Data Types and Cleaning Data

Real data is messy, and every value has a type. Numbers, text, dates and true/false values, plus the cleaning steps that fix missing values, duplicates and wrong formats.

Introduction

Before you can study data, you must know what kind of data you are holding. A computer treats the number `85` and the text `"85"` very differently: it can add numbers, but it cannot add text. If a marks column is stored as text by mistake, your average will fail — not because the maths is hard, but because the type is wrong. That is why data types come first.

Then comes the truth nobody warns beginners about: real data is almost never clean. Someone skipped a field. Someone typed the same row twice. Someone wrote a date as "12/3" and someone else as "March 12". Cleaning is not boring extra work. Cleaning is the work that makes every later answer trustworthy.

In this lesson you will learn the four common types, the four common problems, and a simple cleaning order you can use on any small table.

One more reason types matter: mistakes caused by wrong types are often silent. If marks are stored as text, sorting them gives "100, 85, 92" in that order — because as text, "1" comes before "8", so 100 looks smaller than 85. No error appears, but every ranking is wrong. Dates cause the same trap: "12/3/2026" could mean 12 March or December 3, depending on who wrote it. Learning to spot the type of every column, before any maths, is what protects you from these quiet mistakes.

(Roman Urdu: Asli data kabhi saaf nahi hota — kuch khaali hota hai, kuch do baar likha hota hai, kuch ghalat format mein hota hai. Safai ke baghair har jawab ghalat ho sakta hai.)

Key points

- **Numbers** are for maths: marks, prices, ages, counts. Examples: `85`, `49.5`, `0`.
- **Text (strings)** are words and labels: names, cities, subjects. Examples: `"Ali"`, `"Maths"`, `"Lahore"`.
- **Dates** are points in time: birth dates, sale dates, exam dates. They need one fixed format, or sorting and comparing them breaks.
- **True/false (booleans)** are yes-or-no values: passed or failed, present or absent. Examples: `True`, `False`.
- **Missing values** are empty cells — a student with no maths mark, a sale with no price. You must decide: fill it, or leave that row out, and say which you did.
- **Duplicates** are the same row entered twice. They double-count everything: two copies of one sale look like two sales.
- **Wrong formats** are the same fact written different ways: `"85"` as text, `"12/3/2026"` beside `"March 12, 2026"`, `"ali"` beside `"Ali"`.

- **Cleaning order that works:** fix types first, remove exact duplicates, handle missing values, fix spellings and formats, then check the result by eye.
- **Categories are a special kind of text.** A column like subject or city takes only a few repeated values — Maths, English, Urdu — not free sentences. Counting categories (how many of each) is one of the first things you do with such a column.
- **In pandas, `df.dtypes` shows every column's type at once.** Run it the moment you load a table, and fix any column whose type surprises you before you go further.

Examples

Example 1 — Spot the types in a small table

```
Name     | Marks | Passed | Exam date
Ali      | 85    | True   | 2026-03-12
Sara     | 92    | True   | 2026-03-12
Bilal    | (empty) | False | 12 March 2026
Ali      | 85    | True   | 2026-03-12
```

What this means: `Marks` is a number, `Name` is text, `Passed` is true/false, and `Exam date` is a date — but written two ways. The table also has one missing mark and one duplicate row. All four problems sit in four rows. That is normal for real data.

Example 2 — Cleaning the same table, step by step

Plain-Python version — no installs needed:

```
students = [
    {"name": "Ali", "marks": 85, "passed": True, "date": "2026-03-12"},
    {"name": "Sara", "marks": 92, "passed": True, "date": "2026-03-12"},
    {"name": "Bilal", "marks": None, "passed": False, "date": "12 March 2026"},
    {"name": "Ali", "marks": 85, "passed": True, "date": "2026-03-12"},
]

# Step 1: remove exact duplicates
seen = []
clean = []
for row in students:
    if row not in seen:
        seen.append(row)
        clean.append(row)

# Step 2: fix the date format to one style
for row in clean:
    if row["date"] == "12 March 2026":
        row["date"] = "2026-03-12"

print(clean)
print("Rows left:", len(clean))
```

What this means: the duplicate Ali row is gone, and both dates now match. Bilal's missing mark is still `None` — you must now decide: drop Bilal's row for a marks average, or find his real mark. Either choice is fine, but you must say which one you made.

Example 3 — The same cleaning with pandas

This example needs `pip install pandas` first. It uses only standard pandas calls:

```
import pandas as pd

df = pd.DataFrame([
    {"name": "Ali", "marks": 85},
    {"name": "Sara", "marks": 92},
    {"name": "Bilal", "marks": None},
    {"name": "Ali", "marks": 85},
])

df = df.drop_duplicates()
print(df.isna().sum())   # counts missing values per column
df = df.dropna()         # drops rows with a missing value
print(df)
print(df["marks"].mean())  # average of the marks left
```

What this means: `drop_duplicates()` removes the repeated row, `isna().sum()` shows you where the gaps are, and `dropna()` removes rows with gaps. The average here is 88.5 — from Ali and Sara only. If you had averaged without cleaning, the duplicate would have pulled the answer toward Ali's mark.

Example 4 — Turning text numbers into real numbers

A marks column arrived as text, with one value that is not a number at all:

```
import pandas as pd

marks = pd.Series(["85", "92", "not given", "88"])
clean = pd.to_numeric(marks, errors="coerce")
print(clean)
```

Output:

```
0    85.0
1    92.0
2     NaN
3    88.0
```

What this means: `errors="coerce"` turns anything it cannot read as a number into `NaN` (missing) instead of crashing. Now the column is truly numeric, and the one bad value is visible as a gap you must decide about — drop it, or find the real mark — instead of hiding inside text.

Example 5 — Filling a gap with the average, and saying so

```
import pandas as pd

df = pd.DataFrame({"name": ["Ali", "Sara", "Bilal", "Ayesha"],
                   "marks": [85, 92, None, 88]})

average = df["marks"].mean()
print("Average of known marks:", average)  # 88.33

df["marks"] = df["marks"].fillna(average)
print(df)
print("Average after filling:", df["marks"].mean())  # 88.33, unchanged
```

What this means: filling with the average keeps the overall average the same, which is why it is a popular choice — but it is still a guess, and it hides the fact that one mark was never recorded. It is fine to do, as long as your write-up says: "1 missing mark filled with the column average (88.33)." Compare this with Example 3, where dropping the row changed the average to 88.5 — same data, two honest choices, two slightly different answers.

Example 6 — Real-world case: a bank's loan application forms

A bank receives thousands of loan forms. Some applicants leave the income box empty. Some apply twice, once online and once on paper. Some write their salary as "50,000" and others as "50000". Before anyone decides who gets a loan, a cleaning team fixes all of this: duplicates merged, income turned into real numbers, missing incomes flagged for follow-up. If cleaning is skipped, the bank might approve the same person twice, or reject someone whose salary was stored as text and read as zero. Cleaning here is not office tidiness — it is the step that keeps real money decisions fair.

(Roman Urdu: Bank ke form ki safai seedha paisay ke faislon par asar karti hai — duplicate ya ghalat type ka matlab ghalat faisla ho sakta hai.)

Quick recap

- Four common types: numbers, text, dates, true/false.
- Four common problems: missing values, duplicates, wrong formats, wrong types.
- Clean in order: types, duplicates, missing values, spellings and formats.
- Always say what you did about missing values — dropped, or filled, and how.
- Check the cleaned table by eye before you analyse it.

Common mistakes

1. **Averaging a column that still has text in it** — the maths fails or silently skips rows. Fix: check the type of each column first, and convert text numbers to real numbers before any calculation.

2. **Deleting missing values without saying so** — your reader thinks the average covers everyone, when it covers only some. Fix: write one line: "2 rows dropped for missing marks; average is over the other 28 students."
3. **Filling a missing mark with zero** — zero means "scored nothing", which is very different from "mark not recorded". Fix: leave it missing, or fill it with the column average and say so — never silently with zero.
4. **Sorting a column that is still text** — "100" sorts before "85" as text, so rankings and top-10 lists come out wrong with no error message. Fix: convert to numbers first, then sort.
5. **Fixing a spelling in one place but not everywhere** — "Maths" and "maths" count as two different subjects, and "Lahore" and "lahore" as two different cities. Fix: make one rule (for example, capitalise the first letter of every value) and apply it to the whole column, as Example 1 of lesson DS.7 does.

Practice

1. Copy the messy table from Example 1 onto paper. Circle every problem you can find — there are at least four.
2. In plain Python, write a list of 5 dictionaries with one missing value and one duplicate, then remove the duplicate and count the missing values.
3. Take any small table you own — a class list, a shop list — and rewrite every date in it in one single format.

1. In pandas, build a Series of ["70", "85", "absent", "90"], convert it with `pd.to_numeric(errors="coerce")`, and print the result. Which value became missing, and why is that better than a crash?
2. Take the table from Example 3 and fill Bilal's missing mark with the column average instead of dropping his row. What is the new average, and how does it compare with 88.5?
3. Find one column in any table you own where the same fact is written two ways (dates, names, spellings). Write the single rule you would apply to make the whole column consistent.

Practice quiz

1. What is a missing value in a dataset?

- A wrong number
- An empty gap where a value should be
- A duplicate row
- A date written wrong

Answer: An empty gap where a value should be

2. Why must data be cleaned before analysis?

- Cleaning makes files smaller
- Dirty data gives wrong answers, no matter how good the analysis is
- Clean data looks nicer
- Tools refuse dirty data

Answer: Dirty data gives wrong answers, no matter how good the analysis is

3. What should you do with exact duplicate rows?

- Keep them, they add weight
- Remove them, or they count the same thing twice
- Rename them
- Move them to the end

Answer: Remove them, or they count the same thing twice

Watch the video lesson: https://youtu.be/0iBWstWxrqA

Read online: https://scopetreasure.com/notes/ds-data-types-cleaning · More free tools and notes at https://scopetreasure.com