Advanced Data Management (CSCI 640/490)

Dataframes

Dr. David Koop

Relational Algebra

- Definition: A procedural language consisting of a set of operations that take one or two relations as input and produce a new relation as their result.
- Six **basic** operators
  - select: σ
  - project: ∏
  - union: ∪
  - set difference: –
  - Cartesian product: x
  - rename: ρ
-

[A. Silberschatz et al.]

Equivalent Queries

- Example: Find information about courses taught by instructors in the Physics department
- Query 1:

$$\sigma_{\text{dept\_name=“Physics”}} (\texttt{instructor} \bowtie_{\text{instructor.ID = teaches.ID}} \texttt{teaches})$$

- Query 2

$$(\sigma_{\text{dept\_name=“Physics”}} (\texttt{instructor})) \bowtie_{\text{instructor.ID = teaches.ID}} \texttt{teaches}$$

- The **order** of joins is one focus of some of the work on query optimization

[A. Silberschatz et al.]

Components of SQL

- **Data Definition Language (DDL)**: the specification of information about relations, including schema, types, integrity constraints, indices, storage
- **Data Manipulation Language (DML)**: provides the ability to query information from the database and to insert tuples into, delete tuples from, and modify tuples in the database.
- **Integrity**: the DDL includes commands for specifying integrity constraints.
- **View definition**: The DDL includes commands for defining views.
- Also: **Transaction control**, **embedded and dynamic SQL**, **authorization**

[A. Silberschatz et al.]

Create Table

- An SQL relation is defined using the create table command:

  `create table r` $(A_1\ D_1,\ A_2\ D_2,\ \ldots,\ A_n\ D_n,\ (C_1),\ \ldots,\ (C_k))$

  - `r` is the **name** of the relation
  - each A_i is an **attribute name** in the schema of relation `r`
  - D_i is the **data type** of values in the domain of attribute A_i

C_i are integrity constraints

- Example:

```
create table instructor(
        ID              char(5),
        name            varchar(20),
        dept_name       varchar(20),
        salary          numeric(8,2));
```

[A. Silberschatz et al.]

Create Table

- An SQL relation is defined using the create table command:

  `create table r` $(A_1\ D_1, A_2\ D_2, \ldots, A_n\ D_n, (C_1), \ldots, (C_k))$

  - `r` is the **name** of the relation
  - each A_i is an **attribute name** in the schema of relation `r`
  - D_i is the **data type** of values in the domain of attribute A_i

C_i are integrity constraints

- Example:

```
create table instructor(
        ID              char(5),
        name            varchar(20),
        dept_name       varchar(20),
        salary          numeric(8,2));
```

[A. Silberschatz et al.]

Create Table

- An SQL relation is defined using the create table command:

  `create table r` $(A_1\ D_1, A_2\ D_2, \ldots, A_n\ D_n, (C_1), \ldots, (C_k))$

  - `r` is the **name** of the relation
  - each A_i is an **attribute name** in the schema of relation `r`
  - D_i is the **data type** of values in the domain of attribute A_i

C_i are integrity constraints

- Example:

```
create table instructor(
        ID              char(5),
        name            varchar(20),
        dept_name       varchar(20),
        salary          numeric(8,2));
```

[A. Silberschatz et al.]

Create Table

- An SQL relation is defined using the create table command:

  `create table r` $(A_1\ D_1, A_2\ D_2, \ldots, A_n\ D_n, (C_1), \ldots, (C_k))$

  - `r` is the **name** of the relation
  - each A_i is an **attribute name** in the schema of relation `r`
  - D_i is the **data type** of values in the domain of attribute A_i

C_i are integrity constraints

- Example:

```
create table instructor(
         ID            char(5),
         name          varchar(20),
         dept_name     varchar(20),
         salary        numeric(8,2));
```

[A. Silberschatz et al.]

Basic Query Structure

- A typical SQL query has the form:

  `select` $A_1, A_2, \ldots, A_n$

  `from` $r_1, r_2, \ldots, r_m$

  `where` P

  - A_i represents an **attribute**
  - r_i represents a **relation**
  - P is a **predicate**.

- The result of an SQL query is a **relation**

[A. Silberschatz et al.]

Select

- The **select** clause lists the attributes desired in the result of a query
  - corresponds to the projection operation of the relational algebra
- Example: Find the names of all instructors

  -
    ```
    select name
    from instructor;
    ```
- Example: Find the department names of all instructors (no duplicates)

  -
    ```
    select distinct dept_name
      from instructor;
    ```
- Example: Find the monthly salary of each instructor

  -
    ```
    select ID, name, salary/12 as monthly_salary
    ```

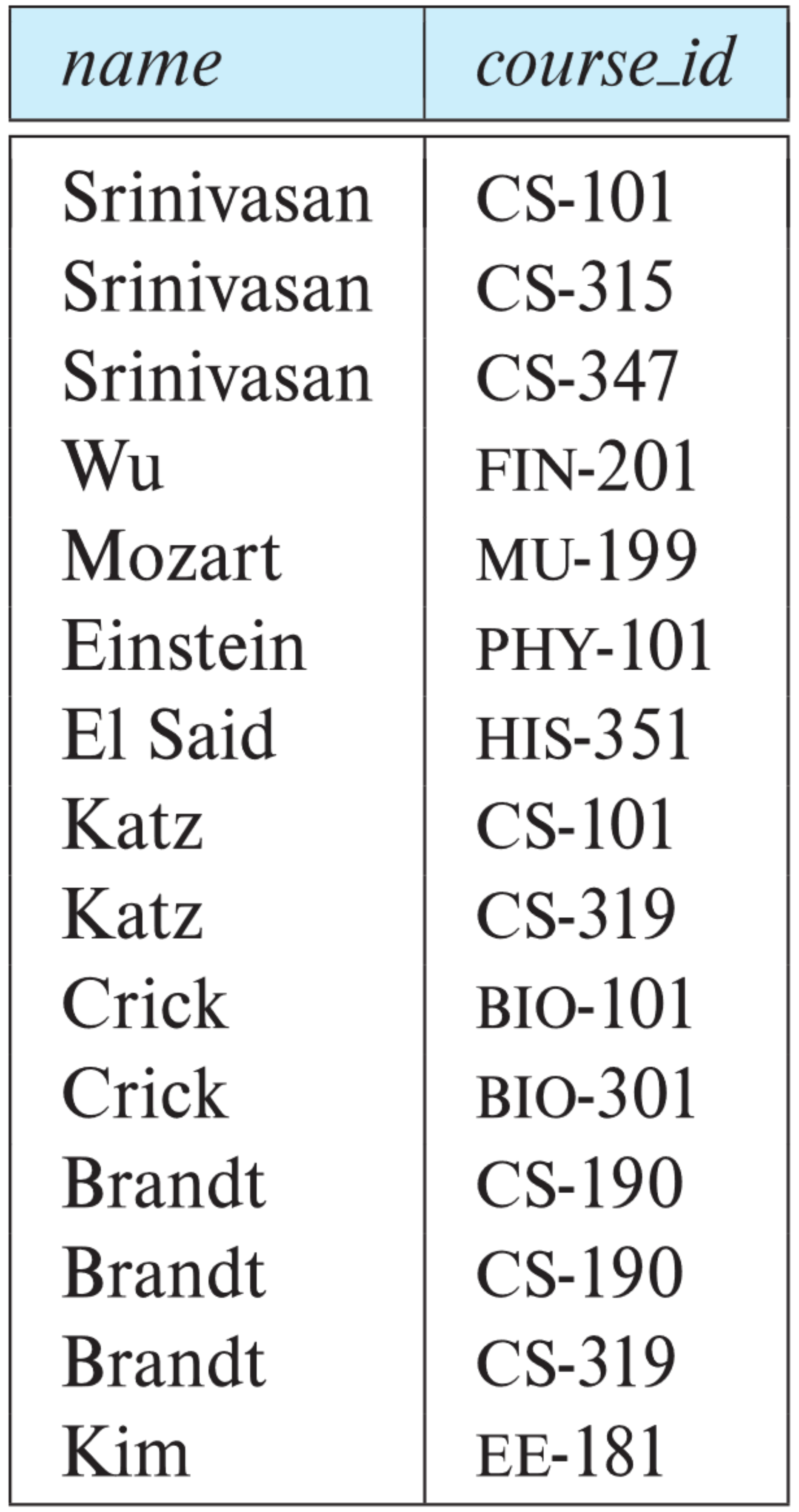
[A. Silberschatz et al.]

From & Where Clauses

- Find the names of all instructors who have taught some course and that course_id

  -
    ```
    select name, course_id
    from instructor, teaches
    where instructor.ID = teaches.ID
    ```
- Find the names of all instructors in the Art department who have taught some course and the course_id

  -
    ```
    select name, course_id
    from instructor, teaches
    where instructor.ID = teaches.ID
    and instructor.dept_name = 'Art'
    ```



name	*course_id*
Srinivasan	CS-101
Srinivasan	CS-315
Srinivasan	CS-347
Wu	FIN-201
Mozart	MU-199
Einstein	PHY-101
El Said	HIS-351
Katz	CS-101
Katz	CS-319
Crick	BIO-101
Crick	BIO-301
Brandt	CS-190
Brandt	CS-190
Brandt	CS-319
Kim	EE-181

Result of “For all instructors in the university who have find their names and the course ID of all courses they

[A. Silberschatz et al.]

Aggregate Functions

- Find the average salary of instructors in the Computer Science department

  ```
  select avg (salary)
  from instructor
  where dept_name = 'Comp. Sci.';
  ```

- Find the total number of instructors who teach a course in the Spring 2018 semester

  ```
  select count(distinct ID)
  from teaches
  where semester = 'Spring' and year = 2018;
  ```

- Find the number of tuples in the course relation

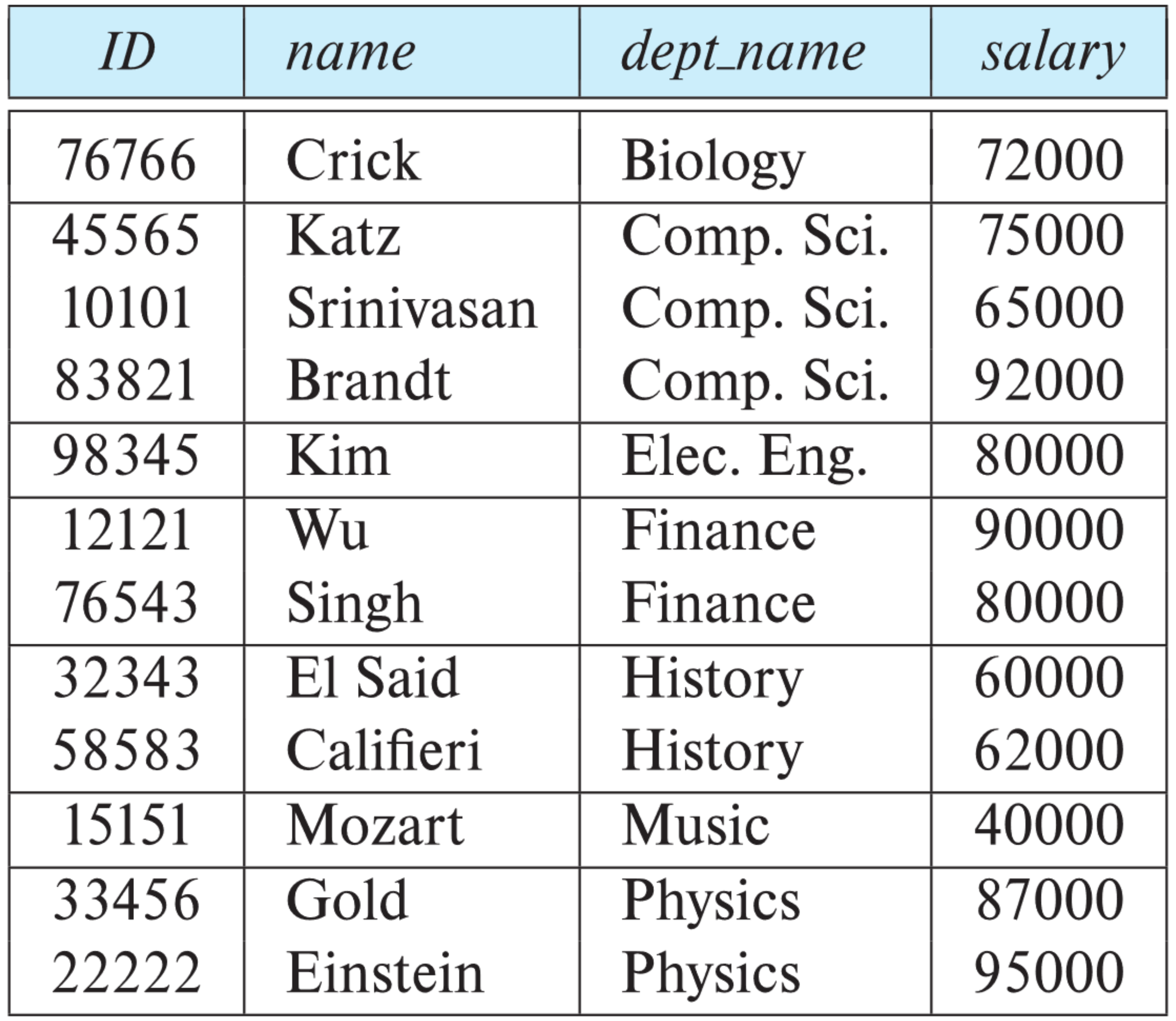
  ```
  select count(*)
  from course;
  ```

[A. Silberschatz et al.]

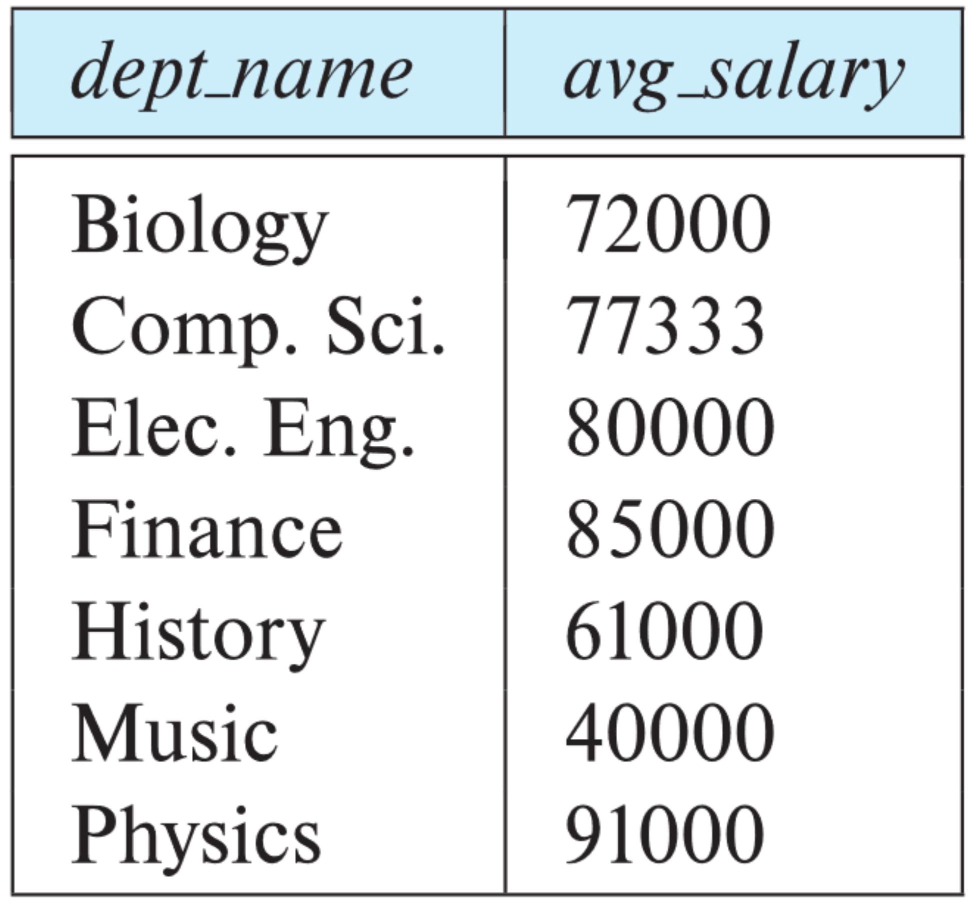
Group By

- Find the average salary of instructors in each department

  -
    ```
    select dept_name, avg(salary) as avg_salary
    from instructor
    group by dept_name;
    ```



ID	*name*	*dept_name*	*salary*
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000



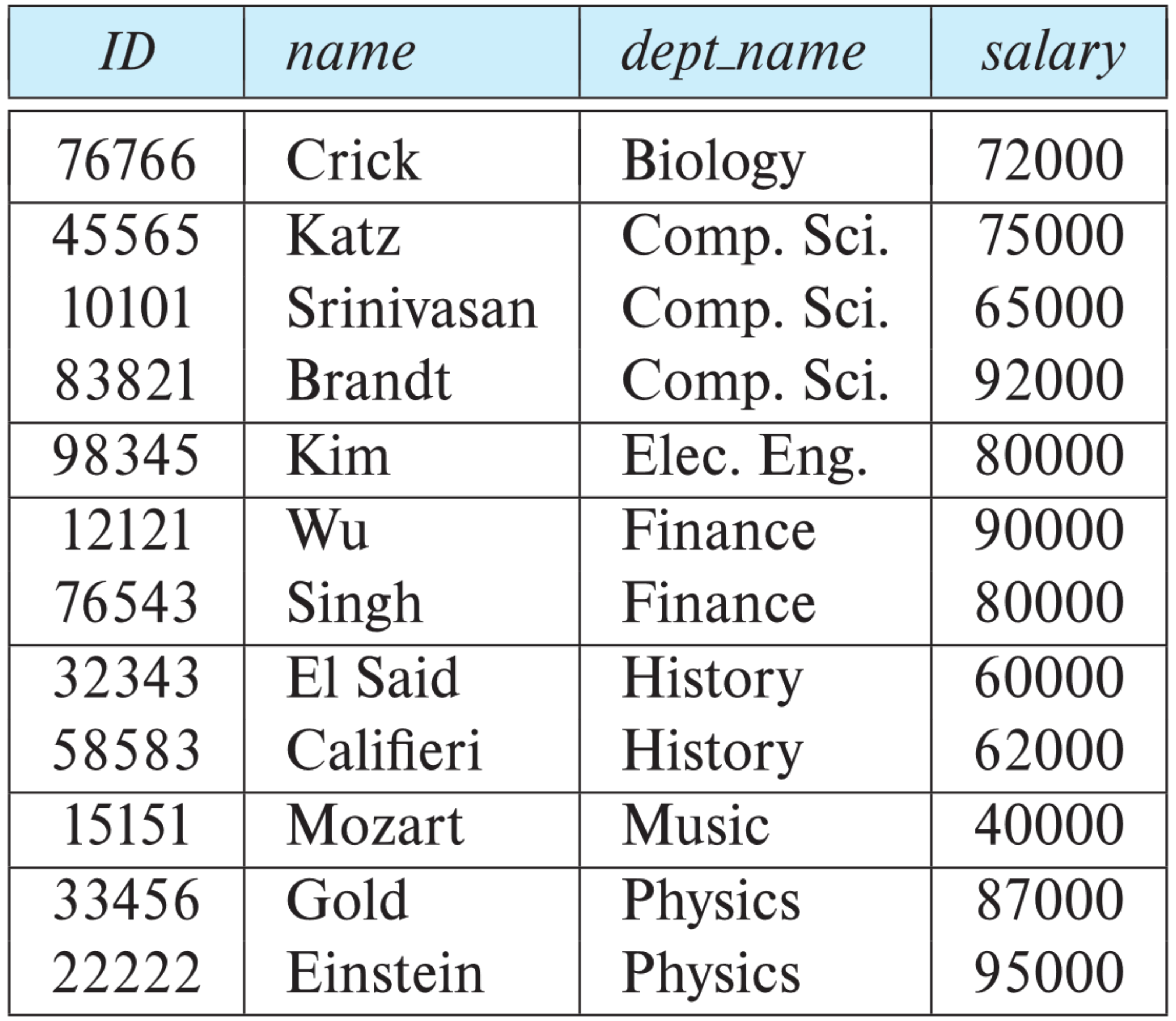
dept_name	*avg_salary*
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

[A. Silberschatz et al.]

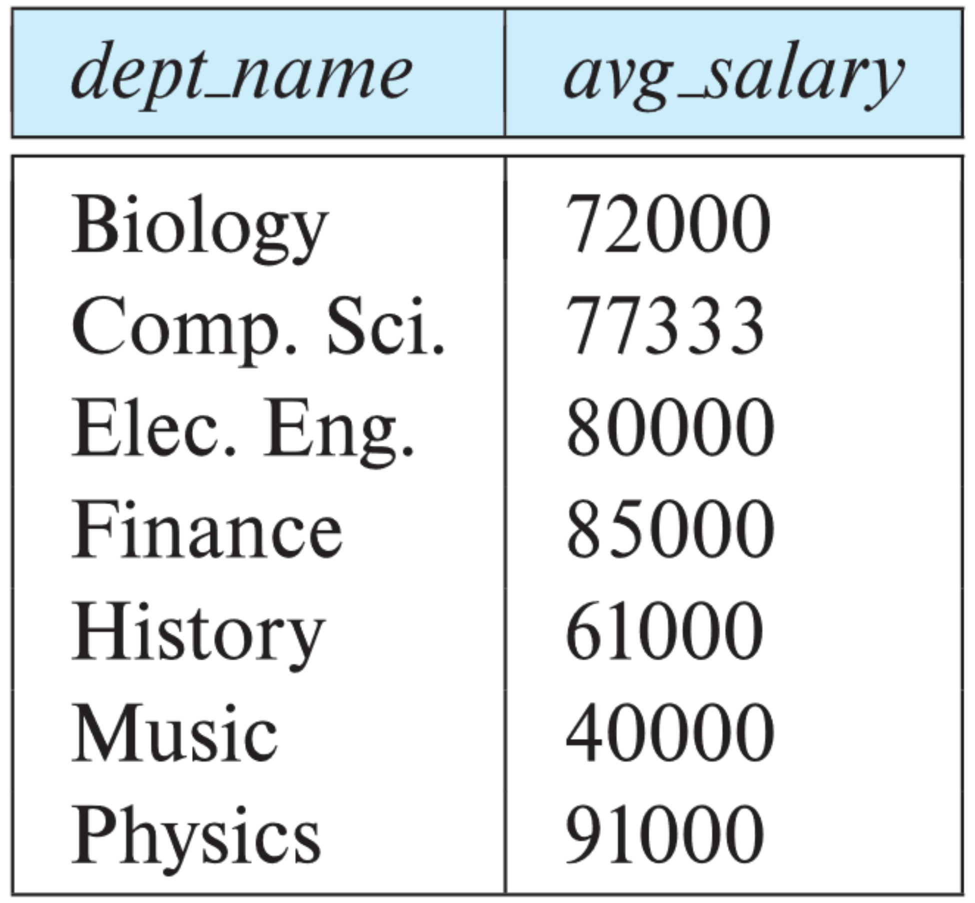
Group By

- Find the average salary of instructors in each department

```
- select dept_name, avg(salary) as avg_salary
  from instructor
  group by dept_name;
```



ID	*name*	*dept_name*	*salary*
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000



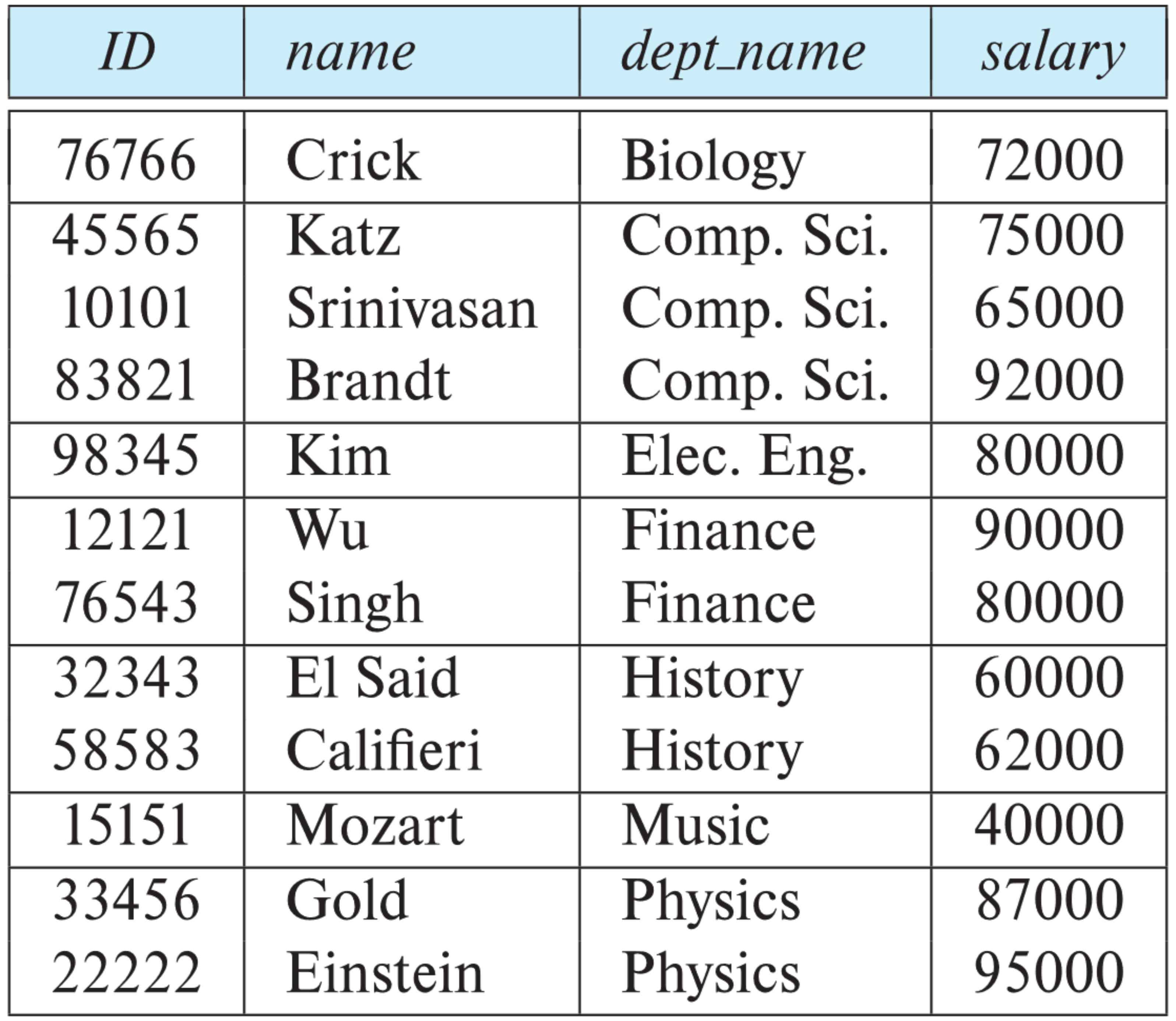
dept_name	*avg_salary*
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

[A. Silberschatz et al.]

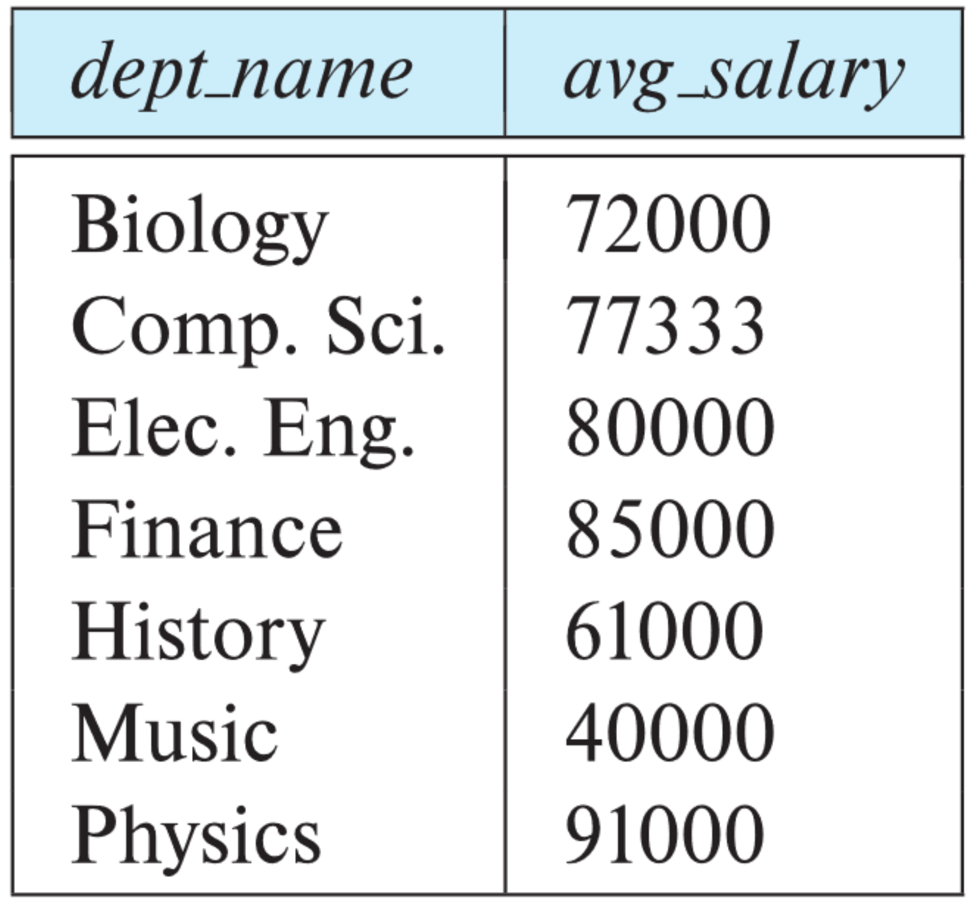
Group By

- Find the average salary of instructors in each department

```
- select dept_name, avg(salary) as avg_salary
  from instructor
  group by dept_name;
```



ID	*name*	*dept_name*	*salary*
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000



dept_name	*avg_salary*
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

[A. Silberschatz et al.]

Deletion

- Delete all instructors: `delete from instructor;`
- Delete all instructors from the Finance department

    -
      ```
      delete from instructor
      where dept_name= 'Finance’;
      ```
- Delete all tuples in the instructor relation for those instructors associated with a department located in the Watson building

    -
      ```
      delete from instructor
      where dept_name in (select dept_name
                          from department
                          where building = 'Watson');
      ```

[A. Silberschatz et al.]

Deletion

- Delete all instructors: **delete from** `instructor;`
- Delete all instructors from the Finance department

  -
    ```
    delete from instructor
    where dept_name= 'Finance’;
    ```
- Delete all tuples in the instructor relation for those instructors associated with a department located in the Watson building

  -
    ```
    delete from instructor
    where dept_name in (select dept_name
                          from department
                          where building = 'Watson');
    ```

[A. Silberschatz et al.]

Insertion

- Add a new tuple to course

  -
    ```
    insert into course
       values ('CS-437', 'Database Systems', 'Comp. Sci.', 4);
    ```
- or…

  -
    ```
    insert into course(course_id, title, dept_name, credits)
       values ('CS-437', 'Database Systems', 'Comp. Sci.', 4);
    ```
- Add a new tuple to `student` with `tot_creds` set to `null`

  -
    ```
    insert into student
       values ('3003', 'Green', 'Finance', null);
    ```

[A. Silberschatz et al.]

Updates

- Give a 5% salary raise to all instructors

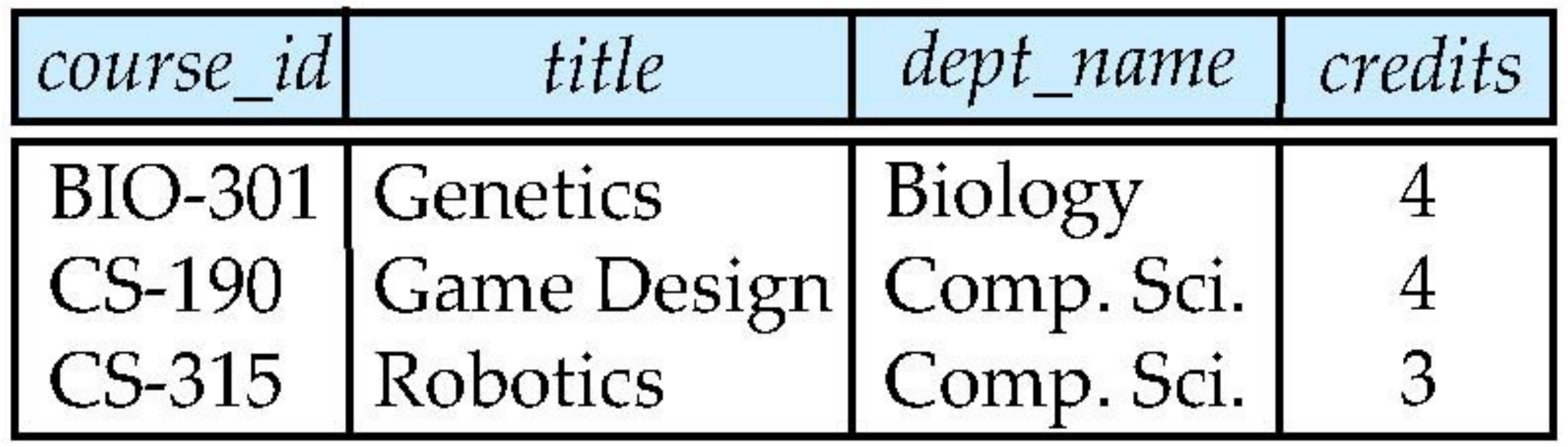
  -
    ```
    update instructor
    set salary = salary * 1.05
    ```
- Give a 5% salary raise to those instructors who earn less than 70000

  -
    ```
    update instructor
    set salary = salary * 1.05
    where salary < 70000;
    ```
- Give a 5% salary raise to instructors whose salary is less than average

  -
    ```
    update instructor
    set salary = salary * 1.05
    where salary < (select avg(salary) from instructor);
    ```

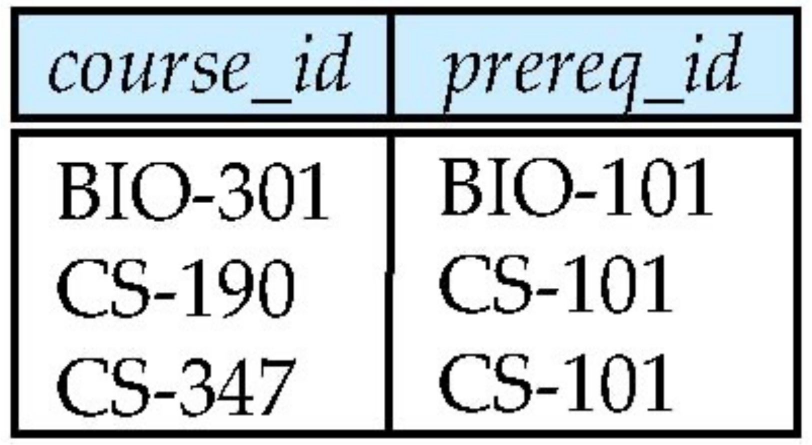
[A. Silberschatz et al.]

Joins



course_id	title	dept_name	credits
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3

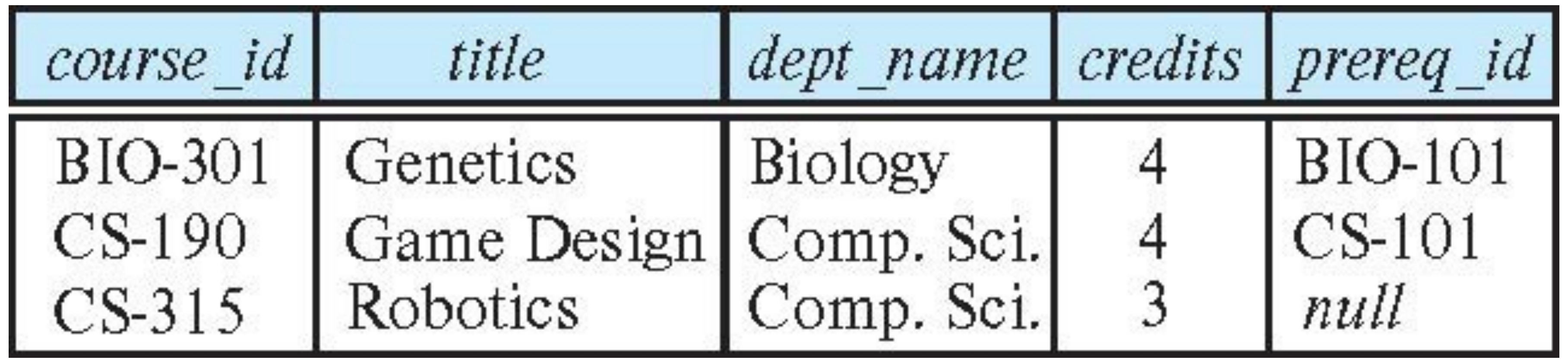
`course`



course_id	prereq_id
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

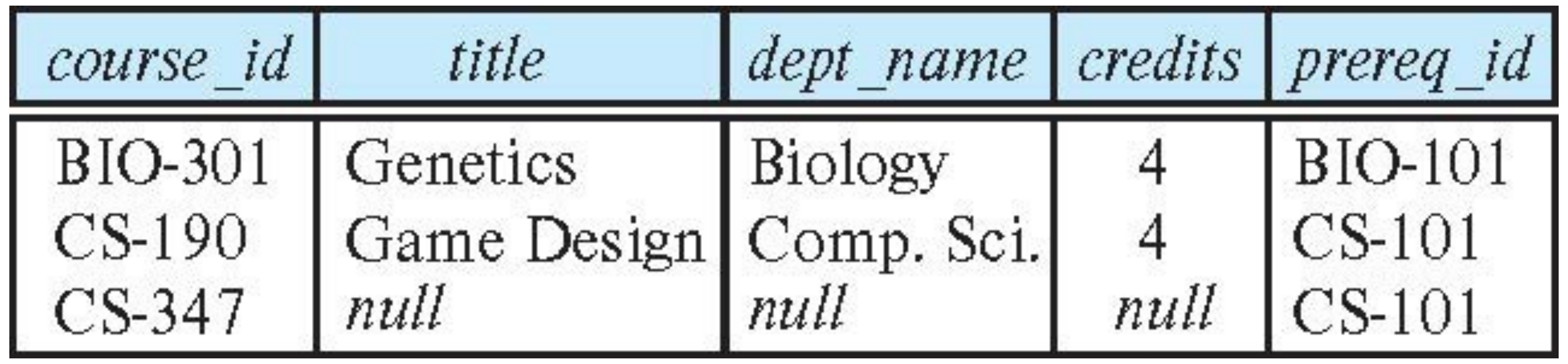
`prereq`

Left Join



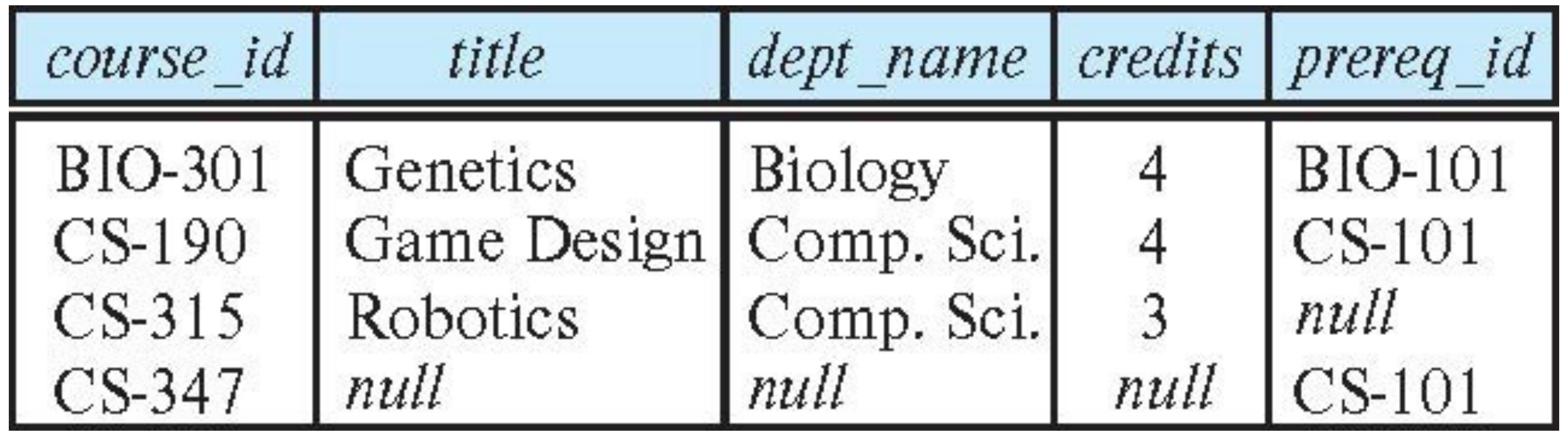
course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	*null*

Right Join



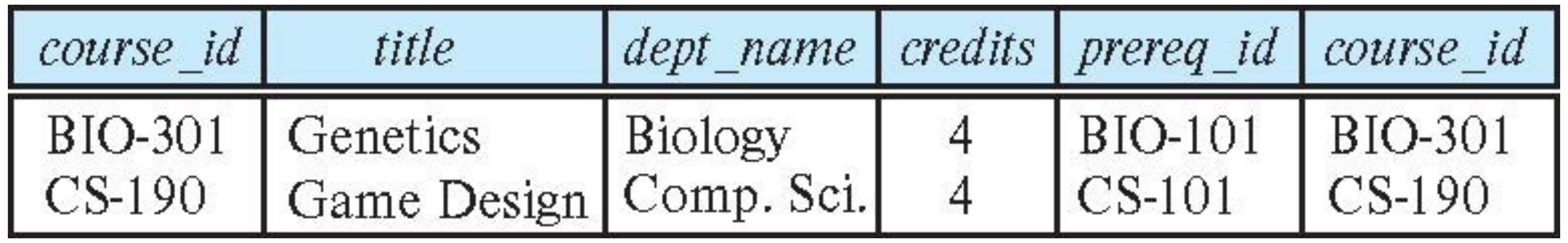
course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-347	*null*	*null*	*null*	CS-101

(Full) Outer Join



course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	*null*
CS-347	*null*	*null*	*null*	CS-101

Inner Join



course_id	title	dept_name	credits	prereq_id	course_id
BIO-301	Genetics	Biology	4	BIO-101	BIO-301
CS-190	Game Design	Comp. Sci.	4	CS-101	CS-190

[A. Silberschatz et al.]

Assignment 1

- Data analysis using python (and a few standard libraries)
- Do not use pandas, polars, or database queries for this assignment!
- Turn in a Jupyter notebook (.ipynb file)
  - You can download and edit a1.ipynb provided with the assignment
  - Upload the final notebook to Blackboard
  - Make sure your code runs from top to bottom!

Arrays

What is the difference between an array and a list (or a tuple)?

Arrays

- Usually a fixed size—lists are meant to change size
- Are mutable—tuples are not
- Store only one type of data—lists and tuples can store anything
- Are faster to access and manipulate than lists or tuples
- Can be multidimensional:
  - Can have list of lists or tuple of tuples but no guarantee on shape
  - Multidimensional arrays are rectangles, cubes, etc.

Why NumPy?

- Fast **vectorized** array operations for data munging and cleaning, subsetting and filtering, transformation, and any other kinds of computations
- Common array algorithms like sorting, unique, and set operations
- Efficient descriptive statistics and aggregating/summarizing data
- Data alignment and relational data manipulations for merging and joining together heterogeneous data sets
- Expressing conditional logic as array expressions instead of loops with `if-elif-else` branches
- Group-wise data manipulations (aggregation, transformation, function application).

[W. McKinney, Python for Data Analysis]

Creating arrays

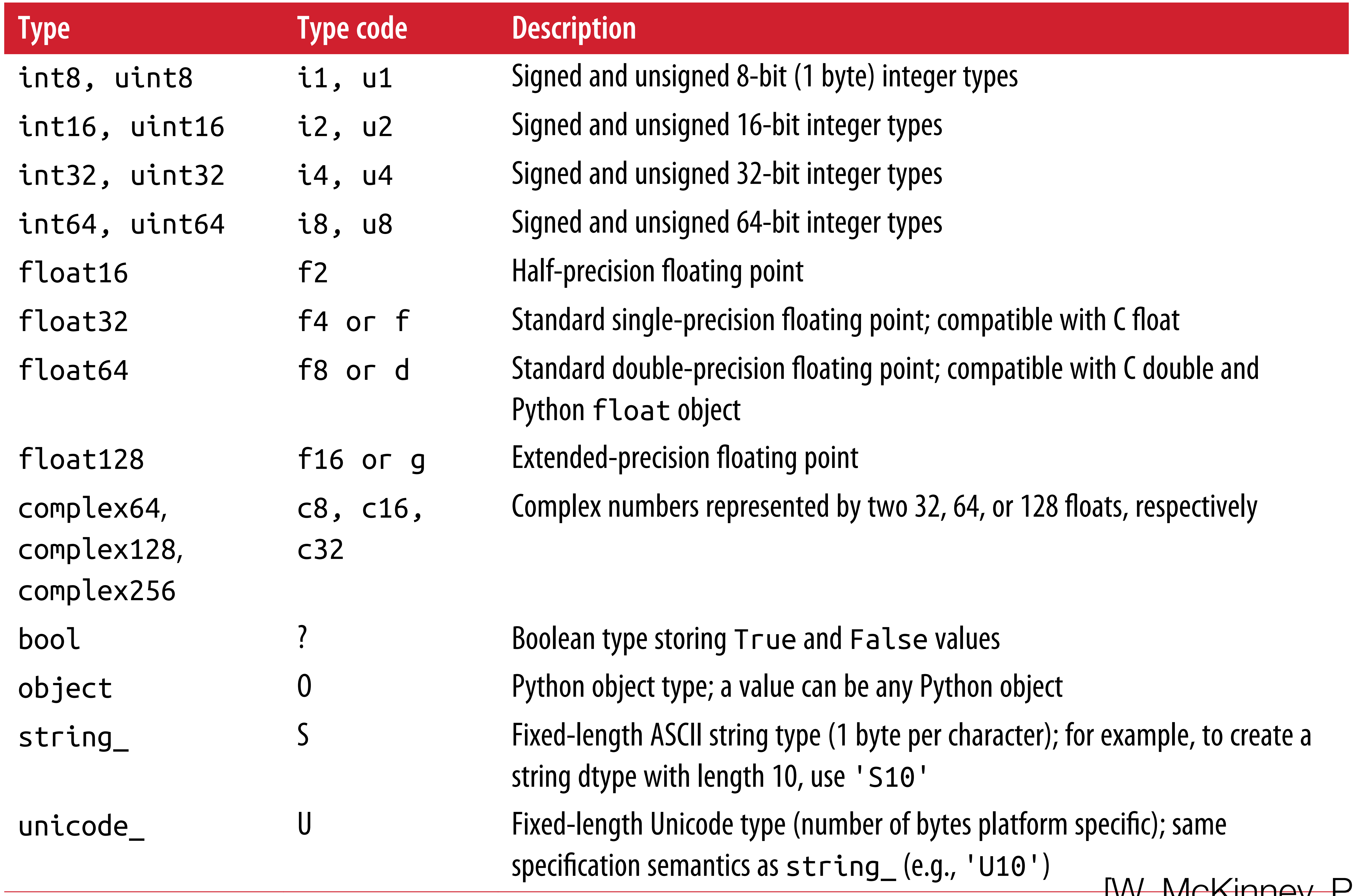
- `import numpy as np`
- `data1 = [6, 7, 8, 0, 1]`
  `arr1 = np.array(data1)`
- `data2 = [[1.5,2,3,4],[5,6,7,8]]`
  `arr2 = np.array(data2)`
- `data3 = np.array([6, "abc", 3.57]) # !!! check !!!`
- Can check the type of an array in `dtype` property
- Types:
  - `arr1.dtype # dtype('int64')`
  - `arr3.dtype # dtype('<U21'), unicode plus # chars`

Types

- "But I thought Python wasn't stingy about types…"
- numpy aims for speed
- Able to do array arithmetic
- int16, int32, int64, float32, float64, bool, object
- Can specify type explicitly
  - `arr1_float = np.array(data1, dtype='float64')`
- `astype` method allows you to convert between different types of arrays:

```
arr = np.array([1, 2, 3, 4, 5])
arr.dtype
float_arr = arr.astype(np.float64)
```

numpy data types (dtypes)



Type	Type code	Description
`int8, uint8`	`i1, u1`	Signed and unsigned 8-bit (1 byte) integer types
`int16, uint16`	`i2, u2`	Signed and unsigned 16-bit integer types
`int32, uint32`	`i4, u4`	Signed and unsigned 32-bit integer types
`int64, uint64`	`i8, u8`	Signed and unsigned 64-bit integer types
`float16`	`f2`	Half-precision floating point
`float32`	`f4 or f`	Standard single-precision floating point; compatible with C float
`float64`	`f8 or d`	Standard double-precision floating point; compatible with C double and Python `float` object
`float128`	`f16 or g`	Extended-precision floating point
`complex64, complex128, complex256`	`c8, c16, c32`	Complex numbers represented by two 32, 64, or 128 floats, respectively
`bool`	?	Boolean type storing `True` and `False` values
`object`	O	Python object type; a value can be any Python object
`string_`	S	Fixed-length ASCII string type (1 byte per character); for example, to create a string dtype with length 10, use `'S10'`
`unicode_`	U	Fixed-length Unicode type (number of bytes platform specific); same specification semantics as `string_` (e.g., `'U10'`)

[W. McKinney, Python for Data Analysis]

Array Shape

- Our normal way of checking the size of a collection is… `len`
- How does this work for arrays?

-
  ```
  arr1 = np.array([1,2,3,6,9])
  len(arr1) # 5
  ```

-
  ```
  arr2 = np.array([[1.5,2,3,4],[5,6,7,8]])
  len(arr2) # 2
  ```
- All dimension lengths → shape: `arr2.shape # (2,4)`
- Number of dimensions: `arr2.ndim # 2`
- Can also reshape an array:
  - `arr2.reshape(4,2)`
  - `arr2.reshape(-1,2) # what happens here?`

Array Programming

- Lists:

  -
    ```
    c = []
    for i in range(len(a)):
        c.append(a[i] + b[i])
    ```
- How to improve this?

Array Programming

- Lists:

  -
    ```
    c = []
    for i in range(len(a)):
        c.append(a[i] + b[i])
    ```
  - `c = [aa + bb for aa, bb in zip(a,b)]`
- NumPy arrays:
  - `c = a + b`
- More functional-style than imperative
- **Internal iteration** instead of external

Operations

- `a = np.array([1,2,3])`
  `b = np.array([6,4,3])`
- (Array, Array) Operations (**Element-wise**)
  - Addition, Subtraction, Multiplication
  - `a + b # array([7, 6, 6])`
- (Scalar, Array) Operations (**Broadcasting**):
  - Addition, Subtraction, Multiplication, Division, Exponentiation
  - `a ** 2 # array([1, 4, 9])`
  - `b + 3 # array([9, 7, 6])`

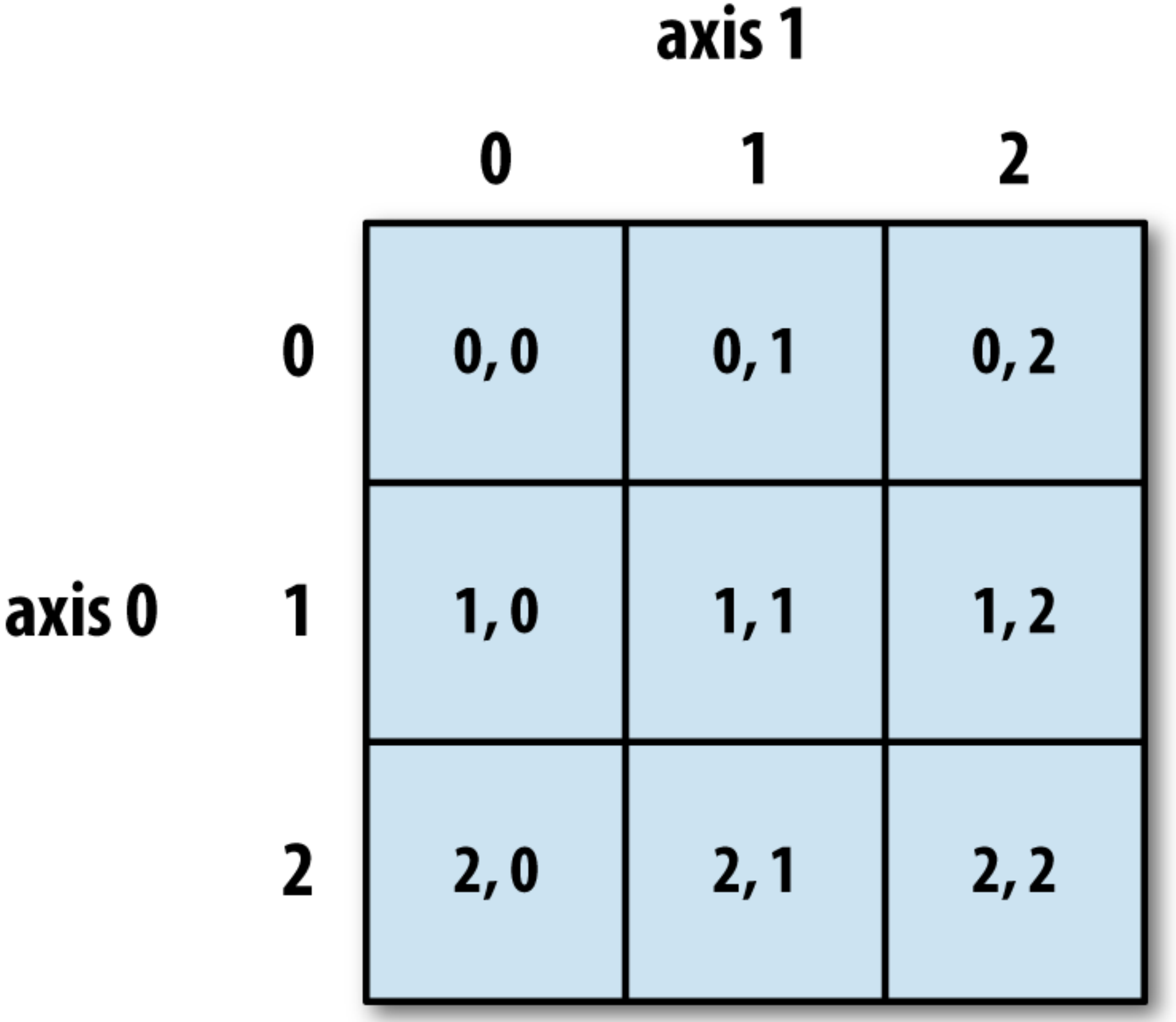
More on Array Creation

- Zeros: `np.zeros(10)`
- Ones: `np.ones((4,5)) # shape`
- Empty: `np.empty((2,2))`
- _like versions: pass an existing array and matches shape with specified contents
- Range: `np.arange(15) # constructs an array, not iterator!`

Indexing

- Same as with lists plus shorthand for 2D+
  - `arr1 = np.array([6, 7, 8, 0, 1])`
  - `arr1[1]`
  - `arr1[-1]`
- What about two dimensions?
  - `arr2 = np.array([[1.5,2,3,4],[5,6,7,8]])`
  - `arr[1][1]`
  - `arr[1,1] # shorthand`

2D Indexing



		axis 1		
		0	1	2
	0	0, 0	0, 1	0, 2
axis 0	1	1, 0	1, 1	1, 2
	2	2, 0	2, 1	2, 2

[W. McKinney, Python for Data Analysis]

Slicing

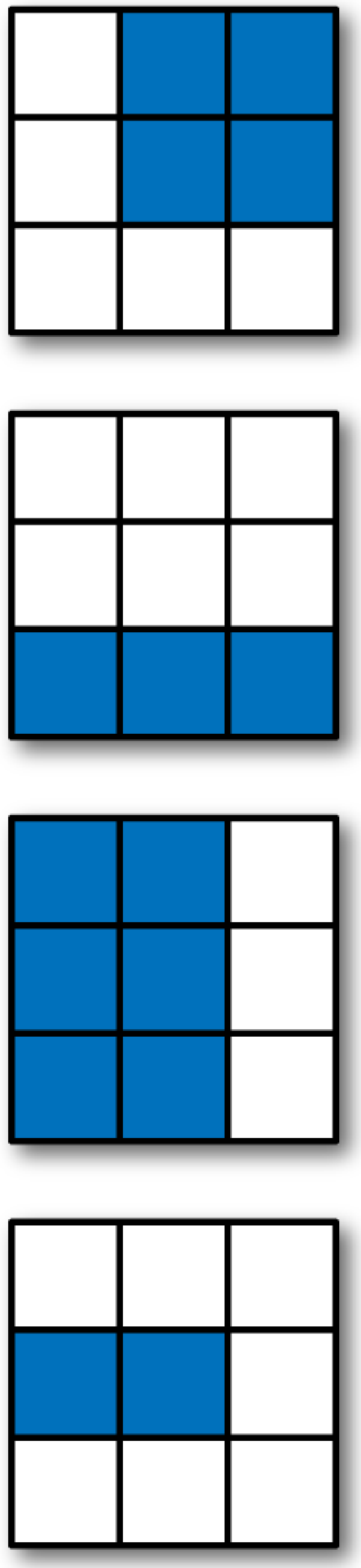
- 1D: Similar to lists
  - `arr1 = np.array([6, 7, 8, 0, 1])`
  - `arr1[2:5] # np.array([8,0,1]), sort of`
- Can **mutate** original array:
  - `arr1[2:5] = 3 # supports assignment`
  - `arr1 # the original array changed`
- Slicing returns **views** (copy the array if original array shouldn't change)
  - `arr1[2:5] # a view`
  - `arr1[2:5].copy() # a new array`

Slicing

- 2D+: comma separated indices as shorthand:
  - `arr2 = np.array([[1.5,2,3,4],[5,6,7,8]])`
  - `a[1:3,1:3]`
  - `a[1:3,:] # works like in single-dimensional lists`
- Can combine index and slice in different dimensions
  - `a[1,:] # gives a row`
  - `a[:,1] # gives a column`

2D Array Slicing

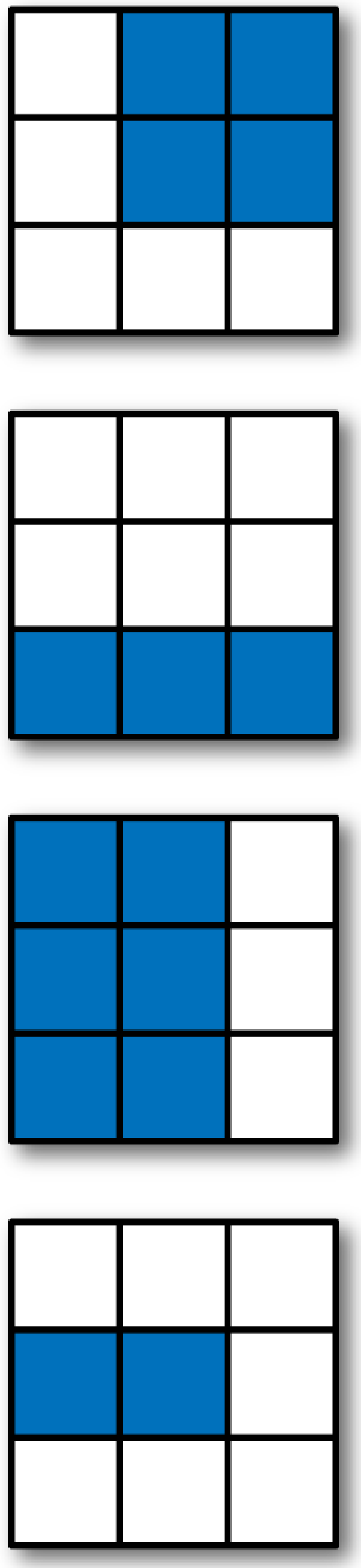
How to obtain the blue slice from array `arr`?



[W. McKinney, Python for Data Analysis]

2D Array Slicing

How to obtain the blue slice from array `arr`?

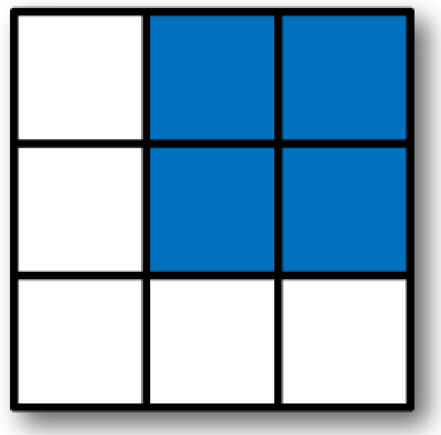
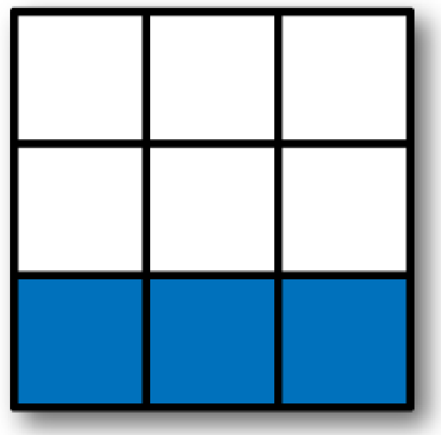
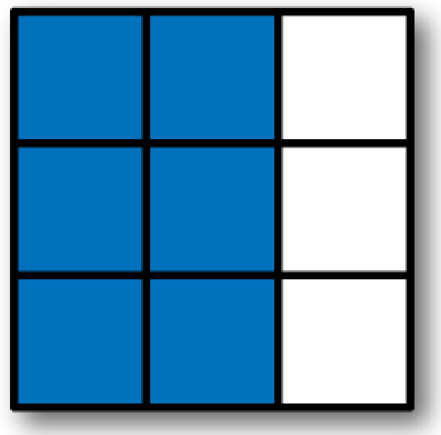
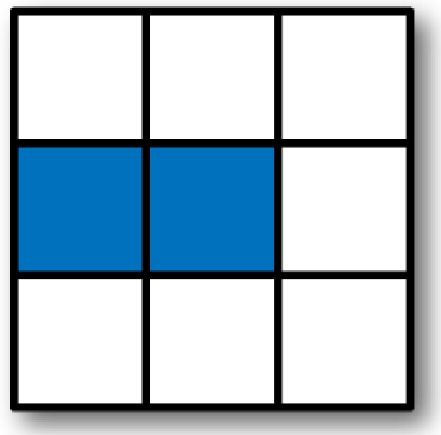


Expression	Shape
arr[:2, 1:]	(2, 2)

[W. McKinney, Python for Data Analysis]

2D Array Slicing

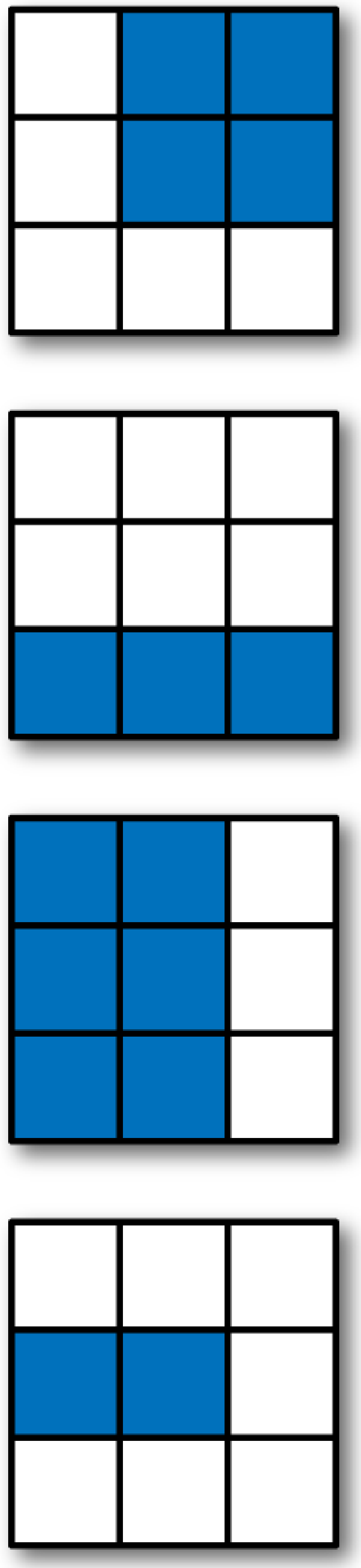
How to obtain the blue slice from array `arr`?

	Expression	Shape
	arr[:2, 1:]	(2, 2)
	arr[2] arr[2, :] arr[2:, :]	(3,) (3,) (1, 3)
		
		

[W. McKinney, Python for Data Analysis]

2D Array Slicing

How to obtain the blue slice from array `arr`?

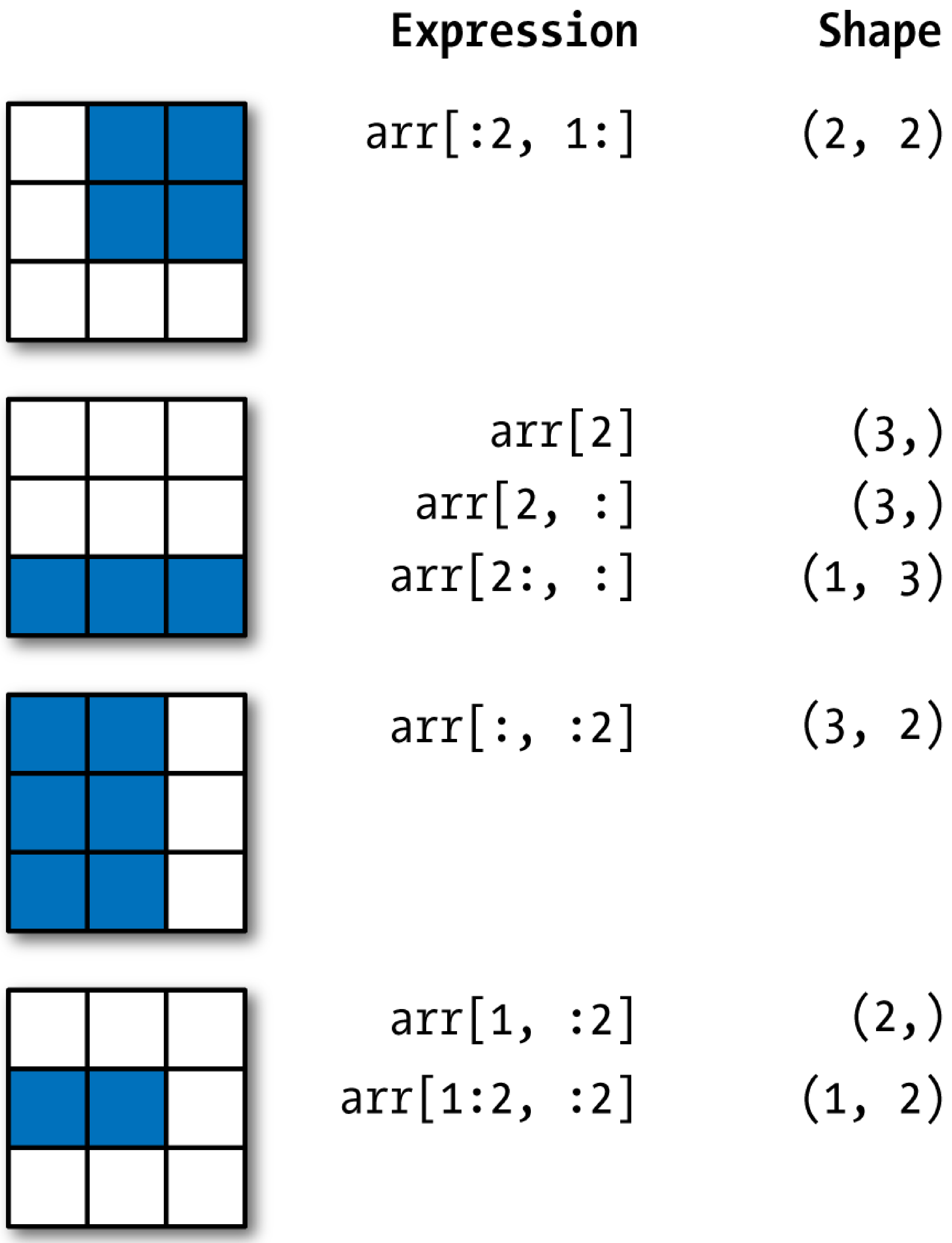


Expression	Shape
`arr[:2, 1:]`	(2, 2)
`arr[2]`	(3,)
`arr[2, :]`	(3,)
`arr[2:, :]`	(1, 3)
`arr[:, :2]`	(3, 2)

[W. McKinney, Python for Data Analysis]

2D Array Slicing

How to obtain the blue slice from array `arr`?



Expression	Shape
`arr[:2, 1:]`	`(2, 2)`
`arr[2]`	`(3,)`
`arr[2, :]`	`(3,)`
`arr[2:, :]`	`(1, 3)`
`arr[:, :2]`	`(3, 2)`
`arr[1, :2]`	`(2,)`
`arr[1:2, :2]`	`(1, 2)`

[W. McKinney, Python for Data Analysis]

Reshaping

- reshape:
  - `arr2.reshape(4,2) # returns new view`
- resize:
  - `arr2.resize(4,2) # no return, modifies arr2 in place`
- flatten:
  - `arr2.flatten() # array([1.5,2.,3.,4.,5.,6.,7.,8.])`
- ravel:
  - `arr2.ravel() # array([1.5,2.,3.,4.,5.,6.,7.,8.])`
- flatten and ravel look the same, but ravel is a **view**

Boolean Indexing

- `names == 'Bob'` gives back booleans that represent the element-wise comparison with the array `names`
- Boolean arrays can be used to index into another array:
  - `data[names == 'Bob']`
- Can even mix and match with integer slicing
- Can do boolean operations (`&`, `|`) between arrays (just like addition, subtraction)
  - `data[(names == 'Bob') | (names == 'Will')]`
- Note: `or` and `and` do not work with arrays
- We can set values too! `data[data < 0] = 0`

Array Transformations

- Transpose
  - `arr2.T # flip rows and columns`
- Stacking: take iterable of arrays and stack them horizontally/vertically
  - `arrh1 = np.arange(3)`
  - `arrh2 = np.arange(3,6)`
  - `np.vstack([arrh1, arrh2])`
  - `np.hstack([arr1.T, arr2.T]) # ???`

numpy Functions

- Unary: abs, sqrt, log, ceil, sin, cos, tan, arccos, arcsin, …
- Binary: add, subtract, multiple, divide, … <, >, >=, <=, ==, !=
- Statistics: sum, mean, std, min, max, argmin, argmax
- Boolean: any, all
- Others: sort, unique
- Linear Algebra (`numpy.linalg`)
- Pseudorandom Number Generation (`numpy.random`)

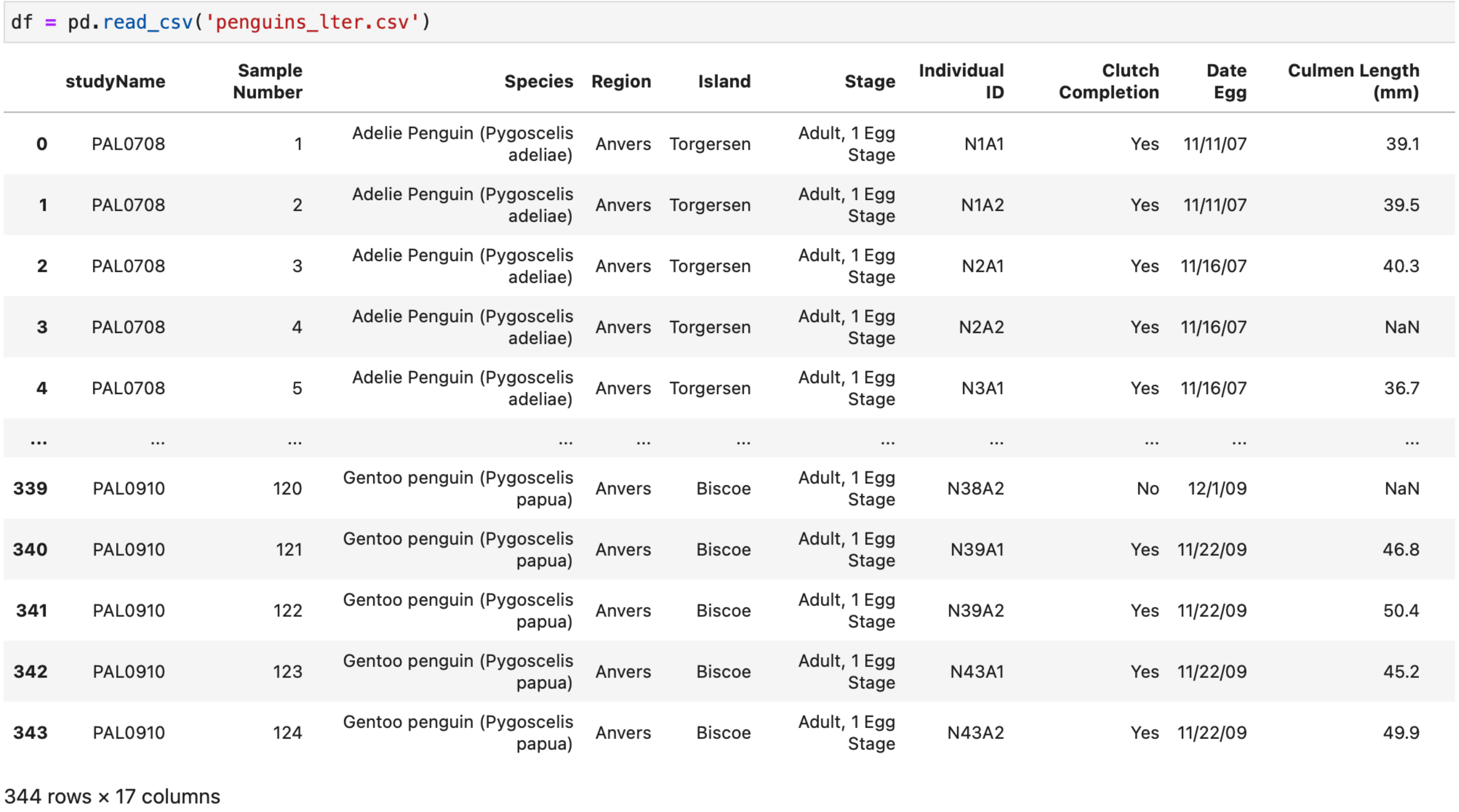
Dataframes

History of Dataframes

- Originally in *Statistical Models in S*, [J. M. Chambers & T. J. Hastie, 1992]
- R, open-source alternative to S, developed in 2000 (with dataframes)
- Pandas, 2009
- Spark, 2010 (resilient distributed dataset [RDD], Dataset API)
- Polars, 2020

[D. Petersohn, 2022]

pandas Dataframe

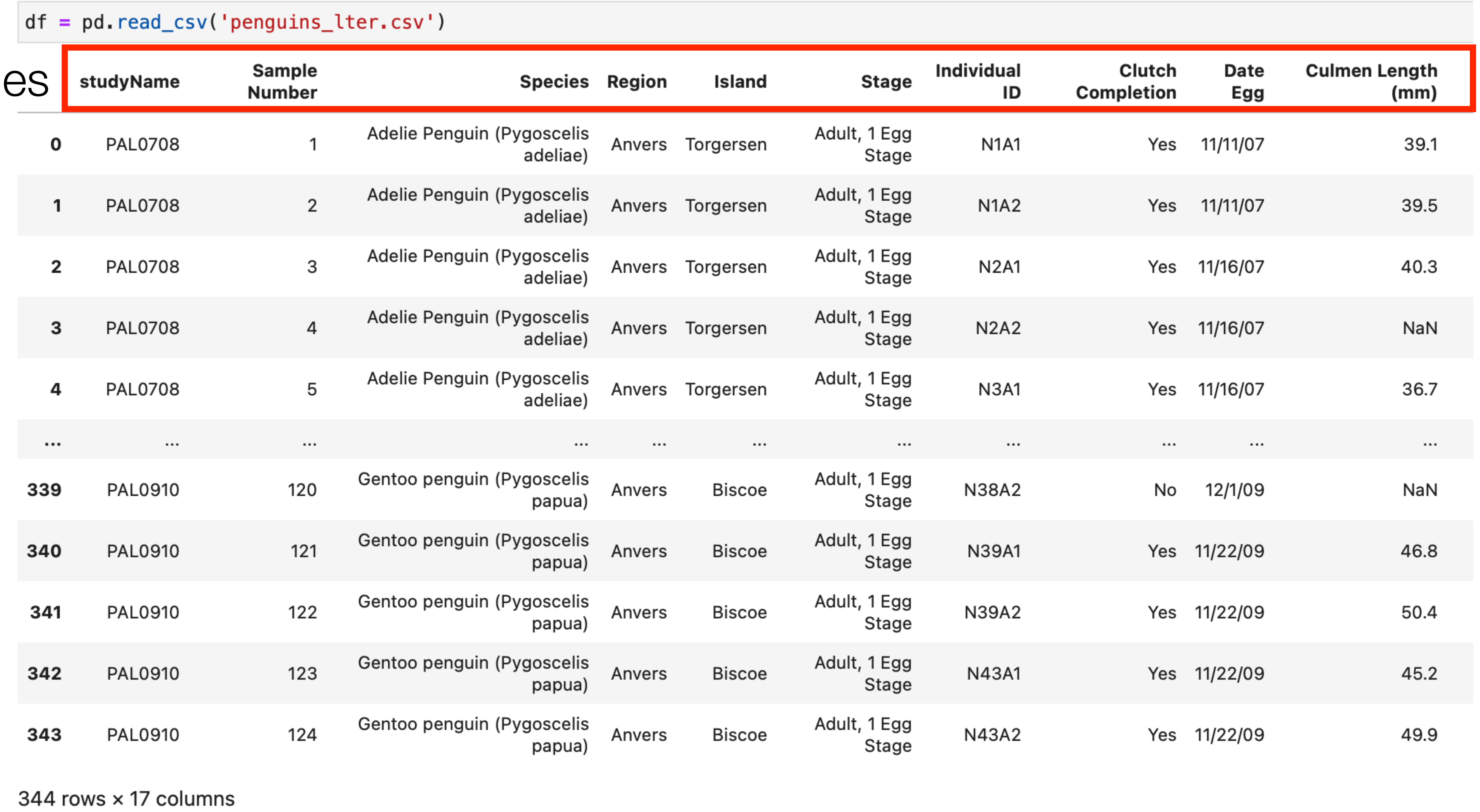


```
df = pd.read_csv('penguins_lter.csv')
```

	studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
0	PAL0708	1	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A1	Yes	11/11/07	39.1
1	PAL0708	2	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A2	Yes	11/11/07	39.5
2	PAL0708	3	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A1	Yes	11/16/07	40.3
3	PAL0708	4	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A2	Yes	11/16/07	NaN
4	PAL0708	5	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N3A1	Yes	11/16/07	36.7
...	...	...	...	...	...	...	...	...	...	...
339	PAL0910	120	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N38A2	No	12/1/09	NaN
340	PAL0910	121	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A1	Yes	11/22/09	46.8
341	PAL0910	122	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A2	Yes	11/22/09	50.4
342	PAL0910	123	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A1	Yes	11/22/09	45.2
343	PAL0910	124	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A2	Yes	11/22/09	49.9

344 rows × 17 columns

pandas Dataframe



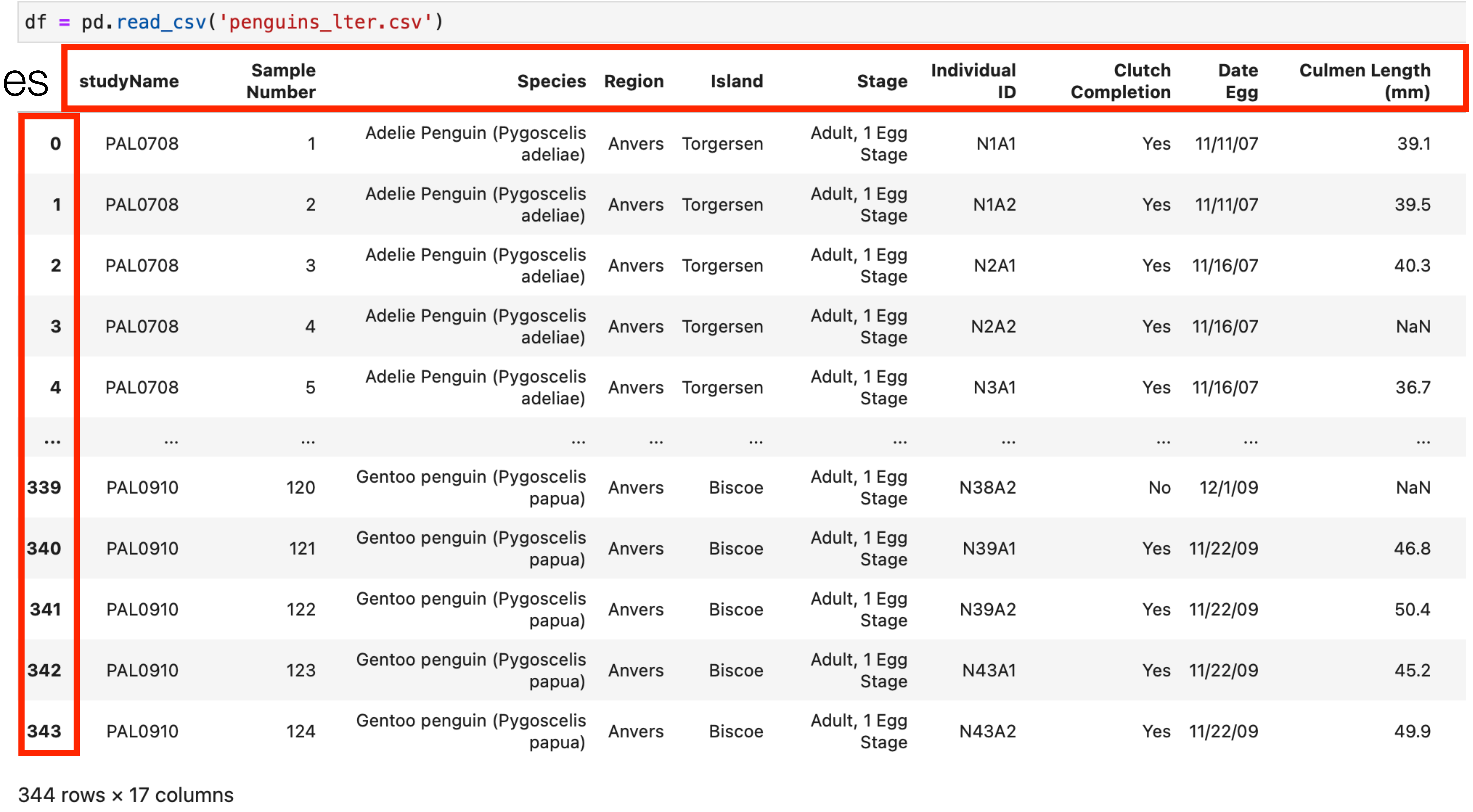
```
df = pd.read_csv('penguins_lter.csv')
```

Column Names

	studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
0	PAL0708	1	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A1	Yes	11/11/07	39.1
1	PAL0708	2	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A2	Yes	11/11/07	39.5
2	PAL0708	3	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A1	Yes	11/16/07	40.3
3	PAL0708	4	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A2	Yes	11/16/07	NaN
4	PAL0708	5	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N3A1	Yes	11/16/07	36.7
...	...	...	...	...	...	...	...	...	...	...
339	PAL0910	120	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N38A2	No	12/1/09	NaN
340	PAL0910	121	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A1	Yes	11/22/09	46.8
341	PAL0910	122	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A2	Yes	11/22/09	50.4
342	PAL0910	123	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A1	Yes	11/22/09	45.2
343	PAL0910	124	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A2	Yes	11/22/09	49.9

344 rows × 17 columns

pandas Dataframe



```
df = pd.read_csv('penguins_lter.csv')
```

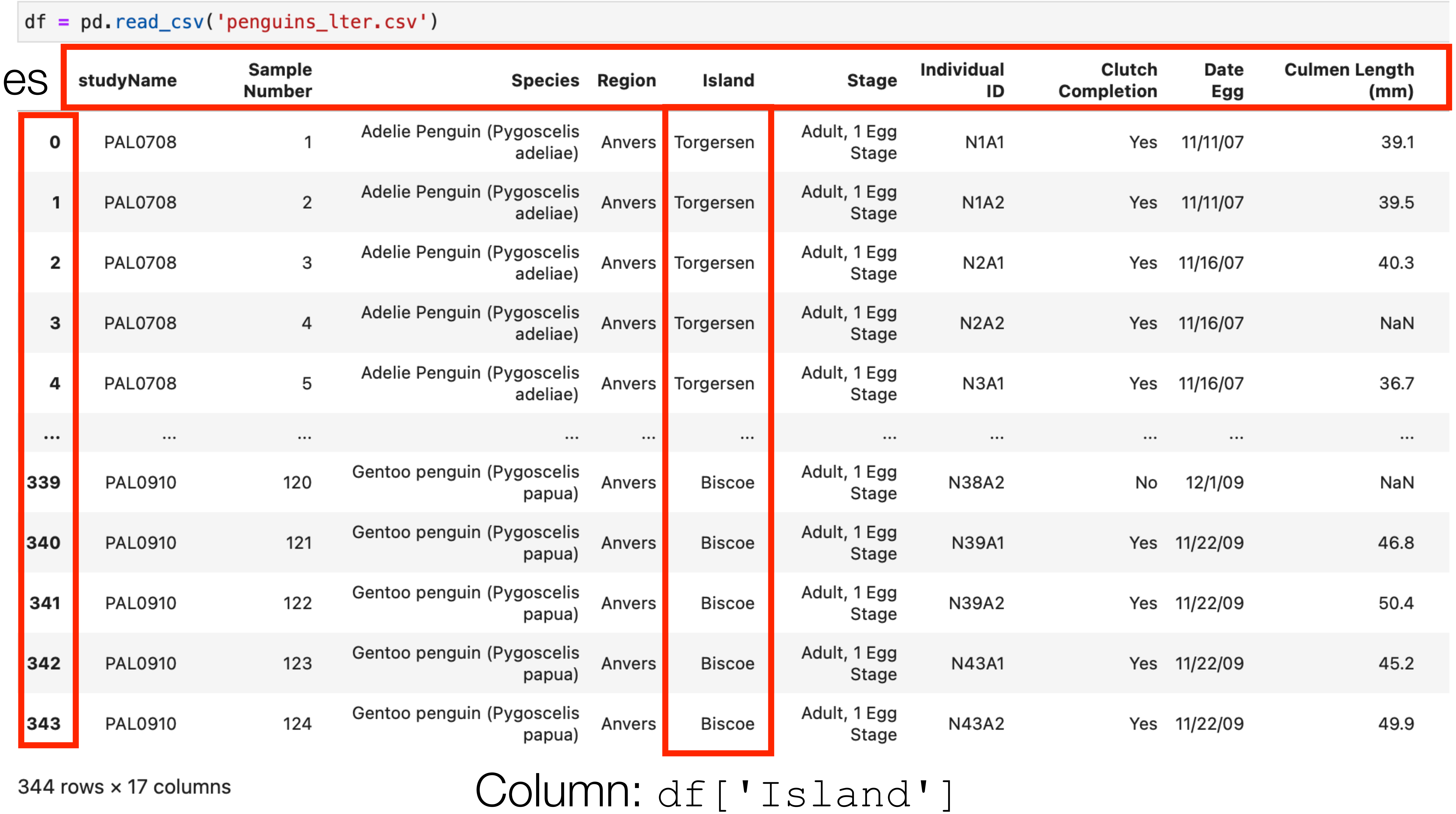
Column Names

Index

	studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
0	PAL0708	1	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A1	Yes	11/11/07	39.1
1	PAL0708	2	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A2	Yes	11/11/07	39.5
2	PAL0708	3	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A1	Yes	11/16/07	40.3
3	PAL0708	4	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A2	Yes	11/16/07	NaN
4	PAL0708	5	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N3A1	Yes	11/16/07	36.7
...	...	...	...	...	...	...	...	...	...	...
339	PAL0910	120	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N38A2	No	12/1/09	NaN
340	PAL0910	121	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A1	Yes	11/22/09	46.8
341	PAL0910	122	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A2	Yes	11/22/09	50.4
342	PAL0910	123	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A1	Yes	11/22/09	45.2
343	PAL0910	124	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A2	Yes	11/22/09	49.9

344 rows × 17 columns

pandas Dataframe



```
df = pd.read_csv('penguins_lter.csv')
```

Column Names

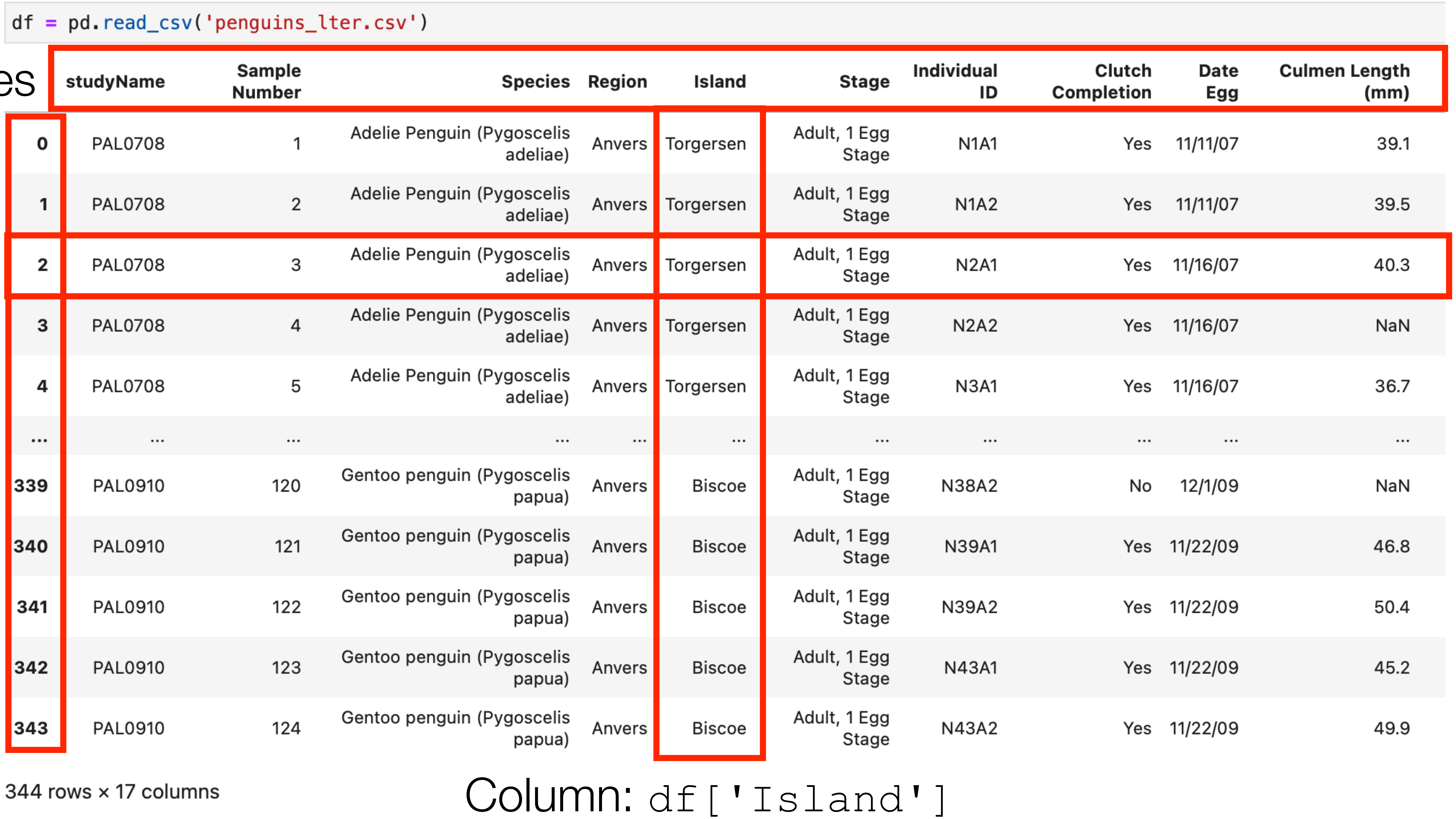
	studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
0	PAL0708	1	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A1	Yes	11/11/07	39.1
1	PAL0708	2	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A2	Yes	11/11/07	39.5
2	PAL0708	3	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A1	Yes	11/16/07	40.3
3	PAL0708	4	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A2	Yes	11/16/07	NaN
4	PAL0708	5	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N3A1	Yes	11/16/07	36.7
...	...	...	...	...	...	...	...	...	...	...
339	PAL0910	120	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N38A2	No	12/1/09	NaN
340	PAL0910	121	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A1	Yes	11/22/09	46.8
341	PAL0910	122	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A2	Yes	11/22/09	50.4
342	PAL0910	123	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A1	Yes	11/22/09	45.2
343	PAL0910	124	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A2	Yes	11/22/09	49.9

344 rows × 17 columns

Index

Column: `df['Island']`

pandas Dataframe



```
df = pd.read_csv('penguins_lter.csv')
```

	studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
0	PAL0708	1	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A1	Yes	11/11/07	39.1
1	PAL0708	2	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A2	Yes	11/11/07	39.5
2	PAL0708	3	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A1	Yes	11/16/07	40.3
3	PAL0708	4	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A2	Yes	11/16/07	NaN
4	PAL0708	5	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N3A1	Yes	11/16/07	36.7
...	...	...	...	...	...	...	...	...	...	...
339	PAL0910	120	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N38A2	No	12/1/09	NaN
340	PAL0910	121	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A1	Yes	11/22/09	46.8
341	PAL0910	122	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A2	Yes	11/22/09	50.4
342	PAL0910	123	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A1	Yes	11/22/09	45.2
343	PAL0910	124	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A2	Yes	11/22/09	49.9

344 rows × 17 columns

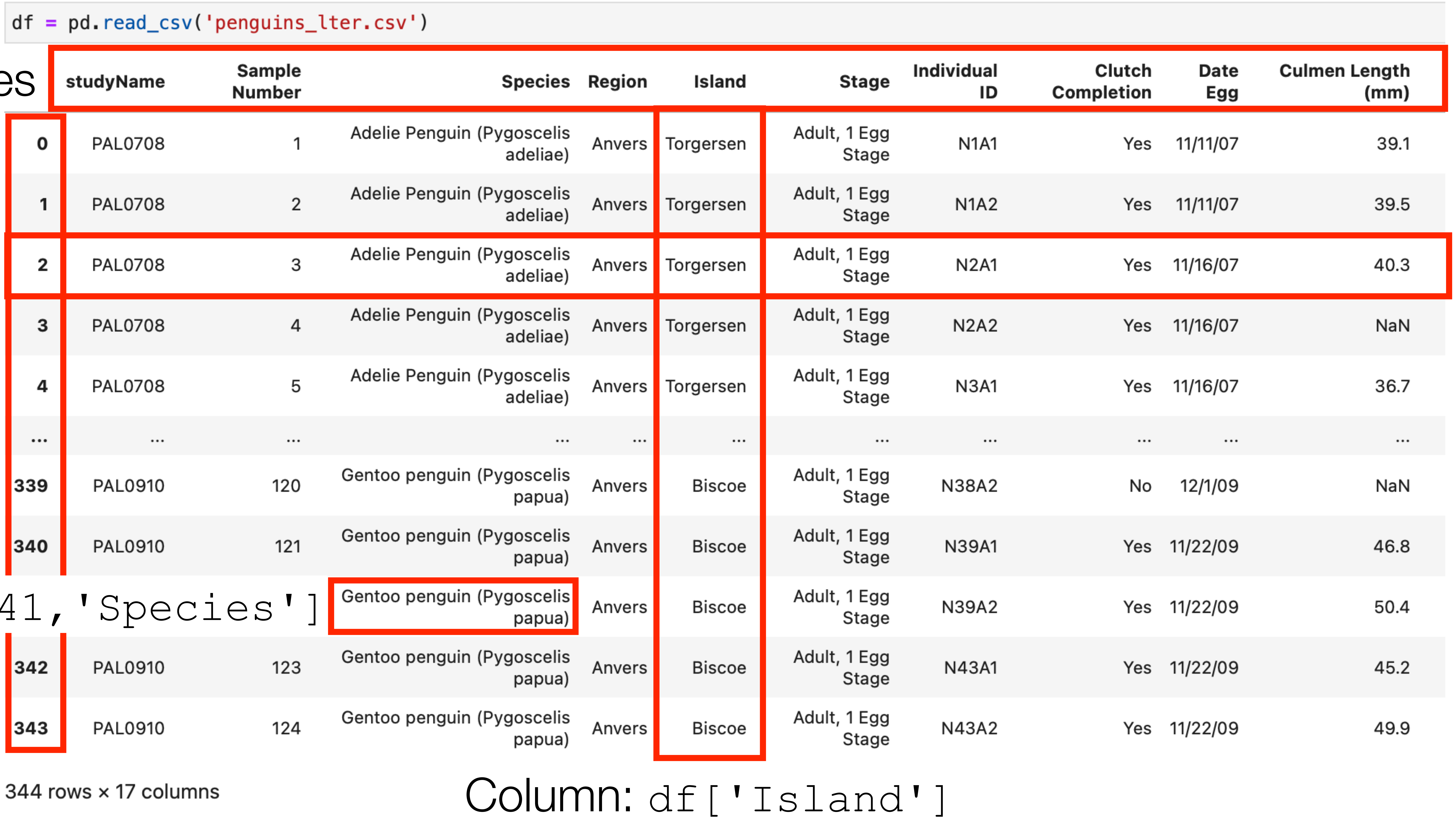
Column Names

Row: `df.loc[2]`

Index

Column: `df['Island']`

pandas Dataframe



```
df = pd.read_csv('penguins_lter.csv')
```

	studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
0	PAL0708	1	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A1	Yes	11/11/07	39.1
1	PAL0708	2	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A2	Yes	11/11/07	39.5
2	PAL0708	3	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A1	Yes	11/16/07	40.3
3	PAL0708	4	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A2	Yes	11/16/07	NaN
4	PAL0708	5	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N3A1	Yes	11/16/07	36.7
...	...	...	...	...	...	...	...	...	...	...
339	PAL0910	120	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N38A2	No	12/1/09	NaN
340	PAL0910	121	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A1	Yes	11/22/09	46.8
	PAL0910	122	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A2	Yes	11/22/09	50.4
342	PAL0910	123	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A1	Yes	11/22/09	45.2
343	PAL0910	124	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A2	Yes	11/22/09	49.9

344 rows × 17 columns

Column Names

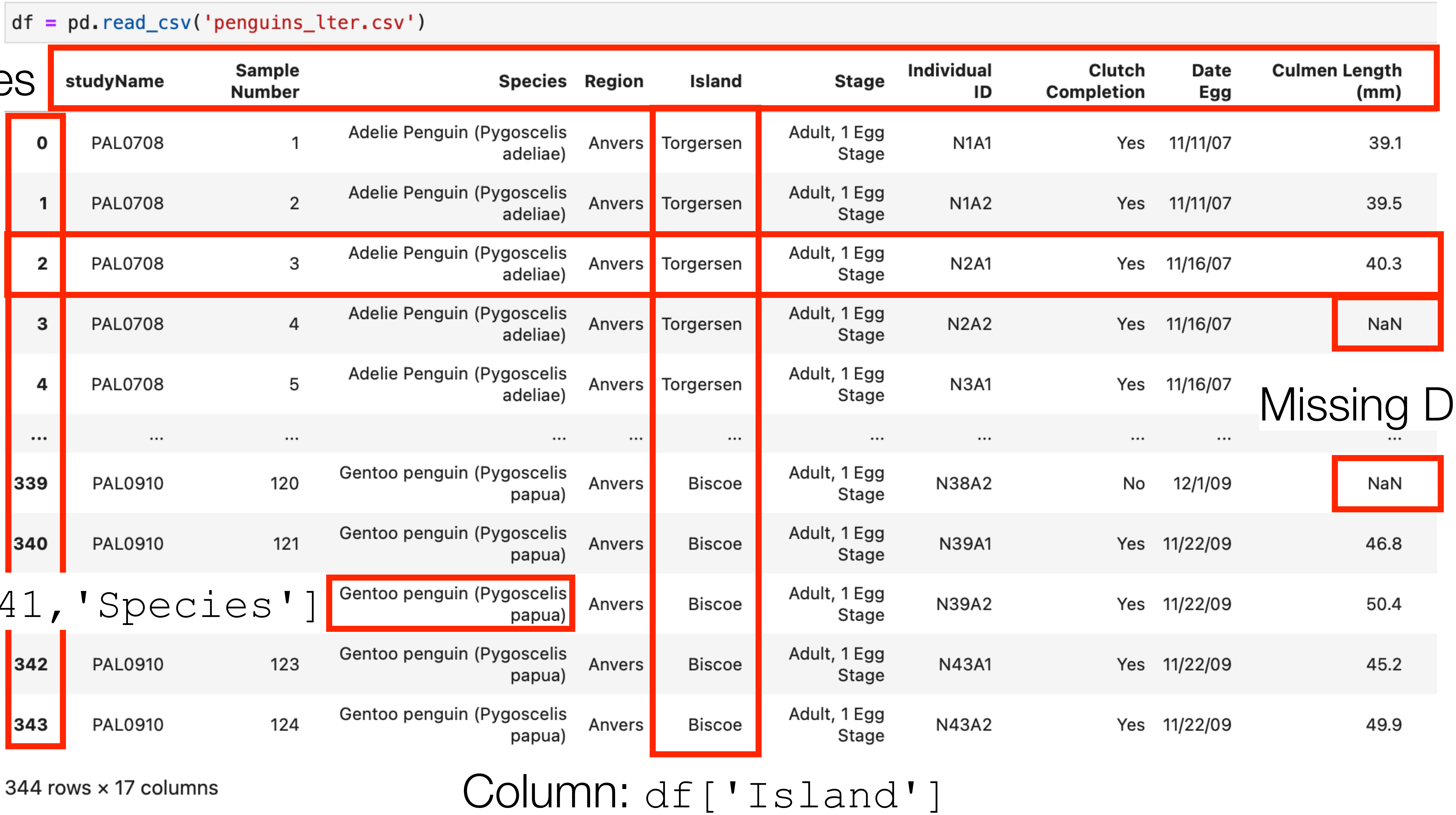
Row: `df.loc[2]`

Index

Cell: `df.loc[341,'Species']`

Column: `df['Island']`

pandas Dataframe



```
df = pd.read_csv('penguins_lter.csv')
```

	studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
0	PAL0708	1	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A1	Yes	11/11/07	39.1
1	PAL0708	2	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N1A2	Yes	11/11/07	39.5
2	PAL0708	3	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A1	Yes	11/16/07	40.3
3	PAL0708	4	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N2A2	Yes	11/16/07	NaN
4	PAL0708	5	Adelie Penguin (Pygoscelis adeliae)	Anvers	Torgersen	Adult, 1 Egg Stage	N3A1	Yes	11/16/07	...
...	...	...	...	...	...	...	...	...	...	...
339	PAL0910	120	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N38A2	No	12/1/09	NaN
340	PAL0910	121	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A1	Yes	11/22/09	46.8
	PAL0910	122	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N39A2	Yes	11/22/09	50.4
342	PAL0910	123	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A1	Yes	11/22/09	45.2
343	PAL0910	124	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N43A2	Yes	11/22/09	49.9

344 rows × 17 columns

Column Names

Row: `df.loc[2]`

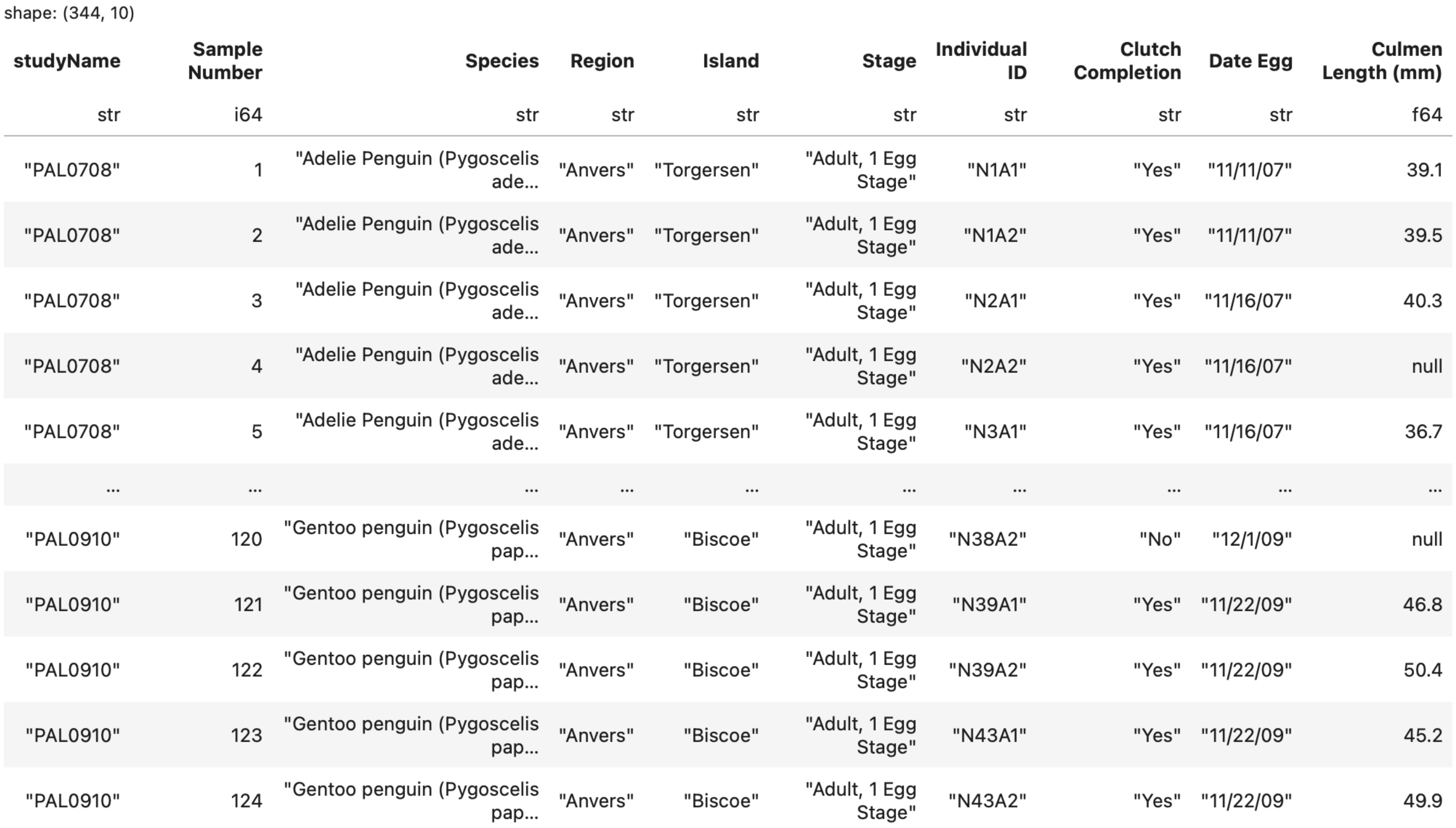
Index

Cell: `df.loc[341,'Species']`

Missing Data

Column: `df['Island']`

polars Dataframe

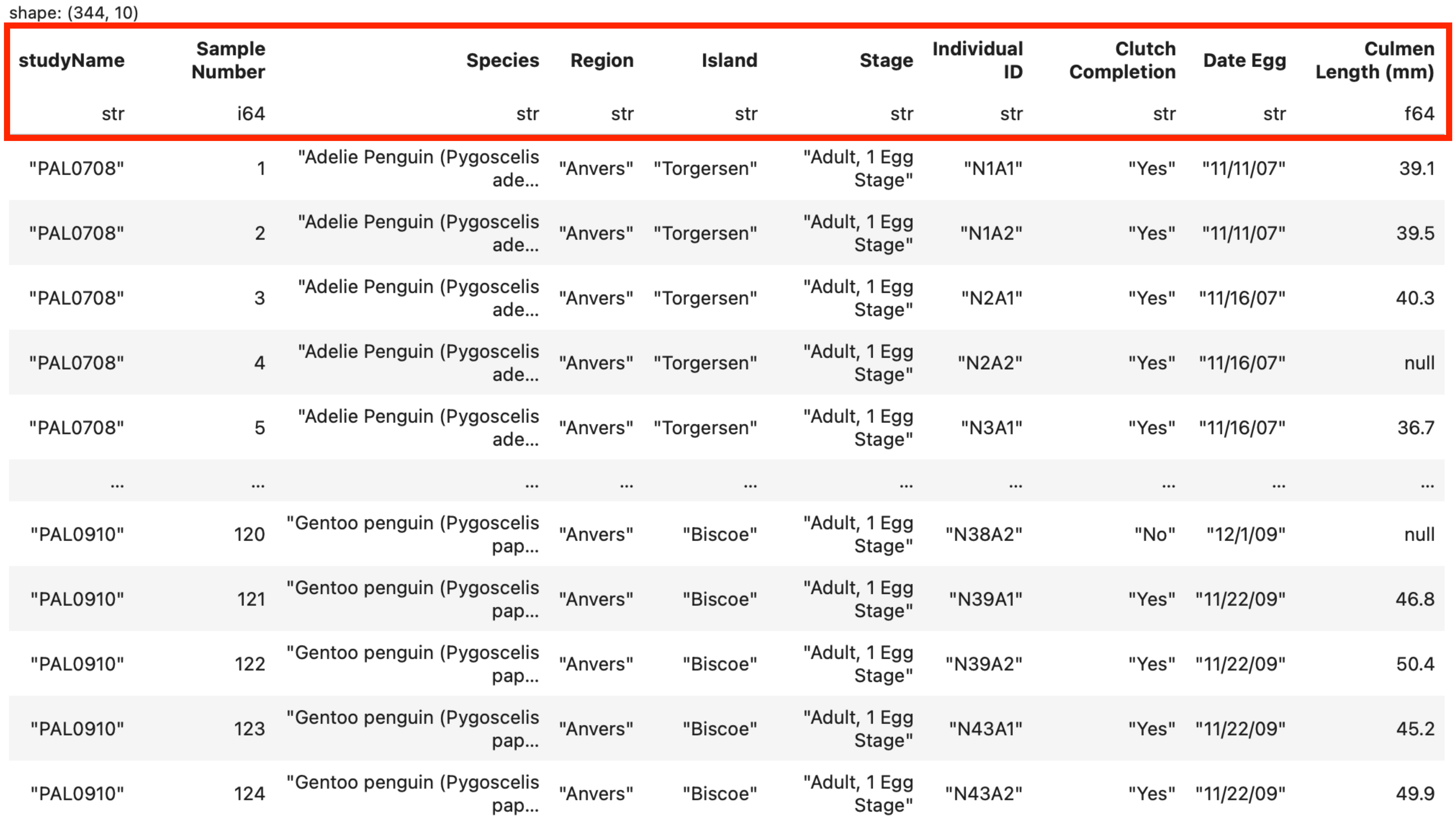


shape: (344, 10)

studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
str	i64	str	str	str	str	str	str	str	f64
"PAL0708"	1	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N1A1"	"Yes"	"11/11/07"	39.1
"PAL0708"	2	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N1A2"	"Yes"	"11/11/07"	39.5
"PAL0708"	3	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N2A1"	"Yes"	"11/16/07"	40.3
"PAL0708"	4	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N2A2"	"Yes"	"11/16/07"	null
"PAL0708"	5	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N3A1"	"Yes"	"11/16/07"	36.7
…	…	…	…	…	…	…	…	…	…
"PAL0910"	120	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N38A2"	"No"	"12/1/09"	null
"PAL0910"	121	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N39A1"	"Yes"	"11/22/09"	46.8
"PAL0910"	122	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N39A2"	"Yes"	"11/22/09"	50.4
"PAL0910"	123	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N43A1"	"Yes"	"11/22/09"	45.2
"PAL0910"	124	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N43A2"	"Yes"	"11/22/09"	49.9

polars Dataframe

Column Names & Types

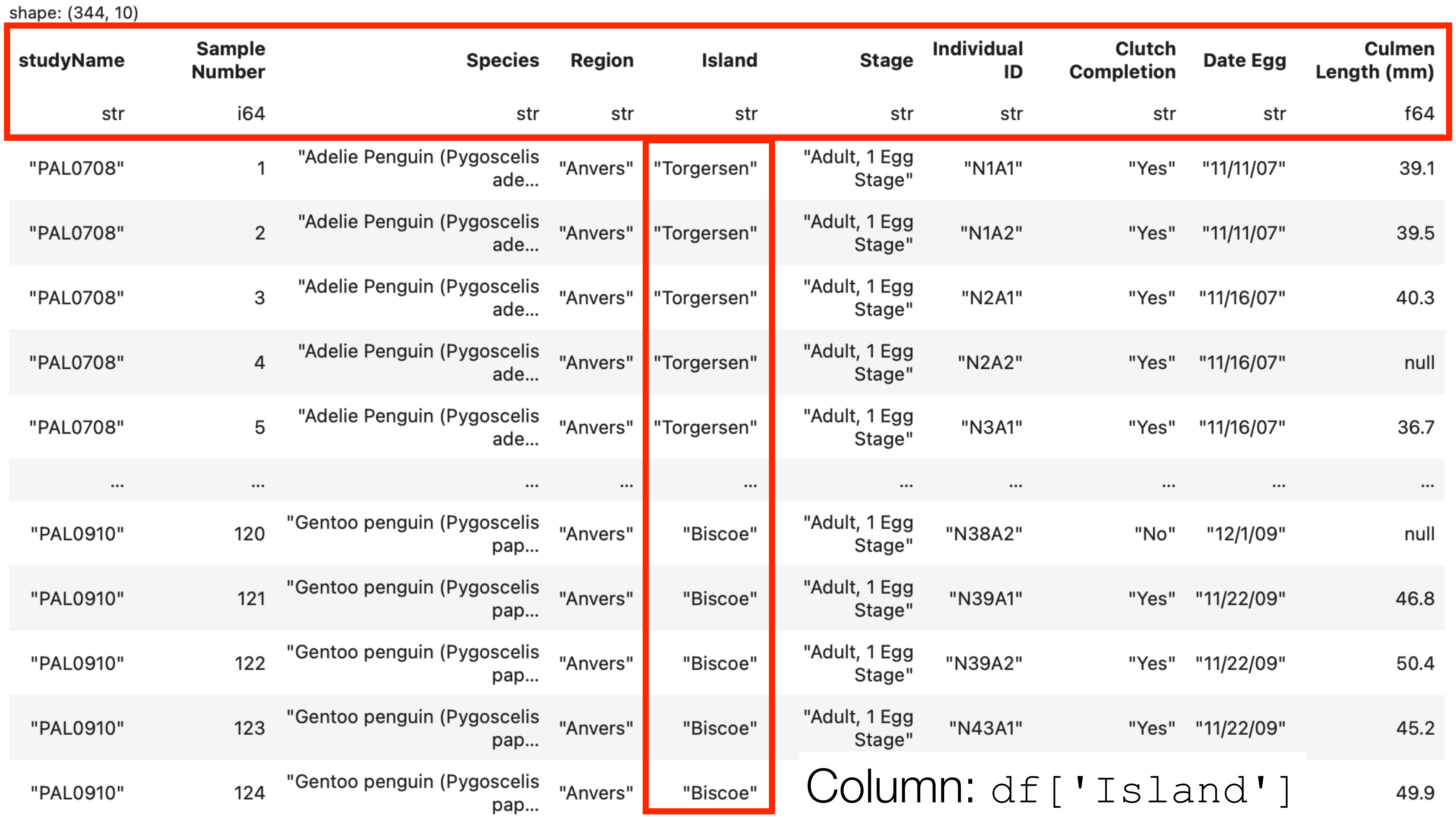


shape: (344, 10)

studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
str	i64	str	str	str	str	str	str	str	f64
"PAL0708"	1	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N1A1"	"Yes"	"11/11/07"	39.1
"PAL0708"	2	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N1A2"	"Yes"	"11/11/07"	39.5
"PAL0708"	3	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N2A1"	"Yes"	"11/16/07"	40.3
"PAL0708"	4	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N2A2"	"Yes"	"11/16/07"	null
"PAL0708"	5	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N3A1"	"Yes"	"11/16/07"	36.7
…	…	…	…	…	…	…	…	…	…
"PAL0910"	120	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N38A2"	"No"	"12/1/09"	null
"PAL0910"	121	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N39A1"	"Yes"	"11/22/09"	46.8
"PAL0910"	122	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N39A2"	"Yes"	"11/22/09"	50.4
"PAL0910"	123	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N43A1"	"Yes"	"11/22/09"	45.2
"PAL0910"	124	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N43A2"	"Yes"	"11/22/09"	49.9

polars Dataframe

Column Names & Types



shape: (344, 10)

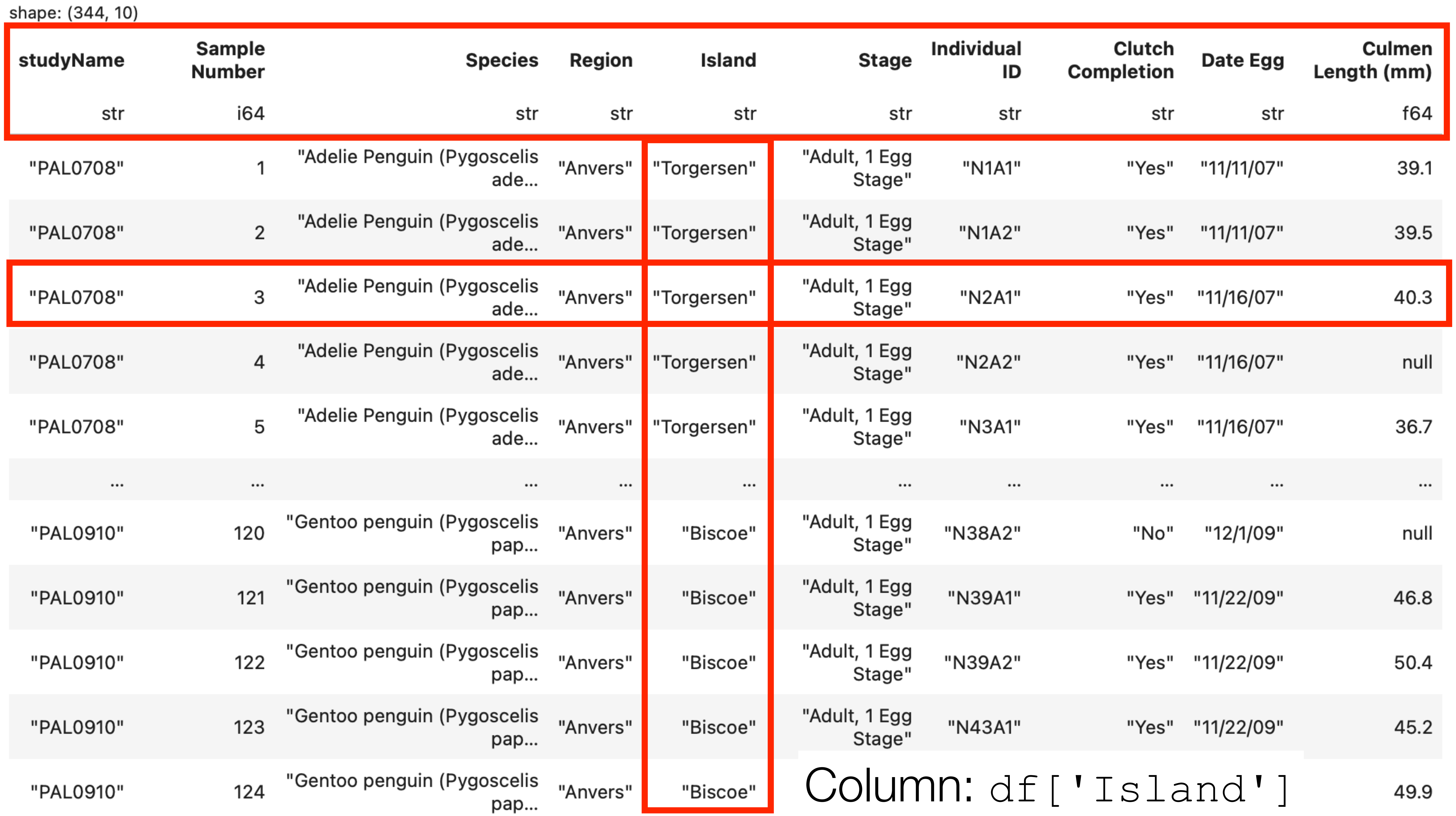
studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
str	i64	str	str	str	str	str	str	str	f64
"PAL0708"	1	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N1A1"	"Yes"	"11/11/07"	39.1
"PAL0708"	2	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N1A2"	"Yes"	"11/11/07"	39.5
"PAL0708"	3	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N2A1"	"Yes"	"11/16/07"	40.3
"PAL0708"	4	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N2A2"	"Yes"	"11/16/07"	null
"PAL0708"	5	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N3A1"	"Yes"	"11/16/07"	36.7
…	…	…	…	…	…	…	…	…	…
"PAL0910"	120	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N38A2"	"No"	"12/1/09"	null
"PAL0910"	121	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N39A1"	"Yes"	"11/22/09"	46.8
"PAL0910"	122	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N39A2"	"Yes"	"11/22/09"	50.4
"PAL0910"	123	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N43A1"	"Yes"	"11/22/09"	45.2
"PAL0910"	124	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"					49.9

Column: `df['Island']`

polars Dataframe

Column Names & Types

Row: `df[2]`



shape: (344, 10)

studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
str	i64	str	str	str	str	str	str	str	f64
"PAL0708"	1	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N1A1"	"Yes"	"11/11/07"	39.1
"PAL0708"	2	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N1A2"	"Yes"	"11/11/07"	39.5
"PAL0708"	3	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N2A1"	"Yes"	"11/16/07"	40.3
"PAL0708"	4	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N2A2"	"Yes"	"11/16/07"	null
"PAL0708"	5	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N3A1"	"Yes"	"11/16/07"	36.7
…	…	…	…	…	…	…	…	…	…
"PAL0910"	120	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N38A2"	"No"	"12/1/09"	null
"PAL0910"	121	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N39A1"	"Yes"	"11/22/09"	46.8
"PAL0910"	122	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N39A2"	"Yes"	"11/22/09"	50.4
"PAL0910"	123	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N43A1"	"Yes"	"11/22/09"	45.2
"PAL0910"	124	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"					49.9

Column: `df['Island']`

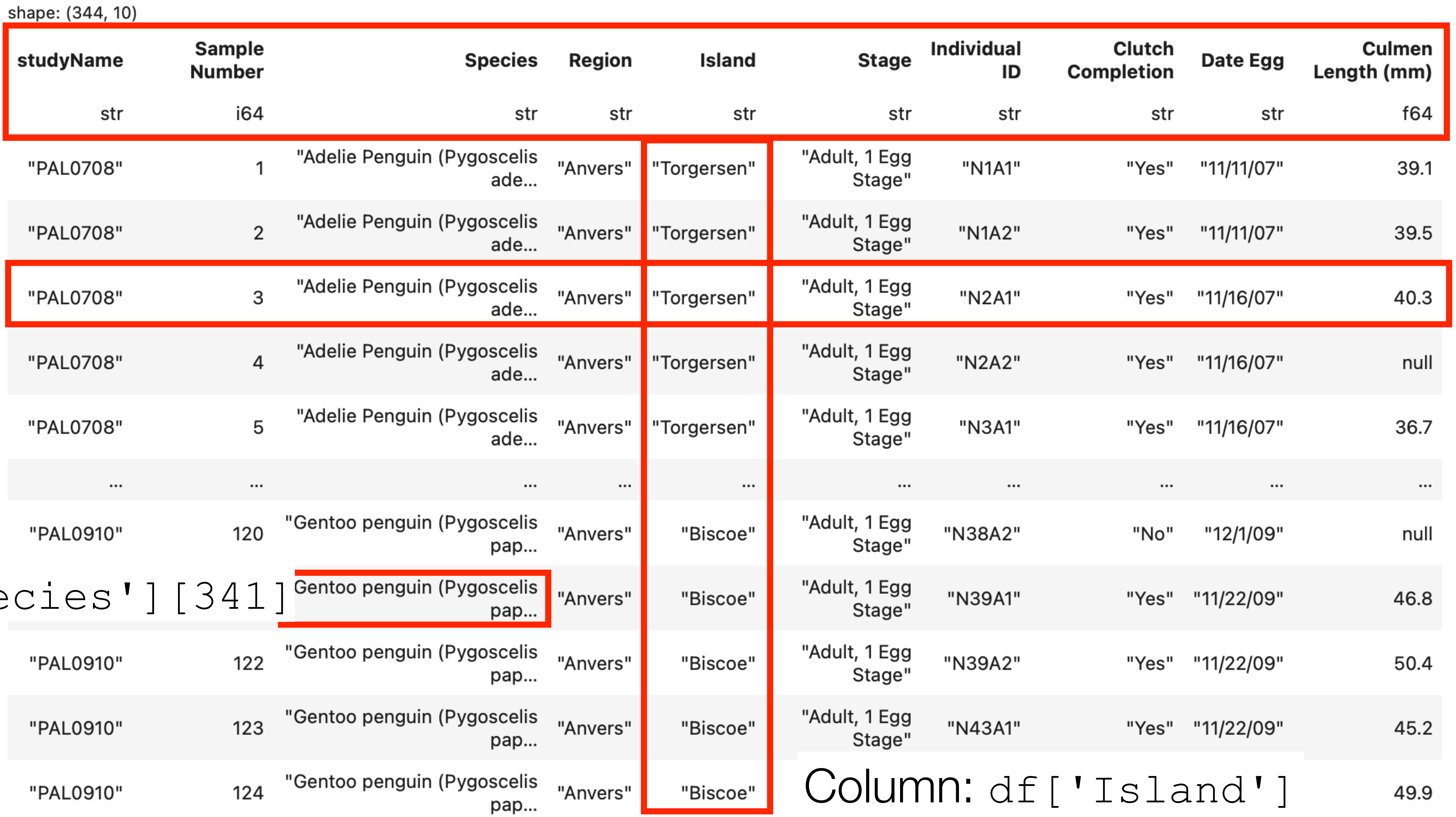
polars Dataframe

Column Names & Types

Row: `df[2]`

Cell: `df['Species'][341]`

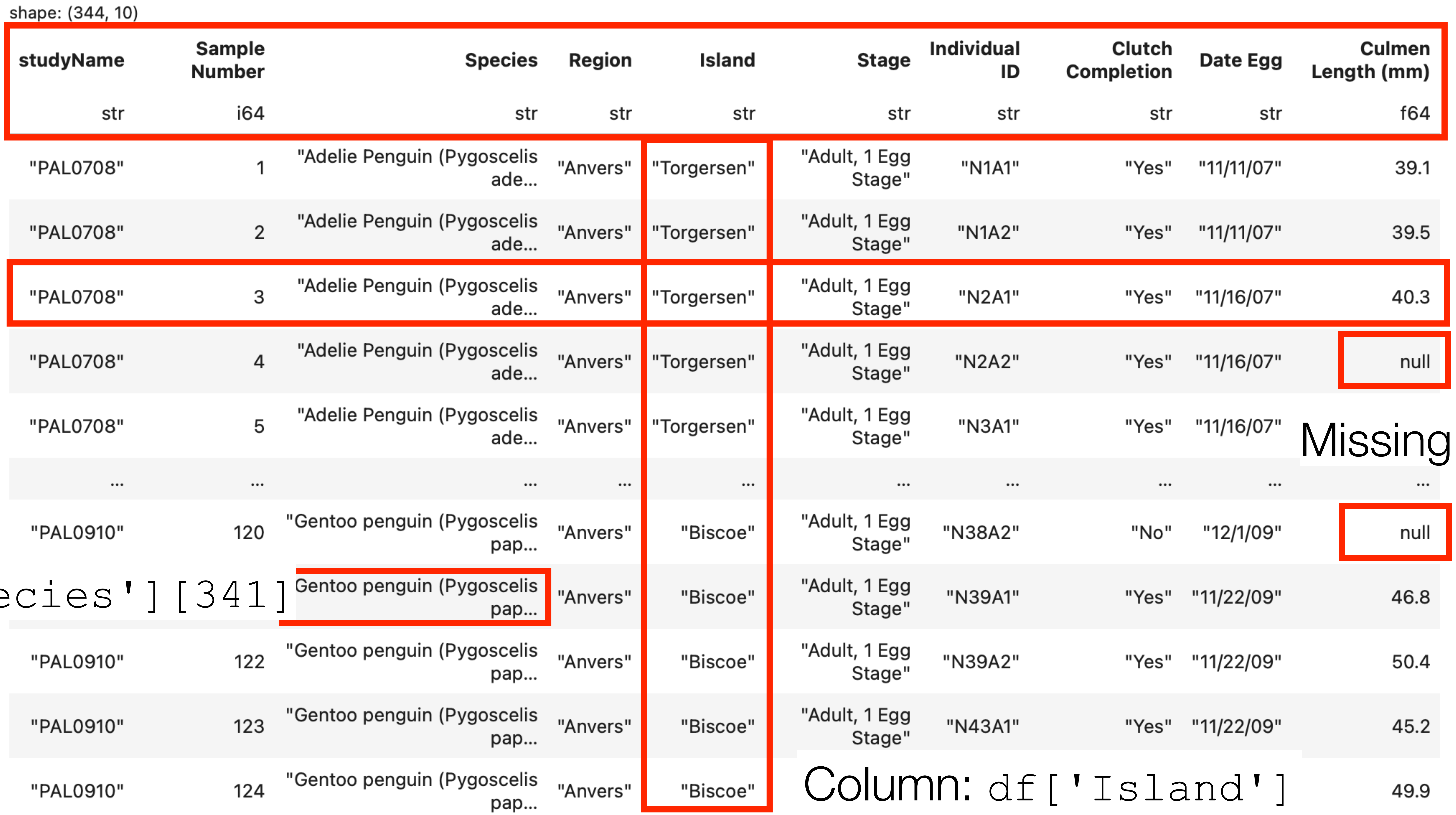
Column: `df['Island']`



shape: (344, 10)

studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
str	i64	str	str	str	str	str	str	str	f64
"PAL0708"	1	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N1A1"	"Yes"	"11/11/07"	39.1
"PAL0708"	2	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N1A2"	"Yes"	"11/11/07"	39.5
"PAL0708"	3	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N2A1"	"Yes"	"11/16/07"	40.3
"PAL0708"	4	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N2A2"	"Yes"	"11/16/07"	null
"PAL0708"	5	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N3A1"	"Yes"	"11/16/07"	36.7
…	…	…	…	…	…	…	…	…	…
"PAL0910"	120	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N38A2"	"No"	"12/1/09"	null
		Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N39A1"	"Yes"	"11/22/09"	46.8
"PAL0910"	122	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N39A2"	"Yes"	"11/22/09"	50.4
"PAL0910"	123	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N43A1"	"Yes"	"11/22/09"	45.2
"PAL0910"	124	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"					49.9

polars Dataframe



Column Names & Types

Row: `df[2]`

Cell: `df['Species'][341]`

Missing Data

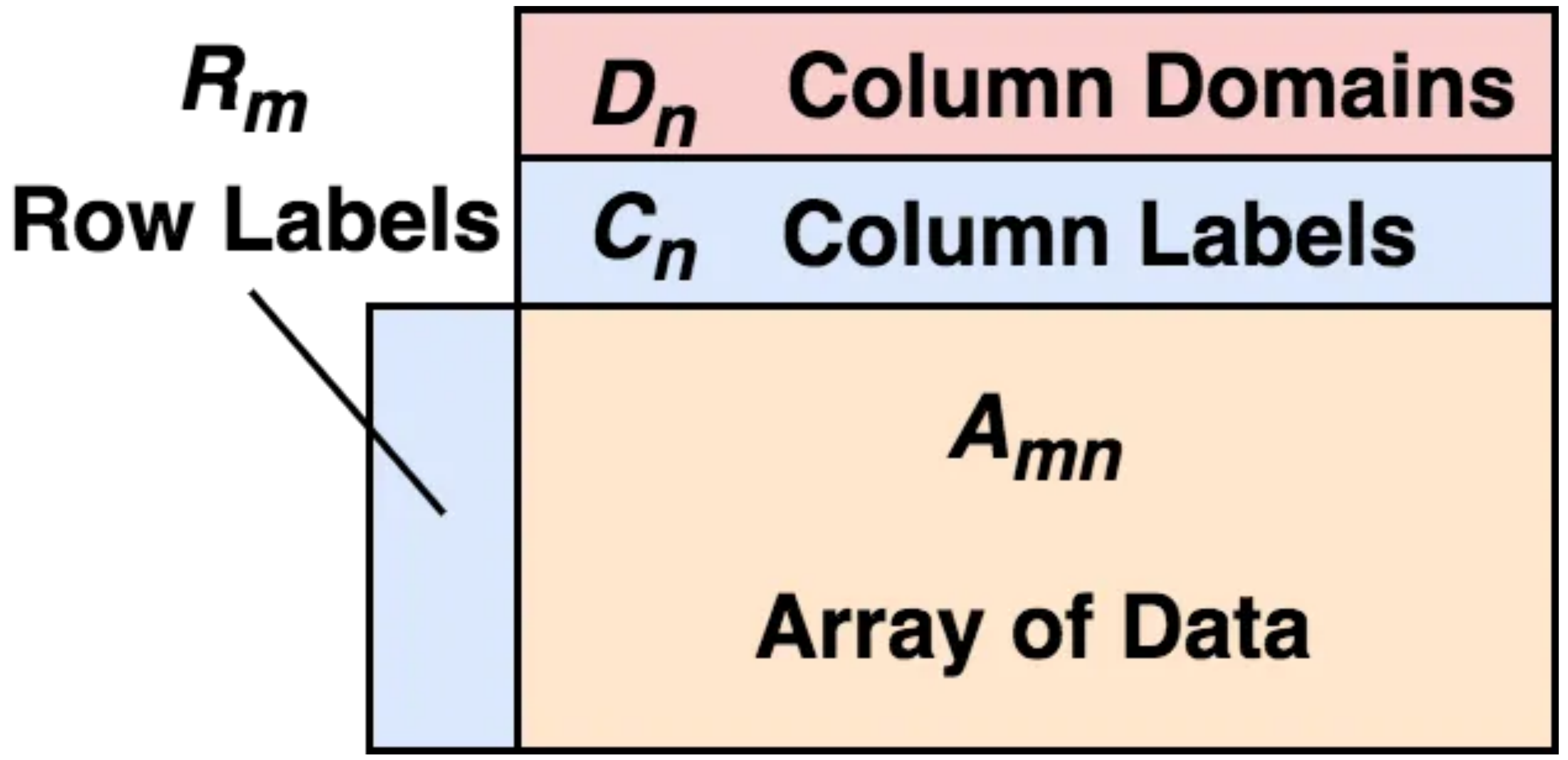
Column: `df['Island']`

shape: (344, 10)

studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Culmen Length (mm)
str	i64	str	str	str	str	str	str	str	f64
"PAL0708"	1	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N1A1"	"Yes"	"11/11/07"	39.1
"PAL0708"	2	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N1A2"	"Yes"	"11/11/07"	39.5
"PAL0708"	3	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N2A1"	"Yes"	"11/16/07"	40.3
"PAL0708"	4	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N2A2"	"Yes"	"11/16/07"	null
"PAL0708"	5	"Adelie Penguin (Pygoscelis ade…	"Anvers"	"Torgersen"	"Adult, 1 Egg Stage"	"N3A1"	"Yes"	"11/16/07"	
…	…	…	…	…	…	…	…	…	…
"PAL0910"	120	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N38A2"	"No"	"12/1/09"	null
		Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N39A1"	"Yes"	"11/22/09"	46.8
"PAL0910"	122	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N39A2"	"Yes"	"11/22/09"	50.4
"PAL0910"	123	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"	"Adult, 1 Egg Stage"	"N43A1"	"Yes"	"11/22/09"	45.2
"PAL0910"	124	"Gentoo penguin (Pygoscelis pap…	"Anvers"	"Biscoe"					49.9

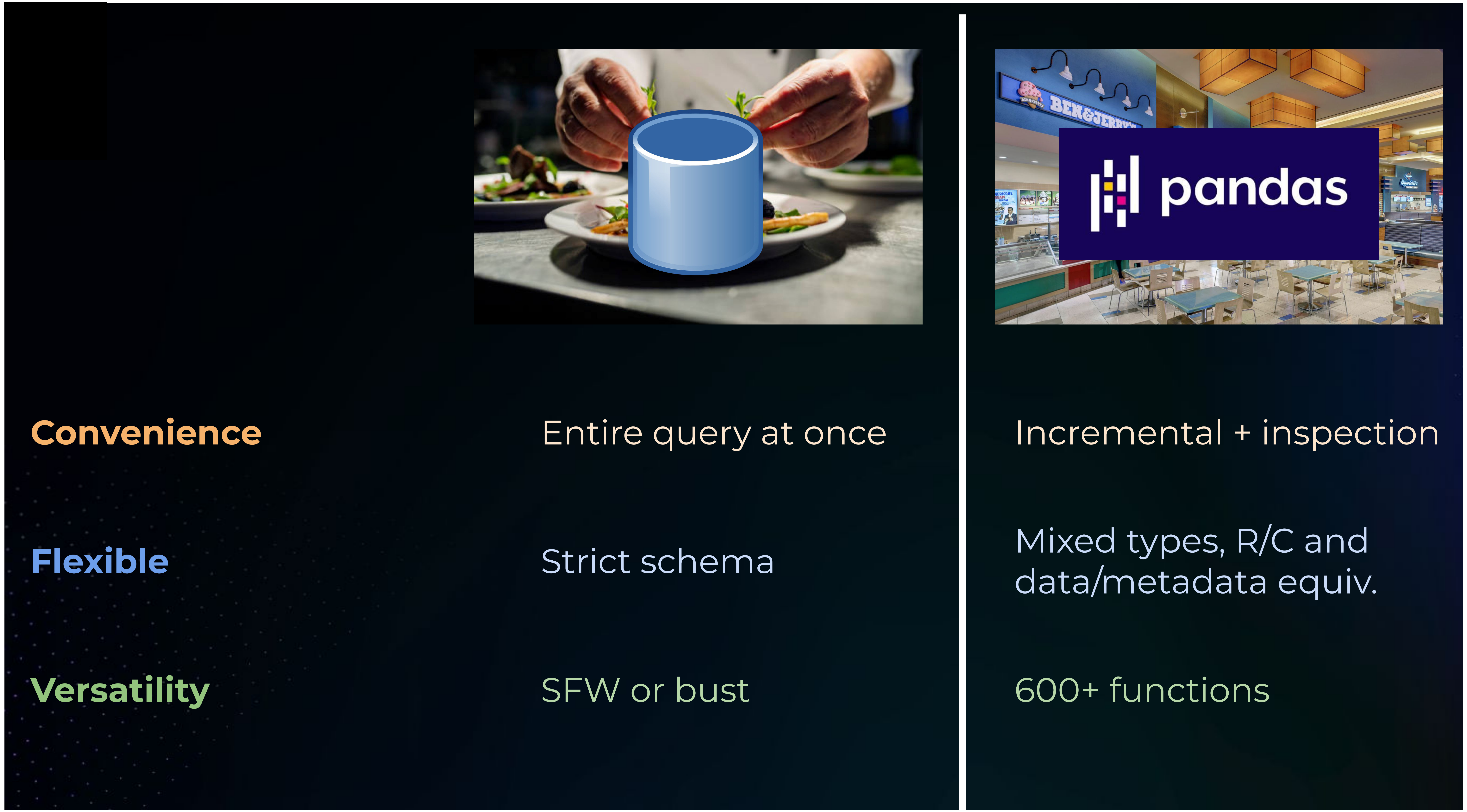
Formalizing Dataframes

- Combines parts of matrices, databases, and spreadsheets
- Ordered rows (unlike databases)
- Types can be inferred at runtime, not the same across all columns
- Lots of "intuitive" functions (600+)



[D. Petersohn, 2022]

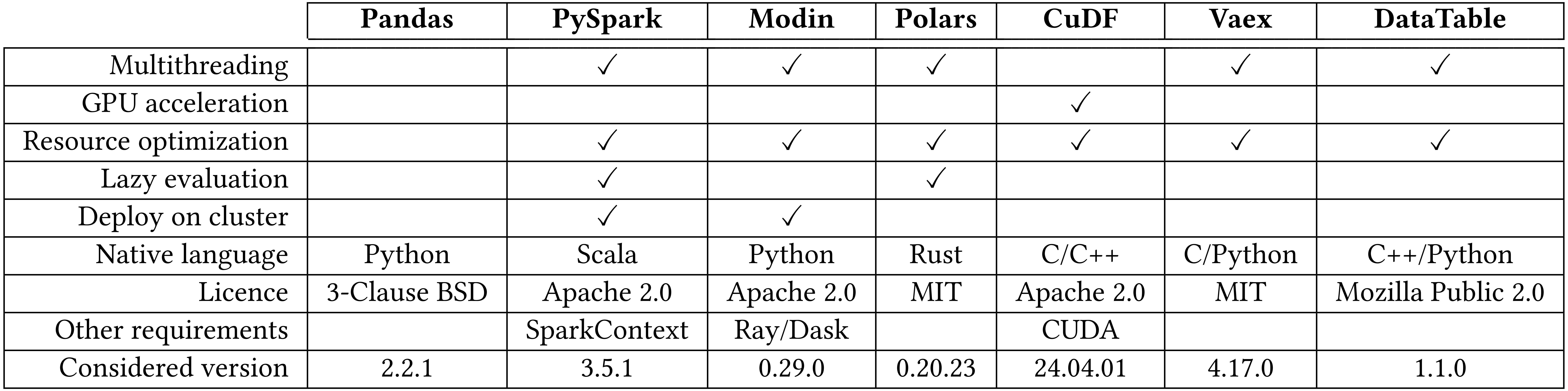
Differences between Databases & Dataframes



Convenience	Entire query at once	Incremental + inspection
Flexible	Strict schema	Mixed types, R/C and data/metadata equiv.
Versatility	SFW or bust	600+ functions

[D. Petersohn, 2022]

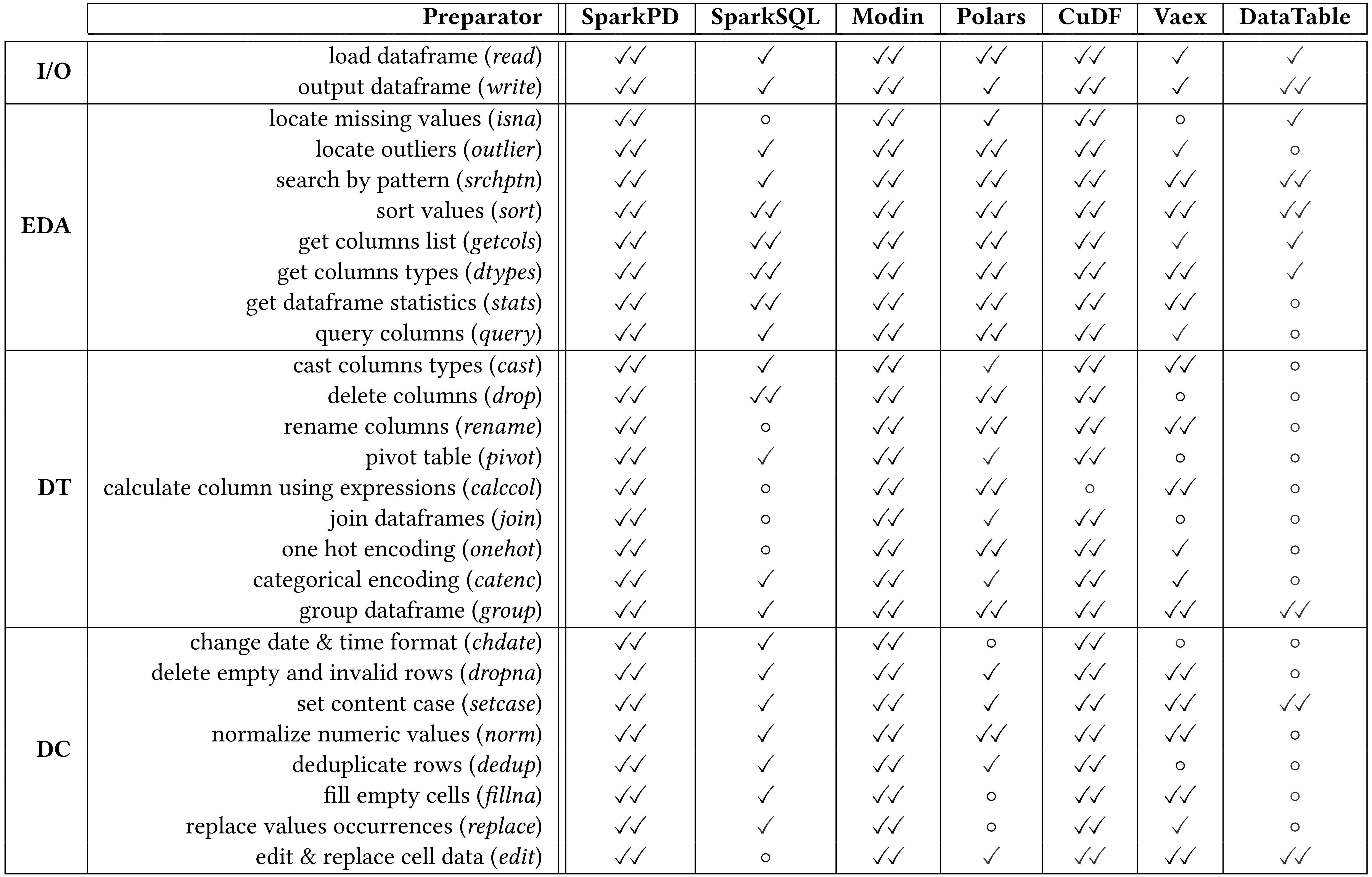
Dataframe Library Comparison



	Pandas	PySpark	Modin	Polars	CuDF	Vaex	DataTable
Multithreading		✓	✓	✓		✓	✓
GPU acceleration					✓		
Resource optimization		✓	✓	✓	✓	✓	✓
Lazy evaluation		✓		✓			
Deploy on cluster		✓	✓				
Native language	Python	Scala	Python	Rust	C/C++	C/Python	C++/Python
Licence	3-Clause BSD	Apache 2.0	Apache 2.0	MIT	Apache 2.0	MIT	Mozilla Public 2.0
Other requirements		SparkContext	Ray/Dask		CUDA		
Considered version	2.2.1	3.5.1	0.29.0	0.20.23	24.04.01	4.17.0	1.1.0

[A. Mozzillo et al., 2025]

Dataframe Library Operations



	Preparator	SparkPD	SparkSQL	Modin	Polars	CuDF	Vaex	DataTable
I/O	load dataframe (*read*)	✓✓	✓	✓✓	✓✓	✓✓	✓	✓
	output dataframe (*write*)	✓✓	✓	✓✓	✓	✓✓	✓	✓✓
EDA	locate missing values (*isna*)	✓✓	○	✓✓	✓	✓✓	○	✓
	locate outliers (*outlier*)	✓✓	✓	✓✓	✓✓	✓✓	✓	○
	search by pattern (*srchptn*)	✓✓	✓	✓✓	✓✓	✓✓	✓✓	✓✓
	sort values (*sort*)	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓
	get columns list (*getcols*)	✓✓	✓✓	✓✓	✓✓	✓✓	✓	✓
	get columns types (*dtypes*)	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓
	get dataframe statistics (*stats*)	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	○
	query columns (*query*)	✓✓	✓	✓✓	✓✓	✓✓	✓	○
DT	cast columns types (*cast*)	✓✓	✓	✓✓	✓	✓✓	✓✓	○
	delete columns (*drop*)	✓✓	✓✓	✓✓	✓✓	✓✓	○	○
	rename columns (*rename*)	✓✓	○	✓✓	✓✓	✓✓	✓✓	○
	pivot table (*pivot*)	✓✓	✓	✓✓	✓	✓✓	○	○
	calculate column using expressions (*calccol*)	✓✓	○	✓✓	✓✓	○	✓✓	○
	join dataframes (*join*)	✓✓	○	✓✓	✓	✓✓	○	○
	one hot encoding (*onehot*)	✓✓	○	✓✓	✓✓	✓✓	✓	○
	categorical encoding (*catenc*)	✓✓	✓	✓✓	✓	✓✓	✓	○
	group dataframe (*group*)	✓✓	✓	✓✓	✓✓	✓✓	✓✓	✓✓
DC	change date & time format (*chdate*)	✓✓	✓	✓✓	○	✓✓	○	○
	delete empty and invalid rows (*dropna*)	✓✓	✓	✓✓	✓	✓✓	✓✓	○
	set content case (*setcase*)	✓✓	✓	✓✓	✓	✓✓	✓✓	✓✓
	normalize numeric values (*norm*)	✓✓	✓	✓✓	✓✓	✓✓	✓✓	○
	deduplicate rows (*dedup*)	✓✓	✓	✓✓	✓	✓✓	○	○
	fill empty cells (*fillna*)	✓✓	✓	✓✓	○	✓✓	✓✓	○
	replace values occurrences (*replace*)	✓✓	✓	✓✓	○	✓✓	✓	○
	edit & replace cell data (*edit*)	✓✓	○	✓✓	✓	✓✓	✓✓	✓✓

√√ pandas compatible

√ Different interface

○ Missing from API

[A. Mozzillo et al., 2025]