Appendix 1 Expression Editor

A1.1 Overview

A1.2 Function Category

A1.3 Transformation

A1.4 Math/Trig

A1.5 Text

A1.6 Date/Time

A1.7 Variable Statistics

A1.8 Information

A1.9 Record Function

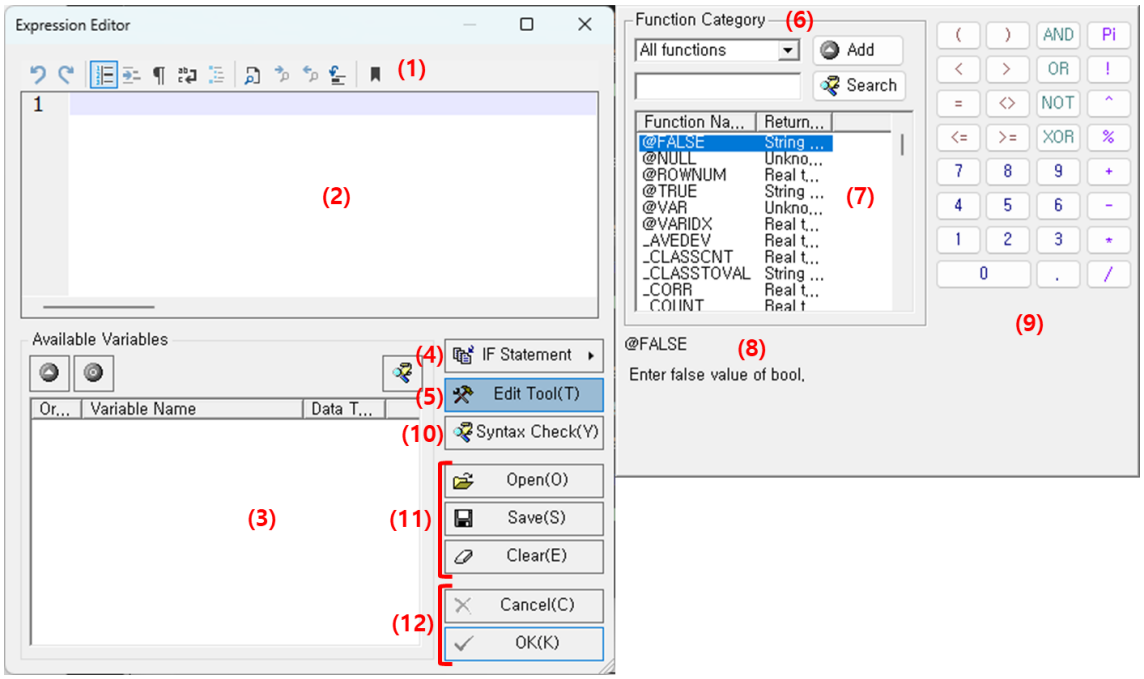
A1.10 Macro

A1.11 Other Functions

A1.1 Overview

The **Expression Editor** is used across various nodes such as *Derived Variable*, *Filter*, *Select*, and *Fill*. It is utilized to create new calculated values or to edit expressions for conditions. Users can edit expressions entirely through mouse operations, or, once they become familiar with the interface, they can type directly into the expression input window for faster editing.

User Interface

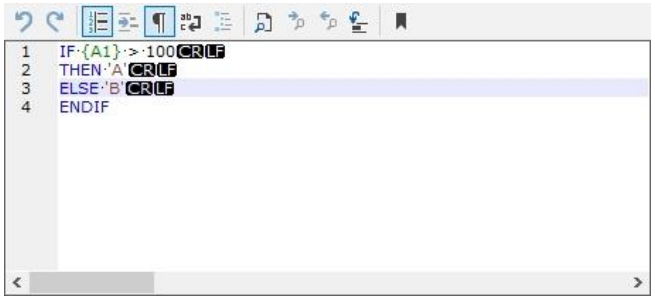
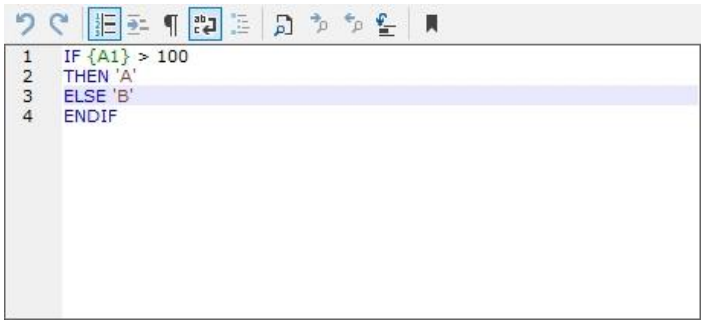


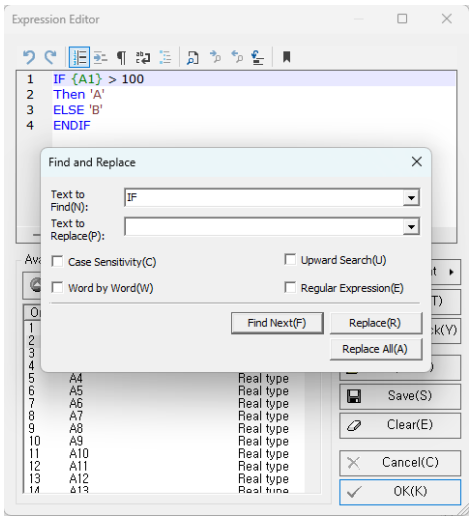
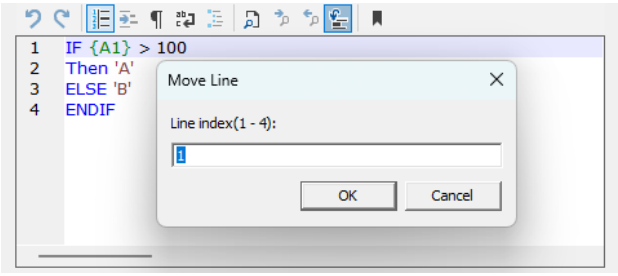
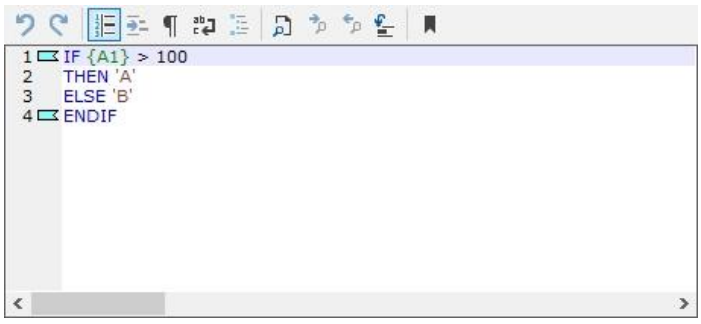
No.	Description
(1)	**Toolbar** Provides expression editing features such as *Undo/Redo*, *Find*, and *Bookmark* in a toolbar format for convenient access.
(2)	**Expression Editing Window** This is where expressions are edited. Users can enter expressions manually or use tools (4) ~ (8) to edit them. By using these tools, expressions used in ECMiner™ can be easily edited through mouse operations.
(3)	**Available Variables** Displays the list of existing variables available in the current node. Select the variable

	you wish to use and double-click it to add it to the editing window. When added, it appears enclosed in curly braces (e.g., {DATE}), indicating that it represents a variable.
(4)	**IF Statement** Clicking the **IF Statement** button to insert the *IF ~ THEN ~ ELSE ~ ENDIF* statement into the expression.
(5)	**Edit Tool** Click the **Edit Tool** button to open the function category panel on the right.
(6)	**Function Category** Allows users to choose from the provided function categories, including **Transformation**, **Math/Trigonometric**, **Text**, **Date/Time**, **Variable Statistics**, and **Information** functions. When the category is changed, the list of corresponding functions appears in section (4).
(7)	**Function List** Displays functions belonging to the selected function category. To insert a function, double-click or press the **Add** button to insert it into the Expression Editing Window. The usage and a brief description of the selected function appear in section (8).
(8)	**Function Description** Displays a brief description and usage information of the currently selected function.
(9)	**Numeric and Symbol Buttons** Provides frequently used numbers and symbols as buttons for quick editing using mouse operations. Clicking a button inserts the corresponding number or symbol into the editing window.
(10)	**Syntax Check** Checks whether the current expression syntax is valid. If an error is detected, an error message is displayed, allowing users to identify and correct the mistake.
(11)	**Saving and Loading Expressions** You can save the current expression or open a previously saved one for reuse. Clicking the Clear button deletes all expressions currently entered. The file extension for the expressions is **.eaf*
(12)	**Confirm or Cancel** Applies or cancels the current expression input. Clicking the **OK** button enters the expression after performing syntax validation. If an invalid expression is entered, the program may not function properly.

Toolbar

The toolbar provides a variety of useful features for expression editing. The icons and their corresponding functions are as follows:

ICON	Toolbar	Description
	Undo (Crtl+Z)	Cancel the last operation.
	Redo (Ctrl+Y)	Redo the previously undone operation.
	Toggle Line Numbers	Displays or hides line numbers.
	Toggle Column Indicator	Displays or hides indentation guides
¶	Toggle Unprint Characters	Displays or hides spaces, tabs, and line break symbol. 
	Toggle Wordwrap	Enables or disables automatic line wrapping. When word wrap is enabled, the horizontal scrollbar is hidden. 
	Toggle Folding	Enables or disables code folding.
	Find (Ctrl+F)	Searches for or replaces specified text.

		
	Find Next (F3)	Finds the next specified text.
	Find Previous (Shift+F3)	Finds the previous specified text.
	Go to line (Ctrl+G)	Moves the cursor to the specified line number. 
	Bookmark Setup (Ctrl+F2)	Enables or disables bookmarks on the current line. 

Useful Features

Additional tools that enhance the efficiency of expression editing.

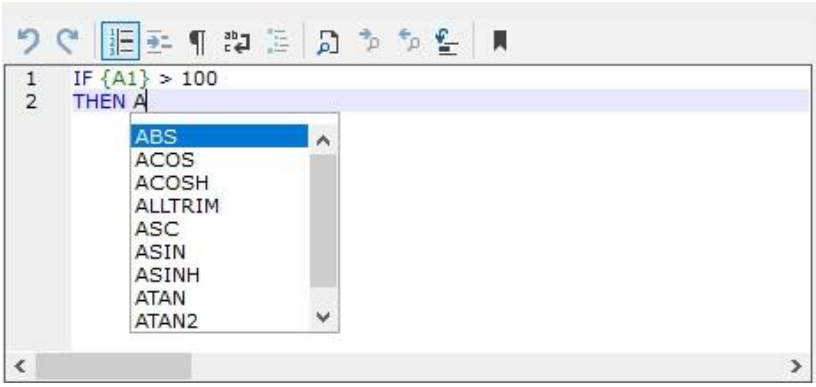
Auto-Completion

When entering expressions, the Expression Editor automatically completes text such as functions, keywords, macros, and data field names after typing their initial characters (or

prefixes). This eliminates the need to type long words in full and helps prevent potential typing errors.

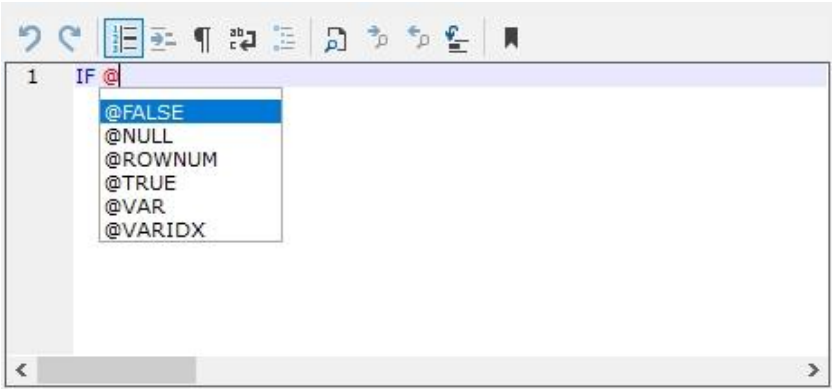
- Functions

  When you type the first letter or part of a function or keyword used in expression editing, a list of matching functions or keywords beginning with that text is displayed.



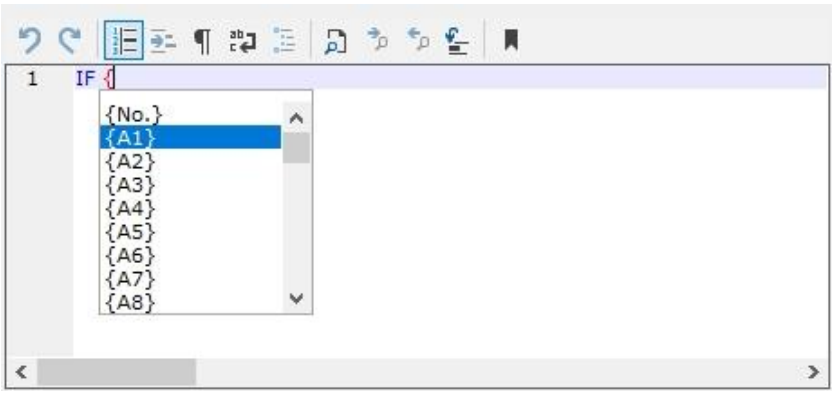
- Macros

  When you type the first character “@”, a list of available macros used in expression editing appears.



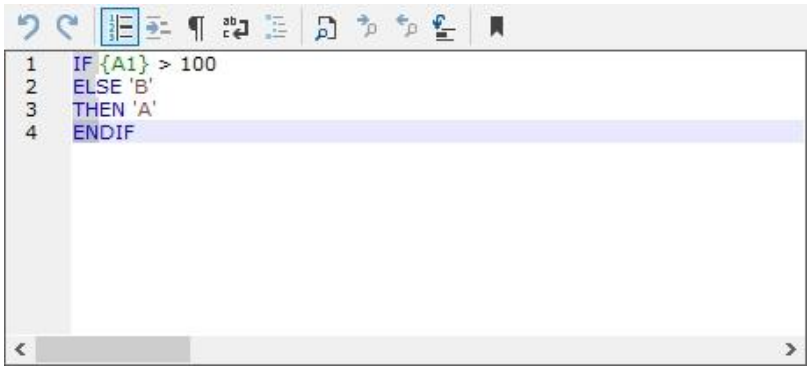
- Data Field

  When you type the first character “{”, a list of available data fields is displayed.



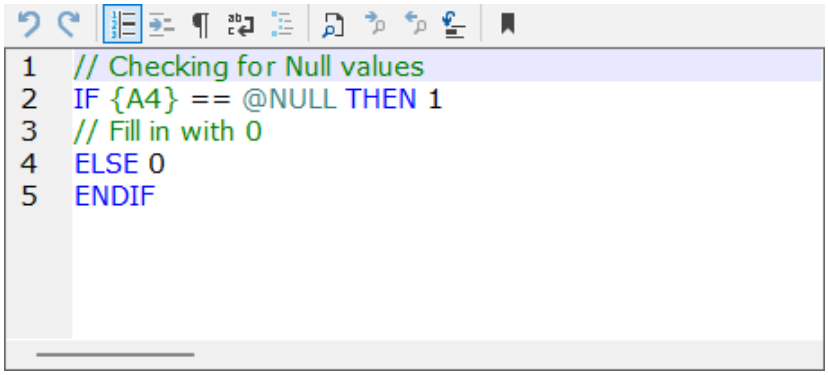
Column Editing

You can switch to column mode by holding **Alt** and dragging the mouse, or by pressing **Alt + Shift + Arrow keys**.



Line Comments

Add the line comment symbol (//) at the beginning of a line to comment out that line.



Color Coding

String→ "123" or '123'- Dark Red

Date Type → #2004-01-01# - Dark Red

Numeric Type → 123 – Black

Functions / IF Statements → ABS() – Blue

Variables → {DATE} – Green

Operators → +, -, *, / – Dark Green

Macros → @ROWNUM – Dark Green

Comments → // Extra Comments – Green

If there is an error in an expression or function, the part containing the error is highlighted in **red** in the Expression Editor window.

A1.2 Function Category

ECMiner™ offers a variety of built-in functions that can be used within the Expression Editor.

Function Category

Category	Description
Transformation	Convert the type of a value — for example, from date to string or from string to numeric.
Math/Trig	Include trigonometric and mathematical functions such as *sin*, *cos*, etc.
Text	Functions used to process string-type data.
Date/Time	Extract values such as year, date, or time from date-type data.
Variable Statistics	Used to obtain basic statistical quantities depending on the type (continuous or discrete) of the currently used variable.
Information	Return information about the state or type of an input value.
Record Function	Perform statistical processing at the record level.
Macro	Provides predefined macros (prefixed with “@”) that can be used to reference system information, iteration counts, or record-level values during expression calculation.

Other Functions	Additional functions implemented based on specific requirements.

Input Formats

In ECMiner™, values must be netered according to the following rules corresponding to each date type.

- **String Data**
  To enter a string directly, enclose it in quotation marks.
  Example: "desired text" or 'desired text'
- **Date/Time Data**
  To enter a date or date/time value, enclose it in hash symbols (#).
  Example: #2004-01-01 13:10:30# is recognized as *January 1, 2004, 1:10:30 PM.*
  If the time is not needed, simply enter #2004-01-01# to represent only the date.
- **Numeric Data**
  Enter numbers without any symbols.
  Examples: 1234.5, 1234, 1.234e3
- **Variable Values**
  Variable values are entered in curly braces as {variable_name}.
  The use of braces {} prevents conflicts with reserved words in ECMiner™ and ensures compatibility with all variable name formats.

Output Formats

The result formats shown in the examples below represent how output data types are displayed:

- When the result is of **string type**, it is displayed as "Result Value".
- When the result is of **date type**, it is displayed as #Result Value#.

(In the actual interface, quotation marks " and hash symbols # are not displayed — only the numeric or text values are shown.)

A1.3 Transformation

Function Name	Description	Return Value	Example

ASC(text)	Returns the ASCII code value of the first character in the input string.	Real type	ASC("ABC") → 65
CHR(num)	Returns the character corresponding to the given ASCII code.	String type	CHR(65) → "A"
DATETOSTR(date)	Converts a date-type value to a string.	String type	DATETOSTR(#2004-1-1#) → "20040101"
FLOAT(num)	Converts an integer to a real number.	Real type	FLOAT(3.4) → 3
IIF(cond, expr1, expr2)	Evaluates and returns expr1 if the condition is true; otherwise, returns expr2.	Unknown	IIF(1>0, 1+2, 1+3) → 3
INT(num)	Converts a real number to an integer by truncating the decimal part.	Integer type	INT(2.371) → 2
STR(num[, num1[, num2]])	Converts a number to a string. num1 specifies the maximum length, and num2 specifies the number of decimal places.	String type	STR(123.345, 6, 2) → "123.35"
STRTODATE(text, fmt)	Converts a string to a date-type value according to the specified format (fmt). Format symbols: y (year), m (month), d (day), H (hour), M (minute), S (second).	Date type	STRTODATE("2004-01-01 11:50", "yyyy-mm-dd HH:MM") → #2004-01-01 11:50:00#
VAL(text)	Converts a string to a numeric value.	Real type	VAL("123") → 123

A1.4 Math/Trig

All of the Math/Trig functions return Real type values.

Function Name	Description	Example

ABS(num)	Returns the absolute value.	ABS(-123) → 123
ACOS(num)	Returns arccosine in radians (0 to π).	ACOS(-0.5) → 2.09440 (2π/3)
ACOSH(num)	Returns inverse hyperbolic cosine.	ACOSH(2) → 1.31696
ASIN(num)	Returns arcsine in radians (−π/2 to π/2).	ASIN(-0.5) → -0.52360 (−π/6)
ASINH(num)	Returns inverse hyperbolic sine.	ASINH(2) → 1.44364
ATAN(num)	Returns arctangent in radians (−π/2 to π/2).	ATAN(1) → 0.78540 (π/4)
ATAN2(num1, num2)	Arctangent of (y, x), radians in (−π, π].	ATAN2(1, 1) → 0.78540 (π/4)
ATANH(num1)	Returns inverse hyperbolic tangent.	ATANH(0.2) → 0.20273
BETACDF(num1, num2, num3)	Beta CDF with shape (num1, num2) at num3.	BETACDF(2, 2, 1) → 1.0
BETAINV(num1, num2, num3)	Inverse Beta CDF with shape (num1, num2) at prob num3.	BETAINV(2, 2, 1) → 1.0
BETAMEAN(num1, num2)	Mean of Beta distribution (num1, num2).	BETAMEAN(2, 2) → 0.5
BETAPDF(num1, num2, num3)	Beta PDF with shape (num1, num2) at num3.	BETAPDF(2, 2, 1) → 0
BETAVARIANCE(num1, num2)	Variance of Beta distribution.	BETAVARIANCE(2, 2) → 0.05
BINOCDF(n, p, k)	Binomial CDF with (n, p) at k.	BINOCDF(4, 0.3, 3) → 0.9919

BINOINV(n, p, q)	Inverse Binomial CDF (n, p) at prob q.	BINOINV(4, 0.3, 0.9919) → 3
BINOMEAN(n, p)	Mean of Binomial (n, p).	BINOMEAN(10, 0.5) → 5
BINOPDF(n, p, k)	Binomial PDF with (n, p) at k.	BINOPDF(4, 0.3, 3) → 0.07560
BINOVARIANCE(n, p)	Variance of Binomial.	BINOVARIANCE(10, 0.5) → 2.5
CEILING(num)	Smallest integer ≥ num.	CEILING(2.1) → 3
CEILING2(num1, num2)	Rounds num1 up to a multiple of num2.	CEILING2(6.4) → 8
CHICDF(v, x)	Chi-square CDF (df=v) at x.	CHICDF(1, 0.3) → 0.41612
CHIINV(v, q)	Inverse Chi-square CDF (df=v) at prob q.	CHIINV(1, 0.41612) → 0.3
CHIMEAN(v)	Mean of Chi-square with df v.	CHIMEAN(10) → 10
CHIPDF(v, x)	Chi-square PDF (df=v) at x.	CHIPDF(1, 0.3) → 0.62691
CHIVARIANCE(v)	Variance of Chi-square with df v.	CHIVARIANCE(10) → 20
COMB(n, k)	Number of combinations “n choose k”.	COMB(4, 2) → 6
COS(rad)	Returns cosine.	COS(1.047) → 0.50017
COSH(num)	Returns hyperbolic cosine.	COSH(4) → 27.30823
DEGREE(rad)	Converts radians to degrees.	DEGREE(3.14159265) → 180
DIGAMMA(num)	Digamma function (derivative of lgamma).	DIGAMMA(1) → -0.57722
ERF(num)	Error function.	ERF(1) → 0.8427
ERFC(num)	Complementary error function.	ERFC(1) → 0.15776

EVEN(num)	Rounds up to the nearest even integer.	EVEN(1.5) → 2
EVCDF(mu, sigma, x)	Extreme value (Gumbel) CDF at x.	EVCDF(0, 1, 0.4) → 0.77504
EVINV(mu, sigma, q)	Inverse extreme value CDF at prob q.	EVINV(0, 1, 0.77504) → 0.40001
EVMEAN(mu, sigma)	Mean of extreme value distribution.	EVMEAN(0, 1) → 0.57722
EVPDF(mu, sigma, x)	Extreme value PDF at x.	EVPDF(0, 1, 0.4) → 0.3356
EVVARIANCE(mu, sigma)	Variance of extreme value distribution.	EVVARIANCE(0, 1) → 1.64493
EXP(num)	Computes e^num.	EXP(1) → 2.71828
EXPCDF(λ, x)	Exponential CDF (rate λ) at x.	EXPCDF(1, 0.4) → 0.32968
EXPINV(λ, q)	Inverse Exponential CDF at prob q.	EXPINV(1, 0.32968) → 0.4
EXPMEAN(λ)	Mean of Exponential (1/λ or as defined).	EXPMEAN(10) → 10
EXPPDF(λ, x)	Exponential PDF at x.	EXPPDF(1, 0.4) → 0.67032
EXPVARIANCE(λ)	Variance of Exponential.	EXPVARIANCE(10) → 100
FACT(num)	Factorial.	FACT(4) → 24
FISHER(z)	Fisher transform.	FISHER(0.5) → 0.54931
FISHERINV(y)	Inverse Fisher transform.	FISHERINV(0.54931) → 0.5
FLOOR(num)	Greatest integer ≤ num.	FLOOR(2.1) → 2

FCDF(v1, v2, x)	F-distribution CDF at x.	FCDF(2, 4, 3) → 0.84
FINV(v1, v2, q)	Inverse F CDF at prob q.	FINV(2, 4, 0.84) → 3
FMEAN(v1, v2)	Mean of F distribution.	FMEAN(5, 5) → 1.66667
FPDF(v1, v2, x)	F-distribution PDF at x.	FPDF(2, 4, 3) → 0.064
FVARIANCE(v1, v2)	Variance of F distribution.	FVARIANCE(5, 5) → 8.88889
GAMCDF(k, θ, x)	Gamma CDF (shape k, scale θ) at x.	GAMCDF(1, 2, 1) → 0.39347
GAMINV(k, θ, q)	Inverse Gamma CDF at prob q.	GAMINV(1, 2, 0.39347) → 1
GAMMEAN(k, θ)	Mean of Gamma (kθ).	GAMMEAN(3, 3) → 9
GAMPDF(k, θ, x)	Gamma PDF at x.	GAMPDF(1, 2, 1) → 0.30327
GAMVARIANCE(k, θ)	Variance of Gamma (kθ²).	GAMVARIANCE(3, 3) → 27
GEOCDF(p, k)	Geometric CDF at k.	GEOCDF(0.3, 2) → 0.657
GEOINV(p, q)	Inverse Geometric CDF at prob q.	GEOINV(0.3, 0.657) → 3
GEOMEAN(p)	Mean of Geometric.	GEOMEAN(0.25) → 3
GEOPDF(p, k)	Geometric PDF at k.	GEOPDF(0.3, 2) → 0.147
GEOVARIANCE(p)	Variance of Geometric.	GEOVARIANCE(0.25) → 12
GEVCDF(mu, sigma, k, x)	Generalized extreme value CDF at x.	GEVCDF(0,1,0,2) → 0.87342

GEVINV(mu, sigma, k, q)	Inverse GEV CDF at prob q.	GEVINV(0,1,0,0.87342) → 1.99997
GEVMEAN(mu, sigma, k)	Mean of GEV.	GEVMEAN(1,3,0.1) → 3.05886
GEVPDF(mu, sigma, k, x)	GEV PDF at x.	GEVPDF(0,1,0,2) → 0.1182
GEVVARIANCE(mu, sigma, k)	Variance of GEV.	GEVVARIANCE(1,3,0.1) → 20.03617
GPCDF(K, sigma, theta, x)	Generalized Pareto CDF at x.	GPCDF(0,1,0,2) → 0.86466
GPINV(K, sigma, theta, q)	Inverse GP CDF at prob q.	GPINV(0,1,0,0.86466) → 1.99997
GPMEAN(K, sigma, theta)	Mean of GP.	GPMEAN(0.1,3,1) → 4.33333
GPPDF(K, sigma, theta, x)	GP PDF at x.	GPPDF(0,1,0,2) → 0.13534
GPVARIANCE(K, sigma, theta)	Variance of GP.	GPVARIANCE(0.1,3,1) → 13.88889
HYGECDF(m, k, n, x)	Hypergeometric CDF at x.	HYGECDF(100,20,10,2) → 0.68122
HYGEINV(m, k, n, q)	Inverse Hypergeometric CDF at prob q.	HYGEINV(100,20,10,0.68122) → 2

HYGEMEAN(m, k, n)	Mean of Hypergeometric.	HYGEMEAN(10,5,4) → 2
HYGEPDF(m, k, n, x)	Hypergeometric PDF at x.	HYGEPDF(100,20,10,2) → 0.31817
HYGEVARIANCE(m, k, n)	Variance of Hypergeometric.	HYGEVARIANCE(10,5,4) → 0.66667
LGAMMA(num)	Natural log of Gamma function.	LGAMMA(3) → 0.69315
LN(num)	Natural logarithm.	LN(86) → 4.45435
LOG(num, base)	Logarithm with given base.	LOG(7, 2) → 2.80735
LOG10(num)	Base-10 logarithm.	LOG10(86) → 1.93450
LOGBETA(num1, num2)	Natural log of Beta function.	LOGBETA(5, 3) → -4.65396
LOGNCDF(mu, sigma, x)	Lognormal CDF at x.	LOGNCDF(0,1,2) → 0.75589
LOGNINV(mu, sigma, q)	Inverse lognormal CDF at prob q.	LOGNINV(0,1,0.75589) → 1.99999
LOGNMEAN(mu, sigma)	Mean of lognormal.	LOGNMEAN(-2, 2) → 1
LOGNPDF(mu, sigma, x)	Lognormal PDF at x.	LOGNPDF(0,1,2) → 0.15687
LOGNVARIANCE(mu, sigma)	Variance of lognormal.	LOGNVARIANCE(-2, 2) → 53.59815
MOD(num1, num2)	Remainder of num1 / num2.	MOD(5, 2) → 1

NBINCDF(r, p, k)	Negative binomial CDF at k.	NBINCDF(3, 0.5, 2) → 0.5
NBININV(r, p, q)	Inverse negative binomial CDF at prob q.	NBININV(3, 0.5, 0.5) → 2
NBINMEAN(r, p)	Mean of negative binomial.	NBINMEAN(10, 0.5) → 10
NBINPDF(r, p, k)	Negative binomial PDF at k.	NBINPDF(3, 0.5, 2) → 0.1875
NBINVARIANCE(r, p)	Variance of negative binomial.	NBINVARIANCE(10, 0.5) → 20
NCFCDF(v1, v2, δ, x)	Noncentral F CDF at x.	NCFCDF(5,20,10,2) → 0.2719
NCFINV(v1, v2, δ, q)	Inverse noncentral F CDF at prob q.	NCFINV(5,20,10,0.2719) → 2
NCFMEAN(v1, v2, δ)	Mean of noncentral F.	NCFMEAN(10,100,4) → 1.42857
NCFPDF(v1, v2, δ, x)	Noncentral F PDF at x.	NCFPDF(5,20,10,2) → 0.26075
NCFVARIANCE(v1, v2, δ)	Variance of noncentral F.	NCFVARIANCE(10,100,4) → 0.42517
NCTCDF(v, δ, x)	Noncentral t CDF at x.	NCTCDF(10,1,2) → 0.80761
NCTINV(v, δ, q)	Inverse noncentral t CDF at prob q.	NCTINV(10,1,0.80761) → 1.99999
NCTMEAN(v, δ)	Mean of noncentral t.	NCTMEAN(4,1) → 1.25331
NCTPDF(v, δ, x)	Noncentral t PDF at x.	NCTPDF(10,1,2) → 0.00006
NCTVARIANCE(v, δ)	Variance of noncentral t.	NCTVARIANCE(4,1) → 2.4292

NCX2CDF(v, δ, x)	Noncentral χ² CDF at x.	NCX2CDF(4,2,2) → 0.13048
NCX2INV(v, δ, q)	Inverse noncentral χ² CDF at prob q.	NCX2INV(4,2,0.13048) → 2.00003
NCX2MEAN(v, δ)	Mean of noncentral χ².	NCX2MEAN(4,2) → 6
NCX2PDF(v, δ, x)	Noncentral χ² PDF at x.	NCX2PDF(4,2,2) → 0.10763
NCX2VARIANCE(v, δ)	Variance of noncentral χ².	NCX2VARIANCE(4,2) → 16
NORMCDF(mu, sigma, x)	Normal CDF at x.	NORMCDF(-1,1,2) → 0.99865
NORMDIST(x, mu, sigma)	Normal CDF at x (if x<mu, returns 1−CDF).	NORMDIST(6,5,1) → 0.84134; NORMDIST(4,5,1) → 0.84134
NORMINV(mu, sigma, q)	Inverse Normal CDF at prob q.	NORMINV(-1,1,0.99865) → 1.99998
NORMMEAN(mu, sigma)	Mean of Normal.	NORMMEAN(0,2) → 0
NORMPDF(mu, sigma, x)	Normal PDF at x.	NORMPDF(-1,1,2) → 0.00443
NORMVARIANCE(mu, sigma)	Variance of Normal.	NORMVARIANCE(0,2) → 4
ODD(num)	Rounds up to the nearest odd integer.	ODD(1.1) → 3
PERM(n, k)	Number of permutations.	PERM(4, 2) → 12
POISCDF(λ, k)	Poisson CDF at k.	POISCDF(4, 2) → 0.2381

POISINV(λ, q)	Inverse Poisson CDF at prob q.	POISINV(4, 0.2381) → 1
POISMEAN(λ)	Mean of Poisson.	POISMEAN(2) → 2
POISPDF(λ, k)	Poisson PDF at k.	POISPDF(4, 2) → 0.14653
POISVARIANCE(λ)	Variance of Poisson.	POISVARIANCE(2) → 2
POWER(num1, num2)	Returns num1 to the power of num2.	POWER(4, 0.5) → 2
PRODUCT(num1, num2, …)	Product of all arguments.	PRODUCT(1, 2) → 2
RADIANS(deg)	Converts degrees to radians.	RADIANS(180) → 3.14159 (π)
RAND()	Generates a random number between 0 and 1000.	RAND() → *(e.g.)* 534.12
RAYLCDF(σ, x)	Rayleigh CDF at x.	RAYLCDF(1, 2) → 0.86466
RAYLINV(σ, q)	Inverse Rayleigh CDF at prob q.	RAYLINV(1, 0.86466) → 1.99998
RAYLMEAN(σ)	Mean of Rayleigh.	RAYLMEAN(2) → 2.50663
RAYLPDF(σ, x)	Rayleigh PDF at x.	RAYLPDF(1, 2) → 0.27067
RAYLVARIANCE(σ)	Variance of Rayleigh.	RAYLVARIANCE(2) → 1.71681
ROUND(num, digit)	Rounds to digit decimal places.	ROUND(12.345, 2) → 12.35
ROUNDDOWN(num, digit)	Rounds down to digit places.	ROUNDDOWN(2.50663, 2) → 2.50

ROUNDUP(num, digit)	Rounds up to digit places.	ROUNDUP(2.50663, 2) → 2.51
SIN(rad)	Returns sine.	SIN(3.141592) → 0
SINH(num)	Returns hyperbolic sine.	SINH(1) → 1.17520
SQRT(num)	Positive square root.	SQRT(4) → 2
TAN(rad)	Returns tangent.	TAN(0.785) → 0.99920
TANH(num)	Returns hyperbolic tangent.	TANH(-2) → -0.96403
TCDF(v, x)	Student’s t CDF at x.	TCDF(1, 0.5) → 0.64758
TINV(v, q)	Inverse t CDF at prob q.	TINV(1, 0.64758) → 0.49999
TMEAN(v)	Mean of t distribution.	TMEAN(4) → 0
TOMAX(var, limit)	Returns var if var ≤ limit, else limit.	TOMAX(9,10) → 9; TOMAX(11,10) → 10
TPDF(v, x)	Student’s t PDF at x.	TPDF(1, 0.5) → 0.25465
TRIGAMMA(num1)	Trigamma function.	TRIGAMMA(1) → 1.64493
TRUNC(num1, digit)	Truncates to digit decimal places.	TRUNC(0.76, 1) → 0.7
TVARIANCE(v)	Variance of t distribution.	TVARIANCE(4) → 2
UNIDCDF(N, k)	Discrete uniform CDF at k (1…N).	UNIDCDF(50, 20) → 0.4
UNIDINV(N, q)	Inverse discrete uniform CDF at prob q.	UNIDINV(50, 0.4) → 20
UNIDMEAN(N)	Mean of discrete uniform.	UNIDMEAN(5) → 3
UNIDPDF(N, k)	Discrete uniform PDF at k.	UNIDPDF(50, 20) → 0.02

UNIDVARIANCE(N)	Variance of discrete uniform.	*(example not provided)*
UNIFCDF(a, b, x)	Continuous uniform CDF at x.	UNIFCDF(-1, 1, 0.7) → 0.85
UNIFINV(a, b, q)	Inverse continuous uniform CDF at prob q.	UNIFINV(-1, 1, 0.85) → 0.7
UNIFMEAN(a, b)	Mean of continuous uniform.	UNIFMEAN(1, 7) → 4
UNIFPDF(a, b, x)	Continuous uniform PDF at x.	UNIFPDF(-1, 1, 0.7) → 0.5
UNIFVARIANCE(a, b)	Variance of continuous uniform.	UNIFVARIANCE(1, 7) → 3
WBLCDF(k, λ, x)	Weibull CDF at x.	WBLCDF(0.15, 0.8, 0.5) → 0.53846
WBLINV(k, λ, q)	Inverse Weibull CDF at prob q.	WBLINV(0.15, 0.8, 0.53846) → 0.5
WBLMEAN(k, λ)	Mean of Weibull.	WBLMEAN(2, 0.5) → 4
WBLPDF(k, λ, x)	Weibull PDF at x.	WBLPDF(0.15, 0.8, 0.5) → 1.53846
WBLVARIANCE(k, λ)	Variance of Weibull.	WBLVARIANCE(2, 80) → 0.00100

A1.5 Text

Function Name	Description	Return Value	Example
ALLTRIM(text1[, text2])	Removes all occurrences of text2 from text1. If text2 is omitted, removes spaces.	String type	ALLTRIM("AABACD","A") → "BCD"
CHRTRAN(text1, text2, text3)	Replaces every occurrence of text2 in text1 with text3.	String type	CHRTRAN("BBACD","A","C") → "BBCCD"

FIND(text1, text2[, num])	Returns the start position of the num-th occurrence of text2 in text1. (1-based index)	Real type	FIND("ABBACDAB","A",2) → 4
GETIDXSTRCMP([Search Direction: Up 1, Down 0], [Comparison Variable Index @{Variable Name}, [Comparison String 'NNG'], [String Equality Condition 1, Inequality Condition 0], [Condition Repeat Satisfaction Count]]	Based on the current column position, return the satisfied column positions x number of times. Where x represents the number of times the compared string is repeated.	Real type	GETSTRCOMBINE(@{Text_Results(Morpheme)}, GETIDXSTRCMP(1, @{Text_Rreulst(Part of Speech)}, 'NNG', 0,1), @rownum)
GETSTRCOMBINE([Target String Variable @{Field Name}], [Extraction Start Column], [Extraction End Column])	Return the combined characters of the specified string locations of the specified string variable.	String type	
KOR_GET1(text)	Extracts Hangul characters from text.	String type	KOR_GET1("ABC ㄱㄴㄷ") → "ㄱㄴㄷ"
KOR_GET2(text, num1[, num2])	From text[num1…num2] returns Hangul-only substring. If num2 omitted, goes to end.	String type	KOR_GET2("가나다 a b 라마",2,6) → "나다라마"
LEFT(text1, num)	Returns the leftmost num characters of text1.	String type	LEFT("ABBACDAB",3) → "ABB"
LEN(text1)	Returns the character length of text1.	Real type	LEN("ABBACDAB") → 8
LIKE(text1, text2)	Returns TRUE if text1 matches the pattern text2.	String type	LIKE("ABBACD","*CD") → TRUE
LOWER(text1)	Converts text1 to lowercase.	String type	LOWER("AbCD") → "abcd"

LTRIM(text1[, text2])	Removes from the left all leading characters contained in text2.	String type	LTRIM("AABACD","A") → "BACD"
OCCURS(text1, text2)	Counts how many times text2 occurs in text1.	Real type	OCCURS("AABACD", "A") → 3
REPT(text1, num)	Repeats text1 num times.	String type	REPT("A",3) → "AAA"
RIGHT(text1, num)	Returns the rightmost num characters of text1.	String type	RIGHT("AABACD",2) → "CD"
RTRIM(text1[, text2])	Removes from the right all trailing characters contained in text2.	String type	RTRIM("DCABAA","A") → "DCAB"
SPACE(num)	Returns a string of num spaces (REPT(" ", num)).	String type	SPACE(3) → " "
STRSORT(text)	Returns text with its characters sorted (ascending).	String type	STRSORT("adbc") → "abcd"
SUBSTR(text1, num1[, num2])	Returns substring starting at num1 of length num2 (or to end if omitted).	String type	SUBSTR("AABACD",2,3) → "ABA"
TRIM(text1[, text2])	Removes from both ends all characters contained in text2.	String type	TRIM("AABACDA","A") → "BACD"
UPPER(text1)	Converts text1 to uppercase.	String type	UPPER("aaBAcD") → "AABACD"
UTF8(text1)	Decodes percent-encoded UTF-8 (URL decode).	String type	UTF8("%EB%B9%B5") → "빵"

A1.6 Date/Time

Function Name	Description	Return Value	Example
CMONTH(date)	Returns the month name of the given date.	String type	CMONTH(#2004-01-01#) → "January"

CWEEK(date)	Returns the day of the week name of the given date.	String type	CWEEK(#2004-01-01#) → "Thursday"
DATE([nYear, nMonth, nDay])	Returns a date value. If omitted, returns the current date.	Date type	DATE(2004, 1, 1) → #2004-01-01#
DATETIME([nYear, nMonth, nDay [, nHours [, nMinutes [, nSeconds]]]])	Returns a date and time value. If omitted, returns the current date and time.	Date type	DATETIME(2004, 1, 1, 12) → #2004-01-01 12:00:00#
DAY(date)	Returns the day of the month.	Real type	DAY(#2007-04-06#) → 6
DAYOFYEAR(date)	Returns the day number within the year.	Real type	DAYOFYEAR(#2004-02-01#) → 32
DAYS(date1, date2)	Returns the difference in days between two dates.	Real type	DAYS(#2004-01-01#, #2003-02-01#) → 334
GODAY(date, day)	Adds or subtracts a specified number of days to/from a date.	Date type	GODAY(#2004-02-01#, 5) → #2004-02-06#
GOHOUR(date, day)	Adds or subtracts a specified number of hours to/from a date.	Date type	GOHOUR(#2004-01-01 13:30:20#, 2) → #2004-01-01 15:30:20#
GOMINUTE(date, day)	Adds or subtracts a specified number of minutes to/from a date.	Date type	GOMINUTE(#2004-01-01 13:30:20#, 2) → #2004-01-01 13:32:20#
GOMONTH(date, month)	Adds or subtracts a specified number of months to/from a date.	Date type	GOMONTH(#2004-02-01#, 5) → #2004-07-01#
GOSECOND(date, day)	Adds or subtracts a specified number of seconds to/from a date.	Date type	GOSECOND(#2004-01-01 13:30:20#, 2) → #2004-01-01 13:30:22#

GOYEAR(date, year)	Adds or subtracts a specified number of years to/from a date.	Date type	GOYEAR(#2004-02-01#, 5) → #2009-02-01#
HOUR(date)	Returns the hour component (24-hour format).	Real type	HOUR(#2004-01-01 13:30:20#) → 13
HOURS(date1, date2)	Returns the difference in hours between two date/time values.	Real type	HOURS(#2004-01-01 19:00:00#, #2004-01-01 13:00:00#) → 6
MINUTE(date)	Returns the minute component.	Real type	MINUTE(#2004-01-01 13:30:20#) → 30
MINUTES(date1, date2)	Returns the difference in minutes between two date/time values.	Real type	MINUTES(#2004-01-01 13:30:00#, #2004-01-01 13:00:00#) → 30
MONTH(date)	Returns the month number (1–12).	Real type	MONTH(#2004-01-01 13:30:20#) → 1
MONTHS(date1, date2)	Returns the difference in months between two dates.	Real type	MONTHS(#2004-01-01#, #2003-01-01#) → 12
SEC(date)	Returns the second component.	Real type	SEC(#2004-01-01 13:30:20#) → 20
SECS(date1, date2)	Returns the difference in seconds between two date/time values.	Real type	SECS(#2004-01-01 13:30:40#, #2004-01-01 13:30:20#) → 20
TIME()	Returns the current time of stream execution in hh:mm:ss format.	String type	TIME() → #17:30:20#
TRIMDATE(date)	Returns only the time portion of a date/time value.	Date type	TRIMDATE(#2004-01-01 13:30:20#) → #13:30:20#
TRIMTIME(date)	Returns only the date portion, excluding time.	Date type	TRIMTIME(#2004-01-01 13:30:20#) → #2004-01-01#
WEEK(date)	Returns the weekday number (Sunday=1 … Saturday=7).	Real type	WEEK(#2004-01-01 13:30:20#) → 5

YEAR(date)	Returns the year.	Real type	YEAR(#2004-01-01 13:30:20#) → 2004
YEARS(date1, date2)	Returns the difference in years between two dates.	Real type	YEARS(#2004-01-01#, #2003-01-01#) → 1

A1.7 Variable Statistics

Continuous Variables

Function Name	Description	Return Value	Example
_AVEDEV(num)	Mean absolute deviation.	Real type	—
_CORR(num1, num2)	Linear or rank correlation between variables.	Real type	—
_COUNT(num)	Count of numeric data (excludes missing).	String type	—
_CSS(num)	Corrected sum of squares.	Real type	—
_CV(num)	Coefficient of variation.	Real type	—
_DEVSQ(num)	Sum of squared deviations from the sample mean.	Real type	—
_GEOMEAN(num)	Geometric mean.	Real type	—
_HARMMEAN(num)	Harmonic mean.	Real type	—
_IQR(num)	Interquartile range (IQR).	Real type	—
_KURTOSIS(num)	Kurtosis.	Real type	—
_MAD(num, flag)	Median absolute deviation: if flag=0 → `mean(	Real type	X-mean(X)
_MAX(num)	Maximum value.	Real type	—
_MAXINDEX(num)	Row index of the maximum value.	Real type	—
_MAXINGRP(idx1, val, idx2)	For records where var idx1 equals val, returns max of var idx2.	Unknown	—
_MEAN(num)	Mean.	Real type	—
_MEDIAN(num)	Median.	Real type	—
_MIDRANGE(num)	Midrange ((min+max)/2).	Real type	—

_MIN(num)	Minimum value.	Real type	—
_MININDEX(num)	Row index of the minimum value.	Real type	—
_MININGRP(idx1, val, idx2)	For records where var idx1 equals val, returns min of var idx2.	Real type	—
_MODE(num)	Mode.	Real type	—
_MOMENT(num1, num2)	n-th sample moment.	Real type	—
_PERCENTRANK(num)	Percent rank.	Real type	—
_Q0(num)	1st quartile (Q1).	Real type	—
_Q1(num)	2nd quartile.	Real type	—
_Q2(num)	3rd quartile (equals median).	Real type	—
_Q3(num)	4th quartile.	Real type	—
_Q4(num)	5th quartile.	Real type	—
_RANGE(num)	Range (max−min).	Real type	—
_RMS(num)	Root-mean-square (RMS).	Real type	—
_RSQ(num1, num2)	Coefficient of determination (Pearson r^2).	Real type	—
_SKEWNESS(num)	Skewness.	Real type	—
_SQRSUM(num)	Sum of squares.	Real type	—
_STD(num)	Sample standard deviation.	Real type	—
_STDP(num)	Population standard deviation.	Real type	—
_SUM(num)	Sum.	Real type	—
_TRIMMEAN(num1, num2)	Trimmed mean (excludes outliers).	Real type	—
_USS(num)	Uncorrected sum of squares.	Real type	—
_VARIANCE(num)	Sample variance.	Real type	—
_VARP(num)	Population variance.	Real type	—

Discrete Variables

Function Name	Description	Return Value	Example
_CLASSCNT(num)	Number of classes (distinct values).	Real type	—

_CLASSTOVAL(num1, num2)	Returns the original value of class num2.	Real type	—
_MAXCNTINDEX(num)	Index of the most frequent class.	Real type	—

A1.8 Information

Function Name	Description	Return Value	Example
BETWEEN(val, lval, hval)	Returns TRUE if val is between lval and hval.	Integer type	BETWEEN(1,0,2) → TRUE
DATASIZE()	Returns row count of the current dataset.	Real type	DATASIZE() → 100
EQSTR(val1, val2[, …, valN])	Returns TRUE if all strings are equal.	Integer type	EQSTR("A","A") → TRUE
ISALPHA(text)	TRUE if alphabetic.	Integer type	ISALPHA("A") → TRUE
ISBOOL(val)	TRUE if Boolean-evaluable.	Integer type	ISBOOL(1=2) → TRUE; ISBOOL(123) → FALSE
ISDATE(val)	TRUE if date type.	Integer type	ISDATE(#2004-01-01#) → TRUE
ISDIGIT(text)	TRUE if numeric string.	Integer type	ISDIGIT("123") → TRUE
ISIN(val, val1[, …, valN])	TRUE if val matches any of val1…valN.	Integer type	ISIN("BAG","BAG","Nice","Book") → TRUE
ISLOWER(text)	TRUE if lowercase.	Integer type	ISLOWER("a") → TRUE
ISNULL(val)	TRUE if missing/NULL.	Integer type	ISNULL(@NULL) → TRUE
ISNUMERIC(val)	TRUE if integer or real.	Integer type	ISNUMERIC(1.2) → TRUE

ISSTRING(val)	TRUE if string type.	Integer type	ISSTRING("123") → TRUE
ISUPPER(text)	TRUE if uppercase.	Integer type	ISUPPER("A") → TRUE
MAX(val1, val2 [, …, valN])	Maximum of inputs.	Unknown	MAX(1,2,3) → 3
MIN(val1, val2 [, …, valN])	Minimum of inputs.	Unknown	MIN(1,2,3) → 1
SIGN(num)	Returns 1 if >0, -1 if <0, 0 if =0.	Integer type	SIGN(-123) → -1
VARTYPE(val)	Returns one of S(string), N(numeric), B(Boolean), D(date), X(NULL), U(unknown).	String type	VARTYPE("A") → "S"

A1.9 Record Function

Function Name	Description	Return Value	Example
RAVERAGE(f, [varIdx1…N])	Mean across specified variables (all if omitted).	Real type	—
RCNTBTN(val1, val2[, varIdx1…N])	Count of variables with value ≥ val1 and ≤ val2.	Integer type	—
RCNTEQ(val[, varIdx1…N])	Count of variables equal to val.	Integer type	—
RCNTEQGT(val[, varIdx1…N])	Count of variables ≥ val.	Integer type	—
RCNTEQLT(val[, varIdx1…N])	Count of variables ≤ val.	Integer type	—
RCNTGT(val[, varIdx1…N])	Count of variables > val.	Integer type	—
RCNTLT(val[, varIdx1…N])	Count of variables < val.	Integer type	—
RCOUNT(f[, varIdx1…N])	Count per variable (all fields if omitted).	Real type	—

RCSS([varIdx1…N])	Corrected sum of squares (continuous vars if omitted).	Real type	—
RDEVSQ([varIdx1…N])	Sum of squared deviations from sample mean.	Real type	—
RMAXA([varIdx1…N])	Maximum value (includes text/logical; text=0, FALSE=0, TRUE=1).	Real type	—
RMAXVAR([varIdx1…N])	Variable name with the maximum value.	String type	—
RMAXVARIDX([varIdx1…N])	Variable index with the maximum value.	Real type	—
RMEAN([varIdx1…N])	Mean (continuous vars if omitted).	Real type	—
RMINA([varIdx1…N])	Minimum value (includes text/logical; text=0, FALSE=0, TRUE=1).	Real type	—
RMINVAR([varIdx1…N])	Variable name with the minimum value.	String type	—
RMINVARIDX([varIdx1…N])	Variable index with the minimum value.	Real type	—
RRANK(ranking[, varIdx1…N])	Value at the given rank among variables (continuous-only if omitted).	Real type	—
RRANKIDX(ranking[, varIdx1…N])	Variable index at the given rank.	Real type	—
RRMS([varIdx1…N])	Root-mean-square.	Real type	—
RSTD([varIdx1…N])	Standard deviation (continuous-only if omitted).	Real type	—
RSTDA([varIdx1…N])	Sample standard deviation; excludes missing; includes text (text=0).	Real type	—
RSTDP([varIdx1…N])	Population standard deviation; excludes missing and text.	Real type	—
RSTDPA([varIdx1…N])	Population standard deviation; excludes missing; includes text (text=0).	Real type	—
RSUM([varIdx1…N])	Sum (continuous-only if omitted).	Real type	—

RUSS([varIdx1…N])	Uncorrected sum of squares (continuous-only if omitted).	Real type	—
RVAR([varIdx1…N])	Variance (continuous-only if omitted).	Real type	—
RVARA([varIdx1…N])	Sample variance; excludes missing; includes text (text=0).	Real type	—
RVARP([varIdx1…N])	Population variance; excludes text.	Real type	—
RVARPA([varIdx1…N])	Population variance; excludes missing; includes text (text=0).	Real type	—

A1.10Macro

You can use **macros** to reference predefined values. To use a macro, type **@** followed by the name of the desired value.

Macro	Description	Return Value
@FALSE	Enter false value of bool.	String type
@NULL	Inserts a NULL value.	Unknown
@ROWNUM	Returns the index of the record currently being processed.	Real type
@TRUE	Enter true value of bool.	String type
@VAR	Refers to the variable currently being computed when generating multiple derived fields (usable only in multi-derived-field contexts).	Unknown
@VARIDX	Returns the index of the variable currently being computed when generating multiple derived fields (usable only in multi-derived-field contexts).	Real type

A1.11 Other Fuctions

Function Name	Description	Return Value
GETDEFDATE(key)	Returns a predefined date/time value for key.	Date type

GETDEFNUM(key)	Returns a predefined numeric value for key.	Real type
GETDEFSTR(key)	Returns a predefined string value for key.	String type
GETROWNUM(field_index, val)	Searches variable field_index for val and returns the row number of the match.	Real type
GETVALUE(field_index, row)	Returns the row-th value from variable field_index.	Unknown
LARGEST(field_index, k)	Returns the k-th largest (excluding missing values) in variable field_index.	Real type
LOOKUP(field_index1, val, field_index2)	Finds val in variable field_index1 and returns the value at the same row in variable field_index2.	Unknown
MOVINGAVERAGE(field_index, k)	Moving average of variable field_index.	Real type
MOVINGMEDIAN(field_index, k)	Moving median of variable field_index.	Real type
MOVINGSTD(field_index, k)	Moving standard deviation of variable field_index.	Real type
REGSLOPE(field_index, k)	Regression slope of variable field_index with time as X-axis.	Real type
SMALLEST(field_index, k)	Returns the k-th smallest value (ascending order).	Real type