Combining Table

intro

- **What is combining tables?**
  - Combining tables means joining data from multiple sources into one cohesive dataset.
  - It is essential for real-world data analysis where data is often spread across multiple files or tables.
- **Why is it important?**
  - To **consolidate data** for analysis.
  - To handle **data from multiple sources** (e.g., two hospitals merging their records).
  - To prepare data for **further analysis, visualization, or machine learning**.
- **Two main approaches in pandas**:
  - concat: Combining tables **vertically** or **horizontally**.
  - merge: Combining tables based on **keys** (like SQL joins).

concat

- **When to use concat**:
  - When the tables have the **same columns** or the same structure.
  - When you want to **stack data** vertically (rows) or **align data** horizontally (columns).
- **How it works**:
  - pd.concat([df1, df2], axis=0) → Combines tables vertically (default behavior).
  - pd.concat([df1, df2], axis=1) → Combines tables horizontally.
- **Important Parameters**:
  - axis: 0 for rows (vertical), 1 for columns (horizontal).
  - ignore_index: Resets the index after concatenation.
  - keys: Adds hierarchical keys to identify the source of each table.

concat

```
pd.concat([df1,df2])
```

	a	b
0	1	1
1	2	4
2	3	9
0	2	4
1	4	16
2	6	36

```
pd.concat([df1,df2],
          ignore_index = True)
```

	a	b
0	1	1
1	2	4
2	3	9
3	2	4
4	4	16
5	6	36

```
pd.concat([df1,df2],
          keys = ['df1','df2'])
```

		a	b
df1	**0**	1	1
	1	2	4
	2	3	9
df2	**0**	2	4
	1	4	16
	2	6	36

merge

- **When to use merge**:
- When tables share **common columns** or **keys**.
- To combine data in a way similar to **SQL joins** (inner, outer, left, right).
- **How it works**:
- pd.merge(df1, df2, on='key') → Merges tables on a common key.
- pd.merge(df1, df2, left_on='key1', right_on='key2') → Merges tables with different key names.

merge

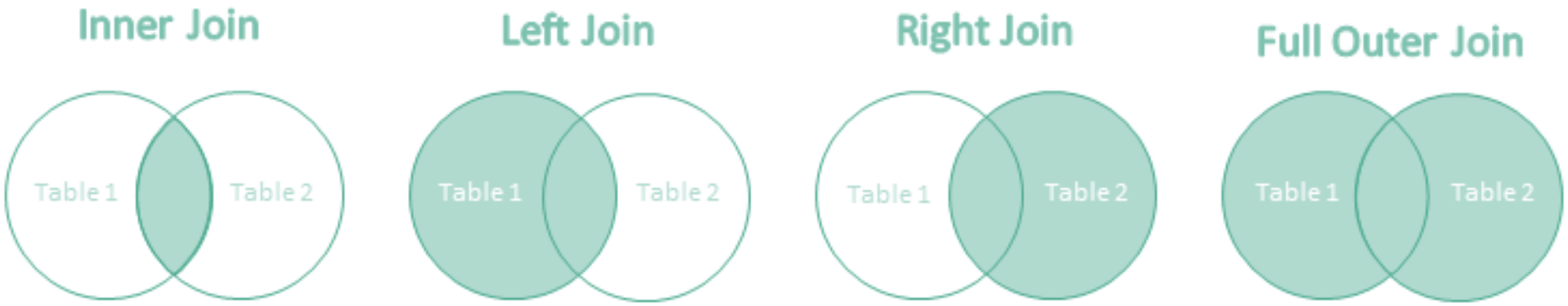
Types of Joins:

•**Inner Join**: Keeps only matching rows.

•**Outer Join**: Keeps all rows from both tables.

•**Left Join**: Keeps all rows from the left table.

•**Right Join**: Keeps all rows from the right table.



merge

Important Parameters:

- on: Specifies the key to merge on.
- how: Specifies the type of join

('inner', 'outer', 'left', 'right').

```
df1.merge(df2, how = "outer",indicator = True)
```

	a	b	_merge
0	1	1	left_only
1	2	4	both
2	3	9	left_only
3	4	16	right_only
4	6	36	right_only

merge

Important Parameters:

- suffixes: Adds suffixes to overlapping column names.

```
week_1.merge(week_2, how = "outer", on = ["cust_id"], suffixes = ["_wk_1", "_wk_2"], indicator = True)
```

	cust_id	item_id_wk_1	item_id_wk_2	_merge
0	6	NaN	1.0	right_only
1	7	5.0	NaN	left_only
2	8	NaN	9.0	right_only
3	9	7.0	NaN	left_only
4	11	3.0	NaN	left_only
...	...	...	...	...

Concat vs merge

Feature	concat	merge
Use Case	Combine tables vertically/horizontally	Combine tables based on keys
Flexibility	Less flexible	Highly flexible with joins
Key Matching	Not required	Required for joins
Behavior	Stacks or aligns data	Matches data using keys

Common Mistakes

- Confusing concat and merge:
  - Use concat for stacking or aligning data.
  - Use merge when combining data based on keys.
- Forgetting ignore_index in concat:
  - Without it, the index may not reset properly.
  - Not specifying how in merge:
  - Default join is inner, which may drop unmatched row

- **concat** is for simple stacking or aligning of tables.
- **merge** is for combining tables based on keys, like SQL joins.
- Choose the right method based on your data and analysis needs.