Learning Power BI: Using the ALL Function to Ignore Filters in DAX Measures

Authored by

Mohammed looti

November 12, 2025

The ability to manipulate the Filter Context is fundamental to building advanced analytical models in Power BI. Often, developers need a measure that calculates a value based on the entire dataset or a specific subset of data, completely ignoring any slicers or filters applied by the user in the report view. This requirement is typically met using the powerful combination of the **CALCULATE** function and the ALL function within DAX (Data Analysis Expressions).

For instance, if your goal is to calculate a Key Performance Indicator (KPI) or a percentage of the grand total, you must first establish that grand total regardless of the current visual filtering. The primary mechanism for achieving this filter removal is the **ALL** function, which temporarily overrides the existing filter context applied by the report user.

We will explore how to construct a measure, such as **Max Points**, that determines the maximum possible value across a column called **Points** in the table **my_data**, while specifically ensuring that any filters applied to the **Team** column are disregarded. This technique ensures that the denominator or benchmark remains constant, providing stable calculations across varying report views.

Max Points = CALCULATE(MAX('my_data'),ALL('my_data'))

Understanding Context Transition and Filter Propagation in Power BI

Before diving into the practical application of filter manipulation, it is essential to grasp the core concepts of context within DAX, specifically the difference between **Row Context** and Filter Context. The Filter Context defines the set of values allowed in each column of the data model based on visual elements like slicers, filters, rows, or columns in a visual. When a standard measure is evaluated, it respects this context, meaning if you filter the report for 'Team A', the measure calculates only using data relevant to 'Team A'.

The challenge arises when a calculation requires a constant value--a denominator or a maximum reference point that is independent of the current visual selection. If we simply calculate the maximum points, and then filter the report view by a specific team, the maximum returned will only be the maximum for that single team. This is usually undesirable when comparing a team's performance against the overall best performance across the league, necessitating a method to bypass the automatic filtering applied by the report canvas.

To overcome this inherent limitation, we must proactively modify the Filter Context during the measure calculation. This is precisely the role of the CALCULATE function, which is often cited as the most powerful and complex function in DAX. **CALCULATE** allows us to change the context in which its internal expression is evaluated, enabling us to introduce or, critically, remove filters using functions like **ALL**.

The Role of the ALL Function in Data Analysis Expressions (DAX)

The ALL function is the primary tool used in DAX for clearing filters. When used as a filter argument within **CALCULATE**, **ALL** instructs the formula engine to ignore existing filters that might be applied by the report structure. It effectively returns all rows in a table, or all values in a column, ignoring any filter context that might otherwise be applied by slicers or visual interactions.

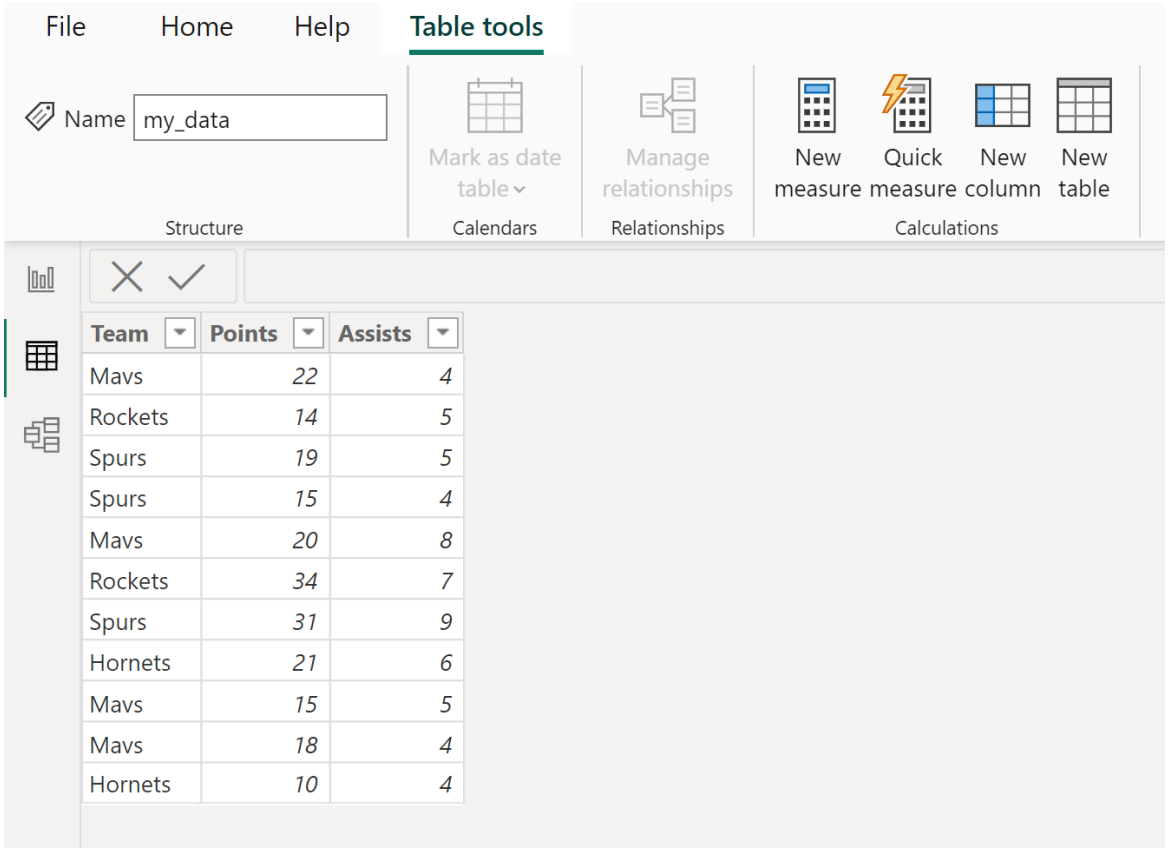
It is important to understand the different forms of **ALL**. If you use `ALL(Table)`, you remove all filters from all columns in that specific table. If you use `ALL(Table)`, as we demonstrate in this example, you only remove filters from the specified column, leaving any filters on other columns in that table or related tables intact. Choosing the correct scope for **ALL** is crucial for ensuring calculation accuracy and optimized performance in complex data models.

In the context of retrieving a grand total or a maximum value across the entire dataset, specifying the column or table to be ignored ensures that the filter applied by the report user (e.g., selecting a team name) is temporarily neutralized just for that specific measure evaluation. This mechanism guarantees that the baseline value remains constant, regardless of the interactive selections made by the consumer of the Power BI report.

Scenario Setup: Calculating Maximum Values Without Filter Modification

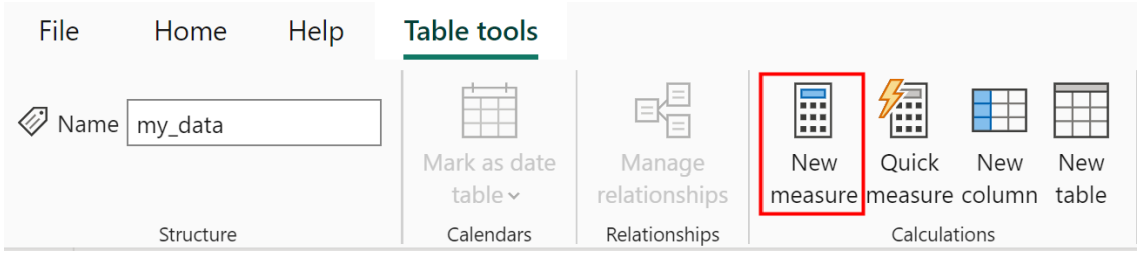
Let us consider a practical scenario using a dataset named **my_data**, which tracks performance metrics for basketball players. This table includes columns for **Team**, **Player Name**, and **Points** scored. Our initial goal is simply to find the maximum points scored globally, and then demonstrate why a simple measure fails when filters are introduced.

The dataset provided below serves as our base data model in Power BI, containing various scores associated with different teams. We can observe that the highest score across all teams is 34.



Team	Points	Assists
Mavs	22	4
Rockets	14	5
Spurs	19	5
Spurs	15	4
Mavs	20	8
Rockets	34	7
Spurs	31	9
Hornets	21	6
Mavs	15	5
Mavs	18	4
Hornets	10	4

To begin the process of calculation, we navigate to the **Table tools** tab within the Power BI desktop ribbon and select the **New measure** icon. This action initiates the creation of a new measure that will be stored within the data model.

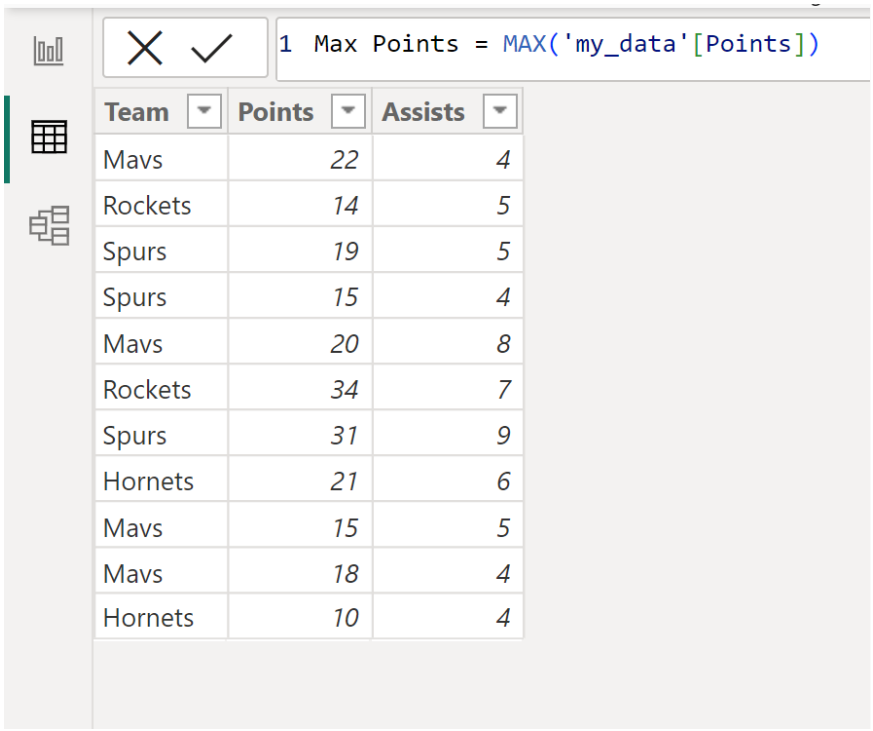


We first define a baseline measure, which we call **Max Points (Filtered)**, using the standard **MAX** function. This measure calculates the maximum value found in the **Points** column within the current filter context.

Max Points = MAX('my_data')

Upon creation, this measure correctly identifies the global maximum value from the entire dataset, which is 34, when no filters are applied to the report canvas. This initial result validates the basic

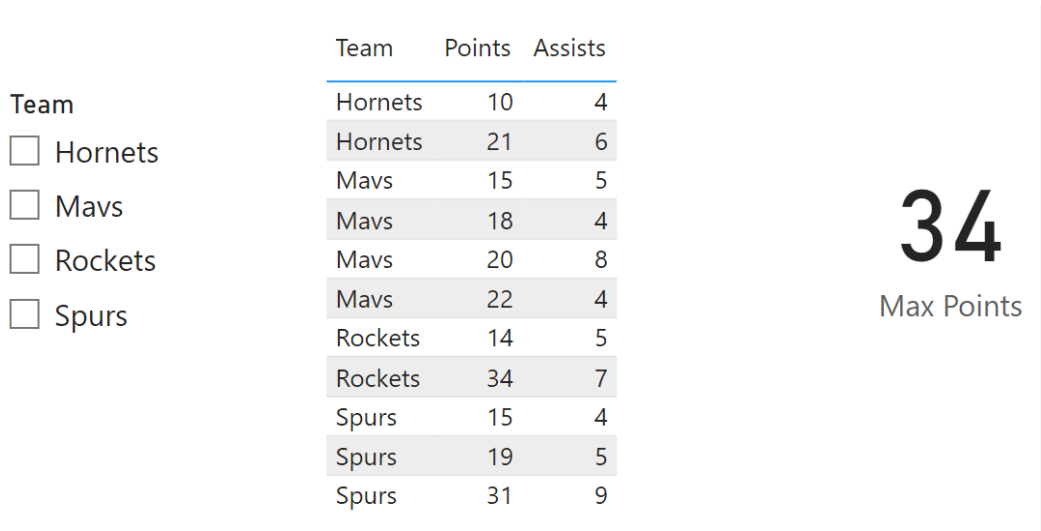
calculation logic before we introduce any context manipulation.



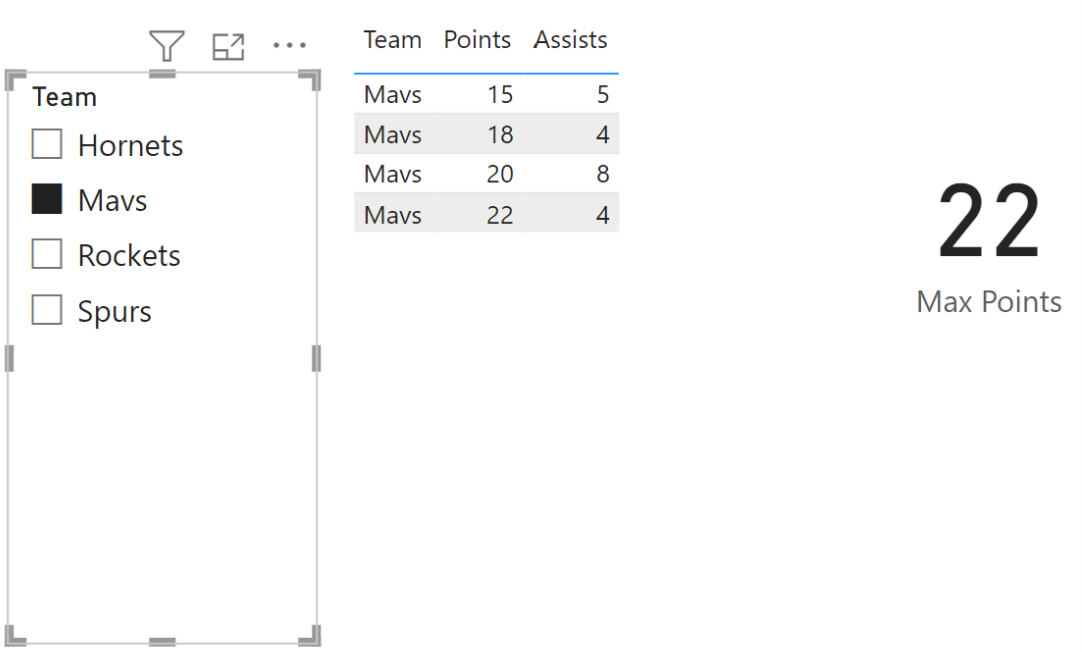
Team	Points	Assists
Mavs	22	4
Rockets	14	5
Spurs	19	5
Spurs	15	4
Mavs	20	8
Rockets	34	7
Spurs	31	9
Hornets	21	6
Mavs	15	5
Mavs	18	4
Hornets	10	4

Demonstrating Filter Impact: Why Standard Measures Fail Under Context

With our standard measure established, we move to the report view to visualize the data. We utilize a card visualization to display the value of the **Max Points (Filtered)** measure. Since the report is currently unfiltered, the card correctly shows the maximum score across the entire dataset, which is **34**.



However, the necessity of overriding filters becomes immediately clear when we introduce a slicer to limit the data. Suppose a user filters the report to view only data pertaining to the **Mavs** team. The existing measure, **Max Points (Filtered)**, respects this new Filter Context and recalculates, providing the maximum score achieved only by the Mavs team. If the Mavs' highest score is 28, the card visualization will instantly update to show 28, thereby losing the original grand maximum of 34, which is the baseline we need for comparison.



This default behavior of DAX measures, while mathematically accurate within the given context, is problematic if the requirement is to show "Max Points Scored in the League" (the global maximum) alongside "Max Points Scored by the Selected Team." To ensure the former value (34) persists, regardless of the team selected in the slicer, we must deploy the filter modification technique using the **CALCULATE** function and the ALL function.

Implementing the Solution: Overriding Filters using CALCULATE and ALL

To create a robust measure that ignores the dynamic filtering on the **Team** column, we must wrap our **MAX** aggregation within the CALCULATE function. The **CALCULATE** function takes an expression (in our case, `MAX('my_data')`) and one or more filter arguments that define the new calculation context.

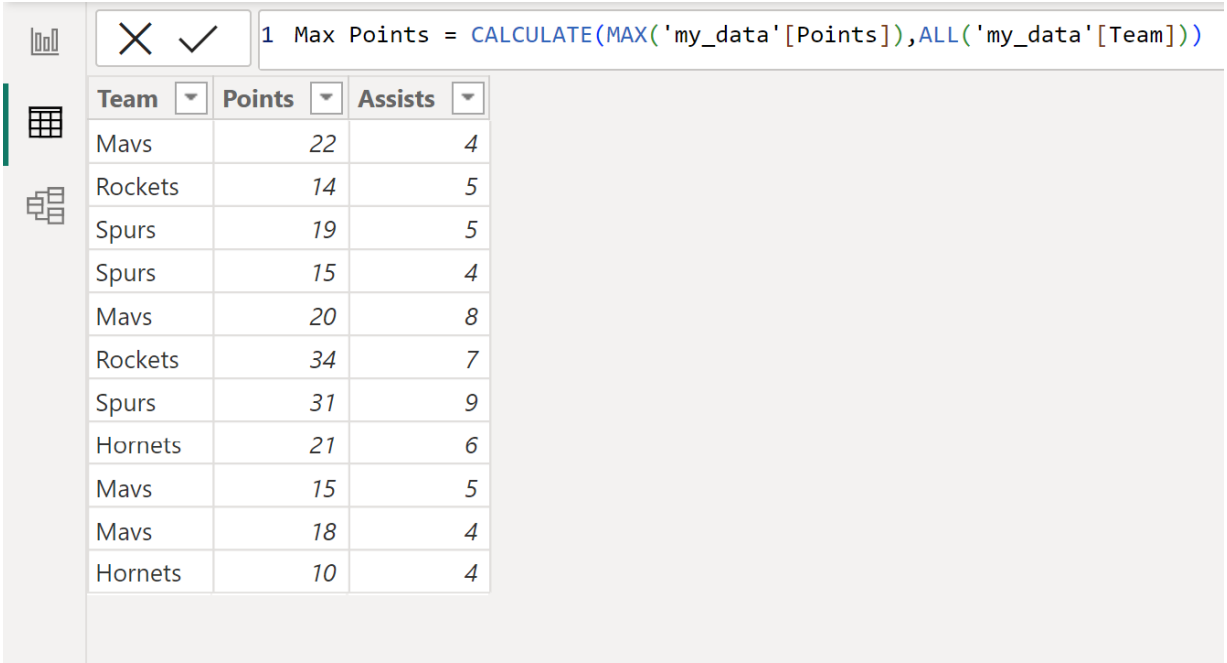
The key modification is the inclusion of the `ALL('my_data')` argument. When **CALCULATE** encounters this, it temporarily removes any external filters that are currently applied to the **Team** column, allowing the inner **MAX** expression to look across all teams present in the data model,

irrespective of the current report selection applied via slicers or visual interactions.

We define our new, corrected measure, **Max Points (Unfiltered)**, using the following definitive DAX syntax:

Max Points = CALCULATE(MAX('my_data'),ALL('my_data'))

This refined measure successfully achieves the objective. It creates a new measure that calculates the overall maximum value in the **Points** column by temporarily lifting the filter context specifically associated with the **Team** column.

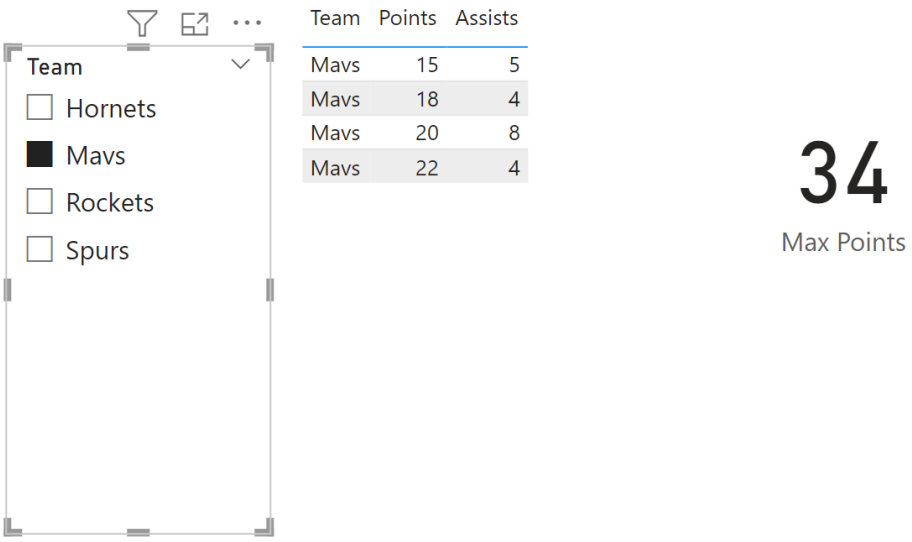


Team	Points	Assists
Mavs	22	4
Rockets	14	5
Spurs	19	5
Spurs	15	4
Mavs	20	8
Rockets	34	7
Spurs	31	9
Hornets	21	6
Mavs	15	5
Mavs	18	4
Hornets	10	4

Verifying the Results and Persistent Grand Totals

The final step involves testing the resilience of our new measure against user interactions. We return to the report view and apply the same filter as before, limiting the data displayed to the **Mavs** team. While the visual elements of the report (such as tables showing individual player scores) will correctly filter down to the Mavs data, the card visualization displaying **Max Points (Unfiltered)** should remain constant.

As demonstrated in the visualization, even when the **Team** slicer is set to filter for the Mavs, the card visualization continues to display **34**. This confirms that the `ALL('my_data')` argument successfully removed the filter context imposed by the slicer, forcing the CALCULATE function to look at all rows in the **Points** column when determining the maximum value.



This technique is vital for calculating comparison metrics, such as percentage of total, where the numerator changes based on selection but the denominator (the grand total or overall maximum) must remain static. The **ALL** function ensures that the foundational reference point is maintained across all filter scenarios, providing stability to comparative analyses.

Additional Resources for Advanced DAX Filter Manipulation

Mastering DAX involves deep familiarity with context modification functions. For those seeking to further refine their skills in calculating dynamic metrics and managing filter context, understanding the full documentation of related functions is highly recommended. The **CALCULATE** function in particular offers immense flexibility for complex business logic, allowing developers to precisely control the evaluation environment.

The primary source for detailed usage, syntax, and advanced examples of the **CALCULATE** function, including its interaction with filter modifiers like **ALL**, **ALLEXCEPT**, and **ALLSELECTED**, can be found in the official Microsoft documentation.

The following resources offer supplementary tutorials and documentation that explain how to perform other common tasks and context manipulation techniques in Power BI:

Tutorial on using the **ALLEXCEPT** function for partial filter removal based on selected columns.

Guide to implementing dynamic row-level security using DAX filters.

Understanding the difference between calculated columns and measures in Power BI and when to use each.