**BLANK**

Returns a blank.

**Syntax**

DAX Copy

BLANK()

**Return value**

A blank.

Example:

=IF( SUM(InternetSales\_USD[SalesAmount\_USD])= 0 , BLANK() , SUM(ResellerSales\_USD[SalesAmount\_USD])/SUM(InternetSales\_USD[SalesAmount\_USD])

**CODE**

Returns a numeric code for the first character in a text string. The returned code corresponds to the character set used by your computer.

| **Operating environment** | **Character set** |
| --- | --- |
| Macintosh | Macintosh character set |
| Windows | ANSI |

**Syntax**

DAX Copy

CODE(text)

**Parameters**

| **Term** | **Definition** |
| --- | --- |
| text | The text for which you want the code of the first character. |

**Return value**

A numeric code for the first character in a text string.

Example:

=CODE("A")

**COMBINEVALUES**

The COMBINEVALUES function joins two or more text strings into one text string. The primary purpose of this function is to support multi-column relationships in DirectQuery models, see **Remarks** for details.

**Syntax**

DAX Copy

COMBINEVALUES(<delimiter>, <expression>, <expression>[, <expression>]…)

**Parameters**

| **Term** | **Definition** |
| --- | --- |
| delimiter | A separator to use during concatenation. Must be a constant value. |
| expression | A DAX expression whose value will be be joined into a single text string. |

**Return value**

The concatenated string.

Example:

EVALUATE DISTINCT(SELECTCOLUMNS(DimDate, "Month", COMBINEVALUES(",", [MonthName], [CalendarYear])))

**CONCATENATE**

Joins two text strings into one text string.

**Syntax**

DAX Copy

CONCATENATE(<text1>, <text2>)

**Parameters**

| **Term** | **Definition** |
| --- | --- |
| text1, text2 | The text strings to be joined into a single text string. Strings can include text or numbers.  You can also use column references. |

**Return value**

The concatenated string.

=CONCATENATE("Hello ", "World")

=CONCATENATE(Customer[LastName], CONCATENATE(", ", Customer[FirstName]))

**CONCATENATEX**

Concatenates the result of an expression evaluated for each row in a table.

**Syntax**

DAX Copy

CONCATENATEX(<table>, <expression>, [delimiter])

**Parameters**

| **Term** | **Definition** |
| --- | --- |
| table | The table containing the rows for which the expression will be evaluated. |
| expression | The expression to be evaluated for each row of the table. |
| delimiter | (optional) A separator to use during concatenation. |

**Return value**

A text string.

CONCATENATEX(Employees, [FirstName] & “ “ & [LastName], “,”)

**EXACT**

Compares two text strings and returns TRUE if they are exactly the same, FALSE otherwise. EXACT is case-sensitive but ignores formatting differences. You can use EXACT to test text being entered into a document.

**Syntax**

DAX Copy

EXACT(<text1>,<text2>)

**Parameters**

| **Term** | **Definition** |
| --- | --- |
| text1 | The first text string or column that contains text. |
| text2 | The second text string or column that contains text. |

**Property Value/Return value**

True or false. (Boolean)

=EXACT([Column1],[Column2])

**FIND**

Returns the starting position of one text string within another text string. FIND is case-sensitive.

**Syntax**

DAX Copy

FIND(<find\_text>, <within\_text>[, [<start\_num>][, <NotFoundValue>]])

**Parameters**

| **Term** | **Definition** |
| --- | --- |
| find\_text | The text you want to find. Use double quotes (empty text) to match the first character in **within\_text**. |
| within\_text | The text containing the text you want to find. |
| start\_num | (optional) The character at which to start the search; if omitted, **start\_num** = 1. The first character in **within\_text** is character number 1. |
| NotFoundValue | (optional) The value that should be returned when the operation does not find a matching substring, typically 0, -1, or BLANK(). |

**Property Value/Return value**

Number that shows the starting point of the text string you want to find.

=FIND("BMX","line of BMX racing goods")

**FIXED**

Rounds a number to the specified number of decimals and returns the result as text. You can specify that the result be returned with or without commas.

**Syntax**

DAX Copy

FIXED(<number>, <decimals>, <no\_commas>)

**Parameters**

| **Term** | **Definition** |
| --- | --- |
| number | The number you want to round and convert to text, or a column containing a number. |
| decimals | (optional) The number of digits to the right of the decimal point; if omitted, 2. |
| no\_commas | (optional) A logical value: if 1, do not display commas in the returned text; if 0 or omitted, display commas in the returned text. |

**Property Value/Return value**

A number represented as text.

=FIXED([PctCost],3,1)

**FORMAT**

Converts a value to text according to the specified format.

**Syntax**

DAX Copy

FORMAT(<value>, <format\_string>)

**Parameters**

| **Term** | **Definition** |
| --- | --- |
| value | A value or expression that evaluates to a single value. |
| format\_string | A string with the formatting template. |

**Return value**

A string containing **value** formatted as defined by **format\_string**.

## Format specifications

The following table identifies characters you can use to create user-defined number formats.

| **Format specification** | **Description** |
| --- | --- |
| None | Displays the number with no formatting. |
| **0** (zero character) | Digit placeholder. Displays a digit or a zero. If the expression has a digit in the position where the zero appears in the format string, displays the digit; otherwise, displays a zero in that position.  If the number has fewer digits than there are zeros (on either side of the decimal) in the format expression, displays leading or trailing zeros. If the number has more digits to the right of the decimal separator than there are zeros to the right of the decimal separator in the format expression, rounds the number to as many decimal places as there are zeros. If the number has more digits to the left of the decimal separator than there are zeros to the left of the decimal separator in the format expression, displays the extra digits without modification. |
| **#** | Digit placeholder. Displays a digit or nothing. If the expression has a digit in the position where the # character appears in the format string, displays the digit; otherwise, displays nothing in that position.  This symbol works like the 0 digit placeholder, except that leading and trailing zeros aren't displayed if the number has fewer digits than there are # characters on either side of the decimal separator in the format expression. |
| **.** (dot character) | Decimal placeholder. The decimal placeholder determines how many digits are displayed to the left and right of the decimal separator. If the format expression contains only # characters to the left of this symbol; numbers smaller than 1 begin with a decimal separator. To display a leading zero displayed with fractional numbers, use zero as the first digit placeholder to the left of the decimal separator. In some locales, a comma is used as the decimal separator. The actual character used as a decimal placeholder in the formatted output depends on the number format recognized by your system. Thus, you should use the period as the decimal placeholder in your formats even if you are in a locale that uses a comma as a decimal placeholder. The formatted string will appear in the format correct for the locale. |
| **%** | Percent placeholder. Multiplies the expression by 100. The percent character (%) is inserted in the position where it appears in the format string. |
| **,** (comma character) | Thousand separator. The thousand separator separates thousands from hundreds within a number that has four or more places to the left of the decimal separator. Standard use of the thousand separator is specified if the format contains a thousand separator surrounded by digit placeholders (0 or #).  A thousand separator immediately to the left of the decimal separator (whether or not a decimal is specified) or as the rightmost character in the string means "scale the number by dividing it by 1,000, rounding as needed." Numbers smaller than 1,000 but greater or equal to 500 are displayed as 1, and numbers smaller than 500 are displayed as 0. Two adjacent thousand separators in this position scale by a factor of 1 million, and an additional factor of 1,000 for each additional separator.  Multiple separators in any position other than immediately to the left of the decimal separator or the rightmost position in the string are treated simply as specifying the use of a thousand separator. In some locales, a period is used as a thousand separator. The actual character used as the thousand separator in the formatted output depends on the Number Format recognized by your system. Thus, you should use the comma as the thousand separator in your formats even if you are in a locale that uses a period as a thousand separator. The formatted string will appear in the format correct for the locale.  For example, consider the three following format strings:  "#,0.", which uses the thousands separator to format the number 100 million as the string "100,000,000".  "#0,.", which uses scaling by a factor of one thousand to format the number 100 million as the string "100000".  "#,0,.", which uses the thousands separator and scaling by one thousand to format the number 100 million as the string "100,000". |
| **:** (colon character) | Time separator. In some locales, other characters may be used to represent the time separator. The time separator separates hours, minutes, and seconds when time values are formatted. The actual character used as the time separator in formatted output is determined by your system settings. |
| **/** (forward slash character) | Date separator. In some locales, other characters may be used to represent the date separator. The date separator separates the day, month, and year when date values are formatted. The actual character used as the date separator in formatted output is determined by your system settings. |
| **E-** , **E+** , **e-** , **e+** | Scientific format. If the format expression contains at least one digit placeholder (0 or #) to the left of E-, E+, e-, or e+, the number is displayed in scientific format and E or e is inserted between the number and its exponent. The number of digit placeholders to the left determines the number of digits in the exponent. Use E- or e- to place a minus sign next to negative exponents. Use E+ or e+ to place a minus sign next to negative exponents and a plus sign next to positive exponents. You must also include digit placeholders to the right of this symbol to get correct formatting. |
| **-+$()** | Literal characters. These characters are displayed exactly as typed in the format string. To display a character other than one of those listed, precede it with a backslash (\) or enclose it in double quotation marks (" "). |
| **\** (backward slash character) | Displays the next character in the format string. To display a character that has special meaning as a literal character, precede it with a backslash (\). The backslash itself isn't displayed. Using a backslash is the same as enclosing the next character in double quotation marks. To display a backslash, use two backslashes.  Examples of characters that can't be displayed as literal characters are the date-formatting and time-formatting characters (a, c, d, h, m, n, p, q, s, t, w, y, /, and :), the numeric-formatting characters (#, 0, %, E, e, comma, and period), and the string-formatting characters (@, &, <, >, and !). |
| **"ABC"** | Displays the string inside the double quotation marks (" "). To include a string in the style argument from within code, you must use Chr(34) to enclose the text (34 is the character code for a quotation mark (")). |
| **"General Date"** | Displays a date and/or time. For example, 3/12/2008 11:07:31 AM. Date display is determined by your application's current culture value. |
| **"Long Date" or "Medium Date"** | Displays a date according to your current culture's long date format. For example, Wednesday, March 12, 2008. |
| **"Short Date"** | Displays a date using your current culture's short date format. For example, 3/12/2008. |
| **"Long Time" or** | Displays a time using your current culture's long time format; typically includes hours, minutes, seconds. For example, 11:07:31 AM. |
| **"Medium Time"** | Displays a time in 12 hour format. For example, 11:07 AM. |
| **"Short Time"** | Displays a time in 24 hour format. For example, 11:07. |
| **(:)** | Time separator. In some locales, other characters may be used to represent the time separator. The time separator separates hours, minutes, and seconds when time values are formatted. The actual character that is used as the time separator in formatted output is determined by your application's current culture value. |
| **(/)** | Date separator. In some locales, other characters may be used to represent the date separator. The date separator separates the day, month, and year when date values are formatted. The actual character that is used as the date separator in formatted output is determined by your application's current culture. |
| **(%)** | Used to indicate that the following character should be read as a single-letter format without regard to any trailing letters. Also used to indicate that a single-letter format is read as a user-defined format. See what follows for additional details. |
| **d** | Displays the day as a number without a leading zero (for example, 1). Use %d if this is the only character in your user-defined numeric format. |
| **dd** | Displays the day as a number with a leading zero (for example, 01). |
| **ddd** | Displays the day as an abbreviation (for example, Sun). |
| **dddd** | Displays the day as a full name (for example, Sunday). |
| **M** | Displays the month as a number without a leading zero (for example, January is represented as 1). Use %M if this is the only character in your user-defined numeric format. |
| **MM** | Displays the month as a number with a leading zero (for example, 01/12/01). |
| **MMM** | Displays the month as an abbreviation (for example, Jan). |
| **MMMM** | Displays the month as a full month name (for example, January). |
| **gg** | Displays the period/era string (for example, A.D.). |
| **h** | Displays the hour as a number without leading zeros using the 12-hour clock (for example, 1:15:15 PM). Use %h if this is the only character in your user-defined numeric format. |
| **hh** | Displays the hour as a number with leading zeros using the 12-hour clock (for example, 01:15:15 PM). |
| **H** | Displays the hour as a number without leading zeros using the 24-hour clock (for example, 1:15:15). Use %H if this is the only character in your user-defined numeric format. |
| **HH** | Displays the hour as a number with leading zeros using the 24-hour clock (for example, 01:15:15). |
| **m** | Displays the minute as a number without leading zeros (for example, 12:1:15). Use %m if this is the only character in your user-defined numeric format. |
| **mm** | Displays the minute as a number with leading zeros (for example, 12:01:15). |
| **s** | Displays the second as a number without leading zeros (for example, 12:15:5). Use %s if this is the only character in your user-defined numeric format. |
| **ss** | Displays the second as a number with leading zeros (for example, 12:15:05). |
| **AM/PM** | Use the 12-hour clock and display an uppercase AM with any hour before noon; display an uppercase PM with any hour between noon and 11:59 P.M. |
| **am/pm** | Use the 12-hour clock and display a lowercase AM with any hour before noon; display a lowercase PM with any hour between noon and 11:59 P.M. |
| **A/P** | Use the 12-hour clock and display an uppercase A with any hour before noon; display an uppercase P with any hour between noon and 11:59 P.M. |
| **a/p** | Use the 12-hour clock and display a lowercase A with any hour before noon; display a lowercase P with any hour between noon and 11:59 P.M. |
| **AMPM** | Use the 12-hour clock and display the AM string literal as defined by your system with any hour before noon; display the PM string literal as defined by your system with any hour between noon and 11:59 P.M. AMPM can be either uppercase or lowercase, but the case of the string displayed matches the string as defined by your system settings. The default format is AM/PM. |
| **y** | Displays the year number (0-9) without leading zeros. Use %y if this is the only character in your user-defined numeric format. |
| **yy** | Displays the year in two-digit numeric format with a leading zero, if applicable. |
| **yyy** | Displays the year in four-digit numeric format. |
| **yyyy** | Displays the year in four-digit numeric format. |
| **z** | Displays the timezone offset without a leading zero (for example, -8). Use %z if this is the only character in your user-defined numeric format. |
| **zz** | Displays the timezone offset with a leading zero (for example, -08) |
| **zzz** | Displays the full timezone offset (for example, -08:00) |

**LEFT**

Returns the specified number of characters from the start of a text string.

**Syntax**

DAX Copy

LEFT(<text>, <num\_chars>)

**Parameters**

| **Term** | **Definition** |
| --- | --- |
| text | The text string containing the characters you want to extract, or a reference to a column that contains text. |
| num\_chars | (optional) The number of characters you want LEFT to extract; if omitted, 1. |

**Property Value/Return value**

A text string.

=CONCATENATE(LEFT('Reseller'[ResellerName],LEFT(GeographyKey,3))

**LEN**

Returns the number of characters in a text string.

**Syntax**

DAX Copy

LEN(<text>)

**Parameters**

| **Term** | **Definition** |
| --- | --- |
| text | The text whose length you want to find, or a column that contains text. Spaces count as characters. |

**Return value**

A whole number indicating the number of characters in the text string.

=LEN([AddressLine1])+LEN([AddressLin2])

**LOWER**

Converts all letters in a text string to lowercase.

**Syntax**

DAX Copy

LOWER(<text>)

**Parameters**

| **Term** | **Definition** |
| --- | --- |
| text | The text you want to convert to lowercase, or a reference to a column that contains text. |

**Property Value/Return value**

Text in lowercase.

=LOWER('New Products'[ProductCode])

**MID**

Returns a string of characters from the middle of a text string, given a starting position and length.

**Syntax**

DAX Copy

MID(<text>, <start\_num>, <num\_chars>)

**Parameters**

| **Term** | **Definition** |
| --- | --- |
| text | The text string from which you want to extract the characters, or a column that contains text. |
| start\_num | The position of the first character you want to extract. Positions start at 1. |
| num\_chars | The number of characters to return. |

**Property Value/Return value**

A string of text of the specified length.

=MID('Reseller'[ResellerName],5,1))

=MID([ResellerName,5])

**REPLACE**

REPLACE replaces part of a text string, based on the number of characters you specify, with a different text string.

**Syntax**

DAX Copy

REPLACE(<old\_text>, <start\_num>, <num\_chars>, <new\_text>)

**Parameters**

| **Term** | **Definition** |
| --- | --- |
| old\_text | The string of text that contains the characters you want to replace, or a reference to a column that contains text. |
| start\_num | The position of the character in **old\_text** that you want to replace with **new\_text**. |
| num\_chars | The number of characters that you want to replace. **Warning:** If the argument, *num\_chars*, is a blank or references a column that evaluates to a blank, the string for *new\_text* is inserted at the position, *start\_num*, without replacing any characters. This is the same behavior as in Excel. |
| new\_text | The replacement text for the specified characters in **old\_text**. |

**Property Value/Return value**

A text string.

=REPLACE('New Products'[Product Code],1,2,"OB")

**REPT**

Repeats text a given number of times. Use REPT to fill a cell with a number of instances of a text string.

**Syntax**

DAX Copy

REPT(<text>, <num\_times>)

**Parameters**

| **Term** | **Definition** |
| --- | --- |
| text | The text you want to repeat. |
| num\_times | A positive number specifying the number of times to repeat text. |

**Property Value/Return value**

A string containing the changes.

=REPT("85",3)

**RIGHT**

RIGHT returns the last character or characters in a text string, based on the number of characters you specify.

**Syntax**

DAX Copy

RIGHT(<text>, <num\_chars>)

**Parameters**

| **Term** | **Definition** |
| --- | --- |
| text | The text string that contains the characters you want to extract, or a reference to a column that contains text. |
| num\_chars | (optional) The number of characters you want RIGHT to extract; is omitted, 1. You can also use a reference to a column that contains numbers. |

If the column reference does not contain text, it is implicitly cast as text.

**Property Value/Return value**

A text string containing the specified right-most characters.

=RIGHT('New Products'[ProductCode],2)

=RIGHT('New Products'[ProductCode],[MyCount])

# SEARCH

Returns the number of the character at which a specific character or text string is first found, reading left to right. Search is case-insensitive and accent sensitive.

## Syntax

DAXCopy

SEARCH(<find\_text>, <within\_text>[, [<start\_num>][, <NotFoundValue>]])

### Parameters

| **Term** | **Definition** |
| --- | --- |
| find\_text | The text that you want to find.  You can use wildcard characters — the question mark (?) and asterisk (\*) — in **find\_text**. A question mark matches any single character; an asterisk matches any sequence of characters. If you want to find an actual question mark or asterisk, type a tilde (~) before the character. |
| within\_text | The text in which you want to search for **find\_text**, or a column containing text. |
| start\_num | (optional) The character position in **within\_text** at which you want to start searching. If omitted, 1. |
| NotFoundValue | (optional) The value that should be returned when the operation does not find a matching substring, typically 0, -1, or BLANK(). |

## Return value

The number of the starting position of the first text string from the first character of the second text string.

=SEARCH("n","printer")

= IFERROR(SEARCH("-",[PostalCode]),-1)

# SUBSTITUTE

Replaces existing text with new text in a text string.

## Syntax

DAXCopy

SUBSTITUTE(<text>, <old\_text>, <new\_text>, <instance\_num>)

### Parameters

| **Term** | **Definition** |
| --- | --- |
| text | The text in which you want to substitute characters, or a reference to a column containing text. |
| old\_text | The existing text that you want to replace. |
| new\_text | The text you want to replace **old\_text** with. |
| instance\_num | (optional) The occurrence of **old\_text** you want to replace. If omitted, every instance of **old\_text** is replaced |

## Property Value/Return value

A string of text.

=SUBSTITUTE([Product Code], "NW", "PA")

# TRIM

Removes all spaces from text except for single spaces between words.

## Syntax

DAXCopy

TRIM(<text>)

### Parameters

| **Term** | **Definition** |
| --- | --- |
| **text** | The text from which you want spaces removed, or a column that contains text. |

## Property Value/Return value

The string with spaces removed.

=TRIM("A column with trailing spaces. ")

=LEN([Calculated Column 1])

# UNICHAR

Returns the Unicode character referenced by the numeric value.

## Syntax

DAXCopy

UNICHAR(number)

### Parameters

| **Term** | **Definition** |
| --- | --- |
| number | The Unicode number that represents the character. |

## Return value

A character represented by the Unicode number

=UNICHAR(65)

=UNICHAR(9733)

# UPPER

Converts a text string to all uppercase letters.

## Syntax

DAXCopy

UPPER (<text>)

### Parameters

| **Term** | **Definition** |
| --- | --- |
| text | The text you want converted to uppercase, or a reference to a column that contains text. |

## Property Value/Return value

Same text, in uppercase.

=UPPER(['New Products'[Product Code])

# VALUE

* 12/09/2018
* 2 minutes to read

* + [[https://github.com/mikekinsman.png?size=32](https://github.com/MicrosoftDocs/query-docs-pr/blob/live/query-languages/dax/value-function-dax.md)](https://github.com/MicrosoftDocs/query-docs-pr/blob/live/query-languages/dax/value-function-dax.md" \o "2 Contributors)

Converts a text string that represents a number to a number.

## Syntax

DAXCopy

VALUE(<text>)

### Parameters

| **Term** | **Definition** |
| --- | --- |
| text | The text to be converted. |

## Return value

The converted number in decimal data type.