# VALIDATION AND TRANSFORMATION LANGUAGE – VTL 2.0 Quick Reference Guide "At a Glance" (*Version June 2018)*

## General Syntax

| | |
|---|---|
| **Variable**: symbol **:=** assigns any **Expression** to a variable (*Variables could not be reassigned)* | **A := 1**<br>*assigns the number 1 to the variable A* |
| **Expression:** any Mathematical or Boolean formula which can be evaluate for a test, assign to a variable or be parameter of a function. | **log(10, 10000)** return **4** (decimal logarithm)<br>**2*3 + power(2, 5) – 1** return **37** (2x3 + $2^5$ - 1)<br>**A:=1; B:=2; C:=3; (C=A+B)** return **True** |
| **Conditional Expression** used for Variable assignment<br>***if*** *Test* ***then*** *expression_1* ***else*** *expression_2* | Maximum **:= If** A > B **Then** A **Else** B<br><br>*Set the variable Maximum to the maximum between A and B* |
| **Comment: //** is used to keep introduce a line comment<br>/* … */ is used to encapsulate a multiple line comment. | *//This is a one line comment*<br>*/*This is*<br>*a multiple lines paragraph comment */* |

## Validation check

| | |
|---|---|
| **Define a Hierarchical Ruleset (Vertical)**: define a set of formulas between codes of a codelist and interacting between different records of a dataset. | **define hierarchical ruleset** *Sample* ( **valuedomain rule** *ref_area*) **is**<br>/* R1 */ EFTA = CH+IS+LI+NO;<br>/* R2 */ BL = BE+LU;<br>/* R3 */ EU27 = EU28 - UK;<br>**end hierarchical ruleset** |
| **Define a DataPoint Ruleset (Horizontal)**: define a set of formula to be applied on a Datapoint and interacting between different components on each record of a dataset. | **define datapoint ruleset** *Sample* (<br>variable *table as T*, *obs_value as OV, obs_status as OS*) **is**<br>**when** *T* = "T01" **then isnull**(*OS*)<br>**or** (*OS* = "P" **and not isnull**(OV))<br>**or** (*OS* = "N" **and** (**isnull**(*OV*)<br>**or** (*OV* = 0))<br>**errorcode**("Invalid Value-Flag combination") **errorlevel**("Error")<br>**end datapoint ruleset** |
| **Apply a Ruleset**: on a dataset (the dataset must have the components on which the ruleset is defined) | **Errors := check_hierarchy(** *Dataset***,** *Sample* **)** // for hierarchical ruleset<br>**Errors := check_datapoint(** *Dataset***,** *Sample* **)** // for datapoint ruleset |
| **Apply on a Formula**: verify any logical and mathematical formula, return the datapoints which are not respecting the rule. | **CurYear**:= Intra_EU_Travellers;<br>**PrevYear**:= **lag** (Intra_EU_Travellers, 1 **over order by** *time_period* ));<br><br>**Errors**:= **check**(abs(**PrevYear** - **CurYear**) / **PrevYear** < 0.03); |
| **Manage Validation Metadata**: Validation metadata can be added to the check function | **Errors**:= *check*(abs(**PrevYear** - **CurYear**) / **PrevYear** < 0.03<br>***errorcode***("Growth Rate is greater than +/-3%")<br>***errorlevel***("Warning") ); |

## Basic scalar types

| | |
|---|---|
| **duration** | Length of a time interval expressed with any precision |
| **number** | Rational number of any magnitude and precision (subtype – Integer) |
| **boolean** | Result of a test or constant **true** or **false** |
| **time** | Time intervals of any duration (subtypes – date, time_period) |
| **string** | Sequence of alphanumeric characters of any length |

## Standard Functions

| | | |
|---|---|---|
| Numeric | Operators: "+" "-" "*" "/" "(" ")" | **(**2 ***** 5**) / (**5 **–** 3**) +** 2 return 7 |
| | Modulo: **mod**( *dividend*, *divisor* ) | **mod** ( 5, 2 ) return 1 |
| | Rounding: **round** or **trunc** with optional second parameter (*number of decimals*) | *Pi*:=3.14159;<br>**round**( *Pi*, 3) return 3.142<br>**trunc** ( *Pi*, 3) return 3.141 |
| | Absolute Value: **abs**() function | **abs**(-3.5) return 3.5 |
| | Power: **power**(*value, exponent*) | **power**(2, 10) return 1024<br>**power**(8, 1/3) return 2 (cubic root) |
| | Square root: **sqrt**(*value*) | **sqrt**(625) return 25 |
| | Logarithm and Exponential: **ln**(*value*), **log**(*base*, *value*) and **exp**(*value*) | **ln**(2,71828..) return 1<br>**exp**(0) return 1<br>**log**(2, 1024) return 10 |
| Boolean | Boolean operators: **and**, **or**, **not, xor** (exclusive or) | *true* **and** *false* return *false*<br>**not** *true* return *false*<br>*true* **or** *false* return *true* |
| | Comparison operators: "**=**" "**<>**" "**>**" "**<**" "**<=**" "**>=**" | 1**<>**1 return *false*<br>*2.5* **>=** *5/2* return *true* |
| | Test element in a list: **in** | 2 **in [**1**,** 2**,** 3**]** return *true*<br>"FR" **in [**"BE"**,**"LU"**]** return *false* |
| | Test if a value is empty: **isnull**(*value*) | **isnull**(*obs_value*) |
| String Functions | Length of a String | **length**("J59_J60") return 7 |
| | Uppercase/lowercase | **upper**("Euro") return "EURO"<br>**lower**("Euro") return "euro" |
| | Trim leading and trailing space: **trim**, **ltrim**, **rtrim** | **trim**(" Good Bye ") = "Good Bye" |
| | Concatenation: **||** operator | "Good" **||** " " **||** "Bye" = "Good Bye" |
| | Sub-String: ***substr****(str, start, length)* | ***substr****("12345", 3, 2) = "34"* |
| | Search: ***instr****(where, what)*<br>*(return the position of the first character of the first occurrence of the string what in the string where)* | **instr**("ABCDE", "CD") = 3<br>**instr**("ABCDE", "CE") = 0<br>(0 is returned when not found) |
| | String-Replace: ***replace****(str, old, new)*<br>*(replace all occurrences of the* ***old*** *by* ***new*** *in the string* ***str****)* | **replace**("123456", "34", "++++") = "12++++56" |

## Sample Dataset Structure: *Intra_EU_Travellers*

Components: **(1) Identifier** **(2) Measure** **(3) Attribute**

| *table* | *freq* | *time_period* | *reporting* | *partner* | *direction* | *age* | *adjust* | *obs_value* | *obs_status* |
|---|---|---|---|---|---|---|---|---|---|
| T01 | A | 2015 | IT | FR | IN | TOT | N | 10 000 | |
| T01 | A | 2015 | PT | ES | OUT | Y18 | N | 8 000 | |
| T02 | Q | 2015-Q1 | FR | DE | IN | Y40 | S | 11 500 | P |
| T02 | Q | 2016-Q2 | DE | FR | OUT | Y40 | N | 12 000 | C |

**(1) Identifiers** are the **KEY** information, the combination of their values must be unique.

**(2) Measures** are Observed Values (one measure only is supported for Dataset Boolean Operation)

**(3) Attributes** are data point metadata, for instance Flag, Footnotes

## Data Transformation

| | |
|---|---|
| **Dataset**: used to refer to the whole dataset object, structure and contents that can be used in formulas. Component of a dataset can be referred using the # symbol (sharp) | CurrentYear := **Intra_EU_Travellers**<br>*check (* **Intra_EU_Travellers**, ***MyRulesSet*** *)*<br>*check (* Intra_EU_Travellers#*reporting* <> Intra_EU_Travellers#*partner* *)* |
| **Dataset Filtering:** Obtain a subset of a dataset based on a given formula keeping the original dataset structure (all components) | TableT01 := Intra_EU_Travellers **[ filter** *table* = "T01" **]**<br>NonAdjustPos := Intra_EU_Travellers **[ filter** *obs_value* > 0 and *adjust* = "N" **]** |
| **Dataset Subset:** Obtain a subset of a dataset based on **identifier constant(s).** This syntax will remove also the component(s) from the resulting dataset. | TableT01 := Intra_EU_Travellers **[ sub** *table* = "T01" **]**<br>NonAdjust_IT := Intra_EU_Travellers **[ sub** *reporting* = "IT", *adjust* = "N" **]** |
| **Dataset Calculation:** add or replace a **measure** or **attribute** using a formula | Value1000Trv := Intra_EU_Travellers /* replace *obs_value* measure */<br>**[ calc** *obs_value* := round(obs_value / 1000, 0) **]**<br>ConfidentialisedValue := Intra_EU_Travellers /* add *conf_value* measure */<br>**[ calc** *conf_value* := **if** obs_status="C" **then** 0 **else** floor(*obs_value*) **]** |
| **Dataset Offset: Analytics** functions enable any switch of obs_value within datapoints (see reference guide for more details). For each sliding window (**partition by**) and for each Measure Component, the operator calculates the function's value according to the specified order (**order by**). Functions are **lag**/**lead** (value swap), **first_value**/**last_value**, **ntil**/**rank**/… (position within a sliding window) and **aggregate** functions | PreviousYear:= **lag** (Intra_EU_Travellers, 1 **over** ( **order by** *time_period* ));<br><br>//Get the maximum *obs_value* for each "country to country" itinerary<br>Top_Travelers:= **last_value** (Intra_EU_Travellers [filter table = "T01"]<br>**over** (**partition by** *reporting*, *partner*, *direction*<br>**order by** *obs_value* ));<br><br>*Remark: without **partition by** clause, all identifiers but **order by** ones are in the sliding window key* |
| **Dataset complete time series**: append to the dataset the missing time series. | *Extended*:= **fill_time_series** (Intra_EU_Travellers) ;<br>*Extended* contains original data series and the missing datapoints |
| **Dataset Time Aggregates**: using one of the **aggregate** function to convert from a smaller to a larger duration**.** | Calc_Year := **sum**(Intra_EU_Travellers)**time_agg** `("A", "Q");` |
| **Dataset Aggregate**: aggregates functions like **Sum**, **Avg**, **Max**, **Min**,… can be used to calculate any aggregates based on a specified aggregation identifiers key (**group by**) | TravellersAverageByRoad := **avg (**Intra_EU_Travellers<br>**group by** [reporting, partner, direction])<br><br>BiggestPartner := **max (**Intra_EU_Travellers **group by** partner) |
| **Dataset Operator and formulas**: as long as dataset have the same identifiers (structure), all classical operators can be applied on the measures. The **exists_in** operator test two sets intersection | GrowthRate := Abs(PreviousYear – CurrentYear) / PreviousYear<br>check ( PassengerTotal > 2 * PassengerAgeUnder18 )<br>check ( **exists_in** ( AllCombinations, MyData ) **errorcode** ("missing series")<br>**errorlevel** ("Error")) |
| **Clauses**: drop, rename, re-define a measure or attribute, add a new identifier, measure or attribute | Intra_EU_Travellers **[ rename** *reporting* **to** *reporter* **]**<br>Intra_EU_Travellers **[ keep** obs_value **]**)<br>Intra_EU_Travellers **[ drop** obs_status **]**<br>Intra_EU_Travellers **[ calc identifier** rule_id := "ErrB" **]** |
| **Dataset set operations**: apply set operations on two datasets (**union**, **intersect**, **symdiff**, **setdiff)** | **union** ( Dataset1, Dataset2) |