One table, assembled on purpose

Start from a spine of the units you must report on, attach one measure at a time, assert at every seam, and carry provenance columns so a reviewer can trace any figure back to the file it came from.

Pierre Robentz Cassion

Lesson 8 of 8

Joining and Reshaping Programme Data — The table you analyse

What this lesson covers

- The table everything else is computed from
- Decide the unit of analysis before you write a line
- Build the spine first
- Attach one measure at a time, and assert at every seam
- Carry provenance
- Assert what must be true of the finished table
- What does not belong in the table
- Save it with its grain in the name
- Where this course leaves you

The table everything else is computed from

- By the end of an assembly you want exactly one table: one row per unit of analysis, one column per measure, and nothing else.

Decide the unit of analysis before you write a line — In Python

```
# unit of analysis: one school x month
# rows: 24 schools x 3 months = 72
```

Decide the unit of analysis before you write a line — In R

```
# unit of analysis: one school x month
# rows: 24 schools x 3 months = 72
```

Build the spine first — In Python

```
schools = roster["school_id"].drop_duplicates()
months = pd.Series(["2024-02", "2024-03", "2024-04"], name="month")

analysis = (
    pd.MultiIndex.from_product([schools, months], names=["school_id", "month"])
    .to_frame(index=False)
)
print(len(analysis), "rows in the spine")
```

Build the spine first — In R

```r
analysis <- tidyr::expand_grid(
  school_id = unique(roster$school_id),
  month = c("2024-02", "2024-03", "2024-04")
)
```

Build the spine first

- Spine first is the single most useful habit in this lesson — A school-month with no data now exists as a row with...

Attach one measure at a time, and assert at every seam — In Python

```
def attach(base, addition, on, name):
    before = len(base)
    out = base.merge(addition, on=on, how="left", validate="one_to_one")
    if len(out) != before:
        raise ValueError(f"{name}: row count changed {before} -> {len(out)}")
    filled = out[addition.columns.difference(on)].notna().any(axis=1).mean()
    print(f"{name}: attached, {filled:.1%} of spine rows matched")
    return out


analysis = attach(analysis, attendance_by_school_month, ["school_id", "month"], "attendance")
analysis = attach(analysis, enrolment_by_school, ["school_id"], "enrolment")
analysis = attach(analysis, feeding_by_school, ["school_id"], "feeding programme")
```

Attach one measure at a time, and assert at every seam — In R

```r
attach_measure <- function(base, addition, by, name) {
  before <- nrow(base)
  out <- dplyr::left_join(base, addition, by = by, relationship = "one-to-one")
  if (nrow(out) != before) {
    stop(sprintf("%s: row count changed %d -> %d", name, before, nrow(out)))
  }
  message(sprintf("%s: attached", name))
  out
}

analysis <- analysis |>
  attach_measure(attendance_by_school_month, c("school_id", "month"), "attendance") |>
  attach_measure(enrolment_by_school, "school_id", "enrolment")
```

Carry provenance — In Python

```
analysis["source_register"] = "school-attendance-2024.v1.csv"
analysis["denominator_source"] = "roster 2024, enrolled students"
analysis["marks_definition"] = "true/false only; Y/N excluded"
analysis["extracted_on"] = "2026-07-27"
```

Carry provenance — In R

```r
analysis <- analysis |>
  mutate(
    source_register    = "school-attendance-2024.v1.csv",
    denominator_source = "roster 2024, enrolled students",
    marks_definition   = "true/false only; Y/N excluded",
    extracted_on       = "2026-07-27"
  )
```

Assert what must be true of the finished table — In Python

```
assert analysis.duplicated(subset=["school_id", "month"]).sum() == 0
assert analysis["attendance_rate"].between(0, 1).all()
assert (analysis["days_present"] <= analysis["days_marked"]).all()
assert len(analysis) == 24 * 3

print(f"analysis table: {len(analysis)} rows, "
      f"{analysis['attendance_rate'].isna().sum()} without an attendance rate")
```

Assert what must be true of the finished table — In R

```r
stopifnot(
  !any(duplicated(analysis[c("school_id", "month")])),
  all(dplyr::between(analysis$attendance_rate, 0, 1), na.rm = TRUE),
  all(analysis$days_present <= analysis$days_marked, na.rm = TRUE),
  nrow(analysis) == 24 * 3
)
```

What does not belong in the table

- **Intermediate columns.** `present_x`, `_merge`, `key_clean` and the six columns you needed for one calculation. Drop...
- **Rows below your reporting threshold.** If you would suppress a cell of four cases in a published table, do not carry...
- **Anything person-level.** An analysis table at school-month grain has no business carrying a student identifier, and...

Save it with its grain in the name — In Python

```
analysis.to_csv("outputs/tables/attendance_by_school_month.csv", index=False)
```

Save it with its grain in the name — In R

```
readr::write_csv(analysis, here::here("outputs", "tables", "attendance_by_school_month.csv"))
```

Save it with its grain in the name — Example

```
scripts/
  01_read_and_validate.R      contract and validation   (cleaning course)
  02_clean.R                  rules and cleaning log    (cleaning course)
  03_assemble.R               spine, joins, assertions  (this course)
  04_report.R                 tables and figures
```

Where this course leaves you

- You can take several files that describe the same programme and produce one table you would defend column by column — with the joins proved, the grain stated, the denominator sourced, the calendar complete, and the difference from what was reported upward written down rather than argued about.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/joining-and-reshaping/joining-08-the-analysis-table