HOW TO GROUP, CONCATENATE & MERGE DATA IN PANDAS

In this tutorial, we show how to group, concatenate, and merge Pandas DataFrames. (New to Pandas? Start with our Pandas introduction or create a Pandas dataframe from a dictionary.)

These operations are very much similar to SQL operations on a row and column database. Pandas, after all, is a row and column in-memory data structure. If you're a SQL programmer, you'll already be familiar with all of this. The only complexity here is that you can join by columns in addition to rows.

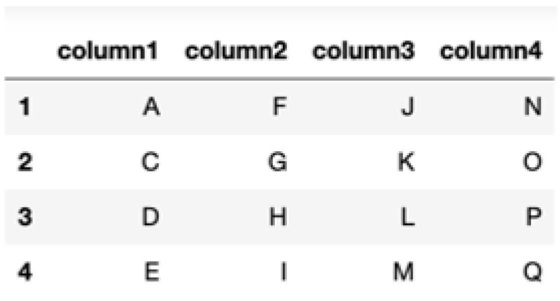
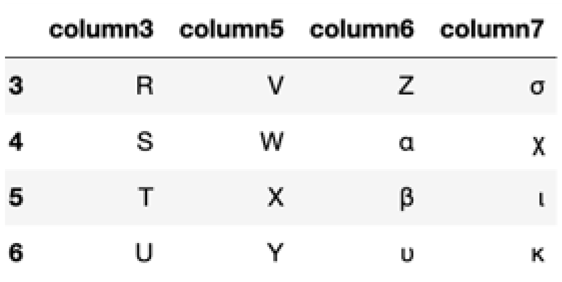
Pandas uses the function concatenation—**concat()**, aka concat. But it's easier to understand if you think of these are **inner joins** (intersection) and **outer joins** (union) of sets, which is how I refer to it below.

(This tutorial is part of our Pandas Guide. Use the right-hand menu to navigate.)

Concatenation (Outer join)

Think of concatenation like an outer join. The result is the same.

Suppose we have dataframes A and B with common elements among the indexes and columns.

Dataframe A	Dataframe B
	

	column1	column2	column3	column4
1	A	F	J	N
2	C	G	K	O
3	D	H	L	P
4	E	I	M	Q

	column3	column5	column6	column7
3	R	V	Z	σ
4	S	W	α	χ
5	T	X	β	ι
6	U	Y	υ	κ

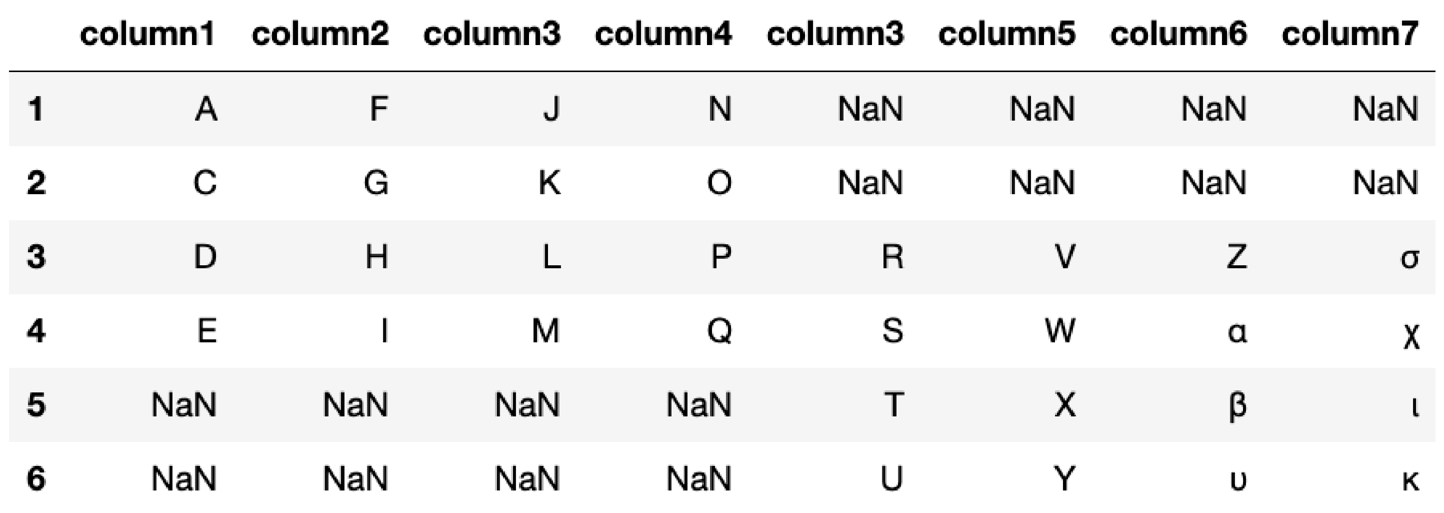
Now concatenate. It's not an append. (There is an append() function for that.) This concat() operation creates a superset of both sets a and b but combines the common rows. It's not an inner join, either, since it lists all rows even those for which there is no common index.

Notice the missing values **NaN**. This is where there are no corresponding dataframe indexes in Dataframe B with the index in Dataframe A.

For example, index 3 is in both dataframes. So, Pandas copies the 4 columns from the first dataframe and the 4 columns from the second dataframe to the newly constructed dataframe. Similarly, index 5 is in Dataframe B but not Dataframe A for columns 1,2, 3. So those columns are marked as missing (NaN).

```
a = pd.DataFrame({'column1': ,

'column2': ,

'column3': ,

'column4': },

index=)

b = pd.DataFrame({'column3': ,

'column5': ,

'column6': ,

'column7': },

index=)

result = pd.concat(, axis=1)
```

Results in:

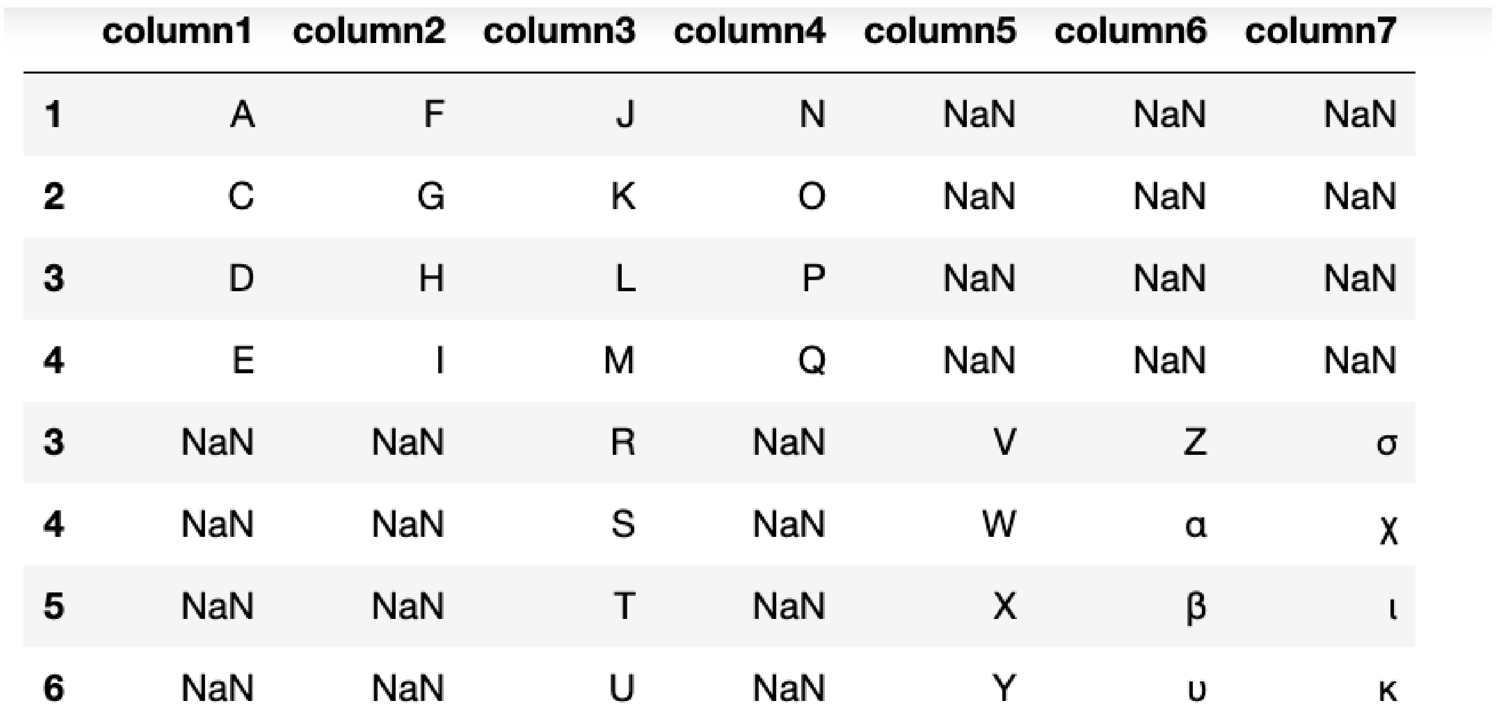


	column1	column2	column3	column4	column3	column5	column6	column7
1	A	F	J	N	NaN	NaN	NaN	NaN
2	C	G	K	O	NaN	NaN	NaN	NaN
3	D	H	L	P	R	V	Z	σ
4	E	I	M	Q	S	W	α	χ
5	NaN	NaN	NaN	NaN	T	X	β	ι
6	NaN	NaN	NaN	NaN	U	Y	υ	κ

Outer join

Here we do an outer join, which, in terms of sets, means the union of two sets. So, all rows are added. An outer join here does not create the intersection of common indexes. But still, for those for which the column does not exist in the set of all columns the value is NaN. That has to be the case since not all columns exist for all rows. So, you have to list all of them but mark some of them as empty.

```
result = pd.concat(, join='outer')
```



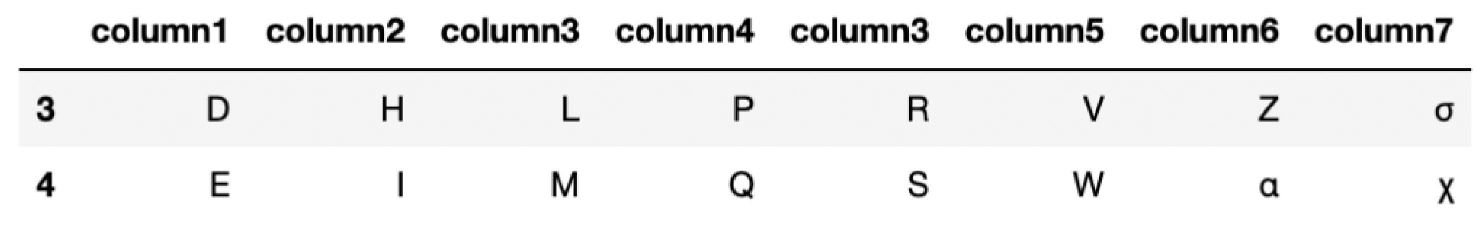
	column1	column2	column3	column4	column5	column6	column7
1	A	F	J	N	NaN	NaN	NaN
2	C	G	K	O	NaN	NaN	NaN
3	D	H	L	P	NaN	NaN	NaN
4	E	I	M	Q	NaN	NaN	NaN
3	NaN	NaN	R	NaN	V	Z	σ
4	NaN	NaN	S	NaN	W	α	χ
5	NaN	NaN	T	NaN	X	β	ι
6	NaN	NaN	U	NaN	Y	υ	κ

Inner join along the 0 axis (Row)

We can do an inner join by index or column. An inner join finds the intersection of two sets.

Let's join along the 0 axis (row). Only indices 3 and 4 are in both dataframes. So, an inner join takes all columns from only those two rows.

```
result = pd.concat(, axis=1,join='inner')
```

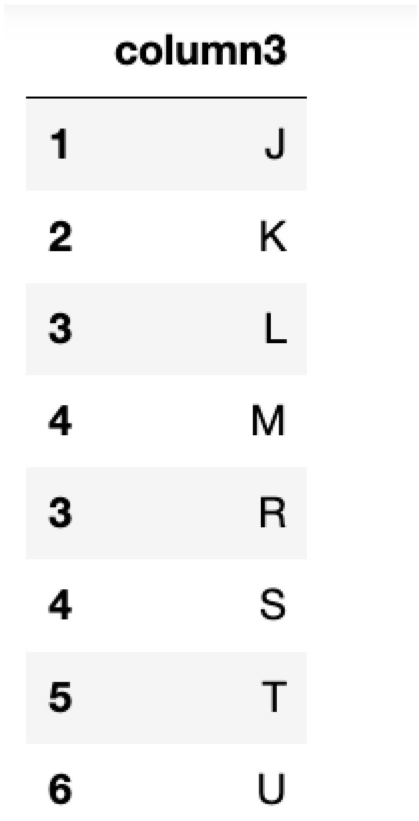


	column1	column2	column3	column4	column3	column5	column6	column7
3	D	H	L	P	R	V	Z	σ
4	E	I	M	Q	S	W	α	χ

Inner join along the 1 axis (Column)

Column3 is the only column common to both dataframe. So, we concatenate all the rows from A with the rows in B and select only the common column, i.e., an inner join along the column axis.

```
result = pd.concat(, axis=0,join='inner')
```



	column3
1	J
2	K
3	L
4	M
3	R
4	S
5	T
6	U

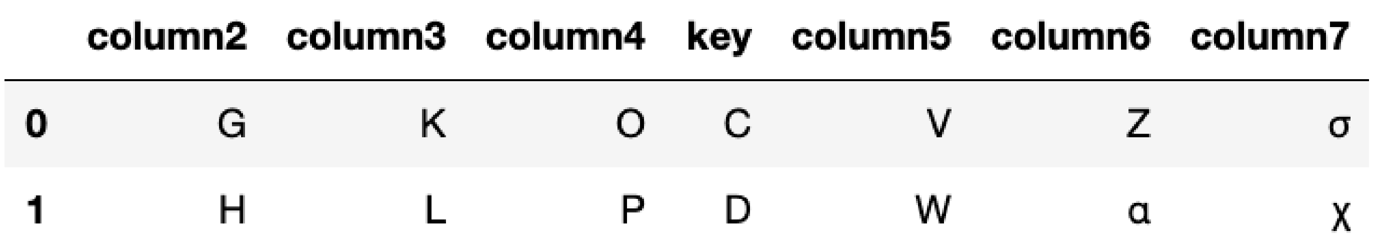
Merge

A merge is like an inner join, except we tell it what column to merge on.

Here, make the first column name (i.e., key value in the dictionary) some common name "key". Then we merge on that.

```
a = pd.DataFrame({'key': ,

'column2': ,

'column3': ,
'column4': },

index=)

b = pd.DataFrame({'key': ,

'column5': ,

'column6': ,

'column7': },

index=)

result=pd.merge(a, b, on='key')
```

The resulting dataframe is the one which has a common key value from the array **key=[]**.You see that only C and D are common to both data frames.



	column2	column3	column4	key	column5	column6	column7
0	G	K	O	C	V	Z	σ
1	H	L	P	D	W	α	χ

Append

This simply appends one dataframe onto another. Here is a Series, which is a DataFrame with only one column. The result is all rows from Dataframe A added to Dataframe B to create Dataframe C.

```
import pandas as pd

a=pd.DataFrame()

b=pd.DataFrame()

c=a.append(b)

c
```

	0
0	1
1	2
2	3
0	4
1	5
2	6

GroupBy

Here is another operation familiar to SQL programmers.

The GroupBy operation is on a single dataframe. We group the values from the column named key and sum them.

```
a=pd.DataFrame({"key":  ,

"values": })

b=a.groupby("key").sum()

b
```

	values
key	
a	2
b	4
c	6

Related reading

- BMC Machine Learning & Big Data Blog
- Pandas: How To Read CSV & JSON Files
- How To Join Tables in Amazon Glue
- Python Development Tools: Your Python Starter Kit
- Snowflake Guide, a series of tutorials