VIRTAVIS LEARNING

FREE CHEAT SHEET

pandas for Data Analytics

The commands you need to load, clean, summarise and chart data in Python.

START WITH

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

GOLDEN RULE

Most methods return a **new** table. Save the result: `df = df.dropna()`

DATA STRUCTURES

The two objects you work with.

Series one labelled column

```
s = pd.Series([10, 20, 30],
  index=["a", "b", "c"])
```

a	10
b	20
c	30

DataFrame a table of Series

```
df = pd.DataFrame({
  "Name": ["Amy", "Ben"],
  "Age": [25, 30]})
```

	Name	Age
0	Amy	25
1	Ben	30

READ AND WRITE DATA

Load files, save results.

Read CSV	`pd.read_csv("sales.csv")`
Read Excel	`pd.read_excel("sales.xlsx")` Needs openpyxl. sheet_name= picks a tab
Read SQL	`pd.read_sql(query, con)` con = a database connection
Parquet	`pd.read_parquet("sales.parquet")`
Save as CSV	`df.to_csv("out.csv", index=False)` index=False skips the row numbers
Save as Excel	`df.to_excel("out.xlsx", index=False)`

INSPECT YOUR DATA

Look before you analyse.

First / last	`df.head()` `df.tail(10)` 5 rows unless you say
Random rows	`df.sample(5, random_state=42)`
Names, types	`df.columns` `df.dtypes`
Count values	`df["Region"].value_counts()` dropna=False also counts blanks
Unique values	`df["Region"].unique()` `df.nunique()`

SORT AND RANK

Put things in order.

Sort	`df.sort_values("Sales", ascending=False)` Largest first. Leave it out for smallest
Sort by two	`df.sort_values(["Region", "Sales"], ascending=[True, False])`
Rank	`df["Rank"] = df["Sales"].rank(ascending=False)` 1 = highest
Top 5 rows	`df.nlargest(5, "Sales")` df.nsmallest() for the bottom 5

SELECT AND FILTER

Pick the rows and columns you need.

One column	`df["Sales"]` Returns a Series
Several columns	`df[["Name", "Sales"]]` Returns a DataFrame
By position	`df.iloc[0:5]` Rows 0 to 4: end is excluded
By label	`df.loc["A":"D"]` Labels A to D: end is included
One cell	`df.loc[0, "Name"]`
One condition	`df[df["Sales"] > 100]`
Two conditions	`df[(df["Region"] == "UK") & (df["Sales"] > 100)]` Brackets round each. & = and, | = or
Match a list	`df[df["Region"].isin(["UK", "US"])]`

CLEAN AND PREPARE

Fix gaps, duplicates and types.

Drop blanks	`df = df.dropna(subset=["Sales"])` No subset = any blank in any column
Fill blanks	`df["Sales"] = df["Sales"].fillna(0)` Only when 0 is really true
Drop duplicates	`df = df.drop_duplicates()`
Rename	`df = df.rename(columns={"old": "new"})`
Text to dates	`df["Date"] = pd.to_datetime(df["Date"], dayfirst=True)` dayfirst=True for UK dates like 05/03
Text to numbers	`df["Sales"] = pd.to_numeric(df["Sales"], errors="coerce")` Values that fail become NaN
Trim spaces	`df["Name"] = df["Name"].str.strip()`
Replace values	`df["Status"] = df["Status"].replace({"Old": "New"})`

RESHAPE

Cross-tabs and pivots.

Pivot table	`df.pivot_table(values="Sales", index="Region", columns="Product", aggfunc="sum")`
Cross-tab	`pd.crosstab(df["Region"], df["Product"])` Counts each combination
Index to column	`df = df.reset_index()` Handy after a groupby

CREATE COLUMNS

Calculate new fields.

Calculate	`df["Profit"] = df["Sales"] - df["Cost"]`
If / else	`df["Band"] = np.where(df["Sales"] > 100, "High", "Low")`
Map values	`df["Area"] = df["Region"].map({"UK": "Europe", "US": "Americas"})` Values not in the map become NaN
Bin into groups	`df["Band"] = pd.cut(df["Age"], bins=[0, 17, 30, 50], labels=["0-17", "18-30", "31-50"])` Right edge included: (17, 30] = 18-30
Text case	`df["Name"] = df["Name"].str.upper()`
Row function	`df["Score"] = df.apply(f, axis=1)` Slow: try maths on whole columns first

GROUP AND SUMMARISE

Totals by category.

One total per group

```
df.groupby("Region")["Sales"].sum()
```

Region	Sales
CA	760
UK	1,250
US	980

Groups come back sorted A to Z. Add `sort=False` to keep the original order.

Several summaries at once (named aggregation)

```
df.groupby("Region").agg(
  Total=("Sales", "sum"),
  Average=("Sales", "mean"),
  Orders=("OrderID", "nunique"))
```

Region	Total	Average	Orders
CA	760	95	8
UK	1,250	125	10
US	980	98	10

Check it adds up: Total = Average × Orders when each row is one order.

CHECK BEFORE YOU TRUST IT

Five checks on any new data.

`df.shape`	**Right size?** Rows and columns match the source?
`df.info()`	**Right types?** Numbers or dates stored as text need converting.
`df.isna().sum()`	**Any gaps?** Blanks per column. Drop, fill or flag them.
`df.duplicated().sum()`	**Any repeats?** Duplicate rows double-count sales.
`df.describe()`	**Sensible values?** Negative quantities? A max of 99999?

MERGE AND COMBINE

Join tables on a shared key.

Join two tables like a lookup or SQL join

```
pd.merge(customers, orders,
  on="ID", how="left")
```

ID	Name
1	Amy
2	Ben
3	Cara

+

ID	Order
1	100
2	200

=

ID	Name	Order
1	Amy	100
2	Ben	200
3	Cara	NaN

Cara has no order, so a left join keeps her with NaN. Compare row counts before and after.



Stack rows	`pd.concat([df1, df2], ignore_index=True)` Same columns, one under the other
Side by side	`pd.concat([df1, df2], axis=1)` Lines rows up by index

DATES AND TIME

Parse, extract, summarise.

Year / month	`df["Year"] = df["Date"].dt.year` `df["Month"] = df["Date"].dt.month`
Day name	`df["Day"] = df["Date"].dt.day_name()`
Month label	`df["Date"].dt.to_period("M")` Year and month, e.g. 2025-03
Date range	`df[df["Date"].between("2025-01-01", "2025-12-31")]` Both ends in, but 31 Dec 10:00 is left out
Monthly totals	`monthly = (df.set_index("Date")["Sales"].resample("ME").sum())` ME = month end. Old "M" errors in pandas 3

VISUALISE

Quick charts with .plot().

Bar chart	`(df.groupby("Region")["Sales"].sum().plot(kind="bar"))` Compare groups
Line chart	`monthly.plot(kind="line")` Trend over time
Histogram	`df["Sales"].plot(kind="hist", bins=20)` See the spread
Box plot	`df.boxplot(column="Sales", by="Region")` Spot outliers
Scatter	`df.plot(kind="scatter", x="Price", y="Qty")` Do two numbers move together?

Check the numbers. Then say what they mean.

By **Sophie Lumpa**, Data and Business Intelligence Specialist. Every snippet tested in pandas 3.0 (works in 2.2+). © 2026 Virtavis Learning · Free to share with credit

Start the course › Python for Data Analytics

VIRTAVIS virtavis.com/courses