-12-18
10013	1963-06-07	Eberhardt	Terkki	M	1985-10-20
10014	1956-02-12	Berni	Genin	M	1987-03-11
10015	1959-08-19	Guoxiang	Nooteboom	M	1987-07-02
10016	1961-05-02	Kazuhito	Cappelletti	M	1995-01-27

```
SELECT *
FROM EMPLOYEES
```

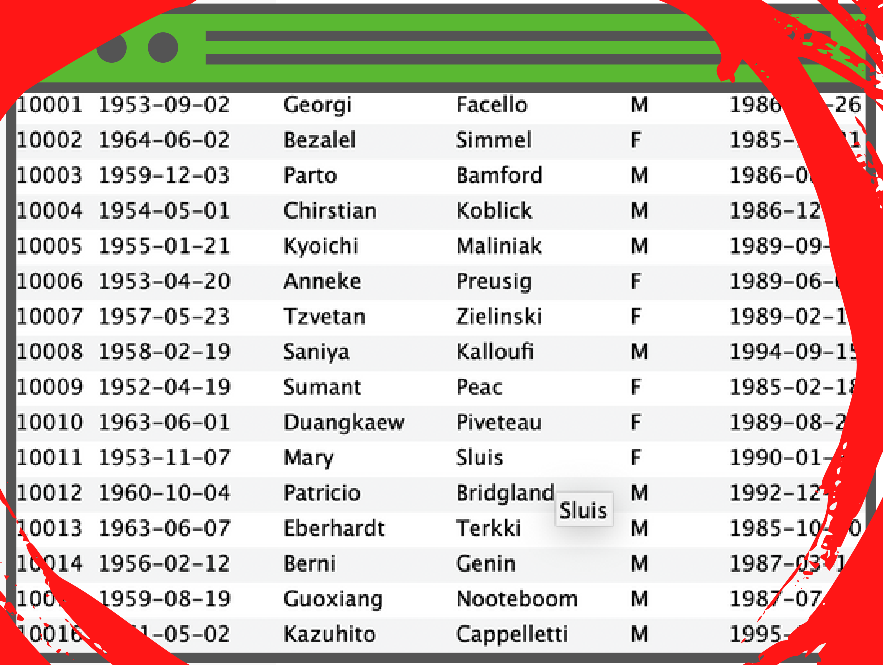


Duangkaew

Mary

```
SELECT FIRST_NAME
FROM EMPLOYEES
```

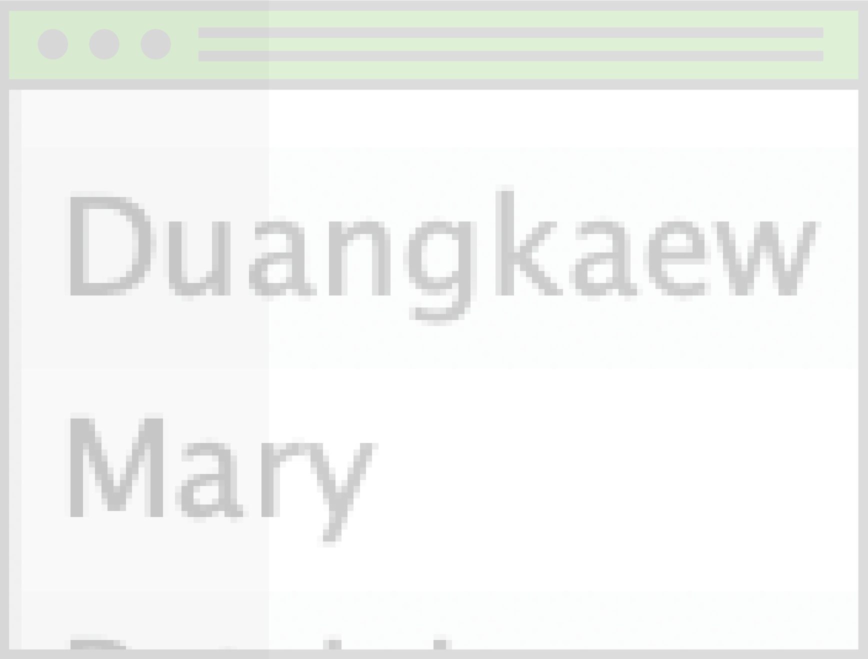
THINK OF THIS AS A DATAFRAME



10001	1953-09-02	Georgi	Facello	M	1986[illegible]-26
10002	1964-06-02	Bezalel	Simmel	F	1985-[illegible]1
10003	1959-12-03	Parto	Bamford	M	1986-0[illegible]
10004	1954-05-01	Chirstian	Koblick	M	1986-12[illegible]
10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-[illegible]
10006	1953-04-20	Anneke	Preusig	F	1989-06-[illegible]
10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-1[illegible]
10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15
10009	1952-04-19	Sumant	Peac	F	1985-02-18
10010	1963-06-01	Duangkaew	Piveteau	F	1989-08-2[illegible]
10011	1953-11-07	Mary	Sluis	F	1990-01-[illegible]
10012	1960-10-04	Patricio	Bridgland	M	1992-12[illegible]
10013	1963-06-07	Eberhardt	Terkki	M	1985-10[illegible]0
10014	1956-02-12	Berni	Genin	M	1987-03[illegible]1
100[illegible]	1959-08-19	Guoxiang	Nooteboom	M	1987-07[illegible]
10016	[illegible]1-05-02	Kazuhito	Cappelletti	M	1995-[illegible]

```
SELECT *
FROM EMPLOYEES
```

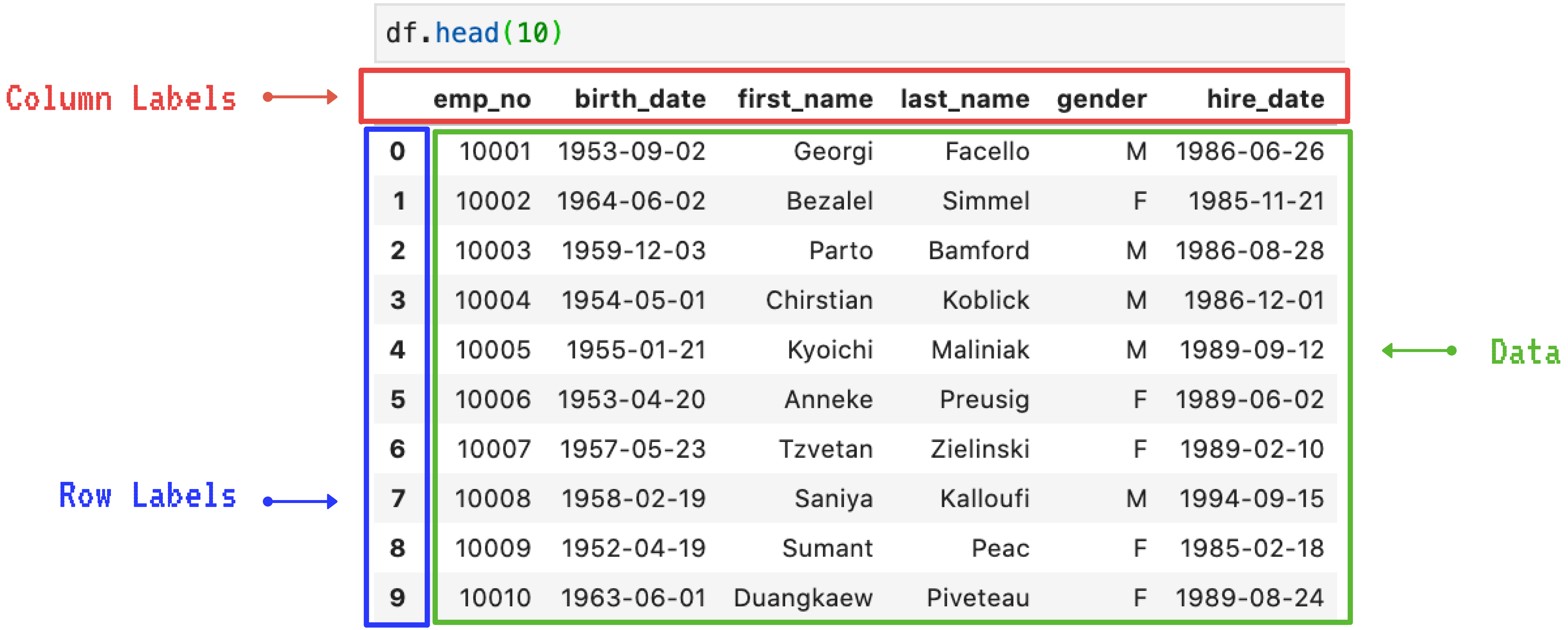
WHAT YOU ALREADY KNOW...



Duangkaew

Mary

```
SELECT NAME
FROM EMPLOYEES
```



```
df.head(10)
```

Column Labels

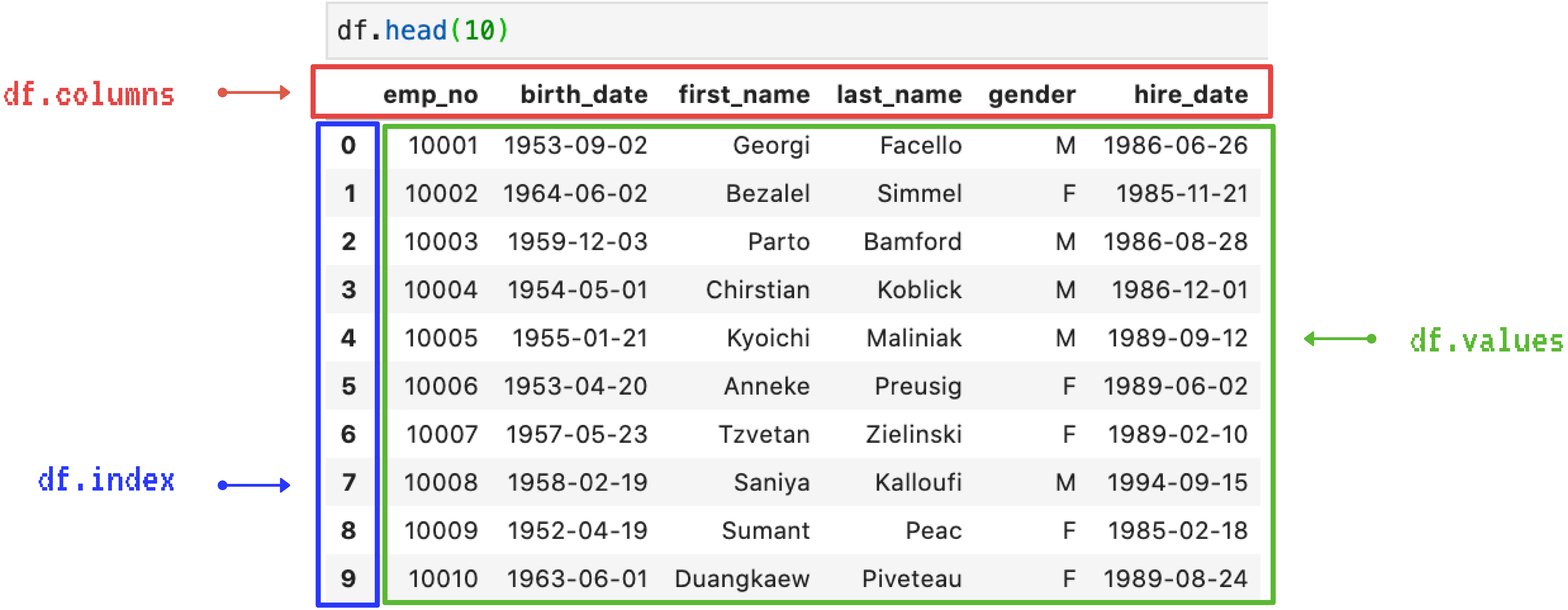
	emp_no	birth_date	first_name	last_name	gender	hire_date
0	10001	1953-09-02	Georgi	Facello	M	1986-06-26
1	10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
2	10003	1959-12-03	Parto	Bamford	M	1986-08-28
3	10004	1954-05-01	Chirstian	Koblick	M	1986-12-01
4	10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
5	10006	1953-04-20	Anneke	Preusig	F	1989-06-02
6	10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10
7	10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15
8	10009	1952-04-19	Sumant	Peac	F	1985-02-18
9	10010	1963-06-01	Duangkaew	Piveteau	F	1989-08-24

Row Labels

Data

THE PANDAS DATAFRAME

A LABELED, TWO-DIMENSIONAL ARRAY



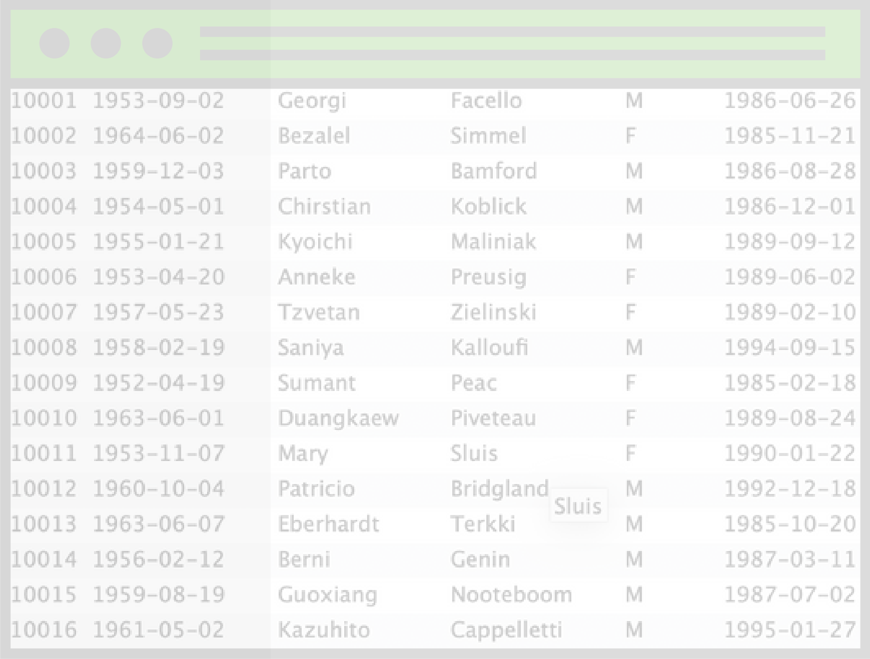
```
df.head(10)
```

	emp_no	birth_date	first_name	last_name	gender	hire_date
0	10001	1953-09-02	Georgi	Facello	M	1986-06-26
1	10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
2	10003	1959-12-03	Parto	Bamford	M	1986-08-28
3	10004	1954-05-01	Chirstian	Koblick	M	1986-12-01
4	10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
5	10006	1953-04-20	Anneke	Preusig	F	1989-06-02
6	10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10
7	10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15
8	10009	1952-04-19	Sumant	Peac	F	1985-02-18
9	10010	1963-06-01	Duangkaew	Piveteau	F	1989-08-24

DATAFRAME COMPONENTS

PANDAS DATAFRAME ATTRIBUTES

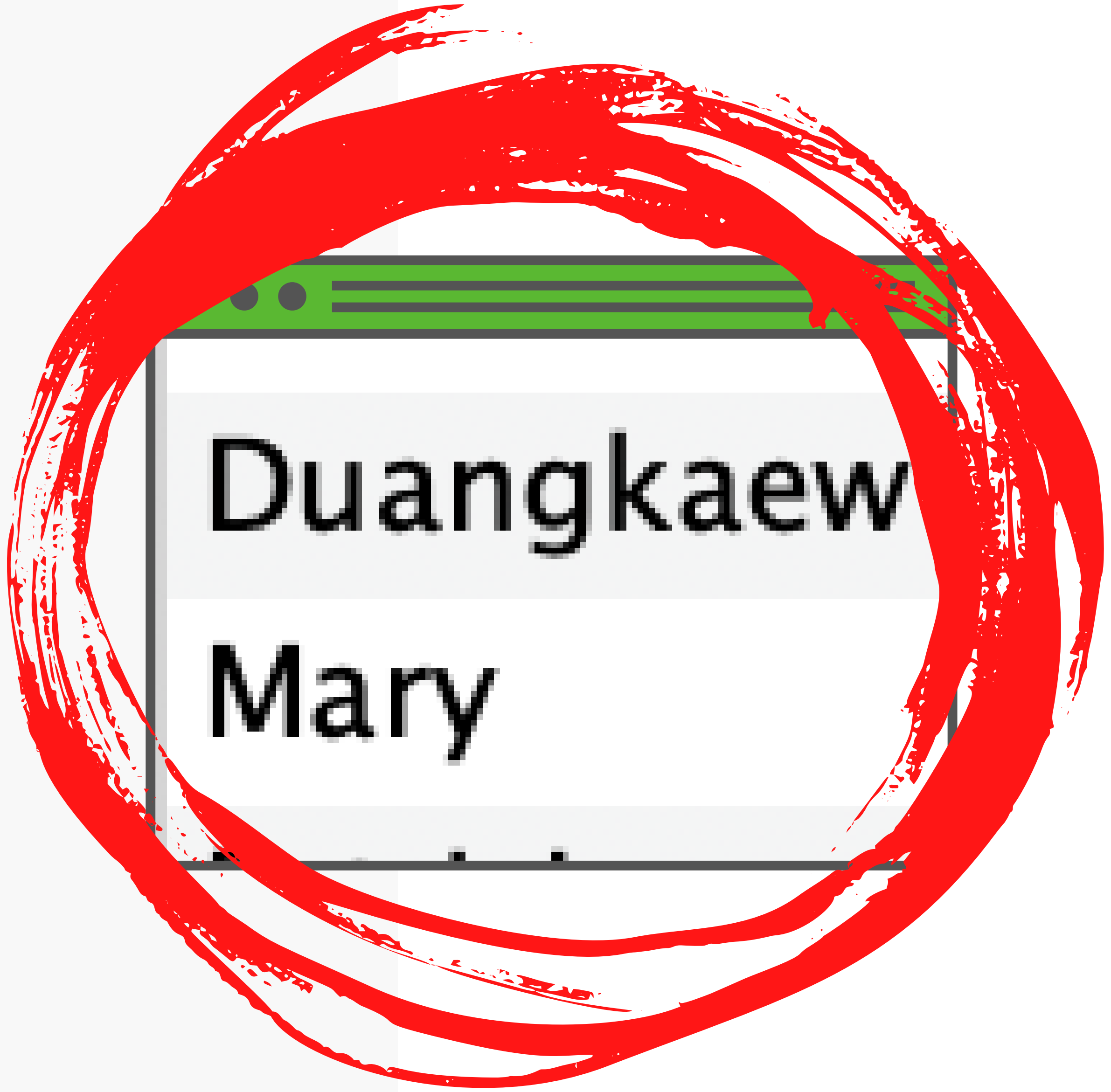
What you already know...



10001	1953-09-02	Georgi	Facello	M	1986-06-26
10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
10003	1959-12-03	Parto	Bamford	M	1986-08-28
10004	1954-05-01	Chirstian	Koblick	M	1986-12-01
10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
10006	1953-04-20	Anneke	Preusig	F	1989-06-02
10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10
10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15
10009	1952-04-19	Sumant	Peac	F	1985-02-18
10010	1963-06-01	Duangkaew	Piveteau	F	1989-08-24
10011	1953-11-07	Mary	Sluis	F	1990-01-22
10012	1960-10-04	Patricio	Bridgland	M	1992-12-18
10013	1963-06-07	Eberhardt	Terkki	M	1985-10-20
10014	1956-02-12	Berni	Genin	M	1987-03-11
10015	1959-08-19	Guoxiang	Nooteboom	M	1987-07-02
10016	1961-05-02	Kazuhito	Cappelletti	M	1995-01-27

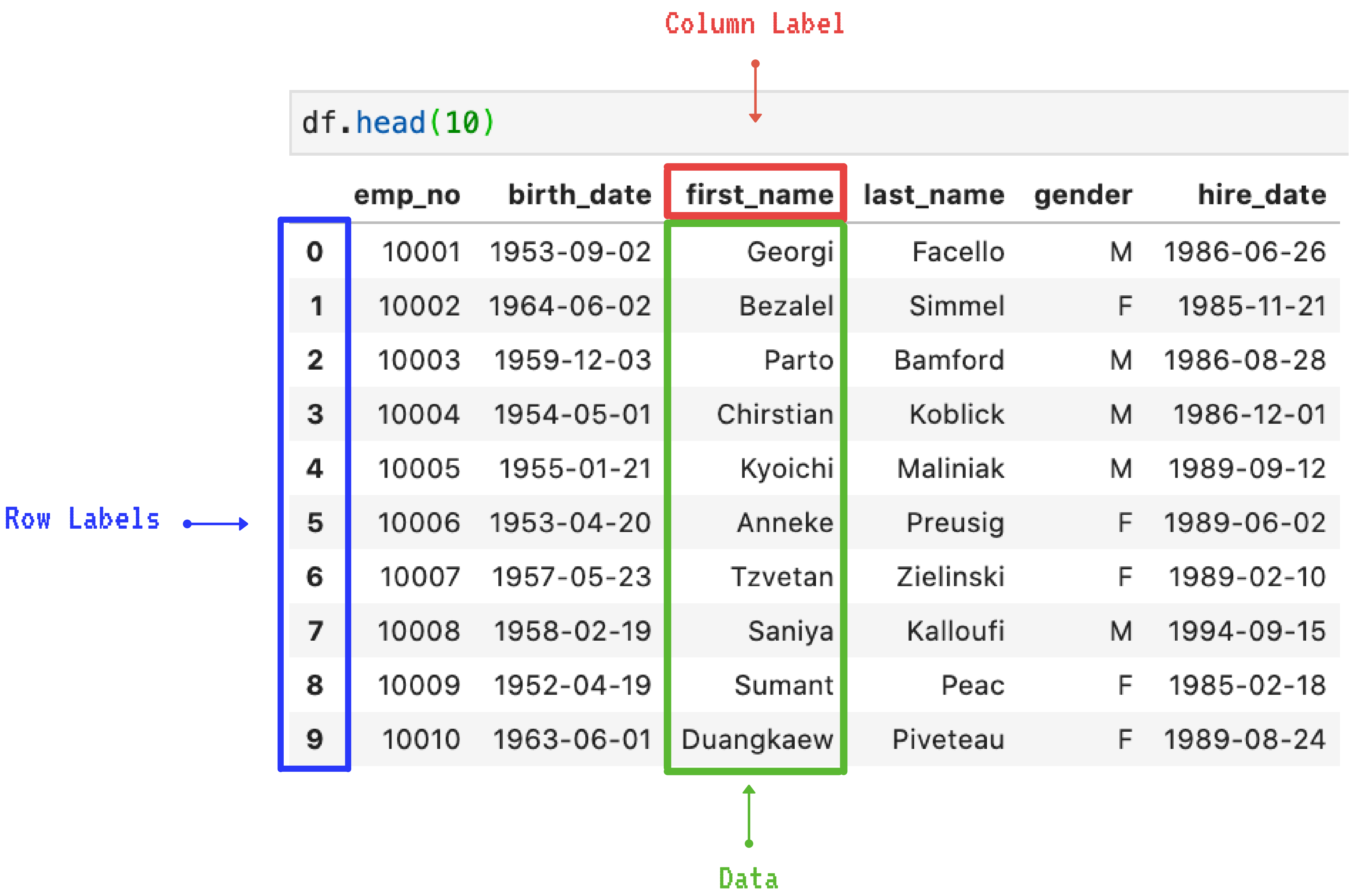
```
SELECT *
FROM EMPLOYEES
```

THINK OF THIS AS A SERIES



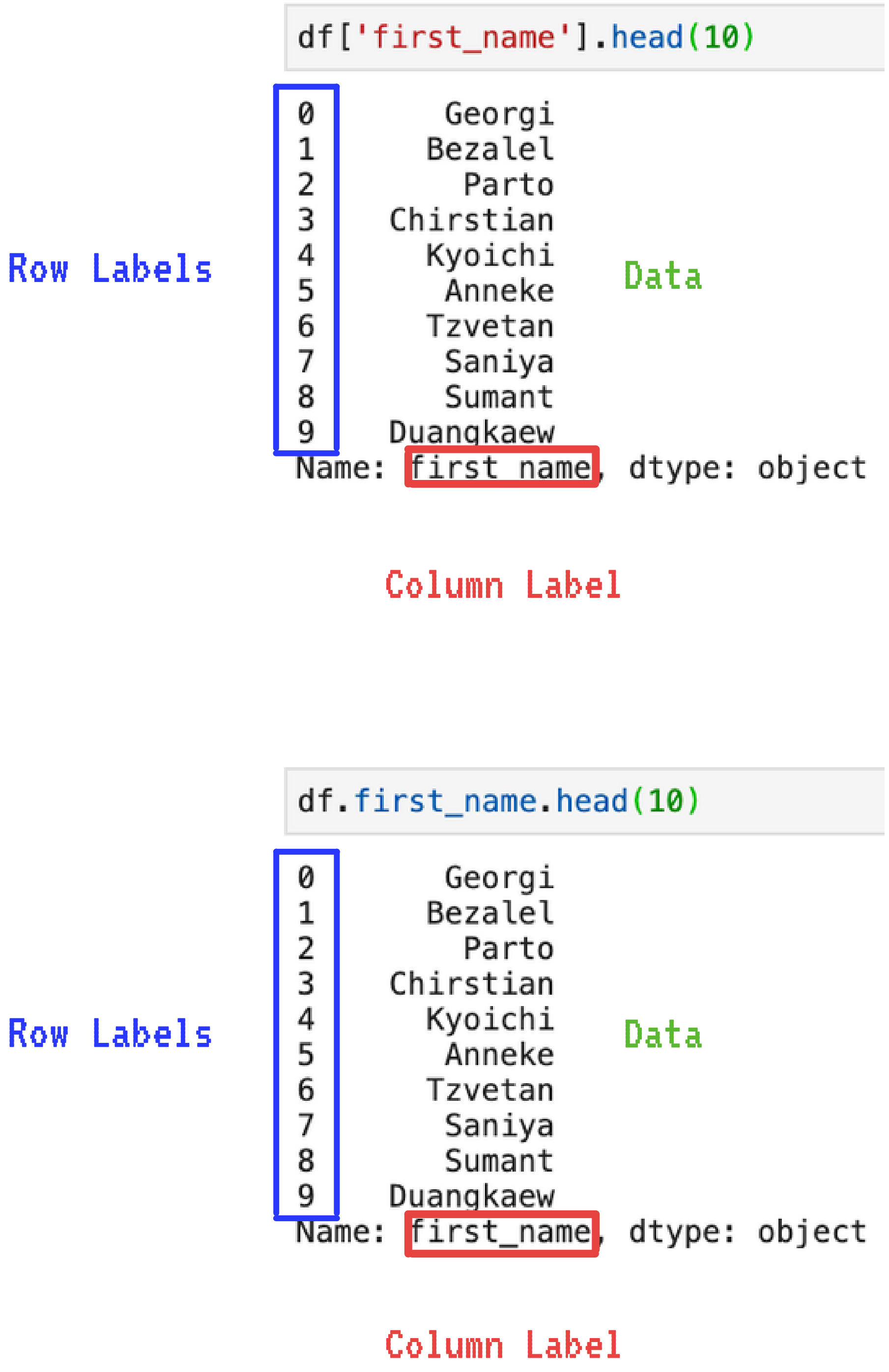
Duangkaew
Mary

```
SELECT NAME
FROM EMPLOYEES
```



```
df.head(10)
```

	emp_no	birth_date	first_name	last_name	gender	hire_date
0	10001	1953-09-02	Georgi	Facello	M	1986-06-26
1	10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
2	10003	1959-12-03	Parto	Bamford	M	1986-08-28
3	10004	1954-05-01	Chirstian	Koblick	M	1986-12-01
4	10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
5	10006	1953-04-20	Anneke	Preusig	F	1989-06-02
6	10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10
7	10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15
8	10009	1952-04-19	Sumant	Peac	F	1985-02-18
9	10010	1963-06-01	Duangkaew	Piveteau	F	1989-08-24



```
df['first_name'].head(10)
```

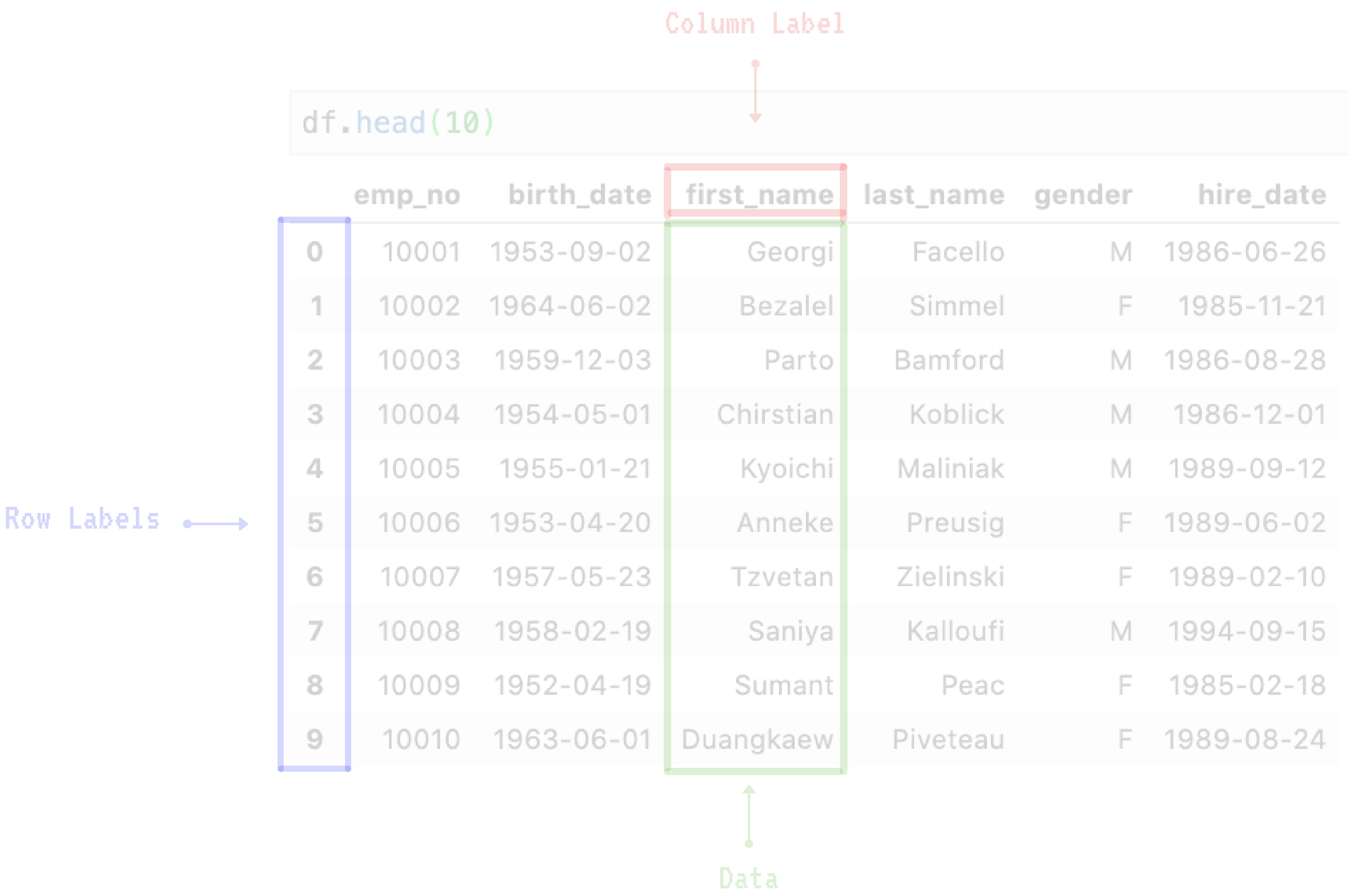
```
0       Georgi
1      Bezalel
2        Parto
3    Chirstian
4      Kyoichi
5       Anneke
6      Tzvetan
7       Saniya
8       Sumant
9    Duangkaew
Name: first_name, dtype: object
```

```
df.first_name.head(10)
```

```
0       Georgi
1      Bezalel
2        Parto
3    Chirstian
4      Kyoichi
5       Anneke
6      Tzvetan
7       Saniya
8       Sumant
9    Duangkaew
Name: first_name, dtype: object
```

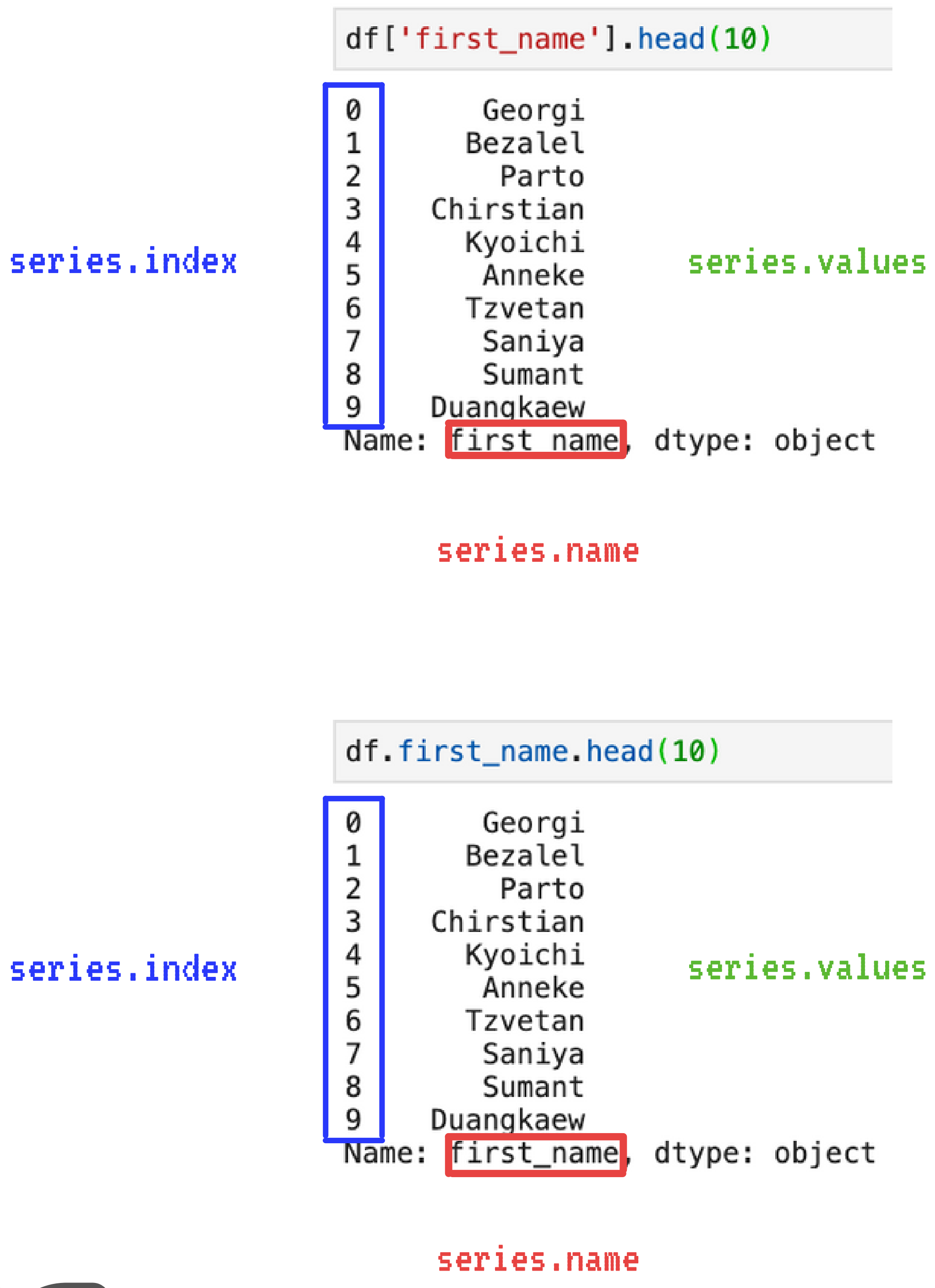
SELECT A SERIES

A LABELED, ONE-DIMENSIONAL ARRAY



```
df.head(10)
```

	emp_no	birth_date	first_name	last_name	gender	hire_date
0	10001	1953-09-02	Georgi	Facello	M	1986-06-26
1	10002	1964-06-02	Bezalel	Simmel	F	1985-11-21
2	10003	1959-12-03	Parto	Bamford	M	1986-08-28
3	10004	1954-05-01	Chirstian	Koblick	M	1986-12-01
4	10005	1955-01-21	Kyoichi	Maliniak	M	1989-09-12
5	10006	1953-04-20	Anneke	Preusig	F	1989-06-02
6	10007	1957-05-23	Tzvetan	Zielinski	F	1989-02-10
7	10008	1958-02-19	Saniya	Kalloufi	M	1994-09-15
8	10009	1952-04-19	Sumant	Peac	F	1985-02-18
9	10010	1963-06-01	Duangkaew	Piveteau	F	1989-08-24



```
df['first_name'].head(10)
```

```
0         Georgi
1        Bezalel
2          Parto
3      Chirstian
4        Kyoichi
5         Anneke
6        Tzvetan
7         Saniya
8         Sumant
9      Duangkaew
Name: first_name, dtype: object
```

```
df.first_name.head(10)
```

```
0         Georgi
1        Bezalel
2          Parto
3      Chirstian
4        Kyoichi
5         Anneke
6        Tzvetan
7         Saniya
8         Sumant
9      Duangkaew
Name: first_name, dtype: object
```

SERIES COMPONENTS

PANDAS SERIES ATTRIBUTES

```
colors = ['red', 'yellow', 'green', 'blue']

pd.Series(colors)

0      red
1      yellow
2      green
3      blue
dtype: object
```

CREATE A SERIES FROM A LIST

USING THE PANDAS SERIES CONSTRUCTOR

```
nums = np.array([5, 10, 15, 20])

pd.Series(nums)

0     5
1     10
2     15
3     20
dtype: int64
```

CREATE A SERIES FROM AN ARRAY

USING THE PANDAS SERIES CONSTRUCTOR

```
data = {'a' : 0, 'b' : 1.5, 'c' : 2, 'd': 3.5}

pd.Series(data)

a      0.0
b      1.5
c      2.0
d      3.5
dtype: float64
```

CREATE A SERIES FROM A DICTIONARY

USING THE PANDAS SERIES CONSTRUCTOR

WHAT'S SO GREAT ABOUT PEACE, LOVE AND PANDAS?

1. Built-in Attributes
2. Built-in Methods
3. Vectorized Operations

SERIES ATTRIBUTES DO...

- return valuable information about our Series object. (think properties)

- use dot notation to access the attributes.

FOR EXAMPLE...

SERIES.DTYPE

Returns the data type of the series.

SERIES.SIZE

Returns an integer representing the number of rows in our Series.

SERIES.SHAPE

Returns a tuple containing number of rows and number of columns.

SERIES ATTRIBUTES DO NOT...

- perform operations or calculations.

- require parentheses.

FOR EXAMPLE...

SERIES.DTYPE

Returns the data type of the series.

SERIES.SIZE

Returns an int representing the number of rows in our Series.

SERIES.SHAPE

Returns a tuple containing number of rows and number of columns.

SERIES METHODS DO...

- perform calculations or operations.
(think functions)

- use dot notation.

- require parentheses and provide parameters for customization.

FOR EXAMPLE...

SERIES.HEAD(N=5)

Returns a new Series made up of the first n rows of our original Series.

SERIES.TAIL(N=5)

Returns a new Series made up of the last n rows of our original Series.

SERIES.VALUE_COUNTS()

Returns a new Series with unique values as the index and a count as values.

SERIES METHODS DO NOT...

- necessarily require us to provide an argument; we can simply use default arguments.

- mutate our original Series.
(inplace=False)

FOR EXAMPLE...

```
SERIES.HEAD(N=5)
```

Returns a new Series made up of the first n rows of our original Series.

```
SERIES.TAIL(N=5)
```

Returns a new Series made up of the last n rows of our original Series.

```
SERIES.VALUE_COUNTS()
```

Returns a new Series with unique values as the index and a count as values.

VECTORIZED OPERATIONS

```
import pandas as pd
```

Get ready to fall in love

This means that I can call a function on an entire Series instead of a single string or scalar value!

```
colors = ['red', 'yellow', 'green', 'blue']

colors_series = pd.Series(colors)

colors_series.str.capitalize()

0      Red
1      Yellow
2      Green
3      Blue
dtype:object
```

LET´S DIVE INTO THE NOTEBOOK!