

Working with Data in pandas: Load, Explore, Clean and Summarise

Build and load tables, look at them, filter rows, sort, add calculated columns, summarise by group, handle missing values and draw a quick chart with pandas.

20 min read · Content last verified 20 October 2026 · Reviewed by Shravya M

Free awareness guide. Not legal or financial advice. <https://www.shravyadigital.com/guides/python-data-with-pandas>

CONTENTS

1. Before you start
2. Chapter 1: Tables in Python: the DataFrame
3. Chapter 2: Looking at your data
4. Chapter 3: Choosing rows: filtering
5. Chapter 4: Sorting and new columns
6. Chapter 5: Summarising by group
7. Chapter 6: Missing values and a quick chart
8. Chapter 7: From data to AI
9. Key words

Before you start

This guide is the reading for **Course 3** of the Python for AI Pathway. It takes about 20 minutes and builds on Courses 1 and 2. Open a Google Colab notebook and try each example.

AI runs on data, and the tool almost everyone uses to work with data in Python is **pandas**. You'll learn to load a table, look at it, pick out rows, sort, summarise by group and draw a simple chart: the same jobs people do in Excel, but repeatable and able to handle millions of rows.

All names and figures here are made up.

Chapter 1: Tables in Python: the DataFrame

pandas stores a table as a **DataFrame**: rows and named columns, like a spreadsheet. First, **import** the library. Colab already has it installed.

```
import pandas as pd

sales = pd.DataFrame({
    "month": ["Jan", "Jan", "Feb", "Feb", "Mar", "Mar"],
    "shop": ["Pune", "Nagpur", "Pune", "Nagpur", "Pune", "Nagpur"],
    "amount": [52000, 31000, 61000, 29000, 58000, 40000],
})

print(sales)
```

Output:

	month	shop	amount
0	Jan	Pune	52000
1	Jan	Nagpur	31000
2	Feb	Pune	61000
3	Feb	Nagpur	29000
4	Mar	Pune	58000
5	Mar	Nagpur	40000

`import pandas as pd` loads pandas under the short name `pd`. Each column is called a **Series**. The numbers on the left are the **index**, the row labels.

In real work you'd load a file instead, for example `pd.read_csv("sales.csv")` for a CSV file or `pd.read_excel("sales.xlsx")` for Excel.

Chapter 2: Looking at your data

Before doing anything, look:

```
print(sales.shape)
print(sales.columns.tolist())
print(sales.head(2))
```

Output:

```
(6, 3)
['month', 'shop', 'amount']
   month  shop  amount
0   Jan   Pune  52000
1   Jan  Nagpur  31000
```

- `shape` gives (rows, columns).
- `columns` lists the column names.
- `head(n)` shows the first n rows; `tail(n)` shows the last.
- `describe()` gives quick statistics for number columns, and `info()` shows each column's type and any missing values.

Pick one column with its name in square brackets, then summarise it:

```
print(sales["amount"].sum())
print(sales["amount"].mean())
```

Output:

```
271000
45166.666666666664
```

Chapter 3: Choosing rows: filtering

To keep only some rows, put a condition in square brackets:

```
big = sales[sales["amount"] > 50000]
print(big)
```

Output:

	month	shop	amount
0	Jan	Pune	52000
2	Feb	Pune	61000
4	Mar	Pune	58000

Combine conditions with `&` (and) or `|` (or), each in brackets:

```
print(sales[(sales["shop"] == "Nagpur") & (sales["amount"] > 30000)])
```

Output:

	month	shop	amount
1	Jan	Nagpur	31000
5	Mar	Nagpur	40000

Chapter 4: Sorting and new columns

Sort with `sort_values` :

```
print(sales.sort_values("amount", ascending=False).head(3))
```

Output:

	month	shop	amount
2	Feb	Pune	61000
4	Mar	Pune	58000
0	Jan	Pune	52000

Create a new column from existing ones, calculated for every row at once:

```
sales["gst"] = sales["amount"] * 0.18  
print(sales[["shop", "amount", "gst"]].head(2))
```

Output:

	shop	amount	gst
0	Pune	52000	9360.0
1	Nagpur	31000	5580.0

No loop needed: pandas works on whole columns at once, which is both simpler and much faster.

Chapter 5: Summarising by group

The most useful question in data work is "what's the total (or average) for each group?".

That's `groupby` :

```
by_shop = sales.groupby("shop")["amount"].sum()
print(by_shop)
```

Output:

```
shop
Nagpur    100000
Pune      171000
Name: amount, dtype: int64
```

Swap `sum()` for `mean()`, `count()`, `max()` or `min()` for other summaries. This is the same idea as a pivot table in Excel.

Chapter 6: Missing values and a quick chart

Real data has gaps. pandas shows a missing value as `NaN` ("not a number"):

```
import numpy as np

marks = pd.DataFrame({"student": ["Asha", "Ravi", "Meera"], "score": [78, np.nan, 91]})
print(marks["score"].isna().sum())
print(marks["score"].fillna(0).tolist())
```

Output:

```
1
[78.0, 0.0, 91.0]
```

- `isna()` finds missing values; `.sum()` counts them.
- `fillna(value)` fills them in; `dropna()` removes rows that have them.
- Decide carefully: filling a missing mark with 0 changes the average. Sometimes removing the row, or asking where the gap came from, is the honest choice.

A chart takes one line. In Colab it appears below the cell:

```
chart = by_shop.plot(kind="bar", title="Sales by shop")
```

Chapter 7: From data to AI

Everything in this course is how AI work actually begins:

- **Collect and load** the data (`read_csv`).
- **Look at it** (`head` , `describe` , `info`) and **clean it** (missing values, wrong types).
- **Choose the useful columns** (filtering and new columns).
- **Summarise** to understand it (`groupby` , charts).

Data scientists often say most of their time goes on this preparation. A model trained on messy or one-sided data gives bad results, as you saw in the AI Foundations Pathway: "garbage in, garbage out". In Course 4 you'll hand a clean table like this to a machine learning model.

The rule: Look at your data before you trust it, and before a model does.

Key words

- **pandas:** the Python library for working with tables of data.
- **DataFrame:** a table with rows and named columns.
- **Series:** one column of a DataFrame.
- **Index:** the row labels.
- **read_csv / read_excel:** load a file into a DataFrame.
- **head / shape / describe / info:** quick ways to look at data.
- **Filtering:** keeping only rows that meet a condition.
- **groupby:** summarise values for each group.
- **NaN:** a missing value.
- **fillna / dropna:** fill in or remove missing values.

Take the course for what you just read

Take the free 10-question assessment and pass with 8 or more. Every course you pass counts towards a Pathway Certificate issued by Shravya Digital, earned when you pass all five courses in the pathway: <https://digital.shravyaworld.org/courses/python-data-with-pandas>

Sources

Checked against these primary sources on 20 October 2026. Rules change: if something here looks out of date, trust the source.

- pandas documentation: 10 minutes to pandas (https://pandas.pydata.org/docs/user_guide/10min.html)
- pandas documentation: group by (https://pandas.pydata.org/docs/user_guide/groupby.html)
- pandas documentation: working with missing data (https://pandas.pydata.org/docs/user_guide/missing_data.html)