Informatica®
10.4

Enterprise Data Preparation Function Reference

Informatica Enterprise Data Preparation Function Reference
10.4
December 2019

© Copyright Informatica LLC 2021

This software and documentation are provided only under a separate license agreement containing restrictions on use and disclosure. No part of this document may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording or otherwise) without prior consent of Informatica LLC.

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation is subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License.

Informatica, the Informatica logo, and Enterprise Data Preparation are trademarks or registered trademarks of Informatica LLC in the United States and many jurisdictions throughout the world. A current list of Informatica trademarks is available on the web at https://www.informatica.com/trademarks.html. Other company and product names may be trade names or trademarks of their respective owners.

Portions of this software and/or documentation are subject to copyright held by third parties. Required third party notices are included with the product.

The information in this documentation is subject to change without notice. If you find any problems in this documentation, report them to us at infa_documentation@informatica.com.

Informatica products are warranted according to the terms and conditions of the agreements under which they are provided. INFORMATICA PROVIDES THE INFORMATION IN THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT.

Publication Date: 2021-06-15

Table of Contents

Preface 6

Informatica Resources. 6

Informatica Network. 6

Informatica Knowledge Base. 6

Informatica Documentation. 6

Informatica Product Availability Matrices. 7

Informatica Velocity. 7

Informatica Marketplace. 7

Informatica Global Customer Support. 7

Chapter 1: Function Reference 8

Function Reference Overview. 8

Function Categories. 9

Date and Time Functions. 9

Math and Trigonometry Functions. 10

Text Functions. 10

Logical Functions. 11

Statistical Functions. 11

Regular Expression Functions. 12

Other Functions. 12

Chapter 2: Functions 13

ABS. 15

ADD_TO_DATE. 16

AND. 16

AVG. 17

AVGIF. 18

CONCATENATE. 19

COUNT. 21

COUNTDISTINCT. 21

COUNTDISTINCTIF. 22

COUNTIF. 23

CURRENT_DATETIME. 24

DATE. 24

DATE_DIFF. 25

DATETIME. 26

DATE_TO_UNIXTIME. 27

DAY. 28

DAYOFWEEK. 29

ENDSWITH. 29

ENDSWITH_IGNORE_CASE. 31
EOMONTH. 32
EXP. 33
EXTRACT_MONTH_NAME. 34
FIND. 36
FIND_IGNORE_CASE. 37
FIND_REGEX. 38
FIRST_CHARACTER_TO_NUMBER. 40
HOUR. 41
IF. 42
ISNULL. 43
LEFT. 44
LEN. 45
LN. 46
LOG. 47
LOWER. 48
MAX. 48
MAXIF. 49
MAXINLIST. 50
MID. 52
MIN. 53
MINIF. 54
MININLIST. 55
MINUTE. 56
MONTH. 57
NOT. 58
NUMBER_TO_CHARACTER. 59
OR. 59
PI. 60
POWER. 61
PROPER_CASE. 62
QUARTER. 62
REMOVE_NON_ALPHANUMERIC_CHARACTERS. 63
RIGHT. 64
ROUND. 65
ROUNDDOWN. 66
ROUNDUP. 68
SECOND. 69
SQRT. 70
STARTSWITH. 71
STARTSWITH_IGNORE_CASE. 72
STDDEV. 74

STDDEVIF. 75
SUBSTITUTE. 76
SUBSTITUTE_IGNORE_CASE. 78
SUBSTITUTE_REGEX. 79
SUM. 80
SUMIF. 81
TODAY. 82
TRIM. 83
TRIM_ALL. 84
UNIXTIME_TO_DATE. 84
UPPER. 85
VARIANCE. 86
VARIANCEIF. 87
YEAR. 88

Index. 89

Preface

Use the Enterprise Data Preparation Function Reference Guide to learn about various functions that you can use as part of Enterprise Data Preparation tasks.

Informatica Resources

Informatica provides you with a range of product resources through the Informatica Network and other online portals. Use the resources to get the most from your Informatica products and solutions and to learn from other Informatica users and subject matter experts.

Informatica Network

The Informatica Network is the gateway to many resources, including the Informatica Knowledge Base and Informatica Global Customer Support. To enter the Informatica Network, visit https://network.informatica.com.

As an Informatica Network member, you have the following options:

- Search the Knowledge Base for product resources.
- View product availability information.
- Create and review your support cases.
- Find your local Informatica User Group Network and collaborate with your peers.

Informatica Knowledge Base

Use the Informatica Knowledge Base to find product resources such as how-to articles, best practices, video tutorials, and answers to frequently asked questions.

To search the Knowledge Base, visit https://search.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Documentation

Use the Informatica Documentation Portal to explore an extensive library of documentation for current and recent product releases. To explore the Documentation Portal, visit https://docs.informatica.com.

If you have questions, comments, or ideas about the product documentation, contact the Informatica Documentation team at infa_documentation@informatica.com.

Informatica Product Availability Matrices

Product Availability Matrices (PAMs) indicate the versions of the operating systems, databases, and types of data sources and targets that a product release supports. You can browse the Informatica PAMs at https://network.informatica.com/community/informatica-network/product-availability-matrices.

Informatica Velocity

Informatica Velocity is a collection of tips and best practices developed by Informatica Professional Services and based on real-world experiences from hundreds of data management projects. Informatica Velocity represents the collective knowledge of Informatica consultants who work with organizations around the world to plan, develop, deploy, and maintain successful data management solutions.

You can find Informatica Velocity resources at http://velocity.informatica.com. If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Marketplace

The Informatica Marketplace is a forum where you can find solutions that extend and enhance your Informatica implementations. Leverage any of the hundreds of solutions from Informatica developers and partners on the Marketplace to improve your productivity and speed up time to implementation on your projects. You can find the Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

You can contact a Global Support Center by telephone or through the Informatica Network.

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
https://www.informatica.com/services-and-training/customer-success-services/contact-us.html.

To find online support resources on the Informatica Network, visit https://network.informatica.com and select the eSupport option.

CHAPTER 1

Function Reference

This chapter includes the following topics:

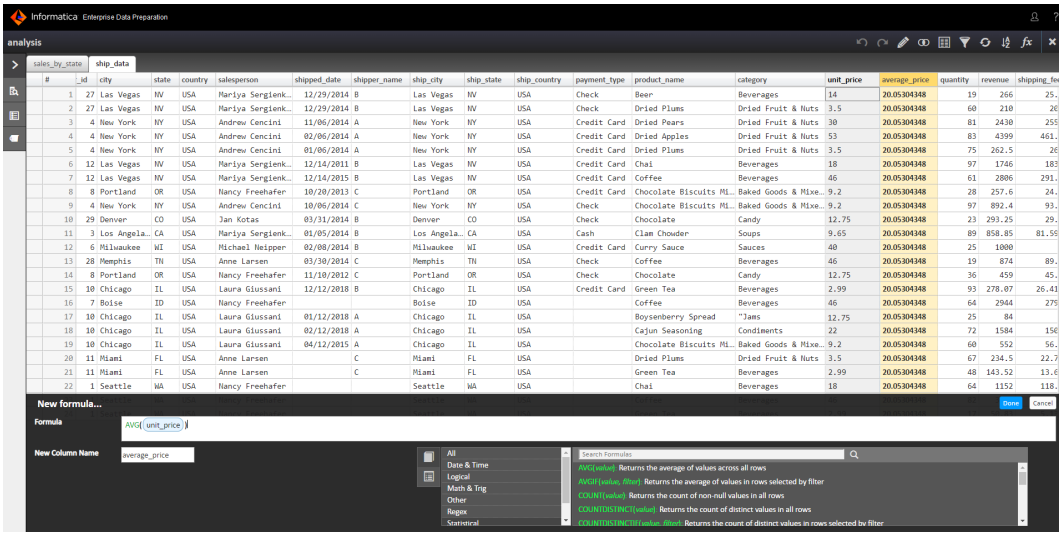
- Function Reference Overview, 8
- Function Categories, 9

Function Reference Overview

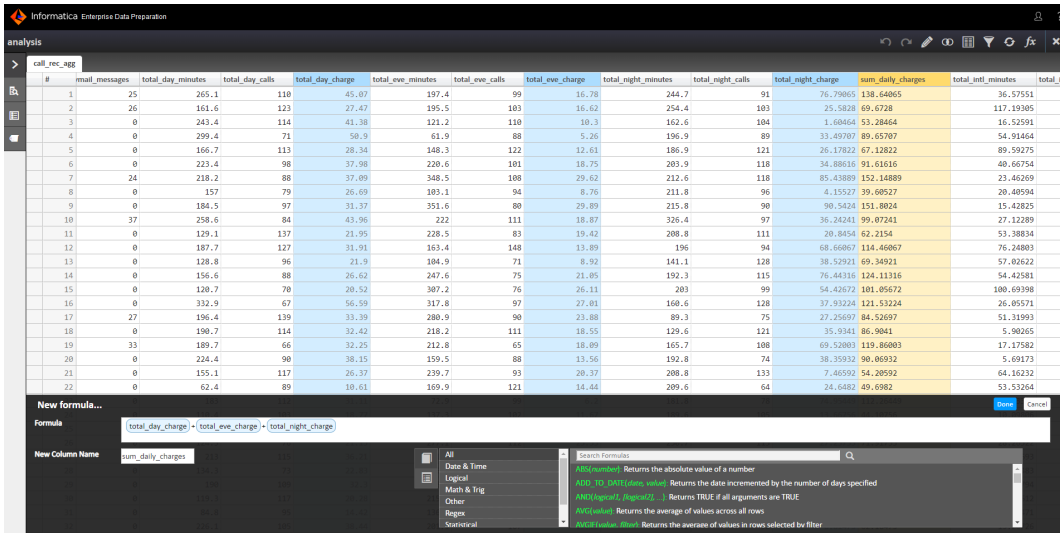
You can use the in-built functions available in Enterprise Data Preparation to perform calculations and return values in a worksheet. These functions help you to prepare the data for a publication. To help you find the function you need, functions are organized into groups, such as Date and Time functions.

Example

This example uses the AVG aggregate function to calculate the average of the values in the unit_price column. The following image shows how results appear:



You can also create your own formulas. The following example adds the total day, evening and night call charges and provides the results in the sum_daily_charges column:



Function Categories

The following types of functions are available in Enterprise Data Preparation:

- Date and time
- Mathematical
- Text
- Logical
- Statistical
- Regular Expressions
- Other

Date and Time Functions

Enterprise Data Preparation includes a group of Date functions to round, truncate, compare dates, extract one part of a date, or perform arithmetic on a date. You can pass any value with a date data type to any of the date functions.

Enterprise Data Preparation includes the following Date and Time functions:

- ADD_TO_DATE
- CURRENT_DATETIME
- DATE
- DATETIME
- DATE_DIFF
- DATE_TO_UNIXTIME
- DAY

- DAYOFWEEK
- EOMONTH
- EXTRACT_MONTH_NAME
- HOUR
- MINUTE
- MONTH
- QUARTER
- SECOND
- TODAY
- UNIXTIME_TO_DATE
- YEAR

Math and Trigonometry Functions

Enterprise Data Preparation includes a group of Math and Trigonometry Functions to perform some of the common mathematical operations including exponents, logarithms, and round off.

Enterprise Data Preparation includes the following Math and Trigonometry Functions:

- ABS
- EXP
- LN
- LOG
- PI
- POWER
- ROUND
- ROUNDDOWN
- ROUNDUP
- SQRT

Text Functions

Enterprise Data Preparation includes a group of text functions to concatenate, extract part of a string, substitute, or perform some basic text manipulation.

Enterprise Data Preparation includes the following Text functions:

- CONCATENATE
- ENDSWITH
- ENDSWITH_IGNORE_CASE
- FIND
- FIND_IGNORE_CASE
- FIRST_CHARACTER_TO_NUMBER
- LEFT
- LEN

- LOWER
- MID
- NUMBER_TO_CHARACTER
- PROPER_CASE
- REMOVE_NON_ALPHANUMERIC_CHARACTERS
- RIGHT
- STARTSWITH
- STARTSWITH_IGNORE_CASE
- SUBSTITUTE
- SUBSTITUTE_IGNORE_CASE
- TRIM
- TRIM_ALL
- UPPER

Logical Functions

Enterprise Data Preparation includes a group of Logical functions that returns boolean values based on the specified conditional validation.

Enterprise Data Preparation includes the following Logical functions:

- AND
- IF
- ISNULL
- NOT
- OR

Statistical Functions

Enterprise Data Preparation includes a group of Statistical functions to perform some basic statistical operations, such as, average calculation, standard deviation calculation, and variance calculation.

Enterprise Data Preparation includes the following Statistical functions:

- AVG
- AVGIF
- COUNT
- COUNTDISTINCT
- COUNTDISTINCTIF
- COUNTIF
- MAX
- MAXIF
- MIN
- MINIF
- STDDEV

- STDDEVIF
- SUM
- SUMIF
- VARIANCE
- VARIANCEIF

Regular Expression Functions

Enterprise Data Preparation includes a group of Regular Expression functions to find and substitute text using Perl compatible regular expressions.

Enterprise Data Preparation includes the following Regular Expression functions:

- FIND_REGEX
- SUBSTITUTE_REGEX

Other Functions

Enterprise Data Preparation includes the following miscellaneous functions:

- MAXINLIST
- MININLIST

CHAPTER 2

Functions

This chapter includes the following topics:

- ABS, 15
- ADD_TO_DATE, 16
- AND, 16
- AVG, 17
- AVGIF, 18
- CONCATENATE, 19
- COUNT, 21
- COUNTDISTINCT, 21
- COUNTDISTINCTIF, 22
- COUNTIF, 23
- CURRENT_DATETIME, 24
- DATE, 24
- DATE_DIFF, 25
- DATETIME, 26
- DATE_TO_UNIXTIME, 27
- DAY, 28
- DAYOFWEEK, 29
- ENDSWITH, 29
- ENDSWITH_IGNORE_CASE, 31
- EOMONTH, 32
- EXP, 33
- EXTRACT_MONTH_NAME, 34
- FIND, 36
- FIND_IGNORE_CASE, 37
- FIND_REGEX, 38
- FIRST_CHARACTER_TO_NUMBER, 40
- HOUR, 41
- IF, 42
- ISNULL, 43
- LEFT, 44

- LEN, 45
- LN, 46
- LOG, 47
- LOWER, 48
- MAX, 48
- MAXIF, 49
- MAXINLIST, 50
- MID, 52
- MIN, 53
- MINIF, 54
- MININLIST, 55
- MINUTE, 56
- MONTH, 57
- NOT, 58
- NUMBER_TO_CHARACTER, 59
- OR, 59
- PI, 60
- POWER, 61
- PROPER_CASE, 62
- QUARTER, 62
- REMOVE_NON_ALPHANUMERIC_CHARACTERS, 63
- RIGHT, 64
- ROUND, 65
- ROUNDDOWN, 66
- ROUNDUP, 68
- SECOND, 69
- SQRT, 70
- STARTSWITH, 71
- STARTSWITH_IGNORE_CASE, 72
- STDDEV, 74
- STDDEVIF, 75
- SUBSTITUTE, 76
- SUBSTITUTE_IGNORE_CASE, 78
- SUBSTITUTE_REGEX, 79
- SUM, 80
- SUMIF, 81
- TODAY, 82
- TRIM, 83
- TRIM_ALL, 84
- UNIXTIME_TO_DATE, 84

- UPPER, 85
- VARIANCE, 86
- VARIANCEIF, 87
- YEAR, 88

ABS

Returns the absolute value of a numeric value.

Syntax

```
ABS( value )
```

The following table describes the argument for this command:

Argument	Required/ Optional	Description
value	Required	Numeric data type. Passes the values for which you want to return the absolute values. Enter a valid transformation expression.

Return Value

Positive numeric value. ABS returns the same data type as the numeric value passed as an argument. If you pass a Double, the function returns a Double. Similarly, if you pass an Integer, the function returns an Integer.

NULL if you pass a null value to the function.

Examples

The following expression returns the difference between two numbers as a positive value, regardless of which number is larger:

```
ABS( PRICE - COST )
```

PRICE	COST	RETURN VALUE
250	150	100
52	48	4
169.95	69.95	100
59.95	NULL	NULL
70	30	40
430	330	100
100	200	100

ADD_TO_DATE

Adds a specified number of days to a datetime value, and returns a date in the same format as the date that you pass to the function. ADD_TO_DATE accepts positive and negative integer values.

Syntax

```
ADD_TO_DATE(date, value)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
date	Required	Date/Time data type. Passes the values that you want to change. You can enter a valid transformation expression.
value	Required	An integer value specifying the number of days by which you want to change the date value. You can enter a valid transformation expression that evaluates to an integer.

Return Value

Date in the same format as the date you pass to this function.

NULL if a null value is passed as an argument to the function.

Examples

The following expression subtracts 10 days from each date in the DATE_SHIPPED port:

```
ADD_TO_DATE( DATE_SHIPPED, -10 )
```

DATE_SHIPPED	RETURN VALUE
Jan 1 1997 12:00:30AM	Dec 22 1996 12:00AM
Jan 31 1997 6:24:45PM	Jan 21 1997 6:24:45PM
Mar 9 1996 5:32:12AM	Feb 29 1996 5:32:12AM *(Leap Year)*
Oct 9 1997 2:30:12PM	Sep 30 1997 2:30:12PM
Mar 3 1996 5:12:20AM	Feb 22 1996 5:12:20AM
NULL	NULL

AND

Returns TRUE if all conditions evaluate to true. Returns FALSE if at least one condition is not true.

Syntax

```
AND( logical1, [logical2], .... )
```

The following table describes the argument for this command:

Argument	Required/ Optional	Description
logical1	Required	Passes the first condition that you want to evaluate.
logical2	Optional	Passes the second condition that you want to evaluate.

Return Value

TRUE if all conditions evaluate to TRUE.

FALSE if any of the conditions evaluate to FALSE.

Examples

The example below validates the following conditions in the insurance policies and based on that the function determines if the policy is eligible for a discount:

- The age of the insurer must be less than or equal to 35.
- No claim was made in last one year.

If both these conditions are true, the IS_ELIGIBLE_FOR_DISCOUNT column returns TRUE indicating that the policy is eligible for a discount.

```
AND( INSURER_AGE<=35, INSURANCE_CLAIMED=FALSE)
```

INSURANCE_ID	INSURER_AGE	INSURANCE_CLAIMED	IS_ELIGIBLE_FOR_DISCOUNT
467834839	26	TRUE	FALSE
467834840	31	FALSE	TRUE
467834841	38	TRUE	FALSE
467834842	47	FALSE	FALSE
467834843	56	TRUE	FALSE
467834844	35	FALSE	TRUE

AVG

Returns the average of all values in a group of rows.

Syntax

```
AVG( value )
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
value	Required	Numeric data type. Passes the values for which you want to calculate an average.

Return Value

Numeric value

NULL if all values passed to the function are NULL.

Nulls

If a value is NULL, AVG ignores the row. If all values passed from the port are NULL, AVG returns NULL.

Examples

The following expression returns the average wholesale cost of all items:

```
AVG( WHOLESALE_COST )
```

WHOLESALE_COST	RETURN VALUE
35.00	56.66666666666667
8.00	56.66666666666667
150.00	56.66666666666667
29.00	56.66666666666667
88.00	56.66666666666667
30.00	56.66666666666667

AVGIF

Returns the average of values within a port. You can apply a filter to limit the rows while calculating the average.

Syntax

```
AVGIF( value, filter)
```

The following table describes the arguments for this command:

Argument	Required / Optional	Description
value	Required	Numeric data type. Passes the values for which you want to calculate an average.
filter	Required	Filters condition that you want to evaluate.

Return Value

Numeric value

NULL if a null value is passed as an argument to the function.

Nulls

If a value is NULL, AVGIF ignores the row. If all values passed from the port are NULL, AVGIF returns NULL.

Examples

The following expression returns the average wholesale cost of flashlights:

```
AVG( WHOLESALE_COST, ITEM_NAME='Flashlight' )
```

ITEM_NAME	WHOLESALE_COST	RETURN VALUE
Flashlight	35.00	31.66
Navigation Compass	8.05	31.66
Regulator System	150.00	31.66
Flashlight	29.00	31.66
Depth/Pressure Gauge	88.00	31.66
Flashlight	31.00	31.66

CONCATENATE

Concatenates two or more strings. CONCATENATE converts all data to text before concatenating the strings.

Syntax

```
CONCATENATE( text1, [text2], ...)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
text1	Required	Any data type except Binary. The first part of the string you want to concatenate. You can enter a valid transformation expression.
[text2]	Required	Any data type except Binary. The second part of the string you want to concatenate. You can enter a valid transformation expression.

Return Value

String

NULL if both string values are NULL.

Nulls

If one of the strings is NULL, CONCAT ignores it and returns the other string.

If both strings are NULL, CONCAT returns NULL.

Examples

The following expression concatenates the names in the FIRST_NAME and LAST_NAME ports:

```
CONCATENATE( FIRST_NAME, LAST_NAME )
```

FIRST_NAME	LAST_NAME	RETURN VALUE
John	Baer	JohnBaer
NULL	Campbell	Campbell
Bobbi	Apperley	BobbiApperley
Jason	Wood	JasonWood

CONCATENATE does not add spaces to separate strings. If you want to add a space between two strings, you can write an expression as shown below:

```
CONCATENATE( text1,' ', [text2])
```

For example, the following expression first concatenates a space at the end of the first name and then concatenates the last name:

```
CONCATENATE(FIRST_NAME, ' ' , LAST_NAME )
```

FIRST_NAME	LAST_NAME	RETURN VALUE
John	Baer	John Baer
NULL	Campbell	Campbell *(includes leading blank)*
Bobbi	Apperley	Bobbi Apperley
Jason	Wood	Jason Wood

COUNT

Returns the number of rows that have non-null values in a column.

Syntax

```
COUNT( value)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
value	Required	The column for which you want to get the count of values.

Return Value

Integer

0 if all values passed to the function are NULL.

Examples

The following expression counts the items excluding null values:

```
COUNT( ITEM_NAME )
```

ITEM_NAME	RETURN VALUE
Flashlight	4
NULL	4
Compass	4
Regulator System	4
NULL	4
Halogen Flashlight	4

COUNTDISTINCT

Returns the number of rows with distinct values in a column.

Syntax

```
COUNTDISTINCT( value)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
value	Required	The column for which you want to get the count of the distinct values.

Return Value

Integer

Examples

The following expression counts the items with distinct names:

```
COUNTDISTINCT( ITEM_NAME )
```

ITEM_NAME	RETURN VALUE
Flashlight	4
NULL	4
Compass	4
Regulator System	4
NULL	4
Flashlight	4

COUNTDISTINCTIF

Returns the number of rows with distinct values in a column based on a specified filter.

Syntax

```
COUNTDISTINCTIF( value, filter)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
value	Required	The column for which you want to count the distinct values.
filter	Required	Limits the rows in the search. The filter condition must be a numeric value or evaluate to TRUE, FALSE, or NULL. You can enter a valid transformation expression.

Return Value

Integer

Examples

The following expression counts the items with distinct names whose quantity in stock is less than 5:

```
COUNTDISTINCTIF( ITEM_NAME, IN_STOCK < 5 )
```

ITEM_NAME	IN_STOCK	RETURN VALUE
Flashlight	4	2
NULL	2	2
Compass	NULL	2
Regulator System	6	2
NULL	8	2
Flashlight	1	2

COUNTIF

Returns the number of rows that have non-null values in a column. You can apply a condition to filter rows before counting them.

Syntax

```
COUNTIF( value, filter)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
value	Required	The column for which you want to get the count of values.
filter	Required	Limits the rows in the search. The filter condition must be a numeric value or evaluate to TRUE, FALSE, or NULL. You can enter a valid transformation expression.

Return Value

Integer

0 if all the values passed to this function are NULL.

Examples

The following expression counts the items with less than 5 quantity in stock, excluding null values:

```
COUNT( ITEM_NAME, IN_STOCK < 5 )
```

ITEM_NAME	IN_STOCK	RETURN VALUE
Flashlight	10	2
NULL	2	2
Compass	NULL	2
Regulator System	4	2
NULL	8	2
Halogen Flashlight	1	2

CURRENT_DATETIME

Returns the current date and time in UTC.

Syntax

```
CURRENT_DATETIME()
```

The CURRENT_DATETIME() function syntax has no arguments.

Return Value

Current date and time.

Examples

If the CURRENT_DATETIME() function is run on 27th October, 2020 at 08:36:22, the function returns the following value.

INPUT	RETURN VALUE
CURRENT_DATETIME()	10/27/2020 08:36:22

DATE

Returns the date based on the input values.

Syntax

```
DATE(year, month, day)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
year	Required	Numeric data type representing the year. Positive 4-digit integer. If you pass this function a 2-digit year, the function returns "00" as the first two digits of the year.
month	Required	Numeric data type representing the month. Positive integer between 1 and 12, where January=1 and December=12.
day	Required	Numeric data type. Positive integer between 1 and 31 based on the number of days in the month.

Return Value

Date as MM/DD/YYYY.

Examples

The following expression creates a date and time from the input ports:

```
DATE(SALE_YEAR, SALE_MONTH, SALE_DAY)
```

SALE_YEAR	SALE_MONTH	SALE_DAY	RETURN VALUE
2020	10	27	10/27/2020
2019	6	15	06/15/2019

DATE_DIFF

Returns the difference between two dates in days. The function subtracts the second date from the first date and returns the difference.

Syntax

```
DATE_DIFF( date1, date2 )
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
date1	Required	Date/Time data type. Passes the values for the first date you want to compare. You can enter any valid transformation expression.
date2	Required	Date/Time data type. Passes the values for the second date you want to compare. You can enter any valid transformation expression.

Return Value

Double value. If *date1* is later than *date2*, the return value is a positive number. If *date1* is earlier than *date2*, the return value is a negative number.

0 if the dates are the same.

NULL if one (or both) of the date values is NULL.

Examples

The following expressions return the number of days between the DATE_PROMISED and DATE_SHIPPED ports:

```
DATE_DIFF( DATE_PROMISED, DATE_SHIPPED )
```

DATE_PROMISED	DATE_SHIPPED	RETURN VALUE
Jan 1 2017 12:00:00AM	Mar 29 2017 12:00:00PM	-87.5
Mar 29 2017 12:00:00PM	Jan 1 2017 12:00:00AM	87.5
NULL	Dec 10 1997 5:55:10PM	NULL
Dec 10 1997 5:55:10PM	NULL	NULL
Jun 3 2017 1:13:46PM	Aug 23 2016 4:20:16PM	283.870486111111
Feb 19 2014 12:00:00PM	Feb 19 2015 12:00:00PM	-366

DATETIME

Returns the date and time based on the input values.

Syntax

```
DATE(year, month, day, hour, minute, second)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
year	Required	Numeric data type representing the year. Positive 4-digit integer. If you pass this function a 2-digit year, Enterprise Data Preparation returns "00" as the first two digits of the year.
month	Required	Numeric data type representing the month. Positive integer between 1 and 12 (January=1 and December=12).
day	Required	Numeric data type. Positive integer between 1 and 31 (except for the months that have less than 31 days: February, April, June, September, and November).

Argument	Required/ Optional	Description
hour	Required	Numeric data type. Positive integer between 0 and 24.
minute	Required	Numeric data type. Positive integer between 0 and 59.
second	Required	Numeric data type. Positive integer between 0 and 59.

Return Value

Date and time as MM/DD/YYYY HH:MI:SS.

Examples

The following expression creates a date and time from the input ports:

```
DATE(SALE_YEAR, SALE_MONTH, SALE_DAY, SALE_HOUR, SALE_MIN, SALE_SEC)
```

SALE_YEAR	SALE_MONTH	SALE_DAY	SALE_HOUR	SALE_MINUTE	SALE_SECOND	RETURN VALUE
2002	10	27	8	36	22	10/27/2002 08:36:22
2000	6	15	15	17	0	06/15/2000 15:17:00
99	12	12	5	30	16	12/12/0099 05:30:16

DATE_TO_UNIXTIME

Converts a Date/Time data type to Unix time.

Syntax

```
DATE_TO_UNIXTIME(date)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
date	Required	Date/Time data type. Passes the values for the date and time that you want to convert to Unix time.

Return Value

Date and time as Unix time.

Examples

The following expression converts each date in the DATE_SHIPPED port to Unix time:

```
DATE_TO_UNIXTIME(DATE_SHIPPED)
```

DATE_SHIPPED	RETURN VALUE
07/19/2015	1437264000
07/23/2013	1374537600
04/06/2021 09:10:05	1617700205
03/23/2021 11:26:13	1332501973

DAY

Returns the day of the month for each date in a port.

Syntax

```
DAY(date)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
date	Required	Date/Time data type. Passes the dates for which you want to return the day of the month. You can enter a valid transformation expression that evaluates to a date.

Return Value

Day of the month, a number from 1 to 31.

NULL if a value in the selected port is NULL or the data type in the selected port is not Date/Time.

Examples

The following expression returns the day from the input port:

```
DAY(DATE_SHIPPED)
```

DATE_SHIPPED	RETURN VALUE
07/19/2015	19
07/23/2013	23
04/06/2021 09:10:05	6
03/23/2021 11:26:13	23

DAYOFWEEK

Returns the day of the week for each date in a port.

Syntax

```
DAYOFWEEK(date)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
date	Required	Date/Time data type. Passes the dates for which you want to return the day of the week. You can enter a valid transformation expression that evaluates to a date.

Return Value

Day of the week.

NULL if a value in the selected port is NULL or the data type in the selected port is not Date/Time.

Examples

The following expression returns the day from the input port:

```
DAYOFWEEK(DATE_SHIPPED)
```

DATE_SHIPPED	RETURN VALUE
07/19/2015	Sunday
07/23/2013	Tuesday
04/06/2021 09:10:05	Tuesday
03/26/2021 11:26:13	Friday

ENDSWITH

Determines whether a string ends with the characters of a specified string. This function returns True if the string ends with the characters. The function returns False if the string does not end with the specified characters.
Note: This function is case-sensitive.

Syntax

```
ENDSWITH(text, ending_text)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
text	Required	Character string. Passes the strings that you want to search. You can enter a valid transformation expression. If you pass a numeric value, the function converts it to a character string.
ending_text	Required	Character string. Passes the characters to be searched for at the end of *text*.

Return Value

TRUE if the string ends with the specified characters.

FALSE if the string does not end with the specified characters.

Examples

In the following example, the order IDs are generated with a suffix of the quantity of items in the order. The QTY column contains the manually entered quantities of the orders. The following expression validates if the quantity entered in the QTY column is correct based on the order IDs in the ORDER_ID column.

```
ENDSWITH( ORDER_ID, QTY )
```

ORDER_ID	QTY	IS_QTY_CORRECT
1572502745-32	32	true
1572502746-13	31	false
1572502747-25	25	true
1572502748-34	341	false

You can enter a string in the *ending_text* parameter with single quotes (' '). The following expression validates if the month of the ORDER_DATE column is January.

```
ENDSWITH(ORDER_DATE, 'January' )
```

ORDER_DATE	RETURN VALUE
08-January	true
10-january	false *(as this function is case-sensitive)*
25-February	false

ORDER_DATE	RETURN VALUE
06-March	false
15-January	true

ENDSWITH_IGNORE_CASE

Determines whether a string ends with the characters of a specified string. This function returns true if the string ends with the characters. The function returns False if the string does not end with the specified characters.
Note: This function is not case-sensitive.

Syntax

```
ENDSWITH_IGNORE_CASE(text, ending_text)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
text	Required	Character string. Passes the strings you want to search. You can enter a valid transformation expression. If you pass a numeric value, the function converts it to a character string.
ending_text	Required	Character string. Passes the characters to be searched for at the end of *text*.

Return Value

TRUE if the string ends with the specified characters.

FALSE if the string does not end with the specified characters.

Examples

In the following example, the order IDs are generated with a suffix of the quantity of items in the order. The QTY column contains the manually entered quantities of the orders. The following expression validates if the quantity entered in the QTY column is correct based on the order IDs in the ORDER_ID column:

```
ENDSWITH_IGNORE_CASE( ORDER_ID, QTY )
```

ORDER_ID	QTY	IS_QTY_CORRECT
1572502745-32	32	true
1572502746-13	31	false

ORDER_ID	QTY	IS_QTY_CORRECT
1572502747-25	25	true
1572502748-34	341	false

You can enter a string in the *ending_text* parameter with single quotes (' '). The following expression validates if the month of the ORDER_DATE column is January:

```
ENDSWITH_IGNORE_CASE(ORDER_DATE, 'January' )
```

ORDER_DATE	RETURN VALUE
08-January	true
10-january	true *(as this function is not case-sensitive)*
25-February	false
06-March	false
15-January	true

EOMONTH

Returns the last day of the month after adding or subtracting a specified number of months to a date.

Syntax

```
EOMONTH(date, months)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
date	Required	Date/Time data type. Passes the date based on which you want to calculate the last day of the month.
months	Required	Positive or negative integer specifying the number of months that you want to add or subtract from the specified date.

Return Value

Date

NULL if a value in the selected port is NULL.

Examples

The following expression returns the last day of the month after two months from each date in the DATE_SHIPPED port:

```
EOMONTH(DATE_SHIPPED, 2)
```

DATE_SHIPPED	RETURN VALUE
09/22/2017	11/30/2017
08/28/2017	10/31/2017

The following expression returns the last day of the month before two months from each date in the DATE_SHIPPED port:

```
EOMONTH(DATE_SHIPPED, -2)
```

DATE_SHIPPED	RETURN VALUE
09/22/2017	07/31/2017
08/28/2017	06/30/2017

The following expression returns the last day of the month from each date in the DATE_SHIPPED port:

```
EOMONTH(DATE_SHIPPED, 0)
```

DATE_SHIPPED	RETURN VALUE
09/22/2017	09/30/2017
08/28/2017	08/31/2017

EXP

Returns e raised to the specified power (exponent), where e=2.71828183. For example, EXP(2) returns 7.3890561. You might use this function to analyze scientific and technical data rather than business data. EXP is the reciprocal of the LN function, which returns the natural logarithm of a numeric value.

Syntax

```
EXP( number )
```

The following table describes the argument for this command:

Argument	Required/ Optional	Description
number	Required	Numeric data type. The value to which you want to raise e. The exponent in the equation e^*value*. You can enter a valid transformation expression.

Return Value

Double value.

NULL if a value passed as an argument to the function is NULL.

Examples

The following expression uses the values stored in the Numbers port as the exponent value:

```
EXP( NUMBERS )
```

NUMBERS	RETURN VALUE
10	22026.46579481
-2	0.13533528
8.55	5166.75541266
NULL	NULL

EXTRACT_MONTH_NAME

Returns the name of the month for each date in a port.

Syntax

```
EXTRACT_MONTH_NAME(date, [fullMonthName])
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
date	Required	Date/Time data type. Passes the dates for which you want to return the name of the month.
[fullMonthName]	Optional	Specifies whether to return the full month name or the abbreviated month name. The fullMonthName must be the constants TRUE or FALSE. If the fullMonthName is set to TRUE, the EXTRACT_MONTH_NAME function returns the full month name. If the fullMonthName is set to FALSE, the EXTRACT_MONTH_NAME function returns the abbreviated month name. Default is false.

Return Value

Month name

NULL if a value in the selected port is NULL or if the data type in the selected port is not Date/Time.

Examples

The following expressions return the abbreviated month name from the input ports:

```
EXTRACT_MONTH_NAME(DATE_SHIPPED)
```

or

```
EXTRACT_MONTH_NAME(DATE_SHIPPED, FALSE)
```

or

```
EXTRACT_MONTH_NAME(DATE_SHIPPED, 0)
```

DATE_SHIPPED	RETURN VALUE
09/22/2017	Sep
06/14/2017	Jun

The following expressions return the full month name from the input ports:

```
EXTRACT_MONTH_NAME(DATE_SHIPPED, TRUE)
```

or

```
EXTRACT_MONTH_NAME(DATE_SHIPPED, 1)
```

DATE_SHIPPED	RETURN VALUE
02/20/2017	February
10/16/2017	October

FIND

Returns the position of one text string inside another text string as a number. If the text is not found, the function returns -1. If there is more than one occurrence of the search string, FIND returns the position of the first occurrence.

Note: This function is case-sensitive.

Syntax

```
FIND(find_text, within_text, [start_num])
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
find_text	Required	Character string. Passes the strings that you want to search. You can enter a valid transformation expression. If you pass a numeric value, the function converts it to a character string.
within_text	Required	Character string. Passes the strings within which the *find_text* strings need to be searched.
[start_num]	Optional	Numeric data type. Specifies the starting position in the text to search in the *within_text* string. Default is 1.

Return Value

Numeric

Examples

The following expression returns the position of the LAST_NAME in the NAME column.

```
FIND( LAST_NAME, NAME )
```

NAME	LAST_NAME	RETURN VALUE
Christ Karol	Karol	8
Robert Kennedy	Kennedy	8
Tom Flogger	flogger	-1 *(as this function is case-sensitive.)*
David Smith Smith	Smith	7 *(returns the position of the first occurrence.)*
Robert Brooke	Broke	-1 *(as the text is not found.)*

The following expression returns the position of the second hyphen in the ITEM_SERIAL_NUM:

```
FIND( '-', ITEM_SERIAL_NUM, 4 )
```

Note: The value in *[start_num]* parameter is set to 4 because all the serial numbers contain a hyphen in the third character. If the value is not set, it will consider the first hyphen.

ITEM_SERIAL_NUM	RETURN VALUE
JN-36M-GRN-302	7
SL-38-GR-01	6
DR-2-F1212/S-1	5
DL-150-1Q09-XL	7
NL-0123-XXS-01	8

FIND_IGNORE_CASE

Returns the position of one text string inside another text string as a number. If the text is not found, the function returns -1. If there is more than one occurrence of the search string, FIND_IGNORE_CASE returns the position of the first occurrence.
Note: This function is not case-sensitive.

Syntax

```
FIND_IGNORE_CASE(find_text, within_text, [start_num])
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
find_text	Required	Character string. Passes the strings that you want to search. You can enter a valid transformation expression. If you pass a numeric value, the function converts it to a character string.
within_text	Required	Character string. Passes the strings within which the *find_text* strings need to be searched.
[start_num]	Optional	Numeric data type. Specifies the starting position in the text to search in the *within_text* string. Default is 1.

Return Value

Numeric

Examples

The following expression returns the position of the LAST_NAME in the NAME column:

```
FIND_IGNORE_CASE( LAST_NAME, NAME )
```

NAME	LAST_NAME	RETURN VALUE
Christ Karol	Karol	8
Robert Kennedy	Kennedy	8
Tom Flogger	flogger	5 *(as this function is not case-sensitive.)*
David Smith Smith	Smith	7 *(returns the position of the first occurrence.)*
Robert Brooke	Broke	-1 *(as the text is not found.)*

The following expression returns the position of the second hyphen in the ITEM_SERIAL_NUM:

```
FIND_IGNORE_CASE( '-', ITEM_SERIAL_NUM, 4 )
```

Note: The value in *[start_num]* parameter is set to 4 as all the serial numbers contain a hyphen in the third character. If the value is not set, the function will consider the first hyphen.

ITEM_SERIAL_NUM	RETURN VALUE
JN-36M-GRN-302	7
SL-38-GR-01	6
DR-2-F1212/S-1	5
DL-150-1Q09-XL	7
NL-0123-XXS-01	8

FIND_REGEX

Returns the starting position of one text string within another text string based on a regular expression pattern. Using this function, you can also validate data patterns, such as IDs, telephone numbers, postal codes, and state names. You must use a perl compatible regular expression syntax.

Syntax

```
FIND_REGEX(find_text, within_text, [group_num])
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
find_text	Required	String data type. Passes the regular expression pattern that you want to match. You must use perl compatible regular expression syntax. Enclose the pattern in single quotes. Enclose each subpattern in parentheses.
within_text	Required	String data type. Passes the strings within which you want to search the regular expression pattern.
[group_num]	Optional	Integer value. Subpattern number of the regular expression you want to match. Use the following guidelines to determine the subpattern number: - no value or 1. Extracts the first regular expression subpattern. - 2. Extracts the second regular expression subpattern. - n. Extracts the n th regular expression subpattern. Default is 1.

Return Value

Numeric

-1 if the data does not match the pattern or if the input is a null value.

Examples

The following expression returns the position of the last name from the employee names:

```
FIND_REGEX( '(\w+)\s+(\w+)', EMPLOYEE_NAME, 2 )
```

EMPLOYEE_NAME	RETURN VALUE
Stephen Graham	9
Juan Carlos	6
Christ Karol	8
Robert Kennedy	8
David Smith Smith	-1
Robert Brooke	8
Jane	-1

The following expression validates a 10-digit telephone number against the pattern:

```
FIND_REGEX( '(\d\d\d-\d\d\d-\d\d\d\d)', PHONE_NUMBER )
```

PHONE_NUMBER	RETURN VALUE
408-555-1212	1
	-1
510-555-1212	1
92 555 51212	-1
650-555-1212	1
415-555-1212	1
831 555 12123	-1

The following expression finds the account numbers starting with 1835:

```
FIND_REGEX(‘(1835.*)’, ACCOUNT_NUMBER )
```

ACCOUNT_NUMBER	RETURN VALUE
18359652649	1
18349552952	-1
18359586596	1
18354765966	1
	-1

FIRST_CHARACTER_TO_NUMBER

Converts the first character of a string to its corresponding ASCII number.

Syntax

```
FIRST_CHARACTER_TO_NUMBER(text)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
text	Required	Character string. Passes the strings for which you want to convert the first character to ASCII number.

Return Value

Numeric

Examples

The following expression converts the first character of the ITEM_SERIAL_NUM into a ASCII number.

```
FIRST_CHARACTER_TO_NUMBER(ITEM_SERIAL_NUM)
```

ITEM_SERIAL_NUM	RETURN VALUE
JN-36M-GRN-302	74
SL-38-GR-01	83
DR-2-F1212/S-1	68
DL-150-1Q09-XL	68
NL-0123-XXS-01	78

HOUR

Returns the hour part of a date as an integer value from 0 (12:00AM) to 23 (11:00PM).

Syntax

```
HOUR(date)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
date	Required	Date/Time data type. Passes the dates for which you want to return the hour value.

Return Value

Integer from 0 to 23.

NULL if a value in the selected port is NULL or if the data type in the selected port is not Date/Time.

Examples

The following expression returns the hour for each date in the DATE_SHIPPED port:

```
HOUR(DATE_SHIPPED)
```

DATE_SHIPPED	RETURN VALUE
Mar 13 1997 12:00:00AM	0
Sep 2 1997 2:00:01AM	2
Aug 22 1997 12:00:00PM	12
June 3 1997 11:30:44PM	23
NULL	NULL

IF

Returns one of two values that you specify based on the results of a condition.

Syntax

```
IF( logical_test, value_if_true, value_if_false )
```

The following table describes the argument for this command:

Argument	Required/ Optional	Description
logical_test	Required	The condition that you want to evaluate. You can enter a valid transformation expression that evaluates to TRUE or FALSE.
value_if_true	Required	Any data type except Binary. The value you want to return if the condition is TRUE. The return value is always the data type specified by this argument. You can enter a valid transformation expression, including another IF expression.
value_if_false	Optional	Any data type except Binary. The value you want to return if the condition is FALSE. You can enter a valid transformation expression, including another IF expression. Default is TRUE.

Return Value

Value specified in *value_if_true* if the condition in *logical_test* evaluates to TRUE.

Value specified in *value_if_false* if the condition in *logical_test* evaluates to FALSE. Returns NULL if you do not specify a value in *value_if_false* parameter.

Examples

The following expression returns the employee name if the value in SALES column is greater than or equal to 100. If the value is less than 100, the function returns "Not Applicable."

```
IF( SALES >= 100, EMP_NAME, 'Not Applicable' )
```

SALES	EMP_NAME	RETURN_VALUE
150	John Smith	John Smith
50	Pierre Bleu	Not Applicable
120	Sally Green	Sally Green
100	Greg Jones	Greg Jones
67	Robert Smith	Not Applicable

ISNULL

Returns whether a value is NULL. ISNULL evaluates an empty string as FALSE.

Note: To test for empty strings, use LENGTH.

Syntax

```
ISNULL( value )
```

The following table describes the argument for this command:

Argument	Required/ Optional	Description
value	Required	The column to evaluate.

Return Value

TRUE if the value is NULL.

FALSE if the value is not NULL.

Examples

The following example checks for null values in the items table:

```
ISNULL( ITEM_NAME )
```

ITEM_NAME	RETURN VALUE
Flashlight	FALSE
NULL	TRUE
Regulator system	FALSE
''	FALSE*Empty string is not NULL*

LEFT

Extracts the specified number of characters from the left side of a string.

Syntax

```
LEFT(text, num_chars)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
text	Required	Character string. Passes the strings from which you want to extract the specified number of characters. You can enter a valid transformation expression. If you pass a numeric value, the function converts it to a character string.
num_chars	Required	Numeric data type. Specifies the number of characters to extract, starting from the left side of the string. The value must be greater than or equal to zero.

Return Value

Characters

NULL if the *num_chars* is less than zero.

Examples

The following expression extracts the first three characters from the strings in the MONTH_NAME column:

```
LEFT( MONTH_NAME, 3 )
```

MONTH_NAME	RETURN VALUE
January	Jan
February	Feb
March	Mar
April	Apr
May	May

LEN

Returns the number of characters in a string, including trailing blanks.

Syntax

```
LEN( text )
```

The following table describes the argument for this command:

Argument	Required/ Optional	Description
text	Required	String datatype. The strings you want to calculate the character length. You can enter any valid transformation expression.

Return Value

Integer representing the length of the string.

NULL if a value passed to the function is NULL.

Examples

The following expression returns the length of each customer name:

```
LEN( CUSTOMER_NAME )
```

CUSTOMER_NAME	RETURN VALUE
Bernice Davis	13
NULL	NULL
John Baer	9
Greg Brown	10

LN

Returns the natural logarithm of a numeric value. For example, `LN(3)` returns `1.09861229`. You usually use this function to analyze scientific data rather than business data.

This function is the reciprocal of the function EXP.

Syntax

```
LN( number )
```

The following table describes the argument for this command:

Argument	Required/ Optional	Description
number	Required	Numeric data type. It must be a positive number, greater than 0. Passes the values for which you want to calculate the natural logarithm. You can enter a valid transformation expression.

Return Value

Double value

NULL if a value passed to the function is NULL.

Examples

The following expression returns the natural logarithm for all values in the NUMBERS port:

```
LN( NUMBERS )
```

NUMBERS	RETURN VALUE
10	2.30258509
125	4.82831374
0.96	-0.04082202

NUMBERS	RETURN VALUE
NULL	NULL
-90	NULL
0	NULL

LOG

Returns the logarithm of a numeric value. You can use this function to analyze scientific data rather than business data.

Syntax

```
LOG( base, exponent )
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
base	Required	The base of the logarithm. Must be a positive numeric value other than 0 or 1. Any valid transformation expression that evaluates to a positive number other than 0 or 1.
exponent	Required	The exponent of the logarithm. Must be a positive numeric value greater than 0. Any valid transformation expression that evaluates to a positive number greater than 0.

Return Value

Double value.

NULL if a value passed to the function is NULL.

Examples

The following expression returns the logarithm for all values in the NUMBERS port:

```
LOG( BASE, EXPONENT )
```

BASE	EXPONENT	RETURN VALUE
15	1	0
.09	10	-0.95624466
NULL	18	NULL
35.78	NULL	NULL
-9	18	NULL

BASE	EXPONENT	RETURN VALUE
0	5	0
10	-2	NULL

LOWER

Converts uppercase string characters to lowercase.

Syntax

```
LOWER( text )
```

The following table describes the argument for this command:

Argument	Required/ Optional	Description
text	Required	Any string value. The argument passes the string values that you want to return as lowercase. You can enter a valid transformation expression that evaluates to a string.

Return Value

Lowercase character string.

NULL if a value in the selected port is NULL.

Examples

The following expression returns all first names to lowercase:

```
LOWER( FIRST_NAME )
```

FIRST_NAME	RETURN VALUE
antonia	antonia
NULL	NULL
THOMAS	thomas
PierRe	pierre
BERNICE	bernice

MAX

Returns the highest string value based on the alphanumeric order within a port.

You can also use MAX to return the latest date or the largest numeric value in a port or group.

Syntax

```
MAX( value )
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
value	Required	String data type. Passes the string values for which you want to return a maximum string value. You can enter a valid transformation expression.

Return Value

String

NULL if all values passed to the function are NULL.

Nulls

If a value is NULL, MAX ignores it. If all values passed from the port are NULL, MAX returns NULL.

Examples

The following expression returns the maximum item name:

```
MAX( ITEM_NAME )
```

MANUFACTURER_ID	ITEM_NAME	RETURN VALUE
101	First Stage Regulator	Wristband Thermometer
102	Electronic Console	Wristband Thermometer
104	Flashlight	Wristband Thermometer
104	Battery (9 volt)	Wristband Thermometer
104	Rope (20 ft)	Wristband Thermometer
104	60.6 cu ft Tank	Wristband Thermometer
107	75.4 cu ft Tank	Wristband Thermometer
108	Wristband Thermometer	Wristband Thermometer

MAXIF

Returns the highest string value based on the alphanumeric order within a port. You can apply a filter to limit the rows in the search.

You can also use MAXIF to return the latest date or the largest numeric value in a port.

Syntax

```
MAXIF( value, filter )
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
value	Required	String data type. Passes the string values for which you want to return a maximum string value. You can enter a valid transformation expression.
filter	Required	Limits the rows in the search. The filter condition must be a numeric value or evaluate to TRUE, FALSE, or NULL. You can enter a valid transformation expression.

Return Value

String

NULL if all values passed to the function are NULL or no rows are selected, for example, the filter condition evaluates to FALSE or NULL for all rows.

Nulls

If a value is NULL, MAXIF ignores it. If all values passed from the port are NULL, MAXIF returns NULL.

Examples

The following expression returns the maximum item name for manufacturer ID 104:

```
MAXIF( ITEM_NAME, MANUFACTURER_ID='104' )
```

MANUFACTURER_ID	ITEM_NAME	RETURN VALUE
101	First Stage Regulator	Rope (20 ft)
102	Electronic Console	Rope (20 ft)
104	Flashlight	Rope (20 ft)
104	Battery (9 volt)	Rope (20 ft)
104	Rope (20 ft)	Rope (20 ft)
104	60.6 cu ft Tank	Rope (20 ft)
107	75.4 cu ft Tank	Rope (20 ft)
108	Wristband Thermometer	Rope (20 ft)

MAXINLIST

Returns the highest string value from a set of ports. You can also use MAXINLIST to return the latest date or the maximum numeric value from a set of ports.

Syntax

```
MAXINLIST( value, [value], ... )
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
value	Required	String, numeric, or date data type. Passes the first set of values that you want to compare and return the maximum value. You can enter any valid transformation expression.
[value]	Required	String, numeric, or date data type. Passes the second set of values that you want to compare and return the maximum value. The data type of both values must be the same. You can enter any valid transformation expression.

Return Value

String, numeric, or date

NULL if any value passed to the function is NULL or the *value* and *[value]* ports contain different data types.

Example

The following expression returns the maximum number of items from the three batches:

```
MAXINLIST ( NO_OF_ITEM_BATCH1, NO_OF_ITEM_BATCH2, NO_OF_ITEM_BATCH3 )
```

NO_OF_ITEM_BATCH1	NO_OF_ITEM_BATCH2	NO_OF_ITEM_BATCH3	RETURN_VALUE
559	698	878	878
846	787	984	984
548	987	658	987
548	349	879	879
896	899	587	899

The following expression returns the latest date from the DATE_PROMISED and DATE_SHIPPED ports:

```
MAXINLIST ( DATE_PROMISED, DATE_SHIPPED )
```

DATE_PROMISED	DATE_SHIPPED	RETURN VALUE
01-01-2017	03-29-2017	03-29-2017
03-29-2017	05-01-2017	05-01-2017
NULL	12-10-2017	NULL
12-15-2017	NULL	NULL
06-03-2017	08-23-2017	08-23-2017
02-19-2014	02-24-2014	02-24-2014

MID

Extracts a specified number of characters from the middle of a string.

Syntax

```
LEFT(text, start, num_chars)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
text	Required	Character string. Passes the strings from which you want to extract the specified number of characters. You can enter a valid transformation expression. If you pass a numeric value, the function converts it to a character string.
start	Required	Numeric data type. Specifies the position of the first character that you want to extract. The value must be greater than zero.
num_chars	Required	Numeric data type. Specifies the number of characters to extract, starting from the position defined in *start* parameter. The value must be greater than or equal to zero.

Return Value

Characters including the trailing space.

NULL if the value in the *num_chars* parameter is less than zero.

NULL if the value in the *start* parameter is less than or equal to zero, or greater than the total number of characters including the trailing space in the string.

Examples

The following expression returns three characters starting from the 5th position from the FIRST_NAME column.

```
MID( FIRST_NAME, 5, 3 )
```

FIRST_NAME	RETURN VALUE
Stephen	hen
Christopher	sto
Tom	NULL

FIRST_NAME	RETURN VALUE
Steve	e
Ladd	

MIN

Returns the lowest string value based on the alphanumeric order in a port. You can also use MIN to return the oldest date or the minimum numeric value in a port or group.

Syntax

```
MIN( value )
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
value	Required	String data type. Passes the values for which you want to return minimum value. You can enter any valid transformation expression.

Return Value

String value.

NULL if all values passed to the function are NULL.

Nulls

If a single value is NULL, MIN ignores it. If all values passed from the port are NULL, MIN returns NULL.

Examples

The following expression returns the minimum item name:

```
MIN ( ITEM_NAME )
```

MANUFACTURER_ID	ITEM_NAME	RETURN VALUE
101	First Stage Regulator	60.6 cu ft Tank
102	Electronic Console	60.6 cu ft Tank
104	Flashlight	60.6 cu ft Tank
104	Battery (9 volt)	60.6 cu ft Tank
104	Rope (20 ft)	60.6 cu ft Tank
104	60.6 cu ft Tank	60.6 cu ft Tank

MANUFACTURER_ID	ITEM_NAME	RETURN VALUE
107	75.4 cu ft Tank	60.6 cu ft Tank
108	Wristband Thermometer	60.6 cu ft Tank

MINIF

Returns the lowest string value based on the alphanumeric order in a port. You can apply a filter to limit the rows in the search.

You can also use MINIF to return the oldest date or the minimum numeric value in a port or group.

Syntax

```
MINIF( value, filter )
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
value	Required	String data type. Passes the values for which you want to return minimum value. You can enter a valid transformation expression.
filter	Required	Limits the rows in the search. The filter condition must be a numeric value or evaluate to TRUE, FALSE, or NULL. You can enter a valid transformation expression.

Return Value

String value

NULL if all values passed to the function are NULL or no rows are selected, for example, the filter condition evaluates to FALSE or NULL for all rows.

Nulls

If a single value is NULL, MINIF ignores it. If all values passed from the port are NULL, MINIF returns NULL.

Examples

The following expression returns the minimum item name for manufacturer ID 104:

```
MINIF ( ITEM_NAME, MANUFACTURER_ID='104' )
```

MANUFACTURER_ID	ITEM_NAME	RETURN VALUE
101	First Stage Regulator	60.6 cu ft Tank
102	Electronic Console	60.6 cu ft Tank
104	Flashlight	60.6 cu ft Tank
104	Battery (9 volt)	60.6 cu ft Tank

MANUFACTURER_ID	ITEM_NAME	RETURN VALUE
104	Rope (20 ft)	60.6 cu ft Tank
104	60.6 cu ft Tank	60.6 cu ft Tank
107	75.4 cu ft Tank	60.6 cu ft Tank
108	Wristband Thermometer	60.6 cu ft Tank

MININLIST

Returns the lowest string value from a set of ports. You can also use MININLIST to return the oldest date or the minimum numeric value from a set of ports.

Syntax

```
MININLIST( value, [value], ... )
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
value	Required	String, numeric, or date data type. Passes the first set of values that you want to compare and return the minimum value. You can enter any valid transformation expression.
[value]	Required	String, numeric, or date data type. Passes the second set of values that you want to compare and return the minimum value. The data type of both values must be the same. You can enter any valid transformation expression.

Return Value

String, numeric, or date

NULL if any value passed to the function is NULL or the *value* and *[value]* ports contain different data types.

Example

The following expression returns the minimum number of items from the three batches:

```
MININLIST ( NO_OF_ITEM_BATCH1, NO_OF_ITEM_BATCH2, NO_OF_ITEM_BATCH3 )
```

NO_OF_ITEM_BATCH1	NO_OF_ITEM_BATCH2	NO_OF_ITEM_BATCH3	RETURN_VALUE
559	698	878	559
846	787	984	787
548	987	658	548

NO_OF_ITEM_BATCH1	NO_OF_ITEM_BATCH2	NO_OF_ITEM_BATCH3	RETURN_VALUE
548	349	879	349
896	899	587	587

The following expression returns the oldest date from the DATE_PROMISED and DATE_SHIPPED ports:

```
MININLIST ( DATE_PROMISED, DATE_SHIPPED )
```

DATE_PROMISED	DATE_SHIPPED	RETURN VALUE
01-01-2017	03-29-2017	01-01-2017
03-29-2017	05-01-2017	03-29-2017
NULL	12-10-2017	NULL
12-15-2017	NULL	NULL
06-03-2017	08-23-2017	06-03-2017
02-19-2014	02-24-2014	02-19-2014

MINUTE

Returns the hour part of a date as an integer value from 0 to 59.

Syntax

```
MINUTE(date)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
date	Required	Date/Time data type. Passes the dates for which you want to return the minute value.

Return Value

Integer from 0 to 59.

NULL if a value in the selected port is NULL or the data type in the selected port is not Date/Time.

Examples

The following expression returns the minute values for each date in the DATE_SHIPPED port:

```
MINUTE(DATE_SHIPPED)
```

DATE_SHIPPED	RETURN VALUE
Mar 13 1997 12:00:00AM	0
Sep 2 1997 2:59:01AM	59
Aug 22 1997 12:02:00PM	2
June 3 1997 11:30:44PM	30
NULL	NULL

MONTH

Returns the month part of a date as an integer value from 1 to 12, where January = 1 and December = 12.

Syntax

```
MONTH(date)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
date	Required	Date/Time data type. Passes the dates for which you want to return the month value.

Return Value

Integer from 1 to 12.

NULL if a value in the selected port is NULL or the data type in the selected port is not Date/Time.

Examples

The following expression returns the month value for each date in the DATE_SHIPPED port.

```
MONTH(DATE_SHIPPED)
```

DATE_SHIPPED	RETURN VALUE
Mar 13 1997 12:00:00AM	3
Sep 2 1997 2:00:01AM	9
Aug 22 1997 12:00:00PM	8

DATE_SHIPPED	RETURN VALUE
December 3 1997 11:30:44PM	12
NULL	NULL

NOT

Returns TRUE if the logical argument is FALSE. Returns FALSE if the logical argument is TRUE.

Syntax

```
NOT( logical )
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
logical	Required	Specify the condition that you want to evaluate.

Return Value

TRUE if the condition evaluates to FALSE.

FALSE if the condition evaluates to TRUE.

Examples

The following expression returns TRUE if the total quantity is not equal to 50.

```
NOT( TOTAL_QTY=50 )
```

TOTAL_QTY	RETURN VALUE
51	TRUE
50	FALSE
53	TRUE
50	FALSE
50	FALSE
48	TRUE

NUMBER_TO_CHARACTER

Converts a ASCII number to the corresponding character.

Syntax

```
NUMBER_TO_CHARACTER(number)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
number	Required	Numeric data type. Passes the ASCII code that you want to convert to character.

Return Value

Character

Examples

The following expression converts the numbers in the ITEM_SERIAL_NUM column into characters:

```
NUMBER_TO_CHARACTER(ITEM_SERIAL_NUM)
```

ITEM_SERIAL_NUM	RETURN VALUE
74	J
83	S
68	D
109	m
121	y
93	]

OR

Returns TRUE if any one of the conditions evaluates to True. Returns FALSE if all the conditions evaluate to False.

Syntax

```
OR( logical1, [logical2], .... )
```

The following table describes the argument for this command:

Argument	Required/ Optional	Description
logical1	Required	Passes the first condition that you want to evaluate.
logical2	Optional	Passes the second condition that you want to evaluate.

Return Value

TRUE if any one of the conditions evaluate to true.

FALSE if all the conditions evaluate to false.

Examples

The example validates the following conditions in insurance policies and determines the eligibility of insurers for a discount:

- The age of the insurer must be less than or equal to 35.
- No claim was made in last one year.

If any one of these conditions is true, the IS_ELIGIBLE_FOR_DISCOUNT column returns TRUE indicating that the policy is eligible for a discount.

```
OR( INSURER_AGE<=35, INSURANCE_CLAIMED=FALSE)
```

INSURANCE_ID	INSURER_AGE	INSURANCE_CLAIMED	IS_ELIGIBLE_FOR_DISCOUNT
467834839	26	TRUE	TRUE
467834840	45	FALSE	TRUE
467834841	38	TRUE	FALSE
467834842	47	FALSE	TRUE
467834843	56	TRUE	FALSE
467834844	35	FALSE	TRUE

PI

Returns the value of pi, 3.141592654.

Syntax

```
PI()
```

The PI() function syntax has no arguments.

Return Value

Value of pi, 3.141592654.

POWER

Returns a value raised to the exponent that you pass to the function.

Syntax

```
POWER( base, exponent )
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
base	Required	Numeric value. This argument is the base value. You can enter a valid transformation expression. If the base value is negative, the exponent must be an integer.
exponent	Required	Numeric value. This argument is the exponent value. You can enter a valid transformation expression. If the base value is negative, the exponent must be an integer. In this case, the function rounds any decimal values to the nearest integer before returning a value.

Return Value

Double value

NULL if you pass a null value to the function.

Examples

The following expression returns the values in the Numbers port raised to the values in the Exponent port:

```
POWER( NUMBERS, EXPONENT )
```

NUMBERS	EXPONENT	RETURN VALUE
10.0	2.0	100
3.5	6.0	1838.265625
3.5	5.5	982.594307804838
NULL	2.0	NULL
10.0	NULL	NULL
-3.0	-6.0	0.00137174211248285
3.0	-6.0	0.00137174211248285
-3.0	6.0	729.0
-3.0	5.5	729.0

PROPER_CASE

Capitalizes the first letter in each word of a string and converts all other letters to lowercase.

Syntax

```
PROPER_CASE(text)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
text	Required	Character string. Passes the strings for which you want to capitalize the first of each word.

Return Value

String

Examples

The following expression capitalizes the first letter of each word in the FULL_NAME column.

```
PROPER_CASE(FULL_NAME)
```

FULL_NAME	RETURN VALUE
christ karol	Christ Karol
robert Kennedy	Robert Kennedy
TOM FLOGGER	Tom Flogger
Steve McGuire	Steve Mcguire

QUARTER

Returns the quarter of a **date** as an integer value from 1 to 4. Quarter is a 3-month period and one-fourth part of the year.

Syntax

```
QUARTER(date)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
date	Required	Date/Time data type. Passes the dates for which you want to return the quarter value.

Return Value

Integer from 1 to 4.

NULL if a value in the selected port is NULL or the data type in the selected port is not Date/Time.

Examples

The following expression returns the quarter for each date in the DATE_SHIPPED port:

```
QUARTER(DATE_SHIPPED)
```

DATE_SHIPPED	RETURN VALUE
Mar 13 1997 12:00:00AM	1
Sep 2 1997 2:00:01AM	3
Aug 22 1997 12:00:00PM	3
December 3 1997 11:30:44PM	4
NULL	NULL

REMOVE_NON_ALPHANUMERIC_CHARACTERS

Removes non-alphanumeric characters from a string.

Syntax

```
REMOVE_NON_ALPHANUMERIC_CHARACTERS(text)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
text	Required	Character string. Passes the strings from which you want to remove the non-alphanumeric characters.

Return Value

Character

Examples

The following expression removes the non-alphanumeric characters from the strings in the ORDER_ID column:

```
REMOVE_NON_ALPHANUMERIC_CHARACTERS(ORDER_ID)
```

ORDER_ID	RETURN VALUE
TS-D___4/2%3-019f	TSD423019f
@!BWS-GKS,145t	BWSGKS145t
DR-2-F1212/S-1	DR2F1212S1

RIGHT

Extracts a specified number of characters from the right side of a string.

Syntax

```
RIGHT(text, num_chars)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
text	Required	Character string. Passes the strings from which you want to extract the specified number of characters. You can enter a valid transformation expression. If you pass a numeric value, the function converts it to a character string.
num_chars	Required	Numeric data type. Specifies the number of characters to extract, starting from the right side of the string. The value must be greater than or equal to zero.

Return Value

Characters

NULL if the *num_chars* is less than zero.

Examples

The following expression extracts the last three characters from the MONTH_NAME column:

```
RIGHT( MONTH_NAME, 3 )
```

MONTH_NAME	RETURN VALUE
January	ary
February	ary
March	rch
April	ril
May	May

ROUND

Rounds off numbers to a specified number of digits or decimal places.

Syntax

```
ROUND( number , digits)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
number	Required	Numeric data type. You can enter a valid transformation expression. Use operators to perform arithmetic before you round off the values.
digits	Required	Positive or negative integer. If you enter a positive value in *digits*, the function rounds off to this number of decimal places. For example, ROUND(12.99, 1) returns 13.0 and ROUND(15.44, 1) returns 15.4. If you enter a negative value in *digits*, the function rounds off this number of digits to the left of the decimal point, returning an integer. For example, ROUND(12.99, -1) returns 10 and ROUND(15.99, -1) returns 20. If you enter decimal value in *digits*, the function rounds off to the nearest integer before evaluating the expression. For example, ROUND(12.99, 0.8) returns 13.0 because the function rounds 0.8 to 1 and then evaluates the expression.

Return Value

Numeric value

If one of the arguments is NULL, ROUND returns NULL.

Examples

The following expression returns the values in the PRICE port rounded to three decimal places:

```
ROUND( PRICE, 3 )
```

PRICE	RETURN VALUE
12.9936	12.994
15.9949	15.995
-18.8678	-18.868
56.9561	56.956
NULL	NULL

You can round digits to the left of the decimal point by passing a negative integer in the *digits* argument:

```
ROUND( PRICE, -2 )
```

PRICE	RETURN VALUE
13242.99	13200.0
1435.99	1400.0
-108.95	-100.0
NULL	NULL

If you pass a decimal value in the *digits* argument, the function rounds it to the nearest integer before evaluating the expression:

```
ROUND( PRICE, 0.8 )
```

PRICE	RETURN VALUE
12.99	13.0
56.34	56.0
NULL	NULL

ROUNDDOWN

Rounds off numbers down to a specified number of digits or decimal places.

Syntax

```
ROUNDDOWN( number , digits)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
number	Required	Numeric data type. You can enter a valid transformation expression. Use operators to perform arithmetic before you round off the values.
digits	Required	Positive or negative integer. If you enter a positive value in *digits*, the function rounds off to this number of decimal places. For example, ROUNDDOWN(12.99, 1) returns 12.9 and ROUNDDOWN(15.44, 1) returns 15.4. If you enter a negative value in *digits*, the function rounds off this number of digits to the left of the decimal point, returning an integer. For example, ROUNDDOWN(12.99, -1) returns 10 and ROUNDDOWN(15.99, -1) returns 10. If you enter decimal value in *digits*, the function rounds off to the nearest integer before evaluating the expression. For example, ROUNDDOWN(12.99, 0.8) returns 12.0 because the function rounds 0.8 to 1 and then evaluates the expression.

Return Value

Numeric value

If one of the arguments is NULL, ROUND returns NULL.

Examples

The following expression returns the values in the PRICE port rounded to three decimal places:

```
ROUNDDOWN( PRICE, 3 )
```

PRICE	RETURN VALUE
12.9936	12.993
15.9949	15.994
-18.8678	-18.867
56.9561	56.956
NULL	NULL

You can round digits to the left of the decimal point by passing a negative integer in the *digits* argument:

```
ROUNDDOWN( PRICE, -2 )
```

PRICE	RETURN VALUE
13242.99	13200.0
1435.99	1400.0
-108.95	-100.0
NULL	NULL

If you pass a decimal value in the *digits* argument, the function rounds it to the nearest integer before evaluating the expression:

```
ROUNDDOWN( PRICE, 0.8 )
```

PRICE	RETURN VALUE
12.99	12.0
56.34	56.0
NULL	NULL

ROUNDUP

Rounds off numbers up to a specified number of digits or decimal places.

Syntax

```
ROUNDUP( number , digits)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
number	Required	Numeric data type. You can enter a valid transformation expression. Use operators to perform arithmetic before you round the values.
digits	Required	Positive or negative integer. If you enter a positive value in *digits*, the function rounds off to this number of decimal places. For example, ROUNDUP(12.99, 1) returns 13.0 and ROUNDUP(15.44, 1) returns 15.5. If you enter a negative value in *digits*, the function rounds off this number of digits to the left of the decimal point, returning an integer. For example, ROUNDUP(12.99, -1) returns 20 and ROUNDUP(15.99, -1) returns 20. If you enter decimal value in *digits*, the function rounds off to the nearest integer before evaluating the expression. For example, ROUNDUP(12.99, 0.8) returns 13.0 because the function rounds 0.8 to 1 and then evaluates the expression.

Return Value

Numeric value

If one of the arguments is NULL, ROUND returns NULL.

Examples

The following expression returns the values in the PRICE port rounded to three decimal places:

```
ROUNDUP( PRICE, 3 )
```

PRICE	RETURN VALUE
12.9936	12.994
15.9949	15.995
-18.8678	-18.868
56.9561	56.957
NULL	NULL

You can round digits to the left of the decimal point by passing a negative integer in the *digits* argument:

```
ROUNDUP( PRICE, -2 )
```

PRICE	RETURN VALUE
13242.99	13300.0
1435.99	1500.0
-108.95	-200.0
NULL	NULL

If you pass a decimal value in the *digits* argument, the function rounds it to the nearest integer before evaluating the expression:

```
ROUNDUP( PRICE, 0.8 )
```

PRICE	RETURN VALUE
12.99	13.0
56.34	57.0
NULL	NULL

SECOND

Returns the seconds part of a date as an integer value from 0 to 59.

Syntax

```
SECOND(date)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
date	Required	Date/Time data type. Passes the dates for which you want to return the second value.

Return Value

Integer from 0 to 59.

NULL if a value in the selected port is NULL or the data type in the selected port is not Date/Time.

Examples

The following expression returns the seconds value for each date in the DATE_SHIPPED port.

```
HOUR(DATE_SHIPPED)
```

DATE_SHIPPED	RETURN VALUE
Mar 13 1997 12:00:00AM	0
Sep 2 1997 2:00:11AM	11
Aug 22 1997 12:03:46PM	46
June 3 1997 11:30:44PM	44
NULL	NULL

SQRT

Returns the square root of a positive numeric value.

Syntax

```
SQRT( number )
```

The following table describes the argument for this command:

Argument	Required/ Optional	Description
number	Required	Positive numeric value. Passes the values for which you want to calculate a square root. You can enter a valid transformation expression.

Return Value

Double value

NULL if a value passed to the function is NULL.

Examples

The following expression returns the square root for the values in the Numbers port:

```
SQRT( NUMBERS )
```

NUMBERS	RETURN VALUE
100	10
-100	NULL
NULL	NULL
60.54	7.78074546557076

STARTSWITH

Determines whether a string starts with the characters of a specified string. The function returns True if the string starts with the specified characters. The function returns False if the string does not start with the specified characters.

Note: This function is case-sensitive.

Syntax

```
STARTSWITH(text, starting_text)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
text	Required	Character string. Passes the strings you want to search. You can enter a valid transformation expression. If you pass a numeric value, the function converts it to a character string.
starting_text	Required	Character string. Passes the characters to be searched for at the beginning of *text*.

Return Value

TRUE if the string starts with the specified characters.

FALSE if the string does not start with the specified characters.

Examples

The following expression validates if the first name in the FIRST_NAME column is correct based on the names in the FULL_NAME colum:

```
STARTSWITH( FULL_NAME, FIRST_NAME )
```

FULL_NAME	FIRST_NAME	RETURN VALUE
Tom Miller	TOM	false *(as this function is case-sensitive.)*
Christ Karol	Christ	true
Robert Smith	Robert	true
Frank Johnson	frank	false

You can enter a string in the *starting_text* parameter with single quotes (' '). The following expression validates if the month in the ORDER_DATE column is January. The date format is MM/DD/YYYY:

```
STARTSWITH(ORDER_DATE, '01')
```

ORDER_DATE	RETURN VALUE
01/08/2002	true
02/23/2002	false
11/15/2003	false
01/11/2004	true
01/28/2004	true

STARTSWITH_IGNORE_CASE

Determines whether a string starts with the characters of a specified string. The function returns True if the string starts with the specified characters. The function returns False if the string does not start with the specified characters.
Note: This function is not case-sensitive.

Syntax

```
STARTSWITH_IGNORE_CASE(text, starting_text)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
text	Required	Character string. Passes the strings you want to search. You can enter any valid transformation expression. If you pass a numeric value, the function converts it to a character string.
starting_text	Required	Character string. Passes the characters to be searched for at the beginning of *text*.

Return Value

TRUE if the string starts with the specified characters.

FALSE if the string does not starts with the specified characters.

Examples

The following expression validates if the first name in the FIRST_NAME column is correct based on the names in the FULL_NAME column.

```
STARTSWITH_IGNORE_CASE( FULL_NAME, FIRST_NAME )
```

FULL_NAME	FIRST_NAME	RETURN VALUE
Tom Miller	TOM	true *(as this function is not case-sensitive.)*
Christ Karol	@Christ	false
Robert Smith	Robert	true
Frank Johnson	Jonson	false

You can enter a string in the *starting_text* parameter with single quotes (' '). The following expression validates if the month in the ORDER_DATE column is January. The date format is MM/DD/YYYY.

```
STARTSWITH_IGNORE_CASE(ORDER_DATE, '01')
```

ORDER_DATE	RETURN VALUE
01/08/2002	true
02/23/2002	false
11/15/2003	false
01/11/2004	true
01/28/2004	true

STDDEV

Returns the standard deviation of the numeric values that you pass to this function. STDDEV is used to analyze statistical data.

Syntax

```
STDDEV( value )
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
value	Required	Numeric data type. This function passes the values for which you want to calculate a standard deviation. You can enter a valid transformation expression.

Return Value

Numeric value

NULL if all values passed to the function are NULL.

Nulls

If a single value is NULL, STDDEV ignores it. If all values are NULL, STDDEV returns NULL.

Examples

The following expression calculates the standard deviation of all rows in the TOTAL_SALES port:

```
STDDEV( TOTAL_SALES )
```

TOTAL_SALES	RETURN VALUE
2198.0	3254.60361129688
2256.0	3254.60361129688
3001.0	3254.60361129688
NULL	3254.60361129688
8953.0	3254.60361129688

The following expression calculates the standard deviation of all rows in the SALES port:

```
STDDEV(SALES)
```

SALES	RETURN VALUE
2198.0	0
2198.0	0
2198.0	0

SALES	RETURN VALUE
2198.0	0

The return value is 0 because each row contains the same number (no standard deviation exists). If there is no standard deviation, the return value is 0.

STDDEVIF

Returns the standard deviation of the numeric values that you pass to this function. STDDEVIF is used to analyze statistical data. You can apply a filter to limit the rows to calculate the standard deviation.

Syntax

```
STDDEVIF( value, filter )
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
value	Required	Numeric data type. This function passes the values for which you want to calculate a standard deviation. You can enter a valid transformation expression.
filter	Required	Limits the rows in the search. The filter condition must be a numeric value or evaluate to TRUE, FALSE, or NULL. You can enter a valid transformation expression.

Return Value

Numeric value

NULL if all values passed to the function are NULL or no rows are selected, for example, the filter condition evaluates to FALSE or NULL for all rows.

Nulls

If a single value is NULL, STDDEVIF ignores it. If all values are NULL, STDDEVIF returns NULL.

Examples

The following expression calculates the standard deviation of all rows greater than $2000.00 in the TOTAL_SALES port:

```
STDDEVIF( TOTAL_SALES, TOTAL_SALES>2000 )
```

TOTAL_SALES	RETURN VALUE
2198.0	3254.60361129688
1010.90	3254.60361129688

TOTAL_SALES	RETURN VALUE
2256.0	3254.60361129688
153.88	3254.60361129688
3001.0	3254.60361129688
NULL	3254.60361129688
8953.0	3254.60361129688

The function does not include the values 1010.90 and 153.88 in the calculation because the filter specifies sales greater than $2,000.

The following expression calculates the standard deviation of all rows in the SALES port:

```
STDDEV(SALES)
```

SALES	RETURN VALUE
2198.0	0
2198.0	0
2198.0	0
2198.0	0

The return value is 0 because each row contains the same number (no standard deviation exists). If there is no standard deviation, the return value is 0.

SUBSTITUTE

Replaces one or more instances of an existing text with a new text in a text string.
Note: This function is case-sensitive.

Syntax

```
SUBSTITUTE(within_text, old_text, new_text)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
within_text	Required	Character string. Passes the strings within which the *old_text* strings needs to be searched.
old_text	Required	Character string. Passes the text that you want to replace.
new_text	Required	Character string. Passes the text you want to replace the old_text with.

Return Value

Character

Examples

The following expression replaces the state codes in the ADDRESS column with the state names:

```
SUBSTITUTE( ADDRESS, STATE_CODE, STATE_NAME )
```

ADDRESS	STATE_CODE	STATE_NAME	RETURN VALUE
591 Memorial Dr, Chicopee MA 1020	MA	Massachusetts	591 Memorial Dr, Chicopee Massachusetts 1020
250 Rt 59, Airmont NY 10901	NY	New York	250 Rt 59, Airmont New York 10901
255 W Main St, Avon CT 6001	CT	Connecticut	255 W Main St, Avon Connecticut 6001
630 Coonial Promenade Pkwy, Alabaster AL 35007	AL	Alabama	630 Coonial Promenade Pkwy, Alabaster Alabama 35007
1818 State Route 3, Fulton NY 13069	NY	New York	1818 State Route 3, Fulton New York 13069

The following expression replaces the date separator '/' with '-':

```
SUBSTITUTE( ORDER_DATE, '/', '-' )
```

ORDER_DATE	RETURN VALUE
03/07/2003	03-07-2003
08/02/2004	08-02-2004
11/01/2005	11-01-2005

ORDER_DATE	RETURN VALUE
12/23/2008	12-23-2008
09/28/2010	09-28-2010

SUBSTITUTE_IGNORE_CASE

Replaces one or more instances of an existing text with a new text in a text string.
Note: This function is not case-sensitive.

Syntax

```
SUBSTITUTE_IGNORE_CASE(within_text, old_text, new_text)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
within_text	Required	Character string. Passes the strings within which the *old_text* strings needs to be searched.
old_text	Required	Character string. Passes the text that you want to replace.
new_text	Required	Character string. Passes the text you want to replace the old_text with.

Return Value

Character

Examples

The following expression replaces the state codes in the ADDRESS column with the state names:

```
SUBSTITUTE_IGNORE_CASE( ADDRESS, STATE_CODE, STATE_NAME )
```

ADDRESS	STATE_CODE	STATE_NAME	RETURN VALUE
591 Memorial Dr, Chicopee MA 1020	MA	Massachusetts	591 Memorial Dr, Chicopee Massachusetts 1020
250 Rt 59, Airmont NY 10901	NY	New York	250 Rt 59, Airmont New York 10901
255 W Main St, Avon CT 6001	CT	Connecticut	255 W Main St, Avon Connecticut 6001

ADDRESS	STATE_CODE	STATE_NAME	RETURN VALUE
630 Coonial Promenade Pkwy, Alabaster AL 35007	AL	Alabama	630 CooniAlabama Promenade Pkwy, Alabamaabaster Alabama 35007 (*It replaces all instances of 'AL' as this function is not case-sensitive.*)
1818 State Route 3, Fulton NY 13069	NY	New York	1818 State Route 3, Fulton New York 13069

The following expression replaces the date separator '/' with '-':

```
SUBSTITUTE_IGNORE_CASE( ORDER_DATE, '/', '-' )
```

ORDER_DATE	RETURN VALUE
03/07/2003	03-07-2003
08/02/2004	08-02-2004
11/01/2005	11-01-2005
12/23/2008	12-23-2008
09/28/2010	09-28-2010

SUBSTITUTE_REGEX

Replaces existing text with the new text based on a regular expression pattern. You must use a perl compatible regular expression syntax.

Syntax

```
SUBSTITUTE_REGEX(within_text, old_text, new_text)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
within_text	Required	String data type. Passes the strings within which the *old_text* string needs to be searched.
old_text	Required	String data type. Passes the character string to be replaced. You must use perl compatible regular expression syntax. Enclose the pattern in single quotes.
new_text	Required	String data type. Passes the new character string.

Return Value

String

Examples

The following expression removes additional spaces from the employee name in the EMPLOYEE_NAME port:

```
SUBSTITUTE_REGEX( EMPLOYEE_NAME, '\s+', ' ' )
```

EMPLOYEE_NAME	RETURN VALUE
Adam Smith	Adam Smith
Greg Sanders	Greg Sanders
Sarah Fe	Sarah Fe
Sam Cooper	Sam Cooper

SUM

Returns the sum of all values in the selected port.

Syntax

```
SUM( value )
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
value	Required	Numeric data type. Passes the values you want to add. You can enter a valid transformation expression.

Return Value

Numeric value

NULL if all values passed to the function are NULL.

Nulls

If a single value is NULL, SUM ignores it. If all values passed from the port are NULL, SUM returns NULL.

Examples

The following expression returns the sum of all values in the SALES port:

```
SUM( SALES )
```

SALES	RETURN VALUE
2500.0	13577.0
1900.0	13577.0
1200.0	13577.0
NULL	13577.0
3458.0	13577.0
4519.0	13577.0

SUMIF

Returns the sum of all values in the selected port. You can apply a filter to limit the rows to calculate the total.

Syntax

```
SUMIF( value, filter )
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
value	Required	Numeric data type. Passes the values that you want to add. You can enter a valid transformation expression.
filter	Required	Limits the rows in the search. The filter condition must be a numeric value or evaluate to TRUE, FALSE, or NULL. You can enter a valid transformation expression.

Return Value

Numeric value

NULL if all the values passed to the function are NULL or no rows are selected, for example, the filter condition evaluates to FALSE or NULL for all rows.

Nulls

If a single value is NULL, SUMIF ignores it. If all the values passed from the port are NULL, SUMIF returns NULL.

Examples

The following expression returns the sum of all values greater than 2000 in the SALES port:

```
SUMIF( SALES, SALES > 2000 )
```

SALES	RETURN VALUE
2500.0	10477.0
1900.0	10477.0
1200.0	10477.0
NULL	10477.0
3458.0	10477.0
4519.0	10477.0

TODAY

Returns the current date in UTC.

Syntax

```
TODAY()
```

The TODAY() function syntax has no arguments.

Return Value

Current date.

Examples

If the TODAY() function is run on 8th April, 2021, the function returns the following value.

INPUT	RETURN VALUE
TODAY()	04/08/2021

TRIM

Removes the whites spaces from the start and end of a text string.

Syntax

```
TRIM(text)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
text	Required	Character string. Passes the strings from which you want to remove the leading and trailing spaces.

Return Value

Character

NULL if a value in the selected port is NULL.

Examples

The following expression removes the leading and trailing spaces from the names in the FULL_NAME column:

```
TRIM( FULL_NAME )
```

FULL_NAME	RETURN VALUE
Christ Karol	Christ Karol
Robert Kennedy	Robert Kennedy
Tom Flogger	Tom Flogger
Steve Mcguire	Steve Mcguire
NULL	NULL

TRIM_ALL

Removes the whites spaces from a text string.

Syntax

```
TRIM_ALL(text)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
text	Required	Character string. Passes the strings from which you want to remove the spaces.

Return Value

Character

NULL if a value in the selected port is NULL.

Examples

The following expression removes the spaces from the names in the FULL_NAME column:

```
TRIM_ALL( FULL_NAME )
```

FULL_NAME	RETURN VALUE
Christ Karol	ChristKarol
Robert Kennedy	RobertKennedy
Tom Flogger	TomFlogger
Steve Mcguire	SteveMcguire
NULL	NULL

UNIXTIME_TO_DATE

Converts a Unix time to a date in UTC.

Syntax

```
UNIXTIME_TO_DATE(unixtime)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
unixtime	Required	Unix time. Passes the values for the Unix time that you want to convert to a date in UTC.

Return Value

Date and time in UTC.

Examples

The following expression converts each Unix time in the DATE_SHIPPED_UNIX port to a date in UTC:

```
UNIXTIME_TO_DATE(DATE_SHIPPED_UNIX)
```

DATE_SHIPPED_UNIX	RETURN VALUE
1437264000	07/19/2015
1374537600	07/23/2013
1617700205	04/06/2021 09:10:05
1332501973	03/23/2021 11:26:13

UPPER

Converts lowercase string characters to uppercase.

Syntax

```
UPPER( text )
```

The following table describes the argument for this command:

Argument	Required/ Optional	Description
text	Required	String data type. Passes the values that you want to change to uppercase text. You can enter a valid transformation expression.

Return Value

Uppercase string

NULL if a value passed to the function is NULL.

Examples

The following expression changes all the names in the FIRST_NAME port to uppercase:

```
UPPER( FIRST_NAME )
```

FIRST_NAME	RETURN VALUE
Ramona	RAMONA
NULL	NULL
THOMAS	THOMAS
PierRe	PIERRE
Bernice	BERNICE

VARIANCE

Returns the variance of a value that you pass to the function. VARIANCE is used to analyze statistical data.

Syntax

```
VARIANCE( value )
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
value	Required	Numeric data type. Passes the values for which you want to calculate a variance. You can enter a valid transformation expression.

Return Value

Double value

NULL if all values passed to the function are NULL.

Nulls

If a single value is NULL, VARIANCE ignores it. If all values passed to the function are NULL or no rows are selected, VARIANCE returns NULL.

Examples

The following expression calculates the variance of all rows in the TOTAL_SALES port:

```
VARIANCE( TOTAL_SALES )
```

TOTAL_SALES	RETURN VALUE
2198.0	10592444.6666667
2256.0	10592444.6666667

TOTAL_SALES	RETURN VALUE
3001.0	10592444.6666667
NULL	10592444.6666667
8953.0	10592444.6666667

VARIANCEIF

Returns the variance of a value that you pass to the function. VARIANCEIF is used to analyze statistical data. You can apply a filter to limit the rows to calculate the variance.

Syntax

```
VARIANCEIF( value ,filter )
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
value	Required	Numeric data type. Passes the values for which you want to calculate a variance. You can enter a valid transformation expression.
filter	Required	Limits the rows in the search. The filter condition must be a numeric value or evaluate to TRUE, FALSE, or NULL. You can enter a valid transformation expression.

Return Value

Double value

NULL if all values passed to the function are NULL or no rows are selected, for example, the *filter* evaluates to FALSE or NULL for all rows.

Nulls

If a single value is NULL, VARIANCEIF ignores it. If all values passed to the function are NULL or no rows are selected, VARIANCEIF returns NULL.

Examples

The following expression calculates the variance of all rows less than 100 in the SALES port:

```
VARIANCEIF( SALES, SALES<100 )
```

SALES	RETURN VALUE
12	2137
112	2137

SALES	RETURN VALUE
97	2137
NULL	2137
23	2137

YEAR

Returns the year of a date.

Syntax

```
YEAR(date)
```

The following table describes the arguments for this command:

Argument	Required/ Optional	Description
date	Required	Date/Time data type. Passes the dates for which you want to return the year value.

Return Value

Year

NULL if a value in the selected port is NULL or if the data type in the selected port is not Date/Time.

Examples

The following expression returns the year for each date in the DATE_SHIPPED port:

```
YEAR(DATE_SHIPPED)
```

DATE_SHIPPED	RETURN VALUE
Mar 13 1997 12:00:00AM	1997
Sep 2 1998 2:00:01AM	1998
Aug 22 2000 12:00:00PM	2000
June 3 2005 11:30:44PM	2005
NULL	NULL

INDEX

A

ABS function
  description 15
absolute values
  obtaining 15
ADD_TO_DATE function
  description 16
aggregate functions
  COUNT 21
  COUNTDISTINCTIF 22
  MAX (string) 48
  MIN (numbers) 53
  STDDEV 74
  SUM 80
  VARIANCE 86
AND function
  description 16
AVGIF function
  description 18

C

character functions
  CONCATENATE function 19
CONCATENATE function
  description 19
  inserting single quotes using 19
concatenating
  strings 19
COUNT function
  description 21
COUNTDISTINCTIF function
  description 22

D

date functions
  ADD_TO_DATE 16
  DATE_DIFF 25
DATE_DIFF function
  date functions 25
date/time values
  adding 16

E

EXP function
  description 33
exponent values
  calculating 33
  returning 61

F

Filter transformation
  using ISNULL function 43

H

high precision
  ABS 15
  ABS function 15
  EXP 33
  LOG 47
  POWER 61
  ROUND (numbers) 65
  ROUND function 65
  STDDEV function 74
  SUM 80
  SUM function 80

I

ISNULL function
  description 43

L

literals
  single quotes in 19
LN function
  description 46
LOG function
  description 47
logarithm
  returning 46, 47
logic functions
  AND 16

M

MAX (string) function
  description 48
MIN (numbers) function
  description 53

N

null values
  checking for 43
  ISNULL 43
numbers
  rounding 65

numeric functions
ABS 15
EXP 33
LN 46
LOG 47
POWER 61
ROUND (numbers) 65
SQRT 70
numeric values
returning absolute value 15
returning logarithms 46, 47
returning square root 70
returning standard deviation 74

P

POWER function
description 61

R

ROUND (numbers) function
description 65
rounding
numbers 65
rows
counting 21, 22

S

single quotes in string literals
using CHR and CONCATENATE functions 19
SQRT function
description 70
square root
returning 70
standard deviation
returning 74
Statistics functions
AVGIF 18
STDDEV function
description 74
string literals
single quotes in 19
string values
returning maximum 48
returning minimum 53
strings
concatenating 19
sum
returning 80
SUM function
description 80

T

test functions
ISNULL 43

V

VARIANCE function
description 86