**Additional time intelligence functions**

**Introduction**

As a data analyst, you’ll often have to analyze data over time by leveraging different functions to generate insights.

This article delves into the complex time intelligence functions you can use to generate these insights. It also demonstrates their utility using examples from Adventure Works.

**A brief overview of time intelligence functions**

Time intelligence functions are a crucial element of data analysis. They can be used to perform calculations over different periods. They are particularly useful for trend analysis, forecasting, and performance comparison.

Some of the most crucial functions that provide time intelligence output can be classified as follows:

1. Time comparison functions: These functions compare one date or time to another. For instance, comparing total revenue with the revenue from the last quarter.
2. Aggregate functions: Aggregations show the year-to-date, month-to-date or anything similar.
3. Information functions: These provide snapshots of information, like a month-opening or year-end balance. These functions are especially important in financial management.

Next, let’s explore some examples of these functions and discover how they can be used to generate time intelligence insights.

**PREVIOUSYEAR**

These functions return all dates from the previous year. They can be instrumental when making Y-o-Y (Year Over Year) comparisons. Other functions in this group include **PREVIOUSMONTH**, **PREVIOUSQUARTER**, and **PREVIOUSDAY**. These functions are used for the historical evaluation of data.

1

PREVIOUSYEAR(<dates>[,<year\_end\_date>])

* **<dates>** is a column containing dates.
* **<year\_end\_date>** is an optional parameter that defines the year-end date.

**Example:** Adventure Works can use the **PREVIOUSMONTH** function to compare this month's sales with those of the previous month. Such a comparison can reveal short-term trends or the immediate impact of any changes on the business strategy.

**NEXTYEAR**

The **NEXTYEAR** function is the forward-looking counterpart to **PREVIOUSYEAR**. The other functions of the group include **NEXTMONTH**, **NEXTQUARTER**, and **NEXTDAY**. These functions are used in projections and forecasts.

The function returns a table that contains a column of all dates in the next year, based on the first date in the dates column in the current context.

1

NEXTYEAR(<dates>[,<year\_end\_date>])

* **<dates>** is a column containing dates.
* **<year\_end\_date>** is an optional parameter that defines the year-end date.

**Example:** If Adventure Works has monthly sales targets for its sales team, it could use **NEXTMONTH** to project whether the team is on track to meet those goals based on the current month's data.

**TOTALYTD**

The year-to-date calculation is an aggregation of values from the beginning of the year to the specified date. **YTD (year-to-date)** can summarize all sales from January 1st of that year to the specified date.

1

TOTALYTD(<expression>, <dates>, [, <filter>][, <year\_end\_date>])

* **<expression>** returns a scalar value.
* **<dates>** is the date column. In this current lesson, you’re using Power BI's default **date** dimension.
* **<filter>** and **<year\_end\_date>** are optional parameters.

**Example:** Adventure Works wants to evaluate its real-time sales performance. To compute this measure, you can use the **TOTALYTD** function in DAX. **TOTALYTD** is a simple function to calculate year-to-date values. In this case, you can calculate the **YTDSales** from the **Total sales** column of the **Sales table**.

**DATESBETWEEN**

This function returns a table that contains all dates between a specified start date and an end date.

1

DATESBETWEEN(<dates>, <start\_date>, <end\_date>)

* **<dates>** is the column containing the dates.
* **<start\_date>** is the date expression at the start of the calculation.
* **<end\_date>** is the date expression that contains the last date for the calculation.

**Example:** Adventure Works wants to evaluate its summer sales. To achieve this, it must create a measure using the **DATESBETWEEN** function in DAX. It can enter (summer months) June 1, 2018, as the start date and August 31, 2018, as the end date for the function’s parameters.

**PARALLELPERIOD**

The **PARALLELPERIOD** function returns a set of dates separated from those in the specified column by specific intervals (such as days, months, quarters, and years). In a business context, it is often used to compare periods from previous years.

1

PARALLELPERIOD(<dates>,<number\_of\_intervals>,<interval>)

* **<dates>** is the column containing the dates.
* The **<number\_of\_intervals>** is the integer value that defines the number of intervals to add or subtract from the date.
* **<interval>** is the unit of time to shift the date. It can be year, quarter, or month.

**Example:** Adventure Works can use **PARALLELPERIOD** to compare this year's sales with those from two years ago. If sales have risen significantly since then, that may indicate successful strategies or growth in the customer base. Conversely, if sales have stagnated or declined, it might signal a need for changes.

**SAMEPERIODLASTYEAR**

**SAMEPERIODLASTYEAR** is a time intelligence function that compares the value for the same period in the previous year. This function is used frequently in retail and e-commerce for year-over-year (YoY) comparisons.

1

SAMEPERIODLASTYEAR(<dates>)

**Example:** Suppose Adventure Works wants to compare Q2 sales of this year with the same period last year. The **SAMEPERIODLASTYEAR** function executes this calculation swiftly. This comparison can highlight seasonal trends, marketing campaign effectiveness, and overall business health.

**CLOSINGBALANCEYEAR**

This function evaluates the **expression** at the last date of the year in the current context.

1

CLOSINGBALANCEYEAR(<expression>,<dates>[,<filter>][,<year\_end\_date>])

* **<expression>** is an expression thatreturns a scalar value.
* **<dates>** is a column containing dates.
* **<filter>** is an expression that specifies a filter to apply to the current context (this is an optional parameter).
* **<year\_end\_date>** is an optional parameter that defines the year-end date.

**Example:** Adventure Works can utilize this function to compute the year-end inventory for each product in categories and subcategories. This helps the company to plan its supply chain.

**Conclusion**

Mastering complex time intelligence functions is an important part of your journey as a data analyst. These functions enable you to deliver powerful insights, helping the business make informed decisions and plan strategically.

Remember, the key is understanding these functions and knowing when and where to use them. The above examples should guide you in choosing the right function for each scenario.