



Data Analyzer – Calculated Value Formulas

Rev. (Y-M-D): 2026-06



Contents

1. Purpose of this document.....	2
2. How calculated values work.....	2
2.1. Two levels of calculation.....	2
3. Creating a calculated value from the interface.....	2
3.1. Editing and removing a calculated value.....	3
4. Formula syntax.....	3
4.1. Referencing an indicator.....	3
4.2. Available aggregation functions.....	3
4.3. Operators and formula functions.....	4
4.4. Negative numbers.....	4
4.5. Nesting and readability.....	4
5. Worked examples.....	5
5.1. Ratios and averages.....	5
5.2. Margins, budgets and variances.....	5
5.3. Shares and percentages.....	6
5.4. Conditional logic.....	6
5.5. Scaling, rounding and readability.....	7
5.6. Row-level calculations.....	8
5.7. Trend and comparison.....	8
6. Aggregated versus individual calculation.....	9
7. Formatting calculated values.....	9
7.1. Number formatting.....	9
7.2. Conditional formatting.....	9
8. Limitations.....	9
8.1. What formulas cannot do.....	9
8.2. How complex a formula can be.....	10
8.3. Data volume: two feeding modes.....	10
8.4. Browser requirements.....	11
9. Offline behaviour.....	11
10. Annex – Troubleshooting common errors.....	13



1. Purpose of this document

This guide explains how to build and use calculated value formulas in the Data Analyzer, the pivot table component of N4V FOR WEBI for SAP BusinessObjects Web Intelligence.

It targets two audiences:

- End users, who create their own indicators directly in the Data Analyzer interface, without writing code.
- Report designers, who prepare calculated values when configuring the component, and need the full syntax reference and edge cases.

It also documents how far formulas can be pushed, the limitations of the Data Analyzer as a whole, and how the component behaves in offline deliverables (chapters 8 and 9).

This document covers calculations performed inside the Data Analyzer. Web Intelligence formulas (variables, measures, calculation contexts) defined upstream in the WebI document are covered by separate documentation.

2. How calculated values work

The Data Analyzer displays aggregated data: every cell in the pivot table is the result of an aggregation function applied to a set of rows from the WebI microcube.

A calculated value is an additional indicator, absent from the source dataset, obtained by combining existing indicators through a formula. Once created, it behaves like any other measure: it can be placed in the Values box, formatted, sorted and exported.

2.1. TWO LEVELS OF CALCULATION

A key distinction:

- Aggregated calculation (default): indicators are aggregated first, then the formula is applied to the totals.
- Individual calculation: the formula is applied row by row on the raw values, and the results are then aggregated.

This choice changes the displayed result. See chapter 6.

3. Creating a calculated value from the interface

No technical knowledge is required for this procedure.

1. Open the Field List of the Data Analyzer.
2. Click the Add calculated value button at the top of the panel.
3. Enter a meaningful name for the new indicator. This name becomes the column header displayed in the table.
4. Build the formula: drag existing indicators into the formula box and combine them with arithmetic operators.
5. Tick the Calculate individual values option if required (see chapter 6).
6. Confirm with Apply. The indicator is added to the Field List and becomes immediately available in the Values box.

Calculated values are grouped in a dedicated folder of the Field List. They are saved with the report and restored when the document is reopened.



3.1. EDITING AND REMOVING A CALCULATED VALUE

An edit button appears next to each calculated value in the Field List, both in the All fields box and in the Values box. It reopens the formula window, which also contains the Remove button; a confirmation is requested.

The Add calculated value button can be hidden to lock down a dashboard intended for distribution. This is set in the component configuration.

4. Formula syntax

4.1. REFERENCING AN INDICATOR

An indicator can never be used on its own: it must always be wrapped in an aggregation function, in parentheses, with its name enclosed in double quotes.

```
sum("Revenue") / count("Orders")
```

Writing "Revenue" * 2 without an aggregation function raises a formula validation error.

4.2. AVAILABLE AGGREGATION FUNCTIONS

These functions determine how data rows are grouped before the formula is applied.

Function	Description
sum	Sum of the values. Default aggregation for numeric fields.
count	Number of values. Default aggregation for text fields.
distinctcount	Number of distinct values.
average	Arithmetic mean.
median	Median value.
product	Product of the values.
min	Minimum value.
max	Maximum value. Default aggregation for date fields.
stdevp	Population standard deviation.
stdevs	Sample standard deviation.
percent	Percentage of the grand total.
percentofcolumn	Percentage of the column total.
percentofrow	Percentage of the row total.
difference	Difference against the neighbouring hierarchy member.
differencepercent	Percentage difference against the neighbouring member.
runningtotals	Running total.
index	Relative weighting index of the cell.

The difference, differencepercent and runningtotals aggregations are computed against neighbouring members of the displayed hierarchy. The result therefore depends on the sorting and filtering applied to the table: to obtain a growth rate over a given period, filter the date dimension so that only the periods to be compared are displayed.



4.3. OPERATORS AND FORMULA FUNCTIONS

Operator / Function	Description and syntax
+	Addition. $a + b$
-	Subtraction. $a - b$
*	Multiplication. $a * b$
/	Division. a / b
^	Power. a^2
<	Less than. $a < b$
<=	Less than or equal. $a <= b$
>	Greater than. $a > b$
>=	Greater than or equal. $a >= b$
==	Equal to. $a == b$
and	Logical AND. $a \text{ and } b$
or	Logical OR. $a \text{ or } b$
if	Condition. $\text{if}(\text{condition}, \text{then}, \text{else})$. The third argument is optional: without it, cells that do not meet the condition are left empty.
abs	Absolute value. $\text{abs}(\text{number})$
round	Rounding. $\text{round}(\text{number}, \text{decimals})$. Without the second argument, rounds to the nearest integer.
min	Minimum of two values. $\text{min}(\text{number1}, \text{number2})$
max	Maximum of two values. $\text{max}(\text{number1}, \text{number2})$
isNaN	True when the value is not a number. $\text{isNaN}(\text{value})$
!isNaN	True when the value is a number. $\text{!isNaN}(\text{value})$

min and max exist in two distinct forms: as an aggregation over one indicator — $\text{min}(\text{"Price"})$ — or as a comparison between two expressions — $\text{min}(\text{sum}(\text{"Price"}), 1000)$.

4.4. NEGATIVE NUMBERS

A negative number must be enclosed in parentheses to be interpreted correctly:

```
(-1) * sum("Margin")
```

4.5. NESTING AND READABILITY

Formulas can be nested to any reasonable depth, and line breaks are allowed inside the formula box. Long expressions are easier to maintain when each condition sits on its own line.

Two habits keep a formula library manageable over time: naming calculated values after the business concept rather than the arithmetic, and building complex indicators in stages instead of one long expression.



5. Worked examples

The examples below use generic indicator names — replace them with the ones from your universe. They are grouped by business intent so that the closest pattern can be located quickly.

5.1. RATIOS AND AVERAGES

Average selling price

Unit value of what was actually sold, rather than an average of averages.

```
sum("Revenue") / sum("Quantity Sold")
```

Average order value

Revenue divided by the true number of orders, not by the number of table rows.

```
sum("Revenue") / distinctcount("Order Number")
```

distinctcount avoids inflating the denominator when an order spans several product lines.

Average revenue per customer

Customer value indicator, unaffected by repeat purchases.

```
sum("Revenue") / distinctcount("Customer ID")
```

Units per transaction

Basket depth, useful for retail and point-of-sale analysis.

```
sum("Quantity Sold") / distinctcount("Transaction ID")
```

Revenue per employee

Productivity ratio combining a financial measure and a headcount measure.

```
sum("Revenue") / distinctcount("Employee ID")
```

Average handling time in hours

Converting a duration expressed in minutes into a readable unit.

```
average("Handling Time Minutes") / 60
```

5.2. MARGINS, BUDGETS AND VARIANCES

Margin rate

Profitability expressed as a percentage of revenue.

```
(sum("Margin") / sum("Revenue")) * 100
```

Gross margin from cost

Rebuilding margin when only revenue and cost of goods are available.

```
sum("Revenue") - sum("Cost of Goods")
```

Budget variance in value

Absolute gap between actuals and budget. A negative result means underspend.

```
sum("Actual") - sum("Budget")
```



Budget variance in percent

Same gap expressed relative to the budget, comparable across cost centres.

```
((sum("Actual") - sum("Budget")) / sum("Budget")) * 100
```

Budget achievement rate

Consumption of the allocated budget, capped for readability.

```
min((sum("Actual") / sum("Budget")) * 100, 100)
```

Absolute deviation

Magnitude of the gap regardless of direction, for ranking the largest anomalies.

```
abs(sum("Actual") - sum("Forecast"))
```

5.3. SHARES AND PERCENTAGES

Share of grand total

Contribution of each row to the overall total.

```
percent("Revenue")
```

Prefer the built-in percent aggregation over a manual ratio: it follows filters and drills automatically.

Share of column total

Vertical distribution, typically the product mix within each period.

```
percentofcolumn("Revenue")
```

Share expressed as a formatted percentage

Use when the display format is itself declared as a percentage.

```
percent("Revenue") / 100
```

Dividing by 100 prevents the value being multiplied by a hundred on screen.

Conversion rate

Ratio between two event counts, expressed as a percentage.

```
(distinctcount("Order ID") / distinctcount("Visit ID")) * 100
```

Weight of one component in a total

Share of a sub-measure within a broader measure.

```
(sum("Freight Cost") / sum("Total Cost")) * 100
```

5.4. CONDITIONAL LOGIC

Safe division

Prevents an error or an empty cell when the denominator is zero.

```
if(sum("Quantity Sold") > 0,  
  sum("Revenue") / sum("Quantity Sold"),  
  0)
```

Threshold flag



Returns 1 or 0, ready to drive conditional formatting.

```
if(sum("Revenue") > 100000, 1, 0)
```

Three-tier segmentation

Nested conditions classify each row into 3, 2 or 1.

```
if(sum("Revenue") > 500000, 3,  
  if(sum("Revenue") > 100000, 2, 1))
```

Multi-criteria selection

Combines two conditions; the result is 1 for true and 0 for false.

```
average("Price") < 4000 and sum("Quantity Sold") > 100
```

Alternative criteria

Flags a row that meets either of two conditions.

```
sum("Overdue Days") > 30 or sum("Disputed Amount") > 0
```

Value shown only above a threshold

Omitting the third argument leaves the other cells empty, which declutters the table.

```
if(sum("Revenue") > 50000, sum("Revenue"))
```

Neutralising non-numeric results

Replaces an undefined result with zero.

```
if(isNaN(sum("Margin") / sum("Revenue")), 0,  
  sum("Margin") / sum("Revenue"))
```

Capped bonus calculation

Applies a rate to revenue while enforcing a ceiling.

```
min(sum("Revenue") * 0.03, 15000)
```

5.5. SCALING, ROUNDING AND READABILITY

Amount in thousands

Reduces column width on large figures.

```
round(sum("Revenue") / 1000, 0)
```

Amount in millions with one decimal

Executive-level reading of a financial table.

```
round(sum("Revenue") / 1000000, 1)
```

Currency conversion

Applies a fixed rate. For a variable rate, prepare the converted measure upstream in WebI.

```
sum("Revenue") * 1.08
```

Inverted sign

Displays costs or credits as positive values.



```
(-1) * sum("Cost")
```

Squared deviation

Building block for dispersion calculations.

```
(sum("Actual") - sum("Target"))^2
```

5.6. ROW-LEVEL CALCULATIONS

The formulas in this section only produce the expected result when Calculate individual values is enabled (chapter 6).

Revenue rebuilt from quantity and unit price

The multiplication must happen row by row before any aggregation.

```
sum("Quantity") * sum("Unit Price")
```

Enable Calculate individual values, otherwise total quantity is multiplied by total price.

Discounted revenue

Applies a discount rate that varies from one row to the next.

```
sum("Quantity") * sum("Unit Price") * (1 - sum("Discount Rate"))
```

Weighted value

Weighting each row by its own coefficient before summing.

```
sum("Score") * sum("Weight")
```

5.7. TREND AND COMPARISON

Period-on-period change

Uses the built-in aggregation, computed against the neighbouring member of the hierarchy.

```
difference("Revenue")
```

Filter the date dimension so that only the two periods to be compared are displayed.

Growth rate

Percentage version of the same comparison.

```
differencepercent("Revenue")
```

Cumulative revenue

Year-to-date reading along the displayed hierarchy.

```
runningtotals("Revenue")
```

Gap to the best performer

Distance between each row and the maximum observed value.

```
max("Revenue") - sum("Revenue")
```



6. Aggregated versus individual calculation

This is the most common source of confusion with calculated values. Consider a dataset reduced to two rows:

Row	Quantity	Price
1	7	100
2	5	200

With the formula `sum("Quantity") * sum("Price")`:

- Aggregated calculation (option cleared): $(7 + 5) \times (100 + 200) = 3,600$.
- Individual calculation (option ticked): $(7 \times 100) + (5 \times 200) = 1,700$.

Individual calculation is the expected behaviour whenever a formula multiplies two indicators row by row — typically revenue rebuilt from a quantity and a unit price. Aggregated calculation suits ratios of totals: margin over revenue, market share, budget achievement.

Rule of thumb: if the formula multiplies two indicators, enable individual calculation. If it divides two totals, leave it disabled.

7. Formatting calculated values

7.1. NUMBER FORMATTING

A calculated value is formatted like any other indicator: decimal places, thousands separator, currency symbol, percentage display. The format is applied from the column context menu or in the component configuration.

Watch out for percentages: the `percent` aggregation returns a value already expressed on a base of 100. If the display format is itself declared as a percentage, divide the formula by 100 to avoid a hundredfold display.

7.2. CONDITIONAL FORMATTING

Conditional formatting applies to calculated values just as it does to source indicators. The condition addresses the cell value through a dedicated marker:

```
#value = 1
```

Combined with a threshold flag (section 5.4), this highlights the rows that meet a business criterion without writing any code.

8. Limitations

This chapter answers two distinct questions: how complex a formula can be, and what the Data Analyzer can handle as a component.

8.1. WHAT FORMULAS CANNOT DO

- Calculated values operate on numeric indicators. String comparison is not supported: to test a dimension, convert the information into a numeric value beforehand.
- Logarithmic and trigonometric functions are not available. Such calculations must be prepared upstream in the Web Intelligence document.



- There is no cell addressing. A formula cannot reference a specific cell, a neighbouring column or a value from another row. Period-on-period logic must go through the difference, differencepercent and runningtotals aggregations, which follow the displayed hierarchy.
- There is no calculation context comparable to the WebI ForEach / In / ForAll operators. A calculated value is evaluated within the context of the cell that displays it.
- Indicator names must be reproduced exactly as they appear in the report, including accents, spaces and apostrophes. An apostrophe inside a name is not a problem, since names are delimited by double quotes.
- A malformed formula is reported by a validation message on save. The most frequent causes are a missing aggregation function, missing double quotes around the indicator name, and a misspelled indicator name.
- A calculated value references indicators by name. Renaming an indicator in the WebI document breaks the formulas that use it: the affected calculated values must be updated.
- Grand totals for a calculated value are obtained by applying the formula to the totals, not by adding up the row results. This is the expected behaviour for a ratio, but can be surprising for other calculation types.

When a calculation cannot be expressed with the available operators, the workaround is always the same: prepare a variable in the Web Intelligence document and expose it to the Data Analyzer as a standard measure.

8.2. HOW COMPLEX A FORMULA CAN BE

There is no documented ceiling on the length of a formula, the depth of nested conditions, or the number of calculated values in a single table. In practice the limit is not syntactic but computational.

Three factors drive the cost:

- Recalculation frequency. Every calculated value is re-evaluated on refresh, and on every layout change — expanding a hierarchy, sorting, filtering, switching between grid and chart.
- Row-level evaluation. A formula with Calculate individual values enabled is evaluated on each source row rather than on aggregated totals, which is significantly heavier on large datasets.
- Nesting. Deeply nested conditions multiply the number of evaluations per cell, but the effect stays marginal compared with the two factors above.

Practical guidance: a handful of nested conditions over a few tens of thousands of rows — the usual size of a WebI-fed table — is comfortable. Dozens of calculated values, several of them row-level, over a large CSV-fed offline dataset is where responsiveness starts to degrade. When that point is reached, move the heaviest calculations upstream, into the WebI query or a variable, or into the CSV preparation, so the Data Analyzer receives a ready-made measure.

8.3. DATA VOLUME: TWO FEEDING MODES

The volume ceiling depends on how the Data Analyzer is fed. The two paths have very different limits, and confusing them is the main cause of unrealistic sizing expectations.

Fed from a Web Intelligence report

In this mode the ceiling is set by SAP, not by the component. The number of rows a custom element can receive from a Web Intelligence block is capped at 50,000. From BI 4.3 SP05 onwards, that ceiling is raised to 250,000.

Beyond the cap, the component does not receive the additional rows. The design consequence is straightforward: a WebI-fed Data Analyzer is intended for analytical perimeters, not for detail-level extracts. Filter and aggregate in the query so that the block stays within the cap.



Check the ceiling that applies to your platform version before sizing a report: 50,000 rows up to BI 4.3 SP04, 250,000 from BI 4.3 SP05.

Fed from an embedded CSV file

Offline deliverables can embed a CSV file instead of a WebI block. The SAP cap no longer applies, and the ceiling becomes the memory available to the browser on the end user's machine.

Two consequences follow:

- Memory consumption is broadly proportional to the size of the dataset, so the practical ceiling varies from one workstation to another. The same deliverable can open on a 32 GB machine and fail on an 8 GB one.
- Row count is not the only driver. Cardinality — the number of distinct values in each field — weighs heavily, as does the number of columns. A narrow file of several hundred thousand rows can be lighter than a wide file of a few tens of thousands.

As an order of magnitude, one million rows is a reasonable working ceiling for a CSV-fed offline deliverable. It is a design guideline rather than an enforced threshold: below it, behaviour is generally comfortable; above it, testing on representative hardware becomes necessary.

A browser out-of-memory error on opening a CSV-fed deliverable is almost always a volume issue, not a defect. The first diagnostic step is to reduce the file and retry on the same machine.

Recommendations in both modes

- Filter at the source. Restrict the perimeter to what is actually analysed rather than loading the full history.
- Keep only the columns that are used. Every unnecessary field adds weight for no analytical value.
- Aggregate upstream when detail is not needed. A pre-aggregated dataset is dramatically lighter than a detail-level extract.
- Use the flat, non-aggregated view sparingly on large volumes: it is the heaviest display mode, since every row is rendered rather than summarised.

8.4. BROWSER REQUIREMENTS

The component runs in any browser supporting ECMAScript 2017 — current versions of Chrome, Edge, Firefox and Safari. Internet Explorer 11 is not supported. Using an up-to-date browser version is recommended, both for performance and for memory management on large datasets.

9. Offline behaviour

The Data Analyzer behaves identically online and offline. A dashboard exported for offline use carries the same component, the same calculation engine and the same interface.

In practical terms:

- Calculated values embedded in the report are preserved and recalculated normally.
- Users can create, edit and remove their own calculated values in the offline deliverable, exactly as they would online.
- The full formula syntax is available — no function or operator is degraded.



Need4Viz Data Analyzer – formulas guide

- The limitations described in sections 8.1 and 8.2 apply unchanged, since they stem from the client-side nature of the component, which is the same in both modes.

The difference lies in the data, not in the calculations. Offline, the dataset is fixed at export time and cannot be refreshed against the source; calculations run on that snapshot. An offline deliverable can also be fed by an embedded CSV file rather than a WebI block, which lifts the SAP row cap and raises the practical ceiling considerably — see section 8.3.



10. Annex – Troubleshooting common errors

Symptom	Likely cause and fix
Wrong formula format message	Indicator not wrapped in an aggregation function, or missing double quotes around its name.
Empty column	if condition without an else branch: cells failing the condition stay empty. Add a default value.
Result multiplied by one hundred	The percent aggregation combined with a percentage display format. Divide the formula by 100.
Implausible result on a multiplication	Aggregated calculation used instead of individual calculation. Tick the corresponding option.
Inconsistent grand total	The total applies the formula to the totals. Expected behaviour for a ratio.
Calculated value lost after a report change	Source indicator renamed or removed in the Web Intelligence document. Recreate or correct the formula.
Blank cells on a difference or running total	The aggregation depends on neighbouring hierarchy members. Check the sorting and the filters applied to the dimension.