

Microsoft Power BI

O que é?

É uma ferramenta de análise de dados e visualização interativa criada pela Microsoft. Ele permite transformar dados brutos de diferentes fontes em relatórios e dashboards (painéis visuais) que facilitam a tomada de decisões com base em dados. Monitoramento e compartilhamento, fácil visualização, etc.

Banco de Dados:

Um sistema onde armazenamos informações organizadas para serem consultadas e usadas depois. Podendo ser uma coleção de tabelas com dados, de forma estruturada.

Importação dos Dados:

O Power BI é flexível para puxar dados de várias fontes, não apenas de bancos de dados, mas também outros tipos de arquivos e serviços.

Fontes de bancos de dados:

- SQL Server
- Oracle
- MySQL
- PostgreSQL

Outras fontes/Arquivos simples:

- Arquivos Excel (.xlsx)
- Arquivos CSV (.csv)
- Arquivos TXT
- JSON

O que é Power Query?

Power Query é uma ferramenta dentro do Power BI que serve para importar, transformar e preparar os dados antes de usar nas análises.

Resumo Geral

Diferença entre Duplicar e Referência no Power Query

Quando clicamos em duplicar a tabela importada no Power Query há uma independência entre as tabelas, quando queremos dependência clicamos em referência.

Tabela Fato

É a tabela que registra eventos ou ações (ex: vendas, acessos, chamadas) e contém os valores numéricos que você quer analisar, como quantidade, valor, tempo. Se observar que possui valores e datas pode afirmar que é uma tabela fato.

Tabela Dimensão

É a tabela que contém informações descritivas (ex: nome do produto, cliente, data) usadas para filtrar, agrupar ou dar contexto aos dados da tabela fato. Não possui duplicatas, repetições dos elementos.

DAX (Data Analysis Expressions)

Uma linguagem de fórmulas usada dentro do Power BI (e também no Excel avançado) para fazer cálculos e criar análises nos dados. Feito especialmente para analisar dados grandes e criar relatórios inteligentes no Power BI.

Podemos utilizar as funções DAX de 3 formas: Tabela, Medidas e Colunas calculadas.

- Criar colunas calculadas (ex: criar uma nova coluna que soma preços e impostos).
- Criar medidas (ex: calcular o total de vendas, média, contagem, etc.).
- Fazer comparações entre datas, como crescimento em relação ao mês anterior.
- Filtrar dados de forma inteligente (ex: somar só as vendas de um produto específico).

Exemplo:

Se você tem uma tabela com produtos e preços, pode usar DAX para:

Preço Total = Produto[Preço Unitário] * Produto[Quantidade] → Coluna Calculada
(Intellisense lê direto as funções da tabela por estar em uma tabela específica)

Ou

Total Vendas = SUM(Vendas[Valor]) → Medida (Utiliza "SUM" pois o intellisense não consegue ver na parte de "Nova Medida")

Coluna Calculada x Medida

As colunas calculadas criam uma nova coluna dentro da tabela, calculando um valor para cada linha individualmente. Isso significa que o resultado é armazenado permanentemente na tabela, ocupando espaço no modelo de dados. Essas colunas são úteis quando você precisa que o valor calculado faça parte da estrutura da tabela para usar em outras operações.

Já as medidas funcionam de forma diferente: elas não criam uma nova coluna na tabela, mas fazem cálculos dinamicamente no momento em que você exibe o relatório ou visual. Ou seja, a medida calcula o resultado "na hora", considerando os filtros e contextos aplicados, sem ocupar espaço fixo no modelo. Por isso, medidas costumam ser mais

eficientes para somas, médias, totais e análises que precisam se adaptar conforme o usuário interage com o relatório.

Por isso, é preferível usar medidas sempre que possível, pois o cálculo só é feito quando realmente necessário, tornando o processamento mais leve e o modelo mais rápido.

Resumindo para o uso básico:

- Coluna calculada = resultado por linha.
- Medida = resultado agregado (total, média, etc.) dinâmico.

Relacionamento

No Power BI, relacionamento é o que liga duas tabelas entre si.

Quando há várias tabelas diferentes — como produtos, clientes, vendas, datas — o Power BI precisa entender como os dados de uma tabela se conectam com os da outra.

Isso é feito criando um relacionamento entre elas, usando colunas em comum, como ID

Sem relacionamento, o Power BI **não sabe como combinar os dados**.

Por exemplo:

- A tabela de **vendas** diz que o cliente 105 comprou o produto 20.
- Mas... quem é o cliente 105? E qual é o produto 20?

Essas informações estão em outras tabelas (Clientes e Produtos).

Para que o Power BI entenda que o cliente 105 é o “João”, e o produto 20 é uma “Caneta”, você precisa ligar (relacionar) essas tabelas entre si.

O que significa “um para muitos”?

Esse é o tipo de relacionamento mais comum:

- Na tabela Dimensão, por exemplo tabela produto, cada produto aparece **uma vez só** → isso é o **lado “um”**. (Não há repetição na tabela d)
- Na tabela de **vendas**, o mesmo produto pode aparecer **várias vezes** → isso é o **lado “muitos”**. (Há repetição da tabela Fato)

O Relacionamento serve para tudo se conectar e fazer mais sentido na leitura dos dados.

Drill Down e Through

Drill Down e Drill Through no Power BI são formas de explorar dados com mais profundidade:

- **Drill Down:** você clica em um gráfico para ver detalhes de um nível inferior (ex: de ano para mês).
- **Drill Up:** volta para o nível anterior.
- **Drill Through:** você clica em um item (ex: nome de um cliente) e vai para outra página com detalhes específicos só daquele item. Uma opção que é ativada na página na seção “Visualização” → Drill Through e colocar isso dentro de um botão redirecionando para essa página criada com informações de determinados dados.

FASES IMPORTANTES

Fase 1: Início do Projeto

Criar um novo relatório

- Começa com uma tela em branco.

Fase 2: Importação dos dados

Clicar em "Obter dados" → Escolher a fonte

- Pode ser Excel, CSV, SQL Server, Google Sheets, etc.
- Caso tenha vários arquivos semelhantes, use a opção "Pasta" para importar todos de uma vez.

Selecionar os arquivos e clicar em "Transformar dados"

- *Importante:* Usar "Transformar dados" para entrar no Power Query e tratar os dados antes de levá-los para o modelo.
- "Carregar" só se os dados já estiverem perfeitos (o que raramente acontece).

Fase 3: Tratamento dos dados no Power Query

- Aqui pode:
 - Renomear colunas
 - Alterar tipos de dados (texto, número, data, etc.)
 - Remover linhas em branco ou colunas inúteis
 - Juntar várias tabelas (se necessário)

- Criar colunas novas (se precisar)
- Aplicar filtros
- Corrigir erros nos dados

Finalizar o tratamento e clicar em "Fechar e aplicar"

- Isso leva os dados tratados para o Power BI, onde será feito o modelo e o dashboard.

Fase 4: Modelagem dos dados

Relacionar as tabelas corretamente

- Vá até a aba de **Modelo** (ícone de 3 bolinhas ligadas)
- Crie relações entre tabelas (ex: Clientes[ID] com Vendas[ID_Cliente])
- **Tabelas Fato**: geralmente são os dados principais (ex: vendas, compras)
- **Tabelas Dimensão**: informações auxiliares (ex: clientes, produtos, datas)

Fase 5: Criação das medidas e cálculos (DAX)

Criar **medidas** como:

- Soma de vendas
- Cálculo de lucro
- Margem percentual
- Quantidade vendida
- Média de avaliações, etc.

Fase 6: Criação do dashboard

Ir para a aba de **Relatório** (ícone com gráfico)

Inserir visuais (gráficos, tabelas, KPIs, filtros, etc.)

Organizar os elementos no layout de forma clara e profissional

Fase 7: Finalização e publicação

Ajustar formatações e cores

Publicar no Power BI Service (nuvem) caso queira compartilhar

Criar relatórios interativos com filtros, slicers e bookmarks

Resumo Detalhado das Partes

ETL e MODELAGEM DE DADOS É RECOMENDADO REASSISTIR O CURSO PELO FATO DE SER ALGO MAIS PRÁTICO QUE TEÓRICO

Linguagem DAX

Criar cálculos personalizados e análises avançadas em cima dos dados já carregados no seu modelo. DAX é a linguagem usada no Power BI para criar cálculos personalizados e análises avançadas. Ele permite ir além dos filtros e cliques da interface, possibilitando medir lucros, criar KPIs, comparar períodos, filtrar dados com múltiplas condições e trabalhar com relações entre tabelas. Aprender DAX é essencial para transformar dados em respostas inteligentes e flexíveis, tornando suas análises muito mais poderosas e profissionais, servindo também para dinamizar tarefas.

| | COLUNA CALCULADA | MEDIDA |
|---------------------------|---|--|
| DEFINIÇÃO | Campo novo adicionado à tabela, uma nova coluna, resultados linha por linha | Cálculo dinâmico e agregado, feito na hora do visual |
| CALCULADA QUANDO | No momento de carregamento do modelo | No momento de visualização (dinamicamente) |
| ARMAZENAMENTO | Fica armazenada na memória | Não ocupa espaço fixo |
| VISÍVEL EM FILTROS | Sim | Não (É só um valor, não uma coluna) |

Coluna calculada (Estáticas): Aparece como coluna na tabela, visível. Podendo ser criada por exemplo: Valor vendido = fVendas[valorproduto] * fVendas[quantidade]

Medidas (Dinâmicas): Aparece em cartões, matrizes, etc. Não aparece diretamente na tabela, e pode ser criada assim: Soma Valor Vendido = SUM(fVendas[valorvendido])

Agregação é o processo de **resumir um conjunto de dados em um único valor**, utilizando funções, por exemplo.

Funções Agregadoras

COUNT - conta quantos valores existem em uma coluna, ignorando os vazios, mesmo que estejam repetidos. Conta as linhas não nulas de uma coluna.

DISTINCTCOUNT - conta quantos valores diferentes (únicos) existem em uma coluna, também ignorando os vazios.

COUNTROWS - Recebe uma tabela em seu parâmetro, e faz a contagem de linhas dessa tabela.

SUM, AVERAGE - Somar ou tirar média de **uma coluna só**

Ex:

TotalQuantidade = SUM(Vendas[Quantidade]) ;

QtdPedidosComValor = COUNT(Pedidos[Valor])

Funções Agregadoras Iterativas

SUMX , AVERAGEX...

- ♦ Iteram linha por linha em uma tabela
- ♦ Avaliam uma expressão (geralmente envolvendo colunas)
- ♦ Depois somam (SUMX) ou calculam a média (AVERAGEX) dos resultados dessas expressões.

Terminam com X

Ex:

TotalVendas = SUMX(fVendas, fVendas[Quantidade] * fVendas[Preço Unitário])

Função Condicional Simples

IF - A função IF avalia uma condição e retorna um valor se ela for verdadeira, e outro valor se for falsa.

Ex: Classificação = IF(fVendas[price] + fVendas[frete] > 12000, "PREMIUM", "PLATINUM")
condição , verdadeiro, falso (opcional)

SWITCH - Para múltiplas condições;

Ex:

SWITCH(TRUE(), [price] + [frete] > 12000, "PREMIUM", [price] + [frete] > 7000, "GOLD", "PLATINUM"

→ "PLATINUM" após a vírgula seria equivalente ao Else.

Função de Tabela

FILTER - Filtrar registros de uma tabela com base em uma condição, sobrescreve filtros que já existem, diferente o KEEPFILTER que mantém os filtros

EX: FILTER(fVendas, Vendas[Quantidade] > 10), é comum usá-las dentro de funções iterativas, CALCULATE

ALLSELECTED - Retorna todas as linhas da tabela ou coluna, respeitando os filtros aplicados Externos pelo usuário na interface.

ALL - Retorna todas as linhas de uma tabela ou todos os valores de uma coluna, removendo todos os filtros aplicados; pode ter uma ou mais colunas. Além de não levar em conta qualquer tipo de filtro externo aplicado, não leva em conta as linhas de outras colunas, exemplo da tabela abaixo.

Dada a tabela:

| CATEGORIA | PRODUTO | QUANTIDADE | FRETE | VALOR |
|-----------|-----------|------------|-------|-------|
| LEGUME | BETERRABA | 20 | 2,0 | 4,0 |
| LEGUME | CENOURA | 15 | 1,0 | 2,0 |
| FRUTA | BANANA | 35 | 5,0 | 8,0 |
| FRUTA | MAÇÃ | 50 | 3,0 | 5,0 |

Começaria criando uma coluna calculada com o total por linha de cada produto:

Total por Linha = fVendas[QUANTIDADE] * (fVendas[FRETE] + fVendas[VALOR])

Ficando assim - coluna calculada na tabela FATO:

| CATEGORIA | PRODUTO | QUANTIDADE | FRETE | VALOR | TOTAL POR LINHA |
|-----------|-----------|------------|-------|-------|--------------------|
| LEGUME | BETERRABA | 20 | 2,0 | 4,0 | $20 * (2+4) = 120$ |
| LEGUME | CENOURA | 15 | 1,0 | 2,0 | $15 * (1+2) = 45$ |
| FRUTA | BANANA | 35 | 5,0 | 8,0 | $35 * (5+8) = 455$ |
| FRUTA | MAÇÃ | 50 | 3,0 | 5,0 | $50 * (3+5) = 400$ |

Após disso, criaria uma medida:

Mínimo = CALCULATE(MIN(fVendas[Total por Linha]), ALL(fVendas[Total por Linhas]))

| CATEGORIA | PRODUTO | QUANTIDADE | FRETE | VALOR | Mínimo |
|-----------|-----------|------------|-------|-------|--------|
| LEGUME | BETERRABA | 20 | 2,0 | 4,0 | 45 |
| LEGUME | CENOURA | 15 | 1,0 | 2,0 | 45 |
| FRUTA | BANANA | 35 | 5,0 | 8,0 | 45 |
| FRUTA | MAÇÃ | 50 | 3,0 | 5,0 | 45 |

VALUES - Quando um nome de coluna é fornecido, retorna uma tabela com uma única coluna de valores exclusivos. Quando um nome de tabela é fornecido, retorna uma tabela com as mesmas colunas. Recebe apenas uma tabela ou uma coluna.

Se fosse criado utilizando uma medida dessa maneira:

Mínimo = CALCULATE(MIN(fVendas[Total por Linha]), VALUES(fVendas[Total por Linhas]))

| CATEGORIA | PRODUTO | QUANTIDADE | FRETE | VALOR | Mínimo |
|-----------|-----------|------------|-------|-------|--------|
| LEGUME | BETERRABA | 20 | 2,0 | 4,0 | 120 |
| LEGUME | CENOURA | 15 | 1,0 | 2,0 | 45 |
| FRUTA | BANANA | 35 | 5,0 | 8,0 | 455 |
| FRUTA | MAÇÃ | 50 | 3,0 | 5,0 | 420 |

OBS: VALUES(fVendas[order_id]) e ALL(fVendas[order_id]) retornam uma tabela com os

valores únicos da coluna order_id. Ambos removem duplicatas e espaços em brancos porque, quando usados numa única coluna, essas funções sempre retornam o conjunto distinto de valores para que você possa trabalhar com listas limpas, seja respeitando filtros (VALUES) ou ignorando-os (ALL).

Isso ocorre porque, no DAX, extrair uma coluna isolada cria uma tabela de valores distintos, facilitando análises e cálculos. Para obter todas as linhas, inclusive repetições, é preciso usar a tabela inteira, não só a coluna. O que daria o mesmo valor final se usasse um COUNTROWS(ALL/VALUES).

MAS HÁ DIFERENÇA!

ALL continua não respeitando os filtros, enquanto **VALUES** respeita os filtros. Tendo os resultados de acordo com as tabelas feitas acima.

Função Manipulação de Contexto

CALCULATE - Recebe em seu parâmetro: **CALCULATE(Expressão, [FILTRO])**, essa expressão pode ser uma medida ou um cálculo, e pode usar um ou mais filtros. Usada para criar medidas condicionais (Ex: vendas em determinada região ou ano), ignorar ou substituir filtros de relatórios, fazer análises de comparação entre períodos ou categorias, usar em conjunto com FILTER, ALL, VALUES, etc.

Ex:

Vendas Altas = CALCULATE(

SUM(FatoVendas[ValorVenda]),

FILTER(FatoVendas, FatoVendas[ValorVenda] > 1000))

A função **CALCULATE** serve para unir um cálculo com uma condição. Sozinho, o **SUM** apenas soma todos os valores de uma coluna, e o **FILTER** apenas filtra dados, mas não faz nenhuma conta. O CALCULATE entra como uma ponte entre os dois.

Ele pega o cálculo que queremos (como somar) e aplica o filtro que escolhemos (como Estado = "SP"). Quando usamos CALCULATE(SUM(Tabela[ValorVenda]), Tabela[Estado] = "SP"), estamos dizendo: "Some os valores da coluna ValorVenda, mas só das linhas onde o Estado for SP". Assim, o CALCULATE permite que o cálculo seja feito dentro de um filtro personalizado, mesmo que esse filtro não esteja visível na tela do relatório.

Podemos usar operadores lógicos na filtragem personalizada para buscar por mais estados ou categorias.

KEEPFILTERS - Usado normalmente dentro de CALCULATE, é um filtro que mantém filtros selecionados;

Ex:

FILTRO SELECIONADO APENAS PARA “SP”

```
Medida = CALCULATE(  
    SUM(Vendas[ValorVenda]),  
    KEEPFILTERS(Vendas[Categoria] = "Eletrônicos"))
```

→ Vai realizar a somatória de Eletrônicos apenas do estado de SP, se fosse FILTER faria a somatória de todos os estados, mas apenas na categoria “Eletrônicos”. Ambos respeitam o contexto das linhas de outras colunas!

CALCULATETABLE - A função CALCULATETABLE no DAX serve para retornar uma tabela modificando o contexto de filtro. Pode usar mais de um argumento no filtro. Ela aplica filtros sobre uma tabela base e recalcula o resultado de acordo com essas condições. É usada principalmente quando se quer gerar uma nova tabela temporária baseada em critérios específicos, como “todas as vendas do Sul” ou “clientes com mais de 10 compras”. Ela é muito usada dentro de funções como SUMX, ADDCOLUMNS, FILTER, onde é preciso iterar sobre dados filtrados. Diferente da CALCULATE, que retorna um único valor (como uma soma), a CALCULATETABLE retorna um conjunto de linhas

Variáveis

VAR - Reduz o número de medidas criadas, melhora desempenho e organização do código no DAX, usa VAR Nome_da_VAR e no final usa RETURN e chama a variável.

Ex:

Total Metas =

VAR TotalAbsoluto =

```
CALCULATE(  
    [Total Vendido],  
    ALL(dVendedores),  
    VALUES(dVendedores[seller_state]))
```

Explicação:

Criado uma medida “Total Metas” e após isso foi criado uma variável, CALCULATE recalcula a medida [Total Vendido] removendo todos os filtros da tabela dVendedores, mas mantendo apenas o filtro de seller_state (estado do vendedor).

Entendendo a Lógica:

```

Valor Meta por Período =
VAR TotalAbs =
    CALCULATE([Total Vendido],
        ALL(dCalendario),
        VALUES(dCalendario[Ano]))

VAR PercentualRelativoMes =
    DIVIDE([Total Vendido], TotalAbs)

VAR MetaAbs =
    CALCULATE([Total Metas],
        ALL(dCalendario),
        VALUES(dCalendario[Ano]))

RETURN
MetaAbs * PercentualRelativoMes

```

Para construirmos nossa lógica de cálculo, começamos com a **primeira variável**, chamada TotalAbsoluto. Ela usa a função CALCULATE para forçar o cálculo do [Total Vendido], porém **ignorando os filtros aplicados sobre a tabela de vendedores** com ALL(dVendedores), mas mantendo o filtro dos **estados dos vendedores** com VALUES(dVendedores[seller_state]). O resultado é um valor **total de vendas por estado**, independentemente de outros filtros aplicados, como mês ou ano.

Em seguida, temos a **segunda variável**, chamada PercentualRelativoMes. Aqui usamos a função DIVIDE para calcular a **porcentagem que cada mês representa dentro do total absoluto**. Fazemos isso dividindo [Total Vendido] por [TotalAbsoluto]. Como [Total Vendido] é uma **medida dependente do contexto**, ao colocá-la em uma matriz com ano e mês, ela **automaticamente se recalcula para cada linha**, ou seja, para cada mês de cada ano. Isso gera uma **porcentagem mês a mês**, mostrando a contribuição daquele mês no total geral do estado.

Por fim, temos a **terceira variável**, chamada MetaRelativa. Nela, usamos CALCULATE para obter a [Total Metas] (uma medida com a meta de vendas), **removendo o filtro completo de calendário** com ALL(dCalendario) e **mantendo apenas o filtro de ano** com VALUES(dCalendario[ano]). Isso nos dá a **meta total para cada ano**. Depois, multiplicamos esse valor pela PercentualRelativoMes. O resultado final será a **meta de vendas proporcional para cada mês**, ou seja, uma **meta distribuída mês a mês dentro de cada ano**, de forma proporcional à venda que aquele mês representou no total de vendas do estado.

Função Inteligência de Tempo

SAMEPERIODLASTYEAR - A função SAMEPERIODLASTYEAR no DAX retorna uma tabela de datas equivalente ao mesmo período do ano anterior.

Ela é usada, por exemplo, para comparar vendas de janeiro de 2024 com janeiro de 2023, mantendo a mesma granularidade (mês, dia, etc.), normalmente usada com CALCULATE.

A função SAMEPERIODLASTYEAR recebe como parâmetro uma coluna ou tabela de datas e retorna o mesmo intervalo de datas do ano anterior. Ela é usada principalmente para comparar valores entre anos, respeitando automaticamente os filtros aplicados no relatório. Por exemplo, se você selecionar o ano de 2023, essa função retorna o mesmo período em 2022.

Ex:

```
VendasAnoAnterior = CALCULATE([TotalVendas],  
SAMEPERIODLASTYEAR('Calendario'[Data]))
```

OU

Acumulado ano passado =

```
CALCULATE(  
    [Total vendido],  
    SAMEPERIODLASTYEAR(DATESYTD(dCalendario[Data])))
```

PREVIOUSYEAR - A função PREVIOUSYEAR recebe obrigatoriamente uma coluna de datas como argumento e, opcionalmente, um intervalo de datas personalizado. Ela retorna todas as datas do ano completo anterior ao período atual, pegando o total absoluto desse ano.

Quando usada em uma medida, calcula o total do ano anterior inteiro e compara esse valor com cada linha do período atual, como cada mês de um ano, repetindo o mesmo total anual para todas as linhas.

Diferente da SAMEPERIODLASTYEAR, que retorna o mesmo intervalo do ano anterior para cada linha, permitindo comparação mês a mês, a PREVIOUSYEAR sempre traz o total anual completo do ano anterior aplicado a todas as linhas do contexto atual.

DATEADD - A função DATEADD é uma função de inteligência de tempo que recebe como argumentos obrigatórios uma coluna de datas, um número inteiro (positivo ou negativo) para deslocar o período, e uma unidade de tempo (como dia, mês ou ano). Ela retorna uma tabela com as datas deslocadas conforme os parâmetros. O argumento do número e da unidade são obrigatórios para definir o deslocamento, não há argumentos opcionais além desses três.

DATEADD é usada para comparar períodos relativos, como calcular valores do mês ou ano anterior. No exemplo, ela desloca as datas da coluna dCalendario[Data] em -1 mês, fazendo a medida calcular o total vendido no mês anterior ao contexto atual

Ex: Total Vendido Mês Anterior = CALCULATE([Total Vendido],
DATEADD(dCalendario[Data], -1, MONTH))

Explicação de Lógica - Exercício

Variação ano a ano =

VAR variacao = [Total vendido] - [Total Vendido do Ano Anterior]

RETURN

DIVIDE(variacao, [Total Vendido do Ano Anterior])

→ Essa medida calcula a variação percentual de vendas de um ano para o anterior. Primeiro, ela cria uma variável chamada *variacao*, que armazena a diferença entre o total vendido no ano atual e o total vendido no ano anterior. Em seguida, ela retorna essa diferença dividida pelo total vendido do ano anterior, usando a função *DIVIDE*, que evita erro de divisão por zero. O resultado mostra o quanto as vendas cresceram ou caíram em relação ao ano anterior, em termos percentuais. É usada em uma matriz para comparar o desempenho ano a ano.

TOTALYTD - A função *TOTALYTD* no DAX é uma função de inteligência de tempo que calcula o total acumulado desde o início do ano até a data fornecida. Ela é usada principalmente para análises anuais, permitindo comparar o desempenho até a data atual de forma contínua, acumulando os valores ao longo do ano.

<expressão>: A medida ou cálculo que você deseja somar (como o total de vendas).

<coluna_de_datas>: A coluna de datas contínuas, geralmente a coluna de data do seu calendário.

<filtros> (opcional): Filtros adicionais que você pode aplicar, se necessário.

Ex: Total Acumulado = *TOTALYTD*([TotalVendas], 'Calendario'[Data])

Ela vai acumular o valor da medida [TotalVendas] mês a mês, respeitando o ano de cada linha. No final de cada ano, o valor acumulado será reiniciado e começará de novo no próximo ano.

DATESBETWEEN - A função *DATESBETWEEN* no DAX é usada para retornar uma tabela de datas entre duas datas específicas. Ela permite filtrar e trabalhar com um intervalo de datas personalizável, o que é útil para cálculos dentro de um determinado período, como um intervalo de dias, meses ou anos.

<coluna_de_datas>: A coluna que contém as datas que você deseja filtrar (geralmente uma coluna de data de um calendário).

<data_inicial>: A data de **início** do intervalo.

<data_final>: A data de **fim** do intervalo.

Ex:

```
Vendas acumulado =  
CALCULATE(  
    [Total vendido],  
    DATESBETWEEN(  
        dCalendario[Data],  
        BLANK(),  
        MAX(dCalendario[Data])))
```

O CALCULATE é responsável por modificar o contexto de avaliação e calcular a medida [Total vendido] (que deve ser uma medida já criada para somar o valor de vendas).

A função DATESBETWEEN cria um intervalo de datas entre a data inicial e a data final para o cálculo.

Aqui, o valor de BLANK() como data inicial tem um comportamento interessante. BLANK() basicamente significa "sem data", e ele indica que a função vai começar do começo (do primeiro valor possível) da coluna de datas.

Usando BLANK() como data inicial, a função pega o primeiro valor de data disponível na coluna 'dCalendario'[Data].

A data final é determinada por MAX(dCalendario[Data]), que retorna a última data do contexto atual (por exemplo, se você está olhando para 2023 e o mês é fevereiro, ele vai considerar até fevereiro de 2023).

DATESINPERIOD - Retorna uma tabela de datas contínuas, indo para trás ou para frente a partir de uma data de referência. Exige quatro argumentos obrigatórios: a coluna de datas, a data inicial, o número do intervalo e a unidade (como DAY, MONTH ou YEAR).

Média móvel (porque amplia o período de análise em torno de uma data base, suavizando

oscilações e deixando o gráfico mais claro e fácil de interpretar), totais acumulados personalizados, análise de períodos móveis (ex: últimos 3 meses, últimos 7 dias etc.).

Ex:

`DATESINPERIOD('Calendario'[Data], MAX('Calendario'[Data]), -3, MONTH)`

→ Pega os últimos 3 meses até a data atual do co

- **DATESYTD** entende o ano automaticamente com base no calendário.
Ele começa sempre no primeiro dia do ano (ou fiscal, se configurado) e vai até a data atual do contexto.
- **DATESINPERIOD**, por outro lado, não sabe o que é "ano" ou "mês" no calendário.
Ele apenas conta quantos dias, meses ou anos para trás ou para frente, a partir de uma data base.

Função de Classificação

RANKX - A função RANKX no Power BI serve para classificar valores de uma tabela com base em uma expressão, como por exemplo o total de vendas. Ela retorna a posição (ou "ranking") de cada item dentro de um grupo, do maior para o menor (ou vice-versa). É útil, por exemplo, para saber quais produtos mais vendem ou quais regiões têm melhor desempenho.

Ex:

`Ranking Vendas = RANKX(ALL(Produtos), [Total Vendido])`

Função de Filtragem Ordenação

TOPN - É classificada como uma função de filtragem baseada em ordenação.

Ela retorna as N primeiras linhas de uma tabela, com base na ordem decrescente (ou crescente) de uma expressão de valor, como vendas, lucros, etc.

Características:

- Recebe como argumentos:

- Número N (obrigatório): quantas linhas retornar.
- Tabela (obrigatório): de onde tirar os dados.
- Expressão de ordenação (obrigatório): o critério para classificar.
- Ordem (opcional): DESC (padrão) ou ASC.

É usada para **filtrar os melhores (ou piores) registros** com base em uma métrica, como os 5 produtos mais vendidos ou os 3 vendedores com menor faturamento.

Ex:

TOPN(5, Vendas, [TotalVendas], DESC)

→ Retorna os 5 registros da tabela Vendas com os maiores valores em [TotalVendas].

Explicação Lógica

Código:

% Participacao Top Vendedores =

VAR TotalAbsTopVendedores =

CALCULATE(

[Total vendido],

TOPN(

[RankingSelecionado],

ALL(dVendedores[nome]),

[Total vendido]))

VAR participacaoVendedoresTop = DIVIDE([Top Vendedores], TotalAbsTopVendedores)

VAR participacaoTopAbs = DIVIDE([Top Vendedores],[Total vendido])

RETURN

IF(ISINScope(dVendedores[nome]),participacaoVendedoresTop, participacaoTopAbs)

A variável **TotalAbsTopVendedores** usa a função **CALCULATE** para recalcular a medida Total vendido, aplicando um novo filtro. Esse filtro é criado pela função **TOPN**, que seleciona os N primeiros nomes de vendedores com base em seus parâmetros, considerando todos os nomes da tabela sem nenhum filtro aplicado na coluna de nomes, graças ao uso do **ALL(dVendedores[nome])**. O Total vendido dentro do **TOPN** funciona como critério de

ordenação, avaliando para cada nome o total vendido correspondente e é feito o ranqueamento do top N por TOPN. Assim, o **CALCULATE** soma o total vendido apenas para esses N vendedores selecionados, garantindo que o cálculo considere os top N em vendas, independentemente de filtros aplicados na coluna de nomes, mas respeitando possíveis filtros em outras colunas da tabela.

Depois de calcular o total absoluto vendido pelos Top 10 vendedores, a variável **participacaoVendedoresTop** calcula a **participação relativa** de cada vendedor dentro desse grupo de Top 10. Para isso, usa a função **DIVIDE**, dividindo o valor de **[Top Vendedores]** (ou seja, o total vendido por esse vendedor caso ele esteja entre os Top 10) pelo total geral desses Top 10 (**TotalAbsTopVendedores**). Essa proporção representa quanto cada vendedor contribuiu dentro dos Top 10.

A variável **participacaoTopAbs**, por sua vez, calcula a **participação absoluta do grupo Top 10** no total vendido geral, dividindo o total dos Top 10 (**[Top Vendedores]**) pelo **[Total vendido]** de todos os vendedores — assim, mostra qual a fatia que os Top 10 representam no total.

O **ISINSCOPE** verifica se estamos vendo cada vendedor individualmente.

Se sim, mostra a participação de cada um dentro do Top 10.

Se não, mostra a participação total do grupo Top 10 no total geral.

Assim, a medida se adapta ao tipo de visual: detalhado ou resumido.

SELECTEDVALUE(RankingVendedores[ID], 10) foi criada uma tabela chamada RankingVendedores com duas colunas: uma chamada ID (com os valores 5, 10, 20, 30) e outra chamada Texto (com "TOP 5", "TOP 10", "TOP 20", "TOP 30"). Essa tabela foi colocada em um filtro para o usuário escolher qual "TOP N" quer ver.

Depois, foi criada a medida RankingSelecionado =

SELECTEDVALUE(RankingVendedores[ID], 10), que pega o número escolhido pelo usuário (por exemplo, 10). Se o usuário não escolher nada, o valor padrão usado será 10.

Essa variável RankingSelecionado foi então usada dentro de outra medida para deixar o cálculo totalmente dinâmico, se adaptando à seleção feita no filtro.

SELECTEDVALUE - É como um IF/ELSE que:

- retorna o valor único selecionado,
- ou retorna um valor padrão (como um ELSE) se houver vários ou nenhum selecionado.

Design de Dashboards

Parte do curso onde é ensinado a manipular, ajustar e organizar elementos visuais. São mais intuitivos e exige prática, há pouquíssima teoria.

KPI → (Key Performance Indicator) é um indicador visual usado para acompanhar o desempenho de uma métrica em relação a uma meta estabelecida. Ele é muito útil em dashboards para mostrar de forma rápida se algo está indo bem, mal ou dentro do esperado

Ex:

Kpi % Meta =

SWITCH(

TRUE(),

[% Atingimento da meta] < 0.9, "🔴", "✅")

Formato → Editar interações:

Você tem um gráfico de colunas com os anos 2021, 2022 e 2023, e também tem:

- Um gráfico de pizza com vendas por região,
- Um cartão mostrando o total de vendas.

Agora você clica em 2022 no gráfico de colunas.

O que acontece com os outros visuais depende da interação configurada:

Filtro (ícone de funil)

- O gráfico de pizza mostra só os dados de 2022.
- O cartão mostra só o total de vendas de 2022.



Tudo muda com base no ano selecionado.

Realce (ícone de alvo ou semicírculo)

- O gráfico de pizza mantém todos os anos, mas dá destaque visual apenas para os dados de 2022.
- O restante fica mais claro (esmaecido), mostrando comparação.

● Você vê 2022 em destaque, sem esconder os outros anos.
















Sem interação (círculo cortado)

- O gráfico de pizza e o cartão não mudam nada.
- Continuam mostrando os dados completos (de todos os anos).

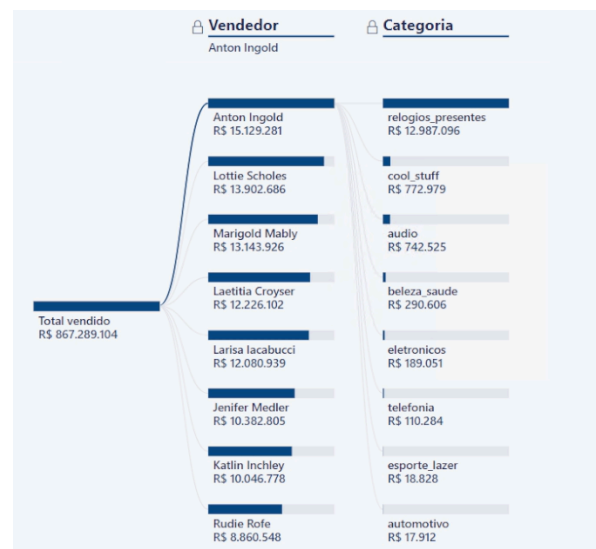
● É como se o clique em 2022 não existisse para eles.

Exibição → Segmentação de dados:

Serve para analisar a sincronia entre as páginas. Quando clico em um ano em determinada aplicam esse mesmo filtro

| imagem | vendedor | faturamento | % meta |
|---|---------------------|-----------------|----------|
|  | Anton Ingold | R\$ 15.129.281 | 652,76% |
|  | Lottie Scholes | R\$ 13.902.686 | 652,76% |
|  | Marigold Mably | R\$ 13.143.926 | 652,76% |
|  | Laetitia Croyser | R\$ 12.226.102 | 652,76% |
|  | Larisa Iacabucci | R\$ 12.080.939 | 652,76% |
|  | Jenifer Medler | R\$ 10.382.805 | 652,76% |
|  | Katlin Inchley | R\$ 10.046.778 | 652,76% |
|  | Rudie Rofe | R\$ 8.860.548 | 652,76% |
|  | Ada Idney | R\$ 8.346.226 | 652,76% |
|  | Rochester Lowthorpe | R\$ 8.027.958 | 70,16% |
|  | Laney Freire | R\$ 6.511.275 | 56,55% |
|  | Dag Bohey | R\$ 6.392.594 | 5508,82% |
|  | Daisey Ramble | R\$ 6.244.638 | 652,76% |
|  | Roseanne | R\$ 5.992.378 | 652,76% |
|  | Total | R\$ 867.289.104 | 177,59% |

de visualização de
ação e melhora a



Quando você adiciona uma **formatação condicional no Total Vendido** da tabela, o Power BI muda a **cor do fundo ou do texto** de cada célula com base no valor. Assim, você consegue **destacar automaticamente os vendedores que venderam mais ou menos**, facilitando a análise visual dentro da tabela que já mostra **nome, estado, total vendido e % da meta**

Ficando dessa maneira:

Tooltips no Power BI são dicas de informação que aparecem quando passamos o mouse sobre um visual (como um gráfico ou tabela).

Exemplo

Imagine um
Quando v

- Nome
- Total
- Por

The screenshot shows a Power BI report with a table of sales data. The table has four columns: 'imagem', 'vendedor', 'faturamento', and '% meta'. The data is sorted by 'faturamento' in descending order. The background color of the table rows alternates between light blue and white. A tooltip menu is open over the table, showing options like 'Cor da tela de fundo', 'Cor da fonte', 'Barras de dados', and 'Ícones'. The 'Barras de dados' option is selected, and a sub-menu is visible showing 'Barras de dados' and 'Ícones'.

| imagem | vendedor | faturamento | % meta |
|--------------|---------------------|------------------------|----------------|
| | Anton Ingold | R\$ 15.129.281 | 652,76% |
| | Lottie Scholes | R\$ 13.902.686 | 652,76% |
| | Marigold Mably | R\$ 13.143.926 | 652,76% |
| | Laetitia Croyser | R\$ 12.226.102 | 652,76% |
| | Larisa Iacabucci | R\$ 12.080.939 | 652,76% |
| | Jenifer Medler | R\$ 10.382.805 | 652,76% |
| | Katlin Inchley | R\$ 10.046.778 | 652,76% |
| | Rudie Rofe | R\$ 8.860.548 | 652,76% |
| | Ada Idney | R\$ 8.346.226 | 652,76% |
| | Rochester Lowthorpe | R\$ 8.027.958 | 70,16% |
| | Laney Freire | R\$ 6.511.275 | 56,55% |
| | Dag Bohey | R\$ 6.392.594 | 5508,82% |
| | Daisy Ramble | R\$ 6.244.638 | 652,76% |
| | Roseanne | R\$ 5.992.378 | 652,76% |
| Total | | R\$ 867.289.104 | 177,59% |

Isso ajuda a ver mais detalhes sem poluir o gráfico. Pode ser criada como uma página convencional e alterando para dica de ferramenta, após isso, deverá ir na página com o

visual que deseja, clicar e em Geral → Ativar Dica de Ferramentas e selecionar a página que foi alterada para dica de ferramenta com o tooltip criado.

Exibição → Indicadores

É possível colocar em foco apenas um visual específico selecionado, podendo seguir um roteiro de apresentação de determinado elemento, visual ou página.

O indicador pode ser utilizado junto a seleção que fica ao lado do indicador. Com isso é possível criar uma apresentação selecionando um visual específico e deixando somente certos elementos do visual de acordo com a seleção sem que mude os elementos originais do relatório.

É possível adicionar botões usando seleção e indicadores (Aulas: 97,98)

Serviço Online do Power BI / Power BI Desktop

Segurança em Nível de Linha (RLS) - É uma forma de mostrar dados diferentes para cada pessoa, com base em quem está acessando o relatório.

Por exemplo:

Se um gerente de SP abrir o relatório, ele só vê os dados de SP.

Se um gerente do RJ abrir, só vê os dados do RJ. Ou seja, o RLS controla automaticamente quais linhas de dados cada usuário pode ver.

RLS Estático:

- Você define manualmente quais dados cada usuário pode ver. Modelagem → Gerenciar funções
- Exemplo: João vê SP, Maria vê RJ – isso é configurado direto no Power BI.

Após ter realizado os filtros desejados de acordo com os objetivos, deverá salvar e publicar novamente no Power BI Online. Ir nos 3 pontos do modelo semântico, clicar em segurança e adicionar os usuários da equipe de acordo com o filtro criado. Válido apenas para visualizador, membros e administradores não recebem o RLS, ou seja, não é aplicado. Além dos usuários serem adicionados a esse filtro devem também ser adicionados ao workspace.

RLS Dinâmico:

- O próprio relatório descobre quem é o usuário e filtra os dados automaticamente.
- Exemplo: João faz login → o Power BI olha uma tabela que diz que João vê SP → mostra só SP pra ele.

Para configurar o RLS Dinâmico no Power BI, o primeiro passo é compreender que o objetivo dessa segurança é permitir que **cada pessoa que acesse o relatório veja apenas os dados que ela tem permissão para visualizar**, com base em sua identidade (normalmente o e-mail corporativo usado para login no Power BI Service). Na Situação 1,

por exemplo, a ideia é que **um usuário final (como um gerente regional)** veja **apenas os dados do seu próprio estado**, e não de outros.

Etapa 1 – Criar a Tabela de Segurança (Controle de Acesso)

O primeiro passo prático é criar uma tabela que mapeia quais usuários têm acesso a quais estados. Essa tabela vai informar ao Power BI que, por exemplo, o usuário `joao.gerente@empresa.com` só pode visualizar dados do estado de São Paulo, enquanto `maria.gerente@empresa.com` deve ver apenas os dados do Rio de Janeiro.

No Power BI Desktop, vá até a aba “Página Inicial” e clique em “Inserir dados”. Na janela que se abre, crie uma tabela com pelo menos duas colunas: uma chamada Email e outra chamada Estado. Essa tabela será o seu controle de acesso. Após preenchê-la, clique em “Carregar”.

Etapa 2 – Relacionar a Tabela de Segurança com os Dados Reais

Agora vem uma parte essencial: relacionar a tabela de segurança com os dados reais do relatório, como a tabela de vendedores, que chamaremos aqui de Vendedores.

Essa etapa é obrigatória porque o Power BI só consegue propagar o filtro que está na tabela de segurança para as demais tabelas do modelo se houver um relacionamento entre elas.

No nosso caso, tanto a tabela de segurança (TabelaSeguranca) quanto a tabela de vendedores (Vendedores) possuem uma coluna chamada Estado. Ao relacionar essas colunas, estamos dizendo para o Power BI:

“Quando eu filtrar a TabelaSeguranca para mostrar apenas os dados do estado do usuário logado, quero que essa filtragem se estenda automaticamente para a tabela de Vendedores.”

Sem esse relacionamento, o Power BI aplicaria o filtro à TabelaSeguranca, mas não filtraria automaticamente os dados da Vendedores, fazendo com que o usuário continuasse vendo tudo.

Para criar o relacionamento:

1. Vá para a aba de modelo (ícone de diagrama à esquerda).
2. Arraste a coluna Estado da TabelaSeguranca até a coluna Estado da Vendedores.
3. Verifique se o relacionamento está como “muitos para um” (many-to-one), com a TabelaSeguranca no lado “um”.
4. Salve o relacionamento.

Agora, quando o Power BI filtrar a tabela de segurança com base no usuário logado, os dados da tabela Vendedores serão automaticamente filtrados também, e o usuário verá somente os vendedores do estado que ele tem acesso.

Etapa 3 – Criar a Regra de Segurança Dinâmica com DAX

Com a estrutura pronta, você vai dizer ao Power BI qual regra ele deve seguir para aplicar o RLS Dinâmico.

Na aba “Modelagem”, clique em “Gerenciar Funções” e depois em “Criar”. Dê um nome para a função, como RLS_Dinamico. Em seguida, selecione a tabela de segurança (TabelaSeguranca) e, no campo de filtro, insira a fórmula:

[Email] = USERPRINCIPALNAME()

Essa fórmula significa: “Filtre a TabelaSeguranca e pegue apenas a linha onde o e-mail da tabela for igual ao e-mail do usuário que está acessando o relatório.” A função USERPRINCIPALNAME() captura automaticamente o login do usuário no Power BI Service.

Salve essa função e, se quiser testar ainda no Desktop, clique em “Ver como função” (View as Role) para simular o comportamento com diferentes e-mails.

Etapa 4 – Publicar o Relatório no Power BI Service

Com tudo configurado, publique o relatório para o Power BI Service (online), pois o RLS só funciona de verdade na versão publicada. Clique em “Publicar” na aba “Página Inicial”, escolha o workspace desejado e aguarde o upload.

Etapa 5 – Testar a Segurança Online

Acesse <https://app.powerbi.com> e entre no workspace onde você publicou o relatório. Vá até o conjunto de dados, clique nos três pontinhos ao lado dele e selecione “**Segurança**”. Você verá a função RLS_Dinamico. Clique nela e depois clique em “**Ver como usuário**” para simular o acesso de um usuário específico. Digite o e-mail que está cadastrado na TabelaSeguranca, e o Power BI mostrará apenas os dados referentes ao estado daquele usuário.

Conclusão

O RLS Dinâmico com controle por estado baseado no usuário final funciona porque:

- A tabela de segurança define quem pode ver o quê.
- O relacionamento entre o campo Estado da tabela de segurança e a tabela de vendedores permite que o filtro da segurança se propague para os dados do

relatório.

- A função DAX com USERPRINCIPALNAME() detecta quem está logado e filtra a tabela de segurança dinamicamente.
- Com tudo isso junto, o Power BI mostra automaticamente somente os dados autorizados para cada pessoa, sem precisar criar regras separadas para cada usuário.

Gateway Pessoal - O gateway pessoal no Power BI é um programa que você instala no seu computador para ajudar o Power BI online a pegar dados que estão guardados na sua máquina ou em arquivos locais.

O que ele faz:

Ele conecta seus dados locais com o Power BI na nuvem e atualiza automaticamente seus relatórios sem precisar fazer isso manualmente.

Como funciona:

Você instala o gateway, conecta ele às suas fontes de dados, publica seu relatório no Power BI online, e o gateway atualiza os dados automaticamente quando programado.

Tudo isso acontece de forma segura e sem você precisar abrir nada no seu computador.

Gateway enterprise: para uso corporativo, instalado em servidores, para múltiplos usuários e fontes.

O que são Dataflows?

Dataflows são processos no Power BI que permitem extrair, transformar e carregar dados (ETL) de várias fontes para um armazenamento comum na nuvem, chamado Common Data Model (CDM).

Para que servem?

- Centralizar e organizar dados para que possam ser usados por vários relatórios e dashboards.
- Evitar repetir o mesmo processo de limpeza e transformação de dados em vários relatórios.
- Facilitar o gerenciamento dos dados, deixando os relatórios mais leves.
- Permitir que diferentes usuários ou equipes trabalhem com os mesmos dados confiáveis e atualizados.

O que fazem?

- Buscam dados de fontes diversas (Excel, banco de dados, APIs, etc.).
- Aplicam transformações (filtrar, unir tabelas, limpar dados, criar colunas).
- Salvam esses dados já prontos no Power BI Service para serem usados por relatórios.

Como funciona?

1. Você cria um dataflow no Power BI Service.
2. Configura as conexões para as fontes de dados.
3. Usa o Power Query (mesmo editor do Power BI Desktop) para transformar e preparar os dados.
4. Salva o fluxo, que atualiza automaticamente os dados na nuvem.
5. Os relatórios conectam-se ao dataflow para usar esses dados prontos.

Os dataflows também usam o Power Query (mesmo visual e lógica), mas rodam direto no Power BI Online, e os dados são salvos na nuvem, no Common Data Model (CDM).

Power Automate

Power Automate no Power BI é uma integração que permite automatizar tarefas e fluxos de trabalho diretamente a partir de relatórios e dashboards do Power BI. Ele conecta o Power BI ao Power Automate (antes chamado de Microsoft Flow), que é uma ferramenta da Microsoft para criar fluxos automatizados entre aplicativos e serviços. Power Automate no Power BI serve para automatizar ações com base em eventos nos relatórios, como por exemplo:

- Quando um valor ultrapassa um limite;
- Quando um usuário clica em um botão (Power Automate visual);
- Quando há atualização de dados

Posso passar um fluxo de dados do power bi para o teams.