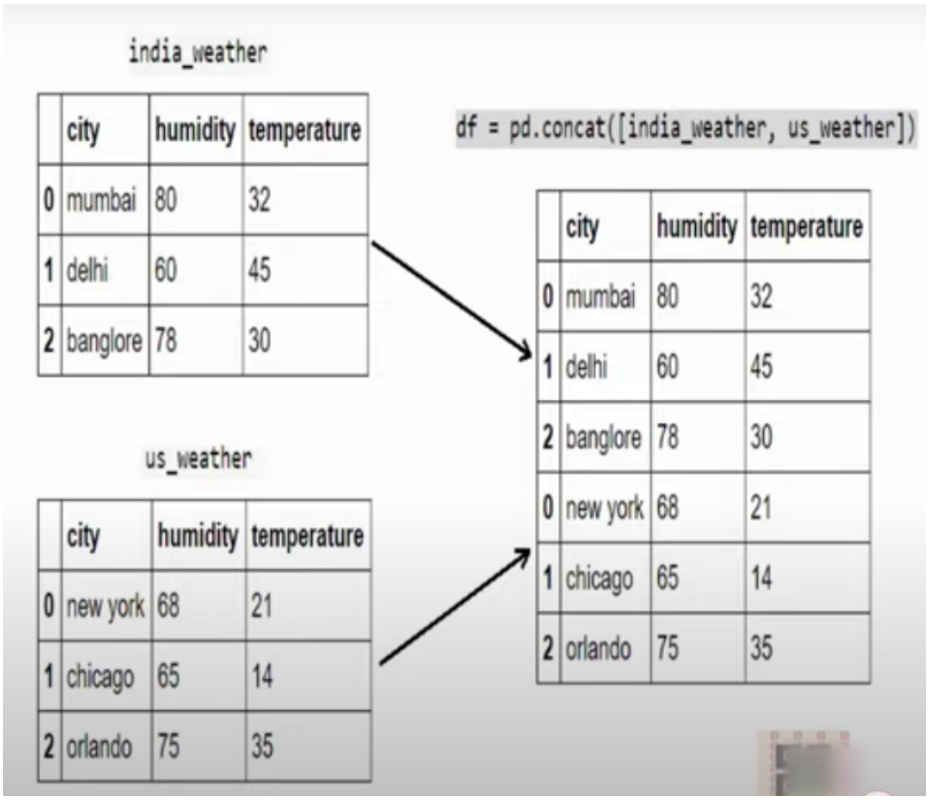

The same indexes are still present.

```
df = pd.concat([india_weather, us_weather], ignore_index=True)
df
```

	city	humidity	temperature
0	mumbai	80	32
1	delhi	60	45
2	banglore	78	30
3	new york	68	21
4	chicago	65	14
5	orlando	75	35

```
df = pd.concat([india_weather, us_weather], keys=["india", "us"])
df
```

		city	humidity	temperature
india	0	mumbai	80	32
	1	delhi	60	45
	2	banglore	78	30
us	0	new york	68	21
	1	chicago	65	14
	2	orlando	75	35

A subset index is created classifying the cities further into Country.

```
df = pd.concat([temperature_df,windspeed_df], axis=1)
df
```

	city	temperature	city	windspeed
0	mumbai	32	mumbai	7
1	delhi	45	delhi	12
2	banglore	30	banglore	9

The code appends the windspeed_df and temperature_df on a vertical axis. Therefore the city column is consolidated into one.

```
windspeed_df = pd.DataFrame({
    "city": ["delhi","mumbai"],
    "windspeed": [7,12],
}, index=[1,0])
windspeed_df
```

	city	windspeed
1	delhi	7
0	mumbai	12

```
df = pd.concat([temperature_df,windspeed_df], axis=1)
df
```

	city	temperature	city	windspeed
0	mumbai	32	mumbai	12.0
1	delhi	45	delhi	7.0
2	banglore	30	NaN	NaN

If the cities do not match up on the rows, you must change the index columns to where they match up.

0	mumbai	32
1	delhi	45
2	banglore	30

```
s = pd.Series(["Humid","Dry","Rain"], name="event")
s
```

```
0    Humid
1      Dry
2     Rain
Name: event, dtype: object
```

```
df = pd.concat([temperature_df, s],axis=1)
df
```

	city	temperature	event
0	mumbai	32	Humid
1	delhi	45	Dry
2	banglore	30	Rain

This adds a series as a new row.

MERGE

Out[1]:

	city	temperature
0	new york	21
1	chicago	14
2	orlando	35

In [2]:
```
df2 = pd.DataFrame({
    "city": ["chicago","new york","orlando"],
    "humidity": [65,68,75],
})
df2
```

Out[2]:

	city	humidity
0	chicago	65
1	new york	68
2	orlando	75

In [3]:
```
df3=pd.merge(df1,df2,on="city")
df3
```

Out[3]:

	city	temperature	humidity
0	new york	21	68
1	chicago	14	65
2	orlando	35	75

The merge is done on the city therefore Pandas looks into correlating cities and matches up.

	city	temperature
0	new york	21
1	chicago	14
2	orlando	35
3	baltimore	32

```
df2 = pd.DataFrame({
    "city": ["chicago","new york","san francisco"],
    "humidity": [65,68, 71],
})
df2
```

	city	humidity
0	chicago	65
1	new york	68
2	san francisco	71

```
df3=pd.merge(df1,df2,on="city")
df3
```

	city	temperature	humidity
0	new york	21	68
1	chicago	14	65

When the cities do not match up perfectly, and we match on cities then we will join on what is in common between the two dataframes.

Out[4]:

	city	temperature
0	new york	21
1	chicago	14
2	orlando	35
3	baltimore	32

In [6]:
```
df2 = pd.DataFrame({
    "city": ["chicago","new york","san francisco"],
    "humidity": [65,68, 71],
})
df2
```

Out[6]:

	city	humidity
0	chicago	65
1	new york	68
2	san francisco	71

In [8]:
```
df3=pd.merge(df1,df2,on="city", how="outer")
df3
```

Out[8]:

	city	temperature	humidity
0	new york	21.0	68.0
1	chicago	14.0	65.0
2	orlando	35.0	NaN
3	baltimore	32.0	NaN
4	san francisco	NaN	71.0

By specifying an outer join we are able to retain all data, and fill NaNs for missing.

Out[4]:

	city	temperature
0	new york	21
1	chicago	14
2	orlando	35
3	baltimore	32

In [6]:
```
df2 = pd.DataFrame({
    "city": ["chicago","new york","san francisco"],
    "humidity": [65,68, 71],
})
df2
```

Out[6]:

	city	humidity
0	chicago	65
1	new york	68
2	san francisco	71

In [10]:
```
df3=pd.merge(df1,df2,on="city", how="left")
df3
```

Out[10]:

	city	temperature	humidity
0	new york	21	68.0
1	chicago	14	65.0
2	orlando	35	NaN
3	baltimore	32	NaN

Because we are making a left join on the df1, ‘San Francisco’ is left behind because it is not in common with df2 & it was not originally in df1.

```
df3=pd.merge(df1,df2,on="city", how="outer", indicator=True)
df3
```

	city	temperature	humidity	_merge
0	new york	21.0	68.0	both
1	chicago	14.0	65.0	both
2	orlando	35.0	NaN	left_only
3	baltimore	32.0	NaN	left_only
4	san francisco	NaN	71.0	right_only

By turning on the indicator, the merge column will explicitly show where the data came from.

0	new york	65	21
1	chicago	68	14
2	orlando	71	35
3	baltimore	75	38

```
df2 = pd.DataFrame({
    "city": ["chicago","new york","san diego"],
    "temperature": [21,14,35],
    "humidity": [65,68,71]
})
df2
```

	city	humidity	temperature
0	chicago	65	21
1	new york	68	14
2	san diego	71	35

```
df3=pd.merge(df1,df2,on="city",suffixes=('_left','_right'))
df3
```

	city	humidity_left	temperature_left	humidity_right	temperature_right
0	new york	65	21	68	14
1	chicago	68	14	65	21

When joining with duplicate columns we can specify through the suffixes parameter.