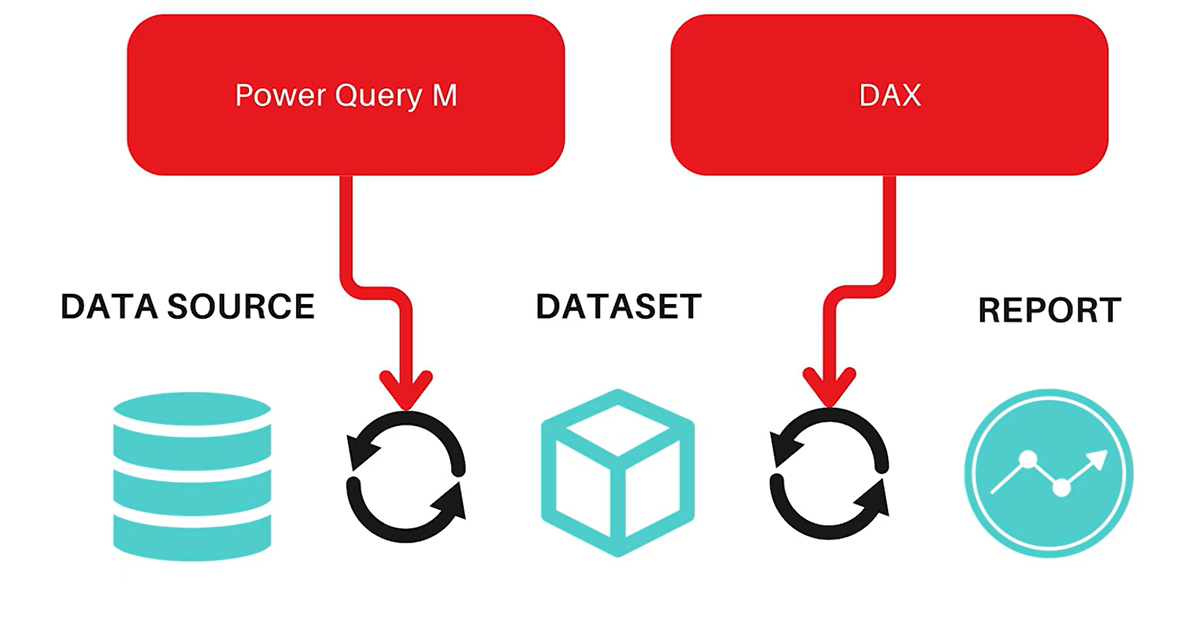
# POWER BI

## EDX

## SEMANA 1

### Início



So, if Power Query uses the M language

to do all of its amazing stuff,

what does the data model in Power BI use?

If the data set is technically a different component

that runs in the background of Power BI

separate from the report,

how does your report communicate with the data set?

The answer to both questions is DAX.

DAX is the other big language used in Power BI

and it stands for Data Analysis eXpressions.

DAX serves multiple purposes in Power BI.

Behind the scenes, it's actually the query language

that is used by the report

to retrieve the data it needs from the dataset.

For example, if you build a line chart in your report,

Power BI will dynamically generate a DAX query

based on the data you put in that chart

and will then send that DAX query down to the dataset

to calculate and then retrieve the data it needs,

and then it will render the results in your chart.

**We'll show this process in action**

in one of the lessons in this course.

The most direct way

that you'll use DAX in Power BI development

is by writing measures.

Measures are formulas stored in your data model

that dynamically calculate in our report

depending on what filters we have applied

and how we're using them in our visuals.

Now, because they are dynamically calculated,

the outputs of these measures

are not actually stored in our dataset.

Only the formula is stored in our data model,

and this formula is constantly being evaluated

and reevaluated by Power BI

as you and your users are interacting with the report.

So, when you think about it,

measures are just another sort of DAX query,

and they're evaluated the same way.

The report uses DAX to send a request down to the dataset,

calculate what it needs,

and then renders that result in your report.

Many people get confused by this

because these two languages serve two similar

but distinct purposes in the world of Power BI.

Remember that Power Query M is used to extract,

transform and load data from your data sources

into your report's data set,

whereas DAX is used to communicate

between the report and the data set

through measures and queries.

It's a two-stage process, more or less,

where Power Query M

processes the first stage of loading the data

and then hands it off to DAX to analyze the data.

Okay, let's try to synthesize

everything we've just talked about

using a real-world example.

One of the first major projects that I developed

when I showed up as a developer at Davidson College

was to create an interactive Power BI report

that would allow each of our departments at the college

to explore their yearly budgets and expenses.

The budget managers needed to see

the amount that they were budgeted

to spend on software licenses, for example,

compared to what they are actually

spending on those licenses,

or employee salaries and benefits,

or consulting, or travel expenses.

Plus, some departments earn revenue

and they wanna see

not only how their budgeted revenue

compares to their actual revenue,

but they also want to see

how profitable their department is

when comparing their actual revenue

to their actual expenses.

Now, to give you some context,

our finance system at the college

sat on top of a massive Oracle database

and consisted of several different modules

such as an accounts payable system

for sending payments to our vendors,

an accounts receivable system

for processing tuition payments from students

as well as revenue from other sources,

a payroll system for calculating

our employees' earnings and deductions

so we can issue paychecks to them,

and then the general ledger system

that organizes all of our finance transactions

into a logical system of funds, cost centers and accounts,

and allows them to create budget scenarios for those.

Furthermore, our finance system

holds 30 years' worth of data,

but I really only needed to include

five years' worth of data in this report.

So in this example,

the data source I was using was a huge finance database,

but the data set that I needed for my Power BI solution

was just a small subset of that data source.

To begin, I used Power Query to connect to the database

where I selected only the tables

from the general ledger system

because those were the only ones I needed.

Then I filtered those tables down

to only the last five years' worth of data

and got rid of any columns I did not need.

I closed Power Query at that point and loaded the model,

which produced my data set.

The data set consisted of several tables,

the most central of which was our transaction history table.

At this point in Power BI Desktop, I made sure that

the relationships between these tables were correct

and I wrote a whole bunch of measures in DAX

to create the calculations I needed,

such as summing the transaction amounts,

then writing more advanced calculations

to get things like cumulative totals,

year over year amounts and so forth.

Once all my measures were defined,

I began visualizing that data in Power BI Desktop,

producing an interactive report

that allowed our department managers

to see their budget data in a whole new way

that they would've never seen before.

Power Query M handled the back end of the process

where we connected to the data source

and transformed the data

so it could be loaded into our data set,

and DAX handled the calculations needed

to visually represent the data in our report.

This whole course is built around a similar process.

We'll use Power Query to connect to data

and learn various ways you can transform it.

Then we'll load it into our data models,

write some measures with DAX,

and practice different methods of visualizing the results,

and we'll repeat this pattern several times

with different use cases.

So it's important to grasp

what exactly is going on behind the scenes.

Like I said at the start of this lesson,

I'll reinforce these points that I've made here

throughout the course,

so don't worry right now if it seems a little opaque.

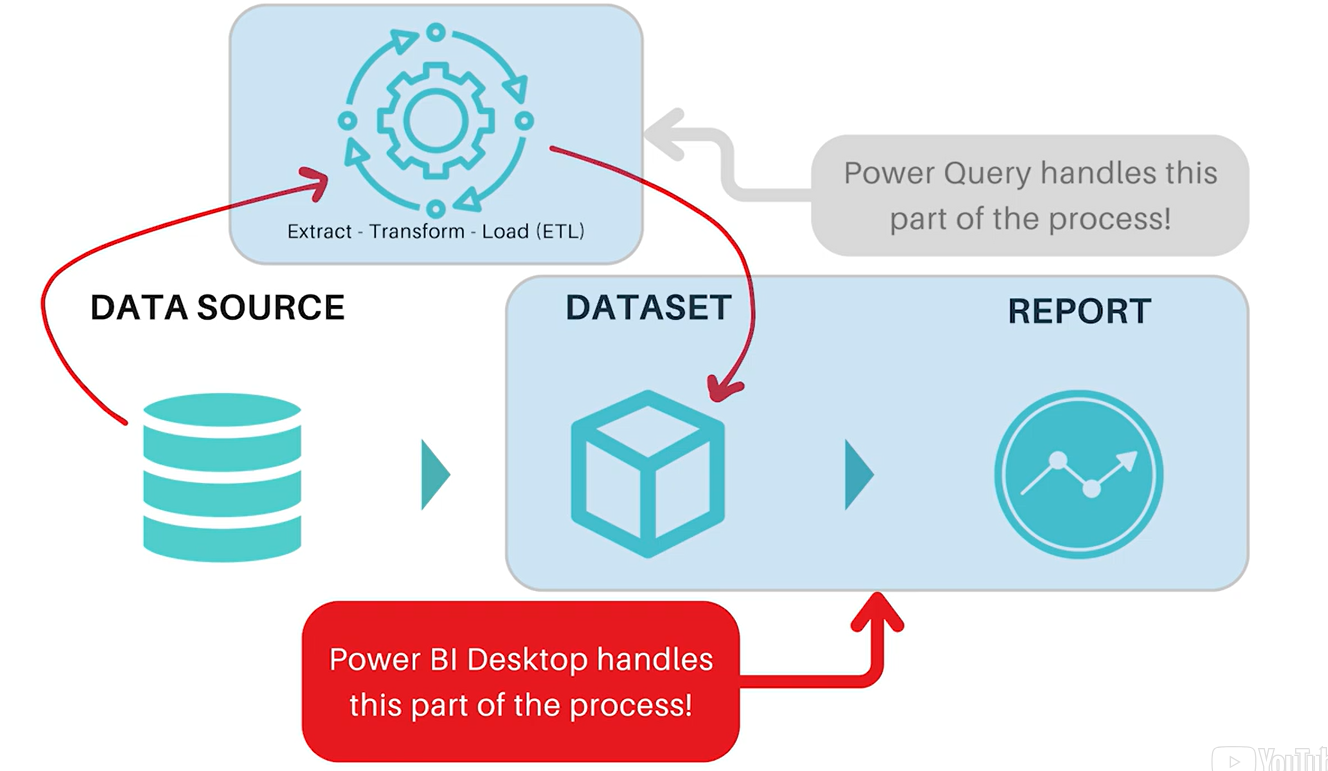
It will make more sense

as we continue our learning together.

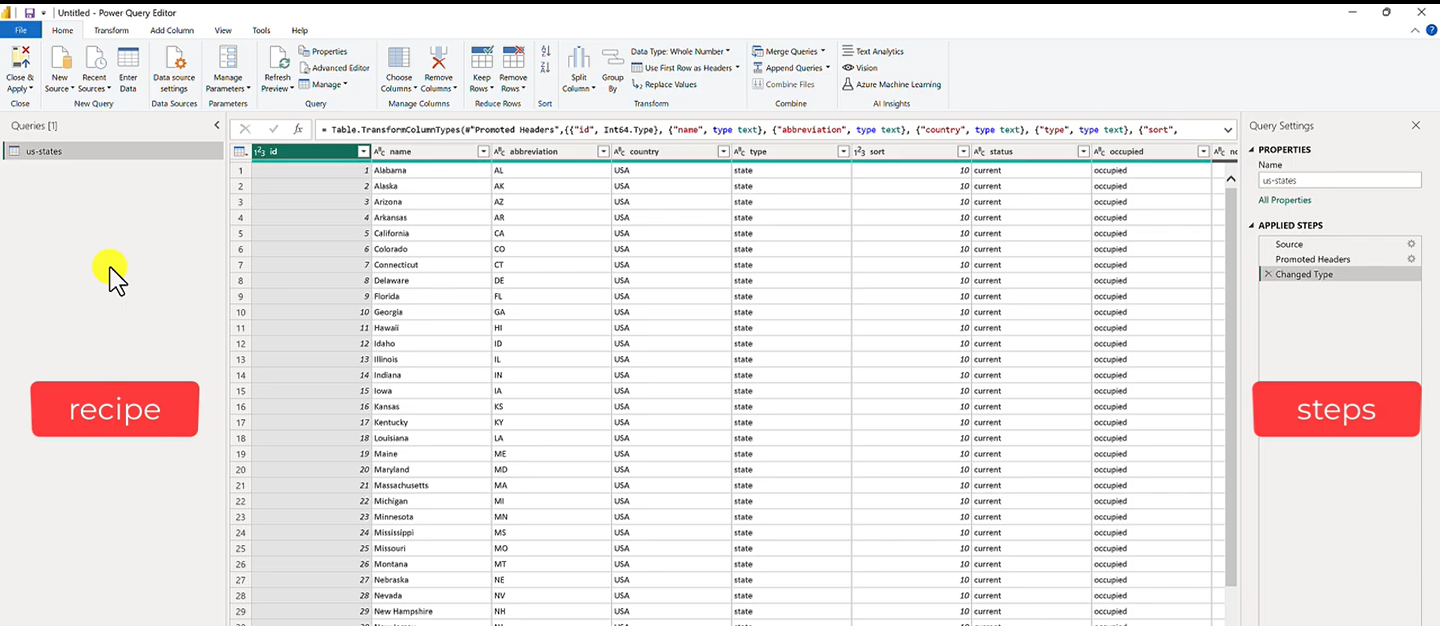
### M IS USED TO

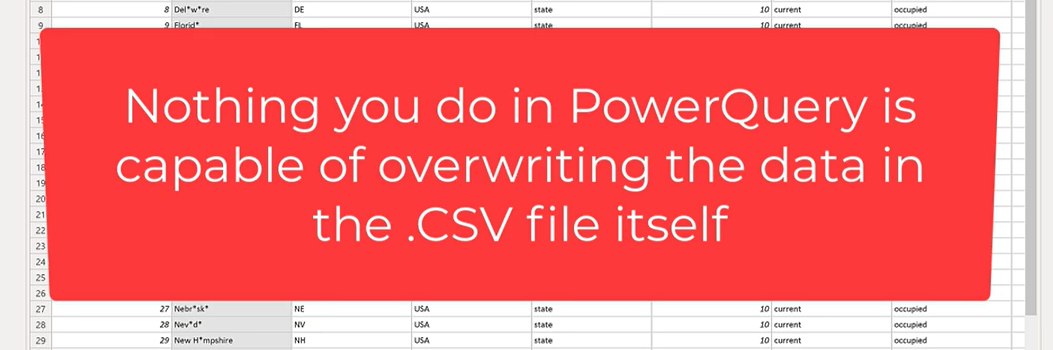
Remember that Power Query M is used to extract, transform and load data from your data sources into your report's data set, whereas DAX is used to communicate between the report and the data set through measures and queries.

**It's a two-stage process, more or less,** where Power Query M processes the first stage of loading the data and then hands it off to DAX to analyze the data.

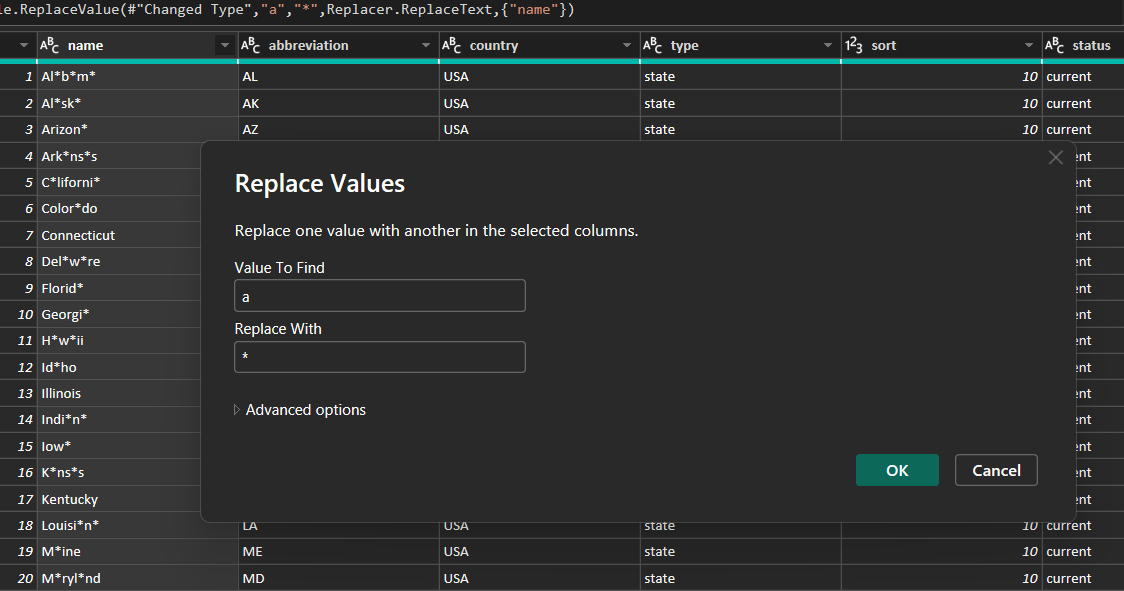


### POWER QUERY





Vai mudar apenas no POWER BI, O CSV CONTINUA IGUAL



<https://learn.microsoft.com/en-us/credentials/certifications/data-analyst-associate/?practice-assessment-type=certification#certification-exams>

### **✅ Próximos Passos**

1. **Estude o conteúdo oficial**: Acesse o curso PL-300T00-A no [Microsoft Learn](https://learn.microsoft.com/pt-br/training/courses/pl-300t00).
2. **Utilize o guia de estudo**: Consulte o [guia de estudo da Microsoft](https://learn.microsoft.com/pt-br/credentials/certifications/resources/study-guides/pl-300).
3. **Pratique com simulados**: Realize simulados para testar seus conhecimentos e identificar áreas de melhoria.
4. **Agende o exame**: Após se sentir preparado, agende o exame PL-300 através do [site oficial da Microsoft](https://learn.microsoft.com/pt-br/credentials/certifications/data-analyst-associate/).

### **🔄 Unpivot (Despivotar)**

**Unpivotar** significa transformar colunas em linhas. Esse processo é útil quando você possui dados distribuídos em várias colunas que representam categorias ou atributos, e deseja reorganizá-los em um formato mais adequado para análise.

#### **Exemplo:**

**Tabela Original:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Produto** | **Jan** | **Fev** | **Mar** |
| A | 10 | 12 | 14 |
| B | 8 | 9 | 11 |

**Após Unpivot:**

|  |  |  |
| --- | --- | --- |
| **Produto** | **Mês** | **Valor** |
| A | Jan | 10 |
| A | Fev | 12 |
| A | Mar | 14 |
| B | Jan | 8 |
| B | Fev | 9 |
| B | Mar | 11 |

No Power Query, você pode realizar essa transformação selecionando as colunas "Jan", "Fev" e "Mar", clicando com o botão direito e escolhendo **"Despivotar Colunas"**. [Suporte Microsoft+14Microsoft Learn+14ornit.com.br+14](https://learn.microsoft.com/pt-pt/power-query/unpivot-column?utm_source=chatgpt.com)

### **🔁 Pivot (Dinamizar)**

**Pivotar** é o processo inverso: transforma linhas em colunas. É útil quando você deseja reorganizar dados categóricos em colunas para facilitar comparações ou análises específicas. [Suporte Microsoft+15YouTube+15forum.xperiun.com+15](https://www.youtube.com/watch?v=CV_QveJN6k8&utm_source=chatgpt.com)

#### **Exemplo:**

**Tabela Original:**

|  |  |  |
| --- | --- | --- |
| **Produto** | **Mês** | **Valor** |
| A | Jan | 10 |
| A | Fev | 12 |
| A | Mar | 14 |
| B | Jan | 8 |
| B | Fev | 9 |
| B | Mar | 11 |

**Após Pivot:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Produto** | **Jan** | **Fev** | **Mar** |
| A | 10 | 12 | 14 |
| B | 8 | 9 | 11 |

Para realizar essa transformação no Power Query, selecione a coluna "Mês" e, em seguida, escolha **"Dinamizar Colunas"** no menu de transformação. [Microsoft Learn](https://learn.microsoft.com/pt-br/power-query/pivot-columns?utm_source=chatgpt.com)

### **📌 Dicas Importantes (UNPIVOT – PIVOT)**

* **Unpivot** é ideal para transformar dados de formato largo para longo, facilitando análises temporais ou categóricas.
* **Pivot** é útil para criar tabelas resumo, onde categorias se tornam colunas.
* Essas operações são acessíveis no Power Query através do menu **Transformar**.

### DAX

Diferenças entre CALCULATED COLUMNS AND MEASURES

2015 Population =

CALCULATE(

SUM (TAB\_US\_States\_Pop[Population]),

'TAB\_US\_States\_Pop'[Year] = 2015

)

CALCULATE CAN APPLIES FILTERS TO DAX CALCULATIONS.

DAX PARAMETERS = ARGUMENTS = SYNONYMOUS

I use commas to separate the parameters

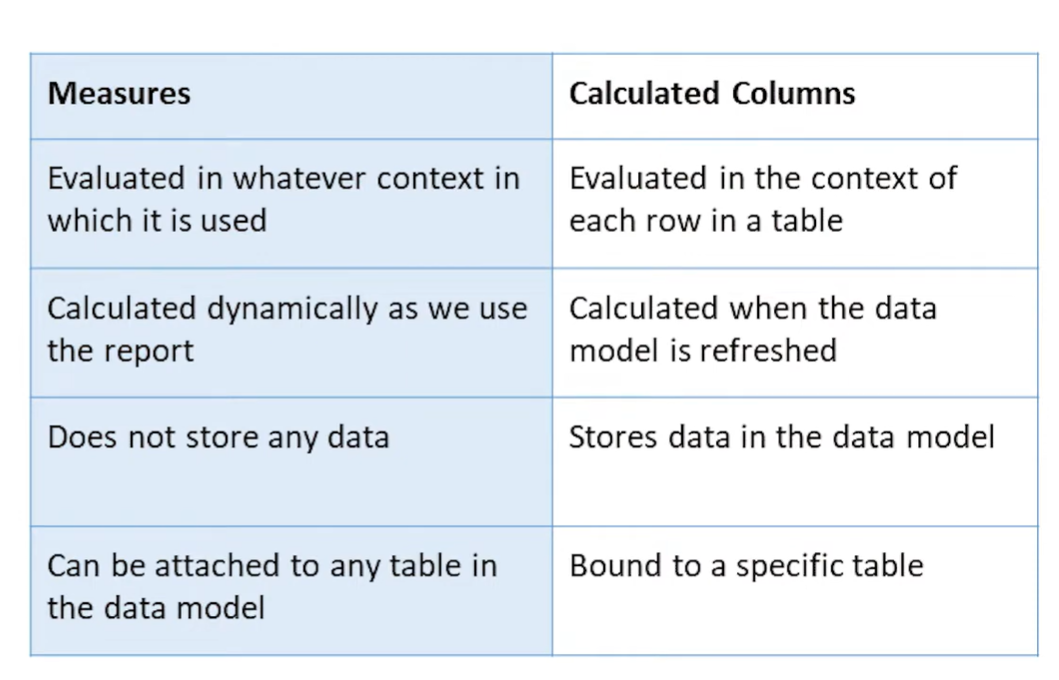
[CALCULATE FUNCTION](https://learn.microsoft.com/en-us/dax/calculate-function-dax)

## **Syntax**

DAX

CALCULATE(<expression>[, <filter1> [, <filter2> [, …]]])

Usamos DAX pra criar a CALCULATED COLUMN e a MEASURE, mas a **MEASURE não** foi **adicionada à tabela, a CALCULATED COLUMN foi.**



GRANULARITY =

CARDINALITY