Dashboard Function Reference

5.2.0

August 2014

This document applies to Apama 5.2.0 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 2013-2014 Software AG, Darmstadt, Germany and/or Software AG USA Inc., Reston, VA, USA, and/or its Subsidiaries and or/its Affiliates and/or their licensors.

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software AG.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA Inc. and/or its Subsidiaries and/or its Affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at http://documentation.softwareag.com/legal/.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are located at http://documentation.softwareag.com/legal/ and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices and license terms, please refer to "License Texts, Copyright Notices and Disclaimers of Third Party Products." This document is located at http://documentation.softwareag.com/legal/ and/or in the root installation directory of the licensed product(s).

Document ID: PAM-Dashboard_Function_Reference-5.2.0-20140808@233876

Table of Contents

Preface

About this documentation

This document provides reference information on Dashboard functions, which allow you to perform calculations, filtering, formatting, and other operations on correlator data. Instead of attaching a visualization object directly to a correlator data table or variable, you can specify data tables and variables as arguments to a Dashboard function, and then attach the visualization object to the function. Objects that are attached to functions update dynamically as the function arguments are updated, just as visualization objects that are attached directly to correlator data update dynamically as correlator data is updated.

This document assumes that you have already read *Introduction to Apama*, as well as Building Dashboards in *Developing Apama Applications*.

Preface

How this book is organized

The information in this book is organized as follows:

- Introduction to Dashboard functions, as well as the Dashboard Builder Functions panel and Edit Function dialog.
- Descriptions of functions that operate on and return numerical values or text strings.
- Descriptions of functions that operate on or return tabular data.

Preface

Documentation roadmap

On Windows platforms, the specific set of documentation provided with Apama depends on whether you choose the Developer, Server, or User installation option. On UNIX platforms, only the Server option is available.

Apama provides documentation in three formats:

- HTML viewable in a Web browser

- PDF
- Eclipse Help (if you select the Apama Developer installation option)

On Windows, to access the documentation, select Start > All Programs > Software AG > Apama 5.2 > Apama Documentation . On UNIX, display the `index.html` file, which is in the `doc` directory of your Apama installation directory.

The following table describes the PDF documents that are available when you install the Apama Developer option. A subset of these documents is provided with the Server and User options.

Title	Contents
What's New in Apama	Describes new features and changes since the previous release.
Installing Apama	Instructions for installing the Developer, Server, or User Apama installation options.
Introduction to Apama	Introduction to developing Apama applications, discussions of Apama architecture and concepts, and pointers to sources of information outside the documentation set.
Using Apama Studio	Instructions for using Apama Studio to create and test Apama projects; write, profile, and debug EPL programs; write JMon programs; develop custom blocks; and store, retrieve and playback data.
Developing Apama Applications in Event Modeler	Instructions for using Apama Studio's Event Modeler editor to develop scenarios. Includes information about using standard functions, standard blocks, and blocks generated from scenarios.
Developing Apama Applications in EPL	Introduces Apama's Event Processing Language (EPL) and provides user guide type information for how to write EPL programs. EPL is the native interface to the correlator. This document also provides information for using the standard correlator plug-ins.
Apama EPL Reference	Reference information for EPL: lexical elements, syntax, types, variables, event definitions, expressions, statements.
Developing Apama Applications in Java	Introduces the Apama in-process API for Java, referred to as JMon, and provides user guide type information for how to write Java programs that run on the correlator. Reference information in Javadoc format is also available.
Building Dashboards	Describes how to create dashboards, which are the end-user interfaces to running scenario instances and data view items.
Dashboard Property Reference	Reference information on the properties of the visualization objects that you can include in your dashboards.

Title	Contents
Dashboard Function Reference	Reference information on dashboard functions, which allow you to operate on correlator data before you attach it to visualization objects.
Developing Adapters	Describes how to create adapters, which are components that translate events from non-Apama format to Apama format.
Developing Clients	Describes how to develop C, C++, Java, or .NET clients that can communicate with and interact with the correlator.
Writing Correlator Plug-ins	Describes how to develop formatted libraries of C, C++ or Java functions that can be called from EPL.
Deploying and Managing Apama Applications	Describes how to: • Use the Management & Monitoring console to configure, start, stop, and monitor the correlator and adapters across multiple hosts. • Deploy dashboards over wide area networks, including the internet, and provide dashboards with effective authorization and authentication. • Improve Apama application performance by using multiple correlators, and saving and reusing a snapshot of a correlator's state. • Use the Apama ADBC adapter to store and retrieve data in JDBC, ODBC, and Apama Sim databases. • Use the Apama Web Services Client adapter to invoke Web Services. • Use correlator-integrated messaging for JMS to reliably send and receive JMS messages in Apama applications. • Use Universal Messaging to connect correlators.
Using the Dashboard Viewer	Describes how to view and interact with dashboards that are receiving run-time data from the correlator.

Preface

Contacting customer support

You may open Apama Support Incidents online via the eService section of Empower at http://empower.softwareag.com. If you are new to Empower, send an email to `empower@softwareag.com` with your name, company, and company email address to request an account.

If you have any questions, you can find a local or toll-free number for your country in our Global Support Directory at https://empower.softwareag.com/public_directory.asp and give us a call.

Preface

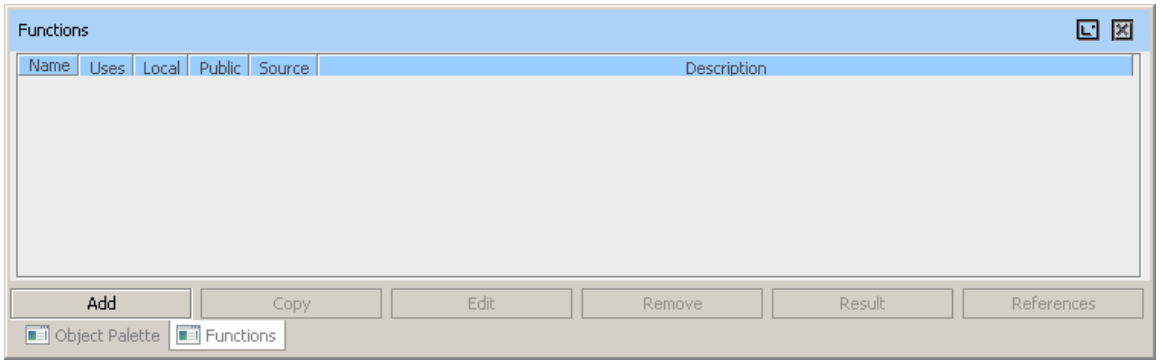
Chapter 1: Introduction to Dashboard Functions

Dashboard functions allow you to perform calculations, filtering, formatting, and other operations on correlator data. Instead of attaching a visualization object directly to a correlator data table or variable, you can specify data tables and variables as arguments to a dashboard function, and then attach the visualization object to the function.

Working with functions

Follow these steps to add, edit, remove, or examine functions:

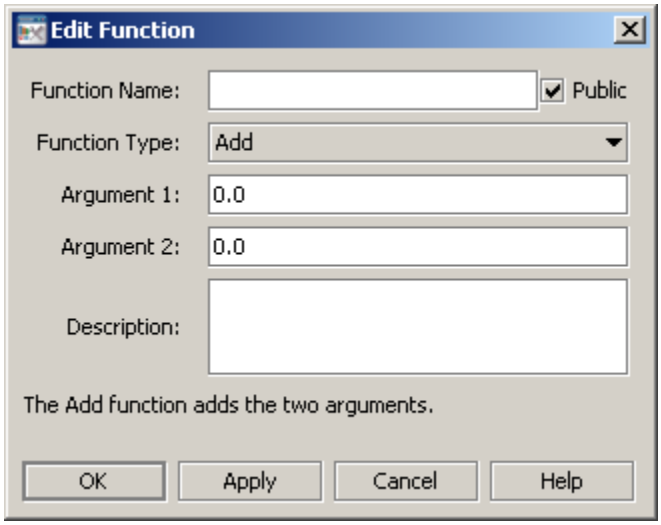
1. Open the dashboard file that contains (or will contain) the visualization object that you want to attach to a function.
2. In the Dashboard Builder, select Functions from the Tools menu. The Functions panel appears.



   The Functions panel has the following buttons:

   - Add: Adds a new function. Brings up the Edit Function dialog.
   - Copy: Copies the selected function. Brings up the Edit Function Name dialog.
   - Edit: Allows you to edit the selected function. Brings up the Edit Function dialog.
   - Remove: Removes the selected function.
   - Result: Brings up a dialog that displays the result of executing the selected function.
   - References: Brings up a dialog that lists all the objects that directly reference the selected function. When you choose an object in that list, it is selected according to its type. When you select a display object, Builder highlights the object in the drawing area. When you select a function, Builder brings up the dialog with the designated object selected.

3. To add or edit a function, click Add or Edit In the Functions panel. The Edit Function dialog appears.



4. Fill in the fields of the Edit Function dialog, and click OK (to apply the values and close the dialog) or Apply (to apply the values and leave the dialog open).

   The Edit Function dialog has the following fields:

   - Function Name: Specify a name that is unique among functions that have been added to the current dashboard file. The name must not contain spaces. The name function is not allowed.
   - Function Type: Select the function that you want to add and specify arguments for. The dropdown list includes built-in functions as well as user-defined functions (see Creating Custom Functions in the Using Dashboard Functions section of *Building Dashboards*).
   - Argument fields: Once you select a function for the Function Type field, the dialog is populated with the argument fields that are appropriate to the selected function. For each argument field, you can either enter a value or attach the argument to data. To attach the argument to data, right-click in the argument field, select Attach to Data, and select a data source. An argument that has been attached to data is displayed in green. Double-click to edit the data attachment. Right-click and select Attach to Data to change the data source. Right-click and select Detach from Data to remove the attachment.
   - Description: Include a description of any length. This description will be visible in the #Attach to Function Data dialog.

   In the Edit Function dialog, when you are editing a function whose argument refers to another function, you can edit the referenced function without leaving the dialog. When you right-click an argument that contains a reference to a function, an additional item, Edit Function, appears in the popup menu. If you select it, the Edit Function dialog for that function replaces the current Edit Function dialog. If you have unsaved changes you are prompted to save or discard them, or cancel the operation. In addition a button, Back, appears in the Edit Function dialog that takes you back to the function you were previously editing.

Once a function has been added in this way (with arguments specified), you can use the Attach to Function Data dialog in order to attach it to properties of visualization objects in the current dashboard file. Objects that are attached to functions update dynamically as the function arguments are updated, just as visualization objects that are attached directly to correlator data update dynamically as correlator data is updated.

Note that the added functions can be edited only from within the dashboard file that was opened when the functions were added. In addition, a function is only available for use within the

dashboard file that was opened when the function was added, or (for public functions) within a file that includes the dashboard file that was opened when the function was added.

The reference documentation on dashboard functions is divided into two sections:

- "Scalar Functions" on page 13: Describes built-in dashboard functions that operate on and return numerical values or text strings.
- "Tabular Functions" on page 27: Describes built-in dashboard functions that operate on or return tabular data.

See also Using Dashboard Functions in *Building Dashboards*.

Introduction to Dashboard Functions

Chapter 2: Scalar Functions

This section lists the Dashboard functions that operate on and return numerical values or text strings.

Add

Returns the result of adding the two arguments.

Arguments

This function has the following arguments:

- Argument1: Numeric value to be added.
- Argument2: Numeric value to be added.

The function returns a numeric value.

Scalar Functions

Average

Returns the average of the two arguments.

Arguments

This function has the following arguments:

- Argument1: Numeric value.
- Argument2: Numeric value.

The function returns a numeric value.

Scalar Functions

Boolean Expression

Returns 1 (true) if the result of performing a specified comparison is true; returns 0 (false) otherwise.

Arguments

The function has the following arguments:

- Value 1: Text string that specifies the first value to compare.
- Operator: Text string that specifies the comparison operator. Supply one of the following:
  - and

- or
- xor
- =
- !=
- >
- <
- >=
- <=

- Value 2: Text string that specifies the second value to compare.

This function returns a numerical value.

Scalar Functions

Concatenate

Returns the result of combining the two arguments into a single text string.

Arguments

This function has the following arguments:

- Value1: Numeric value or text string.
- Value2: Numeric value or text string.

The function returns a text string.

Scalar Functions

Correlator Time Format

Converts a correlator timestamp to either epoch time in milliseconds or the specified date/time format.

Arguments

- Correlator Time: Correlator timestamp that you want to convert. This argument is required. If it is not specified or if the specified value is invalid the string `"0"` is returned and any specified formatting is not applied.
- Format: Optional. Leave this field blank to convert the correlator timestamp to epoch time. The returned value is in milliseconds. Or, specify a date/time pattern to use to format the correlator timestamp. You can specify any pattern format supported by the Java class `SimpleDateFormat`. If you specify an invalid value the string `"0"` is returned.

A correlator timestamp is in seconds with a decimal point before the milliseconds, for example, `1043189336.2`.

This function returns a string.

Scalar Functions

Date Add

Returns the result of adding the specified number (which may be negative) of date part intervals to the specified date, and returns a string representing the resulting date/time.

Arguments

This function has the following arguments:

- Date: Text string specifying the date to which is added the specified number of date parts. This must be either a formatted date/time string or a Java standard date/time argument in milliseconds from Jan 1, 1970.
- Number: Numeric value. The number of date parts to add to the specified date.
- Date Part: Text string specifying the date part, the specified number of which are to be added to the specified date. Specify s, m, h, d, w, M, q or y for seconds, minutes, hours, days, weeks, months, quarters, or years.
- Date Format: Text string specifying the format of the function result. Specify a pattern string suitable for use with the Java `SimpleDateFormat` class. For example, the format MMMM dd, yyyy hh:mm:ss a results in dates of the form exemplified by August 30, 2003 05:32:12 PM. If no Date Format is given, the string is returned in the form exemplified by 08/30/03 05:32 PM. Use q, qqq or qqqq for short, medium or long versions of quarter notation. For example, qqq-yyyy results in a string of the form exemplified by Qtr 1-2005.

The function returns a text string.

Scalar Functions

Date Ceiling

Returns the ceiling of Date with respect to Date Part. In other words, the function determines which Date Part interval contains the Date, and returns a string representing the start value of the next Date Part interval.

Arguments

This function has the following arguments:

- Date: Text string specifying the date whose ceiling is to be returned. This must be either a formatted date/time string or a Java standard date/time argument in milliseconds from Jan 1, 1970.
- Date Part: Text string specifying the date part with respect to which the specified date's ceiling is to be returned. Specify s, m, h, d, w, M, q or y for seconds, minutes, hours, days, weeks, months, quarters, or years.

- Date Format: Text string specifying the format of the function result. Specify a pattern string suitable for use with the Java `SimpleDateFormat` class. For example, the format MMMM dd, yyyy hh:mm:ss a results in dates of the form exemplified by August 30, 2003 05:32:12 PM. If no Date Format is given, the string is returned in the form exemplified by 08/30/03 05:32 PM. Use q, qqq or qqqq for short, medium or long versions of quarter notation. For example, qqq-yyyy results in a string of the form exemplified by Qtr 1-2005.

The function returns a text string.

Scalar Functions

Date Compare

Compares Date 1 and Date 2, each rounded down to the nearest Date Part.

Return Value

If Date 1 (rounded down to the nearest Date Part) is less than Date 2 (rounded down to the nearest Date Part), the function returns -1. If Date 1 (rounded down to the nearest Date Part) is greater than Date 2 (rounded down to the nearest Date Part), the function returns 1. If Date 1 (rounded down to the nearest Date Part) equals Date 2 (rounded down to the nearest Date Part), the function returns 0.

For example, comparing 08/30/03 05:32 PM to 08/30/03 04:47 PM with Date Part set to m (for minute resolution) returns 1, while setting Date Part to d (for day resolution) causes this function to return 0.

Arguments

This function has the following arguments:

- Date 1: Text string specifying one of the dates to be compared. It must be either a formatted date/time string or a Java standard date/time argument in milliseconds from Jan 1, 1970.
- Date 2: Text string specifying the other date to be compared. It must be either a formatted date/time string or a Java standard date/time argument in milliseconds from Jan 1, 1970.
- Date Part: Text string that controls the resolution of the comparison. Specify s, m, h, d, w, M, q or y for seconds, minutes, hours, days, weeks, months, quarters or years.

The function returns a number.

Scalar Functions

Date Difference

Returns the integer number of Date Part intervals by which Date 1 (rounded down to the nearest Date Part) is less than Date 2 (rounded down to the nearest Date Part). For example, the difference between 05/12/05 05:32 PM and 05/15/05 04:47 PM with Date Part set to d (for day) returns 3.

Arguments

This function has the following arguments:

- Date 1: Text string specifying the earlier date. It must be either a formatted date/time string or a Java standard date/time argument in milliseconds from Jan 1, 1970.
- Date 2: Text string specifying the later date. It must be either a formatted date/time string or a Java standard date/time argument in milliseconds from Jan 1, 1970.
- Date Part: Text string that specifies the date part with respect to which the difference is to be calculated. Specify s, m, h, d, w, M, q or y for seconds, minutes, hours, days, weeks, months, quarters or years.

The function returns a number.

Scalar Functions

Date Floor

Returns the floor of Date with respect to Date Part. In other words, the function determines which Date Part interval contains Date, and returns a string representing the starting date/time value of that interval.

Arguments

This function has the following arguments:

- Date: Text string specifying the date whose floor is to be returned. This must be either a formatted date/time string or a Java standard date/time argument in milliseconds from Jan 1, 1970.
- Date Part: Text string specifying the date part with respect to which the specified date's floor is to be returned. Specify s, m, h, d, w, M, q or y for seconds, minutes, hours, days, weeks, months, quarters, or years.
- Date Format: Text string specifying the format of the function result. Specify a pattern string suitable for use with the Java `SimpleDateFormat` class. For example, the format MMMM dd, yyyy hh:mm:ss a results in dates of the form exemplified by August 30, 2003 05:32:12 PM. If no Date Format is given, the string is returned in the form exemplified by 08/30/03 05:32 PM. Use q, qqq or qqqq for short, medium or long versions of quarter notation. For example, qqq-yyyy results in a string of the form exemplified by Qtr 1-2005.

The function returns a text string.

Scalar Functions

Date Format

Returns a string representing the specified date in the specified format.

Arguments

This function has the following arguments:

- Date: Text string specifying the date to be formatted. This must be either a formatted date/time string or a Java standard date/time argument in milliseconds from Jan 1, 1970.

- Date Format: Text string specifying the format of the function result. Specify a pattern string suitable for use with the Java `SimpleDateFormat` class. For example, the format MMMM dd, yyyy hh:mm:ss a results in dates of the form exemplified by August 30, 2003 05:32:12 PM. If no Date Format is given, the string is returned in the form exemplified by 08/30/03 05:32 PM.

The function returns a text string.

Scalar Functions

Date Now

Returns a string representing the current date and time in the specified format.

Arguments

This function has the following argument:

- Date Format: Text string specifying the format of the function result. Specify a pattern string suitable for use with the Java `SimpleDateFormat` class. For example, the format MMMM dd, yyyy hh:mm:ss a results in dates of the form exemplified by August 30, 2003 05:32:12 PM. If no Date Format is given, the string is returned in the form exemplified by 08/30/03 05:32 PM.

The function returns a text string.

Scalar Functions

Delta

Returns the rate of change of Value over the specified time interval.

Arguments

This function has the following arguments:

- Value: The numeric value whose rate of change is to be returned.
- Interval: Numeric value specifying the time interval, in seconds, for which the rate of change is to be calculated. If no value is given, the absolute delta is returned.

The function returns a number.

Scalar Functions

Divide

Returns the result of dividing the first argument by the second.

Arguments

This function has the following arguments:

- Argument1: Numeric value specifying the dividend.

- Argument2: Numeric value specifying the divisor.

The function returns a number.

Scalar Functions

Duration

Returns a string representing the specified duration in the specified format.

Arguments

This function has the following arguments:

- Duration: Numeric value specifying the duration, in milliseconds, to be formatted.
- Duration Format: Text string specifying the format of the function result. This string may contain 0 or more of the characters d, s, and . (for example ds) indicating that days, seconds, or milliseconds are to be included, in addition to hours and minutes, in the returned string. If no Duration Format is specified, the string is returned in the form exemplified by 15:32 (hours:minutes).

The function returns a text string.

Scalar Functions

Evaluate Expression As Double

Returns the result of evaluating a specified expression that contains variables, each of which has an associated function argument. The result is returned as a double. Boolean true or false values are returned as 1.0 and 0.0 respectively.

Arguments

The function has the following arguments:

- Expression: Text string that specifies the expression to evaluate. Prefix variable names with%. Use standard arithmetic and logical operators. You can also use a variety of mathematical and string functions, as well as numeric and string constants. Enclose string constants in double quotes.
- Expression variable arguments: When the Expression field of the Edit Function dialog is activated (by pressing Enter or navigating to another field), the dialog displays a text field for each variable. For each field, enter a numeric value or text string. Values whose form is numeric are substituted into the expression as numbers; otherwise they are substituted into the expression as strings.

If a value whose form is numeric needs to be treated as a string, for example to serve as an argument to a string function, surround the variable in Expression with double quotes. Variables enclosed in double quotes are always used as strings. An example of such an expression is length("%var1") + %var2.

This function returns a numerical value.

Scalar Functions

Evaluate Expression As String

Returns the result of evaluating a specified expression that contains variables, each of which has an associated function argument. The result is returned as a text string. Boolean true or false values are returned as 1.0 and 0.0 respectively.

Arguments

The function has the following arguments:

- Expression: Text string that specifies the expression to evaluate. Prefix variable names with%. Use standard arithmetic and logical operators. You can also use a variety of mathematical and string functions, as well as numeric and string constants. Enclose string constants in double quotes.
- Expression variable arguments: When the Expression field of the Edit Function dialog is activated (by pressing Enter or navigating to another field), the dialog displays a text field for each variable. For each field, enter a numeric value or text string. Values whose form is numeric are substituted into the expression as numbers; otherwise they are substituted into the expression as strings.

If a value whose form is numeric needs to be treated as a string, for example to serve as an argument to a string function, surround the variable in Expression with double quotes. Variables enclosed in double quotes are always used as strings. An example of such an expression is length("%var1") + %var2.

This function returns a text string.

Scalar Functions

Format Number

Returns a string representing the specified number in the specified format.

For example, if Number To Format is 50, and Format is $, the function returns $50.00.

Arguments

This function has the following arguments:

- Number To Format: Numeric value specifying the number to be formatted.
- Format: Text string specifying the format of the function result. The format can be specified based on the Java format specification, or with the following shorthand: $ for US dollar money values, $$ for US dollar money values with additional formatting, or () for non-money values, formatted similar to money. Both positive and negative formats can be supplied, for example: #,###;(#,###).

The function returns a text string

Scalar Functions

Get Substitution

Returns the current value of the given Substitution String.

Arguments

This function has the following argument:

- Substitution String: Text string specifying the substitution whose value is to be returned.

The function returns a text string.

Scalar Functions

Init Local Variable

Initializes the local variable to the specified value. If the variable already has a non-empty value, this function does nothing.

Arguments

This function has the following arguments:

- Variable Name: Text string specifying the variable whose value is to be initialized.
- Initial Value: Numeric value or text string specifying the initial value to which the variable is to be set.

This function is useful for initializing a local variable to a value supplied by a data attachment.

Scalar Functions

isWindowsOS

Returns 1 if the dashboard is not running in an applet and the operating system it is running on is Windows; returns 0 otherwise.

Arguments

This function has no arguments.

The function returns a number.

Scalar Functions

Max

Returns larger of the two arguments.

Arguments

This function has the following arguments:

- Argument1: Numeric value.

- Argument2: Numeric value.

This function returns a number.

Scalar Functions

Min

Returns smaller of the two arguments.

Arguments

This function has the following arguments:

- Argument1: Numeric value.
- Argument2: Numeric value.

This function returns a number.

Scalar Functions

Modulo

Divides Value by Divisor and returns the remainder.

Arguments

This function has the following arguments:

- Value: Numeric value to be divided by Divisor.
- Divisor: Numeric value by which to divide Value.

This function returns a number.

Scalar Functions

Multiply

Returns the result of multiplying the first argument by the second.

Arguments

This function has the following arguments:

- Argument1: Numeric value specifying one of the factors.
- Argument2: Numeric value specifying the other factor.

The function returns a number.

Scalar Functions

Percent

Returns the percentage of Value, given the range defined by Min Value and Max Value.

Arguments

This function has the following arguments:

- Value: Numeric value specifying one of the factors.
- Min Value: Numeric value specifying one of the factors.
- Max Value: Numeric value specifying the other factor.

The function returns a number between 0 and 100.

Scalar Functions

Quick Set Sub

Sets a substitution string to the specified value.

This function executes very quickly because, unlike the standard Set Substitution function, it does not search for and modify data attachments that use the substitution, it does not apply the change to child panels of the current panel, nor does it change the value of the local variable, if any, that is mapped to the substitution.

Arguments

This function has the following arguments:

- Substitution String: Text string specifying the substitution whose value is to be set.
- Value: Numeric value or text string specifying the value to which Substitution String is to be set.

This function is suitable for setting a substitution used only in a command or drilldown, but is not suitable for setting a substitution used in data attachments.

Scalar Functions

Replace Value

Returns the replacement string that Replacement Values associates with Value.

For example, if Value is Windows NT and Replacement Values is 'Windows NT':winnt Windows2000:win2k, the text string returned is winnt.

Arguments

This function has the following arguments:

- Value: Text string whose associated replacement string is to be returned.

- Replacement Values: Text string specifying value/replacement-string pairs. This is a space-separated list of pairs of the form *value:replacement-string*. Use a colon to separate the value from its associated replacement string. Use a space to separate one pair from another in the list. If a value or replacement string contains a space or colon, enclose that value or replacement string in single quotes.
- Return Value if No Match: Numerical value that controls what is returned if none of the pairs in Replacement Values has a value that matches Value. If Return Value if No Match is set to 1, Value is returned when no match is found. If Return Value if No Match is set to 0, the empty string is returned when no match is found.

The function returns a text string.

Scalar Functions

Set Substitution

Sets the given substitution string to the given value, and returns the value.

Arguments

This function has the following arguments:

- Substitution String: Text string specifying the substitution whose value is to be set.
- Value: Numeric value or text string specifying the value to which Substitution String is to be set.

The function returns a text string.

Scalar Functions

Set Substitutions By Lookup

Sets multiple substitutions based on the values in a specified lookup table.

Arguments

The function has the following arguments:

- Key: Text string or numeric value. This value identifies a row of Lookup Table, provided it matches a value in Lookup Table's first column.
- Lookup Table: Table whose first column contains key values and whose remaining columns contain substitution values. These remaining columns have as names the names of substitution variables (and, in particular, they start with $).

Key is compared against the values in the first column of Lookup Table in order to determine which row of the lookup table to use to set substitution values. For each additional column in Lookup Table (where the column name starts with $), a substitution is set. The substitution name is the name of the column and the substitution value is the value from that column in the row whose first column matches Key.

This function returns a table.

Scalar Functions

Subtract

Returns the result of subtracting the second argument from the first.

Arguments

This function has the following arguments:

- Argument1: Numeric value to be subtracted from.
- Argument2: Numeric value to subtract.

The function returns a number.

Scalar Functions

Validate Substitutions

Validates a substitution string against the given table of valid values. Returns a substitution string with only valid values. The returned string is identical to the specified substitution string, except that any values from the specified string that are not found in the first column of the given table are replaced with the first value in the first column of the given table.

Arguments

The function has the following argument:

- Substitution String: Text string consisting a semicolon-separated list of substitutions (variable-name/value pairs) whose values are to be validated
- Valid Value Table: Table whose first column contains all values that are to be considered valid
- Clear If Invalid: If set to 1, the function returns an empty string if the substitution is not found in the table.
- Allow Multiple Values: If set to 1 this will allow the substitution to be a semicolon-separated string of values; each value will be tested for validity and the final result will be assembled from all the valid values (if any).

This function returns a text string.

Scalar Functions

Chapter 3: Tabular Functions

This section lists the Dashboard functions that operate on or return tabular data.

Add All Rows Or Columns

Calculates the sum across cells for each row or column of the specified Table.

Usage notes

If Return Column is 1, the function returns a table with one column. The *n*th cell of the returned column contains the sum of the numerical cells in the *n*th row of the table specified by the argument Table. (If there are no numerical cells in the row, the returned cell contains 0.)

If Return Column is 0, the function returns a table with one row. The *n*th cell of the returned row contains the sum of the numerical cells in the *n*th column of the table specified by the argument Table. (If there are no numerical cells in the column, the returned cell contains N/A, by default—but see the argument Result Label Column, below.) The *n*th column of the returned one-row table is labeled with the label of the *n*th column of the table specified by the argument Table.

Arguments

This function has the following arguments:

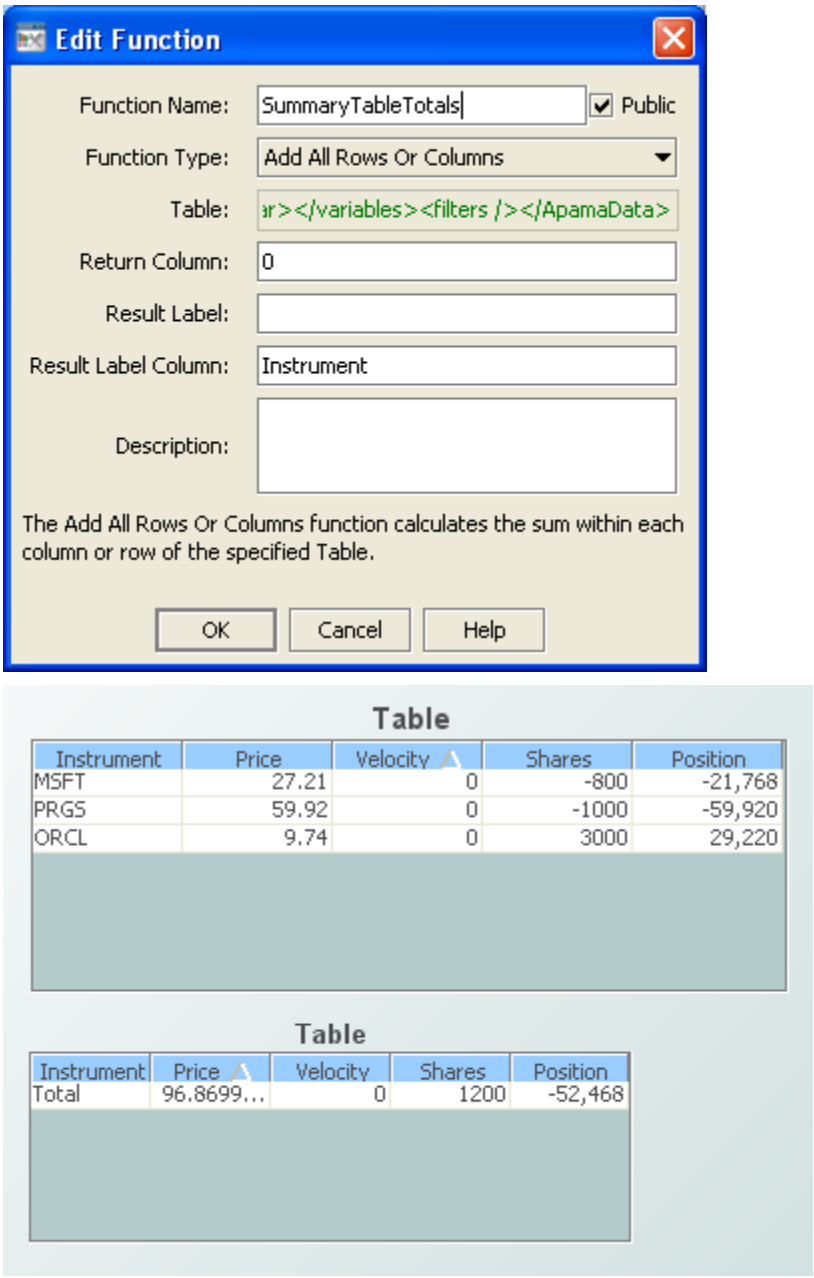
- Table: Table for which row or column sums are to be calculated.
- Return Column: Numeric value that controls whether to return a column or a row of result values. To get a column of result values, one value for each row, set Return Column to 1. To get a row of result values, one value for each column, set Return Column to 0.
- Result Label: Text string that specifies a label for the result row or column. If not specified, the label text is Total. If Return Column is 0, the label appears only if Result Label Column is set to a column of Table that has no numeric values. If Return Column is 1, the label text always appears and Result Label Column is ignored.
- Result Label Column: Text string that specifies the column in which Result Label appears, if Return Column is 0. The specified column must have no numeric values in order for the label to appear. If Return Column is 1, this argument is ignored.

This function returns a table.

Example

The second table below is attached to the function defined by the following dialog. The first table's data table is attached to the argument Table.

Figure 1. Add All Rows Or Columns example



Tabular Functions

Add Columns

Returns a table that includes a column reflecting the sum of two specified columns of a specified table, the sum of a specified value and a specified column, or the sum of two specified values.

Usage notes

- Case 1: Sum of two specified columns of a specified table. This is the case if First Column Name or Numeric Value and Second Column Name or Numeric Value both specify a column of Table.
- Case 2: Sum of a specified value and a specified column. This is the case if one of First Column Name or Numeric Value and Second Column Name or Numeric Value specifies a column of Table and the other specifies a numeric value.
- Case 3: Sum of two specified values. This is the case if First Column Name or Numeric Value and Second Column Name or Numeric Value both specify a numeric value.

In case 1, the *n*th cell in the returned column corresponds to the *n*th cell of the table specified by the argument Table. Each cell of the returned column contains the result of adding the corresponding row's cell in First Column Name or Numeric Value to the cell in Second Column Name or Numeric Value.

In case 2, the *n*th cell in the returned column corresponds to the *n*th cell of the table specified by the argument Table. Each cell of the returned column contains the result of adding the corresponding row's cell in the specified column to the specified numeric value.

In case 3, each cell of the returned column contains the sum of the two specified numeric values.

In the returned table, the sum column is preceded by copies of all the columns in Table.

Arguments

The function has the following arguments:

- Table: Table that contains the columns to be summed.
- First Column Name or Numeric Value: Text string specifying the first column to be included in the sum, or numerical value to be included in the sum.
- Second Column Name or Numeric Value: Text string specifying the second column to be included in the sum, or numerical value to be included in the sum.
- Result Column Name: Text string that specifies the name of the column containing the sums. You must supply a value for this argument.

This function returns a table.

Tabular Functions

Average All Rows Or Columns

Calculates the average across cells for each row or column of the specified Table.

Usage notes

If Return Column is 1, the function returns a table with one column. The *n*th cell of the returned column contains the average of the numerical cells in the *n*th row of the table specified by the argument Table. (If there are no numerical cells in the row, the returned cell contains 0.)

If Return Column is 0, the function returns a table with one row. The *n*th cell of the returned row contains the average of the numerical cells in the *n*th column of the table specified by the argument Table. (If there are no numerical cells in the column, the returned cell contains N/A, by default—

but see the argument Result Label Column, below.) The *n*th column of the returned one-row table is labeled with the label of the *n*th column of the table specified by the argument Table.

Arguments

This function has the following arguments:

- Table: Table for which row or column averages are to be calculated.
- Return Column: Numeric value that controls whether to return a column or a row of result values. To get a column of result values, one value for each row, set Return Column to 1. To get a row of result values, one value for each column, set Return Column to 0.
- Result Label: Text string that specifies a label for the result row or column. If not specified, the label text is Average. If Return Column is 0, the label appears only if Result Label Column is set to a column of Table that has no numeric values. If Return Column is 1, the label text always appears and Result Label Column is ignored.
- Result Label Column: Text string that specifies the column in which Result Label appears, if Return Column is 0. The specified column must have no numeric values in order for the label to appear. If Return Column is 1, this argument is ignored.

This function returns a table.

Tabular Functions

Average Columns

Returns a table that includes a column reflecting the average of two specified columns of a specified table, the average of a specified value and a specified column, or the average of two specified values.

Usage notes

- Case 1: Average of two specified columns of a specified table. This is the case if First Column Name or Numeric Value and Second Column Name or Numeric Value both specify a column of Table.
- Case 2: Average of a specified value and a specified column. This is the case if one of First Column Name or Numeric Value and Second Column Name or Numeric Value specifies a column of Table and the other specifies a numeric value.
- Case 3: Average of two specified values. This is the case if First Column Name or Numeric Value and Second Column Name or Numeric Value both specify a numeric value.

In case 1, the *n*th cell in the returned column corresponds to the *n*th cell of the table specified by the argument Table. Each cell of the returned column contains the result of computing the average of the corresponding row's cell in First Column Name or Numeric Value and the cell in Second Column Name or Numeric Value.

In case 2, the *n*th cell in the returned column corresponds to the *n*th cell of the table specified by the argument Table. Each cell of the returned column contains the result of computing the average of the corresponding row's cell in the specified column and the specified numeric value.

In case 3, each cell of the returned column contains the average of the two specified numeric values.

In the returned table, the average column is preceded by copies of all the columns in Table.

Arguments

The function has the following arguments:

- Table: Table that contains the columns to be averaged.
- First Column Name or Numeric Value: Text string specifying the first column to be included in the average, or numerical value to be included in the average.
- Second Column Name or Numeric Value: Text string specifying the second column to be included in the average, or numerical value to be included in the average.
- Result Column Name: Text string that specifies the name of the column containing the averages. You must supply a value for this argument.

This function returns a table.

Tabular Functions

Baseline Over Time

Calculates a baseline average of the values in the specified table over the specified number of specified date part intervals, and offsets the timestamp to a specified reference time.

Arguments

The function has the following arguments:

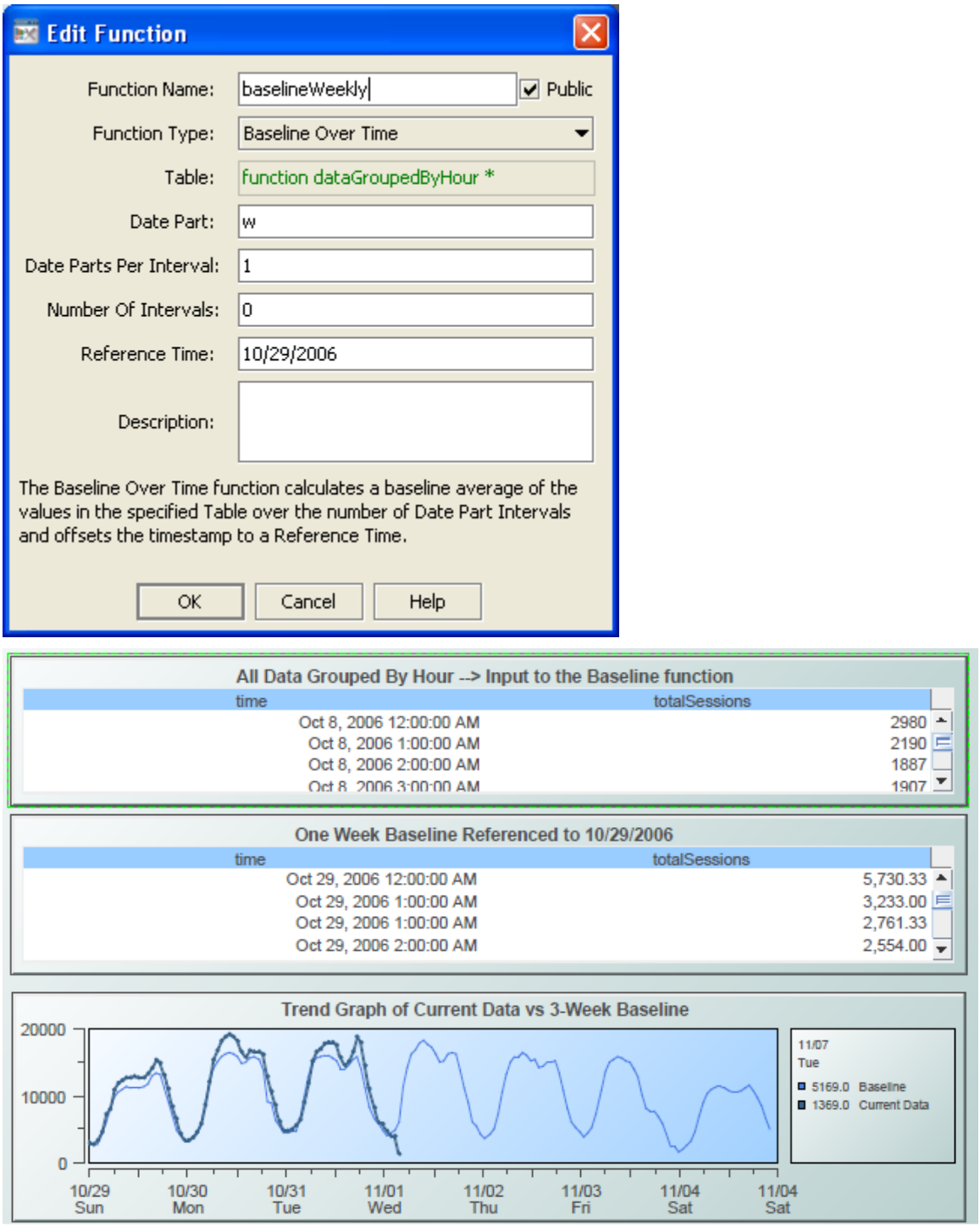
- Table: Data table for which the baseline is to be calculated. The specified table must contain a time column and a number column.
- Date Part: Text string specifying the date unit to use. Enter s, m, h, d, w, M, q, or y, for seconds, minutes, hours, days, weeks, months, quarters, or years. If left blank, the argument defaults to seconds.
- Date Parts Per Interval: Number of date parts in each interval over which the baseline is to be calculated.
- Number Of Intervals: Number of intervals over which the baseline is to be calculated. If this argument is set to 0, the baseline is calculated over all the data in the table.
- Reference Time: After the baseline average has been calculated over the range of data specified, all values in the resulting time column are offset to start at the given reference time. This provides an easy way for the baseline to be plotted in a trend graph against a current set of values.

The function returns a table.

Example

The trend graph below is attached to the function defined by the following dialog. In the trend graph, the thin, light blue line is the baseline, the average of the data over three one-week periods. The dark blue line is the current data. The first table's data table is attached to the argument Table. The second table shows the baseline data. Note that the current data for Sunday is lower than the rest of the current data. The graph shows that this is not anomalous, since the baseline data for Sunday is also lower than the rest of the baseline data.

Figure 2. Baseline Over Time example



Tabular Functions

Buffer Table Rows

Appends all rows of the input table to a buffer table that contains rows from previous updates. Returns the buffer table.

Usage notes

This function is useful for buffering a table argument to another function in cases where the updates to the table may arrive rapidly (for example, from an event-driven data source), in order to ensure that the other function receives all rows.

Arguments

The function has the following arguments:

- Table: Table whose rows are to be appended to the buffer table.
- Number Of Rows: Numeric value that specifies the number of rows in the returned buffer table. If necessary, older rows are removed to maintain this value.

Note: If the result of this function is used as the input to another function, all rows are removed from the buffer table after the other function is updated, regardless of value of Number of Rows.

This function returns a table.

Tabular Functions

Combine

Returns the result of combining two specified tables into a single table.

Usage notes

When Combine Rows is 0, the result contains the columns from Table 1 followed by the columns from Table 2. Each result row consists of the *n*th row from Table 1 followed by the *n*th row from Table 2. If Table 1 and Table 2 have a different number of rows, trailing result rows are padded with cells that are contain 0 or the empty string.

When Combine Rows is 1 and Ignore Column Names is 0, the result contains the rows from Table 1 followed by the rows from Table 2. The result table contains the column labels from Table 1 followed by the column labels that appear only in Table 2. In the result table, 0 or the empty string appears in cells that are in rows from one table and in a column that appears only in the other table.

When Combine Rows is 1 and Ignore Column Names is 1, the result contains the rows from Table 1 followed by the rows from Table 2. The result table contains only columns that appear in both Table 1 and Table 2.

Arguments

This function has the following arguments:

- Table 1: Table to be included in the combination operation.
- Table 2: Table to be included in the combination operation.
- Combine Rows: Numerical value that determines whether rows or columns are merged. When Combine Rows is 0, the result contains the columns from Table 1 followed by the columns from

Table 2. When Combine Rows is 1, the result contains the rows from Table 1 followed by the rows from Table 2.

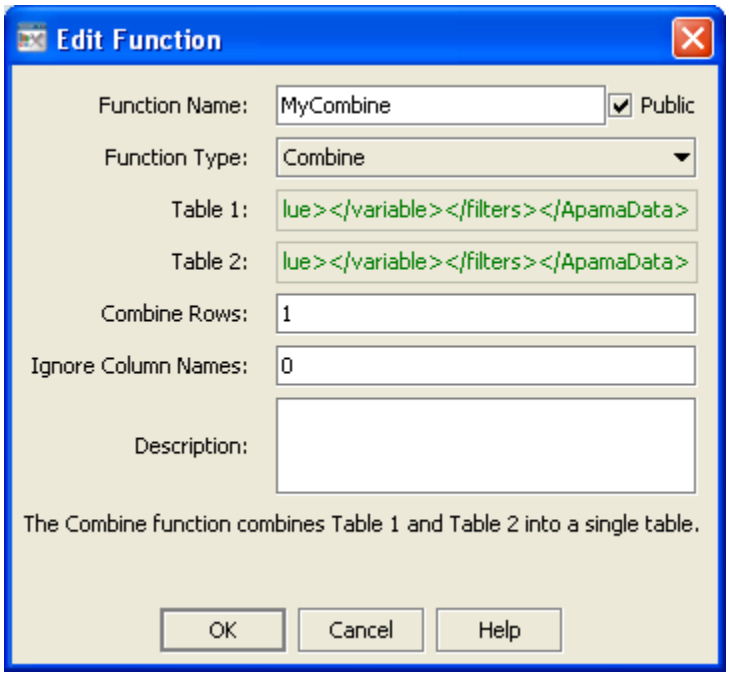
- Ignore Column Names: Numerical value that determines which columns are included in the result. When Combine Rows is 1 and Ignore Column Names is 1, the result table contains only columns that appear in both Table 1 and Table 2.When Combine Rows is 1 and Ignore Column Names is 0, the result table contains the column labels from Table 1 followed by the column labels that appear only in Table 2. This argument is ignored when Combine Rows is 0.

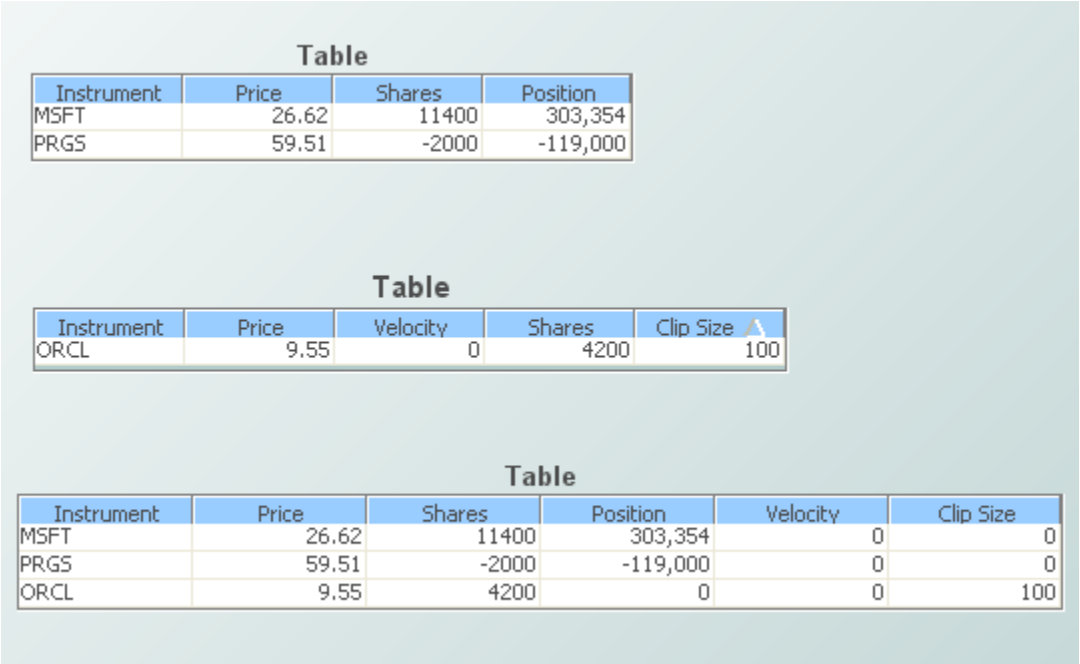
This function returns a table.

Example

The third table below is attached to the function defined by the following dialog. The first table's data table is attached to the argument Table 1, and the second table's data table is attached to the argument Table 1.

Figure 3. Combine example





Table

Instrument	Price	Shares	Position
MSFT	26.62	11400	303,354
PRGS	59.51	-2000	-119,000

Table

Instrument	Price	Velocity	Shares	Clip Size
ORCL	9.55	0	4200	100

Table

Instrument	Price	Shares	Position	Velocity	Clip Size
MSFT	26.62	11400	303,354	0	0
PRGS	59.51	-2000	-119,000	0	0
ORCL	9.55	4200	0	0	100

Tabular Functions

Concatenate Columns

Creates a string concatenation of the values in the given table columns separated by the given character(s), and returns the results in a new table column. The column names are specified as a semicolon-separated string. The separator can be a single character such as . or / but it can also be a string such as and.

Arguments

This function has the following arguments:

- Table: Table whose column values are to be concatenated
- Names of Columns to Concatenate: Columns whose values are to be concatenated
- Separating Character(s): Separator character, such as . or /, or separator string, such as and
- Result Column Name: Name of the result column

This function returns a table.

Tabular Functions

Convert Columns

Returns a copy of the specified table that is modified by converting the specified columns to the specified types.

Usage notes

When converting from numeric types (other than Long) to the Time type, columns are first converted to Long and then to Time. If a String column entry cannot be parsed as a Time, the resulting entry is blank.

Arguments

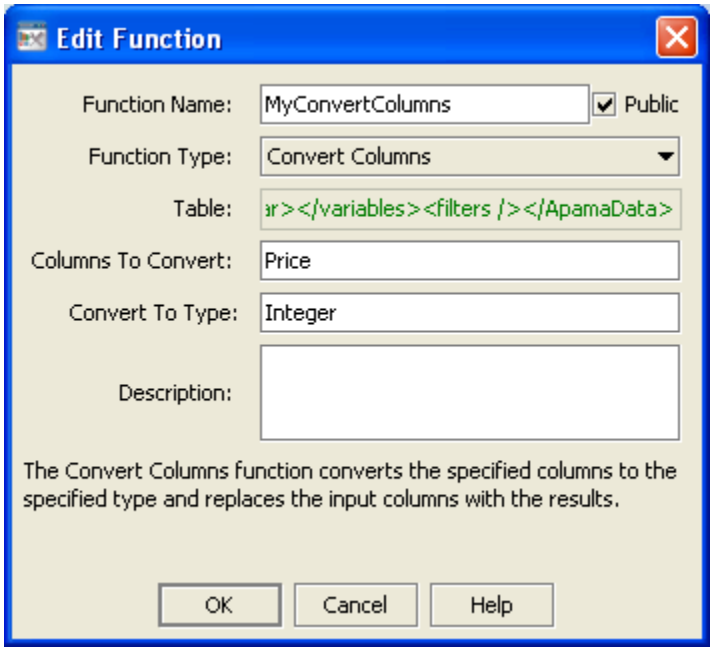
The function has the following arguments:

- Table: Table that contains the columns to convert.
- Columns To Convert: Text string that specifies the columns to convert. Supply a single column name or a semi-colon delimited list of column names.
- Columns To Type: Text string that specifies the target types of the conversion. Supply a single type name or a semi-colon delimited list of type names. Use the following type names: Boolean, Integer, Long, Float, Double, String, or Time. Type names may be abbreviated to the first letter.

This function returns a table.

The second table below is attached to the function defined by the following dialog. The first table's data table is attached to the argument Table.

Figure 4. Convert columns example



Table

Instrument	Price	Velocity	Shares	Position
MSFT	27.02	0.0125	0	0
PRGS	60.23	-0.0143	-2200	-132,506
ORCL	10.16	0	-800	-8,136

Table

Instrument	Price	Velocity	Shares	Position
MSFT	27	0.0125	0	0
PRGS	60	-0.0143	-2200	-132,506
ORCL	10	0	-800	-8,136

Tabular Functions

Copy

Copies the specified Table.

Arguments

The function has the following argument:

- Table: Table to be copied.

This function returns a table.

Tabular Functions

Count

Returns the number of rows in the specified table.

Arguments

The function has the following argument:

- Table Column: Table whose rows are to be counted.

This function returns a numeric value.

Tabular Functions

Count By Bands

Divides a specified range of values into a specified number of bands and counts the number of rows in a specified data table column that contain a value that lies within each band.

Usage notes

If Return Cumulative Percents is set to 0, this function returns a table containing one column that holds the midpoint values of each band (one row for each band), as well as one column containing the counts.

If Return Cumulative Percents is set to 1, the returned columns will contain the cumulative percentage of the total count in each cell, rather than the individual counts.

Arguments

The function has the following arguments:

- Table: Data table column.
- Number of Bands: Numerical value that specifies the number of bands into which to divide the specified range.
- Include Min/Max: Numerical value (0 for false and 1 for true) that determines whether the range of values is specified by Min Value and Max Value, or by the values in Table.
- Min Value: Numerical value that, together with Max Value, specifies the range of values that is divided into the bands, if Include Min/Max is 1.
- Max Value: Numerical value that, together with Min Value, specifies the range of values that is divided into the bands, if Include Min/Max is 1.
- Return Cumulative Percent: Numerical value (0 or 1) that determines whether counts or cumulative percentages are returned. If set to 1, he function returns the cumulative percentage of the total count in each cell, rather than the individual counts.

This function returns a table.

Tabular Functions

Count Unique Values

Returns a table that lists unique values and their counts from the specified table column.

Arguments

The function has the following arguments:

- Table Column: Data table column whose values are to be counted.
- Value List: Table column that specifies values for which a count is to be performed. If you do not supply this argument, counts are returned only for values present in Table Column. Use this argument to include rows in the returned table for values that are not always present in Table Column. If you specify this argument, the returned table includes a row for each specified value, even if the count for some values is 0.
- Restrict to Value List: Numerical value (0 or 1) that determines whether a count is performed *only* for values in Value List. If Restrict to Value List is set to 0, all unique values from the Table Column are included in the returned table. If Restrict to Value List is set to 1 and Value List is specified, only rows from the Value List are included.

- Use Column Names: Numerical value (0 or 1) that determines whether original column names are retained in the returned table. If Use Column Names is set to 1, then original column names are retained. If set to 0, columns are given generic names (for example, Subtotal1 and Total1). Generic column names are useful when the data attachment for the Table argument uses a substitution that causes the column names to change when the substitution changes.

This function returns a table.

Tabular Functions

Count Unique Values By Time

Returns a table that lists unique values and their counts from a specified table, sorted by a specified number of specified date part Intervals. The Table must contain a time column and a value column. The returned table contains an interval column, a column for each unique value, and counts for number of intervals specified or for all data in the Table if the Number of Intervals is 0.

Arguments

The function has the following arguments:

- Table: Data table column whose values are to be counted.
- Date Parts Per Interval: Number of date parts in each interval by which the counts are to be sorted.
- Number Of Intervals: Number of intervals by which the counts are to be sorted.
- Date Part: Text string that specifies the date unit to use. Enter s, m, h, d, w, M, q, or y, for seconds, minutes, hours, days, weeks, months, quarters, or years.
- Date Format: Text string that specifies the format of the function result. Specify a pattern string suitable for use with the Java `SimpleDateFormat` class.
- Value List: Table column that specifies values for which a count is to be performed. If you do not supply this argument, counts are returned only for values present in Table. Use this argument to include rows in the returned table for values that are not always present in Table. If you specify this argument, the returned table includes a row for each specified value, even if the count for some values is 0.
- Restrict to Value List: Numerical value (0 or 1) that determines whether a count is performed *only* for values in Value List. If Restrict to Value List is set to 0, all unique values from the Table Column are included in the returned table. If Restrict to Value List is set to 1 and Value List is specified, only rows from the Value List are included.
- Use Column Names: Numerical value (0 or 1) that determines whether original column names are retained in the returned table. If Use Column Names is set to 1, original column names are retained. If set to 0, columns are given generic names (for example, Subtotal1 and Total1). Generic column names are useful when the data attachment for the Table argument uses a substitution that causes the column names to change when the substitution changes.

This function returns a table.

Tabular Functions

Create Selector List

Returns a two column table containing selector names and their corresponding values to be presented in a dropdown list. The first column contains selector names and the second column contains their values. The returned table consists of the first two columns of a specified table, optionally modified by sorting, and with the optional addition of a row that contains a specified selector name in the first column and the value * in the second column. If the input table has only one column its contents are used for both the selector names and values.

Arguments

This function has the following arguments:

- Selector Table: Table whose data is to be presented in a dropdown list.
- All Selector Name: Text string that controls whether an initial row is added to the returned table that contains. If you specify a value, a row is added whose first column contains the specified value and whose second column contains the value *.
- Sort Values: Numeric value that controls whether the returned rows are sorted. Set this argument to 1 in order to sort the returned rows by selector name. The sort is numerical, if all the selector names are numbers; otherwise the sort is alphabetical.
- Sort Descending: Numeric value that controls whether returned rows are sorted in descending order, if the argument Sort Values is set to 1.Set Sort Descending to 1 in order to sort selector names in descending order. Set the argument to 0 (or leave it blank)) in order to sort selector names in ascending order.

This function returns a table.

Tabular Functions

Delta And Rate Rows

Returns a table that includes a rate-of-change column as wellas a delta column for each specified data column. The function returns a table including, for the specified columns, new values for the difference between this update and the previous, along with the rate of change per second. The new values may be appended to the input table in columns named by prefixing Delta and Rate to the column name, or the delta columns may replace the corresponding input columns (the rate columns will still be appended to the table).

This function has the following arguments:

- Table: Table of interest
- Delta Column Names: Nnames of one or more columns for which deltas will be calculated. At least one name must be given.
- Index Column Names: Names of one or more columns that uniquely identify a row in the table. If left blank, the default is to calculate deltas for all rows as if they had the same value. The values

contained in each index column are concatenated to form a unique index used to organize the resulting summary data..

- Time Column Name: Name of a timestamp column that will be used to calculate the rate of change. A name must be given. If the specified column is not found in the data it will be added, and its values will be taken from the current time on each update.
- Replace Data With Deltas: If set to 1, the delta values replace the original values in the same column in the returned table; otherwise they are in new columns appended to the table.
- Display Negative values: If set to 1, the delta values less than zero will be displayed with a negative sign and the value; otherwise they will be displayed as zero.

This function returns a table.

Tabular Functions

Delta Rows

Computes the delta between incoming rows of tabular data. This function returns a table that includes, for the specified columns, new values for the difference between the current update and the previous update.

Arguments

The function has the following arguments:

- Table: Table for which
- Delta Column Names: Text string that specifies the name of one or more columns for which deltas are to be calculated. Separate column names by a semicolon. This field cannot be left blank.
- Index Column Names: Text string that specifies the name of one or more columns that uniquely identify a row in the table. Separate column names by a semicolon. If left blank, the function calculates deltas for all rows as if they had the same value. The values contained in each index column are concatenated to form a unique index that is used to organize the resulting summary data.
- Replace Data With Deltas: Numerical value (0 or 1) that determines whether the returned table includes columns with the original values from Table. If set to 1, delta values replace original values in columns with the original names. If set to 0, new columns are added. The new columns use the original columns names with Delta prefixed.
- Display Negative Values: If set to 1, delta values less than zero are displayed with a negative sign. If set to 0, delta values less than zero are displayed as zero.

This function returns a table.

Tabular Functions

Distinct Values

Returns a table with a single column that lists all unique values from a specified column of a specified table.

Arguments

This function has the following arguments:

- Table: Table that contains the column whose unique values are to be returned
- Column Name: Name of the column whose unique values are to be returned
- Sort Values: Numeric value that controls whether the returned values are sorted. Set this to 1 in order to sort the values. Values are sorted in numerical order, if all values are numbers; otherwise they are sorted alphabetically.
- Sort Descending: Numeric value that controls whether the returned values are sorted in descending order, if Sort Values is set to 1. Set Sort Descending to 1 in order to sort the values in descending order. Set the argument to 0 (or leave it blank) in order to sort the values in ascending order.
- Use Column Name: Numeric value that controls the label of the returned column. Set this argument to 1 in order to use the original column name (the value of the Column Name argument). Set the argument to 0 (or leave it blank) in order to use the name Values. Use a generic name when you want a display that is independent of the value of Table, for example, because the value of Table uses a substitution that causes column names to change when the substitution changes.

This function returns a table.

Tabular Functions

Divide Columns

Returns a table that includes a column reflecting the quotient of specified columns of a specified table, the quotient of a specified value and a specified column, or the quotient of two specified values.

Usage notes

- Case 1: Quotient of specified columns of a specified table. This is the case if First Column Name or Numeric Value and Second Column Name or Numeric Value both specify a column of Table.
- Case 2: Quotient of a specified value and a specified column. This is the case if one of First Column Name or Numeric Value and Second Column Name or Numeric Value specifies a column of Table and the other specifies a numeric value.
- Case 3: Quotient of two specified values. This is the case if First Column Name or Numeric Value and Second Column Name or Numeric Value both specify a numeric value.

In case 1, the *n*th cell in the returned column corresponds to the *n*th cell of the table specified by the argument Table. Each cell of the returned column contains the result of dividing the corresponding row's cell in First Column Name or Numeric Value by the cell in Second Column Name or Numeric Value.

In case 2, the *n*th cell in the returned column corresponds to the *n*th cell of the table specified by the argument Table. Each cell of the returned column contains the result of dividing the corresponding row's cell in the specified column by the specified numeric value.

In case 3, each cell of the returned column contains the quotient of the two specified numeric values.

In the returned table, the quotient column is preceded by copies of all the columns in Table.

Arguments

The function has the following arguments:

- Table: Table that contains the columns to be divided.
- First Column Name or Numeric Value: Text string specifying the column to whose values are to be used as dividends, or numerical value to be as dividends.
- Second Column Name or Numeric Value: Text string specifying the column to whose values are to be used as divisors, or numerical value to be as divisors.
- Result Column Name: Text string that specifies the name of the column containing the quotients. You must supply a value for this argument.

This function returns a table.

Tabular Functions

Ensure Columns

Returns a copy of a specified table, modified where necessary to guarantee that given columns columns have specified types.

Arguments

The function has the following arguments:

- Table: The specified table.
- Column Name(s): Text string that specifies the columns to be modified if necessary. Supply a single column name or a semicolon-separated list of column names.
- Column Type(s): Text string that specifies the types. Supply a single name or a semicolon-separated list of names. The *n*th element of Column Type(s) is the type to be gauranteed for the column specified by the *n*th element of Column Name(s).
- Values: Semicolon-separated list of values to substitute when a value must be modified. The *n*th element of Values is used for the column named by the *n*th element of Column Name(s).

This function returns a table.

Tabular Functions

Ensure Timestamp Column

Returns a copy of a specified table, supplemented if necessary to include a timestamp column with a specified name. The added column is filled with the current time.

Arguments

The function has the following arguments:

- Table: The specified table.
- Column Name: Name to be used for the timestamp column, if one is added.
- Append Column: If set to 1, the timestamp column is appended to the end of the table; otherwise it is inserted as the first column.

This function returns a table.

Tabular Functions

Evaluate Expression By Row

Returns the result of evaluating a specified expression for each row of a specified table. The specified expression contains variables, each of which has an associated column of the specified table.

Usage notes

The returned table has all the columns of the specified table, followed by a column that contains the evaluation results. The *n*th row of the results column contains the result evaluating the expression with values from the *n*th row of the specified table substituted for expression variables.

Boolean true and false values are returned as 1.0 and 0.0 respectively.

Arguments

The function has the following arguments:

- Expression: Text string that specifies the expression to evaluate. Prefix variable names with%. Use standard arithmetic and logical operators. You can also use a variety of mathematical and string functions, as well as numeric and string constants. Enclose string constants in double quotes.
- Table: The table whose data is to be substituted into the Expression for evaluation.
- Result Column Name: Text string that specifies the name of the result column.
- Result Column Type: Text string that specifies the type of values in the result column. Specify either double or string (or the abbreviations d or s).
- Expression variable arguments: When the Expression field of the Edit Function dialog is activated (by pressing Enter or navigating to another field), the dialog displays a text field for each variable. For each field, enter a text string that names a column of Table. Column values are substituted for the corresponding variables in Expression. The types of the values are taken from the types of the columns. Numeric and boolean values are converted to double. Date columns are not supported.

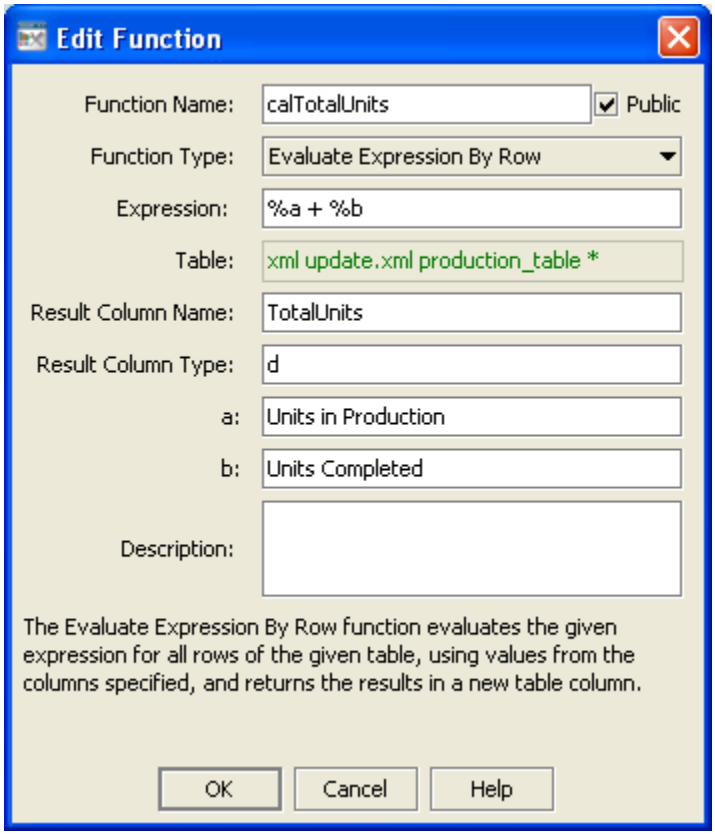
If a value whose form is numeric needs to be treated as a string, for example to serve as an argument to a string function, surround the variable in Expression with double quotes. Variables enclosed in double quotes are always used as strings. An example of such an expression is length("%var1") + %var2.

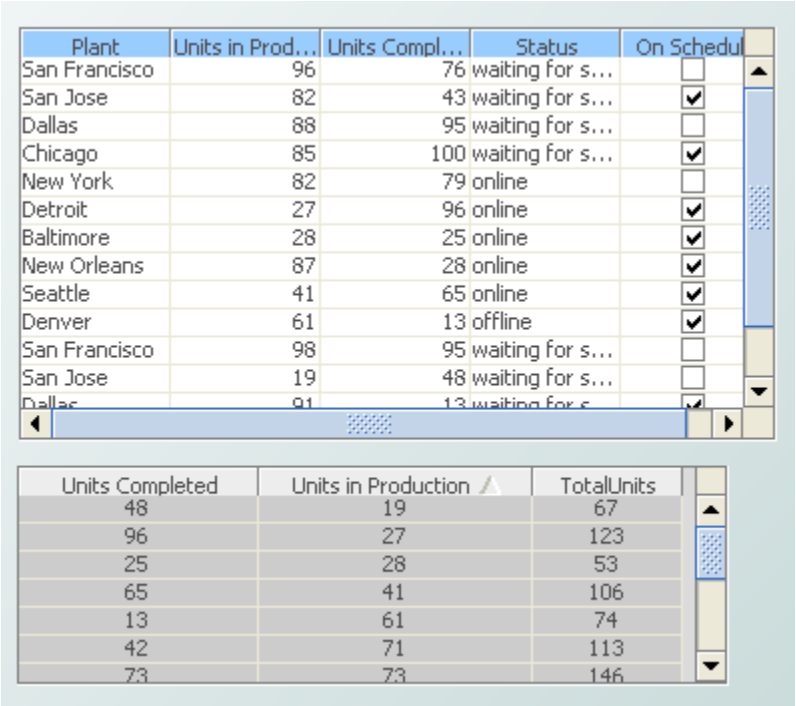
This function returns a text string.

Example

The second table below is attached to the function defined by the following dialog. The first table's data table is attached to the argument Table.

Figure 5. Evaluate Expression By Row example





Plant	Units in Prod...	Units Compl...	Status	On Schedul
San Francisco	96	76	waiting for s...	☐
San Jose	82	43	waiting for s...	☑
Dallas	88	95	waiting for s...	☐
Chicago	85	100	waiting for s...	☑
New York	82	79	online	☐
Detroit	27	96	online	☑
Baltimore	28	25	online	☑
New Orleans	87	28	online	☑
Seattle	41	65	online	☑
Denver	61	13	offline	☑
San Francisco	98	95	waiting for s...	☐
San Jose	19	48	waiting for s...	☐
Dallas	91	13	waiting for s...	☑

Units Completed	Units in Production	TotalUnits
48	19	67
96	27	123
25	28	53
65	41	106
13	61	74
42	71	113
73	73	146

Tabular Functions

Filter And Extract Matches

Returns a table containing all rows from a specified table in which the value of a specified column matches a specified pattern. For each matching row, each token from the specified column that matches a group in the pattern is extracted to a new column.

Arguments

This function has the following arguments:

- Table: Table from which matching rows are to be extracted.
- Filter Column Name: Text string specifying the column of Table to be searched for matches.
- Pattern: Text string that is either a simple string that optionally uses * as a wildcard, or a regular expression as described in the following Web page:

  See http://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html
- Pattern Is Reg Expr: Numerical value (0 or 1) that determines whether the pattern is interpreted as a simple string (that optionally uses * as a wildcard) or as a regular expression. If this argument is 0 (the default), Pattern is interpreted as a simple string. Otherwise, Pattern is interpreted as a regular expression.
- Number of New Columns: Numerical value that specifies the number of new columns to be added to the result table to contain the matching groups extracted from the filter column.
- New Column Names: Text string that specifies the name of each new column. Separate column names with a semicolon.

This function returns a table.

Example

Consider the following arguments:

- Table: Table that includes a Customer Name column.
- Filter Column Name: CustomerName.
- Pattern: * *
- Pattern Is Reg Expr: 0
- Number of New Columns: 2
- New Column Names: FirstName;LastName

In this case, if a row of Table contains Joe Smith in the CustomerName column, the corresponding row in the result table contains Joe in the FirstName column and Smith in the LastName column.

Tabular Functions

Filter By Pattern

Returns a table containing all rows from a specified table in which the value of a specified column matches a specified pattern.

Arguments

This function has the following arguments:

- Table: Table from which matching rows are to be extracted.
- Filter Column Name: Text string specifying the column of Table to be searched for matches.
- Pattern: Text string that is either a simple string that optionally uses * as a wildcard, or a regular expression as described in the following Web page:

  http://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html
- Pattern Is Reg Expr: Numerical value (0 or 1) that determines whether the pattern is interpreted as a simple string (that optionally uses * as a wildcard) or as a regular expression. If this argument is 0 (the default), Pattern is interpreted as a simple string. Otherwise, Pattern is interpreted as a regular expression.

This function returns a table.

Tabular Functions

Filter By Row

Returns a table containing all rows from a specified table in which the values of specified columns matches the values in specified lists of values.

Arguments

This function has the following arguments:

- Table: Table from which matching rows are to be extracted.
- Filter Column Name: Text string that specifies a list of column names, the columns of Table to be searched for matches. Separate column names with
- Filter Value: Text string that specifies a *list of lists* of values to be matched. Separate values with a coma. Separate lists with a semicolon (for example, val1,val2;val3,val4;val5,val6). Enter * for Filter Value in order to display all rows in the table. To use * as a literal comparative value, enclose it in single quotes. To use ; as a literal comparative value, enclose it in single quotes. If a filter value contains a space or a semicolon, enclose the entire value in single quotes.

A row from Table is included in the returned table if and only if the n^{th} Filter Column Name matches a value in FilterValue's n^{th} list, for all n from 1 to the number of specified column names.

This function returns a table.

Tabular Functions

Filter By Time Range

Returns a copy of a specified table that contains only those rows in which the value of a specified column falls within a specified time range.

Arguments

The function has the following arguments:

- Table: Table whose rows are to be copied.
- Date/Time Column Name: Text string that specifies a column of Table that contains a timestamp. If this argument is not supplied, the first column of Table is assumed to contain a timestamp.
- Time Range Start: Text string that specifies the start of the desired time range. If this argument is not supplied, the time range is unbounded at the lower end.
- Time Range End: Text string that specifies the end of the desired time range. The time range itself does not include this value, but does include a value that is one second less than Time Range End. If this argument is not supplied, the time range is unbounded at the upper end.

This function returns a table.

Tabular Functions

First Table Rows

Returns a table containing one of the following:

- First n rows of a specified table, for a specified number, n.

- First *n* rows for each unique combination of values in a specified set of columns in a specified table, for a specified number, *n*.

Arguments

This function has the following arguments:

- Table: Table some of whose rows are to be returned.
- Index Column Names: Text string that specifies the column or columns to be used to form indexes. Separate column names with a semicolon. If this argument is not supplied, the function returns the first Number of Rows of Table. If this argument is supplied, the function forms indexes by concatenating the values contained in each index column, and returns Number of Rows rows for each index.
- Number of Rows: Numerical value that specifies the number rows to be returned, or the number of rows with each index value to be returned.

The function returns a table.

Tabular Functions

Format Table Columns

Returns a copy of a specified table with specified formats applied to specified columns.

Arguments

This function has the following arguments:

- Table to Format: Table whose columns are to be formatted.
- Column Format(s): Text string that specifies the columns to format and the formats to use. The string consists of *column-name*:*column-format*. Separate pairs with a semicolon. Separate column name from column format with a colon. Enclose a column name in single quotes if it contains a space.

Specify the column format based on the Java format specification, or with the following shorthand: $ for money values, $$ for money values with additional formatting, or () for non-money values, formatted similar to money. For example, if Column Format(s) contains the pair 'Units Completed':$, the Units Completed column in the returned table is formatted for money. Both positive and negative formats can be supplied, for example: #,###;(#,###).

The function accepts date/time patterns for formatting columns of type Date. For example consider a column Timestamp with the value Sep 06, 2008 7:27:36 AM. If it is formatted with 'Timestamp':'MM/dd/yyyy' the result is 09/06 /2008. If it is formatted with 'Timestamp':'hh:mm:ss' the result is 07:27:36. For the full list of formatting codes, see the Java documentation for the class SimpleDateFormat.

The function returns a table.

Tabular Functions

Get Data Server Connection Status

Returns a table that contains status information for the Data Server being used by the current dashboard.

Columns

The table has the following columns:

- Name: `__default` for the default Data Server, or the name of a named Data Server
- Connected: `True` if the server connection is operational; `False` otherwise
- Status: `OK` if connection is operational, `no connection` if there is no connection to the server, or `no service` if there is an HTTP connection to the `rtvdata` servlet but the servlet has no connection to its Data Server
- ConnectionString: URL for an HTTP connection to the `rtvdata` servlet or *hostname*:*port* for a direct socket connection to a Data Server
- ReceiveCount: Number of data transmissions (pushes) received from the server
- ReceiveTime: Time of the most recent data transmission from the server
- Config: Srting that idenitifes Data Server version

Arguments

The function has no arguments.

This function returns a table.

Tabular Functions

Group By Time

Returns a table that contains a summary of all the data in a given table, aggregated over time. The summary data in the returned table is grouped into a specified number of specified time intervals over a specified time range.

Arguments

The function has the following arguments:

- Table: Table whose data is to be summarized.
- Group Type: Text string that specifies the type of aggregation to perform. Enter one or more of the following: sum, count, average, min, max. The default is sum. For multiple group types, use a semicolon-separated list and set Use Column Names to 0.
- Date/Time Column Name: Text string that specifies a column of Table that contains a timestamp. If this argument is not supplied, the first column of Table is assumed to contain a timestamp.
- Date Part: Text string that specifies the date unit to use. Enter s, m, h, d, w, M, q, or y, for seconds, minutes, hours, days, weeks, months, quarters, or years. The default unit is seconds.
- Date Parts Per Interval: Numerical value that specifies he size of each interval in Date Parts.

- Number Of Intervals: Numerical value that specifies the number of intervals to include in the summary table. The returned table contains one row for each interval. If set to 0, the number of intervals is determined from the range of data in Table, or by the specified time range.
- Time Range Start: Text string that specifies the start of the desired time range. If this argument is not supplied, the time range is unbounded at the lower end.
- Time Range End: Text string that specifies the end of the desired time range. The time range itself does not include this value, but does include a value that is one second less than Time Range End. If this argument is not supplied, the time range is unbounded at the upper end.
- Restrict To Time Range: Numerical value (0 or 1) that determines whether Time Range Start and Time Range End are ignored. If set to 1, the resulting summary table includes only those time intervals within the specified range. If set to 0, the specified time range is ignored.
- Use Column Names: Numerical value (0 or 1) that determines whether original column names are retained in the returned table. If Use Column Names is set to 1, original column names are retained. If set to 0, columns are given generic names. Set this to 0 if you specify multiple groups types.

This function returns a table.

Tabular Functions

Group By Time and Unique Values

Returns a table that contains a summary of all the data in a given table, aggregated over both time and index columns. The summary data in the returned table is grouped into specified time intervals, with a further breakdown by unique values in specified index columns.

Arguments

The function has the following arguments:

- Table: Table whose data is to be summarized.
- Group Type: Text string that specifies the type of aggregation to perform. Enter one or more of the following: sum, count, average, min, max. The default is sum. For multiple group types, use a semicolon-separated list and set Use Column Names to 0. The default is sum.
- Date/Time Column Name: Text string that specifies a column of Table that contains a timestamp. If this argument is not supplied, the first column of Table is assumed to contain a timestamp.
- Date Part: Text string that specifies the date unit to use. Enter s, m, h, d, w, M, q, or y, for seconds, minutes, hours, days, weeks, months, quarters, or years. The default unit is seconds.
- Date Parts Per Interval: Numerical value that specifies he size of each interval in Date Parts.
- Number Of Intervals: Numerical value that specifies the number of intervals to include in the summary table. The returned table contains one row for each interval. If set to 0, the number of intervals is determined from the range of data in Table, or by the specified time range.
- Time Range Start: Text string that specifies the start of the desired time range. If this argument is not supplied, the time range is unbounded at the lower end.

- Time Range End: Text string that specifies the end of the desired time range. The time range itself does not include this value, but does include a value that is one second less than Time Range End. If this argument is not supplied, the time range is unbounded at the upper end.
- Restrict To Time Range: Numerical value (0 or 1) that determines whether Time Range Start and Time Range End are ignored. If set to 1, the resulting summary table includes only those time intervals within the specified range. If set to 0, the specified time range is ignored.
- Index Column Names: Text string that specifies the column or columns to be used to form indexes. Separate column names with a semicolon. If this argument is not supplied, the function aggregates only by time interval. If this argument is supplied, the function forms indexes by concatenating the values contained in each index column, and aggregates by index value within time-interval aggregations.
- Value List: Table column that contains a set of values to be included in the set of values for the first index column. This is useful if you want the summary table to include values that may or may not be in the Table data.
- Restrict To Value List: Numerical value (0 or 1) that determines whether the table includes only rows that include the values of Value List. If set to 1, only such values are included.
- Use Column Names: Numerical value (0 or 1) that determines whether original column names are retained in the returned table. If Use Column Names is set to 1, original column names are retained. If set to 0, columns are given generic names. Set this to 0 if multiple group types are specified.
- Restrict To Data Combinations: Numerical value (0 or 1) that determines whether the returned table is restricted to only those combinations of values found in the specified index columns that occur in the data. If set to 0, the returned table contains all possible combinations of unique values found in the specified index columns.

This function returns a table.

Tabular Functions

Group by Unique Values

Returns a table that contains a summary of all the data in a given table. The summary data in the returned table is grouped by unique values in specified index columns.

Arguments

The function has the following arguments:

- Table: Table whose data is to be summarized.
- Group Type: Text string that specifies the type of aggregation to perform. Enter one or more of the following: sum, count, average, min, max. The default is sum. For multiple group types, use a semicolon-separated list and set Use Column Names to 0..
- Index Column Names: Text string that specifies the column or columns to be used to form indexes. Separate column names with a semicolon. If this argument is not supplied, the function uses the first column of Table as the index column. If this argument is supplied, the function forms indexes by concatenating the values contained in each index column, and aggregates by index value.

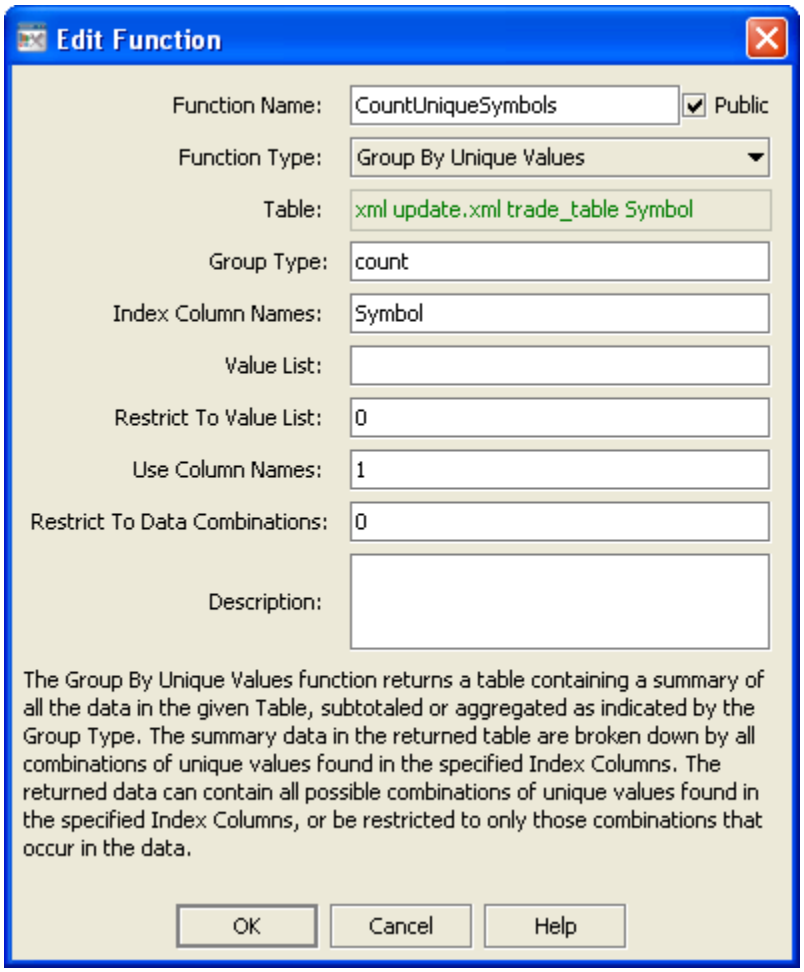
- Value List: Table column that contains a set of values to be included in the set of values for the first index column. This is useful if you want the summary table to include values that may or may not be in the Table data.
- Restrict To Value List: Numerical value (0 or 1) that determines whether the table includes only rows that include the values of Value List. If set to 1, only such values are included.
- Use Column Names: Numerical value (0 or 1) that determines whether original column names are retained in the returned table. If Use Column Names is set to 1, original column names are retained. If set to 0, columns are given generic names.
- Restrict To Data Combinations: Numerical value (0 or 1) that determines whether the returned table is restricted to only those combinations of values found in the specified index columns that occur in the data. If set to 0, the returned table contains all possible combinations of unique values found in the specified index columns. Set this to 0 if multiple group types are specified.

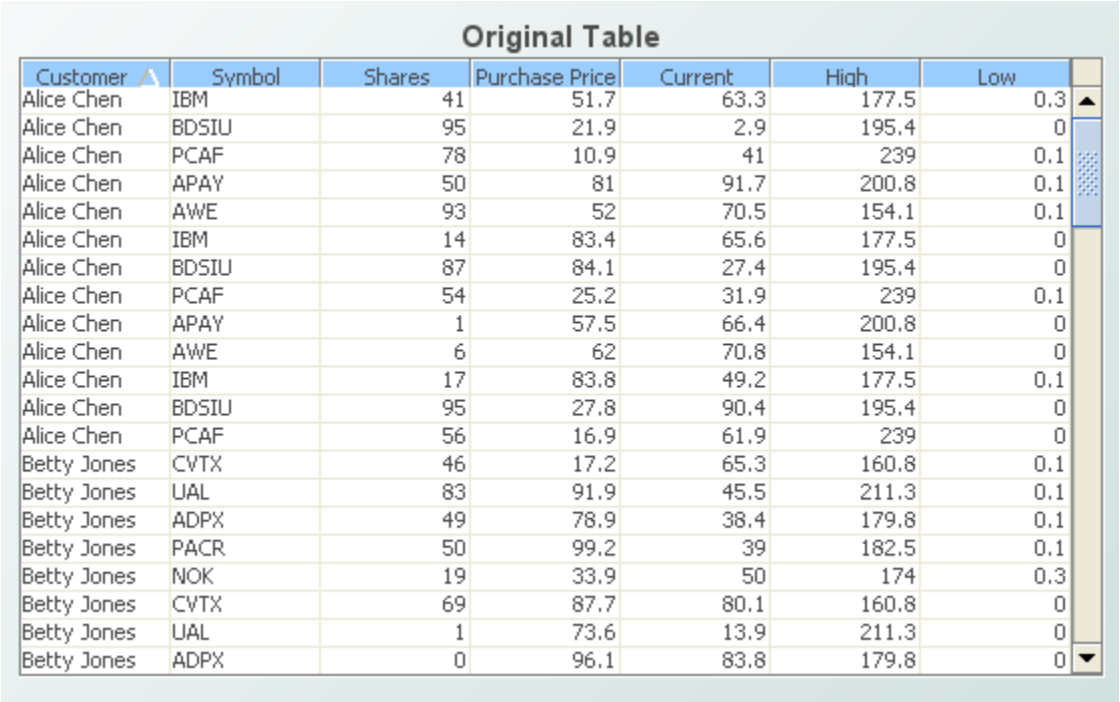
This function returns a table.

Examples

The pie graph and the second table below are attached to the function defined by the following dialog. The first table's data table is attached to the argument Table.

Figure 6. Group By Unique Values example 1



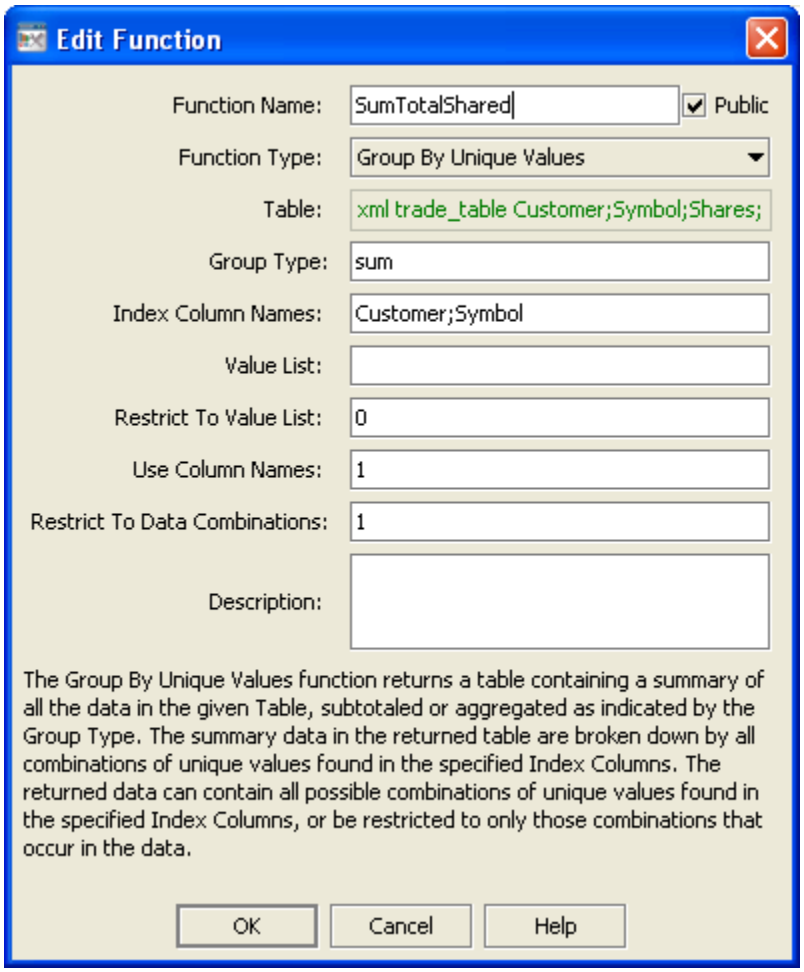


Original Table

Customer	Symbol	Shares	Purchase Price	Current	High	Low
Alice Chen	IBM	41	51.7	63.3	177.5	0.3
Alice Chen	BDSIU	95	21.9	2.9	195.4	0
Alice Chen	PCAF	78	10.9	41	239	0.1
Alice Chen	APAY	50	81	91.7	200.8	0.1
Alice Chen	AWE	93	52	70.5	154.1	0.1
Alice Chen	IBM	14	83.4	65.6	177.5	0
Alice Chen	BDSIU	87	84.1	27.4	195.4	0
Alice Chen	PCAF	54	25.2	31.9	239	0.1
Alice Chen	APAY	1	57.5	66.4	200.8	0
Alice Chen	AWE	6	62	70.8	154.1	0
Alice Chen	IBM	17	83.8	49.2	177.5	0.1
Alice Chen	BDSIU	95	27.8	90.4	195.4	0
Alice Chen	PCAF	56	16.9	61.9	239	0
Betty Jones	CVTX	46	17.2	65.3	160.8	0.1
Betty Jones	UAL	83	91.9	45.5	211.3	0.1
Betty Jones	ADPX	49	78.9	38.4	179.8	0.1
Betty Jones	PACR	50	99.2	39	182.5	0.1
Betty Jones	NOK	19	33.9	50	174	0.3
Betty Jones	CVTX	69	87.7	80.1	160.8	0
Betty Jones	UAL	1	73.6	13.9	211.3	0
Betty Jones	ADPX	0	96.1	83.8	179.8	0

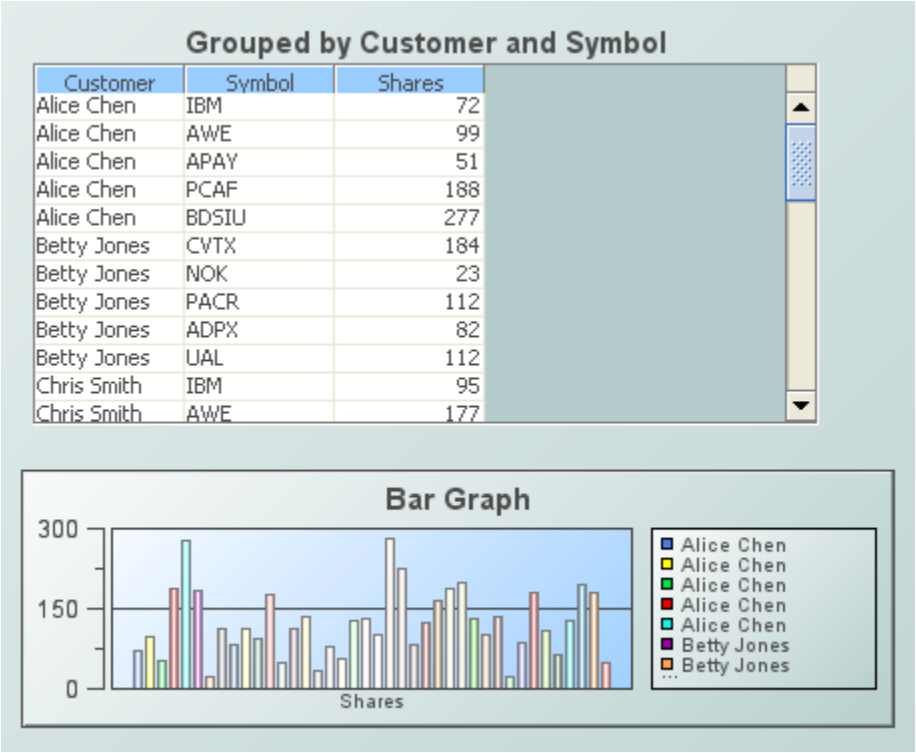
The bar graph and the second table below are attached to the function defined by the following dialog. The first table's data table is attached to the argument Table.

Figure 7. Group By Unique Values example 2



Original Table

Customer	Symbol	Shares	Purchase Price	Current	High	Low
Alice Chen	IBM	41	51.7	63.3	177.5	0.3
Alice Chen	BDSIU	95	21.9	2.9	195.4	0
Alice Chen	PCAF	78	10.9	41	239	0.1
Alice Chen	APAY	50	81	91.7	200.8	0.1
Alice Chen	AWE	93	52	70.5	154.1	0.1
Alice Chen	IBM	14	83.4	65.6	177.5	0
Alice Chen	BDSIU	87	84.1	27.4	195.4	0
Alice Chen	PCAF	54	25.2	31.9	239	0.1
Alice Chen	APAY	1	57.5	66.4	200.8	0
Alice Chen	AWE	6	62	70.8	154.1	0
Alice Chen	IBM	17	83.8	49.2	177.5	0.1
Alice Chen	BDSIU	95	27.8	90.4	195.4	0
Alice Chen	PCAF	56	16.9	61.9	239	0
Betty Jones	CVTX	46	17.2	65.3	160.8	0.1
Betty Jones	UAL	83	91.9	45.5	211.3	0.1
Betty Jones	ADPX	49	78.9	38.4	179.8	0.1
Betty Jones	PACR	50	99.2	39	182.5	0.1
Betty Jones	NOK	19	33.9	50	174	0.3
Betty Jones	CVTX	69	87.7	80.1	160.8	0
Betty Jones	UAL	1	73.6	13.9	211.3	0
Betty Jones	ADPX	0	96.1	83.8	179.8	0



Tabular Functions

Join

Returns the result of performing an inner join of two specified tables on specified columns. The result contains all columns from Left Table followed by all columns from Right Table, and contains all rows for which the value in Left Column exactly matches the value in Right Column. Left Column Name and Right Column Name can each specify a semicolon-separated list of *n* column names, in which case a match occurs if the *i*th value in Left Column Name exactly matches the *i*th value in Right Column Name, for all *i* between 1 and *n*, inclusive.

Arguments

The function has the following arguments:

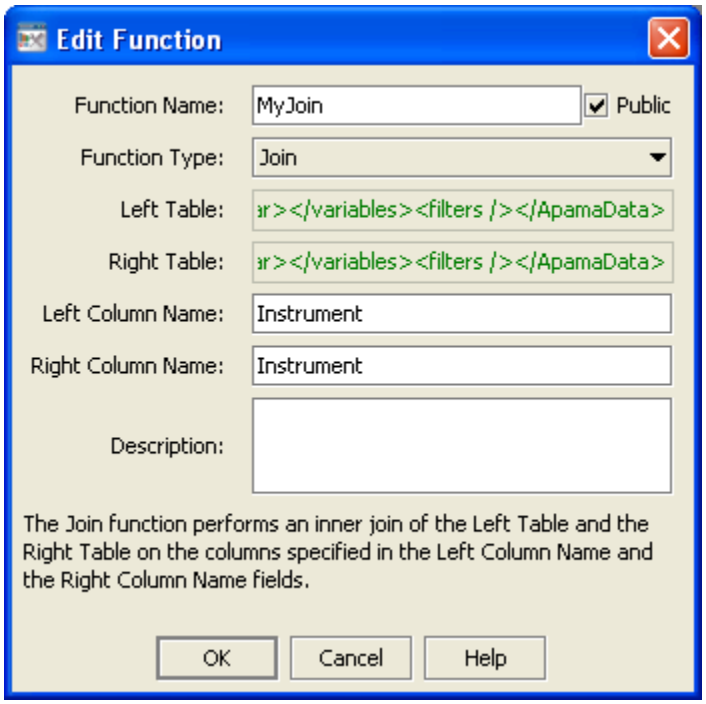
- Left Table: Table on which the join is to be performed.
- Right Table: Table on which the join is to be performed.
- Left Column Name: Text string that specifies the column or columns from Left Table on which the join is to be performed. If left blank, the row name, up to the first colon (if it contains a colon), is used instead of a column value.
- Right Column Name: Text string that specifies the column or columns from Right Table on which the join is to be performed. If left blank, the row name, up to the first colon (if it contains a colon), is used instead of a column value.

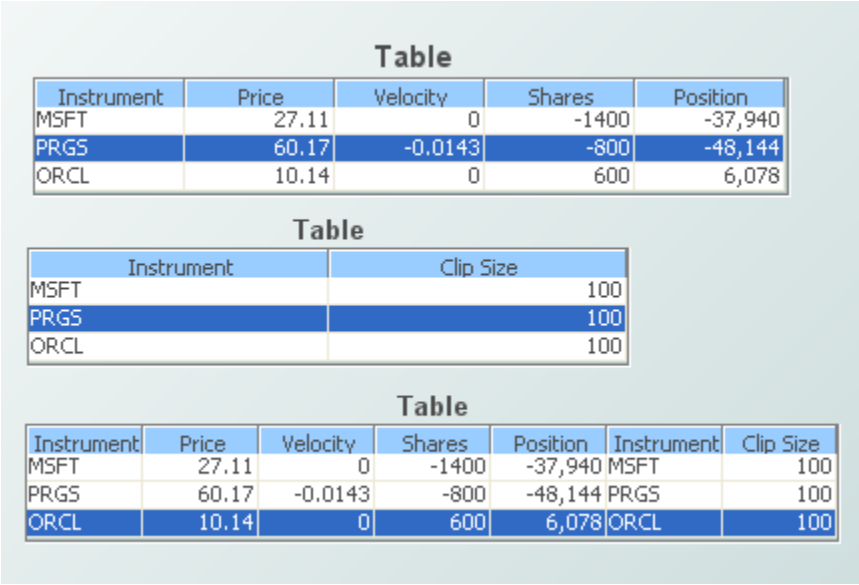
This function returns a table.

Example

The third table below is attached to the function defined by the following dialog. The first table's data table is attached to the argument Left Table and the second table's data table is attached to the argument Right Table.

Figure 8. Join Function example





Table

Instrument	Price	Velocity	Shares	Position
MSFT	27.11	0	-1400	-37,940
PRGS	60.17	-0.0143	-800	-48,144
ORCL	10.14	0	600	6,078

Table

Instrument	Clip Size
MSFT	100
PRGS	100
ORCL	100

Table

Instrument	Price	Velocity	Shares	Position	Instrument	Clip Size
MSFT	27.11	0	-1400	-37,940	MSFT	100
PRGS	60.17	-0.0143	-800	-48,144	PRGS	100
ORCL	10.14	0	600	6,078	ORCL	100

Tabular Functions

Join Outer

Performs an outer join of the Left Table and the Right Table on the columns specified in the Left Column Name and the Right Column Name fields. The joined table contains all columns from the Left Table followed by all columns from the Right Table, and contains all rows where the value in the Left Column exactly matches the value in the Right Column, plus additional rows according to the Outer Join Type, which may be left, right, or full.

Usage notes

In a left outer join, the result table includes all the rows from the left table; in a right outer join it includes all the rows from the right table, and in a full outer join it includes all the rows from both tables. In any row where there is no match for the join column value, the cells from the other table contain null values. (Null values are represented as blank for strings, 0 for integers and longs, `NaN` for floats and doubles, and `NULL_DATE` for dates.)

Left Column Name and Right Column Name can each specify a semicolon-separated list of *n* column names, in which case a match occurs if the *i*th value in Left Column Name exactly matches the *i*th value in Right Column Name, for all *i* between 1 and *n*, inclusive.

For a full join or right join, if the Left Table is null, the result is Right Table. For a full join or left join, if Right Table is null, the result is Left Table. In all other cases the result is null.

Arguments

The function has the following fields:

- Left Table: The first table to be joined.
- Right Table: The second table to be joined.
- Left Column Name: (Optional) The column or columns in the left table to be joined with the column or columns specified in the Right Column Name field. If this field is left blank, the row name, up to the first : if it contains a :, is used instead of a column value.
- Right Column Name: (Optional) The column or columns in the right table to be joined with the column or columns specified in the Left Column Name field. If this field is left blank, the row name, up to the first : if it contains a :, is used instead of a column value.
- Outer Join Type: Specified as left, right, or full, which may be abbreviated to their first letters. If this field is left blank a full outer join is performed.

This function returns a table.

Tabular Functions

Last Table Rows

Returns a table containing one of the following:

- Last *n* rows of a specified table, for a specified number, *n*.
- Last *n* rows for each unique combination of values in a specified set of columns in a specified table, for a specified number, *n*.

Arguments

This function has the following arguments:

- Table: Table some of whose rows are to be returned.
- Index Column Names: Text string that specifies the column or columns to be used to form indexes. Separate column names with a semicolon. If this argument is not supplied, the function returns the last Number of Rows of Table. If this argument is supplied, the function forms indexes by concatenating the values contained in each index column, and returns Number of Rows rows for each index.
- Number of Rows: Numerical value that specifies the number rows to be returned, or the number of rows with each index value to be returned.

The function returns a table.

Tabular Functions

Mark Time Gaps

Returns a table that results from supplementing a given trend table with special rows that indicate longer-than-expected time intervals between timestamps of adjacent rows of the given trend table. For a trend graph attached to the returned table, the trend line will contain a break (or a specified characater) wherever time gaps occurred.

Usage Notes

The table is constructed as follows: If the time interval between any two rows in the table is greater than the expected interval, insert 2 new rows between those rows in which the value of each column to be marked is set to NaN or other specified value. (NaN indicates "not a number". A trend graph will break a trace line when a NaN is encountered). The timestamp of the first new row is set to a value of 1 msec more than the timestamp of the last row before the graph and the timestamp of the second new row is set to a value of 1 msec less than the timestamp of the next row after the gap. It is assumed that the table is sorted by timestamp in ascending order. On the second and subsequent updates of this function, the timestamp of the first row in the table is compared to the timestamp of the last row from the previous update.

Arguments

This function has the following arguments:

- Table: Table to be checked for time gaps. The table must have a timestamp column and must be sorted by timestamp in ascending order.
- Name of Timestamp Column: Name of the table's timestamp column
- Expected Interval: The maximum time interval that should occur between consecutive rows in the table. If this interval is exceeded, it is considered a gap. Specify the interval in seconds or specify a value followed by m, h, d, for minutes, hours, or days.
- Names of Columns to Mark: Names of the columns to be marked with the specified value when rows are added to mark a gap. Separate multiple column names with semicolons. If no column names are specified then all columns with floating point values will be marked.

- Mark Columns Width: The value to be assigned when marking columns in the rows added to mark a gap. The default is NaN, but any numeric value can be used.

The function returns a table.

Tabular Functions

Max All Rows or Columns

Determines the maximum cell value for each row or column of the specified Table.

Usage notes

If Return Column is 1, the function returns a table with one column. The *n*th cell of the returned column contains the maximum of the numerical cells in the *n*th row of the table specified by the argument Table. (If there are no numerical cells in the row, the returned cell contains 0.)

If Return Column is 0, the function returns a table with one row. The *n*th cell of the returned row contains the maximum of the numerical cells in the *n*th column of the table specified by the argument Table. (If there are no numerical cells in the column, the returned cell contains N/A, by default—but see the argument Result Label Column, below.) The *n*th column of the returned one-row table is labeled with the label of the *n*th column of the table specified by the argument Table.

Arguments

This function has the following arguments:

- Table: Table for which row or column maximums are to be determined.
- Return Column: Numeric value that controls whether to return a column or a row of result values. To get a column of result values, one value for each row, set Return Column to 1. To get a row of result values, one value for each column, set Return Column to 0.
- Result Label: Text string that specifies a label for the result row or column. If not specified, the label text is Maximum. If Return Column is 0, the label appears only if Result Label Column is set to a column of Table that has no numeric values. If Return Column is 1, the label text always appears and Result Label Column is ignored.
- Result Label Column: Text string that specifies the column in which Result Label appears, if Return Column is 0. The specified column must have no numeric values in order for the label to appear. If Return Column is 1, this argument is ignored.

This function returns a table.

Tabular Functions

Max Columns

Returns a table that includes a column reflecting the larger of two specified columns of a specified table, the larger of a specified value and a specified column, or the larger of two specified values.

Usage notes

- Case 1: Larger of two specified columns of a specified table. This is the case if First Column Name or Numeric Value and Second Column Name or Numeric Value both specify a column of Table.
- Case 2: Larger of a specified value and a specified column. This is the case if one of First Column Name or Numeric Value and Second Column Name or Numeric Value specifies a column of Table and the other specifies a numeric value.
- Case 3: Larger of two specified values. This is the case if First Column Name or Numeric Value and Second Column Name or Numeric Value both specify a numeric value.

In case 1, the *n*th cell in the returned column corresponds to the *n*th cell of the table specified by the argument Table. Each cell of the returned column contains the larger of the corresponding row's cell in First Column Name or Numeric Value and the cell in Second Column Name or Numeric Value.

In case 2, the *n*th cell in the returned column corresponds to the *n*th cell of the table specified by the argument Table. Each cell of the returned column contains the larger of the corresponding row's cell in the specified column and the specified numeric value.

In case 3, each cell of the returned column contains the larger of the two specified numeric values.

In the returned table, the column that reflects the larger value is preceded by copies of all the columns in Table.

Arguments

The function has the following arguments:

- Table: Table that contains the columns to be compared.
- First Column Name or Numeric Value: Text string specifying the first column to be compared, or numerical value to be compared.
- Second Column Name or Numeric Value: Text string specifying the second column to be compared, or numerical value to be compared.
- Result Column Name: Text string that specifies the name of the column containing the maximums. You must supply a value for this argument.

This function returns a table.

Tabular Functions

Min All Rows or Columns

Determines the minimum cell value for each row or column of the specified Table.

Usage notes

If Return Column is 1, the function returns a table with one column. The *n*th cell of the returned column contains the minimum of the numerical cells in the *n*th row of the table specified by the argument Table. (If there are no numerical cells in the row, the returned cell contains 0.)

If Return Column is 0, the function returns a table with one row. The *n*th cell of the returned row contains the minimum of the numerical cells in the *n*th column of the table specified by the argument Table. (If there are no numerical cells in the column, the returned cell contains N/A, by default—

but see the argument Result Label Column, below.) The *n*th column of the returned one-row table is labeled with the label of the *n*th column of the table specified by the argument Table.

Arguments

This function has the following arguments:

- Table: Table for which row or column minimums are to be determined.
- Return Column: Numeric value that controls whether to return a column or a row of result values. To get a column of result values, one value for each row, set Return Column to 1. To get a row of result values, one value for each column, set Return Column to 0.
- Result Label: Text string that specifies a label for the result row or column. If not specified, the label text is Minimum. If Return Column is 0, the label appears only if Result Label Column is set to a column of Table that has no numeric values. If Return Column is 1, the label text always appears and Result Label Column is ignored.
- Result Label Column: Text string that specifies the column in which Result Label appears, if Return Column is 0. The specified column must have no numeric values in order for the label to appear. If Return Column is 1, this argument is ignored.

This function returns a table.

Tabular Functions

Min Columns

Returns a table that includes a column reflecting one of the smaller of two specified columns of a specified table, the smaller of a specified value and a specified column, or the smaller of two specified values.

Usage notes

- Case 1: Smaller of two specified columns of a specified table. This is the case if First Column Name or Numeric Value and Second Column Name or Numeric Value both specify a column of Table.
- Case 2: Smaller of a specified value and a specified column. This is the case if one of First Column Name or Numeric Value and Second Column Name or Numeric Value specifies a column of Table and the other specifies a numeric value.
- Case 3: Smaller of two specified values. This is the case if First Column Name or Numeric Value and Second Column Name or Numeric Value both specify a numeric value.

In case 1, the *n*th cell in the returned column corresponds to the *n*th cell of the table specified by the argument Table. Each cell of the returned column contains the smaller of the corresponding row's cell in First Column Name or Numeric Value and the cell in Second Column Name or Numeric Value.

In case 2, the *n*th cell in the returned column corresponds to the *n*th cell of the table specified by the argument Table. Each cell of the returned column contains the smaller of the corresponding row's cell in the specified column and the specified numeric value.

In case 3, each cell of the returned column contains the smaller of the two specified numeric values.

In the returned table, the column that reflects the smaller value is preceded by copies of all the columns in Table.

Arguments

The function has the following arguments:

- Table: Table that contains the columns to be compared.
- First Column Name or Numeric Value: Text string specifying the first column to be compared, or numerical value to be compared.
- Second Column Name or Numeric Value: Text string specifying the second column to be compared, or numerical value to be compared.
- Result Column Name: Text string that specifies the name of the column containing the minimums. You must supply a value for this argument.

This function returns a table.

Tabular Functions

Modulo Columns

Returns a table that includes a column reflecting the remainder of division performed on specified columns of a specified table, the remainder of division performed on a specified value and a specified column, or the remainder of division performed on two specified values.

Usage notes

- Case 1: Remainder of division performed on specified columns of a specified table. This is the case if First Column Name or Numeric Value and Second Column Name or Numeric Value both specify a column of Table.
- Case 2: Remainder of division performed on a specified value and a specified column. This is the case if one of First Column Name or Numeric Value and Second Column Name or Numeric Value specifies a column of Table and the other specifies a numeric value.
- Case 3: Remainder of division performed on two specified values. This is the case if First Column Name or Numeric Value and Second Column Name or Numeric Value both specify a numeric value.

In case 1, the *n*th cell in the returned column corresponds to the *n*th cell of the table specified by the argument Table. Each cell of the returned column contains the remainder of dividing the corresponding row's cell in First Column Name or Numeric Value by the cell in Second Column Name or Numeric Value.

In case 2, the *n*th cell in the returned column corresponds to the *n*th cell of the table specified by the argument Table. Each cell of the returned column contains the remainder of dividing the corresponding row's cell in the specified column by the specified numeric value.

In case 3, each cell of the returned column contains the remainder of dividing the first value by the second.

In the returned table, the remainder column is preceded by copies of all the columns in Table.

Arguments

The function has the following arguments:

- Table: Table that contains the columns to be divided.
- First Column Name or Numeric Value: Text string specifying the column to whose values are to be used as dividends, or numerical value to be as dividends.
- Second Column Name or Numeric Value: Text string specifying the column to whose values are to be used as divisors, or numerical value to be as divisors.
- Result Column Name: Text string that specifies the name of the column containing the remainders. You must supply a value for this argument.

This function returns a table.

Tabular Functions

Multiply Columns

Returns a table that includes a column reflecting the product of two specified columns of a specified table, the product of a specified value and a specified column, or the product of two specified values.

Usage notes

- Case 1: Product of two specified columns of a specified table. This is the case if First Column Name or Numeric Value and Second Column Name or Numeric Value both specify a column of Table.
- Case 2: Product of a specified value and a specified column. This is the case if one of First Column Name or Numeric Value and Second Column Name or Numeric Value specifies a column of Table and the other specifies a numeric value.
- Case 3: Product of two specified values. This is the case if First Column Name or Numeric Value and Second Column Name or Numeric Value both specify a numeric value.

In case 1, the *n*th cell in the returned column corresponds to the *n*th cell of the table specified by the argument Table. Each cell of the returned column contains the result of multiplying the corresponding row's cell in First Column Name or Numeric Value by the cell in Second Column Name or Numeric Value.

In case 2, the *n*th cell in the returned column corresponds to the *n*th cell of the table specified by the argument Table. Each cell of the returned column contains the result of multiplying the corresponding row's cell in the specified column by the specified numeric value.

In case 3, each cell of the returned column contains the product of the two specified numeric values.

In the returned table, the product column is preceded by copies of all the columns in Table.

Arguments

The function has the following arguments:

- Table: Table that contains the columns to be summed.
- First Column Name or Numeric Value: Text string specifying the first column to be included in the product, or numerical value to be included in the product.
- Second Column Name or Numeric Value: Text string specifying the second column to be included in the product, or numerical value to be included in the product.

- Result Column Name: Text string that specifies the name of the column containing the products. You must supply a value for this argument.

This function returns a table.

Tabular Functions

Percent Columns

Returns a table that includes a column reflecting the quotient of specified columns of a specified table, the quotient of a specified value and a specified column, or the quotient of two specified values. Values are expressed as percentages.

Usage notes

- Case 1: Quotient of specified columns of a specified table, expressed as a percentage. This is the case if First Column Name or Numeric Value and Second Column Name or Numeric Value both specify a column of Table.
- Case 2: Quotient of a specified value and a specified column, expressed as a percentage. This is the case if one of First Column Name or Numeric Value and Second Column Name or Numeric Value specifies a column of Table and the other specifies a numeric value.
- Case 3: Quotient of two specified values, expressed as a percentage. This is the case if First Column Name or Numeric Value and Second Column Name or Numeric Value both specify a numeric value.

In case 1, the *n*th cell in the returned column corresponds to the *n*th cell of the table specified by the argument Table. Each cell of the returned column contains the result, expressed as a percentage, of dividing the corresponding row's cell in First Column Name or Numeric Value by the cell in Second Column Name or Numeric Value.

In case 2, the *n*th cell in the returned column corresponds to the *n*th cell of the table specified by the argument Table. Each cell of the returned column contains the result, expressed as a percentage, of dividing the corresponding row's cell in the specified column by the specified numeric value.

In case 3, each cell of the returned column contains the quotient, expressed as a percentage, of the two specified numeric values.

In the returned table, the percent column is preceded by copies of all the columns in Table.

Arguments

The function has the following arguments:

- Table: Table that contains the columns to be divided.
- First Column Name or Numeric Value: Text string specifying the column to whose values are to be used as dividends, or numerical value to be as dividends.
- Second Column Name or Numeric Value: Text string specifying the column to whose values are to be used as divisors, or numerical value to be as divisors.
- Result Column Name: Text string that specifies the name of the column containing the percentages. You must supply a value for this argument.

This function returns a table.

Pivot On Unique Values

Returns a table in which row data from a specified table is rotated into columns.

Arguments

The function has the following arguments:

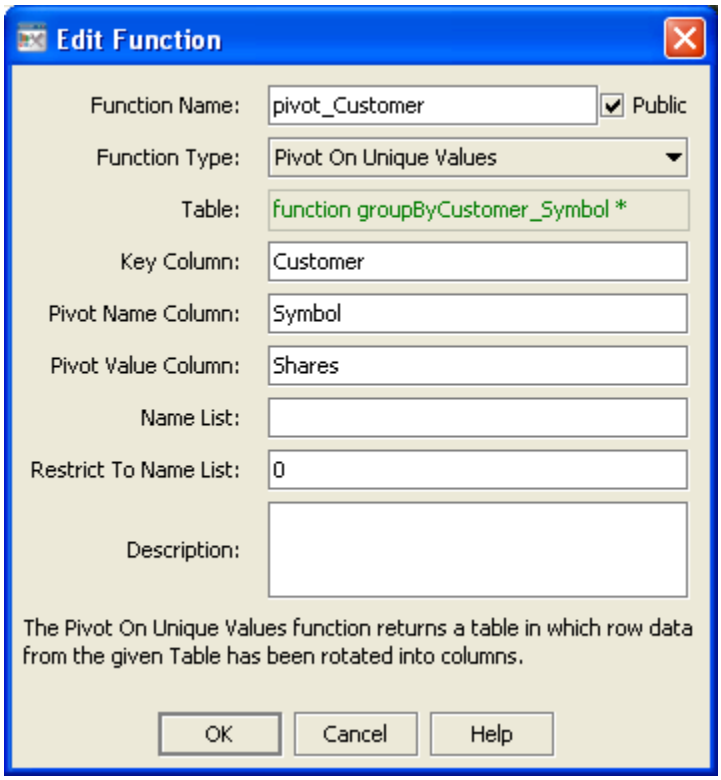
- Pivot Name Column: Text string that specifies the column containing values that become new column names in the returned table.
- Key Column: Text string that specifies the column used to group rows containing unique names in Pivot Name Column into a single row.
- Pivot Value Column: Text string that specifies the column containing the data of interest. All consecutive rows that contain the same value in Key Column have the data in the Pivot Value Column subtotaled into the same row of the resulting table, in the appropriate column.
- Name List: Text string that specifies values for which columns should be included in the returned table. To include columns in the returned table for names that are not present in Pivot Name Column, specify a semicolon-separated list of names.
- Restrict to Name List: Numerical value that determines whether the returned table contains columns *only* for items in Name List. If set to 0 or if Name List is not specified, all unique values from Pivot Name Column are included in the returned table; otherwise only values from the Name List are included.

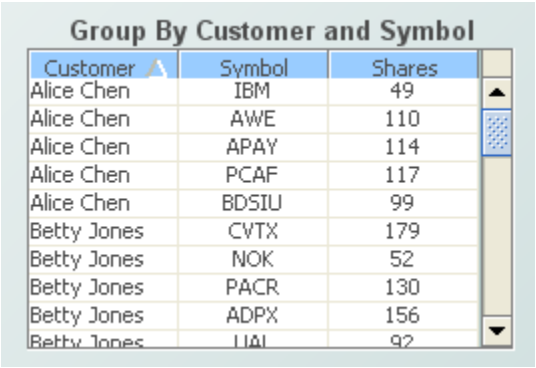
This function returns a table.

Example

The bar chart and the second table (labeled Pivot Customer), below, are attached to the function defined by the following dialog. The first table's data table is attached to the argument Table.

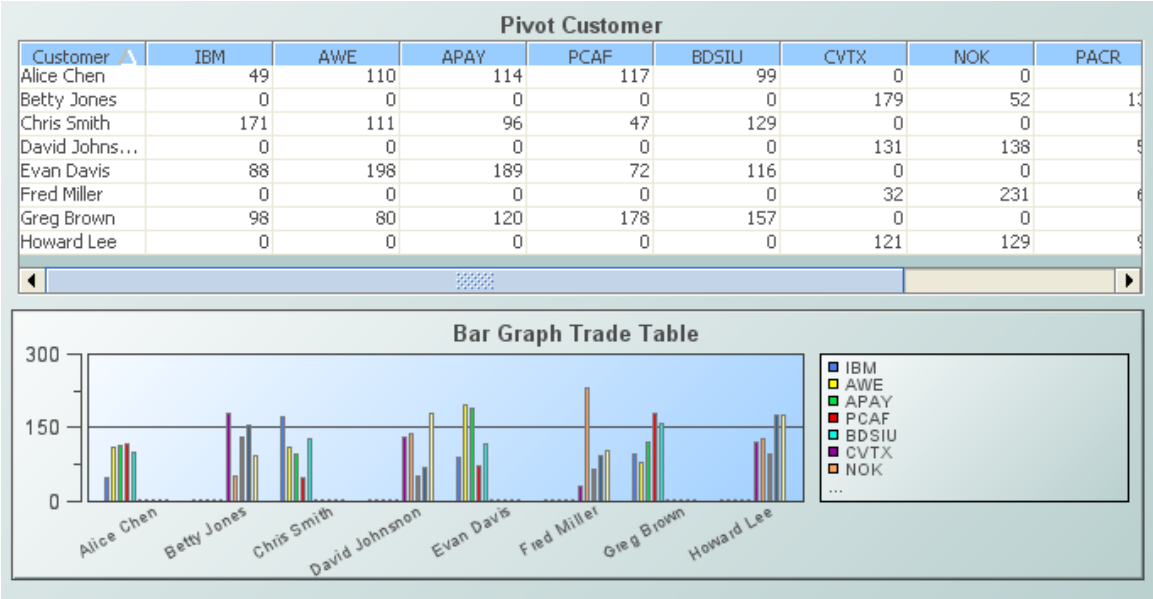
Figure 9. Pivot On Unique Values example





Group By Customer and Symbol

Customer	Symbol	Shares
Alice Chen	IBM	49
Alice Chen	AWE	110
Alice Chen	APAY	114
Alice Chen	PCAF	117
Alice Chen	BDSIU	99
Betty Jones	CVTX	179
Betty Jones	NOK	52
Betty Jones	PACR	130
Betty Jones	ADPX	156
Betty Jones	UAL	92



Tabular Functions

Reference

Returns a reference to the specified table without copying the contents.

Arguments

The function has the following argument:

- Table: The table for which a reference is to be returned.

This function returns a table.

Tabular Functions

Rename Columns

Returns a copy of a specified table with specified columns renamed with specified new names.

Arguments

The function has the following arguments:

- Table: The table whose columns are to be renamed.
- Column Name(s): Text string that specifies the columns to be renamed. Supply a single column name or a semicolon-separated list of column names.
- New Name(s): Text string that specifies the new column names. Supply a single name or a semicolon-separated list of names. The *n*th element of New Name(s) is the new name of the column specified by the *n*th element of Column Name(s). The number of names in Column Name(s) must be less than or equal to the number of names in New Name(s).

This function returns a table.

Tabular Functions

Select Column

Returns a one-column table containing only a specified column from a specified table.

Arguments

The function has the following arguments:

- Table: Table from which the column is to be selected.
- Select Column Name: Text string that specifies the name of the column to select.

This function returns a table.

Tabular Functions

Set Column Type

Returns a copy of a specified table, with specified columns modified to use specified types.

Arguments

The function has the following arguments:

- Table: Table from which the column is to be selected.
- Column Types: Text string that specifies column-name/type pairs. Separate pairs with spaces. Within each pair, separate column-name from type with a colon, that is, each pair has the following form:
  - *column-name:type*

  For *type*, supply one of the following:
  - STRING
  - INTEGER
  - LONG
  - DOUBLE
  - FLOAT
  - BOOLEAN
  - DATE

  If *column-name* contains a space or a colon, it must be enclosed in single quotes. Here is an example:
  - apama.timestamp:DATE 'Max Value':INTEGER Active:BOOLEAN

This function returns a table.

Tabular Functions

Sort Table

Returns a table with the same rows as a specified table but with the rows sorted according to the values in a specified column or columns.

If you specify multiple columns, the returned table is sorted on the first column specified, and then on the second column, and so forth.

Arguments

This function has the following arguments:

- Table: Table to sort.
- Sort Column Name: Text string that specifies the column or columns whose values determine the sort order. Separate column names with a semicolon. If the columns contain text, the sort order is alphabetic, unless the text consists entirely of numbers, in which case the sort order is numeric.
- Sort Descending: Numerical value (0 or 1) that determines whether the sort order is ascending or descending. If set to 1, the sort order is descending; otherwise, the sort order is ascending.

Note: If an invalid column name is entered, the original table is returned.

This function returns a table.

Tabular Functions

Split String

Returns a table with the result of splitting a given string using a specified separator. The returned table contains one column, with a row for each resulting substring.

Arguments

The function has the following arguments:

- String: Text string to be split.
- Separator: Text string consisting of a regular expression that specifies the separator. Use the regular expression form suitable for use with the Java `Pattern` class. See http://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html
- Results Column Name: Text string that specifies the name of the returned column.

This function returns a table.

Tabular Functions

String to Table

Returns a table whose cell values are specified by a string that uses specified row and column delimiters.

Arguments

The function has the following arguments:

- String: Text string to be converted into a table.
- Row Delimiter: Text string that specifies the delimiter by which the data for one row is separated (in String) from the data for the next row.

  Note, if you specify a delimiter that consists of more than one character, those characters are treated as a sequence of delimiters and a new row is created when any one of them is encountered.
- Column Delimiter: Text string that specifies the delimiter by which the data for one cell in a row is separated (in String) from the data for the next cell in that row. If the table is to contain only one column, do not specify a value for Column Delimiter.

  Note, if you specify a delimiter that consists of more than one character, those characters are treated as a sequence of delimiters and a new column is created when any one of them is encountered.
- Column Name Mode: Text string that specifies how the function should determine column names for the returned table. Specify one of the following:
  - AUTO: Use the generated names col0, col1, col2, and so forth.
  - STATIC: Use the names specified in Column Names.
  - STRING: Use the values specified in the first row of String.

  `AUTO` is the default setting; leaving this unset or set to a value other than `AUTO`, `STATIC`, or `STRING` defaults to `AUTO`.
- Column Names: Text string that specifies the column names to use in the returned table, if Column Name Mode is Static.
- Allow Empty Rows/Columns: Specifies whether or not empty cells will be created when empty tokens in the string are encountered. Setting this argument to "`true`" or "`1`" means empty cells will be created. For example, in a string that uses "," (comma) as a delimiter, a row represented by `1, , ,4` will result in a row with `1` in the first column followed by two empty cells, followed by `4`. Setting this to "`false`", "`0`", or leaving it unset (the default) means that empty tokens will be ignored. In this case the "`1, , ,4`" example will create a row with `1` in the first column, followed by `4` in the second column, followed by two blank columns.

This function returns a table.

Tabular Functions

Subtotal By Time

Returns a table that contains subtotals for the data in a given table. Subtotals are provided for a specified number of specified time intervals.

Arguments

The function has the following arguments:

- Table: Table for which subtotals are to be provided. The Table must contain a time column and a number column.
- Date Parts Per Interval: Numerical value that specifies he size of each interval in Date Parts.
- Number Of Intervals: Numerical value that specifies the number of intervals to include in the summary table. The returned table contains one row for each interval. If set to 0, one subtotal row is provided for the entire table.
- Date Part: Text string that specifies the date unit to use. Enter s, m, h, d, w, M, q, or y, for seconds, minutes, hours, days, weeks, months, quarters, or years. The default unit is seconds.
- Date Format: Text string that specifies the format of times in the returned table. Specify a pattern string suitable for use with the Java `SimpleDateFormat` class. For example, the format EEE, hh:mm a results in a string of the form exemplified by Wed, 05:32 PM. Use q, qqq or qqqq for short, medium or long versions of quarter notation. For example, qqq-yyyy results in a string of the form exemplified by Qtr 1-2005. If no Date Format is given, dates are expressed in the form exemplified by 08/30/03 05:32 PM. If no Date Format is given, the type of the first column in the returned table is Date; otherwise it is String.
- Use Column Names: Numerical value (0 or 1) that determines whether original column names are retained in the returned table. If Use Column Names is set to 1, original column names are retained. If set to 0, columns are given generic names. Generic column names are useful when the data attachment for the Table argument uses a substitution that causes the column names to change when the substitution changes.

This function returns a table.

Tabular Functions

Subtotal By Unique Values

Returns a table that lists all of the unique values found in the first column of a specified table, along with the sum of the values in the corresponding fields of the remaining columns.

Arguments

The function has the following arguments:

- Table Columns: Table for which subtotals are to be provided. The function uses the first column of Table Columns as the index column. Subtotals are provided for the remaining columns of Table Columns.
- Value List: Table column that contains a set of values to be included in the set of values for the index column. This is useful if you want the returned table to include values that may or may not be in the Table Columns index column.

- Restrict To Value List: Numerical value (0 or 1) that determines whether the table includes only rows that include the values of Value List. If set to 1, only such values are included.
- Use Column Names: Numerical value (0 or 1) that determines whether original column names are retained in the returned table. If Use Column Names is set to 1, original column names are retained. If set to 0, columns are given generic names. Generic column names are useful when the data attachment for the Table argument uses a substitution that causes the column names to change when the substitution changes.

This function returns a table.

Tabular Functions

Subtract Columns

Returns a table that includes a column reflecting the difference between two specified columns of a specified table, the difference between a specified value and a specified column, or the difference between two specified values.

Usage notes

- Case 1: Difference between two specified columns of a specified table. This is the case if First Column Name or Numeric Value and Second Column Name or Numeric Value both specify a column of Table.
- Case 2: Difference between a specified value and a specified column. This is the case if one of First Column Name or Numeric Value and Second Column Name or Numeric Value specifies a column of Table and the other specifies a numeric value.
- Case 3: Difference between two specified values. This is the case if First Column Name or Numeric Value and Second Column Name or Numeric Value both specify a numeric value.

In case 1, the *n*th cell in the returned column corresponds to the *n*th cell of the table specified by the argument Table. Each cell of the returned column contains the result of subtracting the corresponding row's cell in Second Column Name or Numeric Value from the cell in First Column Name or Numeric Value.

In case 2, the *n*th cell in the returned column corresponds to the *n*th cell of the table specified by the argument Table. Each cell of the returned column contains the result of subtracting the specified numeric value from the corresponding row's cell in the specified column.

In case 3, each cell of the returned column contains the subtracting the second specified value from the first.

In the returned table, the sum column is preceded by copies of all the columns in Table.

Arguments

The function has the following arguments:

- Table: Table that contains the columns to be used in the subtraction operations.
- First Column Name or Numeric Value: Text string specifying minuend, the column to be subtracted from, or numerical value to be subtracted from.
- Second Column Name or Numeric Value: Text string specifying subtrahend, the column to be subtracted, or numerical value to be subtracted.

- Result Column Name: Text string that specifies the name of the column containing the differences. You must supply a value for this argument.

This function returns a table.

Tabular Functions

Table Contains Values

Returns a copy of the specified Table with a new boolean column containing a value of true for each row where the value in the Comparison Column is contained in the one and only column of the Comparison Table.

This function has the following arguments:

- Table: Table to be supplemented with the new boolean-valued column of comparison results.
- Comparison Column Name: Name of the column in Table whose values are searched for in Comparison Table
- Result Column Name: Name of the boolean result column to add to Table. If no name is specified, the column is named Result.
- Comparison Table: Single-column table in which to look for values from Comparison Column of Table

This function returns a table.

Tabular Functions