🧩 Merging in Pandas

🔹 What is Merging?

Merging combines two DataFrames based on common columns or keys, similar to JOINs in SQL. It's used to bring related data together — e.g., customer info + transaction history.

🔹 Common Function

```
pd.merge(df1, df2, on='key_column', how='join_type')
```

Parameter	Description
`df1`, `df2`	The two DataFrames you want to merge
`on`	The column(s) that both DataFrames share
`how`	The type of join: `'inner'`, `'left'`, `'right'`, `'outer'`

🔹 Types of Joins

Join Type	Keeps Rows From	Description	Example Output
`inner`	Both	Keeps only matching keys	Intersection
`left`	Left DataFrame	All rows from left, matching from right	Like SQL LEFT JOIN
`right`	Right DataFrame	All rows from right, matching from left	Like SQL RIGHT JOIN
`outer`	Both	All keys from both (fills missing with NaN)	Union

🔹 Example

```
import pandas as pd

# Example DataFrames
customers = pd.DataFrame({
    'cust_id': [1, 2, 3],
    'name': ['Ana', 'Ben', 'Cai']
})

orders = pd.DataFrame({
    'cust_id': [2, 3, 4],
    'order_total': [250, 400, 150]
})

# Inner join: only customers with matching orders
merged_df = pd.merge(customers, orders, on='cust_id', how='inner')
print(merged_df)
```

Output:

```
   cust_id  name  order_total
0        2   Ben          250
1        3   Cai          400
```

🔹 Common Parameters

Parameter	Usage	Example
`on='col'`	Same column name in both DataFrames	`pd.merge(df1, df2, on='id')`
`left_on`, `right_on`	Different column names	`pd.merge(df1, df2, left_on='cust_id', right_on='customer_id')`
`suffixes=('_x', '_y')`	Handle overlapping column names	`pd.merge(df1, df2, on='id', suffixes=('_old', '_new'))`
`indicator=True`	Track merge source	Adds `_merge` column: `'left_only'`, `'right_only'`, `'both'`
`validate='1:m'`	Check merge integrity	Options: `'1:1'`, `'1:m'`, `'m:1'`, `'m:m'`

🔹 Quick Examples

Left Join: Keep all customers (even without orders)

```
all_customers = pd.merge(customers, orders, on='cust_id', how='left')
# Result: Ana (NaN), Ben (250), Cai (400)
```

Right Join: Keep all orders (even with invalid customer IDs)

```
all_orders = pd.merge(customers, orders, on='cust_id', how='right')
# Result: Ben (250), Cai (400), Unknown customer 4 (150, name=NaN)
```

Outer Join: Keep everything

```
full_data = pd.merge(customers, orders, on='cust_id', how='outer')
# Result: Ana (NaN), Ben (250), Cai (400), Unknown customer 4 (150, name=NaN)
```

🔹 Notes

- Can't merge the tables → Look for the common key columns (e.g., `unique_id`, `order_id`)
- Use `on` for matching column names.
- Use `left_on` and `right_on` if the key columns have different names.
- Use `suffixes=('_left', '_right')` to handle overlapping column names.
- **Choose join type based on your question:**
  - "All customers regardless of activity?" → `how='left'`
  - "Only active customers?" → `how='inner'`
  - "Data quality check for orphaned records?" → `how='outer', indicator=True`