Merging/Joining Tables

When performing data analysis, it's common to work with data scattered across multiple tables or files. To build a complete dataset for analysis, you often need to combine these tables into one, bringing together related pieces of information from each. This process, known as merging or joining, is a critical data preparation step.

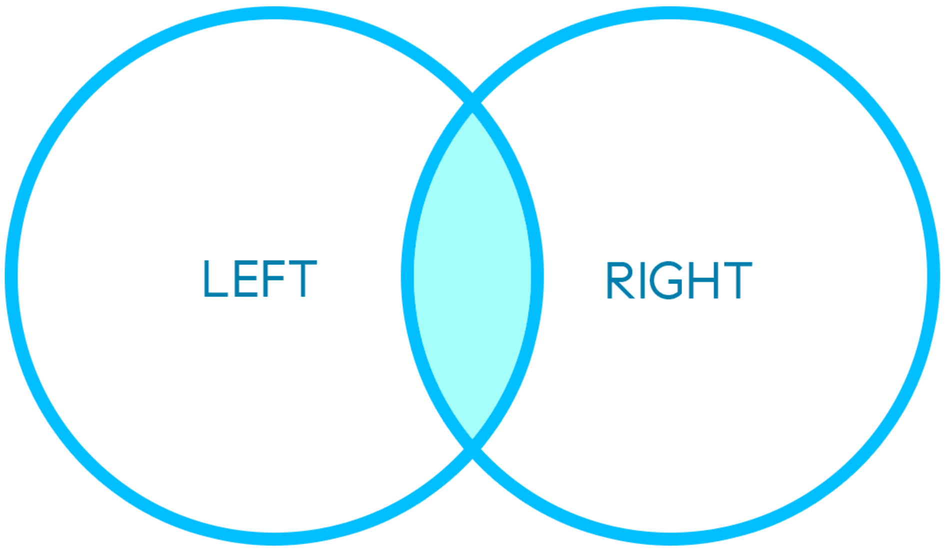
In this chapter, we'll explore the skills and methods needed to merge tables using Python's Pandas library, a popular tool for data manipulation and analysis. We'll cover how to join tables effectively, understand the types of joins available, and how to choose the best method for different datasets. Additionally, we'll delve into techniques for handling issues like missing data and overlapping columns, which are common challenges when merging tables.

Type of joins:

When merging tables, the way you join them determines which records appear in the final dataset. In Pandas, you have access to four main types of joins: **inner**, **left**, **right**, and **outer**. Each join type has specific use cases depending on your data and analysis goals.

Inner join:

An inner join returns only the rows where **there is a match in both tables**. If a record appears in only one of the tables, it won't be included in the final output. This join type is useful when you only want the common records between two datasets.



In [1]:
```
import pandas as pd
```

```
df1 = pd.DataFrame({ 'CustomerID': [1, 2, 3, 4], 'Order': ['A', 'B', 'C', 'D'] }
df1
```

Out[1]:

	CustomerID	Order
0	1	A
1	2	B
2	3	C
3	4	D

In [2]:

```
df2 = pd.DataFrame({'CustomerID': [3, 4, 5, 6], 'Amount': [100, 150, 200, 250]})
df2
```

Out[2]:

	CustomerID	Amount
0	3	100
1	4	150
2	5	200
3	6	250

In [3]:

```
# Inner join
merged_df = pd.merge(df1, df2, on='CustomerID', how='inner')
merged_df
```

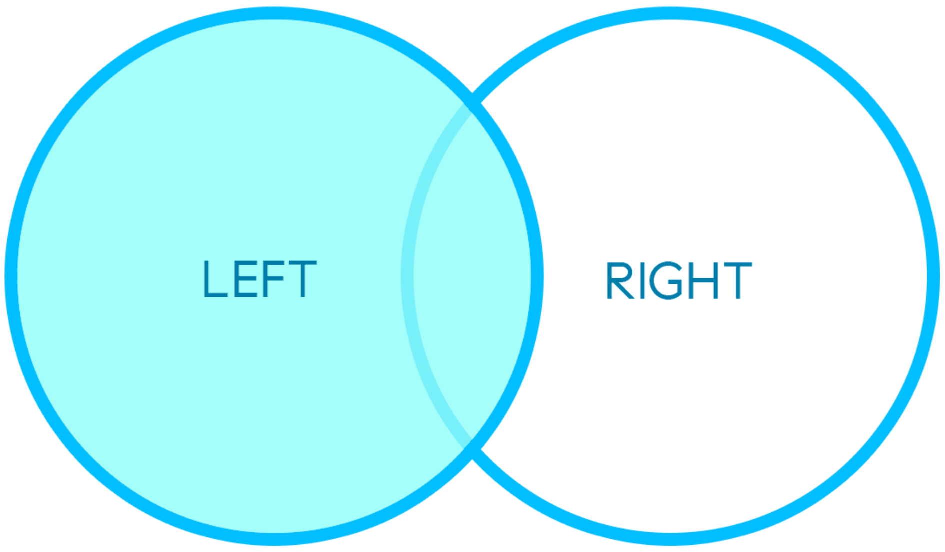
Out[3]:

	CustomerID	Order	Amount
0	3	C	100
1	4	D	150

This example merges on the `CustomerID` column, returning only the rows where `CustomerID` is common to both `df1` and `df2`.

Left join:

A left join keeps all rows from the left table and matches rows from the right table where possible. If there's no matching row in the right table, `NaN` is inserted in the resulting columns.



```
In [4]: df1
```

Out[4]:

	CustomerID	Order
0	1	A
1	2	B
2	3	C
3	4	D

```
In [5]: df2
```

Out[5]:

	CustomerID	Amount
0	3	100
1	4	150
2	5	200
3	6	250

```
In [6]: merged_df = pd.merge(df1, df2, on='CustomerID', how='left')
        merged_df
```

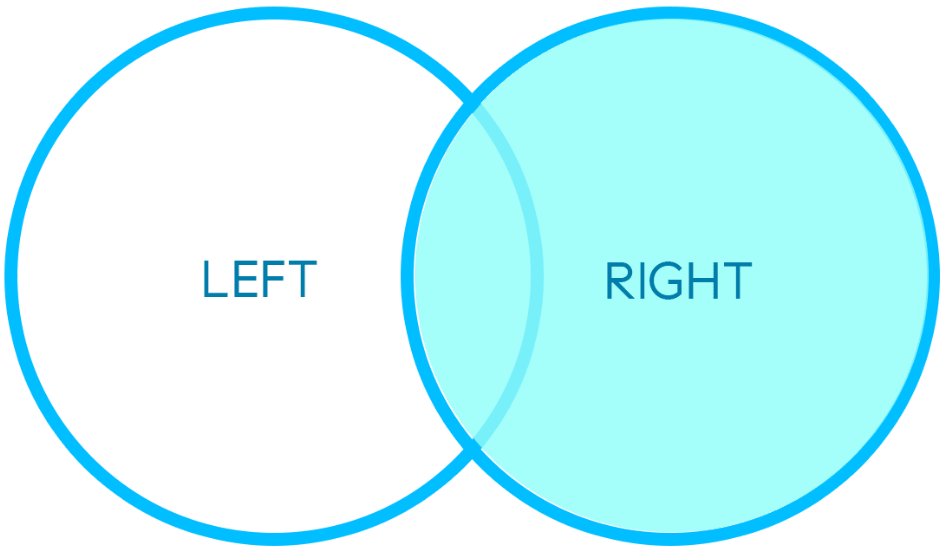
Out[6]:

	CustomerID	Order	Amount
0	1	A	NaN
1	2	B	NaN
2	3	C	100.0
3	4	D	150.0

Here, the left join keeps all `CustomerID` s from `df1` , filling in `NaN` for those without a match in `df2` .

Right join:

The right join is the opposite of the left join, retaining all rows from the right table and matching rows from the left where possible.



In [7]:
```
df1
```

Out[7]:

	CustomerID	Order
0	1	A
1	2	B
2	3	C
3	4	D

In [8]:
```
df2
```

Out[8]:

	CustomerID	Amount
0	3	100
1	4	150
2	5	200
3	6	250

In [9]:
```
merged_df = pd.merge(df1, df2, on='CustomerID', how='right')
merged_df
```

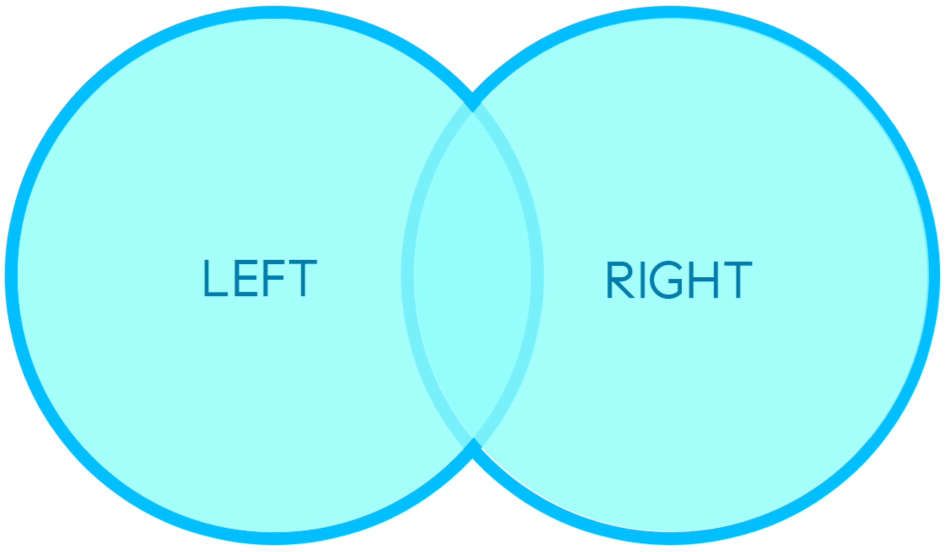
Out[9]:

	CustomerID	Order	Amount
0	3	C	100
1	4	D	150
2	5	NaN	200
3	6	NaN	250

This result keeps all rows from `df2`, filling in `NaN` for rows in `df1` without a match.

Outer join:

An outer join includes all rows from both tables, filling `NaN` where there are missing matches in either table. This join is helpful when you want a complete view of both datasets, regardless of whether every row has a match.



In [10]:
```
df1
```

Out[10]:

	CustomerID	Order
0	1	A
1	2	B
2	3	C
3	4	D

In [11]:
```
df2
```

Out[11]:

	CustomerID	Amount
0	3	100
1	4	150
2	5	200
3	6	250

In [12]:
```
merged_df = pd.merge(df1, df2, on='CustomerID', how='outer')
merged_df
```

Out[12]:

	CustomerID	Order	Amount
0	1	A	NaN
1	2	B	NaN
2	3	C	100.0
3	4	D	150.0
4	5	NaN	200.0
5	6	NaN	250.0

With the outer join, you get all rows from both tables, with `NaN` values where matches were not found.

Indicator parameter:

In [13]:
```
# df2.loc[3, 'CustomerID'] = 0
# df2

merged_df = pd.merge(df2, df1, on='CustomerID', how='outer', indicator=True)
merged_df
```

Out[13]:

	CustomerID	Amount	Order	_merge
0	1	NaN	A	right_only
1	2	NaN	B	right_only
2	3	100.0	C	both
3	4	150.0	D	both
4	5	200.0	NaN	left_only
5	6	250.0	NaN	left_only

Summary:

Selecting the right join depends on the data and questions you're addressing. Here's a quick guide:

- **Inner join**: Use when you need only records that have matches in both tables.
- **Left join**: Use when you want to keep all records from the first (left) table and match where possible in the second (right) table.
- **Right join**: Use when you want to keep all records from the second (right) table and match where possible in the first (left) table.
- **Outer join**: Use when you want a complete record of all rows, regardless of whether they have matches in both tables.

Practice:

We'll demonstrate merging and joining concepts using **food prices in palestine** dataset. We have two files: **"west bank.csv"**, which lists food prices of all items available in west bank for January 2023, and **"gaza.csv"**, which also lists food prices of all items available in Gaza for January 2023. By merging these files, we'll be able to compare the prices between the two regions for insightful analysis.

In [2]:
```
import pandas as pd
wb = pd.read_csv("data/west bank.csv")
wb.head(4)
```

Out[2]:

	date	region	category	commodity	price
0	1/15/2023	West Bank	cereals and tubers	Bread	4.62
1	1/15/2023	West Bank	cereals and tubers	Potatoes (medium size)	4.05
2	1/15/2023	West Bank	cereals and tubers	Rice (small grain, imported)	149.78
3	1/15/2023	West Bank	cereals and tubers	Wheat flour	188.33

In [3]:
```
gaza = pd.read_csv("data/gaza.csv")
gaza.head(4)
```

Out[3]:

	date	region	category	commodity	price
0	1/15/2023	Gaza Strip	cereals and tubers	Bread	2.89
1	1/15/2023	Gaza Strip	cereals and tubers	Potatoes (medium size)	1.95
2	1/15/2023	Gaza Strip	cereals and tubers	Rice (small grain, imported)	154.50
3	1/15/2023	Gaza Strip	cereals and tubers	Wheat flour (locally processed)	110.00

In [5]:
```
df = pd.read_csv("data/wfp_food_prices_pse.csv")
df = df[df['date'] == '1/15/2023']
df = df[['date', 'region', 'category', 'commodity', 'price']]
gaza = df[df['region'] == "Gaza Strip"]
wb = df[df['region'] == "West Bank"]
wb.to_csv("data/west bank.csv", index=False)
gaza.to_csv("data/gaza.csv", index=False)
```

Identify commodities that are priced higher in the West Bank compared to Gaza, and vice versa.

In [8]:

```python
m =  wb.merge(gaza, on="commodity", how='inner')
m[m['price_x'] < m['price_y']]
```

Out[8]:

	date_x	region_x	category_x	commodity	price_x	date_y	region_y	categor
2	1/15/2023	West Bank	cereals and tubers	Rice (small grain, imported)	149.78	1/15/2023	Gaza Strip	cereals a tub
16	1/15/2023	West Bank	non-food	Fuel (petrol-gasoline)	6.59	1/15/2023	Gaza Strip	non-fc
18	1/15/2023	West Bank	oil and fats	Oil (maize)	32.05	1/15/2023	Gaza Strip	oil and f
19	1/15/2023	West Bank	oil and fats	Oil (olive)	28.87	1/15/2023	Gaza Strip	oil and f

In [14]:

```python
merged = gaza.merge(wb, how="inner", on="commodity", suffixes=("_gaza", "_wb"))
merged[merged['price_gaza'] < merged['price_wb']]
```

Out[14]:

	date_gaza	region_gaza	category_gaza	commodity	price_gaza	date_wb	region_
0	1/15/2023	Gaza Strip	cereals and tubers	Bread	2.89	1/15/2023	West B
1	1/15/2023	Gaza Strip	cereals and tubers	Potatoes (medium size)	1.95	1/15/2023	West B
3	1/15/2023	Gaza Strip	meat, fish and eggs	Eggs	14.67	1/15/2023	West B
4	1/15/2023	Gaza Strip	meat, fish and eggs	Fish (frozen)	11.50	1/15/2023	West B
5	1/15/2023	Gaza Strip	meat, fish and eggs	Meat (beef)	41.80	1/15/2023	West B
6	1/15/2023	Gaza Strip	meat, fish and eggs	Meat (chicken)	14.88	1/15/2023	West B
7	1/15/2023	Gaza Strip	meat, fish and eggs	Meat (goat, with bones)	54.22	1/15/2023	West B
8	1/15/2023	Gaza Strip	milk and dairy	Cheese (goat)	16.57	1/15/2023	West B
9	1/15/2023	Gaza Strip	milk and dairy	Labaneh	7.66	1/15/2023	West B
10	1/15/2023	Gaza Strip	milk and dairy	Milk (pasteurized)	5.97	1/15/2023	West B
11	1/15/2023	Gaza Strip	milk and dairy	Milk (powder)	93.45	1/15/2023	West B
12	1/15/2023	Gaza Strip	milk and dairy	Yogurt	5.58	1/15/2023	West B
13	1/15/2023	Gaza Strip	miscellaneous food	Salt	1.50	1/15/2023	West B
14	1/15/2023	Gaza Strip	miscellaneous food	Sugar	3.03	1/15/2023	West B
15	1/15/2023	Gaza Strip	miscellaneous food	Tea	12.75	1/15/2023	West B
17	1/15/2023	Gaza Strip	non-food	Water (drinking)	1.60	1/15/2023	West B
20	1/15/2023	Gaza Strip	pulses and nuts	Beans (fava, small, tinned)	2.50	1/15/2023	West B
21	1/15/2023	Gaza Strip	pulses and nuts	Chickpeas	5.09	1/15/2023	West B
22	1/15/2023	Gaza Strip	pulses and nuts	Lentils	6.13	1/15/2023	West B
23	1/15/2023	Gaza Strip	vegetables and fruits	Apples (red)	4.06	1/15/2023	West B

	date_gaza	region_gaza	category_gaza	commodity	price_gaza	date_wb	region_
24	1/15/2023	Gaza Strip	vegetables and fruits	Bananas (medium size)	3.30	1/15/2023	West B
25	1/15/2023	Gaza Strip	vegetables and fruits	Cauliflower	2.50	1/15/2023	West B
26	1/15/2023	Gaza Strip	vegetables and fruits	Cucumbers (greenhouse)	1.70	1/15/2023	West B
27	1/15/2023	Gaza Strip	vegetables and fruits	Eggplants (large)	1.92	1/15/2023	West B
28	1/15/2023	Gaza Strip	vegetables and fruits	Onions (dry, local)	2.02	1/15/2023	West B
29	1/15/2023	Gaza Strip	vegetables and fruits	Tomatoes (greenhouse)	2.20	1/15/2023	West B

Identify commodities that are available for sale in the West Bank but not in Gaza.

In [19]:
```
m2 = pd.merge(wb, gaza, on='commodity', how='right',suffixes=("_wb", "_gaza"))
m2[m2['price_wb'].isna()]
```

Out[19]:

	date_wb	region_wb	category_wb	commodity	price_wb	date_gaza	region_gaza	c
3	NaN	NaN	NaN	Wheat flour (locally processed)	NaN	1/15/2023	Gaza Strip	

In [20]:
```
wb.merge(gaza, on="commodity", how="left",  suffixes=("_wb", "_gaza"))
```

Out[20]:

	date_wb	region_wb	category_wb	commodity	price_wb	date_gaza	region_gaza
0	1/15/2023	West Bank	cereals and tubers	Bread	4.62	1/15/2023	Gaza Strip
1	1/15/2023	West Bank	cereals and tubers	Potatoes (medium size)	4.05	1/15/2023	Gaza Strip
2	1/15/2023	West Bank	cereals and tubers	Rice (small grain, imported)	149.78	1/15/2023	Gaza Strip
3	1/15/2023	West Bank	cereals and tubers	Wheat flour	188.33	NaN	NaN
4	1/15/2023	West Bank	meat, fish and eggs	Eggs	20.16	1/15/2023	Gaza Strip
5	1/15/2023	West Bank	meat, fish and eggs	Fish (frozen)	14.86	1/15/2023	Gaza Strip
6	1/15/2023	West Bank	meat, fish and eggs	Meat (beef)	49.73	1/15/2023	Gaza Strip
7	1/15/2023	West Bank	meat, fish and eggs	Meat (chicken)	16.20	1/15/2023	Gaza Strip
8	1/15/2023	West Bank	meat, fish and eggs	Meat (goat, with bones)	80.16	1/15/2023	Gaza Strip
9	1/15/2023	West Bank	milk and dairy	Cheese (goat)	23.75	1/15/2023	Gaza Strip
10	1/15/2023	West Bank	milk and dairy	Labaneh	9.65	1/15/2023	Gaza Strip
11	1/15/2023	West Bank	milk and dairy	Milk (pasteurized)	7.12	1/15/2023	Gaza Strip
12	1/15/2023	West Bank	milk and dairy	Milk (powder)	96.09	1/15/2023	Gaza Strip
13	1/15/2023	West Bank	milk and dairy	Yogurt	6.02	1/15/2023	Gaza Strip
14	1/15/2023	West Bank	miscellaneous food	Salt	1.88	1/15/2023	Gaza Strip
15	1/15/2023	West Bank	miscellaneous food	Sugar	3.98	1/15/2023	Gaza Strip
16	1/15/2023	West Bank	miscellaneous food	Tea	16.51	1/15/2023	Gaza Strip
17	1/15/2023	West Bank	non-food	Fuel (petrol-gasoline)	6.59	1/15/2023	Gaza Strip
18	1/15/2023	West Bank	non-food	Water (drinking)	3.75	1/15/2023	Gaza Strip
19	1/15/2023	West Bank	oil and fats	Oil (maize)	32.05	1/15/2023	Gaza Strip
20	1/15/2023	West Bank	oil and fats	Oil (olive)	28.87	1/15/2023	Gaza Strip

	date_wb	region_wb	category_wb	commodity	price_wb	date_gaza	region_gaza
21	1/15/2023	West Bank	pulses and nuts	Beans (fava, small, tinned)	3.37	1/15/2023	Gaza Strip
22	1/15/2023	West Bank	pulses and nuts	Chickpeas	8.15	1/15/2023	Gaza Strip
23	1/15/2023	West Bank	pulses and nuts	Lentils	7.56	1/15/2023	Gaza Strip
24	1/15/2023	West Bank	vegetables and fruits	Apples (red)	7.79	1/15/2023	Gaza Strip
25	1/15/2023	West Bank	vegetables and fruits	Bananas (medium size)	4.07	1/15/2023	Gaza Strip
26	1/15/2023	West Bank	vegetables and fruits	Cauliflower	3.46	1/15/2023	Gaza Strip
27	1/15/2023	West Bank	vegetables and fruits	Cucumbers (greenhouse)	4.36	1/15/2023	Gaza Strip
28	1/15/2023	West Bank	vegetables and fruits	Eggplants (large)	3.52	1/15/2023	Gaza Strip
29	1/15/2023	West Bank	vegetables and fruits	Onions (dry, local)	3.89	1/15/2023	Gaza Strip
30	1/15/2023	West Bank	vegetables and fruits	Tomatoes (greenhouse)	3.93	1/15/2023	Gaza Strip

Identify commodities that are NOT available for sale in the West Bank but available in Gaza.

```
In [ ]:
```

```
In [21]: wb.merge(gaza, on="commodity", how="right", suffixes=("_wb", "_gaza"))
```

Out[21]:

	date_wb	region_wb	category_wb	commodity	price_wb	date_gaza	region_gaza
0	1/15/2023	West Bank	cereals and tubers	Bread	4.62	1/15/2023	Gaza Strip
1	1/15/2023	West Bank	cereals and tubers	Potatoes (medium size)	4.05	1/15/2023	Gaza Strip
2	1/15/2023	West Bank	cereals and tubers	Rice (small grain, imported)	149.78	1/15/2023	Gaza Strip
3	NaN	NaN	NaN	Wheat flour (locally processed)	NaN	1/15/2023	Gaza Strip
4	1/15/2023	West Bank	meat, fish and eggs	Eggs	20.16	1/15/2023	Gaza Strip
5	1/15/2023	West Bank	meat, fish and eggs	Fish (frozen)	14.86	1/15/2023	Gaza Strip
6	1/15/2023	West Bank	meat, fish and eggs	Meat (beef)	49.73	1/15/2023	Gaza Strip
7	1/15/2023	West Bank	meat, fish and eggs	Meat (chicken)	16.20	1/15/2023	Gaza Strip
8	1/15/2023	West Bank	meat, fish and eggs	Meat (goat, with bones)	80.16	1/15/2023	Gaza Strip
9	1/15/2023	West Bank	milk and dairy	Cheese (goat)	23.75	1/15/2023	Gaza Strip
10	1/15/2023	West Bank	milk and dairy	Labaneh	9.65	1/15/2023	Gaza Strip
11	1/15/2023	West Bank	milk and dairy	Milk (pasteurized)	7.12	1/15/2023	Gaza Strip
12	1/15/2023	West Bank	milk and dairy	Milk (powder)	96.09	1/15/2023	Gaza Strip
13	1/15/2023	West Bank	milk and dairy	Yogurt	6.02	1/15/2023	Gaza Strip
14	1/15/2023	West Bank	miscellaneous food	Salt	1.88	1/15/2023	Gaza Strip
15	1/15/2023	West Bank	miscellaneous food	Sugar	3.98	1/15/2023	Gaza Strip
16	1/15/2023	West Bank	miscellaneous food	Tea	16.51	1/15/2023	Gaza Strip
17	1/15/2023	West Bank	non-food	Fuel (petrol-gasoline)	6.59	1/15/2023	Gaza Strip
18	1/15/2023	West Bank	non-food	Water (drinking)	3.75	1/15/2023	Gaza Strip
19	1/15/2023	West Bank	oil and fats	Oil (maize)	32.05	1/15/2023	Gaza Strip

	date_wb	region_wb	category_wb	commodity	price_wb	date_gaza	region_gaza
20	1/15/2023	West Bank	oil and fats	Oil (olive)	28.87	1/15/2023	Gaza Strip
21	1/15/2023	West Bank	pulses and nuts	Beans (fava, small, tinned)	3.37	1/15/2023	Gaza Strip
22	1/15/2023	West Bank	pulses and nuts	Chickpeas	8.15	1/15/2023	Gaza Strip
23	1/15/2023	West Bank	pulses and nuts	Lentils	7.56	1/15/2023	Gaza Strip
24	1/15/2023	West Bank	vegetables and fruits	Apples (red)	7.79	1/15/2023	Gaza Strip
25	1/15/2023	West Bank	vegetables and fruits	Bananas (medium size)	4.07	1/15/2023	Gaza Strip
26	1/15/2023	West Bank	vegetables and fruits	Cauliflower	3.46	1/15/2023	Gaza Strip
27	1/15/2023	West Bank	vegetables and fruits	Cucumbers (greenhouse)	4.36	1/15/2023	Gaza Strip
28	1/15/2023	West Bank	vegetables and fruits	Eggplants (large)	3.52	1/15/2023	Gaza Strip
29	1/15/2023	West Bank	vegetables and fruits	Onions (dry, local)	3.89	1/15/2023	Gaza Strip
30	1/15/2023	West Bank	vegetables and fruits	Tomatoes (greenhouse)	3.93	1/15/2023	Gaza Strip

Compile a list of all food prices available in both Gaza and the West Bank.

```
In [22]: wb.merge(gaza, on = 'commodity', how='outer')
```

Out[22]:

	date_x	region_x	category_x	commodity	price_x	date_y	region_y	cat
0	1/15/2023	West Bank	vegetables and fruits	Apples (red)	7.79	1/15/2023	Gaza Strip	ve a
1	1/15/2023	West Bank	vegetables and fruits	Bananas (medium size)	4.07	1/15/2023	Gaza Strip	ve a
2	1/15/2023	West Bank	pulses and nuts	Beans (fava, small, tinned)	3.37	1/15/2023	Gaza Strip	pu
3	1/15/2023	West Bank	cereals and tubers	Bread	4.62	1/15/2023	Gaza Strip	ce
4	1/15/2023	West Bank	vegetables and fruits	Cauliflower	3.46	1/15/2023	Gaza Strip	ve a
5	1/15/2023	West Bank	milk and dairy	Cheese (goat)	23.75	1/15/2023	Gaza Strip	
6	1/15/2023	West Bank	pulses and nuts	Chickpeas	8.15	1/15/2023	Gaza Strip	pu
7	1/15/2023	West Bank	vegetables and fruits	Cucumbers (greenhouse)	4.36	1/15/2023	Gaza Strip	ve a
8	1/15/2023	West Bank	vegetables and fruits	Eggplants (large)	3.52	1/15/2023	Gaza Strip	ve a
9	1/15/2023	West Bank	meat, fish and eggs	Eggs	20.16	1/15/2023	Gaza Strip	r a
10	1/15/2023	West Bank	meat, fish and eggs	Fish (frozen)	14.86	1/15/2023	Gaza Strip	r a
11	1/15/2023	West Bank	non-food	Fuel (petrol-gasoline)	6.59	1/15/2023	Gaza Strip	r
12	1/15/2023	West Bank	milk and dairy	Labaneh	9.65	1/15/2023	Gaza Strip	
13	1/15/2023	West Bank	pulses and nuts	Lentils	7.56	1/15/2023	Gaza Strip	pu
14	1/15/2023	West Bank	meat, fish and eggs	Meat (beef)	49.73	1/15/2023	Gaza Strip	r a
15	1/15/2023	West Bank	meat, fish and eggs	Meat (chicken)	16.20	1/15/2023	Gaza Strip	r a
16	1/15/2023	West Bank	meat, fish and eggs	Meat (goat, with bones)	80.16	1/15/2023	Gaza Strip	r a
17	1/15/2023	West Bank	milk and dairy	Milk (pasteurized)	7.12	1/15/2023	Gaza Strip	
18	1/15/2023	West Bank	milk and dairy	Milk (powder)	96.09	1/15/2023	Gaza Strip	
19	1/15/2023	West Bank	oil and fats	Oil (maize)	32.05	1/15/2023	Gaza Strip	oil

	date_x	region_x	category_x	commodity	price_x	date_y	region_y	cat
20	1/15/2023	West Bank	oil and fats	Oil (olive)	28.87	1/15/2023	Gaza Strip	oil
21	1/15/2023	West Bank	vegetables and fruits	Onions (dry, local)	3.89	1/15/2023	Gaza Strip	ve a
22	1/15/2023	West Bank	cereals and tubers	Potatoes (medium size)	4.05	1/15/2023	Gaza Strip	ce
23	1/15/2023	West Bank	cereals and tubers	Rice (small grain, imported)	149.78	1/15/2023	Gaza Strip	ce
24	1/15/2023	West Bank	miscellaneous food	Salt	1.88	1/15/2023	Gaza Strip	misce
25	1/15/2023	West Bank	miscellaneous food	Sugar	3.98	1/15/2023	Gaza Strip	misce
26	1/15/2023	West Bank	miscellaneous food	Tea	16.51	1/15/2023	Gaza Strip	misce
27	1/15/2023	West Bank	vegetables and fruits	Tomatoes (greenhouse)	3.93	1/15/2023	Gaza Strip	ve a
28	1/15/2023	West Bank	non-food	Water (drinking)	3.75	1/15/2023	Gaza Strip	r
29	1/15/2023	West Bank	cereals and tubers	Wheat flour	188.33	NaN	NaN	
30	NaN	NaN	NaN	Wheat flour (locally processed)	NaN	1/15/2023	Gaza Strip	ce
31	1/15/2023	West Bank	milk and dairy	Yogurt	6.02	1/15/2023	Gaza Strip	

```
In [23]: wb.merge(gaza, on="commodity", how="outer", suffixes=("_wb", "_gaza"))
```

Out[23]:

	date_wb	region_wb	category_wb	commodity	price_wb	date_gaza	region_gaza
0	1/15/2023	West Bank	vegetables and fruits	Apples (red)	7.79	1/15/2023	Gaza Strip
1	1/15/2023	West Bank	vegetables and fruits	Bananas (medium size)	4.07	1/15/2023	Gaza Strip
2	1/15/2023	West Bank	pulses and nuts	Beans (fava, small, tinned)	3.37	1/15/2023	Gaza Strip
3	1/15/2023	West Bank	cereals and tubers	Bread	4.62	1/15/2023	Gaza Strip
4	1/15/2023	West Bank	vegetables and fruits	Cauliflower	3.46	1/15/2023	Gaza Strip
5	1/15/2023	West Bank	milk and dairy	Cheese (goat)	23.75	1/15/2023	Gaza Strip
6	1/15/2023	West Bank	pulses and nuts	Chickpeas	8.15	1/15/2023	Gaza Strip
7	1/15/2023	West Bank	vegetables and fruits	Cucumbers (greenhouse)	4.36	1/15/2023	Gaza Strip
8	1/15/2023	West Bank	vegetables and fruits	Eggplants (large)	3.52	1/15/2023	Gaza Strip
9	1/15/2023	West Bank	meat, fish and eggs	Eggs	20.16	1/15/2023	Gaza Strip
10	1/15/2023	West Bank	meat, fish and eggs	Fish (frozen)	14.86	1/15/2023	Gaza Strip
11	1/15/2023	West Bank	non-food	Fuel (petrol-gasoline)	6.59	1/15/2023	Gaza Strip
12	1/15/2023	West Bank	milk and dairy	Labaneh	9.65	1/15/2023	Gaza Strip
13	1/15/2023	West Bank	pulses and nuts	Lentils	7.56	1/15/2023	Gaza Strip
14	1/15/2023	West Bank	meat, fish and eggs	Meat (beef)	49.73	1/15/2023	Gaza Strip
15	1/15/2023	West Bank	meat, fish and eggs	Meat (chicken)	16.20	1/15/2023	Gaza Strip
16	1/15/2023	West Bank	meat, fish and eggs	Meat (goat, with bones)	80.16	1/15/2023	Gaza Strip
17	1/15/2023	West Bank	milk and dairy	Milk (pasteurized)	7.12	1/15/2023	Gaza Strip
18	1/15/2023	West Bank	milk and dairy	Milk (powder)	96.09	1/15/2023	Gaza Strip
19	1/15/2023	West Bank	oil and fats	Oil (maize)	32.05	1/15/2023	Gaza Strip
20	1/15/2023	West Bank	oil and fats	Oil (olive)	28.87	1/15/2023	Gaza Strip

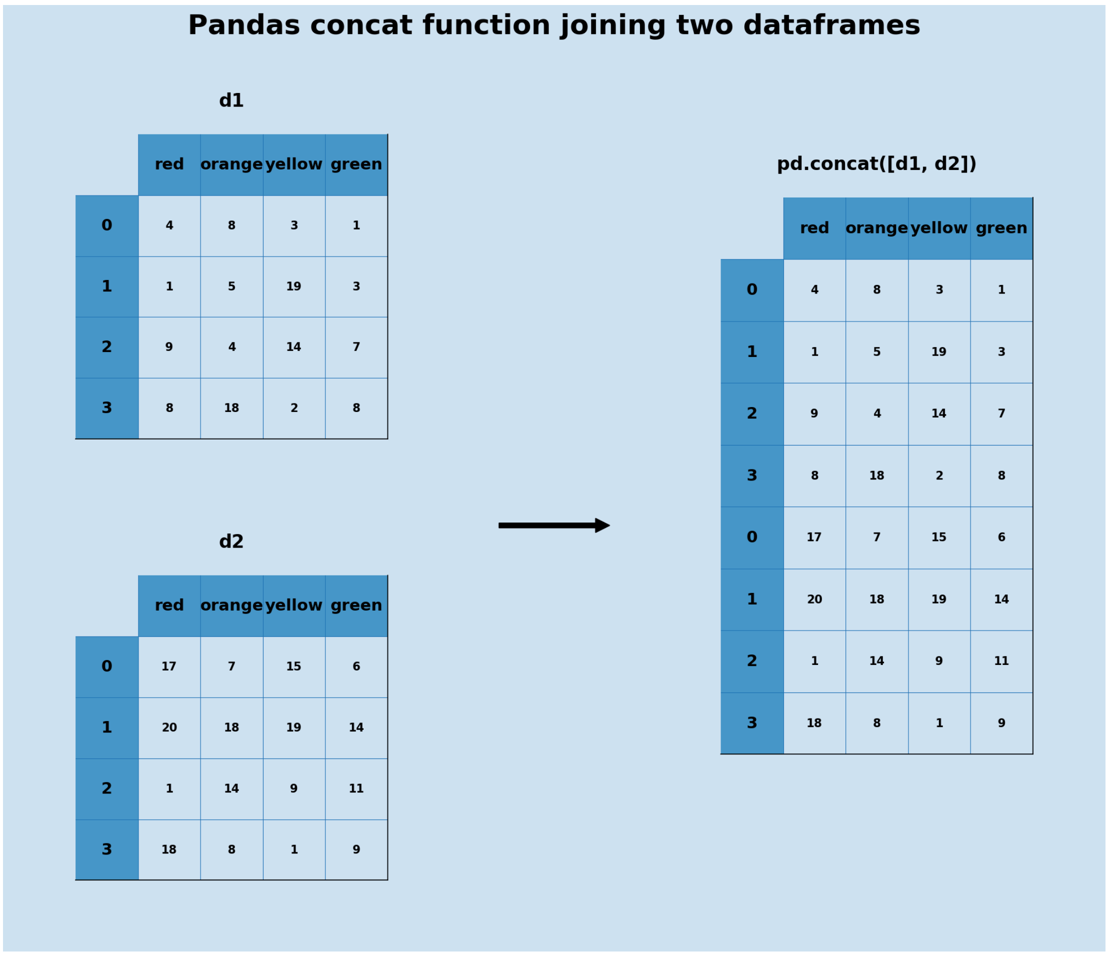
	date_wb	region_wb	category_wb	commodity	price_wb	date_gaza	region_gaza
21	1/15/2023	West Bank	vegetables and fruits	Onions (dry, local)	3.89	1/15/2023	Gaza Strip
22	1/15/2023	West Bank	cereals and tubers	Potatoes (medium size)	4.05	1/15/2023	Gaza Strip
23	1/15/2023	West Bank	cereals and tubers	Rice (small grain, imported)	149.78	1/15/2023	Gaza Strip
24	1/15/2023	West Bank	miscellaneous food	Salt	1.88	1/15/2023	Gaza Strip
25	1/15/2023	West Bank	miscellaneous food	Sugar	3.98	1/15/2023	Gaza Strip
26	1/15/2023	West Bank	miscellaneous food	Tea	16.51	1/15/2023	Gaza Strip
27	1/15/2023	West Bank	vegetables and fruits	Tomatoes (greenhouse)	3.93	1/15/2023	Gaza Strip
28	1/15/2023	West Bank	non-food	Water (drinking)	3.75	1/15/2023	Gaza Strip
29	1/15/2023	West Bank	cereals and tubers	Wheat flour	188.33	NaN	NaN
30	NaN	NaN	NaN	Wheat flour (locally processed)	NaN	1/15/2023	Gaza Strip
31	1/15/2023	West Bank	milk and dairy	Yogurt	6.02	1/15/2023	Gaza Strip

Concatenating dataframes:

`concat()`:

1. Concatenate date vertically:

The `concat` function in Pandas allows you to combine data from multiple DataFrames along either rows (vertical) or columns (horizontal). When you merge data vertically, you're stacking rows from one DataFrame on top of another, which is often useful when you have similar data split across different files or subsets and want to combine them into a single DataFrame for further analysis.



d1

	red	orange	yellow	green
0	4	8	3	1
1	1	5	19	3
2	9	4	14	7
3	8	18	2	8

d2

	red	orange	yellow	green
0	17	7	15	6
1	20	18	19	14
2	1	14	9	11
3	18	8	1	9

pd.concat([d1, d2])

	red	orange	yellow	green
0	4	8	3	1
1	1	5	19	3
2	9	4	14	7
3	8	18	2	8
0	17	7	15	6
1	20	18	19	14
2	1	14	9	11
3	18	8	1	9

How vertical concatenation works:

- All DataFrames **should ideally have the same columns**, as concat will align data based on column names. If a column is missing in any DataFrame, `NaN` values will fill those cells.
- By default, concat will **keep the indices from each DataFrame**, though you can reset or ignore the index.
- To concatenate data vertically, use `axis=0` as parameter to `concat` function

Example:

Suppose we have quarterly sales data for the same year stored in separate DataFrames:

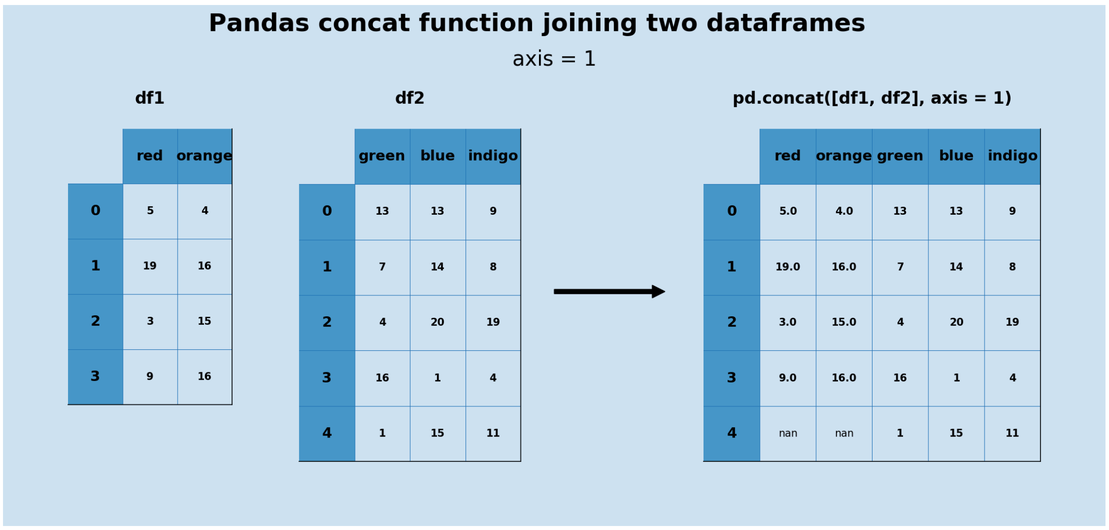
In [24]:
```python
import pandas as pd

# Sample DataFrames for quarterly sales
data_q1 = pd.DataFrame({'Product': ['A', 'B', 'C'], 'Sales': [500, 300, 200]})
data_q2 = pd.DataFrame({'Product': ['A', 'B', 'C'], 'Sales': [600, 350, 220]})

# Concatenate along rows
annual_sales = pd.concat([data_q1, data_q2], axis=0, ignore_index=True)
print(annual_sales)
```

```
  Product  Sales
0       A    500
1       B    300
2       C    200
3       A    600
4       B    350
5       C    220
```

This combines `data_q1` and `data_q2` vertically into a single DataFrame, making it easier to analyze total sales across the year. The `ignore_index=True` parameter resets the index to create a continuous sequence in the merged DataFrame.

2. Concatenate date horizontally:

In addition to stacking data vertically, the `concat` function can also be used to merge data horizontally by combining the DataFrames side-by-side, adding columns rather than rows. This approach is useful when you have related data split across DataFrames with a shared index or key, and you want to bring in additional columns.



Pandas concat function joining two dataframes
axis = 1
df1
df2
pd.concat([df1, df2], axis = 1)

	red	orange
0	5	4
1	19	16
2	3	15
3	9	16

	green	blue	indigo
0	13	13	9
1	7	14	8
2	4	20	19
3	16	1	4
4	1	15	11

	red	orange	green	blue	indigo
0	5.0	4.0	13	13	9
1	19.0	16.0	7	14	8
2	3.0	15.0	4	20	19
3	9.0	16.0	16	1	4
4	nan	nan	1	15	11

In horizental concatenation we use `axis=1` as parameter to `concat` function

Example:

Suppose you have two DataFrames, one with quarterly sales data and another with quarterly profit data for the same products:

In [25]:
```python
import pandas as pd

# Sample DataFrames
sales_data = pd.DataFrame({
    'Product': ['A', 'B', 'C'],
    'Sales_Q1': [500, 300, 200]
})
sales_data.set_index('Product', inplace=True)

profit_data = pd.DataFrame({
    'Product': ['A', 'B', 'C'],
    'Profit_Q1': [50, 30, 20]
})
```

```
profit_data.set_index('Product', inplace=True)

# Concatenate along columns (horizontal merge)
combined_data = pd.concat([sales_data, profit_data], axis=1)
combined_data
```

Out[25]:

Product	Sales_Q1	Profit_Q1
A	500	50
B	300	30
C	200	20

To consider:

- **When the index is different**: By default, concat aligns rows based on the index. If indices don't match, `NaN` values will fill in for missing entries.
- **When the rows are different**: If one DataFrame has rows that the other doesn't, the unmatched rows will have `NaN` values for the missing columns.
- **Duplicate columns**: When combining on `axis=1`, avoid duplicate column names, as concat will append `_x` and `_y` suffixes to duplicate names.