pandas Python data frames

Read read_csv, read_parquet → inspect → **Clean** select, astype, fillna → transform → **Shape** groupby, merge, pivot → export → **Write** to_parquet, to_csv

CONCEPTS

A DataFrame is a set of Series (columns) sharing one index (row labels). Most operations align on that index, not on position.

Copy-on-Write
pandas 3 uses Copy-on-Write: any derived frame or column acts as a copy. Chained assignment like df["a"][0] = 1 never changes df; use df.loc[0, "a"] = 1.

READ AND WRITE

`pd.read_csv("<file>")`
Read a CSV file

`pd.read_parquet("<file>")`
Read a Parquet file

`df.to_csv("<file>", index=False)`
Write CSV without the index

`df.to_parquet("<file>")`
Write Parquet (keeps dtypes)

INSPECT

`df.head(<n>) / df.tail(<n>)`
First or last rows (default 5)

INSPECT (cont.)

`df.info()`
Columns, dtypes, non-null counts

`df.describe()`
Summary statistics

`df.shape`
(rows, columns)

`df.dtypes`
Type of each column

`df["<col>"].value_counts()`
Count each distinct value

SELECT

`df["<col>"]`
One column as a Series

`df[["<a>", "<b>"]]`
Several columns as a DataFrame

`df.loc[<rows>, <cols>]`
Select by label

`df.iloc[<rows>, <cols>]`
Select by position

`df[df["<col>"] > 0]`
Filter rows with a mask

`df[(<cond1>) & (<cond2>)]`
Combine conditions

`df[df["<col>"].isin([...])]`
Rows whose value is in a list

CHANGE COLUMNS

`df["<new>"] = df["<a>"] * 2`
Add or overwrite a column

`df.assign(<new>=lambda d: d.<a> + d.<b>)`
Add columns, return a new frame

`df.loc[<mask>, "<col>"] = <value>`
Set values where a condition holds

`df.rename(columns={"<old>": "<new>"})`
Rename columns

`df.drop(columns=["<col>"])`
Remove columns

`df["<col>"].astype("<dtype>")`
Convert type (int64, float, str, category)

MISSING DATA

`df.isna().sum()`
Count missing values per column

`df.dropna()`
Drop rows with any missing value

`df.fillna(<value>)`
Fill missing values

SORT AND DEDUPE

`df.sort_values("<col>", ascending=False)`
Sort rows by a column

`df.drop_duplicates(subset=["<col>"])`
Drop duplicate rows

`df.set_index("<col>")`
Use a column as row labels

`df.reset_index(drop=True)`
Renumber rows 0..n-1

GROUP AND AGGREGATE

`df.groupby("<key>")["<col>"].sum()`
Total per group

`df.groupby("<key>").agg(<out>=("<col>", "sum"))`
Named aggregation

COMBINE

`df.merge(<other>, on="<key>")`
Join on a column (inner)

`df.merge(<other>, on="<key>", how="left")`
Left join

`pd.concat([<df1>, <df2>], ignore_index=True)`
Stack frames vertically

RESHAPE

`df.pivot_table(index="<r>", columns="<c>", values="<v>", aggfunc="sum")`
Spreadsheet-style pivot

`df.melt(id_vars=["<id>"], var_name="<k>", value_name="<v>")`
Wide to long

TEXT

`df["<col>"].str.lower()`
Lowercase (any str method)

`df["<col>"].str.contains("<pat>", na=False)`
Regex match as a mask

DATES AND TIME SERIES

`pd.to_datetime(df["<col>"])`
Parse text as datetimes

`df["<col>"].dt.year`
Date part
(also .month, .day_name())

APPLY VS VECTORISED

`df["<a>"] + df["<b>"]`
Column arithmetic, no loops