IT SHOULD WORK, BUT IT DOESN'T

The Cognos Cookbook

Recipes for the reports that should work, but don't

> **This is one complete recipe from the book**, reproduced exactly as it ships, together with the evidence key every recipe is graded against. Nothing here is summarised or softened for the sample. If the full book disagrees with your release, it says so in the same sentence, which is the point of the key on the next page.

RobCo**Tek**

How to read a recipe

Every recipe ends with a **Status** line, and it is not decoration. It names exactly what grade of evidence the recipe rests on, so you never have to guess whether you are reading something we ran or something we read. The grades:

Grade	What it means
VERIFIED	Reproduced on the live Cognos Analytics 12.1.2 trial. The behavior was made to happen and the result recorded.
ENGINE-BASIS VERIFIED	Reproduced on the database bench (SQL Server, Oracle, Db2, PostgreSQL, Spark) but not through Cognos. The SQL behavior is measured; how Cognos wraps it is stated separately.
TIER-A SOURCED / DOCUMENTATION VERIFIED	Taken from IBM's own current documentation and *not* reproduced here. Reliable as a statement of what IBM says; not evidence of what your release does.
NOT REPRODUCIBLE BY DESIGN	Cannot be provoked on demand, and the reason is given. Deliberately never upgraded to VERIFIED.
BUILT AND RUNNING	Used only for the bench itself, in Appendix A.

A qualifier in parentheses narrows the claim to the part that was captured: **VERIFIED (mechanism)** means the mechanism was reproduced while some illustrative render was not, and **VERIFIED (core)** means the load-bearing behavior was captured while a secondary detail was not. Where a recipe is captured in part, the Status line says which part, in the same sentence.

Two habits follow from this, and they are the honest ones. Trust a VERIFIED claim about 12.1.2 more than the same claim about your release, because versions drift. And treat a TIER-A SOURCED line as a lead to check in your own environment rather than as a finding, because that is exactly what it is.

Recipe 2. A masked column does not add up to its own total

Symptom. A column carries a two-decimal display mask and an overall total. Read the column down the screen, add the numbers up, and you get a different answer than the total row shows, usually by a cent or two. Nothing errors, every individual cell looks right, and the same disagreement survives the export to Excel. A reviewer adding up a printed page will find it before you do.

Why. Cognos totals the **underlying** values and formats the result; it does not total what is on the screen. The mask rounds each row for display only, so the rows you can see are each off by a fraction of a cent, those fractions accumulate, and the total, which never saw the rounding, does not match the sum of the rounded rows. Measured on 12.1.2: five rows displaying `357.00 / 595.67 / 387.67 / 638.67 / 437.00` add to `2,416.01`, and the total reads `2,416.00`, which is the exact full-precision figure.

Exporting to Excel does not expose the difference and does not fix it. Excel gets the full-precision numbers *plus* the mask as a cell format, so the total cell agrees with the report total, `SUM()` over the column agrees with both, and the visible cells still refuse to add up. The inconsistency is between displayed and underlying values, and it travels into every render target identically.

> **Correction to a widely repeated explanation.** This is often described as a screen-versus-Excel disagreement, on the theory that the screen totals rounded values while Excel sums full-precision ones. That is not what 12.1.2 does. Both surfaces total the underlying values and both produce the same number. Do not go looking for a discrepancy between the two outputs; there isn't one. The discrepancy is inside a single output, between the rows and the total.

Fix. You have to decide which property you actually need, because you cannot have both.

- **If the column must add up** — an invoice, a reconciliation, anything a reader will total by hand — round the **value**, not the display. Change the expression from `[quantity] / 3` to `round([quantity] / 3, 2)` and drop the mask's claim to authority. Measured: the rows become `356.99 / 595.67 / 387.66 / 638.66 / 436.98`, the total becomes `2,415.96`, and the rows add to the total exactly, on screen and in Excel alike.
- **If the total must be correct**, keep the mask and stop presenting a total that invites addition: show the total to full precision, or label it as computed on unrounded values, or remove the visible row total and let the number stand alone.

The trade you are making is real, and it is easy to miss. In the measured case, rounding the value moved the total from `2,416.00` to `2,415.96`. The true total is `2,416.00`. Forcing the column to add up made the total wrong by four cents. Consistency and accuracy are in direct conflict here; pick the one the report is for.

Watch out. `round()` is applied **below the grain you are looking at**, on the detail rows, before aggregation. Region 4 in the measured run makes this visible: quantity `1,071` divided by 3 is exactly `357`, so rounding at the displayed grain would leave `357.00` untouched. The report shows `356.99`. Rounding is happening on rows you never see and the residuals accumulate downward, which is why the tied total drifts low rather than landing on the true figure. Do not assume `round()` on an aggregated column rounds the number in front of you.

Two further traps. A display-only format mask is never a rounding instruction to anything downstream; Excel formats with it but sums the raw value underneath, so a mask can never make figures reconcile. And if a number has to tie out to an external figure, round it at the source or in the expression and verify the tie-out against that external figure, not against the report's own total.

Status: VERIFIED 2026-07-26 (12.1.2 On Demand). Evidence: `lab/trial_capture/rounding/FINDINGS.md`. Default totals confirmed computed on underlying values; the screen-versus-Excel framing corrected; the `round()` accuracy cost and sub-grain application newly documented. Report-spec shape for a list total (`<dataItemListSummary>` in the query plus a `<listOverallGroup>` `<listFooter>` referencing `Total(<item>)`) recorded in the same file.

The rest of the book

47 recipes across prompts and parameters, filters and suppression, formatting, expressions and sorting, rendering and export, drill-through, plus a supplement on native and pass-through SQL. An error index that names the layer that raised each code, a five-engine dialect matrix, and the lab build that produced the evidence.

Every claim carries its grade. Where the commonly repeated fix does not work on 12.1.2, the recipe says so and gives the one that does. A workaround that fails at execution costs more than no workaround at all.

robcotek.com