CASSION · COURSE

Data Quality Assessment

Run a routine data quality assessment the way a donor auditor will — verification factors, consistency over time, and a report that names the fix as well as the fault.

Instructor	Pierre Robentz Cassion
Level	Intermediate
Learner hours	12 h
Taught in	Python · R
Updated	July 28, 2026
Sectors	Public Health · Nutrition
Standards and methodologies	UNICEF indicator definitions · Core Humanitarian Standard (CHS) · Results-Based Management (RBM)

Read and run the course online at data-analysis.cassion.dev/courses/data-quality-assessment

What you will be able to do

- Score a dataset against the five quality dimensions with a measure for each, rather than calling it "good" or "poor"
- Compute reporting completeness and timeliness as indicators in their own right, with their own denominators
- Recount a source register against what was reported upward and turn the difference into a verification factor with a tolerance band
- Choose which facilities to verify when you cannot visit them all, and state what your sample can and cannot conclude
- Detect digit preference, heaping and implausibly stable series, and calibrate the check against what chance produces
- Write a DQA report whose every finding carries a root cause, a corrective action, an owner and a deadline

Prerequisites

None. This course assumes no prior programming experience.

Contents

About this course

Everyone in this sector has been in the meeting where somebody says the data quality is poor. It is never a useful sentence. Poor how — incomplete, late, inconsistent with the register, or internally contradictory? Poor everywhere, or in four facilities out of thirty-eight? Poor enough to change the decision, or poor in a way that moves the number by half a point?

A data quality assessment is the discipline that replaces that sentence with numbers. It has a shape the sector has agreed on: **five dimensions, each with a measure; a recount against the source; a sample you can defend; and a report where every finding has an owner and a date.** Donors run it on you, and running it on yourself first is the difference between a finding and a surprise.

This course completes the module. *Data Cleaning and Validation* taught you to find and fix defects in a file. *Joining and Reshaping* taught you to put files together and reconcile them. This one turns both into a routine somebody else can run on a schedule — and into a document that changes what happens next quarter, which is the only reason to do any of it.

Every technique is shown in Python and in R. The worked examples run against the DHIS2-shaped vaccination extract, where 642 of 2,736 facility-months carry no report at all, and against the SMART survey, where one measurement team's heights end in .0 or .5 sixty-nine percent of the time.

One warning about tone, because it decides whether this work is useful. A DQA that reads as an accusation produces defensive reporting, and defensive reporting is worse data. **The finding is about a system, not about a person**, and the course keeps returning to how that is written.

Part I

What quality means here

CHAPTER 1

Five dimensions, five measures

Lesson 1 of 8 · 75 min

Calling the data poor is not a finding. Accuracy, completeness, timeliness, consistency and integrity — each with a number attached, computed on the same extract, and a scorecard that survives being disagreed with.

1.1 The sentence to stop saying

"The data quality is poor." Everyone nods, nothing happens, and the same sentence is said next quarter.

It fails because it is unactionable in three separate ways. It does not say **which** quality — a late report and a fabricated report are different problems with different fixes. It does not say **how much** — poor everywhere or poor in four facilities. And it does not say **so what** — whether the defect is large enough to change the number anyone is going to act on.

The five dimensions exist to fix all three at once. They are not a framework to recite; they are five questions, each of which has a number as its answer.

1.2 The five, and the measure for each

Dimension	The question	The measure
Completeness	Did everyone report, on everything?	Reporting rate; missing-value ra
Timeliness	Did the report arrive in time to be used?	Share submitted by the deadline
Accuracy	Does the reported figure match the source?	Verification factor: recount over
Consistency	Do the numbers agree with each other and with last month?	Rate of failed internal rules; outl
Integrity	Was the value produced by measurement or by something else?	Digit preference; heaping; implau

Two things about that table are worth arguing over, because both come up.

Accuracy is the only one that needs the source document. The other four are computable from the extract you already have, at your desk, this afternoon. That asymmetry drives the whole design of a DQA: you use the four desk dimensions to decide where to spend the expensive fifth.

Integrity is not an accusation. It measures whether numbers look like measurements. A heaped age distribution usually means nobody asked for a birth certificate, not that anybody invented anything, and the lesson on it spends most of its time on that distinction.

1.3 Compute all five on one extract

Take the routine vaccination extract — 38 facilities, twelve months, six antigens.

```
import pandas as pd

vax = pd.read_csv("vaccination-coverage-2024.v1.csv", parse_dates=["period"])
vax["reported"] = vax["report_submitted"] == True

scorecard = {}
```

```python
# Completeness
scorecard["reporting_rate"] = vax["reported"].mean()
scorecard["field_completeness"] = 1 - vax.isna().mean().mean()

# Consistency: doses cannot exceed the target population
rule_fail = (vax["doses_administered"] > vax["target_population"]).mean()
scorecard["rule_failure_rate"] = rule_fail

# Integrity: share of reported values ending in zero
reported = vax[vax["reported"]]
scorecard["round_number_share"] = (reported["doses_administered"] % 10 == 0).mean()

print(pd.Series(scorecard).round(3))
```

```r
library(dplyr)
library(readr)

vax <- read_csv("vaccination-coverage-2024.v1.csv")

scorecard <- tibble::tibble(
  reporting_rate     = mean(vax$report_submitted),
  field_completeness = 1 - mean(is.na(as.matrix(vax))),
  rule_failure_rate  = mean(vax$doses_administered > vax$target_population),
  round_number_share = mean(
    vax$doses_administered[vax$report_submitted] %% 10 == 0
  )
)

round(scorecard, 3)
```

Measure	Value	Read as
Reporting rate	76.5%	642 of 2,736 facility-months carry no report
Field completeness	100%	no blank cells — which is not the same as no missing data
Rule failure rate	0%	no facility reports more doses than its target population
Round-number share	10.4%	what chance produces; nothing to see

Look at rows two and one together. **Field completeness is 100% and reporting completeness is 76.5%**, and only the second one matters. Every non-reporting facility-month is present in the file as a row of zeroes with a flag, so a blank-cell count says the data is perfect. This is the defect the whole module keeps circling: a missing report that arrives as a zero.

1.4 Weight the dimensions by what they cost

A scorecard with five equal numbers implies the five matter equally. They do not, and which matters most depends entirely on what the data is for.

- For a **coverage figure**, completeness dominates. A 76.5% reporting rate makes any district total a lower bound, and no amount of accuracy in the reports you did receive fixes that.
- For a **caseload figure used for procurement**, accuracy dominates. Ordering therapeutic food against an over-reported caseload wastes money; against an under-reported one, children go without.

- For an **early warning indicator**, timeliness dominates. A perfectly accurate report arriving six weeks after the outbreak has zero value.

Say which dimension you weighted and why, in the report. A composite quality score that hides the weighting is the same defect as an indicator that hides its denominator.

1.5 Do not collapse it to one number

Somebody will ask for a single score out of 100. Resist, and offer the scorecard instead, for a specific reason: **the five dimensions have different fixes, and the single number tells you nothing about which one to apply.**

A district at 82% because it is late is fixed by moving a deadline. A district at 82% because a third of its facilities never report is fixed by finding out why they stopped. Both score 82. Only one of them is fixed by a training workshop, and neither is fixed by the workshop somebody will propose.

If a composite is genuinely required — some donor templates demand it — publish it **beside** the components, never instead of them.

```python
scorecard_table = pd.DataFrame({
    "dimension": ["Completeness", "Timeliness", "Accuracy", "Consistency", "Integrity"],
    "measure": ["Reporting rate", "Submitted by deadline", "Verification factor",
                "Rule failure rate", "Round-number share"],
    "value": [0.765, None, None, 0.0, 0.104],
    "source": ["extract", "submission log", "facility visit", "extract", "extract"],
})
print(scorecard_table)
```

```r
scorecard_table <- tibble::tribble(
  ~dimension,     ~measure,                ~value, ~source,
  "Completeness", "Reporting rate",         0.765, "extract",
  "Timeliness",   "Submitted by deadline",     NA, "submission log",
  "Accuracy",     "Verification factor",       NA, "facility visit",
  "Consistency",  "Rule failure rate",       0.000, "extract",
  "Integrity",    "Round-number share",      0.104, "extract"
)
```

The `source` column is the useful one. It says immediately which numbers you can produce today and which need someone to travel, and that is the shape of every DQA plan.

1.6 What comes next

Two of the five dimensions are almost never reported at all, and both are computable from data you already hold. The next lesson computes them properly — reporting completeness and timeliness, each with its own denominator, and the trap of dividing by the facilities that reported.

CHAPTER 2

The two indicators nobody computes

Lesson 2 of 8 · 75 min

Reporting completeness and timeliness, each with its own denominator. Eleven of thirty-eight facilities reported in August, and every coverage figure for that month is a statement about those eleven.

2.1 Two indicators, and neither is in the report

Open almost any routine health report from this sector and you will find coverage, caseload, cure rates and stock-outs. You will very rarely find the two numbers that say how much of the district those figures describe.

They are cheap. Both come out of the extract you already have — one of them does — and both change how every other number in the report should be read.

2.2 Reporting completeness

The reporting rate is the share of expected reports that were actually submitted. The word doing the work is **expected**.

```python
import pandas as pd

vax = pd.read_csv("vaccination-coverage-2024.v1.csv", parse_dates=["period"])
vax["reported"] = vax["report_submitted"] == True

print(f"overall reporting rate: {vax['reported'].mean():.1%}")

by_month = vax.groupby(vax["period"].dt.strftime("%Y-%m"))["reported"].mean()
print((by_month * 100).round(0))
```

```r
library(dplyr)
library(readr)

vax <- read_csv("vaccination-coverage-2024.v1.csv")

mean(vax$report_submitted)

vax |>
  mutate(month = format(period, "%Y-%m")) |>
  summarise(reporting_rate = mean(report_submitted), .by = month) |>
  arrange(month)
```

Month	Reporting rate
January	92%
February	82%
March	74%
April–July	82–87%
August	**29%**
September	**45%**
October	95%
November–December	76–87%

August and September are not a wobble. Eleven of thirty-eight facilities reported in August and seventeen in September. **Any August coverage figure is a statement about eleven facilities**, and publishing it beside July's without saying so compares two different districts.

Note also that the collapse and the recovery are district-wide and simultaneous. That is the signature of a system event — a supply of paper forms, a strike, a server, a supervisor who left — and not of facilities individually failing. A DQA that raises 27 separate facility findings for August has misread the pattern.

2.3 Break it down three ways, always

One number hides everything worth knowing. Break the reporting rate by time, by reporting unit, and by unit type.

```python
by_facility = vax.groupby("facility_id")["reported"].mean().sort_values()
print(by_facility.head(6))
print(f"{(by_facility == 1).sum()} facilities reported every month")

by_type = vax.groupby("facility_type")["reported"].mean()
print((by_type * 100).round(1))
```

```r
vax |> summarise(rate = mean(report_submitted), .by = facility_id) |> arrange(rate) |>
    head(6)

vax |> summarise(rate = mean(report_submitted), .by = facility_type)
```

Facility type	Reporting rate
Health centre	82.8%
District hospital	75.0%
Health post	71.7%

The worst six facilities all sit at 58% — seven months of twelve — and only two of thirty-eight reported every month. The gradient by type is the actionable finding: **health posts report worst**, which points at supervision distance, staffing or transport rather than at any individual facility.

2.4 The denominator trap

The one mistake that makes a reporting rate useless:

```python
# WRONG: the share of reports that were submitted, among reports that were submitted
wrong = vax[vax["reported"]]["reported"].mean()          # always 1.0
```

```
# WRONG
vax |> filter(report_submitted) |> summarise(rate = mean(report_submitted))
```

Obvious written out. It is not obvious when the filter is six lines earlier in a pipeline, and it is exactly what happens when someone reads a DHIS2 export that only contains submitted reports. **If the extract has no rows for non-reporting units, the reporting rate cannot be computed from it at all** — you need the list of expected units from somewhere else.

```
EXPECTED_FACILITIES = pd.read_csv("facility-master-list.csv")["facility_id"]

expected = len(EXPECTED_FACILITIES) * vax["period"].nunique()
submitted = vax.loc[vax["reported"], ["facility_id", "period"]].drop_duplicates().shape[0]
print(f"{submitted} of {expected} expected facility-months")
```

```
expected  <- nrow(facility_master) * n_distinct(vax$period)
submitted <- vax |> filter(report_submitted) |> distinct(facility_id, period) |> nrow()
```

The master list is the authority, exactly as the administrative population frame was in the previous course. A facility that closed and one that stopped reporting look identical in the extract and are completely different findings, and only the master list tells them apart.

2.5 Timeliness needs a field you may not have

Timeliness is the share of reports submitted by the deadline, and the median lateness of the rest.

```
submissions["days_late"] = (
    submissions["submitted_on"] - submissions["deadline"]
).dt.days

print(f"on time: {(submissions['days_late'] <= 0).mean():.1%}")
print(f"median days late, of those late: "
      f"{submissions.loc[submissions['days_late'] > 0, 'days_late'].median():.0f}")
```

```
submissions <- submissions |>
  mutate(days_late = as.integer(submitted_on - deadline))

c(on_time = mean(submissions$days_late <= 0),
  median_late = median(submissions$days_late[submissions$days_late > 0]))
```

This extract cannot support that calculation, and saying so is itself a DQA finding. There is no submission timestamp in the file — only `report_submitted`, a flag with no date. DHIS2 records a `lastUpdated` per data value set and a completeness date per registration, so the field exists in the system and was not carried into the export.

Write that up as it stands:

> **Finding.** Timeliness cannot be assessed. The monthly extract carries a submission flag but no submission date, although DHIS2 records one. **Corrective action.** Add the completeness date to the standard extract. **Owner.** HMIS focal point. **By.** Next quarterly extract.

A dimension you cannot measure is a finding about the reporting system, not a gap in your assessment. Report it in the same table as the rest.

2.6 Read every other figure through these two

The point of computing these first is what they license you to say afterwards.

- **Coverage, at 76.5% reporting.** A district total is a lower bound, not an estimate. Say "among reporting facilities" in the label, every time.
- **A month-on-month trend.** August's fall is a reporting fall until proven otherwise. Plot the reporting rate on the same chart as the indicator, and the question answers itself.
- **A facility ranking.** Facilities reporting seven months of twelve cannot be ranked against facilities reporting twelve. Either restrict the ranking to a common set of months or rank on a per-month average and show the count.

```python
monthly = (
    vax[vax["reported"]]
    .groupby(vax["period"].dt.strftime("%Y-%m"))
    .apply(lambda g: g["doses_administered"].sum() / g["target_population"].sum())
    .rename("coverage")
    .to_frame()
)
monthly["reporting_rate"] = by_month
print(monthly.round(3))
```

```r
vax |>
  mutate(month = format(period, "%Y-%m")) |>
  summarise(
    coverage = sum(doses_administered[report_submitted]) /
               sum(target_population[report_submitted]),
    reporting_rate = mean(report_submitted),
    .by = month
  )
```

> Two columns, always adjacent. A coverage figure without its reporting rate is an unlabelled number, and the label is the part that stops it being misread.

2.7 What comes next

Completeness and timeliness are about whether a report arrived. The next lesson is about whether the report was right — the recount against the source register, the verification factor it produces, and the tolerance band that turns it into a decision.

Part II

Checking against the source

CHAPTER 3

The recount, and the number it produces

Lesson 3 of 8 · 75 min

A verification factor is the recount over the reported figure. The district sits at 1.012, which nobody would question, and one school in twenty-four sits at 1.42.

3.1 Accuracy is the dimension that needs a visit

The other four dimensions come out of the extract. This one does not. Accuracy means "does the reported figure match the source", and the source is a paper register in a cupboard, a tally sheet, or a tablet that has not synced since March.

The method is the oldest one in this course and it has not been improved on: **go and count it again.** Then divide.

3.2 The verification factor

```
verification factor = recount / reported
```

What you counted from the source, over what was reported upward. A VF of 1.00 means the report matched the register. It is a ratio, not a percentage of error, and getting that the right way round matters when you write it down.

Both directions are findings, and they are not the same finding.

- **VF above 1** — the register holds more than was reported. Under-reporting. Usually a transcription or aggregation loss: a column not added up, a page missed, an electronic extract that dropped values it did not recognise. The programme did more work than it got credit for.
- **VF below 1** — the report claims more than the register supports. Over-reporting. Sometimes double-counting or a definition mismatch; occasionally something worse. This is the direction that gets a grant suspended, so it is the direction to be most careful about asserting.

3.3 Compute it on a register you have

Take the school attendance register. The electronic extract carries `true` and `false` marks; one school also recorded some days as `Y` and `N`, which the extract's boolean cast did not recognise. A verification visit reading the paper register would count those days as present. The extract did not.

That gives a real recount and a real reported figure, on the same data.

```
import pandas as pd

attendance = pd.read_csv("school-attendance-2024.v1.csv")
roster = pd.read_csv("school-roster-2024.v1.csv")

# The two students on the roster twice are ambiguous; exclude them from the
```

```python
# verification and say so, rather than assigning them arbitrarily.
duplicated = roster["student_id"].duplicated(keep=False)
clean_roster = roster[~duplicated]

marks = attendance.merge(
    clean_roster[["student_id", "school_id"]], on="student_id",
    how="inner", validate="many_to_one",
)

vf = (
    marks.assign(
        reported=marks["present"] == "true",
        recount=marks["present"].isin(["true", "Y"]),
    )
    .groupby("school_id")[["recount", "reported"]]
    .sum()
)
vf["verification_factor"] = vf["recount"] / vf["reported"]
print(vf.sort_values("verification_factor", ascending=False).head())
```

```r
library(dplyr)

clean_roster <- roster |>
  group_by(student_id) |> filter(n() == 1) |> ungroup()

vf <- attendance |>
  inner_join(select(clean_roster, student_id, school_id), by = "student_id",
             relationship = "many-to-one") |>
  summarise(
    recount  = sum(present %in% c("true", "Y")),
    reported = sum(present == "true"),
    .by = school_id
  ) |>
  mutate(verification_factor = recount / reported) |>
  arrange(desc(verification_factor))
```

School	Recount	Reported	VF
SCH09	2,456	1,734	**1.416**
SCH01	2,945	2,945	1.000
SCH02	2,569	2,569	1.000
… 22 more			1.000
District	**61,764**	**61,042**	**1.012**

3.4 The district figure is the trap

Read the last row first, because it is the row that appears in most reports. **The district verification factor is 1.012** — one and a bit percent — and no auditor in the world would raise a finding on that.

Now read the first row. One school in twenty-four under-reported its attendance by 42%, and it is invisible in the district total because the other twenty-three are exact.

This is the same argument the previous course made about averaging verification factors, and it is worth making twice because it is the most common way a DQA produces a clean bill of health for a district with a real problem in it. **Report the distribution and the exceptions. The mean is the least informative statistic available.**

```python
print(vf["verification_factor"].describe())
outliers = vf[(vf["verification_factor"] - 1).abs() > 0.05]
print(f"{len(outliers)} of {len(vf)} schools outside the tolerance band")
```

```r
summary(vf$verification_factor)
vf |> filter(abs(verification_factor - 1) > 0.05)
```

3.5 The tolerance band

A DQA needs a threshold decided before you look, or every finding becomes a negotiation.

The convention most donor frameworks use is **0.95 to 1.05** — within 5% is acceptable, outside it is a finding. It is a convention, not a law, and two adjustments are legitimate:

- **Widen it for small numbers.** A facility reporting 12 cases has a VF of 1.08 if one case was missed. On small counts, use an absolute difference as well as a ratio: "outside 5% *and* more than 5 units".
- **Narrow it for money.** Where the figure drives a payment — results-based financing, a per-beneficiary reimbursement — 5% is a lot of money and the band is usually tighter, set by the contract rather than by you.

```python
TOLERANCE = 0.05
MIN_ABSOLUTE = 5

vf["difference"] = vf["recount"] - vf["reported"]
vf["finding"] = (
    ((vf["verification_factor"] - 1).abs() > TOLERANCE)
    & (vf["difference"].abs() > MIN_ABSOLUTE)
)
print(vf[vf["finding"]])
```

```r
vf <- vf |>
  mutate(
    difference = recount - reported,
    finding = abs(verification_factor - 1) > 0.05 & abs(difference) > 5
  )
```

Write the band into the protocol before the visit. A threshold chosen after seeing the numbers is not a threshold, and everyone in the room will know it.

3.6 The recount is a protocol, not an afternoon

The arithmetic is trivial. Everything that makes the number trustworthy happens before it.

- **Fix the period exactly.** "March" is not a period. "1 to 31 March, by date of service, not date of entry" is.
- **Fix the definition exactly.** Count what the indicator's numerator says, not what the register's column heading says. Most VFs far from 1 are definition mismatches, and they are not errors by anyone.

- **Recount independently.** The person who compiled the report should not do the recount. Not because of dishonesty — because they will reproduce the same interpretation, and the interpretation is what you are testing.
- **Record what you counted from.** Register, tally sheet, individual cards. Two sources in one facility often disagree, and which one you counted is part of the finding.

3.7 What a VF cannot tell you

Three limits worth stating in the report, because someone will over-read the number otherwise.

It cannot tell you the register is right. A VF of 1.00 means the report matches the register. If the register itself was filled in at the end of the week from memory, both are wrong together and the VF is silent.

It does not generalise from the facilities you visited unless you sampled properly. Which is the next lesson.

It is not an accusation. A VF of 1.42 in one school, with twenty-three others at exactly 1.00, is the shape of a systems defect — one school's data entry convention, unrecognised by the extract — not of one head teacher inflating numbers. Writing it up as the latter guarantees the next round is worse.

3.8 What comes next

You cannot visit thirty-eight facilities. The next lesson is how to choose the ones you do visit, how many is enough, and what a sample of six licenses you to say about the other thirty-two.

CHAPTER 4

Choosing which six facilities to visit

Lesson 4 of 8 · 75 min

Three sampling strategies that answer three different questions, how many is enough, and the sentence you are allowed to write about the thirty-two facilities you did not visit.

4.1 You have a week and thirty-eight facilities

A verification visit costs a vehicle, a day, two people and fuel. The budget covers six. Everything in this lesson follows from that arithmetic, and the first thing to be clear about is that **which six you choose determines what your finding is allowed to mean.**

Three strategies, three questions, and mixing them up is the commonest DQA methods failure.

Strategy	Answers	Cannot answer
Random	"What is the district's data quality?"	"Where are the problems?"
Risk-based	"Are the facilities we suspect actually wrong?"	"What is the district's data quality?"
Census of the largest	"Is the district total right?"	Anything about small facilities

4.2 Random: the only one that generalises

If you want a statement about the district, the sample has to be drawn without reference to what you expect to find.

```python
import pandas as pd

facilities = vax[["facility_id", "facility_type"]].drop_duplicates()

sample = facilities.sample(n=6, random_state=20260728)
print(sample)
```

```r
set.seed(20260728)

sample <- facilities |> dplyr::slice_sample(n = 6)
```

Seed it and record the seed. A DQA whose sample cannot be reproduced invites the question of whether the facilities were chosen after somebody knew what was in them, and the seed answers it in one line.

Stratify where a subgroup matters. Health posts report worst, so a simple random sample of six could easily contain none of them:

```python
sample = (
    facilities.groupby("facility_type", group_keys=False)
    .apply(lambda g: g.sample(n=min(2, len(g)), random_state=20260728))
)
```

```
sample <- facilities |>
  group_by(facility_type) |>
  slice_sample(n = 2) |>
  ungroup()
```

Stratifying by type gives you a statement about each type as well as about the district, and it costs nothing.

4.3 Risk-based: where the problems are

If the purpose is to fix things rather than to describe them, choose the facilities the desk analysis already flagged. The next two lessons are entirely about producing that ranking.

```
risk = (
    vax.groupby("facility_id")
    .agg(reporting_rate=("reported", "mean"))
    .assign(round_share=lambda d: d.index.map(round_number_share))
)
risk["score"] = (1 - risk["reporting_rate"]) + risk["round_share"]
print(risk.sort_values("score", ascending=False).head(6))
```

```
risk <- vax |>
  summarise(reporting_rate = mean(report_submitted), .by = facility_id) |>
  left_join(round_shares, by = "facility_id") |>
  mutate(score = (1 - reporting_rate) + round_share) |>
  arrange(desc(score))
```

This finds more problems per vehicle-day than random sampling, which is why most real DQAs use it. It also **cannot be generalised**, and the report must say so:

> Six facilities were selected on the basis of desk-review flags. Findings describe those facilities and are not representative of the district.

Every DQA report that omits that sentence and then quotes a district-level verification factor is making a claim its design does not support.

4.4 Census of the largest

A third option, and often the most useful for a total. Order facilities by volume, and visit enough of them to cover most of the numerator.

```
volume = (
    vax[vax["reported"]]
    .groupby("facility_id")["doses_administered"].sum()
    .sort_values(ascending=False)
)
share = volume.cumsum() / volume.sum()
print(share.head(10).round(3))
```

```
volume <- vax |>
  filter(report_submitted) |>
  summarise(doses = sum(doses_administered), .by = facility_id) |>
  arrange(desc(doses)) |>
  mutate(cumulative = cumsum(doses) / sum(doses))
```

If eight facilities account for half the doses, verifying those eight bounds the error on the district total regardless of what the other thirty do. **This is the right design when the question is "can I trust the total I am reporting to the donor"** — and the wrong one when the question is about service quality in remote posts, because it systematically ignores them.

4.5 How many is enough

There is no single answer, but there is a way to think about it that survives being challenged.

For a **yes/no question** — is this facility's reporting acceptable — you are estimating a proportion, and precision improves with the square root of the sample. Six facilities out of thirty-eight gives a confidence interval so wide that almost no result is distinguishable from any other. Be honest about that rather than reporting "33% of facilities had findings" from a sample of six.

For **detecting whether a problem exists at all**, small samples are far more capable. If 30% of facilities have a defect, the chance that six randomly chosen facilities contain none of them is about 12% — so a clean sample of six is reasonable evidence that a *widespread* problem is absent, and no evidence at all about a rare one.

```
p = 0.30
n = 6
print(f"chance of finding none: {(1 - p) ** n:.1%}")
```

```
(1 - 0.30)^6
```

That single line is worth putting in the report. It converts "we visited six and found nothing" into a statement with a number attached.

4.6 LQAS: designed for exactly this

Lot Quality Assurance Sampling is the formalisation of the paragraph above, and this sector already uses it for coverage surveys. The idea is to stop trying to *estimate* and start trying to *decide*: pick a threshold, a sample size and a decision rule, and accept or reject.

> Visit 12 facilities. If 3 or more have a verification factor outside 0.95–1.05, the district's reporting is classified as needing corrective action.

The virtue is that the sample can be small because you are answering a smaller question. The cost is that you get a verdict rather than a figure, and someone will ask for the figure anyway. Say up front which you designed for.

4.7 Combine, and label

Real DQAs usually do all three, and that is fine as long as the report separates them.

```
plan = pd.DataFrame({
    "facility_id": ["FAC012", "FAC033", "FAC004", "FAC019", "FAC007", "FAC021"],
    "reason": ["random", "random", "risk: reporting 58%", "risk: round numbers",
               "volume: top 5", "volume: top 5"],
})
```

```
plan <- tibble::tribble(
  ~facility_id, ~reason,
  "FAC012", "random",
  "FAC033", "random",
  "FAC004", "risk: reporting 58%",
  "FAC019", "risk: round numbers",
  "FAC007", "volume: top 5",
  "FAC021", "volume: top 5"
)
```

The `reason` column is the methods section. With it, a reader knows which findings generalise and which do not. Without it, the report has one sample and three incompatible claims, and a reviewer who notices will discount all of them.

4.8 What comes next

Risk-based sampling needs a risk ranking, and so far you have only reporting rates. The next two lessons build the rest of it from the extract you already hold — trends that move impossibly, values that repeat, and digits that betray a number nobody measured.

Part III

Finding the sites worth visiting

CHAPTER 5

Comparing a facility to its own past

Lesson 5 of 8 · 75 min

Ratio to the rolling median, the small-count problem that makes the biggest outliers meaningless, and the series so stable it is worth a second look.

5.1 Consistency is a comparison, and you get to choose what against

The consistency dimension asks whether the numbers hang together. In practice that means comparing each reported value against something, and there are two candidates.

Against its peers. Facility A reported 40 doses; the district median is 25. Tempting and usually wrong — facilities differ enormously in catchment size, and a large facility will be flagged every month for being large.

Against its own past. Facility A reported 40 doses this month, against its own median of 25 for the year. That is a comparison worth making, because a facility is a reasonable control for itself.

Use the second. Every check in this lesson is a facility against its own history.

5.2 Ratio to the median

The simplest useful statistic, and it survives the fact that these series are short.

```python
import pandas as pd

reported = vax[vax["reported"]].copy()

reported["own_median"] = reported.groupby(["facility_id", "antigen"])[
    "doses_administered"
].transform("median")
reported["ratio"] = reported["doses_administered"] / reported["own_median"]

print(reported.nlargest(5, "ratio")[
    ["facility_id", "antigen", "period", "doses_administered", "own_median", "ratio"]
])
```

```r
library(dplyr)

reported <- vax |>
  filter(report_submitted) |>
  mutate(own_median = median(doses_administered),
         ratio = doses_administered / own_median,
         .by = c(facility_id, antigen))

reported |> slice_max(ratio, n = 5)
```

Median, not mean. The mean is dragged by the very outlier you are looking for, so a value can inflate its own comparison baseline and hide. This is the same reason SMART plausibility checks use robust statistics.

Where a series is long enough, prefer a *rolling* median so a genuine seasonal level shift does not flag every month after it:

```
reported = reported.sort_values(["facility_id", "antigen", "period"])
reported["rolling_median"] = (
    reported.groupby(["facility_id", "antigen"])["doses_administered"]
    .transform(lambda s: s.rolling(6, min_periods=3, center=True).median())
)
```

```
reported <- reported |>
  arrange(facility_id, antigen, period) |>
  mutate(rolling_median = zoo::rollapply(doses_administered, 6, median,
                                         partial = TRUE, align = "center"),
         .by = c(facility_id, antigen))
```

5.3 The small-count problem

Run the ratio check on this extract and the top of the list is this:

Facility	Antigen	Month	Value	Own median	Ratio
FAC013	mcv2	December	5	3	1.67
FAC037	opv3	September	13	8	1.62
FAC030	mcv2	October	8	5	1.60

The largest outlier in the entire district is **a facility that gave five second doses of measles vaccine instead of three.** Two doses. Nobody is going to investigate that, and if your DQA report leads with it, nobody will read the rest.

This is the defining failure of ratio-based flagging, and it is guaranteed rather than unlucky: a ratio has a small denominator at the bottom of the range, so the most extreme ratios always come from the smallest counts. The check is doing exactly what it was asked to do and the question was wrong.

5.4 Put an absolute floor beside the ratio

```
RATIO_HIGH, RATIO_LOW, MIN_ABSOLUTE = 2.0, 0.5, 10

reported["difference"] = reported["doses_administered"] - reported["own_median"]
reported["flag"] = (
    ((reported["ratio"] > RATIO_HIGH) | (reported["ratio"] < RATIO_LOW))
    & (reported["difference"].abs() >= MIN_ABSOLUTE)
)
print(f"{reported['flag'].sum()} flags from {len(reported)} facility-antigen-months")
```

```
reported <- reported |>
  mutate(
    difference = doses_administered - own_median,
    flag = (ratio > 2 | ratio < 0.5) & abs(difference) >= 10
  )
```

Applied here, nothing at all exceeds twice its own median, and eleven facility-antigen-months fall below half of theirs. The largest genuine movement is FAC005's penta3 in October — 13 doses against a median of 43, a real drop of thirty.

That is the finding, and it is one line long. A check that produces one investigable item from 2,094 observations is working; a check that produces two hundred is a check nobody will run twice.

5.5 The direction is not symmetric

A drop and a spike mean different things, and a DQA that treats them the same misses most of what is actually happening.

- **A drop** is usually real or reporting. Stock-out, staff absence, a strike, a facility that started reporting to a different system. Ask about the month, not about the number.
- **A spike** is usually a campaign, an outreach round, or catch-up after a stock-out — all legitimate — or double counting, which is not. Check whether the months either side dip correspondingly, which is the signature of a period boundary problem rather than of extra activity.

```python
reported["previous"] = reported.groupby(["facility_id", "antigen"])[
    "doses_administered"].shift(1)
reported["next"] = reported.groupby(["facility_id", "antigen"])[
    "doses_administered"].shift(-1)

spike_and_dip = reported[
    (reported["ratio"] > 1.5)
    & (reported["previous"] < reported["own_median"] * 0.7)
]
```

```r
reported <- reported |>
  mutate(previous = lag(doses_administered),
         next_value = lead(doses_administered),
         .by = c(facility_id, antigen))
```

A high month preceded by a low one is very often one month's work recorded in the next month's return. That is a timeliness finding wearing a consistency costume, and the fix is a deadline, not a recount.

5.6 Series that are too stable

The opposite check, and the one people forget. Real service delivery is noisy. A facility reporting a nearly identical figure every month is worth a look.

```python
stability = (
    reported.groupby(["facility_id", "antigen"])["doses_administered"]
    .agg(["mean", "std", "size"])
)
stability["cv"] = stability["std"] / stability["mean"]
print(stability[stability["size"] >= 8].nsmallest(5, "cv"))
```

```r
reported |>
  summarise(mean = mean(doses_administered),
            cv = sd(doses_administered) / mean(doses_administered),
            n = n(),
            .by = c(facility_id, antigen)) |>
  filter(n >= 8) |>
```

```
  slice_min(cv, n = 5)
```

The most stable series in this extract has a coefficient of variation of about 0.06 — six percent month to month. That is low, and it is **not evidence of anything**. A large facility with a stable catchment and a regular clinic day can easily produce that.

State the limit plainly, because this check is the one most easily misused: a low coefficient of variation puts a facility on the visit list. It does not go in the report as a finding, and it certainly does not go in a sentence containing the word "fabricated".

5.7 Rank facilities, do not flag rows

The output of this lesson is not a list of flagged months. It is a ranking of reporting units, which is what the sampling lesson needed.

```
risk = (
    reported.groupby("facility_id")
    .agg(flags=("flag", "sum"), months=("flag", "size"))
    .assign(flag_rate=lambda d: d["flags"] / d["months"])
    .sort_values("flag_rate", ascending=False)
)
print(risk.head(6))
```

```
risk <- reported |>
  summarise(flags = sum(flag), months = n(), .by = facility_id) |>
  mutate(flag_rate = flags / months) |>
  arrange(desc(flag_rate))
```

A facility with four flags across twelve months is a candidate for a visit. A single flagged month in a facility that is otherwise steady is a question for a phone call. **The unit of action is the facility, because the vehicle goes to a facility.**

5.8 What comes next

Trend checks compare a number to its own past. The next lesson looks inside the number itself — the last digit, the heaping on multiples of five, and the honest limits of what those can tell you about how a measurement was produced.

CHAPTER 6

What the last digit tells you

Lesson 6 of 8 · 75 min

Digit preference, heaping and the calibration that separates a finding from noise. One team's heights end in .0 or .5 sixty-nine percent of the time; the district's most suspicious facility turns out to be chance.

6.1 Measurements have a texture

A number produced by reading an instrument has a particular texture: the last digit is close to uniform, because the true value is equally likely to fall anywhere within the smallest division of the scale.

A number produced by estimating, rounding, or recalling has a different texture. It clusters on fives and zeros, on whole years, on multiples of ten. That clustering is **digit preference**, and it is measurable without going anywhere.

It is the integrity dimension's main tool, and this lesson spends as much time on what it cannot show as on what it can.

6.2 The check

```python
import pandas as pd

smart = pd.read_csv("smart-nutrition-survey-2024.v1.csv")

smart["terminal"] = (smart["height_cm"] * 10).round() % 10
by_team = (
    smart.assign(rounded=smart["terminal"].isin([0, 5]))
    .groupby("team")
    .agg(rounded_share=("rounded", "mean"), n=("rounded", "size"))
)
print((by_team["rounded_share"] * 100).round(0))
```

```r
library(dplyr)

smart |>
  mutate(terminal = round(height_cm * 10) %% 10,
         rounded = terminal %in% c(0, 5)) |>
  summarise(rounded_share = mean(rounded), n = n(), .by = team)
```

Team	Heights ending .0 or .5	Children measured
1	18%	217
2	**69%**	**248**
3	18%	248
4	23%	217

Two of ten possible terminal digits is 20% under no preference. Teams 1, 3 and 4 sit at 18%, 18% and 23% — which is what measurement looks like. Team 2 sits at 69%.

That is not a marginal result and it does not need a test. Nothing about a population makes one team's children's heights land on half-centimetres three and a half times as often as everybody else's. It is how the team read the board: rounding to the nearest half centimetre instead of reading the millimetre.

6.3 Age heaping, and why it is usually nobody's fault

The same phenomenon on ages, where it is nearly universal in this sector.

```python
ages = smart["age_months"].dropna()
whole_years = (ages % 12 == 0).mean()

print(f"{whole_years:.0%} of ages fall on an exact multiple of 12 months")
print(ages.value_counts().reindex([22, 23, 24, 25, 26]))
```

```r
smart |>
  filter(!is.na(age_months)) |>
  summarise(whole_years = mean(age_months %% 12 == 0))

smart |> count(age_months) |> filter(age_months %in% 22:26)
```

Age in months	22	23	24	25	26
Children	21	15	**66**	19	17

Sixty-six children at exactly twenty-four months against fifteen and nineteen either side, and eighty at thirty-six against twenty-eight and seventeen. Across the survey, **24% of ages fall on an exact whole year, against about 9% expected** if ages were spread evenly.

This matters for a specific, concrete reason and not as a curiosity. The WHO growth standards switch reference table at 24 months, and the SMART age groups are bounded at 12-month marks — so heaping puts a large clump of children exactly on the boundaries where classification changes.

And the cause is almost never carelessness. Birth certificates are not universal; a caregiver asked how old a child is will answer "two years"; a local events calendar helps but has limits. **Write it up as a finding about the age ascertainment method**, with a corrective action about calendars and probing, not as an enumerator error.

6.4 Calibrate against what chance produces

Now the part that separates a usable check from an accusation generator. Run the round-number check on the vaccination extract:

```python
reported = vax[vax["reported"]].copy()
base_rate = (reported["doses_administered"] % 10 == 0).mean()

by_facility = (
    reported.assign(round_number=reported["doses_administered"] % 10 == 0)
    .groupby("facility_id")
    .agg(rounds=("round_number", "sum"), n=("round_number", "size"))
)
by_facility["share"] = by_facility["rounds"] / by_facility["n"]
print(f"district base rate: {base_rate:.3f}")
print(by_facility.nlargest(3, "share"))
```

```
base_rate <- mean(vax$doses_administered[vax$report_submitted] %% 10 == 0)

by_facility <- vax |>
  filter(report_submitted) |>
  summarise(rounds = sum(doses_administered %% 10 == 0), n = n(), .by = facility_id) |>
  mutate(share = rounds / n) |>
  arrange(desc(share))
```

The district base rate is 10.4% — exactly what you would expect. The worst facility, FAC020, reports a round number 14 times out of 60, or 23%. More than double the district rate. On its own that looks like a finding.

Test it:

```
from scipy.stats import binomtest

worst = by_facility.nlargest(1, "share").iloc[0]
p = binomtest(int(worst["rounds"]), int(worst["n"]), base_rate,
              alternative="greater").pvalue
print(f"p = {p:.4f}, Bonferroni across 38 facilities: {min(1, p * 38):.3f}")
```

```
worst <- by_facility[1, ]
p <- binom.test(worst$rounds, worst$n, base_rate, alternative = "greater")$p.value
c(p = p, bonferroni = min(1, p * 38))
```

p = 0.003 on its own, and 0.11 after correcting for having looked at thirty-eight facilities. Three facilities fall below p = 0.05 uncorrected, and 1.9 are expected by chance. Three against 1.9 is nothing.

There is a second reason to let it go. FAC020's monthly doses run from five to thirteen, so "ends in zero" almost always means the value was exactly ten. On counts that small, the check has no power to distinguish rounding from arithmetic.

> The result is: **no evidence of digit preference in the vaccination reporting.** That is a real finding, it took four lines, and it is the one you should hope for.

6.5 Why the null case matters more

Most of what makes this check dangerous is that it is almost always run on data where nothing is wrong, by someone hoping to find something.

Three rules that keep it honest.

- **Compute the base rate from the data**, not from theory. Terminal digits are not uniform when counts are small, when a form has a default value, or when doses come in vials of ten.
- **Correct for the number of units you looked at.** Testing thirty-eight facilities at 5% produces about two positives from clean data, every time.
- **Require a mechanism.** Team 2's 69% has one — the team read the measuring board to the nearest half centimetre. If you cannot name a plausible mechanism, you have a number and not a finding.

6.6 What you may write

The wording is not a courtesy; it decides what happens next.

Do not write	Write
"FAC020's data appear fabricated"	"FAC020 reports round numbers more often than the district average; the
"Team 2's measurements are unreliable"	"Team 2's height readings end in .0 or .5 in 69% of cases against 18–23% f
"Ages are inaccurate"	"24% of ages fall on an exact whole year against 9% expected, indicating a

The right-hand column says what was observed, how large it is, and what mechanism it is consistent with. The left-hand column says what somebody did. Only one of the two produces a corrective action, and only one of them survives being shown to the team it describes.

6.7 What comes next

You now have findings — a completeness collapse, a verification factor of 1.42, a team rounding its heights. None of them is yet a cause, and none of them can be acted on. The next lesson is how to get from a defect to the thing that produced it, and why "the staff need training" is almost always the wrong answer.

Part IV

Turning findings into change

CHAPTER 7

From a defect to the thing that caused it

Lesson 7 of 8 · 60 min

Five categories of cause, a stopping rule for the five whys, and why "the staff need training" is the answer that gets written when nobody looked. Test the cause against the data before you put it in the report.

7.1 A finding is not a cause, and only a cause can be fixed

At this point you have findings. Reporting collapsed to 29% in August. One school under-reported attendance by 42%. One survey team rounds heights to the nearest half centimetre. Twenty-four percent of ages are whole years.

Every one of those is a *what*. None of them is a *why*, and a corrective action plan written against a *what* produces the sentence that appears in most DQA reports in this sector:

> **Recommendation.** Train staff on data quality.

That sentence is written when nobody looked for a cause. It is unfalsifiable, it is expensive, and it will not change August's reporting rate, because nothing about August was caused by staff not knowing what they were doing.

7.2 Five categories, and they need different money

Almost every data quality cause in routine programme data falls into one of five categories. Naming which one you are in tells you who has to fix it.

Category	Looks like	Fixed by
Definition	Two sites counting different things under one indicator name	A written indic
Tool	A form that cannot express what happened, or a system that drops values	A form or confi
Capacity	The right tool used wrongly, consistently, by someone who was never shown	Supervision and
Incentive	Reporting that improves when it is looked at, or numbers that meet a target exactly	Changing what
System	Everyone fails at once — no forms, no fuel, no network, no supervisor	Logistics, budge

Two observations about that table, and both come up in every assessment.

Definition is the largest category and the least suspected. A verification factor far from 1 is more often two people counting different things than anyone counting wrongly. It is also the cheapest to fix, and the fix — writing the definition down — prevents recurrence permanently.

Capacity is the smallest category and the most frequently blamed, because it is the only one where the fix is a workshop and a workshop is procurable.

7.3 Let the data narrow it before you ask anyone

The pattern of a defect tells you which category it is in, and you already have the data to see the pattern.

```python
by_month = vax.groupby(vax["period"].dt.strftime("%Y-%m"))["reported"].mean()
by_facility = vax.groupby("facility_id")["reported"].mean()

print("spread across months:  ", round(by_month.max() - by_month.min(), 2))
print("spread across facilities:", round(by_facility.max() - by_facility.min(), 2))
```

```r
by_month <- vax |> mutate(m = format(period, "%Y-%m")) |>
  summarise(r = mean(report_submitted), .by = m)
by_facility <- vax |> summarise(r = mean(report_submitted), .by = facility_id)

c(months = diff(range(by_month$r)), facilities = diff(range(by_facility$r)))
```

Three shapes, three categories:

- **Concentrated in time, spread across units.** August, everywhere. That is a system cause — something failed for the district at once. Twenty-seven facility findings would be twenty-seven wrong answers.
- **Concentrated in one unit, spread across time.** SCH09's `Y/N` coding, every month. That is a tool or definition cause local to one site.
- **Spread across both.** Age heaping, every team, all year. That is a method cause, inherent to how the measurement is taken at all.

Write the shape down before you propose a cause. It rules out three of the five categories for free, and it is the step that stops a system failure being written up as twenty-seven capacity failures.

7.4 Then ask the form

For anything local, look at the instrument before you look at the person.

- What values can the form actually accept? SCH09's registers offered `Y/N` boxes; the extract's boolean cast recognised `true/false`. **Neither the teacher nor the analyst did anything wrong**, and no training fixes it.
- Is the field required? An optional field with a 40% missing rate is a form design decision, not a compliance problem.
- What does the field label say? A column headed "cases" collects whatever the person filling it believes a case is.

7.5 The five whys, with a stopping rule

The technique is standard. What is usually missing is a rule for when to stop, which is why five whys so often terminates at "staff are not motivated".

Stop when you reach something a named person can change with a budget you can estimate. Anything past that point is philosophy.

> Attendance was under-reported by 42% in SCH09. — *Why?* The extract read `Y` as missing. — *Why?* The register used `Y/N` and the form expected `true/false`. — *Why?* The paper register printed in 2019 has yes/no boxes; the digital form was designed later, separately. — **Stop.** The education office can reprint the register or the HMIS

> team can accept both codings. Both are costed in an afternoon.

Going further — *why were they designed separately?* — produces true statements about institutional coordination that no one in this assessment can act on.

7.6 Test the cause against the data

A proposed cause makes a prediction. Check it before you write it down; it takes one query and it is the difference between a finding and a story.

```python
# Cause: one school's register uses Y/N. Prediction: Y/N appears in that
# school only, and in most of its months.
yn = attendance[attendance["present"].isin(["Y", "N"])].merge(
    roster[["student_id", "school_id"]], on="student_id", how="left"
)
print(yn["school_id"].value_counts())
print(yn.groupby("school_id")["attendance_date"].nunique())
```

```r
attendance |>
  filter(present %in% c("Y", "N")) |>
  left_join(select(roster, student_id, school_id), by = "student_id") |>
  count(school_id)
```

If the `Y/N` values turned out to be spread across six schools, the cause is not one register — it is a regional convention, and the fix is different. **A cause that survives its own test is worth writing; one that was never tested is a guess in the voice of a conclusion.**

7.7 Put the cause at the level the fix lives at

The last discipline, and it decides whether the report is fair.

- A **facility-level** cause needs a facility-level action. "FAC020's tally sheet is missing" is fixed by delivering a tally sheet.
- A **district-level** cause needs a district-level action. "No facility received forms in August" is not thirty-eight findings.
- A **system-level** cause needs a system-level action, and naming it is often the most valuable thing in the report even though nobody in the room can fix it today.

Attributing a district-level cause to facilities is the single most damaging error in DQA writing. It produces defensive reporting, the numbers get smoothed, and next quarter's assessment finds a cleaner dataset that is less true.

> The purpose of naming causes is that the next round is better. A report that assigns blame accurately and changes nothing has failed at the only thing it was for.

7.8 What comes next

You have findings, and now each has a cause with a level and a test behind it. The last lesson is the document — what goes in it, in what order, and the four columns that turn a finding into something that is still true when somebody checks in three months.

CHAPTER 8

The report that changes the next round

Lesson 8 of 8 · 60 min

Findings, root cause, corrective action, owner, deadline — five columns, one row per finding. Plus the follow-up round that is the only evidence any of it worked.

8.1 Who actually reads it

Three people, and they read three different things.

- **The programme manager** reads the summary and wants to know whether the figures in last quarter's report can stand.
- **The HMIS or M&E focal point** reads the finding table and wants to know what they are being asked to do.
- **The donor's monitoring adviser**, six months later, reads the methods section and wants to know whether your findings were arrived at defensibly.

Write for all three, in that order, and the structure follows from it.

8.2 The order

1. **What was assessed.** Dataset, period, reporting units in scope, indicators examined. Four lines.
2. **How.** Which dimensions, which desk checks, which facilities were visited and *why those* — the `reason` column from the sampling lesson.
3. **The scorecard.** Five dimensions, five measures, with the one you could not measure shown as a finding rather than a blank.
4. **Findings.** The table below.
5. **What this means for published figures.** The section everyone skips and the programme manager needs.
6. **Limitations.** What the sample cannot support, what you could not verify.

Section five is worth insisting on. A DQA that documents defects without saying whether last quarter's coverage figure should be reissued has left the decision to somebody who now has to read all six sections to make it.

8.3 The finding table

One row per finding, and five columns that are not negotiable.

#	Finding	Root cause
1	Reporting rate fell to 29% in August and 45% in September, district-wide	System: form supply interrupted, a
2	SCH09 under-reported attendance by 42% (VF 1.42)	Tool: register prints Y/N boxes; th
3	Survey team 2 rounds heights to the nearest 0.5 cm (69% vs 18–23%)	Capacity: board read to the half ce
4	Timeliness cannot be assessed; no submission date in the extract	Tool: completeness date not carried
5	24% of ages fall on an exact whole year (9% expected)	Method: age estimated rather than

Three properties make that table work.

Every finding has a number in it. "Reporting was poor in August" is not a finding; "29%, against 82–92% in other months" is. The number is what makes the follow-up round able to say whether anything changed.

Every owner is a role, and a person holds it. "The district" cannot be chased. Write the role, and check that somebody actually occupies it — an action owned by a vacant post is a finding of its own.

Every deadline is a date. "Ongoing" and "as soon as possible" both mean never, and everyone in the room knows it when it is written.

8.4 Findings a reader can check

Write each one so that someone with the same data could reproduce it. That is what separates a DQA from an opinion, and it costs one clause.

```
findings = pd.DataFrame([
    {
        "id": 2,
        "dimension": "Accuracy",
        "unit": "SCH09",
        "measure": "verification factor",
        "value": 1.416,
        "comparator": "district 1.012, 23 of 24 schools exactly 1.000",
        "evidence": "outputs/verification-factors.csv",
    },
])
findings.to_csv("outputs/dqa-findings.csv", index=False)
```

```
findings <- tibble::tribble(
  ~id, ~dimension,  ~unit,   ~measure,              ~value, ~comparator,
   2L, "Accuracy",  "SCH09", "verification factor",  1.416, "district 1.012"
)

readr::write_csv(findings, here::here("outputs", "dqa-findings.csv"))
```

The `comparator` column is the one people leave out. A verification factor of 1.42 means nothing until the reader knows the other twenty-three schools were at 1.000 — that comparison is what turns a number into a finding.

The `evidence` column points at the file that produced it, which is the same discipline as the cleaning log: **the artefact travels with the claim.**

8.5 Score the dimensions, do not grade the district

Publish the scorecard as five measures with their sources. If a composite is demanded, put it beside them and state the weights.

```python
scorecard = pd.DataFrame({
    "dimension": ["Completeness", "Timeliness", "Accuracy", "Consistency", "Integrity"],
    "measure": ["Reporting rate", "Not assessable", "Verification factor",
                "Outlier rate", "Round-number share"],
    "value": ["76.5%", "-", "1.012 district / 1.42 worst unit",
              "11 of 2,094 below half own median", "10.4%, chance level"],
    "finding": [1, 4, 2, None, None],
})
```

```r
scorecard <- tibble::tribble(
  ~dimension,     ~measure,              ~value,                     ~finding,
  "Completeness", "Reporting rate",      "76.5%",                    1L,
  "Timeliness",   "Not assessable",      "-",                        4L,
  "Accuracy",     "Verification factor", "1.012 district / 1.42 max", 2L,
  "Consistency",  "Outlier rate",        "11 of 2,094",              NA_integer_,
  "Integrity",    "Round-number share",  "10.4%, chance level",      NA_integer_
)
```

Note the accuracy row carries two numbers. **A dimension summarised by one number hides the distribution**, and the distribution was the whole finding.

Note also the two rows with no finding attached. Integrity and consistency came back clean, and saying so explicitly is worth as much as the findings — it tells the reader which checks were run and passed, rather than leaving them to wonder whether they were run at all.

8.6 The follow-up round is the point

A DQA is worth nothing on its own. It becomes worth something when the same checks run again and the numbers move.

```python
history = pd.concat([
    pd.read_csv(p).assign(round=p.stem)
    for p in sorted(Path("outputs/dqa").glob("*.csv"))
])
print(history.pivot_table(index="measure", columns="round", values="value"))
```

```r
history <- purrr::map_dfr(
  list.files(here::here("outputs", "dqa"), full.names = TRUE),
  ~ readr::read_csv(.x) |> dplyr::mutate(round = tools::file_path_sans_ext(basename(.x)))
)

tidyr::pivot_wider(history, id_cols = measure, names_from = round, values_from = value)
```

Measure	Q3	Q4	Movement
Reporting rate	76.5%	88.1%	+11.6 pt
Schools outside VF tolerance	1 of 24	0 of 24	closed
Team 2 rounded heights	69%	24%	closed

That table is the deliverable that justifies the work. It also gives every corrective action a verdict — done, not done, or done and did not help — which is the accountability the CHS commitment on information management is actually asking for.

Carry every open action forward. An action that appears in three consecutive reports without closing is itself a finding, and usually a system-level one.

8.7 Presenting it to the people in it

The last thing, and it decides whether the next round is better or quieter.

- **Show it to them first.** A finding a facility sees for the first time in a donor report will be contested, and the contest will be about your method rather than about the data.
- **Lead with the system findings.** Starting with the August form supply establishes that the assessment is looking at the whole chain, not at individuals.
- **Name what was clean.** Twenty-three of twenty-four schools were exact. Say it before you say the twenty-fourth was not.
- **Ask them for the cause.** The people who filled in the register usually know why it looks like that, and they will tell you if the meeting is not a tribunal.

> The measure of a data quality assessment is not how many findings it produced. It is whether next quarter's data is better, and defensive reporting is the one outcome that guarantees it will not be.

8.8 Where this course, and this module, leave you

You can profile a fresh export and log every change you make to it. You can put several files together and prove the joins did what you said. And you can now assess the result the way an auditor will — five dimensions with measures, a recount against the source, a sample you can defend, and a report where every finding has an owner and a date.

That is module 2 complete, and it is the module every sector course afterwards assumes.

Module 3, *Indicators and Measurement*, is what all of this was preparation for. It starts with *Indicator Design and the LogFrame* — writing definitions precise enough that two analysts computing them separately get the same number, which is the defect underneath more than half the findings in this course.

About this edition

This PDF is generated from the same source as the web version of the course, so the two cannot drift. The web edition carries the runnable code, the downloadable datasets and any corrections issued since this file was produced.

> Every dataset referenced in this course is synthetic or fully de-identified. No record traces to a real person.

This course is published in English and in French. The French edition is a professional adaptation using the terminology UNICEF, WHO, WFP, OCHA and national ministries publish in — not a literal translation.

Read and run the course online at data-analysis.cassion.dev/courses/data-quality-assessment

Generated July 28, 2026.