

Moving Functions

Alchemer Dashboard is currently waitlist only. [Visit this page](#) to learn more about Dashboard or join the waitlist!

Moving formulas can be used to smooth out any irregularities in your data to easily recognize trends. The larger the interval you set, the more the peaks and valleys are smoothed out. While the smaller the interval, the closer the moving averages are to the actual data points.

Each of the moving formula accepts a measure, two integers to define the window, and one or more optional attributes.

```
formula (measure,integer,integer,[attribute,attribute,...])
```

Only the measure and integer values are required. If you supply both required and optional values, the formula returns the aggregate of the measure over the given window. You should experiment with only a measure and integers, leaving out the attribute, and then adding it back in. This will help you decide which output best meets your use case.

The time window is (`current - Num1...Current + Num2`), including both endpoints. For example, `1,1` has a window size of 3. To see periods in the past, use a negative number for the second endpoint, as in the example `moving_average(sales, 1, -1, date)`.

For more information on how the time windows work, see this chart:

Past			Now	Future			formula
-3	-2	-1		1	2	3	
							<code>moving_sum (revenue, 1, -1, order date)</code>
							<code>moving_sum (revenue, 2, -2, order date)</code>
							<code>moving_sum (revenue, 3, -3, order date)</code>
							<code>moving_sum (revenue, 2, -1, order date)</code>
							<code>moving_sum (revenue, 3, -2, order date)</code>
							<code>moving_sum (revenue, 0, 0, order date)</code>
							<code>moving_sum (revenue, 1, 0, order date)</code>
							<code>moving_sum (revenue, 2, 0, order date)</code>
							<code>moving_sum (revenue, 0, 1, order date)</code>
							<code>moving_sum (revenue, 0, 2, order date)</code>
							<code>moving_sum (revenue, 1, 1, order date)</code>
							<code>moving_sum (revenue, 1, 2, order date)</code>
							<code>moving_sum (revenue, 2, 1, order date)</code>

The moving formulas are the following:

- `moving_average`, for example `moving_average (revenue, 2, 1, customer region)`

Takes a measure, two integers to define the window to aggregate over, and one or more attributes. Returns the average of the measure over the given window. The attributes are the

ordering columns used to compute the moving average.

- `moving_max` , for example `moving_max (complaints, 1, 2, store name)`

Takes a measure, two integers to define the window to aggregate over, and one or more attributes. Returns the maximum of the measure over the given window. The attributes are the ordering columns used to compute the moving maximum.

- `moving_min` , for example `moving_min (defects, 3, 1, product)`

Takes a measure, two integers to define the window to aggregate over, and one or more attributes. Returns the minimum of the measure over the given window. The attributes are the ordering columns used to compute the moving minimum.

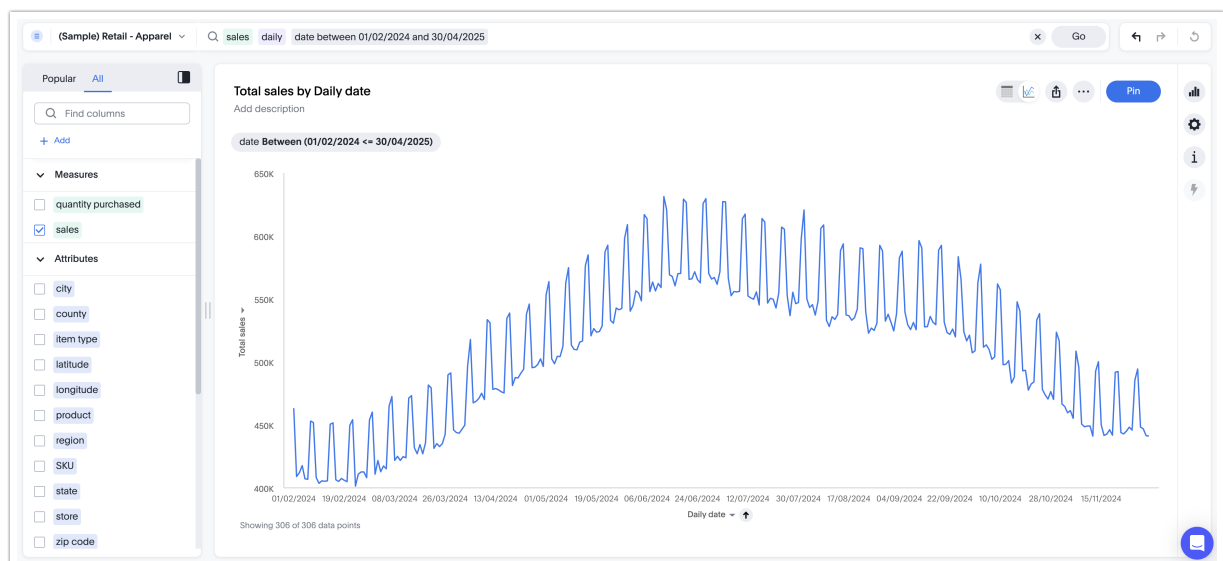
- `moving_sum` , for example `moving_sum (revenue, 1, 1, order date)`

Takes a measure, two integers to define the window to aggregate over, and one or more attributes. Returns the sum of the measure over the given window. The attributes are the ordering columns used to compute the moving sum.

Calculate a Moving Average

This example demonstrates using the `moving_average` formula. To use the moving function in a search:

1. Start a new search, or edit an existing Chart.
2. In the Data panel click **+ Add**, and select **Formula**.



3. Enter the `moving_average` formula, providing a measure, a window, and one or more attributes.

The window includes the previous, current, and next rows. The attributes are the ordering columns

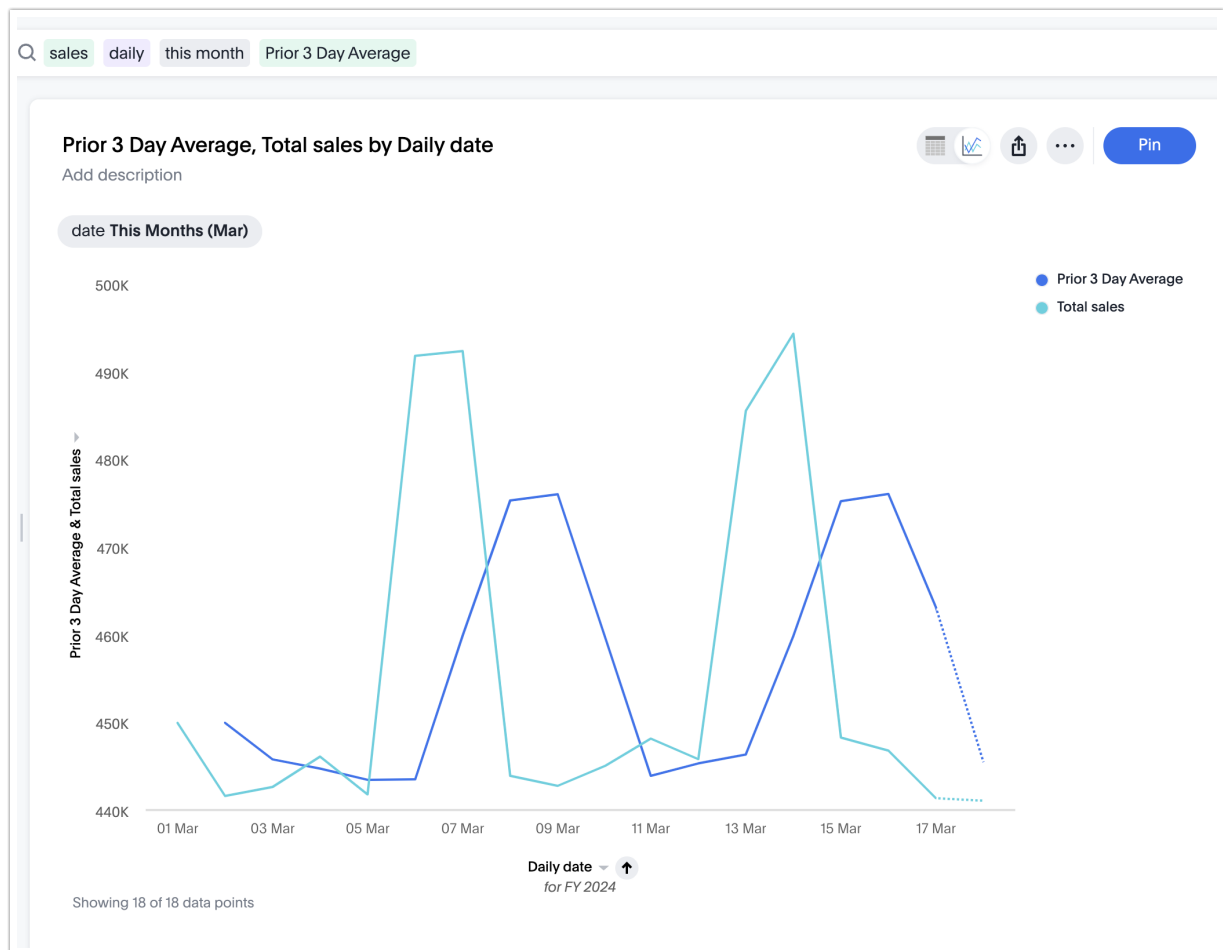
used to compute the moving average. The window is (current - Num1...Current + Num2) with both end points being included in the window. For example, "1,1" will have a window size of 3. To see periods in the past, use a negative number for the second endpoint, as in the example "moving_average(revenue, 1, -1, date)".

The example returns the comparison of the current sales to the prior 3 months average sales.

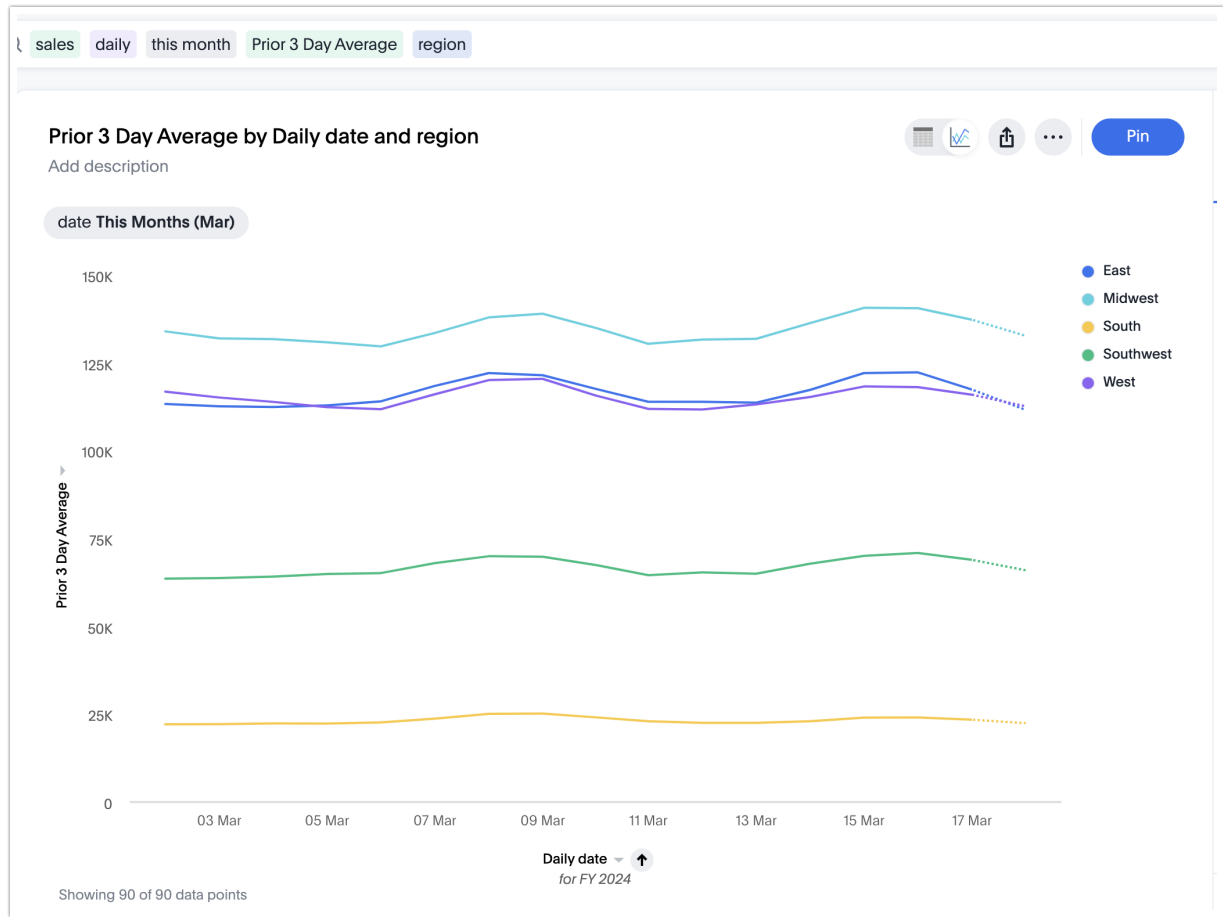
4. Name the formula by entering a title in the top field, and then select **Save**.

The screenshot shows a 'Formula Editor' window. At the top, there is a text input field containing 'Prior 3 Day Average'. Below this is a large text area for the formula, which contains the text '1 moving_average(sales,3,-1,date)'. To the right of the formula area is a search bar with a magnifying glass icon and a close button (X). Below the search bar is a list of categories: Logical, Comparison, Mathematical, Text, and Aggregation, each preceded by a right-pointing chevron. Below the list is the text 'Select an item above to view details'. At the bottom right of the editor are two buttons: 'Cancel' and 'Save'. A green status message 'Formula is ready to save' is visible at the bottom left of the editor area.

5. The formula appears in the search bar and in the table as its own column. A chart displaying the moving average will appear at the bottom.



6. You could also then break down the moving average by region.



Calculate 3 Months Daily Moving Average

1. Start a new search for average monthly sales for the year.

average salesmonthlythis year

Average sales by Monthly date - Fiscal

Add description

date This Years (2023)

Monthly date - Fiscal	Average sales	
Feb 2024	89.27	
Aug 2023	87.52	
Sep 2023	97.14	
Oct 2023	104.2	
Nov 2023	105.91	
Dec 2023	104.7	
Jan 2024	99.87	
Mar 2024	82.42	

2. Name the formula by entering a title in the top field, and then select **Save**.

Formula Editor

Prior 3 Month Daily Average

1 moving_average (group_aggregate(average(sales),query_groups ()), query_filters ()),3,-1,date)

Q Search

> Logical

> Comparison

> Mathematical

> Text

> Aggregation

Select an item above to view details

Formula is ready to save

Cancel

Save

3. A table displaying the 3 months daily moving average will appear at the bottom.

average sales	monthly	this year	Prior 3 Month Daily Average
---------------	---------	-----------	-----------------------------

Average sales, Prior 3 Month Daily Average by Monthly date - Fiscal			
Add description			
date This Years (2023)			
Monthly date - Fiscal	Average sales	Prior 3 Month Daily Average	
Aug 2023	87.52	{Null}	
Sep 2023	97.14	87.52	
Oct 2023	104.2	92.33	
Nov 2023	105.91	96.29	
Dec 2023	104.7	102.42	
Jan 2024	99.87	104.94	
Feb 2024	89.27	103.49	
Mar 2024	82.42	97.95	

Filters and Moving Functions

Filters are applied pre-aggregation.

1.

Create a moving function to calculate the total sales for the last 3 months.

Formula Editor

3 months sum

1 `moving_sum(sales, 2, 0, date)`

Q Search

> Logical

> Comparison

> Mathematical

> Text

> Aggregation

Select an item above to view details

Formula is ready to save

Cancel

Save

2. You can use this to create a ratio for what percentage of sales does the current month contribute to in the total sales for the last 3 months. In this case, just adding a `date` filter will lead to incorrect values. This is because the year filter is applied prior to the `moving_sum` calculation.

In order to ensure that the filters are ignored and that answer is correct, the moving function must be wrapped in a group aggregate function.

Formula Editor

Prior 3 Months Sales

1 group_aggregate (moving_sum(sales, 2,0, date),query_groups(),query_filters()-{date})

Q Search

×

> Logical

> Comparison

> Mathematical

> Text

> Aggregation

Select an item above to view details

Formula is ready to save

Cancel

Save

Calculate a Moving Sum of a Unique Count

Alchemer Dashboard doesn't have a purpose-built function for calculating a moving sum, or other moving formula, for a unique count. However, for many use cases, you can use a `group_aggregate` to facilitate this type of count. Adding a `group_aggregate` to your formula allows you to pass the aggregate function (in this case, `unique count`) in the windowing function (in this case, `moving_sum`).

See the following example for the general formula for a moving sum of a unique count. In this example, we calculate the moving sum of unique customer names, using `monthly date` as our aggregation bucket.

```
moving_sum (group_aggregate (unique count (Customer Name ),query_groups (),query_filters ()),2,0,start_of _month (Date ))
```

The result looks something like this:

Moving Unique Customers

moving_sum (group_aggregate (unique count (Customer...

Monthly Date	↑	moving customers	
Oct 2021		3,548	
Nov 2021		7,111	
Dec 2021		10,699	
Jan 2022		10,760	
Feb 2022		10,779	
Mar 2022		10,818	
Apr 2022		10,840	
May 2022		10,865	
Jun 2022		10,803	
Jul 2022		10,757	
Aug 2022		10,714	
Sep 2022		10,708	
Oct 2022		10,674	
Nov 2022		10,656	
Dec 2022		10,668	

You can use this syntax on any moving formula: average, maximum, minimum, and sum.

This syntax produces a unique count for each individual result, such as **Oct 2021**. However, it doesn't produce a unique count across results. If a customer name appears in both October 2021 and November 2021, for example, that customer would be counted in both those months.

Related Articles