Understanding Join

Data Engineering

Why Join?

- Data Enrichment
- Data Cleaning
- Data Analysis
- Data Transformation

Customer

ID	Name	State
1232	Sara Riazi	Illinois
1240	Jeff Richards	Arizona

Purchase History

ID	Date	ItemName	Value
1232	09/01/23	MacBookPro	$1422
1232	08/29/23	Wireless Microphone	$230

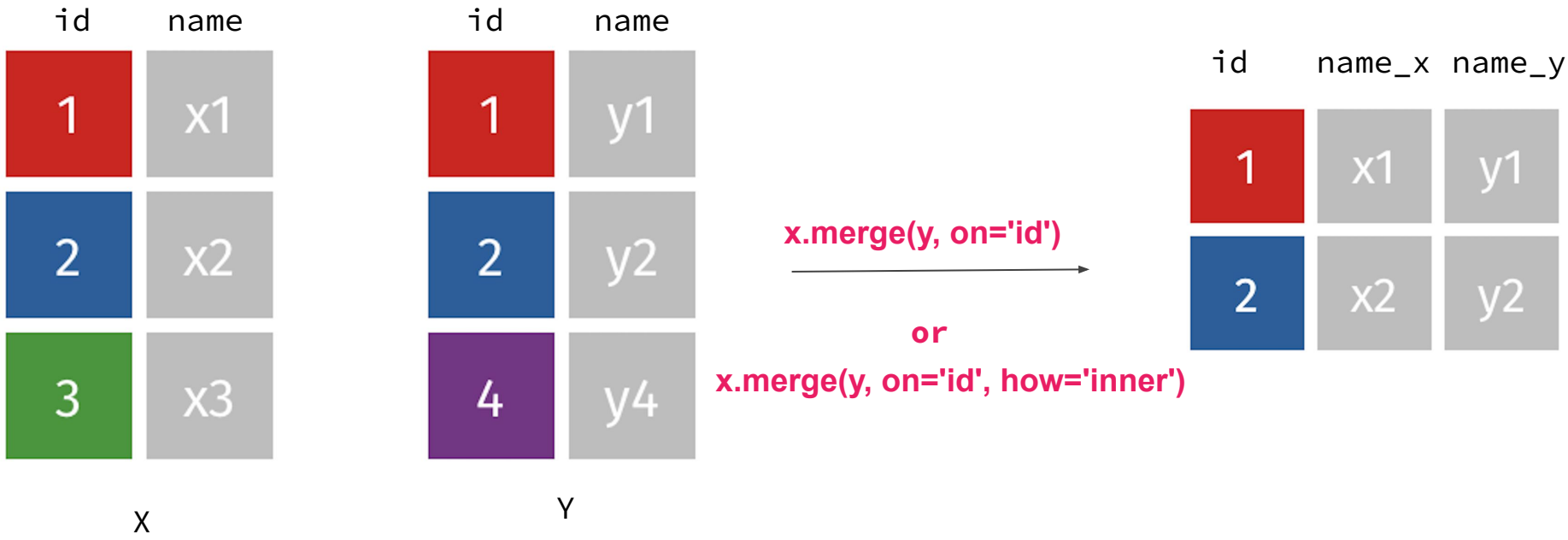
Using .merge() to join two dataframes

```
pd.merge(left, right, how='inner', on=None, left_on=None, right_on=None,suffixes=('_x', '_y'),)
```

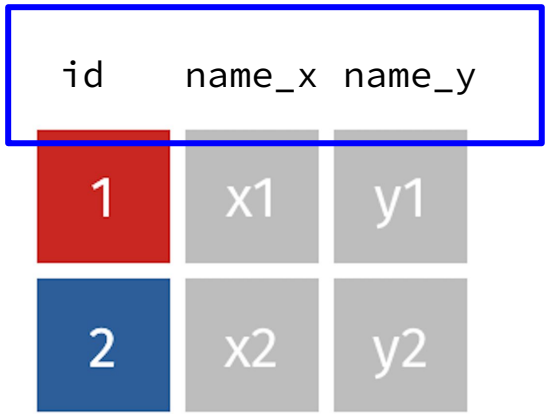
Key Arguments:

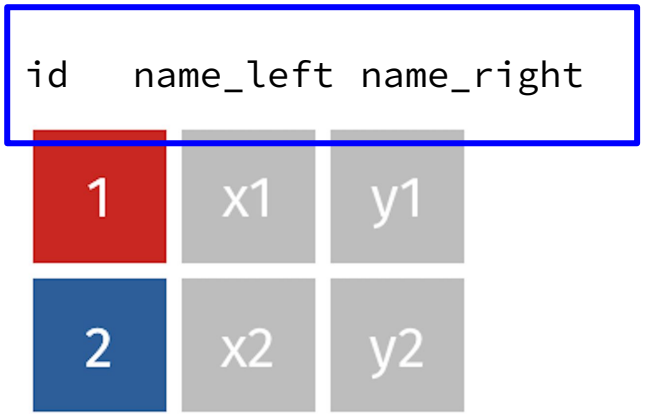
- left, right: The DataFrames you wish to merge.
- how: The type of join to be performed ('left', 'right', 'outer', 'inner').
- on: The column or columns that should be matched to join the DataFrames.
- left_on, right_on: Columns from the left and right DataFrames to use as keys.
- suffixes: Suffix to apply to overlapping column names.

Joining two DataFrames using Merge



Naming the Common Columns

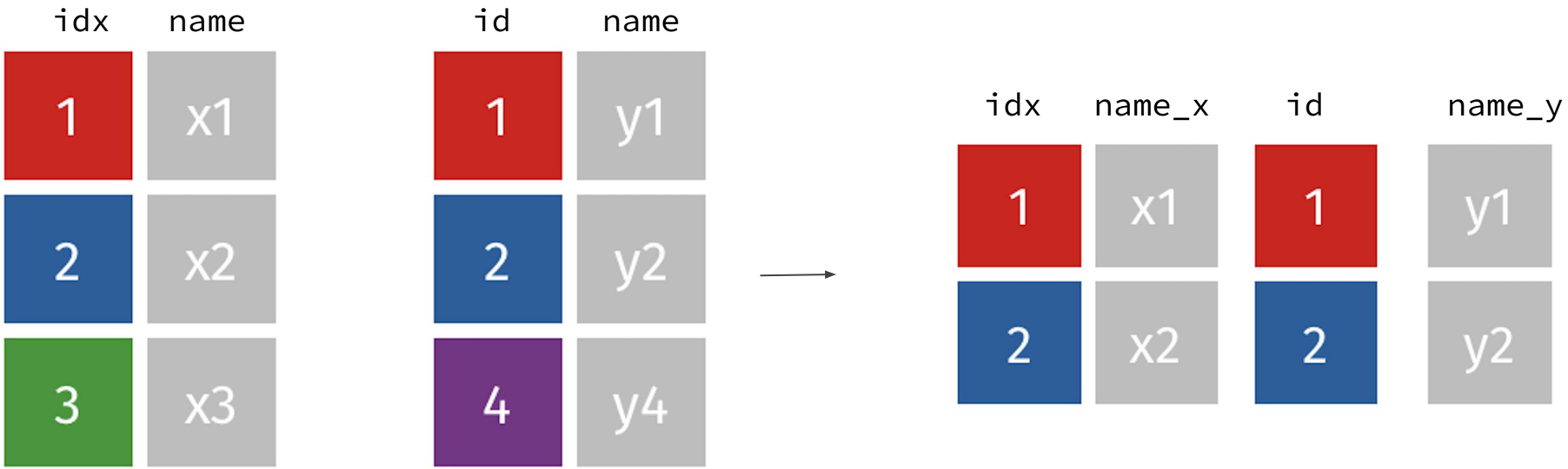




x.merge(y, on='id', how='inner')

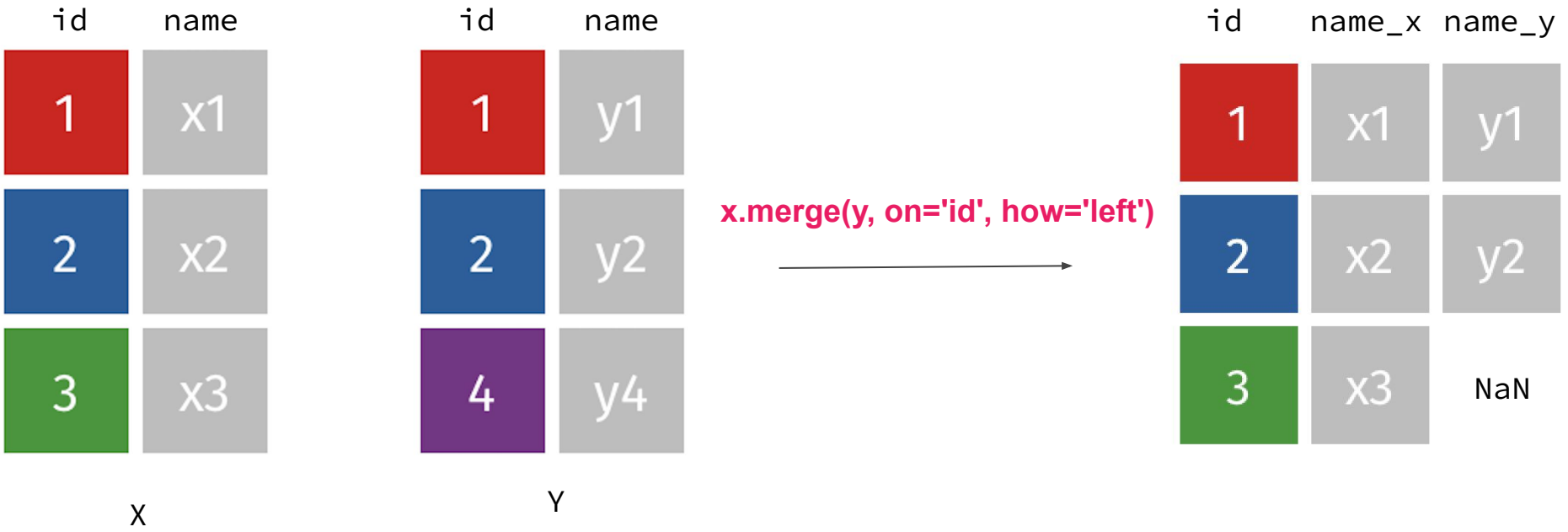
x.merge(y, on='id', how='inner', suffixes=('_left', '_right'))

Merging on Different Columns




```
x.merge(y, left_on='idx', right_on='id', how='inner')
```

Left Join

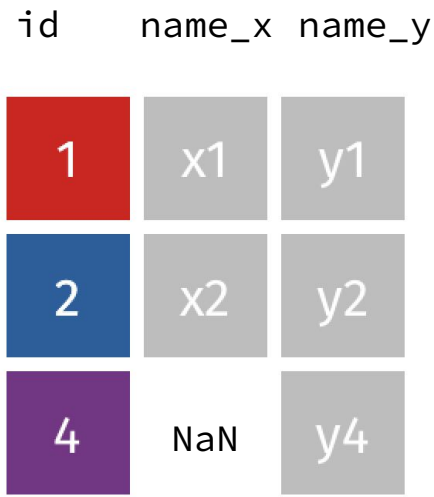

id
name
1
x1
2
x2
3
x3
X
id
name
1
y1
2
y2
4
y4
Y
x.merge(y, on='id', how='left')
id
name_x
name_y
1
x1
y1
2
x2
y2
3
x3
NaN

Right Join

X

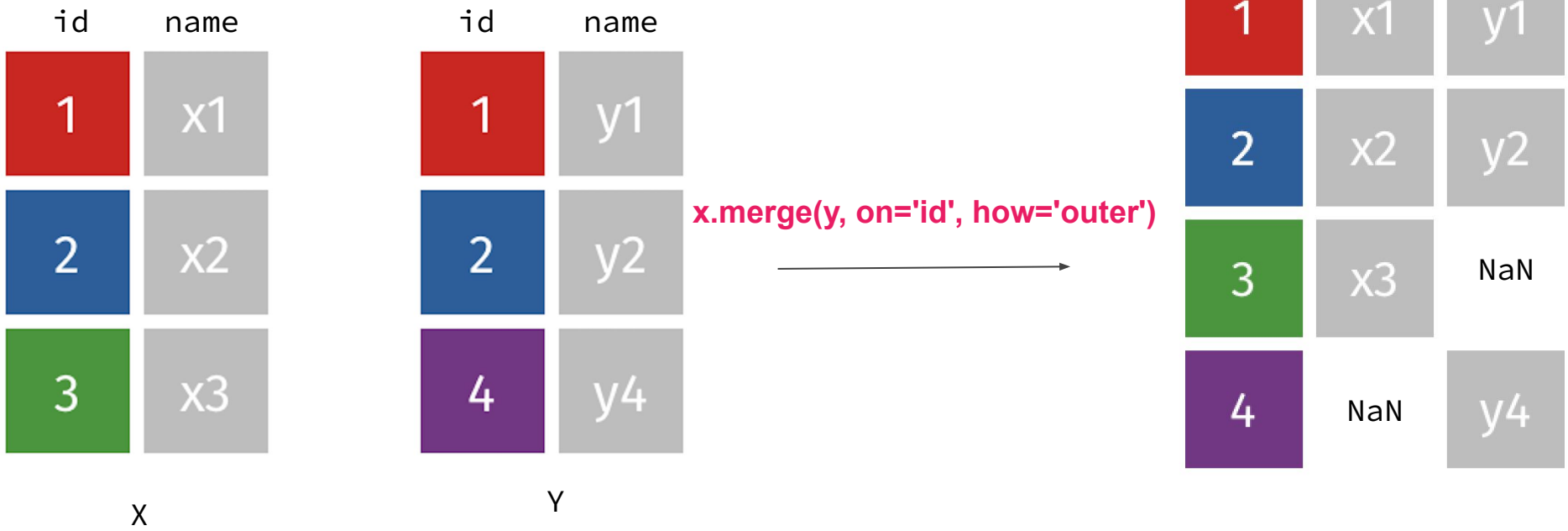
Y

x.merge(y, on='id', how='right')



x.merge(y, on='id', how='right') = y.merge(x, on='id', how='left', suffixes=('_y', '_x'))

Outer Join or Full Outer Join



Outer Join Example

`students:`

	StudentID	Name	CourseID
0	1	Alice	101
1	2	Bob	102
2	3	Charlie	103
3	4	David	104

`courses:`

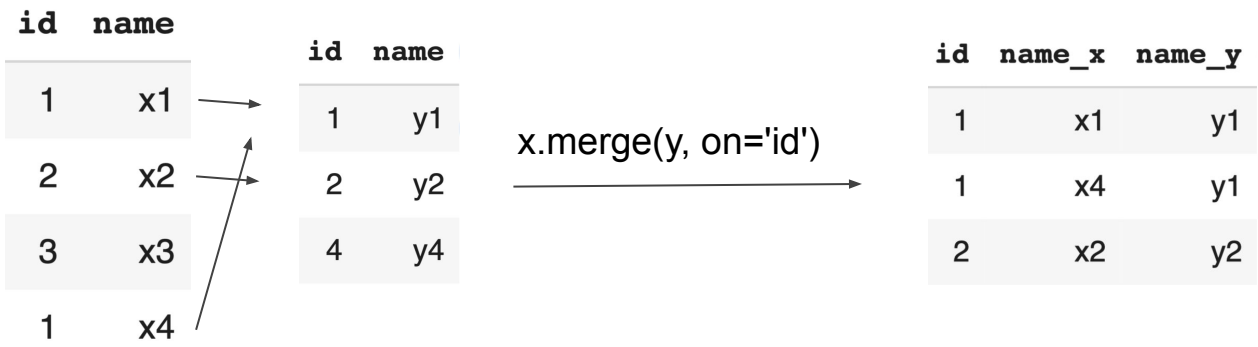
	CourseID	CourseName	Instructor
0	101	Math	Mr. A
1	102	Science	Mrs. B
2	105	History	Mr. C

```
student_course = students.merge(courses, on='CourseID', how='outer')
```

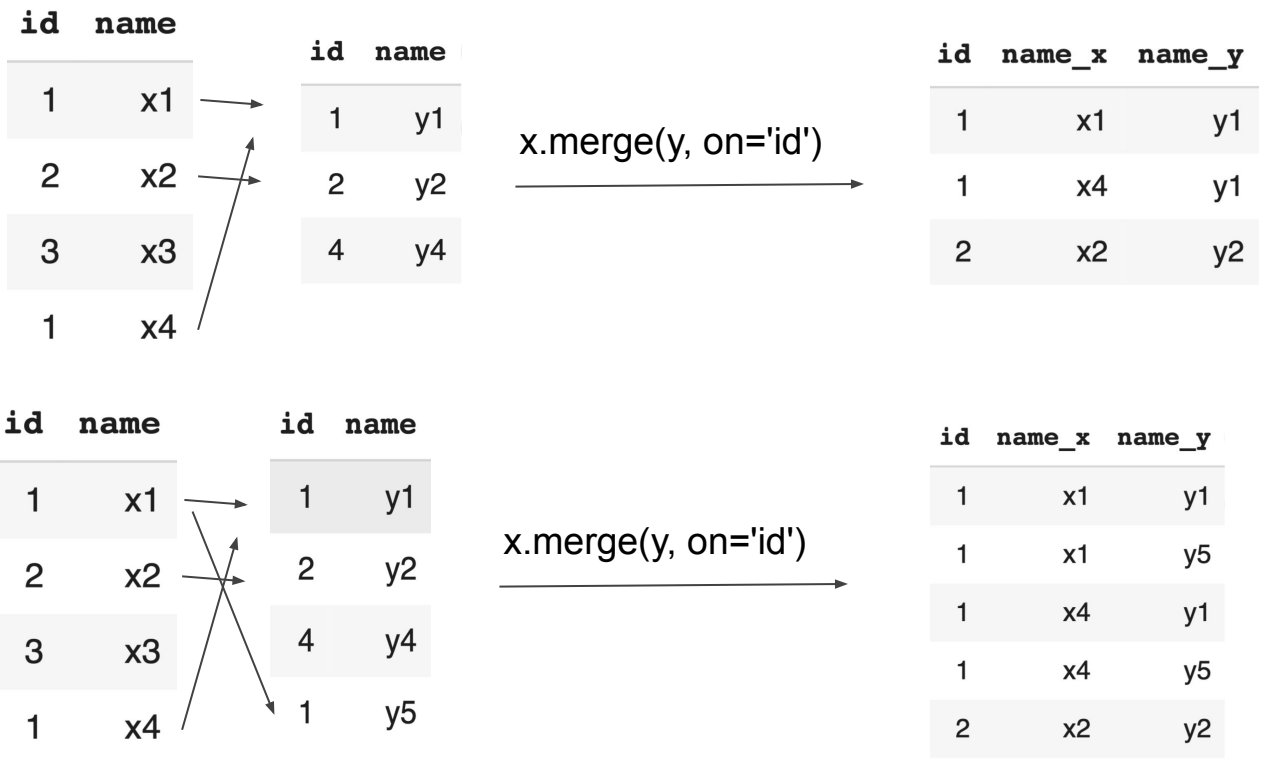
	StudentID	Name	CourseID	CourseName	Instructor
0	1.0	Alice	101	Math	Mr. A
1	2.0	Bob	102	Science	Mrs. B
2	3.0	Charlie	103	NaN	NaN
3	4.0	David	104	NaN	NaN
4	NaN	NaN	105	History	Mr. C

By using an outer join, we ensure that no data is lost from either DataFrame, making it easier to **identify gaps** or **inconsistencies** in the data.

Extra Rows


id name
1 x1
2 x2
3 x3
1 x4
id name
1 y1
2 y2
4 y4
x.merge(y, on='id')
id name_x name_y
1 x1 y1
1 x4 y1
2 x2 y2

Extra Rows


id name
1 x1
2 x2
3 x3
1 x4
id name
1 y1
2 y2
4 y4
x.merge(y, on='id')
id name_x name_y
1 x1 y1
1 x4 y1
2 x2 y2
id name
1 x1
2 x2
3 x3
1 x4
id name
1 y1
2 y2
4 y4
1 y5
x.merge(y, on='id')
id name_x name_y
1 x1 y1
1 x1 y5
1 x4 y1
1 x4 y5
2 x2 y2

Using Join to Filter Data

`baby:`

	Name	Sex	Count	Year
0	Liam	M	19659	2020
1	Noah	M	18252	2020
2	Oliver	M	14147	2020
3	Elijah	M	13034	2020
4	William	M	12541	2020
...	...	...	...	...
2020717	Ula	F	5	1880
2020718	Vannie	F	5	1880
2020719	Verona	F	5	1880
2020720	Vertie	F	5	1880
2020721	Wilma	F	5	1880

2020722 rows × 4 columns

`special_names:`

	Name
0	Archie
1	George
2	Charlotte
3	Oliver
4	Luna
5	Elon
6	Daisy

	Name	Sex	Count	Year
0	Oliver	M	14147	2020
1	Oliver	F	23	2020
2	Oliver	M	13929	2019
3	Oliver	F	22	2019
4	Oliver	M	13469	2018
...	...	...	...	...
1417	Daisy	F	659	1883
1418	Daisy	F	648	1882
1419	Daisy	F	562	1881
1420	Daisy	M	5	1880
1421	Daisy	F	564	1880

1422 rows × 4 columns

```
baby.merge(special_names, how='inner', on='Name')
```