

Tutorial Completo DAX

You said:



Crie um tutorial completo sobre DAX
Utilize um conjunto de Dados conhecido

Faça primeiramente um sumário sobre tudo que voce falará, eu quero ir de zero to hero em Dax

ChatGPT said:



 Memória atualizada

Sumário do Tutorial Completo de DAX: Zero to Hero

1. Introdução ao DAX

- 1.1. O que é DAX?
- 1.2. Aplicações e Usos Comuns
- 1.3. Principais Conceitos e Diferenças do Excel

2. Estrutura Básica do DAX

- 2.1. Sintaxe Básica
- 2.2. Colunas Calculadas vs. Medidas
- 2.3. Contexto de Linha e Contexto de Filtro
- 2.4. Operadores e Funções Básicas

3. Trabalhando com Funções DAX

- 3.1. Funções de Agregação
 - 3.1.1. SUM, AVERAGE, COUNT
 - 3.1.2. SUMX, AVERAGEX
- 3.2. Funções Lógicas
 - 3.2.1. IF, SWITCH
 - 3.2.2. AND, OR, NOT
- 3.3. Funções de Texto
 - 3.3.1. CONCATENATE, FORMAT
 - 3.3.2. LEFT, RIGHT, MID
- 3.4. Funções de Data e Hora
 - 3.4.1. TODAY, NOW, YEAR, MONTH, DAY
 - 3.4.2. DATEDIFF, EOMONTH
- 3.5. Funções de Filtragem
 - 3.5.1. FILTER, ALL, CALCULATE

- 3.5.2. RELATED, RELATEDTABLE

4. Contexto de Filtro e Inteligência de Tempo

- 4.1. Entendendo Contexto de Filtro
 - 4.1.1. ALL, ALLEXCEPT, REMOVEFILTERS
 - 4.1.2. CALCULATE e Contexto de Filtro
- 4.2. Funções de Inteligência de Tempo
 - 4.2.1. SAMEPERIODLASTYEAR, PARALLELPERIOD, DATEADD
 - 4.2.2. TOTALYTD, TOTALMTD, TOTALQTD
 - 4.2.3. DATESBETWEEN, DATESINPERIOD
- 4.3. Criando Tabelas de Data (Calendário)
 - 4.3.1. Função CALENDAR e CALENDARAUTO
 - 4.3.2. Relacionamentos de Data e Inteligência de Tempo

5. Funções Avançadas e Otimização

- 5.1. Funções de Iteração
 - 5.1.1. SUMX, AVERAGEX, MINX, MAXX
 - 5.1.2. RANKX, EARLIER, EARLIEST
- 5.2. Funções de Manipulação de Tabela
 - 5.2.1. ADDCOLUMNS, SUMMARIZE, SELECTCOLUMNS
 - 5.2.2. CROSSJOIN, UNION, INTERSECT, EXCEPT
- 5.3. Técnicas de Modelagem e Desempenho
 - 5.3.1. Otimização de Cálculos
 - 5.3.2. Redução de Cardinalidade
 - 5.3.3. Uso Eficiente de Relacionamentos

6. Práticas e Desafios Reais

- 6.1. Construção de KPIs com DAX
- 6.2. Criação de Medidas Complexas
- 6.3. Análises Avançadas com Inteligência de Tempo
- 6.4. Exercícios Práticos e Estudos de Caso

7. Boas Práticas e Recursos Adicionais

- 7.1. Debugging e Diagnóstico em DAX
- 7.2. Dicas de Performance e Otimização
- 7.3. Ferramentas e Comunidades para Aprofundamento

Conjunto de Dados Utilizado:

Vamos utilizar o conjunto de dados de exemplo "AdventureWorksDW", que é amplamente conhecido e utilizado em tutoriais de Power BI e DAX. Este conjunto de dados contém informações sobre uma empresa fictícia de vendas, abrangendo dados de produtos, clientes, vendas, e muito mais.

Este sumário guiará você desde os conceitos básicos até tópicos avançados, garantindo que você tenha uma compreensão completa e prática do DAX. Em cada seção, explicaremos os conceitos teóricos e aplicaremos esses conceitos em exemplos práticos usando o conjunto de dados "AdventureWorksDW".

You said:



Vamos passar 1 a 1 por cada item? Pode ser, voce so passa após eu dizer que esta tudo bem

ChatGPT said:



 Memória atualizada

Perfeito! Vamos começar com o primeiro item.

1. Introdução ao DAX

1.1. O que é DAX?

DAX, ou Data Analysis Expressions, é uma linguagem de fórmulas usada em Power BI, Excel Power Pivot e SQL Server Analysis Services (SSAS) para manipulação de dados e cálculos avançados. O DAX permite criar colunas calculadas, medidas e tabelas, e realizar cálculos complexos que não são possíveis com fórmulas tradicionais de Excel.

1.2. Aplicações e Usos Comuns

DAX é amplamente utilizado para:

- Criar medidas agregadas, como somas, médias, contagens e percentuais.
- Aplicar cálculos de inteligência de tempo, como comparações de períodos de vendas ou crescimento ano a ano.
- Filtrar e manipular dados dinamicamente para análises personalizadas.
- Realizar cálculos complexos que dependem de múltiplas tabelas e relacionamentos entre elas.

1.3. Principais Conceitos e Diferenças do Excel

Embora DAX e Excel compartilhem algumas funções semelhantes, como `SUM` e `IF`, DAX é muito mais poderoso em ambientes de BI por conta de conceitos como:

- **Contexto de Linha e Contexto de Filtro:** Estes conceitos definem como os dados são avaliados em DAX, permitindo a aplicação de cálculos dinâmicos em grandes conjuntos de dados relacionados.
- **Colunas Calculadas vs. Medidas:** As colunas calculadas são calculadas linha a linha, enquanto as medidas são recalculadas dinamicamente conforme o contexto de filtro, proporcionando maior flexibilidade e poder analítico.

Resumo: DAX é essencial para análise de dados em Power BI e permite a criação de cálculos complexos e dinâmicos, superando as limitações das fórmulas tradicionais do Excel.

Isso cobre a introdução ao DAX. Se estiver tudo bem até aqui, podemos prosseguir para o próximo item!



You said:

Ok

ChatGPT said:



Ótimo! Vamos continuar.

2. Estrutura Básica do DAX

2.1. Sintaxe Básica

A sintaxe do DAX é semelhante à do Excel, mas com algumas diferenças importantes:

- **Funções:** As funções DAX começam com o nome da função seguido por parênteses que contêm os argumentos. Exemplo: `SUM(Table[Column])`.
- **Referências de Colunas e Tabelas:** Em DAX, você se refere a uma coluna dentro de uma tabela usando a notação `Table[Column]`. Exemplo: `Sales[Revenue]`.
- **Operadores:** DAX utiliza operadores matemáticos (`+`, `-`, `*`, `/`), operadores de comparação (`=`, `>`, `<`, `>=`, `<=`, `<>`) e operadores lógicos (`AND`, `OR`, `NOT`).

2.2. Colunas Calculadas vs. Medidas

- **Colunas Calculadas:** São adicionadas diretamente à tabela de dados e calculadas linha a linha. Elas são úteis quando você precisa de um cálculo estático que não muda conforme o contexto de filtro. Exemplo: `Total Sales = Sales[Quantity] * Sales[Price]`.
- **Medidas:** São cálculos dinâmicos que se ajustam automaticamente ao contexto de filtro. As medidas são recalculadas sempre que você interage com seus visuais no Power BI. Exemplo: `Total Revenue = SUM(Sales[Revenue])`.

2.3. Contexto de Linha e Contexto de Filtro

- **Contexto de Linha:** Refere-se à linha atual em uma tabela de dados. Em colunas calculadas, DAX avalia cada linha individualmente.
- **Contexto de Filtro:** Aplica-se a cálculos agregados e medidas, onde os dados são filtrados para fornecer resultados dinâmicos com base na seleção feita nos visuais do Power BI.

2.4. Operadores e Funções Básicas

- **Operadores Matemáticos:** `+`, `-`, `*`, `/`.
- **Operadores de Comparação:** `=`, `>`, `<`, `>=`, `<=`, `<>`.
- **Operadores Lógicos:** `AND`, `OR`, `NOT`.
- **Funções Básicas:** `SUM`, `AVERAGE`, `COUNT`, `MAX`, `MIN`.

Exemplo Prático:

Vamos criar uma coluna calculada e uma medida simples usando o conjunto de dados "AdventureWorksDW".

- **Coluna Calculada:** `Total Sales = Sales[Quantity] * Sales[Unit Price]`
- **Medida:** `Total Revenue = SUM(Sales[Revenue])`

Resumo: A estrutura básica do DAX envolve a compreensão de colunas calculadas e medidas, a importância dos contextos de linha e filtro, e o uso de operadores e funções básicas para manipular dados.

Isso cobre a estrutura básica do DAX. Está pronto para seguir em frente?

You said:



Estou pronto, lembre-se de utilizar sempre o adventureworksdw

ChatGPT said:



Memória atualizada

Perfeito! Vamos continuar.

3. Trabalhando com Funções DAX

3.1. Funções de Agregação

3.1.1. SUM, AVERAGE, COUNT

Essas são funções de agregação básicas usadas para somar, calcular a média e contar valores em uma coluna específica.

- **SUM:** Soma todos os valores de uma coluna.

DAX

```
Total Sales = SUM(Sales[Total Sales Amount])
```

- **AVERAGE:** Calcula a média dos valores de uma coluna.

DAX

```
Average Sales = AVERAGE(Sales[Total Sales Amount])
```

- **COUNT:** Conta o número de valores (não nulos) em uma coluna.

DAX

```
Total Orders = COUNT(Sales[Sales Order Number])
```

3.1.2. SUMX, AVERAGEX

Essas funções iterativas calculam a soma ou a média sobre uma tabela ou expressão específica, linha por linha.

- **SUMX:** Soma os resultados de uma expressão avaliada linha por linha.

DAX

```
Total Revenue by Line = SUMX(Sales, Sales[Quantity] * Sales[Unit Price])
```

- **AVERAGEX:** Calcula a média dos resultados de uma expressão avaliada linha por linha.

DAX

```
Average Revenue by Line = AVERAGEX(Sales, Sales[Quantity] * Sales[Unit Price])
```

Exemplo Prático com AdventureWorksDW:

- **Medida:** ``Total Sales = SUM(Sales[Sales Amount])``
- **Medida Iterativa:** ``Total Profit = SUMX(Sales, Sales[Sales Amount] - Sales[Total Product Cost])``

Essas funções são essenciais para cálculos agregados e para aplicar lógicas linha a linha que somam ou calculam médias de valores derivados de múltiplas colunas.

Resumo: As funções de agregação permitem criar somas, médias e contagens simples e iterativas sobre dados. A diferença entre ``SUM`` e ``SUMX``, por exemplo, é que o ``SUMX`` permite cálculos mais complexos, avaliando cada linha individualmente antes de agregar os resultados.

Isso cobre as funções de agregação. Se estiver tudo bem, podemos passar para as funções lógicas!

You said:



Ok

ChatGPT said:



Vamos seguir em frente.

3.2. Funções Lógicas

As funções lógicas em DAX são usadas para realizar verificações condicionais e retornar resultados baseados em certas condições.

3.2.1. IF, SWITCH

- **IF:** A função `IF` retorna um valor se uma condição for verdadeira e outro valor se for falsa.

DAX

```
High Revenue = IF(Sales[Sales Amount] > 1000, "High", "Low")
```

- **SWITCH:** A função `SWITCH` avalia uma expressão contra uma lista de valores e retorna o resultado correspondente. É útil quando você tem múltiplas condições a verificar.

DAX

```
Sales Category = SWITCH(
    TRUE(),
    Sales[Sales Amount] > 5000, "Very High",
    Sales[Sales Amount] > 1000, "High",
    Sales[Sales Amount] > 500, "Medium",
    "Low"
)
```

3.2.2. AND, OR, NOT

Essas funções são usadas para combinar ou negar condições em DAX.

- **AND:** Retorna TRUE se ambas as condições forem verdadeiras.

DAX

```
Large and Profitable = IF(AND(Sales[Sales Amount] > 1000, Sales[Profit] > 100),
    "Yes", "No")
```

- **OR:** Retorna TRUE se pelo menos uma das condições for verdadeira.

DAX

```
High Revenue or High Profit = IF(OR(Sales[Sales Amount] > 1000, Sales[Profit] > 100),
    "Yes", "No")
```

- **NOT:** Inverte o resultado de uma expressão lógica.

DAX

```
Not Profitable = IF(NOT(Sales[Profit] > 0), "Loss", "Profit")
```

Exemplo Prático com AdventureWorksDW:

- **Coluna Calculada:** `Order Size = IF(Sales[Quantity] > 10, "Large", "Small")`
- **Medida:** `High Sales = IF(Sales[Sales Amount] > 1000, "Yes", "No")`

As funções lógicas são essenciais para criar condições dentro de cálculos, permitindo que você direcione seus resultados com base em regras de negócios ou condições específicas dos seus dados.

Resumo: As funções lógicas como `IF`, `SWITCH`, `AND`, `OR`, e `NOT` permitem que você implemente lógica condicional nos seus cálculos DAX, tornando suas análises mais dinâmicas e personalizadas.

Isso cobre as funções lógicas. Pronto para seguir para as funções de texto?

You said:



ok

ChatGPT said:



Vamos continuar.

3.3. Funções de Texto

As funções de texto em DAX permitem manipular e formatar dados de texto. Elas são úteis para criar novas colunas de texto, formatar strings, ou extrair partes de um texto.

3.3.1. CONCATENATE, FORMAT

- **CONCATENATE:** Junta dois valores de texto em uma única string.

DAX

```
Full Name = CONCATENATE(Employee[First Name], " " & Employee[Last Name])
```

Note que você pode usar o operador `&` para concatenar strings de forma mais compacta.

DAX

```
Full Name = Employee[First Name] & " " & Employee[Last Name]
```

- **FORMAT:** Converte um valor numérico ou de data para uma string de texto formatada.

DAX

```
Formatted Date = FORMAT(Sales[Order Date], "MMMM YYYY")
Formatted Sales = FORMAT(Sales[Sales Amount], "$#,##0.00")
```

3.3.2. LEFT, RIGHT, MID

Essas funções permitem extrair partes específicas de uma string de texto.

- **LEFT:** Retorna o número especificado de caracteres do início de uma string.

DAX

```
First Letter of First Name = LEFT(Employee[First Name], 1)
```

- **RIGHT:** Retorna o número especificado de caracteres do final de uma string.

DAX

```
Last 4 Digits of Phone Number = RIGHT(Employee[Phone Number], 4)
```

- **MID:** Retorna um número especificado de caracteres de uma string, começando em qualquer ponto.

DAX

```
Middle Name Initial = MID(Employee[Middle Name], 1, 1)
```

Exemplo Prático com AdventureWorksDW:

- **Coluna Calculada:** `Customer Full Name = Customer[First Name] & " " & Customer[Last Name]`
- **Medida:** `Formatted Sales = FORMAT(Sales[Sales Amount], "\$#,##0.00")`

As funções de texto são muito úteis para preparar e exibir dados textuais de forma personalizada, além de permitir a extração de informações específicas dentro de strings maiores.

Resumo: As funções de texto como `CONCATENATE`, `FORMAT`, `LEFT`, `RIGHT`, e `MID` são essenciais para manipulação e formatação de dados textuais em suas análises DAX, permitindo a criação de novas colunas e medidas de texto customizadas.

Isso cobre as funções de texto. Se estiver tudo bem, podemos seguir para as funções de data e hora!

You said:



ok

ChatGPT said:



Vamos prosseguir.

3.4. Funções de Data e Hora

As funções de data e hora em DAX permitem manipular, extrair, e calcular com base em valores de data e hora. Elas são fundamentais para análises temporais e para a construção de inteligência de tempo.

3.4.1. TODAY, NOW, YEAR, MONTH, DAY

- **TODAY:** Retorna a data atual.

DAX

```
Today's Date = TODAY()
```

- **NOW:** Retorna a data e a hora atuais.

DAX

```
Current Date and Time = NOW()
```

- **YEAR:** Extrai o ano de uma data.

DAX

```
Order Year = YEAR(Sales[Order Date])
```

- **MONTH:** Extrai o mês de uma data.

DAX

```
Order Month = MONTH(Sales[Order Date])
```

- **DAY:** Extrai o dia de uma data.

DAX

```
Order Day = DAY(Sales[Order Date])
```

3.4.2. DATEDIFF, EOMONTH

- **DATEDIFF:** Calcula a diferença entre duas datas em unidades específicas (dias, meses, anos).

DAX

```
Days Between Order and Ship = DATEDIFF(Sales[Order Date], Sales[Ship Date], DAY)
```

- **EOMONTH:** Retorna o último dia do mês de uma data especificada.

DAX

```
End of Month = EOMONTH(Sales[Order Date], 0)
```

Exemplo Prático com AdventureWorksDW:

- **Coluna Calculada:** `Order Year = YEAR(Sales[Order Date])`
- **Medida:** `Days to Ship = DATEDIFF(Sales[Order Date], Sales[Ship Date], DAY)`

Essas funções permitem criar análises baseadas em datas, como calcular o tempo entre eventos, extrair componentes específicos de uma data, ou determinar a data atual.

Resumo: As funções de data e hora como `TODAY`, `NOW`, `YEAR`, `MONTH`, `DAY`, `DATEDIFF`, e `EOMONTH` são essenciais para manipulação e análise de informações temporais, permitindo criar cálculos que dependem de datas específicas ou intervalos de tempo.

Isso cobre as funções de data e hora. Pronto para continuar para as funções de filtragem?

You said:



ok

ChatGPT said:



Vamos continuar.

3.5. Funções de Filtragem

As funções de filtragem em DAX são fundamentais para manipular e ajustar o contexto de filtro de cálculos, permitindo análises mais específicas e poderosas.

3.5.1. FILTER, ALL, CALCULATE

- **FILTER:** Retorna uma tabela que contém linhas que atendem a uma condição específica. É frequentemente usada dentro de outras funções para aplicar filtros dinâmicos.

DAX

```
Sales Above 1000 = FILTER(Sales, Sales[Sales Amount] > 1000)
```

- **ALL:** Remove todos os filtros de uma tabela ou coluna. É útil para calcular totais ou ignorar filtros de contexto.

DAX

```
Total Sales All Products = CALCULATE(SUM(Sales[Sales Amount]), ALL(Sales[Product ID]))
```

- **CALCULATE:** Modifica o contexto de filtro de uma expressão. É uma das funções mais poderosas e versáteis em DAX.

DAX

```
Sales of Product A = CALCULATE(SUM(Sales[Sales Amount]), Sales[Product ID] = "A")
```

3.5.2. RELATED, RELATEDTABLE

- **RELATED:** Retorna um valor de uma tabela relacionada. É usada em colunas calculadas e permite buscar valores de uma tabela relacionada.

DAX

```
Product Category = RELATED(Product[Category])
```

- **RELATEDTABLE:** Retorna uma tabela contendo todas as linhas relacionadas a uma linha atual da tabela base.

DAX

```
Related Sales = RELATEDTABLE(Sales)
```

Exemplo Prático com AdventureWorksDW:

- **Medida:** `Total Sales = CALCULATE(SUM(Sales[Sales Amount]), Sales[Product ID] = "A")`
- **Coluna Calculada:** `Product Category = RELATED(Product[Category])`

Essas funções de filtragem permitem controlar o contexto em que os cálculos são realizados, oferecendo uma maneira de fazer cálculos complexos e específicos com base em filtros dinâmicos e relacionamentos entre tabelas.

Resumo: As funções de filtragem como `FILTER`, `ALL`, `CALCULATE`, `RELATED`, e `RELATEDTABLE` são essenciais para ajustar e manipular o contexto de filtro em seus cálculos DAX, permitindo criar análises precisas e detalhadas que atendem a requisitos específicos de negócio.

Isso conclui as funções de filtragem. Se estiver tudo certo, podemos avançar para a próxima seção: Contexto de Filtro e Inteligência de Tempo!

You said:



ok

ChatGPT said:



Vamos continuar.

4. Contexto de Filtro e Inteligência de Tempo

4.1. Entendendo Contexto de Filtro

O contexto de filtro em DAX refere-se ao conjunto de valores ativos em uma tabela em um determinado momento. Ele influencia diretamente como os cálculos são realizados, permitindo que os resultados sejam dinâmicos e dependam das seleções feitas nos visuais do Power BI.

4.1.1. ALL, ALLEXCEPT, REMOVEFILTERS

- **ALL:** Remove todos os filtros aplicados em uma tabela ou coluna específica.

DAX

```
Total Sales All Products = CALCULATE(SUM(Sales[Sales Amount]), ALL(Sales[Product ID]))
```

- **ALLEXCEPT:** Remove todos os filtros de uma tabela, exceto aqueles que são aplicados a colunas específicas.

DAX

```
Sales by Category = CALCULATE(SUM(Sales[Sales Amount]), ALLEXCEPT(Sales, Sales[Category]))
```

- **REMOVEFILTERS:** Remove filtros de colunas ou tabelas específicas, semelhante ao ALL, mas pode ser mais explícito em relação ao que é filtrado.

DAX

```
Total Sales without Region Filter = CALCULATE(SUM(Sales[Sales Amount]), REMOVEFILTERS(Sales[Region]))
```

4.1.2. CALCULATE e Contexto de Filtro

- **CALCULATE:** Modifica o contexto de filtro de uma expressão, permitindo cálculos condicionais complexos. Ao usar `CALCULATE`, você pode aplicar novos filtros ou modificar os existentes.

DAX

```
Sales of Specific Product = CALCULATE(SUM(Sales[Sales Amount]), Sales[Product ID] = "A")
```

Exemplo Prático com AdventureWorksDW:

- **Medida:** `Total Sales for All Categories = CALCULATE(SUM(Sales[Sales Amount]), ALL(Sales[Category]))`
- **Medida:** `Sales for Category X = CALCULATE(SUM(Sales[Sales Amount]), Sales[Category] = "X")`

Compreender o contexto de filtro é crucial para a criação de medidas que se ajustem dinamicamente com base nas seleções de filtros, permitindo análises personalizadas e poderosas.

Resumo: O contexto de filtro em DAX é o conjunto de filtros ativos em uma tabela que influencia os resultados dos cálculos. Funções como `ALL`, `ALLEXCEPT`, `REMOVEFILTERS`, e `CALCULATE` permitem manipular esses filtros, possibilitando análises dinâmicas e detalhadas.

Isso cobre o contexto de filtro. Se estiver tudo certo, podemos avançar para as funções de inteligência de tempo!

You said:



ok, esta tudo direitinho ne?

ChatGPT said:



Sim, está tudo direitinho! Estamos seguindo passo a passo, abordando cada conceito com clareza e exemplos práticos usando o conjunto de dados "AdventureWorksDW". Se você tiver alguma dúvida ou precisar de mais detalhes em algum ponto, é só avisar.

Pronto para continuar com as funções de inteligência de tempo?

You said:



Sim

ChatGPT said:



Ótimo, vamos prosseguir.

4.2. Funções de Inteligência de Tempo

As funções de inteligência de tempo em DAX permitem criar cálculos baseados em datas, como comparações de períodos anteriores, cálculos acumulados e análises de tendências ao longo do tempo. Essas funções são fundamentais para qualquer análise que envolva períodos de tempo.

4.2.1. SAMEPERIODLASTYEAR, PARALLELPERIOD, DATEADD

- **SAMEPERIODLASTYEAR:** Retorna um conjunto de datas do mesmo período no ano anterior.

DAX

```
Sales Last Year = CALCULATE(SUM(Sales[Sales Amount]), SAMEPERIODLASTYEAR(Sales[Order Date]))
```

- **PARALLELPERIOD:** Retorna um conjunto de datas deslocadas por um determinado número de períodos, como meses, trimestres ou anos.

DAX

```
Sales 2 Quarters Ago = CALCULATE(SUM(Sales[Sales Amount]), PARALLELPERIOD(Sales[Order Date], -2, QUARTER))
```

- **DATEADD:** Retorna um conjunto de datas ajustado por um intervalo específico, como dias, meses, trimestres ou anos.

DAX

```
Sales Month over Month = CALCULATE(SUM(Sales[Sales Amount]), DATEADD(Sales[Order Date], -1, MONTH))
```

4.2.2. TOTALYTD, TOTALMTD, TOTALQTD

- **TOTALYTD:** Calcula o total acumulado no ano até a data especificada.

DAX

```
Year to Date Sales = TOTALYTD(SUM(Sales[Sales Amount]), Sales[Order Date])
```

- **TOTALMTD:** Calcula o total acumulado no mês até a data especificada.

DAX

```
Month to Date Sales = TOTALMTD(SUM(Sales[Sales Amount]), Sales[Order Date])
```

- **TOTALQTD:** Calcula o total acumulado no trimestre até a data especificada.

DAX

```
Quarter to Date Sales = TOTALQTD(SUM(Sales[Sales Amount]), Sales[Order Date])
```

4.2.3. DATESBETWEEN, DATESINPERIOD

- **DATESBETWEEN:** Retorna todas as datas entre um intervalo de datas especificado.

DAX

```
Sales Between Dates = CALCULATE(SUM(Sales[Sales Amount]), DATESBETWEEN(Sales[Order Date], DATE(2023, 1, 1), DATE(2023, 12, 31)))
```

- **DATESINPERIOD:** Retorna um conjunto de datas em um intervalo especificado antes ou depois de uma data inicial.

DAX

```
Sales Last 30 Days = CALCULATE(SUM(Sales[Sales Amount]), DATESINPERIOD(Sales[Order Date], LASTDATE(Sales[Order Date]), -30, DAY))
```

Exemplo Prático com AdventureWorksDW:

- **Medida:** ``Sales Last Year = CALCULATE(SUM(Sales[Sales Amount]), SAMEPERIODLASTYEAR(Sales[Order Date]))``
- **Medida:** ``Year to Date Sales = TOTALYTD(SUM(Sales[Sales Amount]), Sales[Order Date])``

Essas funções permitem que você analise o desempenho ao longo do tempo, compare resultados com períodos anteriores, e calcule acumulados em diferentes intervalos de tempo.

Resumo: As funções de inteligência de tempo como ``SAMEPERIODLASTYEAR``, ``PARALLELPERIOD``, ``TOTALYTD``, ``DATESBETWEEN``, e ``DATESINPERIOD`` são cruciais para realizar análises temporais, permitindo comparações de períodos e cálculos acumulados em vários intervalos de tempo.

Isso cobre as funções de inteligência de tempo. Podemos avançar para a criação de tabelas de data (calendário)?

You said:



Vamos lá

ChatGPT said:



4.3. Criando Tabelas de Data (Calendário)

Para realizar análises de inteligência de tempo de forma eficiente, é essencial ter uma tabela de datas ou calendário em seu modelo de dados. Essa tabela geralmente contém uma linha para cada dia dentro do intervalo de datas que você deseja analisar e inclui colunas adicionais para ano, mês, trimestre, dia da semana, entre outras.

4.3.1. Função CALENDAR e CALENDARAUTO

- **CALENDAR:** Cria uma tabela de datas com um intervalo de datas definido por você.

DAX

```
Date Table = CALENDAR(DATE(2020, 1, 1), DATE(2023, 12, 31))
```

- **CALENDARAUTO:** Cria uma tabela de datas que abrange o menor e o maior valor de data existente em seu modelo de dados.

DAX

```
Date Table = CALENDARAUTO()
```

Essas tabelas geradas podem ser enriquecidas com colunas adicionais que facilitam a análise temporal.

4.3.2. Relacionamentos de Data e Inteligência de Tempo

Uma vez que você criou sua tabela de datas, é importante criar relacionamentos entre essa tabela e as outras tabelas de fato (como vendas) no seu modelo de dados. Isso permite que as funções de inteligência de tempo funcionem corretamente, pois as medidas dependerão desses relacionamentos para filtrar e agrupar os dados com base nas datas.

Passo a Passo:

1. Criar a Tabela de Datas:

DAX

```
Date Table = CALENDAR(DATE(2020, 1, 1), DATE(2023, 12, 31))
```

2. Adicionar Colunas Derivadas:

DAX

```
Year = YEAR('Date Table'[Date])
Month = MONTH('Date Table'[Date])
Day = DAY('Date Table'[Date])
Quarter = QUARTER('Date Table'[Date])
```

3. Criar Relacionamentos:

- No Power BI, vá até a seção de Modelagem.
- Crie um relacionamento entre a coluna de data da tabela de vendas e a coluna de data da sua nova tabela de datas.

Exemplo Prático com AdventureWorksDW:

- **Tabela de Datas:**

DAX

```
Date Table = CALENDAR(DATE(2010, 1, 1), DATE(2020, 12, 31))
```

- **Relacionamentos:** Conecte `Date Table[Date]` com `Sales[Order Date]` para ativar a inteligência de tempo.

A tabela de datas é uma das partes mais críticas de qualquer modelo de dados em que se deseja aplicar inteligência de tempo, pois ela centraliza todas as operações de tempo, como comparações ano a ano, acumulados e muito mais.

Resumo: Criar uma tabela de datas (calendário) usando as funções `CALENDAR` ou `CALENDARAUTO` e estabelecer relacionamentos adequados é fundamental para realizar análises temporais robustas e aplicar funções de inteligência de tempo com precisão.

Isso encerra a seção sobre criação de tabelas de data. Pronto para avançar para as funções avançadas e otimização?

You said:



Ok

ChatGPT said:



Vamos prosseguir.

5. Funções Avançadas e Otimização

Agora que já cobrimos as bases e as funções essenciais, é hora de explorar funções avançadas e práticas de otimização que permitem criar análises mais sofisticadas e melhorar a performance do seu modelo.

5.1. Funções de Iteração

As funções de iteração em DAX processam uma linha de cada vez, permitindo cálculos mais complexos. Elas são particularmente úteis quando você precisa realizar cálculos agregados que dependem de expressões específicas em cada linha.

5.1.1. SUMX, AVERAGEX, MINX, MAXX

- **SUMX:** Soma os resultados de uma expressão avaliada linha por linha.

DAX

```
Total Profit = SUMX(Sales, Sales[Sales Amount] - Sales[Total Product Cost])
```

- **AVERAGEX:** Calcula a média dos resultados de uma expressão avaliada linha por linha.

DAX

```
Average Profit per Sale = AVERAGEX(Sales, Sales[Sales Amount] - Sales[Total Product Cost])
```

- **MINX:** Retorna o menor valor de uma expressão avaliada linha por linha.

DAX

```
Min Profit per Sale = MINX(Sales, Sales[Sales Amount] - Sales[Total Product Cost])
```

- **MAXX:** Retorna o maior valor de uma expressão avaliada linha por linha.

DAX

```
Max Profit per Sale = MAXX(Sales, Sales[Sales Amount] - Sales[Total Product Cost])
```

5.1.2. RANKX, EARLIER, EARLIEST

- **RANKX:** Retorna a classificação de um valor em uma tabela em relação a outros valores, ordenando os resultados.

DAX

```
Sales Rank = RANKX(ALL(Sales), Sales[Sales Amount])
```

- **EARLIER:** Permite acessar o valor de uma coluna em um contexto de linha anterior. É útil em cálculos complexos em colunas calculadas.

DAX

```
Running Total = SUMX(FILTER(Sales, Sales[Order Date] <= EARLIER(Sales[Order Date])), Sales[Sales Amount])
```

- **EARLIEST:** Similar ao EARLIER, mas retorna o valor mais antigo em um contexto.

DAX

```
Earliest Sale Date = CALCULATE(MIN(Sales[Order Date]), EARLIEST(Sales[Order Date]))
```

Exemplo Prático com AdventureWorksDW:

- **Medida:** `Total Profit = SUMX(Sales, Sales[Sales Amount] - Sales[Total Product Cost])`
- **Medida:** `Sales Rank = RANKX(ALL(Sales), Sales[Sales Amount])`

Essas funções permitem que você realize cálculos mais sofisticados que consideram expressões dinâmicas e classificações complexas.

Resumo: As funções de iteração, como `SUMX`, `AVERAGEX`, `RANKX`, `EARLIER`, e `EARLIEST`, oferecem poderosos recursos para cálculos linha por linha e classificações, permitindo que você crie medidas e colunas calculadas mais avançadas e personalizadas.

Isso cobre as funções de iteração. Se estiver tudo bem, podemos avançar para as funções de manipulação de tabelas!

You said:



Vamos lá

ChatGPT said:



5.2. Funções de Manipulação de Tabela

As funções de manipulação de tabela em DAX permitem criar e modificar tabelas dentro de suas fórmulas. Essas funções são poderosas para criar resumos, realizar junções, e manipular dados em um nível mais avançado.

5.2.1. ADDCOLUMNS, SUMMARIZE, SELECTCOLUMNS

- **ADDCOLUMNS:** Adiciona novas colunas a uma tabela existente com base em expressões calculadas.

DAX

```
Sales with Profit = ADDCOLUMNS(Sales, "Profit", Sales[Sales Amount] - Sales[Total Product Cost])
```

- **SUMMARIZE:** Cria uma nova tabela com uma linha para cada grupo definido, resumindo colunas específicas.

DAX

```
Sales by Category = SUMMARIZE(Sales, Sales[Category], "Total Sales", SUM(Sales[Sales Amount]))
```

- **SELECTCOLUMNS:** Retorna uma nova tabela contendo apenas as colunas especificadas, com a opção de renomear colunas.

DAX

```
Sales Summary = SELECTCOLUMNS(Sales, "Order ID", Sales[Order ID], "Total Sales", Sales[Sales Amount])
```

5.2.2. CROSSJOIN, UNION, INTERSECT, EXCEPT

- **CROSSJOIN:** Retorna a combinação cartesiana de todas as linhas em duas ou mais tabelas.

DAX

```
Products and Customers = CROSSJOIN(Product, Customer)
```

- **UNION:** Retorna uma tabela que contém todas as linhas de duas ou mais tabelas. As tabelas devem ter o mesmo número de colunas.

DAX

```
All Sales = UNION(Sales2022, Sales2023)
```

- **INTERSECT:** Retorna uma tabela contendo apenas as linhas que estão presentes em ambas as tabelas.

DAX

```
Common Customers = INTERSECT(CustomerOrders2022, CustomerOrders2023)
```

- **EXCEPT:** Retorna uma tabela contendo todas as linhas da primeira tabela que não estão presentes na segunda tabela.

DAX

```
Customers Only in 2022 = EXCEPT(CustomerOrders2022, CustomerOrders2023)
```

Exemplo Prático com AdventureWorksDW:

- **Tabela:** `Sales by Category = SUMMARIZE(Sales, Sales[Category], "Total Sales", SUM(Sales[Sales Amount]))`
- **Tabela:** `Products and Customers = CROSSJOIN(Product, Customer)`

Essas funções de manipulação de tabela permitem combinar, resumir, e filtrar tabelas de formas muito flexíveis, proporcionando uma capacidade avançada de modelagem e análise de dados.

Resumo: As funções de manipulação de tabela como `ADDCOLUMNS`, `SUMMARIZE`, `SELECTCOLUMNS`, `CROSSJOIN`, `UNION`, `INTERSECT`, e `EXCEPT` são ferramentas poderosas para criar e modificar tabelas dentro de suas fórmulas, permitindo realizar análises complexas e personalizadas.

Isso cobre as funções de manipulação de tabela. Se estiver tudo bem, podemos avançar para as técnicas de modelagem e desempenho!

You said:



Ok

ChatGPT said:



Vamos continuar.

5.3. Técnicas de Modelagem e Desempenho

Quando você começa a trabalhar com grandes conjuntos de dados ou cálculos complexos em DAX, a otimização e a eficiência do modelo se tornam cruciais. Aqui estão algumas técnicas e práticas recomendadas para melhorar a performance e garantir que seu modelo seja escalável.

5.3.1. Otimização de Cálculos

- **Use Medidas ao Invés de Colunas Calculadas:** Sempre que possível, prefira criar medidas em vez de colunas calculadas. As medidas são recalculadas dinamicamente com base no contexto, o que economiza espaço e melhora a performance.
- **Evite CALCULATE em Expressões Complexas:** Sempre que possível, minimize o uso de `CALCULATE` com condições complexas. Em vez disso, tente simplificar a lógica ou dividi-la em partes menores.
- **Use Variáveis:** O uso de variáveis (`VAR`) em DAX pode melhorar a performance, especialmente em cálculos iterativos, ao armazenar resultados intermediários e evitar a repetição de cálculos caros.

DAX

```
Total Profit with Variables =
VAR TotalSales = SUM(Sales[Sales Amount])
VAR TotalCost = SUM(Sales[Total Product Cost])
RETURN TotalSales - TotalCost
```

5.3.2. Redução de Cardinalidade

- **Reduza a Granularidade de Dados:** Considere agrupar ou resumir dados ao nível necessário para sua análise, em vez de trabalhar com dados transacionais de alta granularidade. Isso pode reduzir significativamente o tamanho do modelo e melhorar a performance.
- **Evite Colunas com Alto Número de Valores Únicos:** Colunas com uma grande quantidade de valores únicos (alta cardinalidade) podem consumir muita memória. Sempre que possível, prefira colunas categóricas ou crie agrupamentos.

5.3.3. Uso Eficiente de Relacionamentos

- **Utilize Relacionamentos Unidirecionais:** Prefira relacionamentos unidirecionais entre tabelas, a menos que seja absolutamente necessário usar relacionamentos bidirecionais, pois os unidirecionais são mais eficientes.
- **Ative/Desative Relacionamentos Dinamicamente:** Use funções como `USERELATIONSHIP` para ativar relacionamentos apenas quando necessário, o que pode melhorar a performance e evitar conflitos de contexto.

DAX

```
Sales with Custom Relationship =  
CALCULATE(SUM(Sales[Sales Amount]), USERELATIONSHIP(Sales[Ship Date], 'Date  
Table'[Date]))
```

Exemplo Prático com AdventureWorksDW:

- **Medida com Variáveis:**

DAX

```
Profit Margin =  
VAR TotalSales = SUM(Sales[Sales Amount])  
VAR TotalCost = SUM(Sales[Total Product Cost])  
RETURN DIVIDE(TotalSales - TotalCost, TotalSales)
```

Essas práticas ajudam a manter seu modelo de dados ágil e rápido, mesmo quando lida com grandes volumes de dados ou cálculos complexos.

Resumo: Técnicas de modelagem e desempenho, como o uso eficiente de medidas, variáveis, redução de cardinalidade, e otimização de relacionamentos, são essenciais para manter a performance e a escalabilidade do seu modelo em DAX.

Isso conclui as técnicas de modelagem e desempenho. Pronto para avançar para as práticas e desafios reais?

You said:



ok

ChatGPT said:



Vamos avançar.

6. Práticas e Desafios Reais

Agora que você tem uma compreensão sólida dos conceitos e técnicas de DAX, é hora de aplicar esse conhecimento em cenários reais. Aqui, vamos trabalhar com práticas e desafios que simulam problemas comuns de negócios e análise de dados.

6.1. Construção de KPIs com DAX

Os KPIs (Indicadores Chave de Performance) são métricas essenciais para acompanhar o desempenho de uma empresa. Em DAX, você pode criar KPIs personalizados para refletir as metas e o desempenho real.

- **Exemplo:** Criar um KPI para monitorar a margem de lucro.

DAX

```
Profit Margin KPI =  
VAR Margin = DIVIDE(SUM(Sales[Sales Amount]) - SUM(Sales[Total Product Cost]),  
SUM(Sales[Sales Amount]))  
RETURN  
    IF(Margin > 0.3, "Good",  
        IF(Margin > 0.2, "Average", "Poor"))
```

6.2. Criação de Medidas Complexas

Medidas complexas combinam várias funções DAX para criar análises avançadas.

- **Exemplo:** Calcular a receita ajustada ao longo do tempo, considerando uma taxa de crescimento anual.

DAX

```
Adjusted Revenue =  
VAR GrowthRate = 0.05  
RETURN  
    SUMX(  
        Sales,  
        Sales[Sales Amount] * (1 + GrowthRate) ^ (YEAR(TODAY()) - YEAR(Sales[Order  
Date]))  
    )
```

6.3. Análises Avançadas com Inteligência de Tempo

Utilizando as funções de inteligência de tempo, você pode realizar comparações avançadas entre períodos e análises de tendências.

- **Exemplo:** Comparar as vendas do mês atual com o mesmo mês do ano anterior.

DAX

```
Sales YoY Comparison =
SUM(Sales[Sales Amount]) -
CALCULATE(SUM(Sales[Sales Amount]), SAMEPERIODLASTYEAR(Sales[Order Date]))
```

6.4. Exercícios Práticos e Estudos de Caso

Vamos simular alguns estudos de caso onde você pode aplicar o que aprendeu:

- **Estudo de Caso 1:** Uma empresa quer saber se deve continuar investindo em um produto específico. Crie uma medida que analise o crescimento das vendas desse produto nos últimos três anos.
- **Estudo de Caso 2:** Analise o impacto de uma promoção de vendas realizada no último trimestre e compare os resultados com o trimestre anterior.

Esses exercícios práticos ajudarão a consolidar seu conhecimento e a se preparar para situações do mundo real, onde você precisará usar DAX para responder perguntas de negócios e fornecer insights valiosos.

Resumo: Práticas e desafios reais como a construção de KPIs, criação de medidas complexas, e análises avançadas com inteligência de tempo são fundamentais para aplicar o conhecimento de DAX em cenários de negócios do dia a dia. Esses exercícios ajudarão a solidificar sua compreensão e a prepará-lo para o uso prático de DAX.

Isso conclui a seção de práticas e desafios reais. Podemos seguir para as boas práticas e recursos adicionais?

You said:



ok

ChatGPT said:



Vamos continuar.

7. Boas Práticas e Recursos Adicionais

A última seção deste tutorial é focada em garantir que você continue a melhorar suas habilidades em DAX e mantenha seu trabalho eficiente e de alta qualidade. Aqui, veremos algumas boas práticas para escrever DAX, bem como recursos adicionais para aprendizado contínuo.

7.1. Debugging e Diagnóstico em DAX

- **Use o DAX Studio:** O DAX Studio é uma ferramenta poderosa para analisar e otimizar suas expressões DAX. Ele permite que você veja o impacto de suas consultas, examine o tempo de execução e explore a estrutura das suas tabelas e medidas.
- **Verifique o Impacto das Medidas:** Sempre que criar uma nova medida, verifique seu impacto no desempenho do modelo. Use ferramentas de diagnóstico no Power BI, como a "Performance Analyzer", para entender quais medidas podem estar causando lentidão.
- **Divida Cálculos Complexos:** Se uma medida ou coluna calculada for muito complexa, considere dividi-la em partes menores para facilitar a depuração e o entendimento. Isso também pode melhorar a performance.

7.2. Dicas de Performance e Otimização

- **Minimize o Uso de Colunas Calculadas:** Sempre que possível, prefira medidas a colunas calculadas, pois estas últimas aumentam o tamanho do modelo e podem afetar o desempenho.
- **Evite Relacionamentos Bidirecionais Desnecessários:** Relacionamentos bidirecionais podem criar loops e complicar o cálculo de medidas. Use-os com moderação e apenas quando necessário.
- **Agrupe Dados Quando Possível:** Se seu modelo estiver ficando muito grande, considere agrupar dados antes de carregá-los no Power BI, reduzindo a granularidade e o tamanho dos dados que precisam ser processados.

7.3. Ferramentas e Comunidades para Aprofundamento

- **Documentação Oficial do DAX:** A [documentação oficial da Microsoft](#) é uma excelente fonte para referências e detalhes sobre todas as funções DAX.
- **Comunidade Power BI:** Participe da [Comunidade Power BI](#), onde você pode fazer perguntas, compartilhar seu trabalho e aprender com outros profissionais.
- **Cursos Avançados:** Considere fazer cursos avançados, como o [curso de DAX da SQLBI](#) para aprofundar seu conhecimento em DAX e práticas de modelagem.
- **Livros Recomendados:** "The Definitive Guide to DAX" de Marco Russo e Alberto Ferrari é um excelente recurso para quem deseja dominar DAX em um nível avançado.

Resumo: Adotar boas práticas como o uso de ferramentas de diagnóstico, otimização de desempenho e participação em comunidades pode ajudar você a continuar aprimorando suas habilidades em DAX. Com esses recursos adicionais, você pode se manter atualizado e resolver problemas de modelagem de dados com maior eficiência.

Isso conclui o nosso tutorial de DAX de "zero to hero". Se você tiver alguma dúvida ou precisar de mais alguma coisa, estou à disposição! Como você se sente em relação a tudo que vimos?

You said:



Qual o arquivo do DW que voce utilizou?

ChatGPT said:



Neste tutorial, mencionei o uso do conjunto de dados **AdventureWorksDW** como exemplo, que é um banco de dados de exemplo fornecido pela Microsoft. Ele contém dados de uma empresa fictícia de vendas e é amplamente utilizado em tutoriais de Power BI, SQL Server e DAX.

O **AdventureWorksDW** inclui várias tabelas como ``Sales``, ``Product``, ``Customer``, ``Date``, entre outras. Essas tabelas são estruturadas para simular um data warehouse (DW) típico, com fatos e dimensões, facilitando análises complexas e o uso de DAX.

Se você precisar de um link ou instruções para obter o banco de dados **AdventureWorksDW** e usá-lo em seus projetos, posso fornecer orientações sobre como baixá-lo e configurá-lo no seu ambiente de trabalho.

You said:



Me manda o link para eu baixar

ChatGPT said:



Você pode baixar o banco de dados **AdventureWorksDW** diretamente do site da Microsoft. Aqui está o link para a página oficial de downloads:

[Download AdventureWorks Sample Databases](#)

Na página, você encontrará várias versões do banco de dados AdventureWorks, incluindo o **AdventureWorksDW** (Data Warehouse). Escolha a versão que melhor se adapta ao seu ambiente de trabalho (SQL Server ou outra versão específica). As instruções de instalação também estão disponíveis na mesma página.

Caso precise de ajuda para instalar ou configurar o banco de dados, estou à disposição!

You said:



Qual que eu baixo? Tem vários disponíveis

ChatGPT said:



Na página de download, você verá várias opções para o banco de dados AdventureWorks. Para o propósito deste tutorial de DAX, você deve baixar a versão **AdventureWorksDW** (Data Warehouse). Aqui estão as opções disponíveis e qual escolher:

1. **AdventureWorksDW2019.bak** (ou uma versão mais recente):

- o Esta é a versão mais recente do banco de dados de Data Warehouse. Se você estiver utilizando o SQL Server 2019 ou posterior, essa é a melhor opção. O arquivo `.bak` é um backup do banco de dados que pode ser restaurado diretamente no SQL Server.

2. **AdventureWorksDW2017.bak** (ou versões anteriores):

- o Se você estiver utilizando uma versão mais antiga do SQL Server (2017, 2016, etc.), escolha a versão correspondente para garantir a compatibilidade.

3. **AdventureWorksDW2019.bacpac**:

- o Se você estiver usando o Azure SQL Database, essa é a opção recomendada. O arquivo `.bacpac` é específico para importação no Azure.

Passo a Passo:

• Para SQL Server (Local ou em um Servidor):

1. Baixe o arquivo `.bak` correspondente à versão do seu SQL Server.
2. Abra o SQL Server Management Studio (SSMS).
3. Restaure o banco de dados usando a opção "Restore Database..." e selecione o arquivo `.bak`.

• Para Azure SQL Database:

1. Baixe o arquivo `.bacpac`.
2. Importe o arquivo `.bacpac` no Azure SQL Database usando o Azure Portal ou o SQL Server Management Studio.

Escolha a versão do AdventureWorksDW que corresponde à sua versão do SQL Server. Se precisar de mais alguma orientação sobre como fazer a restauração ou a configuração, estou aqui para ajudar!

ChatGPT pode cometer erros. Considere verificar informações importantes.