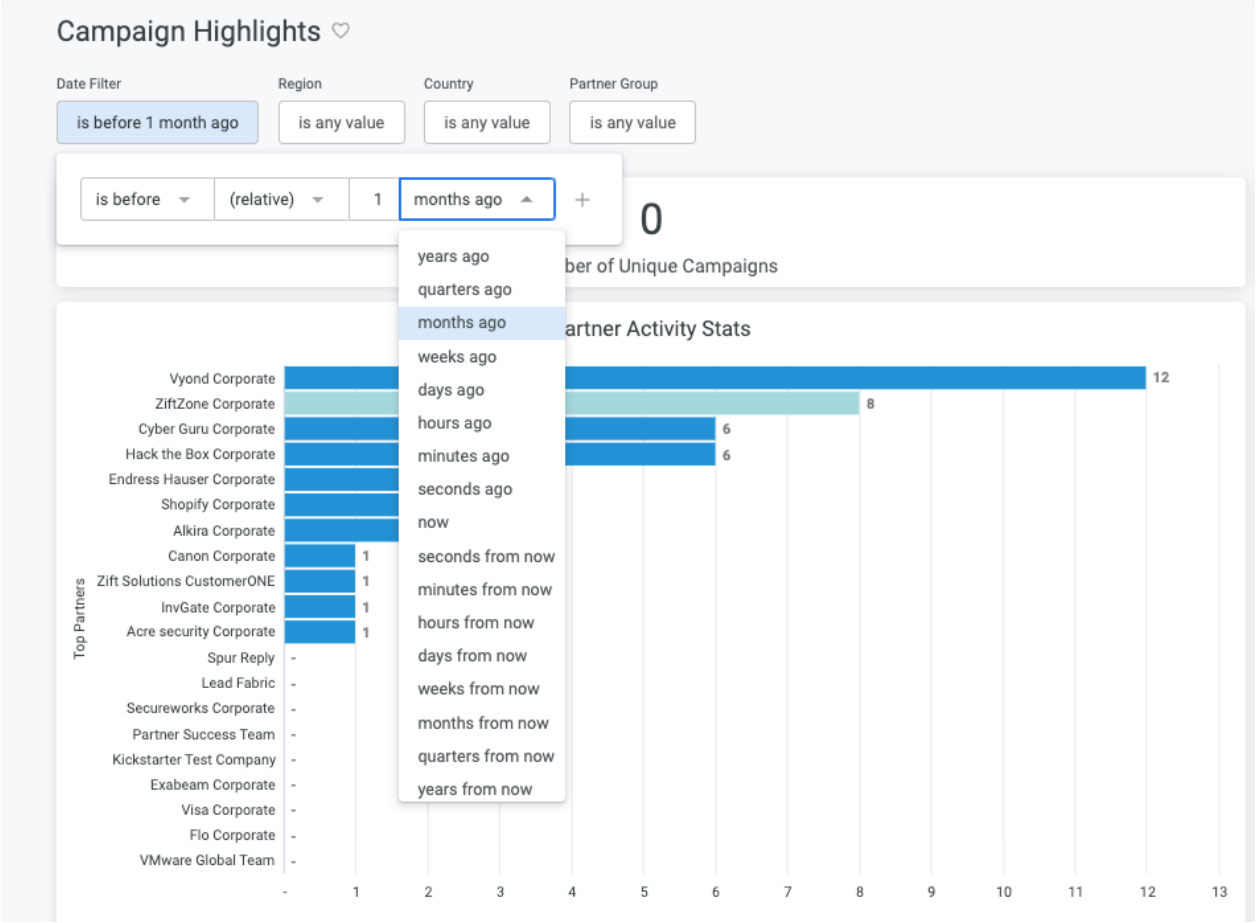

Date and Time

Looker date filtering allows for English phrases to be used instead of SQL date functions.

Basic structure of date and time filters

For the following examples:

- **`{n}`** is an integer.
- **`{interval}`** is a time increment such as hours, days, weeks, or months.

  The phrasing you use determines whether the **`{interval}`** will include partial time periods or only complete time periods. For example, the expression **`3 days`** includes the current, partial day as well as the prior two days. The expression **`3 days ago for 3 days`** includes the previous three complete days and excludes the current, partial day. See the Relative Dates (#relative_dates) section for more information.
- **`{time}`** can specify a time formatted as either YYYY-MM-DD HH:MM:SS or YYYY/MM/DD HH:MM:SS, or a date formatted as either YYYY-MM-DD or YYYY/MM/DD. When using the form

YYYY-MM-DD, be sure to include both digits for the month and day, for example, 2016-01. Truncating a month or day to a single digit is interpreted as an offset, not a date. For example, 2016-1 is interpreted as 2016 minus one year, or 2015.

These are all the possible combinations of date filters:

Combination	Example	Notes
`this {interval}`	`this month`	You can use `this week`, `this month`, `this quarter`, or `this year`. Note that `this day` is not supported. If you want to get data from the current day, you can use `today`.
`{n} {interval}`	`3 days`	
`{n} {interval} ago`	`3 days ago`	
`{n} {interval} ago for {n} {interval}`	`3 months ago for 2 days`	
`before {n} {interval} ago`	`before 3 days ago`	
`before {time}`	`before 2018-01-01 12:00:00`	`before` is not inclusive of the time you specify. The expression `before 2018-01-01` will return data from all dates before 2018-01-01, but it will not return data from 2018-01-01.
`after {time}`	`after 2018-10-05`	`after` is inclusive of the time you specify. So, the expression `after 2018-10-05` will return data from 2018-10-05 and all dates later than 2018-10-05.
`{time} to {time}`	`2018-05-18 12:00:00 to 2018-05-18 14:00:00`	The initial time value is inclusive. The latter time value is not. So the expression `2018-05-18 12:00:00 to 2018-05-18 14:00:00` will return data with the time "2018-05-18 12:00:00" through "2018-05-18 13:59:59".
`this {interval} to {interval}`	`this year to second`	The beginning of each interval is used. For example, the expression `this year to second` returns data from the beginning of the year the query is run through to the beginning of the second the query is run. `this week to day` returns data from the beginning of the week the query is run through to the beginning of the day the query is run.

`{time} for {n} {interval}`	`2018-01-01 12:00:00 for 3 days`	
`today`	`today`	
`yesterday`	`yesterday`	
`tomorrow`	`tomorrow`	
`{day of week}`	`Monday`	Specifying a day of week with a **Dimension Group Date** field returns the most recent date that matches the specified day of week. For example, the expression **`Dimension Group Date matches (advanced) Monday`** returns the most recent Monday. You can also use **`{day of week}`** with the **`before`** and **`after`** keywords in this context. For example, the expression **`Dimension Group Date matches (advanced) after Monday`** returns the most recent Monday and everything after the most recent Monday. The expression **`Dimension Group Date matches (advanced) before Monday`** returns every day before the most recent Monday, but it does not return the most recent Monday. Specifying a day of the week with a **Dimension Group Day of Week** field returns every day that matches the specified day of week. So the expression **`Dimension Group Day of Week matches (advanced) Monday`** returns every Monday.
`next {week, month, quarter, fiscal quarter, year, fiscal year}`	`next week`	The **`next`** keyword is unique in that it requires one of the intervals listed previously and won't work with other intervals.
`{n} {interval} from now`	`3 days from now`	
`{n} {interval} from now for {n} {interval}`	`3 days from now for 2 weeks`	

Date filters can also be combined together:

- **To get OR logic:** Type multiple conditions into the same filter, separated by commas. For

example, `today, 7 days ago` means "today or 7 days ago".

- **To get AND logic:** Type your conditions, one by one, into multiple date or time filters. For example, you could put `after 2014-01-01` into a **Created Date** filter, then put `before 2 days ago` into a **Created Time** filter. This would mean "January 1st, 2014 and after, and before 2 days ago".

Absolute dates

Absolute date filters use the specific date values to generate query results. These are useful when creating queries for specific date ranges.

Example	Description
`2018/05/29`	sometime on 2018/05/29
`2018/05/10 for 3 days`	from 2018/05/10 00:00:00 through 2018/05/12 23:59:59
`after 2018/05/10`	2018/05/10 00:00:00 and after
`before 2018/05/10`	before 2018/05/10 00:00:00
`2018/05`	within the entire month of 2018/05
`2018/05 for 2 months`	within the entire months of 2018/05 and 2018/06
`2018/05/10 05:00 for 5 hours`	from 2018/05/10 05:00:00 through 2018/05/10 09:59:59
`2018/05/10 for 5 months`	from 2018/05/10 00:00:00 through 2018/10/09 23:59:59
`2018`	entire year of 2018 (2018/01/01 00:00:00 through 2018/12/31 23:59:59)
`FY2018`	entire fiscal year starting in 2018 (if your Looker developers have specified that your fiscal year (/looker/docs/reference/param-model-fiscal-month-offset) starts in April then this is 2018/04/01 00:00 through 2019/03/31 23:59)
`FY2018-Q1`	first quarter of the fiscal year starting in 2018 (if your Looker developers have specified that your fiscal year (/looker/docs/reference/param-model-fiscal-month-offset) starts in April then this is 2018/04/01 00:00:00 through 2018/06/30 23:59:59)

Relative dates

Relative date filters allow you to create queries with rolling date values relative to the current date. These are useful when creating queries that update each time you run the query.

For all the examples below, assume today is **Friday, 2018/05/18 18:30:02**. In Looker, weeks start on Monday unless you change that setting with `week_start_day` (/looker/docs/reference/param-model-week-start-day).

Seconds

Example	Description
`1 second`	the current second (2018/05/18 18:30:02)
`60 seconds`	60 seconds ago for 60 seconds (2018/05/18 18:29:02 through 2018/05/18 18:30:01)
`60 seconds ago for 1 second`	60 seconds ago for 1 second (2018/05/18 18:29:02)

Minutes

Example	Description
`1 minute`	the current minute (2018/05/18 18:30:00 through 18:30:59)
`60 minutes`	60 minutes ago for 60 minutes (2018/05/18 17:31:00 through 2018/05/18 18:30:59)
`60 minutes ago for 1 minute`	60 minutes ago for 1 minute (2018/05/18 17:30:00 through 2018/05/18 17:30:59)

Hours

Example	Description
`1 hour`	the current hour (2018/05/18 18:00 through 2018/05/18 18:59)
`24 hours`	the same hour of day that was 24 hours ago for 24 hours (2018/05/17 19:00 through 2018/05/18 18:59)
`24 hours ago for 1 hour`	the same hour of day that was 24 hours ago for 1 hour (2018/05/17 18:00 until 2018/05/17 18:59)

Days

Example	Description

`today`	the current day (2018/05/18 00:00 through 2018/05/18 23:59)
`2 days`	all of yesterday and today (2018/05/17 00:00 through 2018/05/18 23:59)
`1 day ago`	just yesterday (2018/05/17 00:00 until 2018/05/17 23:59)
`7 days ago for 7 days`	the last complete 7 days (2018/05/11 00:00 until 2018/05/17 23:59)
`today for 7 days`	the current day, starting at midnight, for 7 days into the future (2018/05/18 00:00 until 2018/05/24 23:59)
`last 3 days`	2 days ago through the end of the current day (2018/05/16 00:00 until 2018/05/18 23:59)
`7 days from now`	7 days in the future (2018/05/25 00:00 until 2018/05/25 23:59)

Weeks

Example	Description
`1 week`	top of the current week going forward (2018/05/14 00:00 through 2018/05/20 23:59)
`this week`	top of the current week going forward (2018/05/14 00:00 through 2018/05/20 23:59)
`before this week`	anytime until the top of this week (before 2018/05/14 00:00)
`after this week`	anytime after the top of this week (2018/05/14 00:00 and later)
`next week`	the following Monday going forward 1 week (2018/05/21 00:00 through 2018/05/27 23:59)
`2 weeks`	a week ago Monday going forward (2018/05/07 00:00 through 2018/05/20 23:59)
`last week`	synonym for "1 week ago"
`1 week ago`	a week ago Monday going forward 1 week (2018/05/07 00:00 through 2018/05/13 23:59)

Months

Example	Description
`1 month`	the current month (2018/05/01 00:00 through 2018/05/31 23:59)
`this month`	synonym for "0 months ago" (2018/05/01 00:00 through 2018/05/31 23:59)
`2 months`	the past two months (2018/04/01 00:00 through 2018/05/31 23:59)

`last month`	all of 2018/04
`2 months ago`	all of 2018/03
`before 2 months ago`	all time before 2018/03/01
`next month`	all of 2018/06
`2 months from now`	all of 2018/07
`6 months from now for 3 months`	2018/11 through 2019/01

Quarters

Example	Description
`1 quarter`	the current quarter (2018/04/01 00:00 through 2018/06/30 23:59)
`this quarter`	synonym for "0 quarters ago" (2018/04/01 00:00 through 2018/06/30 23:59)
`2 quarters`	the past two quarters (2018/01/01 00:00 through 2018/06/30 23:59)

`1 year`	all of the current year (2018/01/01 00:00 through 2018/12/31 23:59)
`this year`	all of the current year (2018/01/01 00:00 through 2018/12/31 23:59)
`next year`	all of the following year (2019/01/01 00:00 through 2019/12/31 23:59)
`2 years`	the past two years (2017/01/01 00:00 through 2018/12/31 23:59)
`last year`	all of 2017
`2 years ago`	all of 2016
`before 2 years ago`	all time before 2016/01/01 (does not include any days between 2016/01/01 and 2016/05/18)

If your Looker developers have specified using a fiscal year (/looker/docs/reference/param-model-fiscal-month-offset) then you can type `fiscal` in these expressions to use a fiscal year instead of a calendar quarter. For example, you can use `last fiscal year`.

Years

Example	Description

`1 year`	all of the current year (2018/01/01 00:00 through 2018/12/31 23:59)
`this year`	all of the current year (2018/01/01 00:00 through 2018/12/31 23:59)
`next year`	all of the following year (2019/01/01 00:00 through 2019/12/31 23:59)
`2 years`	the past two years (2017/01/01 00:00 through 2018/12/31 23:59)
`last year`	all of 2017
`2 years ago`	all of 2016
`before 2 years ago`	all time before 2016/01/01 (does not include any days between 2016/01/01 and 2016/05/18)