Guia de Estudo Completo: As Funções DAX Mais Importantes para Power BI

Criado pelo Gemini

Setembro de 2025

Sumário

1 Funções de Agregação e Iteradoras (X) 3
- 1.1 SUM e SUMX . . . 3
- 1.2 AVERAGE e AVERAGEX . . . 4
- 1.3 COUNT, COUNTA, COUNTROWS e DISTINCTCOUNT . . . 4

2 A Função Mais Importante: CALCULATE 4
- 2.1 Modificando o Contexto de Filtro . . . 5
- 2.2 Removendo Filtros com ALL . . . 5

3 Funções de Inteligência de Tempo 5
- 3.1 TOTALYTD, TOTALQTD, TOTALMTD . . . 5
- 3.2 SAMEPERIODLASTYEAR . . . 6
- 3.3 DATEADD . . . 6

4 Funções Lógicas 6
- 4.1 IF . . . 6
- 4.2 SWITCH . . . 7
- 4.3 DIVIDE . . . 7

5 Funções de Tabela e Relacionamento 8
- 5.1 FILTER . . . 8
- 5.2 RELATED . . . 8
- 5.3 VALUES e DISTINCT . . . 9

6 Conclusão 9

Introdução ao DAX

DAX (Data Analysis Expressions) é uma linguagem de fórmulas usada no Power BI, Analysis Services e Power Pivot no Excel. Com o DAX, você pode criar novas informações a partir de dados já existentes em seu modelo. Ele permite a criação de colunas calculadas, medidas e tabelas calculadas, que são a base para análises de dados robustas e dinâmicas.

Este guia aborda as funções DAX mais cruciais, divididas por categorias, com explicações claras e exemplos práticos para acelerar seu aprendizado.

Contexto dos Exemplos: Para os exemplos, considere um modelo de dados simples com as seguintes tabelas:

- `Vendas`: Contém detalhes de transações. Colunas: `[Data]`, `[IDProduto]`, `[Quantidade]`, `[PrecoUnitario]`.
- `Produtos`: Tabela de dimensão de produtos. Colunas: `[IDProduto]`, `[NomeProduto]`, `[Categoria]`, `[Cor]`.
- `dCalendario`: Tabela de dimensão de datas. Colunas: `[Data]`, `[Ano]`, `[Mes]`, `[Trimestre]`.

—

1 Funções de Agregação e Iteradoras (X)

Funções de agregação resumem os dados de uma coluna. As funções iteradoras (com sufixo 'X') percorrem uma tabela linha por linha para realizar um cálculo e, ao final, agregam o resultado.

1.1 SUM e SUMX

- **SUM(Coluna)**: Soma todos os números em uma coluna.
- **SUMX(Tabela, Expressão)**: Avalia uma expressão para cada linha de uma tabela e depois soma os resultados.

Exemplo Prático **Problema:** Calcular o faturamento total.
Análise: O faturamento é a soma da multiplicação de `[Quantidade]` por `[PrecoUnitario]` para cada venda. Uma simples `SUM` não funciona, pois precisamos de um cálculo linha a linha. A `SUMX` é a escolha ideal.

```
-- Medida para calcular o Faturamento Total
Faturamento Total =
SUMX (
    Vendas,
    Vendas[Quantidade] * Vendas[PrecoUnitario]
)
```

1.2 AVERAGE e AVERAGEX

- **AVERAGE(Coluna)**: Calcula a média aritmética dos valores em uma coluna.
- **AVERAGEX(Tabela, Expressão)**: Calcula a média de uma expressão avaliada para cada linha de uma tabela.

Exemplo Prático **Problema:** Calcular o preço médio por transação.

```
-- Medida para o Pre o M dio da Venda
Ticket Medio =
AVERAGEX (
    Vendas,
    Vendas[Quantidade] * Vendas[PrecoUnitario]
)
```

1.3 COUNT, COUNTA, COUNTROWS e DISTINCTCOUNT

- **COUNT(Coluna)**: Conta o número de células em uma coluna que contêm números.
- **COUNTA(Coluna)**: Conta o número de células não vazias em uma coluna (números, texto, etc.).
- **COUNTROWS(Tabela)**: Conta o número de linhas em uma tabela.
- **DISTINCTCOUNT(Coluna)**: Conta o número de valores distintos (únicos) em uma coluna.

Exemplo Prático **Problema:** Contar quantos produtos diferentes foram vendidos.

```
-- Medida para contar produtos distintos vendidos
Produtos Distintos Vendidos =
DISTINCTCOUNT ( Vendas[IDProduto] )
```

2 A Função Mais Importante: CALCULATE

`CALCULATE` é a função mais poderosa e flexível do DAX. Ela modifica o contexto de filtro no qual uma expressão é avaliada.

Sintaxe: `CALCULATE(expressão, filtro1, filtro2, ...)`

2.1 Modificando o Contexto de Filtro

Exemplo Prático **Problema:** Calcular o faturamento apenas para a categoria "Eletrônicos".
Análise: Usamos a medida `[Faturamento Total]` que já criamos e aplicamos um novo filtro sobre a tabela `Produtos`.

```
-- Medida de faturamento para uma categoria espec fica
Faturamento Eletronicos =
CALCULATE (
    [Faturamento Total],
    Produtos[Categoria] = "Eletr nicos"
)
```

2.2 Removendo Filtros com ALL

A função `ALL` remove os filtros de uma ou mais colunas, ou de uma tabela inteira. É frequentemente usada dentro de `CALCULATE`.

Exemplo Prático **Problema:** Calcular o percentual do faturamento de cada produto em relação ao faturamento total de todas as categorias.
Análise: Para o denominador, precisamos do faturamento total, ignorando qualquer filtro de categoria ou produto que esteja ativo no visual (gráfico, tabela, etc.).

```
-- Medida para o percentual de participa  o no total
% Faturamento vs. Total =
DIVIDE (
    [Faturamento Total],
    CALCULATE (
        [Faturamento Total],
        ALL ( Produtos ) -- Remove todos os filtros da tabela
    Produtos
    )
)
```

3 Funções de Inteligência de Tempo

Essas funções simplificam cálculos baseados em datas, como comparações anuais, mensais ou trimestrais. Elas exigem uma tabela de calendário bem estruturada, marcada como "tabela de data"no Power BI.

3.1 TOTALYTD, TOTALQTD, TOTALMTD

Calculam o acumulado de uma expressão no ano (*Year-To-Date*), trimestre (*Quarter-To-Date*) ou mês (*Month-To-Date*).

Sintaxe: `TOTALYTD(expressão , datas , [filtro])`

Exemplo Prático **Problema:** Calcular o faturamento acumulado no ano.

```
-- Medida de faturamento acumulado no ano (Year-To-Date)
Faturamento YTD =
TOTALYTD (
    [Faturamento Total],
    dCalendario[Data]
)
```

3.2 SAMEPERIODLASTYEAR

Retorna um conjunto de datas deslocadas um ano para trás no tempo, a partir das datas no contexto atual. É ideal para comparações anuais (Year-over-Year).

Sintaxe: `SAMEPERIODLASTYEAR(datas)`

Exemplo Prático **Problema:** Calcular o faturamento do mesmo período do ano anterior (SPLY).

```
-- Medida de faturamento do mesmo per odo do ano anterior
Faturamento Ano Anterior =
CALCULATE (
    [Faturamento Total],
    SAMEPERIODLASTYEAR ( dCalendario[Data] )
)
```

3.3 DATEADD

Retorna uma tabela com uma coluna de datas, deslocada para frente ou para trás no tempo pelo número de intervalos especificado.

Sintaxe: `DATEADD(datas, número_de_intervalos, intervalo)` Onde `<intervalo>` pode ser `DAY`, `MONTH`, `QUARTER` ou `YEAR`.

Exemplo Prático **Problema:** Comparar o faturamento de um mês com o mês anterior.

```
-- Medida de faturamento do m s anterior
Faturamento Mes Anterior =
CALCULATE (
    [Faturamento Total],
    DATEADD ( dCalendario[Data], -1, MONTH )
)
```

4 Funções Lógicas

4.1 IF

Testa uma condição e retorna um valor se a condição for VERDADEIRA, e outro valor se for FALSA.

Sintaxe: `IF(teste_lógico, resultado_se_verdadeiro, [resultado_se_falso])`

Exemplo Prático **Problema:** Criar uma coluna calculada na tabela `Vendas` para classificar as vendas como "Grande"(valor > R$ 1000) ou "Pequena".

```
-- Coluna Calculada na tabela Vendas
Tamanho da Venda =
IF (
    (Vendas[Quantidade] * Vendas[PrecoUnitario]) > 1000,
    "Grande",
    "Pequena"
)
```

4.2 SWITCH

Avalia uma expressão em relação a uma lista de valores e retorna uma de várias expressões de resultado possíveis. É uma alternativa mais limpa e eficiente para `IF`s aninhados.

Sintaxe: `SWITCH(expressão, valor1, resultado1, valor2, resultado2, ..., [else])`

Exemplo Prático **Problema:** Criar uma coluna calculada na tabela `Produtos` para agrupar as cores em categorias mais simples.

```
-- Coluna Calculada na tabela Produtos
Grupo de Cor =
SWITCH (
    TRUE (),
    Produtos[Cor] = "Azul" || Produtos[Cor] = "Ciano", "Frio",
    Produtos[Cor] = "Vermelho" || Produtos[Cor] = "Laranja", "
    Quente",
    Produtos[Cor] = "Preto" || Produtos[Cor] = "Branco", "Neutro",
    "Outra Cor" -- Valor Else (padr o)
)
```

4.3 DIVIDE

Executa a divisão e fornece um resultado alternativo ou BLANK() em caso de divisão por zero. É a forma mais segura de dividir em DAX.

Sintaxe: `DIVIDE(numerador, denominador, [resultado_alternativo])`

Exemplo Prático **Problema:** Calcular o crescimento do faturamento em relação ao ano anterior (YoY Growth %).

```
-- Medida de crescimento percentual ano a ano
Crescimento YoY % =
VAR FaturamentoAtual = [Faturamento Total]
VAR FaturamentoAnterior = [Faturamento Ano Anterior]
VAR Crescimento = FaturamentoAtual - FaturamentoAnterior

-- Usar DIVIDE para evitar erros de divis o por zero
RETURN
DIVIDE (
    Crescimento,
    FaturamentoAnterior,
    BLANK() -- Retorna vazio se o denominador for zero
)
```

5 Funções de Tabela e Relacionamento

5.1 FILTER

Retorna uma tabela que representa um subconjunto de outra tabela ou expressão. É muito usada dentro de `CALCULATE`.

Sintaxe: `FILTER(tabela, condição_de_filtro)`

Exemplo Prático **Problema:** Calcular o faturamento apenas de produtos caros (preço unitário $>$ R$ 500).

```
-- Medida de faturamento de produtos caros
Faturamento Produtos Caros =
CALCULATE (
    [Faturamento Total],
    FILTER (
        Produtos,
        Produtos[PrecoUnitario] > 500
    )
)
```

5.2 RELATED

Retorna um valor relacionado de outra tabela. Funciona apenas em um contexto de linha e em relacionamentos do tipo "muitos para um"($N \rightarrow 1$).

Sintaxe: `RELATED(Coluna)`

Exemplo Prático **Problema:** Criar uma coluna calculada na tabela `Vendas` que traga a categoria do produto.
Análise: Como existe um relacionamento entre `Vendas` (muitos) e `Produtos` (um), podemos "puxar"qualquer coluna da tabela `Produtos` para a `Vendas`.

```
-- Coluna Calculada na tabela Vendas
Categoria do Produto = RELATED(Produtos[Categoria])
```

5.3 VALUES e DISTINCT

Ambas retornam uma tabela de uma única coluna contendo os valores distintos da coluna especificada. A principal diferença é que `VALUES` também retorna um valor `BLANK` se houver uma violação de integridade referencial, enquanto `DISTINCT` não.

Sintaxe: `VALUES(ColunaOuTabela)`

Exemplo Prático **Problema:** Verificar se apenas um produto foi selecionado em um filtro de relatório.

```
-- Medida para exibir o nome do produto selecionado
Produto Selecionado =
IF (
    HASONEVALUE ( Produtos[NomeProduto] ), -- Verifica se h
    apenas um valor no filtro
    VALUES ( Produtos[NomeProduto] ), -- Se sim, retorna esse valor
    "V rios Produtos" -- Se n o , retorna um texto padr o
)
```

6 Conclusão

Dominar estas funções DAX fundamentais é um passo essencial para transformar seus dados brutos em insights acionáveis no Power BI. A chave para a fluência em DAX é a prática constante e a compreensão profunda dos conceitos de contexto de avaliação (contexto de linha e contexto de filtro).

Continue explorando, testando e aplicando essas fórmulas em seus próprios projetos para solidificar seu conhecimento.