How to Calculate an Average Ignoring Zero Values in Power BI

Authored by

mohammed looti

January 12, 2026

One of the most frequent challenges analysts face when calculating averages within **Power BI** is dealing with zero values. If a dataset contains records where a metric is recorded as zero--perhaps indicating no sales, no attendance, or no points scored--the default averaging behavior will include these zeroes, potentially skewing the overall result and leading to an artificially lower mean. To derive truly meaningful statistical insights, it is often necessary to calculate the average based only on positive, non-zero contributions.

Fortunately, **Power BI**, through its powerful calculation engine utilizing **DAX** (Data Analysis Expressions), provides several elegant solutions for implementing custom filters during aggregation. While functions like `AVERAGEX` can be used for row-by-row iteration, the most robust and versatile method involves combining the `CALCULATE` and `FILTER` functions. This approach allows users to precisely define the context under which the average is computed, ensuring that zero values are completely excluded from the denominator and the summation.

This tutorial will guide you through the process of creating a custom **measure** using **DAX** to accurately compute an average that ignores all zero entries, thus yielding a more representative figure for performance analysis and reporting.

Understanding the Impact of Zero Values on Averages

When you calculate the standard average of a column in **Power BI** using the basic `AVERAGE()` function, the engine sums all values in the column and divides that sum by the total count of rows containing numerical data. This inclusion of zero values can severely distort the interpretation of the results, particularly in scenarios where a zero represents an absence of activity rather than a valid, minimal contribution.

Consider a retail environment where 10 stores are tracked, and two stores had zero sales (due to being closed for renovation). If we average the sales of all 10 stores, the two zero entries drag down the average performance of the active stores. By excluding these two zero values, we obtain a much clearer picture of the average performance among the stores that were actually operational. This context setting is essential for accurate business intelligence reporting and decision-making.

To overcome this limitation, we must leverage the advanced capabilities of the **DAX** language to modify the filter context dynamically before the aggregation occurs. This requires moving beyond simple column aggregations and defining explicit calculation logic.

The Power of CALCULATE and Filter Context Modification

The **CALCULATE** function is arguably the most powerful function in **DAX** because it allows us to

change the context in which an expression is evaluated. When calculating an average while ignoring zeros, we instruct **Power BI** to first apply a specific filter (excluding zeroes) and then perform the aggregation (the average).

The `CALCULATE` function takes the standard aggregation expression (e.g., `AVERAGE('Table')`) as its first argument, followed by one or more filter arguments. It is the filter argument that we use to explicitly remove rows where the value is zero. This guarantees that both the numerator (the sum of values) and the denominator (the count of values) used by `AVERAGE` are based only on non-zero entries.

This method provides superior control compared to simply creating a calculated column, as the **measure** will dynamically respond to any slicers, filters, or dimensional contexts applied in your visual report, ensuring accuracy across all levels of granularity.

Syntax for Ignoring Zero Values in DAX

The specific pattern required in **DAX** to calculate the average value in a column while ignoring any values equal to zero uses the combination of `CALCULATE` and `FILTER`. The `FILTER` function is necessary here because we are applying a conditional criterion (value not equal to zero) to the entire table before the average calculation runs.

The general structure is as follows:

You can use the following syntax in **DAX** to calculate the average value in a column while ignoring any values equal to zero:

```
Avg Points =
CALCULATE (
AVERAGE ( 'my_data' ),
FILTER ( 'my_data', 'my_data' <> 0 )
)
```

This particular example creates a new **measure** named **Avg Points** that calculates the average value in the **Points** column of the table named **my_data** while ignoring any values equal to zero. The **`CALCULATE`** function overrides the existing filter context of the calculation, enforcing the exclusion defined by the **`FILTER`** argument. The expression `'my_data' <> 0` serves as the boolean condition, returning only rows where the value is non-zero.

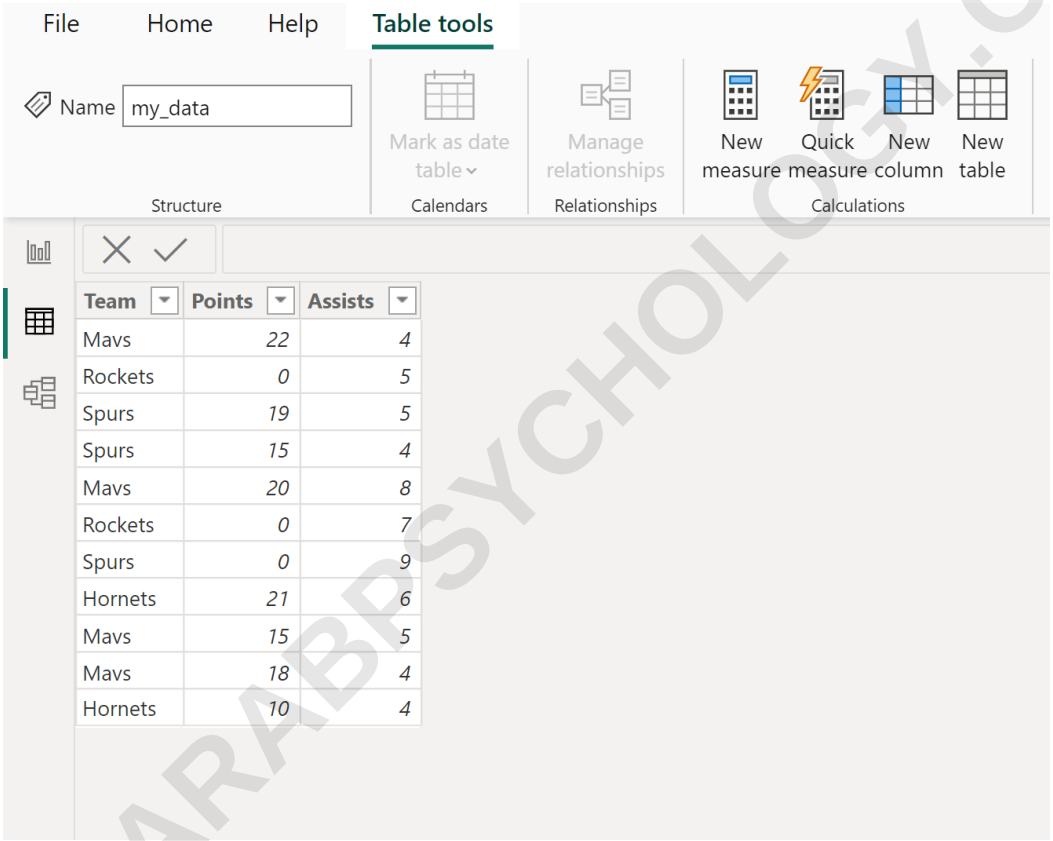
It is important to note the iterative nature implied by the `FILTER` function. It instructs **Power BI** to scan the entire `'my_data'` table, evaluate the condition row by row, and build a virtual table containing only the rows that meet the criterion. The `AVERAGE` function then operates exclusively on

this filtered virtual table.

Example: How to Calculate Average and Ignore Zeros in Power BI

To fully grasp the implementation, let us walk through a practical scenario. Suppose we are tracking the performance of basketball players across various teams. We have collected scoring data, but some players have recorded zero points in specific games, perhaps due to injury, limited playtime, or tactical reasons. We want to determine the true average scoring rate only for games where points were actually registered.

We begin with a table in **Power BI** named **my_data** that contains the player scores:



Team	Points	Assists
Mavs	22	4
Rockets	0	5
Spurs	19	5
Spurs	15	4
Mavs	20	8
Rockets	0	7
Spurs	0	9
Hornets	21	6
Mavs	15	5
Mavs	18	4
Hornets	10	4

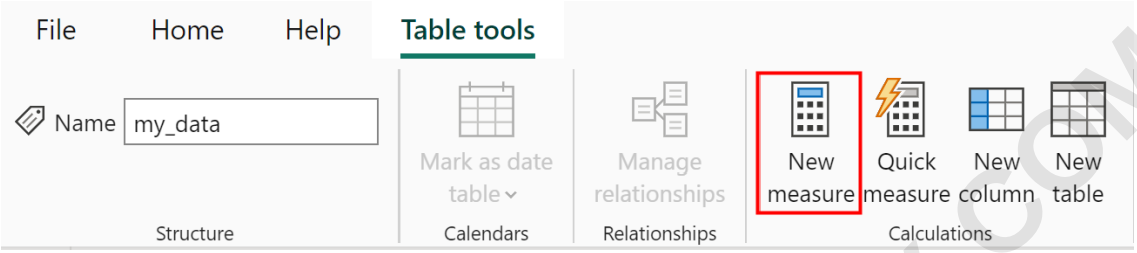
As you can clearly observe, there are several values equal to zero in the **Points** column. If we used a simple average, the zeroes would pull the overall average downwards, potentially misrepresenting the actual scoring efficiency when a player is active.

Creating the Custom DAX Measure

Our goal is to create a new, reusable calculation--a **measure**--that encapsulates the logic for ignoring these zero values. Measures are critical in **Power BI** as they compute results on the fly

based on the current context of the report, ensuring that the calculation is always accurate regardless of which filters or visualizations are active.

To define this new calculation, we must access the data modeling tools. First, navigate to the **Table tools** tab located along the top ribbon in **Power BI** Desktop. Once there, click the **New measure** icon:



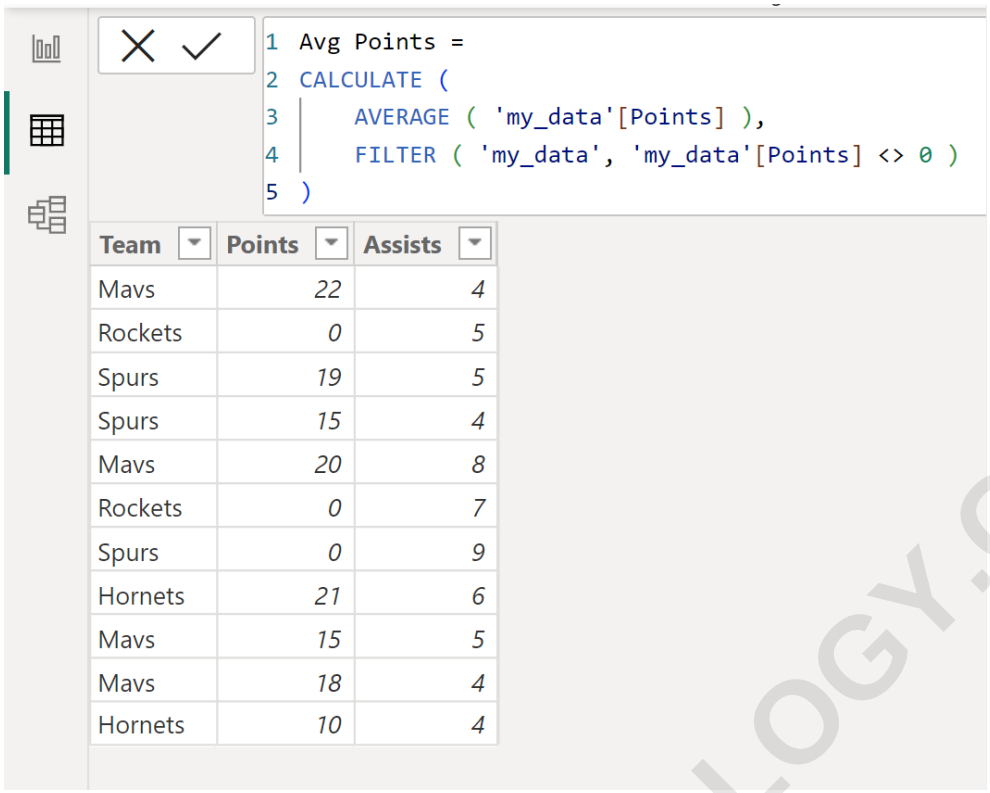
Clicking this icon opens the formula bar, allowing us to input our custom **DAX** expression. Type in the following formula precisely as shown, ensuring that the table and column names match your dataset:

```
Avg Points =
CALCULATE (
AVERAGE ( 'my_data' ),
FILTER ( 'my_data', 'my_data' <> 0 )
)
```

This powerful formula establishes a new measure named **Avg Points**. The logic within the formula instructs the **CALCULATE** function to first filter the table `'my_data'` using the **`FILTER`** function, retaining only rows where the column is not equal to zero. It then calculates the `AVERAGE` of the column within the context of that newly filtered virtual table.

Verification and Rationale of the Result

Once the measure is defined and committed, **Power BI** adds it to your dataset. This measure now contains the aggregated average value based on our specific non-zero requirement. Observe the result of the newly created measure:



```
Avg Points =
CALCULATE (
    AVERAGE ( 'my_data'[Points] ),
    FILTER ( 'my_data', 'my_data'[Points] <> 0 )
)
```

Team	Points	Assists
Mavs	22	4
Rockets	0	5
Spurs	19	5
Spurs	15	4
Mavs	20	8
Rockets	0	7
Spurs	0	9
Hornets	21	6
Mavs	15	5
Mavs	18	4
Hornets	10	4

This new **measure**, **Avg Points**, now holds the average of values in the **Points** column while successfully ignoring all zero entries. The value obtained is **17.5**. This calculated result can be confirmed by manually reviewing the data.

If we look at the original data table, the non-zero point values are: 22, 19, 15, 20, 21, 15, 18, and 10. There are 8 such non-zero entries. The sum of these values is 140. Thus, the calculation performed by our **CALCULATE** measure is:

Average of Points while Ignoring Zeros: (22 + 19 + 15 + 20 + 21 + 15 + 18 + 10) / 8 = 140 / 8 = **17.5**

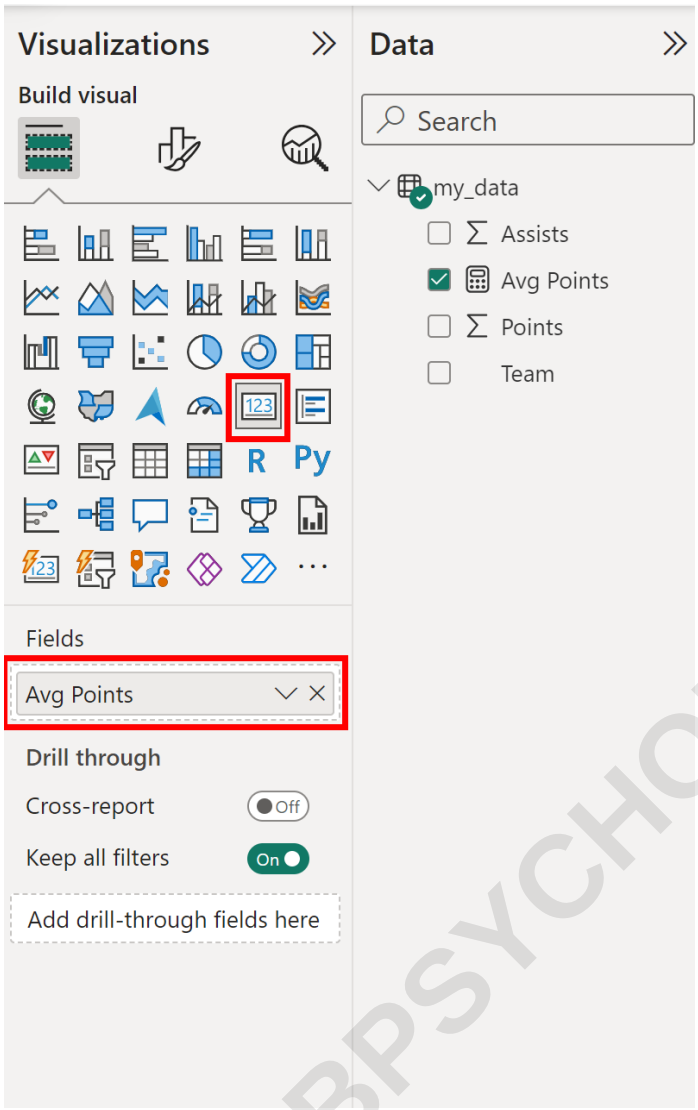
This manual verification confirms that the **DAX** formula accurately performed the required conditional averaging, providing a reliable and defensible metric for performance evaluation.

Displaying the Calculated Average in a Visualization

After defining the measure, the next step is to integrate this crucial metric into your visual report. Measures are typically displayed using Card visualizations for single, aggregated values.

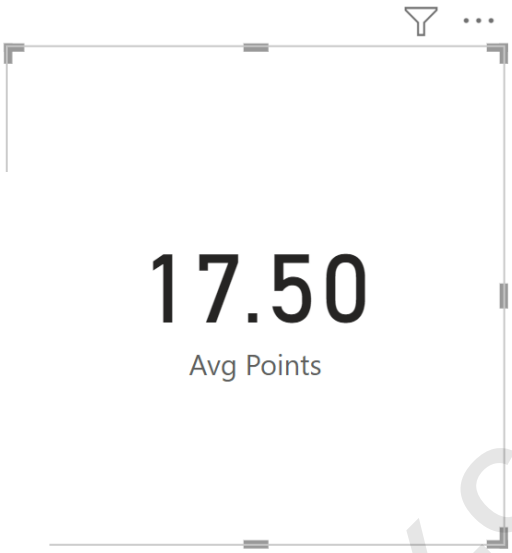
To display this value, switch to the **Report View** in **Power BI** Desktop. Under the **Visualizations** tab, select the **Card** icon. This creates a new visualization container on your report canvas. Next, locate the **Avg Points** measure (it will appear under the table it was created in) and drag it into the

Fields label of the Card visual:



The Card visual will immediately display the calculated average based on the criteria established in the **CALCULATE** function, showcasing the average of **17.5**:

Team	Sum of Points	Sum of Assists
Hornets	31	10
Mavs	75	21
Rockets	0	12
Spurs	34	18
Total	**140**	**61**



This visualization confirms that the average value in the **Points** column, after successfully ignoring all zero entries using the custom filter context, is indeed **17.5**. This figure provides a far more accurate representation of average scoring performance compared to an unfiltered average.

Alternative DAX Approaches for Excluding Values

While the `CALCULATE(AVERAGE(...), FILTER(...))` structure is the most versatile and highly recommended method for excluding values, it is worth noting that **DAX** offers other functions that can achieve similar results, particularly `AVERAGEX`.

`AVERAGEX` is an iterative function designed to calculate the average of an expression evaluated row by row over a table. We could use it to achieve the same result by filtering the table first:

Using AVERAGEX with FILTER: You define the table context first and then the expression. The syntax would be: `Avg Points X = AVERAGEX ( FILTER ( 'my_data', 'my_data' <> 0 ), 'my_data' )`. This achieves the identical outcome by instructing `AVERAGEX` to only iterate over the rows where Points are non-zero.

Using AVERAGEX with Boolean Logic: If performance is paramount and the underlying table is massive, a slightly more concise method might involve multiplying the column value by a Boolean expression, although this is generally less readable. For instance: `Avg Points Y = SUMX('my_data', 'my_data' * ('my_data' <> 0)) / CALCULATE(COUNTROWS('my_data'), FILTER('my_data', 'my_data' <> 0))`. However, this is unnecessarily complex when the `CALCULATE(AVERAGE, FILTER)` method is so clean and effective.

For standard filtering requirements in **Power BI**, adhering to the `CALCULATE(AVERAGE(Column), FILTER(Table, Condition))` pattern provides the best balance of performance, readability, and maintenance ease, making it the industry best practice for conditional averaging.