

How to Rank Values Within Groups in Power BI: A Step-by-Step Guide

Authored by
stats writer

January 27, 2026

RECOMMENDED CITATION

stats writer (2026). *How to Rank Values Within Groups in Power BI: A Step-by-Step Guide*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=128015>

Analyzing and understanding data often requires more than simple aggregation; it demands context. The process of ranking values by group within Power BI is a foundational technique that enables advanced analytical insights. Instead of ranking all items across an entire dataset, this method organizes and sorts data within specific subgroups or categories, allowing analysts to gauge the relative performance or importance of an item within its immediate peer group.

By effectively utilizing the specialized ranking capabilities inherent in Power BI, users gain the ability to arrange data in either a descending or ascending order based on a selected measure, such as total sales, revenue figures, or points scored. This granular approach is essential for quickly identifying top performers, analyzing competitive positioning, or spotting critical outliers within defined segments. Ultimately, implementing grouped ranking in Power BI transforms raw data into a highly organized and digestible view, directly supporting more precise and informed business intelligence decisions.

Power BI: Ranking Values by Group Contextually

The Core Principle of Grouped Ranking

To achieve this contextual ranking within Power BI, we rely heavily on Data Analysis Expressions (DAX). DAX provides the flexibility needed to perform complex calculations that respect specific row and filter contexts. The standard approach involves creating a new calculated column utilizing the powerful RANKX function, combined with context modification functions like FILTER. The following syntax is used to define a new column that effectively displays the rank of values in one column, specifically grouped by the values in another column:

```
Points Rank =  
VARcurrent_team = 'my_data'  
RETURNRANKX(  
  FILTER(  
    'my_data',  
    'my_data' = current_team  
  ),  
  'my_data',  
  ,  
  ,  
  SKIP  
)
```

Deconstructing the DAX Formula for Grouped Ranking

Understanding how this formula works is crucial for customization and advanced use. This specific expression creates a new calculated column named **Points Rank**. Its primary job is to assign a unique ranking to each value found in the **Points** column, but this ranking is strictly performed within the context defined by the **Team** column. This ensures that a player's rank is determined solely by how they compare to other players on their own team, not the entire league.

The formula utilizes several key DAX components. First, we use the VAR function to define a variable, `current_team`, which captures the team value for the row being evaluated. This variable is essential for creating the grouping mechanism. The calculation then returns the result of the RANKX function, which is responsible for the actual ranking process.

Within the RANKX function, the first argument uses the FILTER function. This is the heart of the grouping logic. The FILTER expression limits the context of the ranking calculation to only those rows in the 'my_data' table where the value matches the `current_team` variable. This ensures that RANKX only compares the current row's points against other points within the same team category. The final argument, SKIP, specifies how to handle ties in the ranking, ensuring that tied ranks skip the next available rank number.

Practical Example: Ranking Player Points by Team

To illustrate this technique, let us consider a dataset loaded into Power BI named **my_data**. This table contains crucial performance information for basketball players, specifically tracking the points scored and the team they belong to:

Team	Position	Points
A	Guard	22
A	Guard	14
A	Forward	18
A	Forward	39
A	Center	30
B	Guard	25
B	Forward	18
B	Forward	12
B	Center	17
B	Center	20
C	Guard	22
C	Guard	23
C	Forward	40
C	Center	23
C	Center	28

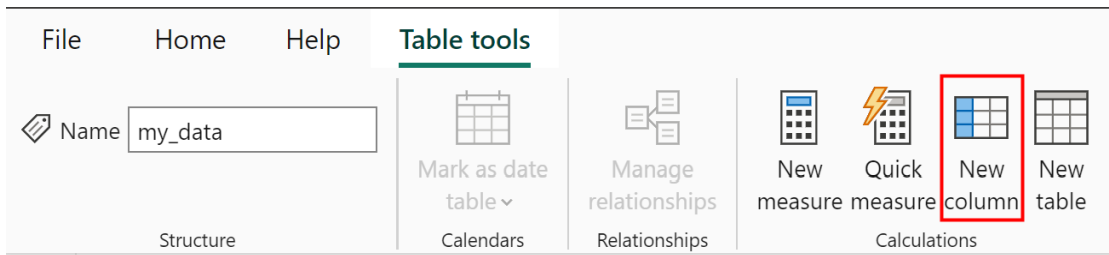
Our objective is clear: we need to augment this table by adding a new, calculated column. This column must display the rank of the values in the **Points** column, but critically, it must be partitioned and grouped entirely by the corresponding values in the **Team** column. This allows us to see who the top scorer is on Team A, Team B, and Team C, individually.

Step-by-Step Implementation in Power BI

Implementing this solution in the Power BI desktop interface is straightforward. Follow these steps to apply the grouped ranking calculation to your dataset:

Navigate to the **Table tools** tab, which is located along the top ribbon interface of Power BI Desktop.

Within the tools ribbon, locate and click on the **New column** icon. This action opens the DAX formula bar, allowing you to define the calculation for the new column.



Once the formula bar is active, input the complete DAX formula provided below. Ensure the table and column names ('my_data', ,) correctly match your specific data model schema:

Points Rank =

VARcurrent_team = 'my_data'

RETURNRANKX(
FILTER(

'my_data',

'my_data' = current_team

),

'my_data',

,

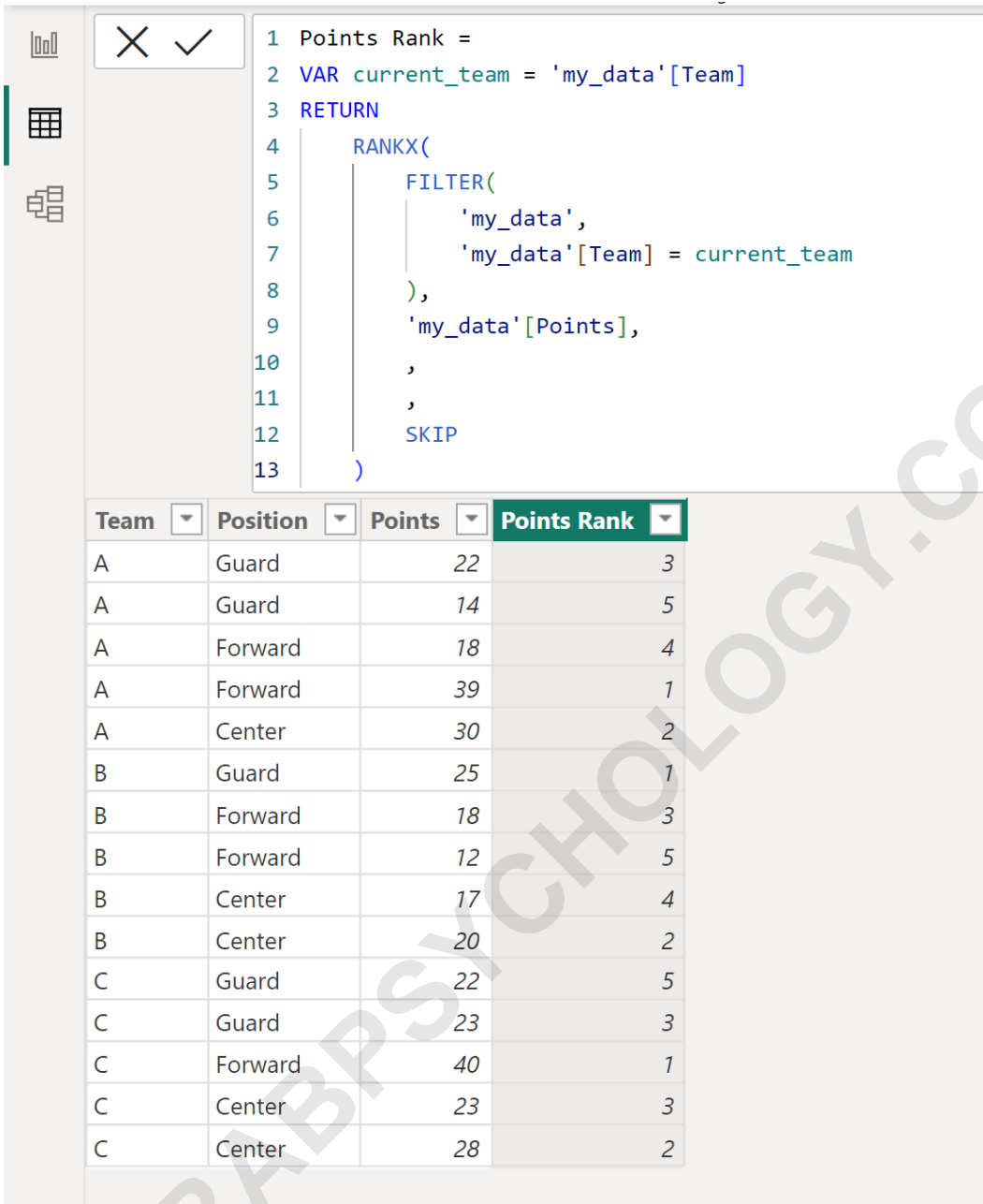
,

SKIP

)

Reviewing the Calculated Results

Executing this DAX definition will instantly generate the new column named **Points Rank**. This column successfully displays the rank of the values in the **Points** column, adhering strictly to the grouping context provided by the **Team** column. The output demonstrates how the ranking resets for each new team encountered in the dataset:



```

1 Points Rank =
2 VAR current_team = 'my_data'[Team]
3 RETURN
4     RANKX(
5         FILTER(
6             'my_data',
7             'my_data'[Team] = current_team
8         ),
9         'my_data'[Points],
10        ,
11        ,
12        SKIP
13    )

```

Team	Position	Points	Points Rank
A	Guard	22	3
A	Guard	14	5
A	Forward	18	4
A	Forward	39	1
A	Center	30	2
B	Guard	25	1
B	Forward	18	3
B	Forward	12	5
B	Center	17	4
B	Center	20	2
C	Guard	22	5
C	Guard	23	3
C	Forward	40	1
C	Center	23	3
C	Center	28	2

By examining the results, the effectiveness of the grouped ranking becomes immediately apparent. For example, considering only the data rows associated with Team A, the ranks are assigned based purely on internal team performance. The following list demonstrates the ranking logic:

The player with the most points on team A (39 points) received a rank of **1**.

The player with the second most points on team A (30 points) received a rank of **2**.

The player with the third most points on team A (22 points) received a rank of **3**.

This precise grouping logic is applied independently to Team B, Team C, and any other team present in the data.

Adjusting Rank Order: Ascending vs. Descending

The default behavior of the RANKX function, when the order parameter is left blank (as shown above), is to rank from high values to low values (descending order). However, situations may require ranking in ascending order--for instance, ranking tasks by time taken, where the lowest time is considered rank 1.

To switch the ranking direction from descending to ascending, you must explicitly specify the value **1** in the second-to-last parameter (the parameter) of the RANKX function. This single modification alters the comparison logic:

```
Points Rank =  
VARcurrent_team = 'my_data'  
RETURNRANKX(  
  FILTER(  
    'my_data',  
    'my_data' = current_team  
  ),  
  'my_data',  
  ,  
  1,  
  SKIP  
)
```

When this revised formula is used, the system will now assign a rank of **1** to the lowest value within each team group, a rank of **2** to the second lowest value in that group, and so on, completely reversing the hierarchy based on the **Points** column.

Further Resources and DAX Documentation

Mastering grouped ranking requires a solid foundation in how DAX manages calculation context. The technique hinges on using VAR to capture the current row's grouping identifier and then applying FILTER within RANKX to override the default filter context and constrain the ranking comparison to the subgroup defined by the variable. This pattern is highly reusable for many complex analytical requirements in Power BI.

For detailed technical specifications, syntax definitions, and examples related to complex ranking scenarios, the complete official documentation for the RANKX function in Power BI is the most authoritative resource.

The following tutorials explain how to perform other common tasks in Power BI:

ARABPSYCHOLOGY.COM