KEEPFILTERS and REMOVEFILTERS

The **KEEPFILTERS** and **REMOVEFILTERS** functions in DAX are powerful tools for managing filter contexts within calculations. They allow analysts to manipulate how filters behave in specific measures, either by preserving or removing certain filters. This control is especially useful when complex filter combinations are necessary, giving data analysts flexibility in achieving desired results.

KEEPFILTERS Function

Definition: The **KEEPFILTERS** function allows you to preserve filters applied in the visual or report context, even when you use functions that typically override or replace filters, such as CALCULATE. When KEEPFILTERS is applied, the specified filters are added to the existing filter context rather than replacing it.

Syntax:

```
KEEPFILTERS(<filter expression>)
```

- **<filter expression>**: This is the filter condition you want to keep in the current context.

How KEEPFILTERS Works

When you use **KEEPFILTERS** within a calculation, it ensures that any filters already applied by the report, visual, or slicer are maintained. Without KEEPFILTERS, CALCULATE might override or replace these filters. KEEPFILTERS helps by "layering" additional filter conditions without removing existing ones.

Usage of KEEPFILTERS Across Various Industries

Here are a few examples illustrating how KEEPFILTERS can be used in different industry scenarios.

1. **Retail**: Analyzing Sales with Multi-Level Category Filters

Suppose a retail analyst wants to calculate total sales for a specific product category and brand, without ignoring any filters applied in the visual context for other attributes, such as region or store.

```
CategorySales =

CALCULATE(

    SUM(Sales[Amount]),

    KEEPFILTERS(Sales[Category] = "Electronics"),

    KEEPFILTERS(Sales[Brand] = "BrandX")

)
```

In this example:

- **KEEPFILTERS** ensures that any existing filters, such as region or store, remain applied, while also adding filters for category and brand.

2. **Finance**: Maintaining Department Filters in Budget Calculations

A financial analyst might want to calculate the budget only for specific departments but retain any additional slicer filters, like year or project type.

```
DepartmentBudget =

CALCULATE(

    SUM(Budget[Amount]),

    KEEPFILTERS(Budget[Department] = "Marketing")

)
```

Here, **KEEPFILTERS** makes sure that the "Marketing" department filter is applied on top of any existing slicer filters, allowing the analyst to see department-specific data without ignoring other contextual filters.

REMOVEFILTERS Function

Definition: The **REMOVEFILTERS** function removes filters from specific columns or entire tables within a calculation, ignoring any filters applied in the report or visual context. This function is particularly helpful when you need to perform calculations without any external filters, giving you complete control over which parts of the data to include.

Syntax:

```
REMOVEFILTERS([<column or table>])
```

- **<column or table>**: This specifies which column(s) or table(s) you want to clear of any filters.

If no column or table is specified, REMOVEFILTERS removes all filters from the current filter context.

How REMOVEFILTERS Works

REMOVEFILTERS essentially "wipes out" the filters applied to specific columns or tables, allowing you to ignore slicers or visual context for certain aspects of your calculation.

Usage of REMOVEFILTERS Across Various Industries

Here are examples showing how REMOVEFILTERS is useful for industry-specific needs.

1. **Sales and Marketing**: Calculating Overall Sales Regardless of Product Line Filter

Suppose a sales analyst wants to calculate overall company sales without being affected by product line or category filters.

```
TotalCompanySales =

CALCULATE(

    SUM(Sales[Amount]),

    REMOVEFILTERS(Sales[ProductLine])

)
```

In this example:

- **REMOVEFILTERS** on Sales[ProductLine] ignores any filters set on product lines, providing an overall sales figure.

2. **Healthcare**: Analyzing Total Patient Visits While Ignoring Date Filter

In a healthcare scenario, a data analyst might need to calculate total patient visits across all dates, regardless of a date filter set in the report.

```
AllTimeVisits =

CALCULATE(

    COUNT(PatientVisits[VisitID]),

    REMOVEFILTERS(PatientVisits[VisitDate])

)
```

Here, **REMOVEFILTERS** on the VisitDate column removes any date-related filtering, ensuring the calculation counts all patient visits.

3. **Human Resources**: Determining Total Headcount without Department Filter

In HR, an analyst might need to know the company-wide headcount without regard to department-level filters.

```
TotalHeadcount =

CALCULATE(

    DISTINCTCOUNT(Employee[EmployeeID]),

    REMOVEFILTERS(Employee[Department])

)
```

By using **REMOVEFILTERS** on Employee[Department], the measure ignores department slicers, calculating the total employee count across the entire organization.

Summary of KEEPFILTERS and REMOVEFILTERS

Function	**Purpose**	**Example Use Case**
KEEPFILTERS	Adds new filters without overriding existing ones	Maintain store and region filters while adding category filters in retail
REMOVEFILTERS	Ignores specific filters from visuals or slicers	Calculate total sales regardless of product line filters

Comparison: When to Use KEEPFILTERS vs. REMOVEFILTERS

- **Use KEEPFILTERS** when you need to add additional filter conditions without disrupting filters already in place from slicers or visuals. It's suitable for when you want granular control within the existing filter context.
- **Use REMOVEFILTERS** when you need to ignore specific filters entirely, such as when calculating a metric at a global level (e.g., total sales across all product lines) regardless of any local filters set in the report context.

Both functions offer significant flexibility, enabling analysts to refine their data context precisely. This control is invaluable in reporting, allowing professionals to tailor their insights to match business needs more effectively.