D8A CHEATSHEET · #002 · PANDAS

The 8 pandas verbs that do 90% of it.

D8A.ACADEMY/RESOURCES/PANDAS-8-VERBS

01 · FILTER ROWS

`df[df.price > 100]`	boolean mask
`df.query("price > 100")`	same, reads better

02 · SELECT COLUMNS

`df[['name', 'price']]`	by name
`df.loc[rows, cols]`	by label
`df.iloc[0:5, 0:2]`	by position

03 · SORT

`df.sort_values('price')`	ascending by default
`ascending=False`	flip it

04 · TRANSFORM

`df.assign(tax=df.price * 0.2)`	new column, chainable
`df['col'].astype(int)`	cast a type

05 · GROUP

`df.groupby('city')`	split into groups
`.size()`	rows per group

06 · AGGREGATE

`.agg(avg=('price', 'mean'))`	named result columns
`.agg(['min', 'max'])`	several at once

07 · JOIN

`df.merge(other, on='id')`	SQL INNER JOIN
`how='left'`	SQL LEFT JOIN

08 · RESHAPE

`df.pivot_table(...)`	long to wide
`df.melt(...)`	wide to long

→ these 8 verbs cover 90% of real notebook work

WORTH MEMORIZING

01 The workhorse pattern

Split, aggregate, rank. The most typed pattern in analytics.

```
revenue = (df
  .groupby('city')
  .agg(orders=('id', 'count'),
       revenue=('amount', 'sum'))
  .reset_index()
  .sort_values('revenue', ascending=False))
```

→ Named aggregations keep your output columns readable: new_name=('col', 'fn').

02 merge is just SQL

df.merge is SQL joins wearing a Python coat.

```
# a LEFT JOIN, pandas edition
orders.merge(customers,
  on='customer_id',
  how='left')

# unmatched rows get NaN
```

→ how= takes inner, left, right, outer. Same semantics as SQL, and the same row fan-out trap (see sheet 001).

03 Trap: the silent copy

df[df.x > 0]['y'] = 1 changes nothing, and tells you almost nothing.

```
# looks fine, writes into a temporary copy
df[df.x > 0]['y'] = 1

# selection + write in one step
df.loc[df.x > 0, 'y'] = 1
```

→ Chained indexing creates an intermediate object; the assignment lands on the copy. .loc does it in place.