PYTHON

pandas

FULL COURSE

Clean • Analyze • Report



LONG VIDEO • 13 MIN

PYTHON • PANDAS FULL COURSE

Python pandas Full Course

DataFrames, cleaning, grouping, merging, dates and real reports

FREE FULL COURSE • MyWebUniversity.com



Wahidullah Lutfy
Founder, MyWebUniversity • CS Teacher

PANDAS

What You Will Learn

Series and DataFrames
The two building blocks of pandas.

Load and inspect
CSV, Excel and more; head, info and describe.

Select and filter
Columns, loc, iloc and conditions.

Clean
Missing values, types, duplicates and text.

Analyze
groupby, pivot tables, merging and dates.

Report
Charts and exports to Excel and CSV.

Python pandas • Full Course

PART 1 · THE BASICS

PANDAS

Series and DataFrames

Python 3.13

```
$ pip install pandas openpyxl
>>> import pandas as pd
>>> pd.__version__
'3.0.5'


>>> s = pd.Series([1200, 950, 1100], index=["Kabul", "Fresno", "Paris"])
>>> s["Fresno"], s.mean()
(950, 1083.33)



>>> df = pd.DataFrame({"city": ["Kabul", "Fresno"], "sales": [1200,
950]})
>>> df
     city  sales
0   Kabul   1200
1  Fresno    950
```

Series

One labeled column of data: values plus an index.

DataFrame

A table: several Series that share the same index.

Built on NumPy

Each column is an array, so math is fast and vectorized.

PANDAS

Loading and Inspecting Data

Python 3.13

```
>>> df = pd.read_csv("sales.csv")
>>> df.head(3)
     city month   sales   rep
0   Kabul   Jan  1200.0   Ali
1  Fresno   Jan   950.0  Sara

>>> df.shape
(6, 4)
>>> df.info()
 2   sales   5 non-null   float64

>>> df.describe()
>>> df["city"].value_counts()
```

Read anything

read_csv, read_excel, read_json, read_sql, read_html, read_parquet.

First look

head, shape, info, describe and value_counts, every single time.

info() tells the truth

Column types and non-null counts reveal missing data immediately.

PANDAS

Selecting Rows and Columns

Python 3.13

```
>>> df["sales"]              # one column -> Series
>>> df[["city", "sales"]]    # several columns -> DataFrame

>>> df.loc[0, "city"]
'Kabul'
>>> df.loc[df["sales"] > 1100, ["city", "sales"]]
     city   sales
0   Kabul  1200.0
4  Fresno  1300.0

>>> df.iloc[0:2, 0:2]          # by position
>>> df.query("city == 'Paris' and sales > 1000")
```

Brackets

df["col"] gives one column; df[["a", "b"]] gives several.

loc vs iloc

loc uses labels and conditions; iloc uses integer positions.

query()

Write filters like a sentence. Combine conditions with and / or.

Python pandas • Full Course

PART 2 · CLEANING DATA

PANDAS

Missing Values and Data Types

Python 3.13

```
>>> df.isna().sum()
sales    1
>>> df["sales"] = df["sales"].fillna(df["sales"].mean())

>>> df = df.dropna(subset=["city"])
>>> df = df.drop_duplicates()

>>> df["date"] = pd.to_datetime(df["date"])
>>> df["units"] = pd.to_numeric(df["units"], errors="coerce")
>>> df["city"] = df["city"].astype("category")
```

Find, then decide

isna().sum() finds gaps. Fill with a value, or drop the row: it depends on the meaning.

Duplicates

drop_duplicates() stops double-counting.

Fix types

to_datetime, to_numeric(errors="coerce") and astype make columns behave correctly.

PANDAS

New Columns, Text and apply

Python 3.13

```
>>> df["tax"] = df["sales"] * 0.08
>>> df["big_sale"] = df["sales"] > 1150

>>> df["city"].str.upper().head(2).tolist()
['KABUL', 'FRESNO']
>>> df["rep"].str.strip().str.title()
>>> df[df["city"].str.contains("ka", case=False)]

>>> df["level"] = df["sales"].apply(lambda s: "high" if s > 1150 else
"normal")
>>> df.loc[df["city"] == "Paris", "region"] = "Europe"
```

Vectorized columns

Math and comparisons on whole columns create new columns instantly.

The .str accessor

upper, strip, title, contains, replace, split: text cleaning for whole columns.

Assign with .loc

In pandas 3, always set values with df.loc[condition, "col"] = value.

Python pandas • Full Course

PART 3 · ANALYSIS AND REPORTING

PANDAS

groupby: Split, Apply, Combine

Python 3.13

```
>>> df.groupby("city")["sales"].agg(["sum", "mean"])
           sum    mean
Fresno  2250.0  1125.0
Kabul   2300.0  1150.0
Paris   2410.0  1205.0

>>> df.groupby(["city", "month"])["sales"].sum()

>>> (df.groupby("rep")["sales"].sum()
...     .sort_values(ascending=False).head(3))
```

Split, apply, combine

Split rows into groups, calculate for each group, combine the results.

agg()

Several statistics at once: sum, mean, count, min, max, median.

Chain it

groupby, then sort_values, then head: a top-3 report in one expression.

PANDAS

Pivot Tables and Merging

Python 3.13

```
>>> df.pivot_table(index="city", columns="month", values="sales",
aggfunc="sum")
month      Feb     Jan     Mar
Fresno     NaN   950.0  1300.0
Kabul   1100.0  1200.0     NaN

>>> reps = pd.DataFrame({"rep": ["Ali", "Sara", "Marie"], "region":
["Asia", "USA", "Europe"]})
>>> full = df.merge(reps, on="rep", how="left")

>>> all_years = pd.concat([sales_2025, sales_2026], ignore_index=True)
```

pivot_table

Rows, columns and a summary, exactly like an Excel pivot table.

merge

Combine tables on a shared key, like a SQL JOIN: inner, left, right, outer.

concat

Stack tables with the same columns on top of each other.

PANDAS

Working with Dates

Python 3.13

```
>>> df["date"] = pd.to_datetime(df["date"])
>>> df["date"].dt.month_name().head(2).tolist()
['January', 'February']

>>> df["weekday"] = df["date"].dt.day_name()
>>> df[df["date"] >= "2026-02-01"]

>>> monthly = df.set_index("date")["sales"].resample("ME").sum()
>>> monthly.rolling(3).mean()
```

The .dt accessor

year, month, month_name, day_name, quarter, and more.

Compare dates

Filter with date strings like "2026-02-01".

resample and rolling

Totals per month or week, and moving averages to smooth trends.

PANDAS

Mini Project: A Monthly Sales Report

report.py — Python 3.13

```
import pandas as pd
df = pd.read_csv("sales.csv", parse_dates=["date"])
df = df.dropna(subset=["sales"]).drop_duplicates()

by_city = df.groupby("city")["sales"].sum().sort_values(ascending=False)
ax = by_city.plot(kind="bar", title="Sales by City")
ax.figure.savefig("sales_by_city.png", dpi=150)


with pd.ExcelWriter("report.xlsx") as xl:
   by_city.to_excel(xl, sheet_name="By City")
   df.to_excel(xl, sheet_name="Clean Data", index=False)
```

Load and clean

parse_dates converts dates while reading; then drop bad rows.

Analyze and chart

.plot() uses Matplotlib behind the scenes. Save it as a PNG.

Export

ExcelWriter writes several sheets into one workbook.

PANDAS

Recap and Practice

Remember

Load, inspect, clean, analyze, report. Always look before you trust.

Try it yourself

Download a public CSV dataset and find the top 5 groups by total.

Watch out

Use .loc for assignments, and check types with info().