Normalise keys & join sources

Bring observations and reference data together while keeping unmatched records visible.

Synthetic example · independently invented data

One-page print edition · A3 landscape

01 Inspect the inputs

02 Apply the method

03 Check the result

INVENTED INPUTS

Two small input tables

Raw key	Value
“ UNIT-A ”	2
“unit-b”	5
“UNIT-D”	7
“unit-a”	3

Four synthetic observations; quotes show surrounding spaces

Key	Group
unit-a	Group 1
unit-b	Group 2
unit-c	Group 3

Three synthetic reference entries

Case and whitespace vary in the observations. Unit D has no entry in the reference table.

SELECTED PYTHON

The working calculation

Pandas is pd. rows and reference hold the two input tables at left.

```
observations = pd.DataFrame(rows)
lookup = pd.DataFrame(reference)
for frame in [observations, lookup]:
    frame['key'] = frame['key'].str.strip().str.lower()
joined = observations.merge(
    lookup, on='key', how='left',
    validate='many_to_one', indicator=True
)
unmatched = joined.loc[joined['_merge'] == 'left_only']
```

The code shown here is executed to calculate this example’s results.

COMPUTED RESULT

Keep the exception visible

Key	Value	Group
unit-a	2	Group 1
unit-b	5	Group 2
unit-d	7	Unmatched
unit-a	3	Group 1

Left-join result after key normalisation

3 + 1

matched + unmatched rows

All four observations are retained

The unmatched row remains in the result for investigation. Normalisation allows both Unit A records to match the same reference entry.

What this demonstrates

Key preparation, join direction and cardinality checks are separate design choices. Together they make the merge easier to verify.

Check the boundary

A duplicate reference key raises an error through many_to_one validation. An inner join would silently omit the unmatched observation from this output.

Thomas Eapen
nowlabs.net/tangerine-woa

New illustrative material explaining the approach. Values, configurations and results shown here are not historical workplace records.