DATEDIF Function

Contents:

Calculates the difference between two valid date values for the specified units of measure.

- Inputs must be column references.
- The first value is used as the baseline to compare the date values.
- Results are calculated to the integer value that is closest to and lower than the exact total; remaining decimal values are dropped.

Wrangle vs. SQL: This function is part of Wrangle , a proprietary data transformation language. Wrangle is not SQL. For more information, see *Wrangle Language*.

Basic Usage

```
datedif(StartDate, EndDate, month)
```

Output: Returns the number of full months that have elapsed between `StartDate` and `EndDate`.

Syntax and Arguments

```
datedif(date1,date2,date_units)
```

Argument	Required?	Data Type	Description
date1	Y	datetime	Starting date to compare
date2	Y	datetime	Ending date to compare
date_units	Y	string	String literal representing the date units to use in the comparison

For more information on syntax standards, see *Language Documentation Syntax Notes*.

date1, date2

Date values to compare using the `date_units` units. If `date2` > `date1`, then results are positive.

- Date values must be column references.
- If `date1` and `date2` have a specified time zone offset, the function calculates the difference including the timezone offsets.
- If `date1` does not have a specified time zone but `date2` does, the function uses the local time in the same time zone as `date2` to calculate the difference. The functions returns the difference without the time zone offset.

Usage Notes:

Required?	Data Type	Example Value
Yes	String (Date column reference)	`LastContactDate`

date_units

Unit of date measurement to calculate between the two valid dates.

Usage Notes:

Required?	Data Type	Example Value
Yes	String	`year`

Accepted Value for date units:

- `year`
- `quarter`
- `month`
- `dayofyear`
- `week`
- `day`
- `hour`
- `minute`
- `second`
- `millisecond`

Examples

Tip: For additional examples, see *Common Tasks*.

Example - aged orders

This example illustrates how to calculate the number of days that have elapsed between the order date and today.

Function:

Item	Description
DATEDIF Function	Calculates the difference between two valid date values for the specified units of measure.
TODAY Function	Derives the value for the current date in UTC time zone. You can specify a different time zone by optional parameter.
IF Function	The `IF` function allows you to build if/then/else conditional logic within your transforms.

Source:

For the orders in the following set, you want to charge interest for those ones that are older than 90 days.

OrderId	OrderDate	Amount
1001	1/31/16	1000

1002	11/15/15	1000
1003	12/18/15	1000
1004	1/15/16	1000

Transformation:

The first step is to create a column containing today's (03/03/16) date value:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`TODAY()`
Parameter: New column name	`'Today'`

You can now use this value as the basis for computing the number of elapsed days for each invoice:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`DATEDIF(OrderDate, Today, day)`

The age of each invoice in days is displayed in the new column. Now, you want to add a little bit of information to this comparison. Instead of just calculating the number of days, you could write out the action to undertake. Replace the above with the following:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`IF((DATEDIF(OrderDate, Today, day) > 90),'Charge interest','no action')`
Parameter: New column name	`'TakeAction'`

To be fair to your customers, you might want to issue a notice at 45 days that the invoice is outstanding. You can replace the above with the following:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`IF(DATEDIF(OrderDate, Today, day) > 90,'Charge interest',IF (DATEDIF(OrderDate, Today, day) > 45),'Send letter','no action'))`
Parameter: New column name	`'TakeAction'`

By using nested instances of the `IF` function, you can generate multiple results in the `TakeAction` column.

For the items that are over 90 days old, you want to charge 5% interest. You can do the following:

Transformation Name	`Edit column with formula`
Parameter: Columns	`Amount`
Parameter: Formula	`IF(TakeAction == 'Charge interest',Amount * 1.05, Amount)`

The above sets the value in the `Amount` column based on the conditional of whether the `TakeAction` column value is `Charge interest`. If so, apply 5% interest to the value in the `Amount` column.

Results:

OrderId	OrderDate	Amount	Today	TakeAction
1001	1/31/16	1000	03/03/16	no action
1002	11/15/15	1050	03/03/16	Charge interest
1003	12/18/15	1000	03/03/16	Send letter
1004	1/15/16	1000	03/03/16	Send letter

Example - dayofyear Calculations

This example demonstrates how `dayofyear` is calculated using the `DATEDIF` function, specifically how leap years and leap days are handled. Below, you can see some example dates. The year 2012 was a leap year.

Source:

dateId	d1	d2	Notes
1	1/1/10	10/10/10	Same year; no leap year
2	1/1/10	10/10/11	Different years; no leap year
3	10/10/11	1/1/10	Reverse dates of previous row
4	2/28/11	4/1/11	Same year; no leap year;
5	2/28/12	4/1/12	Same year; leap year; spans leap day
6	2/29/12	4/1/12	Same year; leap year; d1 = leap day
7	2/28/11	2/29/12	Diff years; d2 = leap day; converted to March 1 in d1 year

Transformation:

In this case, the transform is simple:

Transformation Name	`New formula`
Parameter: Formula type	`Single row formula`
Parameter: Formula	`datedif(d1,d2,dayofyear)`
Parameter: New column name	`'datedifs'`

Results:

dateId	d1	d2	datedifs	Notes
1	1/1/10	10/10/10	282	Same year; no leap year

2	1/1/10	10/10/11	282	Different years; no leap year
3	10/10/11	1/1/10	-282	Reverse dates of previous row
4	2/28/11	4/1/11	32	Same year; no leap year;
5	2/28/12	4/1/12	33	Same year; leap year; spans leap day
6	2/29/12	4/1/12	32	Same year; leap year; d1 = leap day
7	2/28/11	2/29/12	1	Diff years; d2 = leap day; converted to March 1 in d1 year

Rows 1 - 3:

- Row 1 provides the baseline calc.
- In Row 2, the same days of the year are used, but the year is different by a count of 1. However, since we are computing `dayofyear` the result is the same as for Row 1.

> **NOTE:** When computing `dayofyear`, the year value for `d2` is converted to the year of `d1`. The difference is then computed.

- Row 3 represents the reversal of dates in Row 2.

> **NOTE:** Negative values for a `dayofyear` calculation indicate that `d2` occurs earlier in the calendar than `d1`, ignoring year.

Rows 4 - 7: Leap years

- Row 4 provides a baseline calculation for a non-leap year.
- Row 5 uses the same days of year as Row 4, but the year (2012) is a leap year. Dates span a leap date (February 29). Note that the `DATEDIF` result is 1 more than the value in the previous row.

> **NOTE:** When the two dates span a leap date and the year for `d1` is a leap year, then February 29 is included as part of the calculated result.

- Row 6 moves date 1 forward by one day, so it is now on a leap day date. Result is one less than the previous row, which also spanned leap date.
- Row 7 switches the leap date to `d2`. In this case, `d2` is converted to the year of `d1`. However, since it was a leap day originally, in the year of `d1`, this value is March 1. Thus, the difference between the two dates is `1` .

> **NOTE:** If `d2` is a leap date and the year for `d1` is not a leap year, the date used in for `d2` is March 1 in the year of `d1`.