

DAX

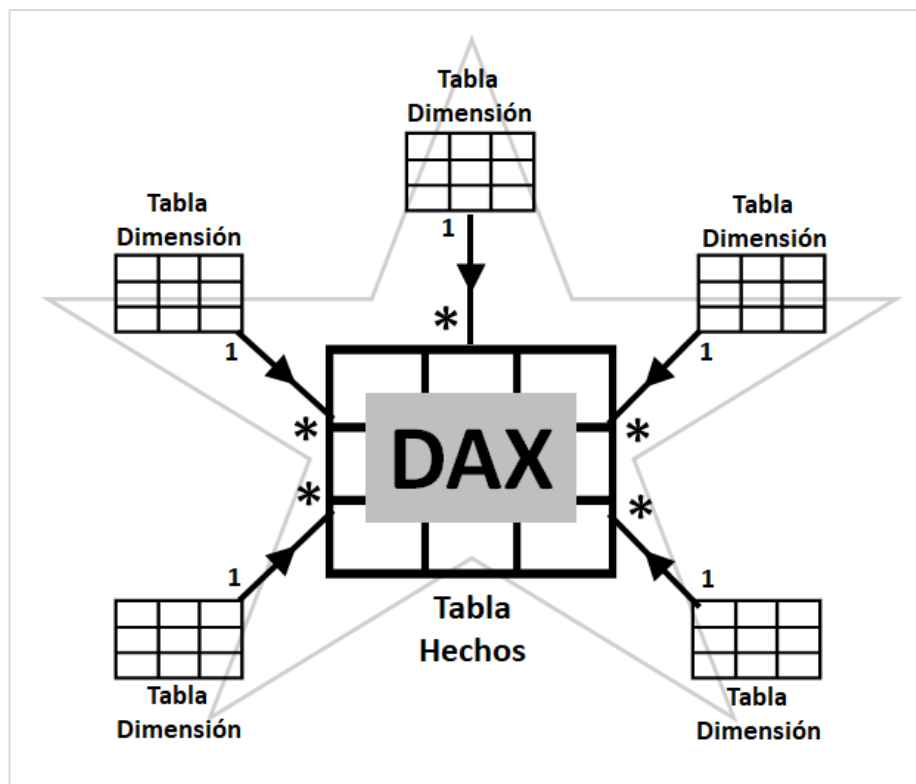
Índex de continguts

Introducció a DAX	2
Expressions DAX	4
Noms de taules i columnes.....	5
Funcions	6
Variables.....	7
Tipus de dades	9
Valors buits	10
Operadors	10
Funcions SI, SWITCH i COALESCE	11
Tipus de càlcul.....	14
Mesures	14
Columnes calculades.....	15
Taules calculades	16
Context de filtratge.....	17
Funcions CALCULATE i CALCULATETABLE.....	18
Context de fila.....	20
Funció RELATED	20
Transició de context	21
Funcions per agregar els valors d'una columna.....	24
Funcions per agregar els valors de diverses columnes	26
Funcions per comptar.....	26
Funció FILTER.....	27
Funció VALUES.....	28
Funció ALL.....	29
Funcions modificadores de CALCULATE: REMOVEFILTERS, ALLEXCEPT, ALLSELECTED.....	31
Funcions USERELATIONSHIP i CROSSFILTER	34

Introducció a DAX

DAX (Data Analysis Expressions) és un llenguatge per treballar amb models tabulars utilitzant fórmules i expressions. A Power BI l'utilitzem per fer els càlculs, un cop hem construït el nostre model de dades.

DAX és un llenguatge peculiar perquè està molt lligat al model de dades i el **resultat d'una fórmula es veurà afectat pels filtres que estiguin aplicats a les taules del model**. Això li atorga una gran potència, perquè podem emprar un sol càlcul en múltiples situacions.



Per exemple, podem crear una fórmula que calculi els ingressos sumant els valors d'una columna. Amb aquesta sola fórmula podem construir un informe que tingui: un gràfic de línies per mostrar l'evolució dels ingressos per mesos, un gràfic de barres amb els ingressos per categoria de producte, una taula amb els ingressos per comunitat autònoma, un selector per canviar d'any i un filtre per seleccionar els rangs d'edats. En aquest informe s'ha utilitzat la mateixa fórmula en tres llocs diferents i en cadascun el resultat serà diferent i quan es canviï d'any o es filtrin els rangs d'edats tornarà a canviar el resultat. La fórmula és la mateixa, però canvien els filtres aplicats a les taules del model.

Aquest potencial s'ha d'utilitzar amb responsabilitat ja que en crear una fórmula sempre hem de tenir en compte on la utilitzarem i com es veurà afectada pels filtres. Per aquesta raó dominar el llenguatge DAX

requereix temps i no n'hi ha prou amb conèixer les expressions amb què podem construir les fórmules, sinó que també cal conèixer diversos conceptes com són:

1. Context de filtratge
2. Context de fila
3. Transició de context

Però comencem introduint els elements del llenguatge.

Expressions DAX

Podem construir expressions DAX utilitzant **operadors** i **funcions**, i el resultat d'una expressió DAX pot ser un escalar (número, text, data) o una taula.

Aquests són alguns exemples d'expressions:

1

2 + 3

5

2

DATE(2020, 9, 1)

01-SEP-2020
0:00:00

3

Ventas

Importe
100,00
200,00

SUM(Ventas[Importe])

300,00

4

Ventas

Cantidad	Precio Unitario
1	100,00
2	200,00

SUMX (Ventas, [Cantidad] * [Precio Unitario])

500,00

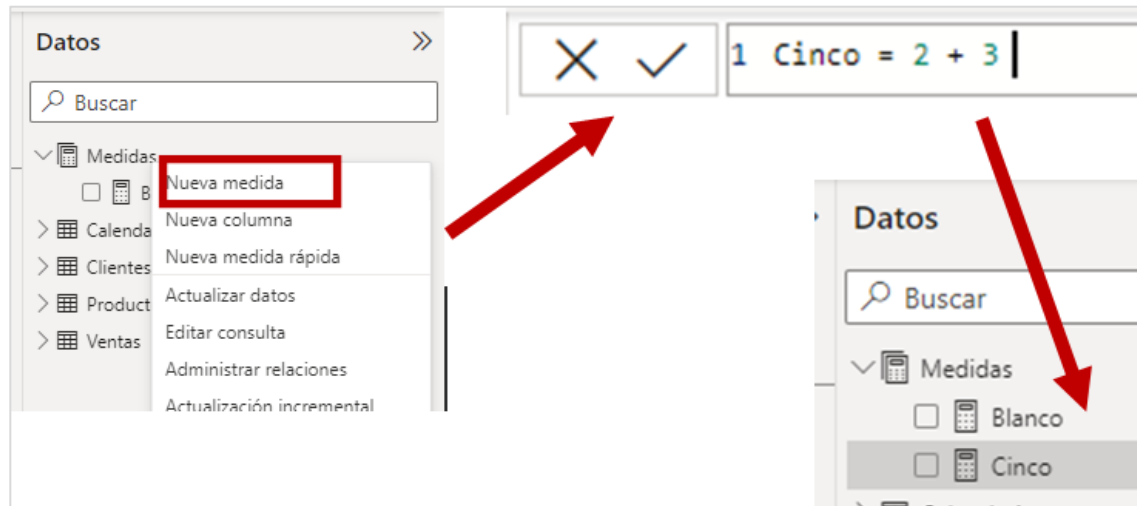
1. El primer exemple és simplement una suma de dos nombres enters que retorna un altre nombre enter.
2. En el segon exemple s'utilitza la **funció DATE** per obtenir una data. A la funció se li estan passant 3 paràmetres: l'any, el mes i el dia.
3. En el tercer exemple es fa servir una altra funció, **SUM**, que executa una operació sobre la taula Ventes del model, sumant els valors de la columna Import de l'esmentada taula. A la funció se li passa un sol paràmetre que és el nom de la columna que s'hi sumarà. El resultat és un nombre decimal.
4. En el quart exemple s'utilitza la funció **SUMX**, que també actua sobre la taula Ventes, però aquesta vegada sobre dues columnes. Aquesta funció recorre cada fila de la taula, multiplica per a cada fila el valor de la columna Quantitat pel valor de la columna Preu Unitari i suma els valors aconseguits. A la funció se li passen dos paràmetres: el primer és la taula i el segon paràmetre és l'expressió que s'executarà a cada fila de la taula. El resultat és un nombre decimal.

DAX **no diferencia entre majúscules i minúscules**, per la qual cosa la següent expressió és totalment vàlida:

```
sumx (
    ventas,
    ventas[cantidad] * ventas[precio unitario]
)
```

Perquè puguis provar aquestes expressions hem preparat un fitxer PBIX amb un model on pots crear **mesures** amb les expressions.

Per crear una mesura, accedeix al panell de dades que està a la dreta i has de fer clic amb el botó dret del ratolí a la taula Mesures i utilitza l'opció Nova mesura. T'agafarà una barra de fórmules amunt a l'esquerra on has d'indicar un nom per a la mesura i l'expressió DAX i donar-li a la tecla Entrer. Veuràs que la mesura es crea a la taula Mesures. Per veure el resultat pots arrossegar la mesura cap al dissenyador d'informe.



Noms de taules i columnes

En una expressió DAX, per referir-nos al nom d'una columna d'una taula fem servir claudàtors `[]` i podem escriure davant el nom de la taula. Per exemple **Vendes [Import]** és la columna **Import** de la taula **Vendes**.

Quan el nom d'una taula conté espais o caràcters especials, tanquem el nom de la taula en cometes simples. Per exemple, per referir-nos a una taula **'Vendes Internet'** i per referir-nos a una columna **'Vendes Internet'[Preu Unitari]**.

Funcions

En els exemples anteriors hem utilitzat algunes funcions DAX i la següent taula mostra les funcions més utilitzades agrupades per categories. Més endavant aprofundirem en algunes d'aquestes funcions.

Agregació	SUMA, SUMA, COMPTA, COUNTX, COUNTROWS, DISTINCTCOUNT, AVERAGE, MIN, MAX
Data i hora	CALENDARI, CALENDARI, DATA, ANY, MES, DIA, EDATE
Filtres	FILTRAR, CALCULAR, TOT, ALLEXCEPT Y ALLNOTBLANKROW, VALORS, DISTINCT
Informació	ISBLANK, ISEMPTY, ISNUMBER, HASONEVALUE
Lògiques	SI, NO, I, O, CANVIAR, COALESCE
Taules	ADDCOLUMNS, SELECTCOLUMNS, ROW, SUMMARIZE, TOPN
Matemàtiques	DIVIDIR
Textos	FORMAT, SUPERIOR, INFERIOR, VALOR
Relacions	RELACIONATS, RELACIONATS, USERRELATIONSHIP, CROSSFILTER
Intel·ligència de temps	DATEADD, DATESYTD, DATESMTD, DATESQTD, TOTALYTD, TOTALQTD, TOTALMTD

En anomenar una funció sempre s'utilitzen els parèntesis i entre els parèntesis s'indiquen els paràmetres. Hi ha algunes funcions sense paràmetres, però la majoria de les funcions tenen un o diversos paràmetres.

Un paràmetre d'una funció pot ser una crida a una altra funció, i en general pot ser qualsevol expressió DAX.

Per exemple, la següent expressió utilitza la funció **DIVIDE** per fer una divisió. Aquesta funció té dos paràmetres per indicar els dos operands de la divisió. El primer paràmetre és una expressió de resta que fa servir dues vegades la funció **SUM** i el segon paràmetre és una altra crida a la funció **SUM**.

```
DIVIDE (
    SUM (Ventas[Precio]) - SUM(Ventas[Coste]),
    SUM (Ventas[Precio])
)
```

Variables

En una expressió DAX es poden definir variables per millorar la llegibilitat i el rendiment. Per exemple, hem modificat l'expressió de divisió que hem vist anteriorment per utilitzar una variable:

```
VAR precio = SUM (Ventas[Precio])
RETURN DIVIDE (
    precio - SUM(Ventas[Coste]),
    precio
)
```

En aquest cas la variable millora el rendiment de l'expressió perquè el càlcul del preu es fa una sola vegada, mentre que en l'expressió original es feia dues vegades.

Per utilitzar variables s'empren les instruccions **VAR** i **RETURN**. Una expressió DAX amb variables sempre ha de retornar el resultat amb la instrucció **RETURN**.

Es poden definir més d'una variable, com es mostra en l'expressió següent:

```
VAR precio = SUM (Ventas[Precio])
VAR coste = SUM(Ventas[Coste])
RETURN DIVIDE (
    precio - coste,
    precio
)
```

La variable cost millora la llegibilitat, però no millora el rendiment perquè el càlcul el cost es fa una sola vegada.

En el següent exemple s'utilitza una expressió amb variables com a paràmetre de la funció **SUMX**.

```
SUMX (  
    Ventas,  
    VAR _cantidad = Ventas[Cantidad]  
    VAR _precioUnitario = Ventas[Precio Unitario]  
    RETURN _cantidad * _precioUnitario  
)
```

Per entendre aquest tercer exemple has de tenir en compte que un bloc que comenci amb VAR i acabi amb RETURN és una expressió DAX i que una expressió DAX es pot fer servir com a paràmetre d'una funció.

Els **noms de les variables no poden coincidir** amb els noms de les paraules reservades del llenguatge DAX ni **amb els noms de les taules** del model. Per aquesta raó, es recomana utilitzar un prefix com per exemple el guió baix "_" i així evitar que es pugui introduir una taula en el model el nom del qual coincideixi amb el d'una variable que existeixi.

És molt important assenyalar que **les variables en DAX no es poden modificar un cop definides**, per la qual cosa es comporten com a constants.

Tipus de dades

A la taula següent es resumeixen els tipus de dades disponibles a DAX. Aquests són els tipus que poden prendre les columnes d'una taula (excepte el tipus Variant) o poden ser el resultat d'una expressió DAX (quan el resultat no és una taula).

Nombre sencer	64 bits (8 bytes) Positiu o negatiu
Nombre decimal	64 bits (8 bytes) Punt flotant Fins a 17 dígits decimals Positiu o negatiu
Moneda	64 bits (8 bytes) Precisió fixa 4 dígits decimals Positiu o negatiu
Data/Hora	Dates vàlides des de l'1 de març de 1900 Punt flotant on la part entera són els dies des del 20 de desembre de 1899 i la part decimal és una fracció del dia.
Booleà	Retorna els valors veritable o fals que es poden representar amb les paraules TRUE i FALSE o també amb les funcions TRUE() i FALSE()
Text	Unicode, cada caràcter ocupa 16 bits (2 bytes) La comparació entre textos és insensible a majúscules i minúscules
Variant	Un tipus especial usat per expressions que poden donar com a resultat tipus de dades diferents depenent d'una condició.

No es pot fer servir com a tipus d'una columna d'una taula.

Valors buits

Qualsevol dels tipus de dades pot contenir un valor buit i DAX maneja els valors buits d'una manera una mica peculiar perquè tracta de convertir-los a un tipus de dada. Per exemple, si utilitzem un valor buit en una expressió lògica, el valor buit sempre es considerarà com a fals. Si fem un valor buit en una suma, el valor buit es comportarà com un 0. No obstant això, si fem servir un valor buit en una multiplicació, el resultat és un altre valor buit.

La funció **BLANK** retorna el valor buit i la podem fer servir quan en una expressió volem retornar un valor buit.

La funció **ISBLANK** permet comprovar si una columna o una expressió tenen un valor buit.

Per exemple, el resultat de la següent expressió és veritable.

```
ISBLANK ( BLANK() )
```

Cal ser molt curosos quan volem comprovar si un valor està buit i la millor manera de fer-ho és utilitzant la funció **ISBLANK**.

Per exemple, el resultat de les següents dues expressions és **veritable** a causa de la conversió automàtica de tipus que fa DAX d'un valor buit:

```
BLANK() = 0  
BLANK() = ""
```

Una alternativa a **ISBLANK** és utilitzar l'operador de **comparació estricta** **==** (dos signes d'igual) que no fa la conversió automàtica. Per exemple, el resultat de les següents expressions és **fals**.

```
BLANK() == 0  
BLANK() == ""
```

Operadors

La següent taula mostra els operadors de DAX.

Aritmètics	<code>+</code> <code>-</code> <code>*</code> <code>/</code> <code>^</code>
Comparació	<code>=</code> <code>==</code> <code>></code> <code><</code> <code>>=</code> <code><=</code> <code><></code>
Lògics	<code>&&</code> <code> </code> <code>DINS</code>
Text concatenat	<code>&</code>

A continuació, comentem alguns d'aquests operadors.

L'operador de divisió / genera un error quan s'intenta dividir per 0. Es recomana doncs utilitzar la funció `DIVIDE` que no genera un error i a més permet indicar un valor alternatiu. En el següent exemple, la primera expressió retorna `BLANK` i la segona expressió retorna `0` ja que se li està passant un tercer paràmetre a `DIVIDE` per indicar el valor alternatiu.

```
DIVIDE (1, 0)
```

```
DIVIDE (1, 0, 0)
```

L'operador de **comparació exacta** `==` ja l'esmentem abans i es diferencia de l'operador de comparació `=` quan comparem amb `BLANK`.

En els **operadors lògics** tenim `&&`; que retorna veritable només si els seus dos operands són veritables. Hi ha la funció **AND** que és equivalent, però quan tenim més de dos operands és més senzill utilitzar l'operador que la funció.

L'operador lògic `||` retorna veritable si algun dels seus operands és `CERT` i existeix la funció equivalent **OR**.

L'operador lògic **IN** retorna veritable si un valor existeix en una llista de valors i es pot utilitzar en lloc de l'operador `||` quan es vol comparar amb diversos valors. Per exemple, les expressions següents són equivalents:

```
Producto[Color] IN { "rojo", "negro", "verde" }
```

```
Producto[Color] = "rojo" || Producto[Color] = "negro" || Producto[Color] = "verde"
```

L'operador de **concatenació** `&` Concatena dos textos i existeix la funció equivalent **CONCATENATE**.

Funcions **SI**, **SWITCH** i **COALESCE**

Veurem a continuació tres funcions que tenen relació amb els operadors i amb els valors buits perquè són funcions que comproven una o diverses condicions per tornar un resultat escalar.

La funció **IF** comprova una condició i si és veritable retorna un valor i si és falsa retorna un altre valor. El segon valor és opcional i en aquest cas retornarà un valor buit quan la condició és falsa.

Per exemple, en les dues expressions següents, la primera retornarà el valor "Bé" si la suma dels imports és major a igual que 1000 i "Regular" si no. Mentre que la segona expressió retornarà un valor blanc quan la suma dels imports sigui menor que 1000.

```
IF ( SUM(Ventas[Importe]) >= 1000, "Bien", "Regular" )

IF ( SUM(Ventas[Importe]) >= 1000, "Bien" )
```

La funció **SWITCH** avalua una expressió i pot retornar diferents resultats que depenguin dels resultats de l'expressió. Per entendre-la, comentarem en detall la següent expressió que retorna un text d'acord amb la quantitat de files de la taula Vendes. Si no hi ha files, retorna el text "Sense vendes", si hi ha una fila retorna el text "1 venda", si hi ha dues files retorna el text "2 vendes" i si hi ha 3 o més files retorna el text "3 o més vendes".

```
SWITCH (
  COUNTROWS(Ventas),      ← Parámetro 1
  BLANK(), "Sin ventas",   ← Parámetros 2 y 3
  1, "1 venta",            ← Parámetros 4 y 5
  2, "2 ventas",           ← Parámetros 6 y 7
  "3 o más ventas"        ← Parámetro 8
)
```

En l'exemple es veu que se li han passat 8 paràmetres a la funció SWITCH. El primer paràmetre és una expressió que compta la quantitat de files d'una taula. El segon i el tercer paràmetre estan en la mateixa línia perquè l'expressió sigui més llegible. En el segon paràmetre es pregunta si el valor retornat per COUNTROWS és BLANK per retornar el text del tercer paràmetre. Així es continua amb els paràmetres quart i cinquè on es comprova si el resultat és 1 per retornar un altre text. I el mateix succeeix amb els paràmetres sisè i setè. L'últim paràmetre està sol en una línia i és el valor que es retornarà si no es compleix cap de les condicions prèvies.

Hi ha una altra manera d'utilitzar **SWITCH** on es poden comprovar diferents condicions i per a això en el primer paràmetre cal passar el valor TRUE i en els següents paràmetres les condicions i el que ha de retornar quan es compleixi la condició. Vegem un exemple on a més farem servir una variable.

```

VAR importeTotal = SUM(Ventas[Importe])
RETURN SWITCH (
    TRUE(),
    importeTotal < 100, "Malo",
    importeTotal < 500, "Regular",
    importeTotal < 1000, "Bien",
    "Muy Bien"
)

```

La funció **COALESCE** retorna el primer valor que no estigui buit de les expressions que li passin com a paràmetres. Cal passar-li 2 paràmetres com a mínim i se li poden passar més.

En el següent exemple li passen dos paràmetres a la funció COALESCE, el primer argument és una funció que retorna la quantitat de files d'una taula, però que si no troba files retorna BLANK. En el segon paràmetre es passa el text "Sense vendes", per tant, quan la funció del primer paràmetre retornada BLANK, es retornarà aquest text.

```

COALESCE ( COUNTROWS(Ventas), "Sin ventas" )

```

En un altre exemple li passen 3 paràmetres a COALESCE amb diferents columnes de data corresponents a la data de factura, la data d'enviament o la data de la comanda, perquè retorni el valor de la primera data que no estigui en blanc.

```

COALESCE ( Ventas[Fecha Factura], Ventas[Fecha Envío], Ventas[Fecha Pedido] )

```

Tipus de càlcul

Ja tenim prou elements per construir expressions DAX, ara anem a veure els tipus de càlculs on podem utilitzar aquestes expressions i que són els següents conceptes:

- Mesures
- Columnes calculades
- Taules calculades

Mesures

Les **mesures** són el tipus de càlcul que més vam utilitzar i que a Power BI Desktop es pot fer clic amb el botó dret en una taula i escollint l'opció Nova mesura o entrant a l'opció Modelat i fent servir el botó Nova mesura.



Quan creem una mesura hem de donar-li un nom i aquesta apareixerà com una columna d'una taula, però aquesta columna no emmagatzema dades perquè només conté l'expressió DAX i el valor que mostra la columna es calcula en el moment de mostrar-la. De fet, els noms de les mesures són globals al model per la qual cosa no poden existir dues mesures amb el mateix nom, encara que estiguin en taules diferents. A més, una mesura es pot moure d'una taula a una altra.

```
Ingresos = SUMX(Ventas, [Precio Unitario] * [Cantidad])
```

Quan estem escrivint l'expressió DAX d'una mesura cal tenir en compte que estem veient el model de dades en conjunt i que no estem situats en cap taula pel que no podem accedir directament a cap columna, sinó que sempre cal utilitzar alguna funció DAX que ens situï en alguna taula.

En crear una mesura podem fer ús d'altres mesures, com es veu en el següent exemple on es calcula el marge utilitzant una expressió que resta el cost de l'ingrés. Tant [Cost] com [Ingres] són mesures. **Quan ens referim a una mesura en una expressió DAX sempre utilitzem els claudàtors i mai indiquem el nom de la taula.**

```
Margen = [Ingresos] - [Costes]
```

Columnes calculades

Les **columnes calculades** es creen amb l'opció Nova columna, ja sigui fent clic dret en una taula o entrant a l'opció Modelat.



Aquestes columnes si que emmagatzemen dades perquè l'expressió DAX només s'executa en el moment de crear-la o quan s'actualitzen les dades.

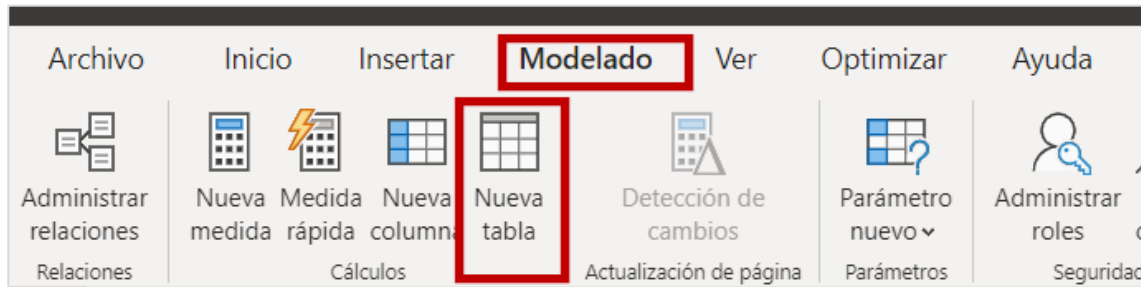
En crear una columna calculada si que estem situats a la taula on la creem pel que podem utilitzar directament els noms de les columnes de l'esmentada taula en l'expressió DAX i l'esmentada expressió **s'executarà per a cada fila de la taula**.

Per exemple, la columna calculada que es mostra a continuació ha estat creada a la taula Vendes i, per tant, té accés a les columnes Preu Unitari i Quantitat d'aquesta taula.

```
Ingresos = [Precio Unitario] * Ventas[Cantidad]
```

Taules calculades

Les **taules calculades** es creen a partir d'una expressió DAX que retorna una taula i igual que les columnes calculades, emmagatzemant dades en el model i l'expressió DAX només s'executa quan es crea la taula o quan s'actualitzen les dades.



En el següent exemple es mostra l'expressió DAX per crear una taula de dates.

```

1 Calendario =
2 SELECTCOLUMNS (
3     CALENDAR (
4         DATE ( YEAR ( MIN ( Ventas[Fecha] ) ), 1, 1 ),
5         DATE ( YEAR ( MAX ( Ventas[Fecha] ) ), 12, 31 )
6     ),
7     "Fecha", [Date],
8     "Año", YEAR ( [Date] ),
9     "Mes", FORMAT ( [Date], "mmm" ),
10    "MesNumero", MONTH ( [Date] ),
11    "Día", DAY ( [Date] ),
12    "Día semana", FORMAT ( [Date], "dddd" ),
13    "DiaSemanaNumero", WEEKDAY ( [Date], 2 )
14 )

```

En l'exemple s'ha utilitzat la funció CALENDAR per generar una taula amb un rang consecutiu de dates que comença el dia 1 de gener de l'any de la primera venda i acaba el dia 31 de desembre de l'any de l'última venda. Després s'ha utilitzat la funció SELECTCOLUMNS per crear una taula amb les diferents columnes de la dimensió Data, com l'any, el dia, el mes, etc.

Context de filtratge

Vam començar a explicar els conceptes fonamentals per entendre com s'executen les expressions DAX que utilitzem en les nostres mesures.

Ara veurem el context de filtratge que es defineix com el conjunt de filtres que existeixen en el model de dades en l' instant en què s'executa una expressió DAX. I aquests filtres afectaran el resultat de l'expressió.

Ho explicarem a través d'un exemple, utilitzant la mesura Ingressos d'un informe, com es mostra en la següent imatge.

```
Ingressos = SUM(Ventas[Importe])
```

L'informe té dues taules on s'utilitza la mesura Ingressos; una taule que mostra els ingressos per dia de la setmana i una altre per categoria de producte i també te una segmentació per filtrar pel color del producte. Per a aquest informe a més de la mesura hem utilitzat la columna Dia setmana de la taula Calendari, les columnes Categoria i Color de la taula Productes.

Día semana	Ingresos	Categoría	Ingresos	Color
lunes	100 €	Accesorio	250 €	<input type="checkbox"/> Negro
martes	1 100 €	Bicicleta	1.280 € 2	<input type="checkbox"/> Rojo
miércoles	400 €	Prenda	100 €	<input type="checkbox"/> Verde
jueves	150 €	Total	1.630 €	
viernes	700 €			
domingo	180 €			
Total	3 1.630 €			

El primer és assenyalar que a cada cel·la on s'utilitza la mesura Ingressos s'està executant l'expressió DAX i s'estan fent els càlculs en el moment de visualitzar les taules. En la imatge anterior, s'han assenyalat en color vermell tres cel·les per analitzar quin és el context de filtratge en cadascuna.

A la cel·la assenyalada amb el número 1 el context de filtratge és una taula amb la columna Dia setmana de la taula Calendari i que només conté el dia de la setmana dimarts.

A la cel·la assenyalada amb el 2 el context de filtratge és una taula amb la columna Categoria de la taula Productes i que només conté Bicicleta.

A la cel·la assenyalada amb el 3, correspon al total d'una taula, el context de filtratge està buit perquè en aquest cas no s'està aplicant cap filtre i, per tant, l'expressió DAX de la mesura està fent servir totes les files de la taula Vendas. El mateix succeeix amb el total de l'altra taula.

Vegem com canvia el context de filtratge quan apliquem filtres al color del producte.

Día semana	Ingresos	Categoría	Ingresos	Color
lunes	100 €	Accesorio	100 €	<input checked="" type="checkbox"/> Negro
martes 1	100 €	Bicicleta 2	280 €	<input checked="" type="checkbox"/> Rojo
viernes	100 €	Prenda	100 €	<input type="checkbox"/> Verde
domingo	180 €			
Total 3	480 €	Total	480 €	

A la cel·la assenyalada amb l'1 el context de filtratge ara contindrà dues taules: una amb la columna Dia de la setmana de la taula Calendari que només conté dimarts i una altra taula amb la columna Color de la taula Productes que conté dues files amb els valors Negre i Vermell.

A la cel·la senyalada amb 2 el context de filtre conté una taula amb les columnes Categoria i Color de la taula Productes i amb dues files on una conté els valors Bicicleta i Negre i l'altra els valors Bicicleta i Vermell a les seves respectives columnes.

A la cel·la assenyalada amb 3 el context de filtre conté una taula amb la columna Color de la taula Productes i amb dues files amb els valors Negre i Vermell. El mateix succeeix amb el total de l'altra taula.

Com hem vist a través d'aquest exemple, el **context de filtre conté taules amb les columnes que s'utilitzen per filtrar**.

Funcions CALCULATE i CALCULATETABLE

Les funcions CALCULATE i CALCULATETABLE són les úniques funcions de DAX que **poden canviar el context de filtratge** i gràcies a elles podem fer coses com calcular diferències contra el total o contra l'any anterior entre altres moltes coses.

L'única diferència entre ambdues funcions és que CALCULATE treballa amb valors escalars i CALCULATETABLE treballa amb taules pel que parlarem només de CALCULATE, però el mateix es pot aplicar a CALCULATETABLE.

A CALCULATE se li passa com a primer paràmetre una expressió i després se li pot passar una quantitat variable de paràmetres indicant els filtres que modificaran el context de filtratge per a aquesta expressió. Llavors CALCULATE comença per fer una còpia del context de filtratge vigent, aplica a aquesta còpia el context de filtres que s'han indicat a partir del segon paràmetre i executa l'expressió del primer paràmetre en aquesta còpia del context de filtratge.

Els **filtres** que li passen a **CALCULATE** poden ser de dos tipus: **una expressió booleana**, com per exemple, `Producte[Color] = "Rojo"`, o **una taula**. Si li passa una expressió booleana, aquesta es convertirà també a una taula. Recorda que el context de filtratge conté taules amb les columnes que s'utilitzen per filtrar.

Vegem un exemple d'una mesura que ha de calcular els ingressos amb un filtre per al color vermell i que anomenarem [Ingresos Rojo], mostrem en la imatge següent.

```
Ingresos Rojo = CALCULATE([Ingresos], Productos[Color] = "Rojo")
```

En la mesura [Ingresos Rojo] s'utilitza la funció **CALCULATE** i li passa com a primer paràmetre la mesura [Ingresos], una mesura que ja hem vist en exemples anteriors. El segon paràmetre de **CALCULATE** és una expressió booleana que s'utilitzarà com a filtre en el nou context de filtratge on s'executarà la mesura [Ingrés].

Quan emprem aquesta mesura en un informe succeirà el següent cada vegada que es vagi a realitzar el càlcul. La funció **CALCULATE** després de crear una còpia del context de filtratge, analitza el segon paràmetre i veu que hi ha un annex de filtre, on es demana filtrar les files de la taula Productes on la columna Color tingui el valor "Rojo". Així doncs, **CALCULATE** modifica el context de filtratge agregant una taula amb la columna Color de la taula Productes i que conté una fila amb el valor "Rojo".

I què succeeix si en el context de filtratge ja existís en aquesta mateixa taula? És a dir, que ja s'ha aplicat un filtre a la columna Color. Doncs que **CALCULATE** substitueix qualsevol filtre que existeixi sobre la columna Color en el context original per aquest nou filtre.

El resultat que hem explicat abans es pot apreciar a la taula que es mostra en la imatge següent que té tres columnes: Color, Ingresos i Ingresos Rojo i on veiem que la columna Ingresos Rojo sempre té el mateix valor i que correspon amb els Ingresos per als productes de color "Rojo".

Color	Ingresos	Ingresos Rojo
Negro	280 €	200 €
Rojos	200 €	200 €
Verde	1.150 €	200 €
Total	1.630 €	200 €

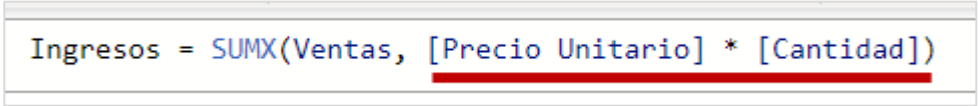
Aquest exemple potser no té una aplicació pràctica, però esperem que hagi servit com un primera aproximació al funcionament de la fuga. Més endavant veurem altres exemples més pràctics.

Context de fila

Anem a parlar ara d'un altre concepte important de DAX. El context de fila és quan **una expressió DAX s'executa en una fila d'una taula** i això pot succeir en dos casos: en una columna calculada i quan utilitzem un **iterador**.

Ja sabem el que és una columna calculada i veiem que l'expressió DAX s'executa per a cada fila de la taula.

Els **iteradors** són les funcions DAX que recorren cadascuna de les files d'una taula i que aquesta taula li passa com a primer paràmetre. Hem vist exemples que utilitzen la funció **SUMX**, com aquest.



```
Ingresos = SUMX(Ventas, [Precio Unitario] * [Cantidad])
```

La funció SUMX és un iterador on el primer paràmetre indica la taula que es recorrerà i el segon paràmetre és una expressió DAX que s'executarà a cada fila d'aquesta taula, per la qual cosa l'expressió marcada en vermell en la imatge anterior s'executarà en un **context de fila** i, per tant, podem fer referència directa a les columnes de la taula per tenir el valor que li correspon a la columna a la fila que estem recorrent.

Hi ha moltes altres funcions iteradores, per exemple, AVERAGEX, ADDCOLUMNS, SELECTCOLUMNS o FILTER.

Amb el context de fila succeeixen dues coses curioses.

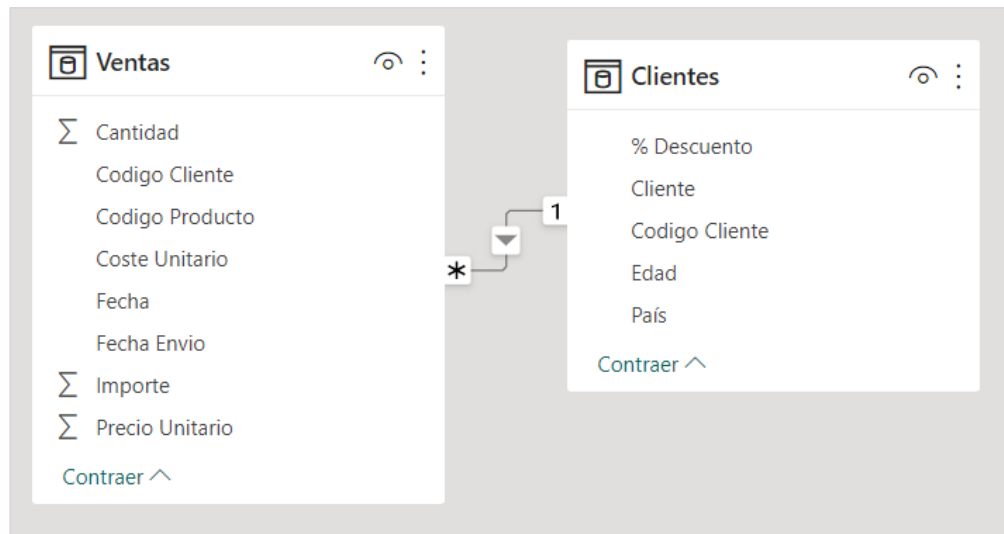
La primera és que quan l'expressió DAX s'està executant en una fila no pot accedir directament a les files d'altres taules, encara que estiguin relacionades entre si, o sigui el **context de fila no es propaga cap a altres taules**. Això veurem com resoldre-ho utilitzant la funció RELATED.

La segona curiositat és que, encara que l'expressió DAX s'estigui executant en una fila i pugui accedir directament als valors de les columnes d'aquesta fila, això no implica que la taula estigui filtrada per aquesta fila. **El context de fila no filtra la taula per la qual cosa el context de filtratge no s'afecta**. Això també veurem com moure'l utilitzant la funció CALCULATE per fer el que s'anomena **transició de context**.

Funció RELATED

La funció **RELATED** s'utilitza quan tenim un context de fila en una **taula de fets** i **volem utilitzar el valor d'una columna en una taula de dimensió**. A RELATED li passa un sol paràmetre que és el nom de la columna a la taula de dimensió i retorna un valor escalar. Ambdues taules han d'estar relacionades entre si.

Per exemple, suposem que tenim un model com el que es mostra en la imatge següent, on la dimensió Client té una columna % Descompte, i volem crear una mesura que calculi els ingressos aplicant el descompte.



Si intentem crear una mesura com la següent, rebrem un error.

```

1 Ingresos con descuento =
2 SUMX (
3     'Ventas',
4     'Ventas'[Precio Unitario] * 'Ventas'[Cantidad]
5     * (1 - 'Clientes'[% Descuento])
6 )

```

! No se puede determinar un valor único para la columna '% Descuento' en la tabla 'Clientes'. Esto puede su

En aquesta mesura estem iterant sobre la taula Vendes per calcular l'Ingrés i necessitem el valor del % de descompte que està a la taula Clients. Però quan tractem d'accedir directament a la columna rebem un error.

La solució està a modificar la mesura per utilitzar la funció RELATED, com es veu en la imatge següent.

```

1 Ingresos con descuento =
2 SUMX (
3     'Ventas',
4     'Ventas'[Precio Unitario] * 'Ventas'[Cantidad]
5     * (1 - RELATED('Clientes'[% Descuento]))
6 )

```

Transició de context

Que el context de fila no filtri la taula pot tenir conseqüències quan utilitzem algunes funcions DAX en una expressió que s'executi en un context de fila.

Per exemple, suposem que creem la mesura que es veu en la imatge següent. Aclarim que és una mesura que no té gaire sentit i que només estem utilitzant amb finalitats periòdiques.

```
Ingreso Mal = SUMX(Ventas, SUM(Ventas[Importe]))
```

Recordem que SUMX és un iterador, per la qual cosa recorrerà cada fila de la taula Vendes i en cadascuna executarà l'expressió SUM (Vendes[Importe]) que el que fa és sumar el valor del columna Import de la taula Vendes.

Podem pensar que cada vegada que SUM s'executa en una fila només va a veure aquesta fila i, per tant, només sumarà un valor. Però no és així, SUM veurà totes les files que li permeti el context de filtratge. Per exemple, si no hi ha cap filtre aplicat, se sumarà l'import de totes les files de la taula cada vegada que es recorri una fila, per la qual cosa el resultat final serà l'import total per la quantitat de files.

En la imatge següent mostrem una taula amb les columnes Categoría, Ingresos i Ingreso Mal on s'aprecia que aquesta mesura dona resultats erronis.

Categoría	Ingresos	Ingreso Mal
Accesorio	250 €	500 €
Bicicleta	1.280 €	5.120 €
Prenda	100 €	100 €
Total	1.630 €	11.410 €

En una situació com aquesta ens pot ajudar la funció **CALCULATE** perquè quan s'utilitza en un **context de fila**, agrega la fila com un filtre en el context de filtratge, el que es coneix amb el nom de **transició de context**.

Per veure-ho a la pràctica vam modificar la mesura que hem vist anteriorment [Ingreso Mal] per utilitzar CALCULATE, com es veu en la imatge següent.

```
Ingreso C = SUMX(Ventas, CALCULATE(SUM(Ventas[Importe])))
```

Hem creat la mesura [Ingreso C] que és similar a la mesura [Ingreso Mal] però hem utilitzat CALCULATE passant-li com a paràmetre l'expressió SUM (Vendes[Importe]). Fixeu-vos que li hem passat un sol paràmetre, pel que no li hem indicat cap filtre. Però **com que hi ha un context de fila, CALCULATE farà una transició de context**, cosa que significa que abans d'executar SUM a cada fila de la taula Vendes, modificarà la còpia del context de filtratge de manera que la taula Vendes estarà filtrada per aquesta fila. Per la qual cosa aquesta vegada SUM veurà només aquesta fila cada vegada que s'executa.

Per comprovar el que hem explicat anteriorment hem ampliat la taula que mostràvem anteriorment per incloure la mesura [Import C] i com veiem en la imatge els valors són iguals que els de la mesura [Ingressos].

Categoría	Ingresos	Ingreso Mal	Ingreso C
Accesorio	250 €	500 €	250 €
Bicicleta	1.280 €	5.120 €	1.280 €
Prenda	100 €	100 €	100 €
Total	1.630 €	11.410 €	1.630 €

Hi ha un detall addicional amb la transició de context i és que quan en una expressió DAX s'utilitza una mesura, DAX es comporta com si l'expressió de la mesura se li estigués passant com a paràmetre a CALCULATE, és a dir, com si s'anomenés de manera implícita a CALCULATE.

Això significa que sempre que s'usi una **mesura** en una expressió DAX que s'executi en un **context de fila** s'efectuarà una **transició de context**.

Per exemple, creem una nova mesura similar a [Ingrés C], però que en lloc de CALCULATE i SUM, utilitzi la mesura [Ingressos] com es veu en la imatge següent.

```
Ingreso M = SUMX(Ventas,[Ingresos])
```

En aquest cas també s'efectuarà la transició de context perquè s'està utilitzant una mesura.

A la següent taula podem veure com els resultats són correctes.

Categoría	Ingresos	Ingreso Mal	Ingreso C	Ingreso M
Accesorio	250 €	500 €	250 €	250 €
Bicicleta	1.280 €	5.120 €	1.280 €	1.280 €
Prenda	100 €	100 €	100 €	100 €
Total	1.630 €	11.410 €	1.630 €	1.630 €

Vegem un exemple més realista en què utilitzarem la funció **CALCULATETABLE**.

Suposem que volem calcular la mitjana de vegades que un client ha comprat pel que creem la següent mesura:

```
Promedio Ventas por Cliente =  
AVERAGEX ( Clientes , COUNTROWS ( Ventas ) )
```

Aquesta mesura té el problema que com que COUNTROWS s'executa en un context de fila, la taula Ventes no estarà filtrada pel client per la qual cosa el valor de la mitjana serà incorrecte.

Això es pot veure a la taula següent, on el total de la columna "Promedio Ventas por Cliente" és igual al total de vendes la qual cosa és incorrecte.

Cliente	Ventas	Promedio Ventas por Cliente
Adele	2	2,00
Ana	2	2,00
Carlos	2	2,00
Joao	1	1,00
Total	7	7,00

Per corregir-ho hem de modificar la mesura per utilitzar la funció CALCULATETABLE, així hi haurà una transició de context per la qual cosa la taula Ventes es filtrarà correctament.

```
Promedio Ventas por Cliente =
AVERAGEX ( Clientes , COUNTROWS ( CALCULATETABLE(Ventas) ) )
```

Cliente	Ventas	Promedio Ventas por Cliente
Adele	2	2,00
Ana	2	2,00
Carlos	2	2,00
Joao	1	1,00
Total	7	1,75

En aquest cas, en lloc de CALCULATETABLE també podríem haver utilitzat la funció **RELATEDTABLE** que és un àlies de CALCULATETABLE, però que només accepta un paràmetre per indicar la taula, per la qual cosa no es poden aplicar filtres addicionals i ens va molt bé en casos com aquests en què només necessitem la transició de context.

```
Promedio Ventas por Cliente =
AVERAGEX ( Clientes , COUNTROWS ( RELATEDTABLE(Ventas) ) )
```

Funcions per agregar els valors d'una columna

Després d'haver introduït els conceptes fonamentals de DAX podem continuar revisant les funcions més utilitzades.

Comencem amb un grup de funcions que ens permeten agregar els valors d'una columna i que són molt fàcils d'utilitzar perquè té un sol paràmetre que és el nom de la columna.

La funció **SUM** suma els valors d'una columna. En l'exemple que es mostra s'estan sumant els valors de la columna Import de la taula Vendes.

```
SUM(Ventas[Importe])
```

La funció **AVERAGE** calcula la mitjana dels valors d'una columna. En l'exemple següent s'està calculant la mitjana del preu unitari dels productes que s'hagin venut.

```
AVERAGE(Ventas[Precio Unitario])
```

La funció **MIN** retorna el mínim valor i es pot utilitzar de dues maneres.

Si se li passa un sol paràmetre ha de ser el nom d'una columna, i retornarà el valor mínim de l'esmentada columna. El resultat del primer exemple serà el valor mínim del preu unitari dels productes venuts.

```
MIN(Ventas[Precio Unitario])
```

Si se li passen dos paràmetres, han de ser expressions que retornin un valor escalat i retornarà el menor valor dels dos. L'expressió del segon exemple retornarà el valor mínim entre 30.000 i el resultat de la mesura [Ingressos].

```
MIN(30000,[Ingressos])
```

La funció **MAX** retorna el màxim valor i es pot utilitzar de dues maneres, igual que MIN.

```
MAX(Ventas[Precio Unitario])
```

```
MAX(30000,[Ingressos])
```

Funcions per agregar els valors de diverses columnes

Les funcions següents també fan **agregacions** com les anteriors, però es diferencien que són **iteradors** perquè van recorrent la taula fila a fila i van **executant una expressió que pot utilitzar diverses columnes**, i agreguen els resultats de l'expressió.

A aquestes funcions se li passen dos paràmetres, el primer és una expressió que retorni una taula, i el segon és l'expressió que s'executarà per cada fila de la taula.

L'expressió del segon paràmetre s'executa en un **context de fila**.

La funció **SUMX** suma el resultat d'aplicar l'expressió del segon paràmetre a cada fila de la taula del primer paràmetre.

```
SUMX ( Ventas, Ventas[Precio Unitario] * Ventas[Cantidad] )
```

La funció **AVERAGEX** calcula la **mitjana** de l'expressió del segon paràmetre sobre les files de la taula indicada en el primer paràmetre.

```
AVERAGEX ( Ventas, Ventas[Precio Unitario] * Ventas[Cantidad] )
```

La funció **MINX** retorna el **menor valor** que s'obtingui d'aplicar l'expressió del segon paràmetre a cada fila de la taula del primer paràmetre.

```
MINX ( Ventas, Ventas[Precio Unitario] * Ventas[Cantidad] )
```

La funció **MAXX** retorna el **valor més gran** que s'obtingui d'aplicar l'expressió del segon paràmetre a cada fila de la taula del primer paràmetre.

```
MAXX ( Ventas, Ventas[Precio Unitario] * Ventas[Cantidad] )
```

Funcions per comptar

Les següents funcions serveixen per comptar les files d'una taula o els valors d'una columna.

La funció **COUNTROWS** **compta les files d'una taula**. Si no troba cap fila, retorna BLANK. Li passa com a paràmetre una expressió que retorni una taula.

```
COUNTROWS (Ventas)
```

La funció **COUNT** compta les files on la columna indicada com a paràmetre **no tingui valors buits**. Si no troba cap fila que no estiguin buides, retornarà BLANK.

```
COUNT ( Ventas[Fecha Envio] )
```

L'expressió de l'exemple de dalt comptarà les vendes que ja s'hagin enviat, o sigui que tinguin data d'enviament.

La funció **DISTINCTCOUNT** compta els valors diferents de la columna que li passa com a paràmetre.

```
DISTINCTCOUNT ( Ventas[Codigo Cliente] )
```

L'expressió de l'exemple comptarà la quantitat de clients que hagin comprat. Si un client va fer diverses compres, ho comptarà una sola vegada.

Funció FILTER

Mentre les funcions anteriors retornen un valor escalar, la funció **FILTER** retorna una **taula**.

Aquesta funció és molt útil perquè ens permet filtrar una taula utilitzant una expressió. La taula es passa com a primer paràmetre i l'expressió com a segon paràmetre, per exemple:

```
FILTER(Clientes,Clientes[País] = "ES")
```

El resultat d'aquesta expressió és una taula que conté els clients d'Espanya.

Amb aquesta expressió podem crear una taula calculada, però no podem crear directament una mesura, perquè el resultat d'una mesura ha de ser un valor escalar. Però el que si podem fer és utilitzar l'expressió com un resultat intermediari per després obtenir el valor que retornarà la mesura. Per exemple, podem crear una mesura per comptar la quantitat de clients d'Espanya.

```
# Clientes ES = COUNTROWS( FILTER(Clientes,Clientes[País] = "ES") )
```

En aquesta mesura amb el nom [# Clientes de España] hem utilitzat FILTER com a paràmetre de COUNTROWS, per la qual cosa es compten les files de la taula amb els clients d'Espanya. Aquesta taula no es crea en el model, només existirà durant moment molt breu mentre s'executa la mesura.

Podem fer una altra versió d'aquesta mesura fent ús d'una variable per guardar temporalment la taula que retorna FILTER i el resultat és el mateix, cosa que potser més llegible.

```
# Clientes ES =
VAR _clientesES = FILTER(Clientes, Clientes[País] = "ES")
RETURN COUNTROWS( _clientesES )
```

Cal tenir en compte que **FILTER és un iterador** i que **l'expressió de filtratge s'executa en un context de fila**, i per aquesta raó podem referir-nos directament a les columnes de la taula.

Vegem alguns exemples més amb FILTER.

```
FILTER ( Clientes, [País] IN {"ES", "FR"} )
```

En aquesta expressió estem utilitzant l'operador IN per filtrar els clients d'Espanya o França.

```
FILTER (
    Clientes,
    [País] IN {"ES", "FR"} && [Edad] IN {25, 40}
)
```

En aquesta altra expressió utilitzem els operadors IN i &&

```
FILTER (Clientes, [Edad] > 30 )
```

I en aquest últim exemple s'estan filtrant els clients majors de 30 anys.

Funció VALUES

La funció **VALUES** retorna una taula amb els valors únics d'una columna.

```
VALUES (Productos[Categoría])
```

En aquest exemple estem creant una taula amb un llistat de categories de productes.

Com vam fer amb FILTER, podem combinar VALUES amb COUNTROWS, per exemple, per comptar la quantitat de categories.

```
COUNTROWS ( VALUES (Productos[Categoría] ) )
```

Funció ALL

Les funcions que hem vist abans són afectades pel context de filtratge, però, la funció **ALL** ignora el context de filtratge i retorna una taula amb totes les seves files.

Es pot fer servir de diverses maneres, per exemple.

```
ALL(Clientes)
```

Si li passa una taula com a paràmetre, el resultat serà la taula completa amb totes les seves files, incloent-hi les files duplicades.

```
ALL(Productos[Categoría])
```

Si se li passa un sol paràmetre que sigui una columna, el resultat serà una taula d'una sola columna amb tots els valors únics de la columna.

```
ALL(Productos[Categoría], Productos[Color])
```

Si li passen dos o més paràmetres amb noms de columnes, el resultat serà una taula amb la combinació de valors únics en aquestes columnes.

ALL s'utilitza en combinació amb altres funcions com COUNTROWS o FILTER quan volem calcular ràtios.

Per exemple, suposem que volem tenir una taula com la següent, amb la quantitat de productes que hi ha en cada categoria, el percentatge de productes en cada categoria contra el total de productes i el percentatge de productes en cada categoria contra els productes en la categoria Bicicleta.

Categoría	# Productos	% vs Todos	% vs Bicicleta
Accesorio	2	40 %	100 %
Bicicleta	2	40 %	100 %
Prenda	1	20 %	50 %
Total	5	100 %	250 %

Per aconseguir-ho, podem crear les següents mesures on s'ha utilitzat la funció ALL per eliminar el filtre de la categoria i poder comptar tots els productes per calcular la ràtio.

```
% vs Todos =
VAR _todosProductos = COUNTROWS(ALL(Productos))
RETURN DIVIDE([# Productos],_todosProductos)
```

```
% vs Bicicleta =
VAR _totalBicicletas = COUNTROWS (
    FILTER(ALL(Productos), Productos[Categoría] = "Bicicleta")
)
RETURN DIVIDE([# Productos], _totalBicicletas)
```

Funcions modificadores de CALCULATE: REMOVEFILTERS, ALLEXCEPT, ALLSELECTED

Ja hem vist que la funció CALCULATE modifica el context de filtratge aplicant els filtres que s'indiquin en els seus paràmetres.

Anem a veure tres funcions que se li poden passar com a paràmetre per modificar el comportament dels filtres. Aquestes funcions es poden utilitzar per calcular ràtios d'una manera similar a l'exemple que mostrem amb ALL.

La primera funció que vam veure és **REMOVEFILTERS** que elimina els filtres de taules o columnes.

```
CALCULATE( [Ingresos], REMOVEFILTERS(Productos) )
```

```
CALCULATE( [Ingresos], REMOVEFILTERS(Ventas) )
```

```
CALCULATE( [Ingresos], REMOVEFILTERS(Productos[Categoría] ) )
```

En els exemples que es mostren a dalt, veiem que hi ha tres maneres d'utilitzar-lo.

En el primer cas elimina els filtres que s'hagin aplicat a la taula de dimensions Productes.

En el segon cas elimina els filtres de la taula de fets Vendes i de totes les taules de dimensions relacionades amb ella.

I en el tercer cas elimina els filtres de la columna Categoría de la taula Productes.

Un detall curiós és que REMOVEFILTERS és un àlies de la funció ALL, per la qual cosa en els exemples podríem substituir REMOVEFILTERS per ALL, però REMOVEFILTERS només es pot utilitzar amb CALCULATE. El que succeeix és que ALL es comporta de manera diferent quan s'utilitza amb CALCULATE i no retorna cap taula, sinó que elimina els filtres. La recomanació és que amb CALCULATE sempre s'utilitzi REMOVEFILTERS perquè sigui més llegible l'expressió.

La segona funció que veurem és **ALLEXCEPT** que elimina els filtres de totes les columnes d'una taula, excepte la de les columnes que es passin com a paràmetres.

Presta atenció als exemples següents que s'assemblen molt, però són diferents.

```
CALCULATE( [Ingresos], ALLEXCEPT ( Productos, Productos[Categoría] ) )
```

```
CALCULATE( [Ingresos], ALLEXCEPT ( Ventas, Productos[Categoría] ) )
```

```
CALCULATE( [Ingresos], ALLEXCEPT ( Ventas, Productos ) )
```

El primer paràmetre d' ALLEXCEPT és una taula a la qual se li eliminaran els filtres. Els següents paràmetres són les columnes relacionades amb aquesta taula que mantindran els seus filtres.

En el primer exemple, s'estan traient tots els filtres de la taula de dimensió Productes, excepte a la columna Categoría.

En el segon exemple la diferència està que la taula que es passa en el primer paràmetre és la taula de fets Vendes, per la qual cosa s'eliminen els filtres de l'esmentada taula i de totes les taules de dimensions relacionades amb ella. Mentre que la columna Categoría de la taula Productes manté els seus filtres.

I en el tercer exemple el primer paràmetre és de nou la taula Vendes, però el segon paràmetre és una taula de dimensions relacionada amb ella. Per la qual cosa s'eliminaran tots filtres de Vendes i de les seves dimensions excepte els filtres de la taula Productes.

L'última funció que veurem és **ALLSELECTED** que elimina els filtres de la visualització on s'utilitza, però manté els filtres aplicats fora de la visualització.

Per exemple, en l'informe que es mostra on hi ha una taula i dues segmentacions, els filtres de la visualització serien l'any, el mes i el dia del mes, i els filtres externs serien les categories i els dies de la setmana.

Año	Mes	Día	Ingresos	Categoría	Día semana
2021	enero	6	400 €	<input checked="" type="checkbox"/> Accesorio	<input type="checkbox"/> lunes
2021	enero	7	150 €	<input checked="" type="checkbox"/> Bicicleta	<input checked="" type="checkbox"/> martes
2021	enero	12	100 €	<input type="checkbox"/> Prenda	<input checked="" type="checkbox"/> miércoles
Total			650 €		<input checked="" type="checkbox"/> jueves
					<input type="checkbox"/> viernes
					<input type="checkbox"/> sábado
					<input type="checkbox"/> domingo

Vegem com les següents expressions funcionarien en l'informe mostrat.

```
CALCULATE( [Ingresos], ALLSELECTED ( ) )
```



```
CALCULATE( [Ingresos], ALLSELECTED (Calendario) )
```

```
CALCULATE( [Ingresos], ALLSELECTED (Calendario[Mes], Calendario[Día]) )
```

En el primer cas s'utilitza ALLSELECTED sense paràmetres, per la qual cosa eliminarà els filtres d'any, mes i dia del mes i mantindrà els filtres de categoria i dia de la setmana.

En el segon cas li passa a ALLSELECTED la taula Calendari, per la qual cosa s'eliminaran els filtres de la visualització que afectin la taula Calendari, però es mantindran els filtres externs, encara que també siguin columnes de la taula Calendari. Per la qual cosa el comportament serà el mateix que en el primer cas.

I en el tercer cas li passen a ALLSELECTED les columnes Mes i Dia de la taula Calendari pel que s'eliminaran ambdós filtres, però es mantindrà el filtre de l'Any, també es mantindran els filtres de categoria i de dia de la setmana.

Per acabar amb aquesta secció et deixem un exemple amb tres mesures que utilitzen les tres funcions que hem explicat. Et convidem que creïs les mesures i l'informe en el model que t'adjuntem i que analitzis per què les mesures donen aquests resultats.

Año	Mes	Día	Ingresos	Ingresos vs Total	Ingresos vs Año Mes Total	Ingresos vs Seleccionado	Categoría	Día semana
2021	enero	6	400 €	25 %	43 %	36 %	<input checked="" type="checkbox"/> Accesorio	<input checked="" type="checkbox"/> lunes
2021	enero	12	100 €	6 %	11 %	9 %	<input checked="" type="checkbox"/> Bicicleta	<input checked="" type="checkbox"/> martes
2021	febrero	15	600 €	37 %	86 %	55 %	<input type="checkbox"/> Prenda	<input checked="" type="checkbox"/> miércoles
Total			1.100 €	67 %	67 %	100 %		<input type="checkbox"/> jueves
								<input type="checkbox"/> viernes
								<input type="checkbox"/> sábado
								<input type="checkbox"/> domingo

Ingresos vs Total =

```
VAR _ingresoTotal = CALCULATE([Ingresos], REMOVEFILTERS())
RETURN DIVIDE([Ingresos], _ingresoTotal)
```

Ingresos vs Año Mes Total =

```
VAR _ingresoTotal = CALCULATE([Ingresos], ALLEXCEPT(Ventas, Calendario[Año], Calendario[Mes]))
RETURN DIVIDE([Ingresos], _ingresoTotal)
```

Ingresos vs Seleccionado =

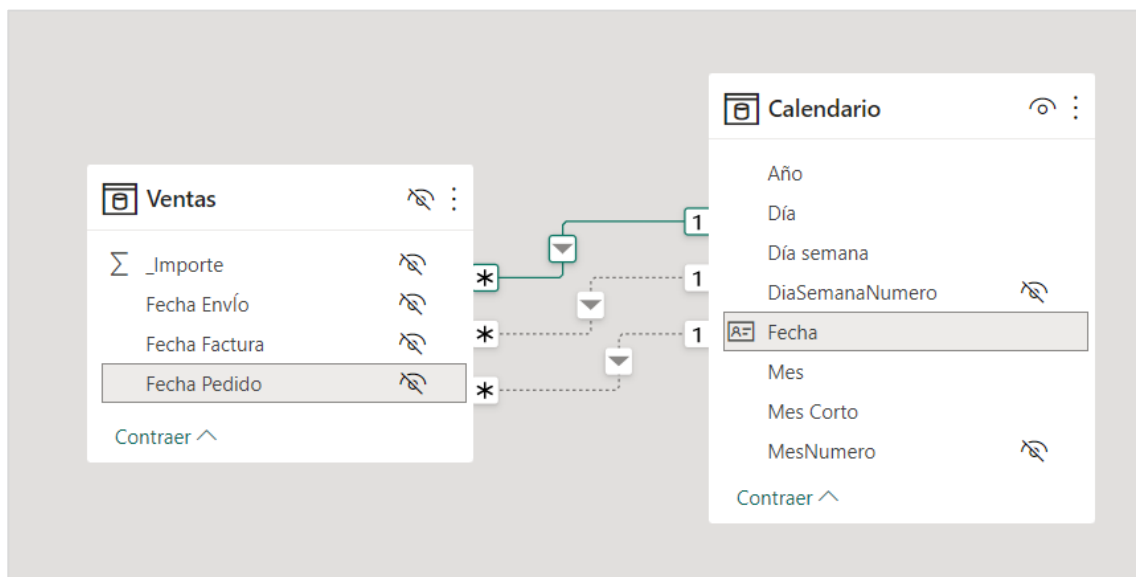
```
VAR _ingresoTotal = CALCULATE([Ingresos], ALLSELECTED())
RETURN DIVIDE([Ingresos], _ingresoTotal)
```

Funcions USERELATIONSHIP i CROSSFILTER

Les dues funcions que tractarem a continuació modifiquen temporalment les relacions del model.

USERELATIONSHIP activa una relació. Si hi ha més relacions entre les mateixes taules, es desactiven les altres. Aquesta funció és útil quan tenim una dimensió que té més d'una relació amb una taula de fets. Aquest tipus de dimensió es coneix amb el nom dimensió amb rols.

Per exemple, en el següent model hi ha 3 relacions entre les taules Vendes i Calendari perquè la taula Vendes té data de comanda, data d'enviament i data de factura. La relació activa és amb la data de comanda.



Si amb aquest model creem un informe on es mostrin els ingressos per mesos, la taula Calendari filtrarà la taula Vendes utilitzant la columna Data Comanda, per la qual cosa estarem calculant els ingressos corresponents a les comandes fetes cada mes.

Però volem ampliar l'informe per mostrar també els ingressos corresponents als enviaments i a les factures, com es mostra a continuació.

Año	Ingresos Pedidos	Ingresos Envíos	Ingresos Facturas
2018	€ 37.844.930,76	€ 30.179.810,54	€ 24.131.765,50
enero	€ 3.011.626,09	€ 2.315.820,45	€ 1.825.029,17
febrero	€ 3.338.730,14	€ 2.551.973,52	€ 1.924.510,10
marzo	€ 3.857.192,67	€ 3.065.889,09	€ 2.451.214,39
abril	€ 2.947.731,31	€ 2.346.705,23	€ 1.925.369,31
mayo	€ 3.393.868,91	€ 2.838.120,98	€ 2.244.327,37
junio	€ 3.113.491,33	€ 2.379.965,45	€ 1.911.350,78
julio	€ 3.278.427,08	€ 2.598.162,58	€ 2.146.146,03
agosto	€ 2.956.156,13	€ 2.395.766,57	€ 1.916.569,87
septiembre	€ 3.342.235,74	€ 2.737.907,73	€ 2.232.224,16
octubre	€ 3.060.518,16	€ 2.421.135,78	€ 1.965.616,54
noviembre	€ 2.957.003,42	€ 2.404.580,67	€ 1.794.890,36
diciembre	€ 2.587.949,78	€ 2.123.782,49	€ 1.794.517,42

Per calcular els ingressos corresponents als enviaments necessitaríem activar la relació entre Calendari [Data] i Vendes [Data Enviamet] i aquí és on podem utilitzar la unció USERELATIONSHIP, de la manera següent.

```

Ingresos Envíos =
CALCULATE (
    [Importe],
    USERELATIONSHIP(Calendario[Fecha], Ventas[Fecha Envío]),
    Ventas[Fecha Envío] <> BLANK()
)

```

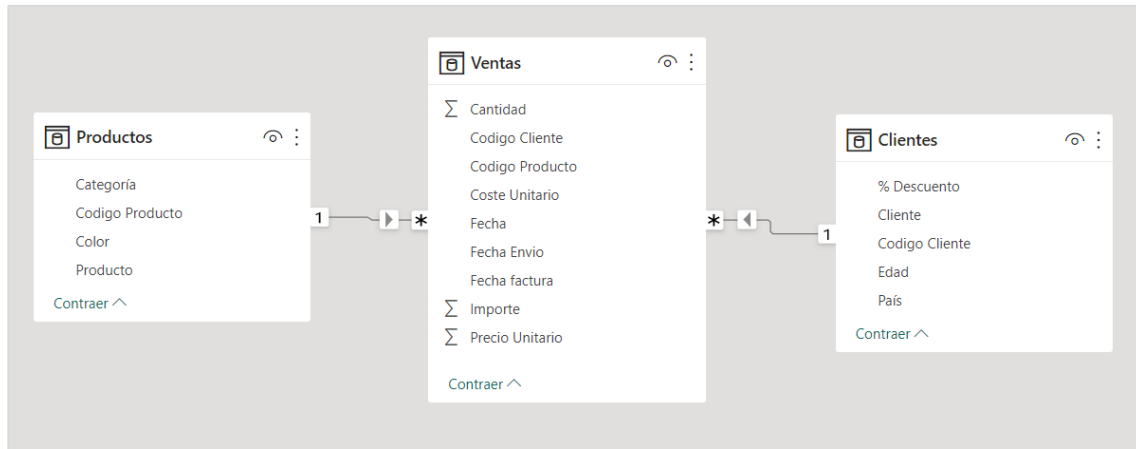
A USERELATIONSHIP li passen com a paràmetres les columnes de la relació que volem activar. I al seu torn passem USERELATIONSHIP com a segon paràmetre de CALCULATE. D'aquesta manera estem activant una relació, però només durant el breu temps en què es fa el càlcul.

Veiem, a més, que s'està passant un altre paràmetre a CALCULATE que és un filtre per garantir que no es tinguin en compte les files amb la data d'enviament buida.

El càlcul dels ingressos corresponents a les factures és molt similar i te'l deixem perquè ho facis pel teu compte.

CROSSFILTER permet canviar la direcció del filtratge d'una relació. Recorda que hem recomanat configurar una direcció única de filtratge des de la taula de dimensió cap a la taula de fets. Però hi ha ocasions en què necessitem temporalment que es filtri en ambdues direccions. En aquests casos podem utilitzar CRESSFILTER.

Per exemple, tenim un model amb les taules Producte, Vendes i Clients, com el que es mostra en la imatge, i volem mostrar en un informe l'edat mitjana dels clients per categoria de producte.



Podem crear una mesura molt senzilla que utilitzi la funció AVERAGE sobre la columna Edat.

```
Edad promedio clientes = AVERAGE (Clientes[Edad] )
```

Però no funcionarà bé, com es veu en la imatge següent on la mitjana d'edat és sempre el mateixa.

Categoría	Edad promedio clientes
Accesorio	36,50
Bicicleta	36,50
Prenda	36,50
Total	36,50

La raó d'aquest error és que el filtre aplicat a la taula Producte no es propaga cap a la taula Clients, sinó que es queda a la taula Vendes. Perquè es propagui cal canviar la direcció del filtratge entre Vendes i Clients perquè sigui en ambdues direccions. Si vols prova ara en el model a canviar la relació.

Però no volem evitar les relacions en ambdues direccions, per la qual cosa la solució és modificar la mesura utilitzant CROSSFILTER.

```
Edad promedio clientes =
CALCULATE (
    AVERAGE (Clientes[Edad] ),
    CROSSFILTER ( Clientes[Código Cliente], Ventas[Código Cliente], Both )
)
```

A CROSSFILTER li passem 3 paràmetres, els dos primers són les columnes de la relació i el tercer paràmetre és la direcció del filtratge. En aquest cas hem indicat ambdues direccions.

En fer aquest canvi, els resultats del nostre informe són correctes.

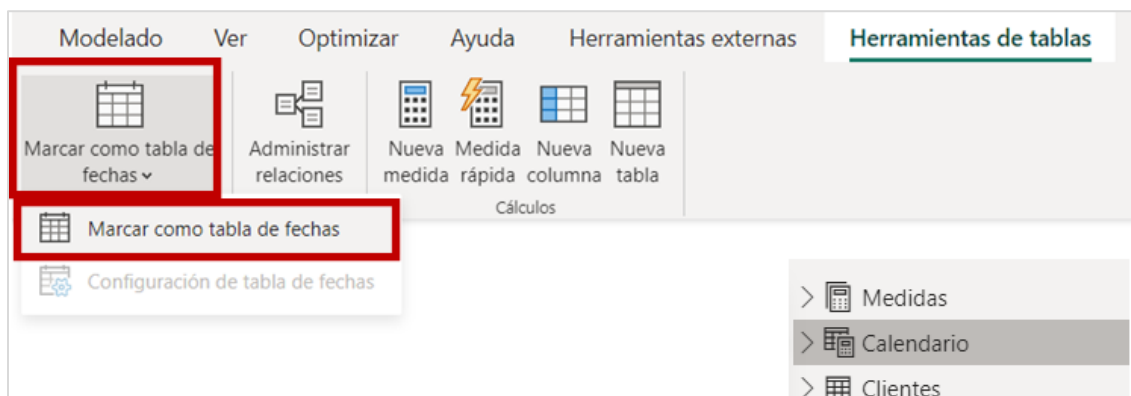
Categoría	Edad promedio clientes
Accesorio	35,00
Bicicleta	36,50
Prenda	25,00
Total	36,50

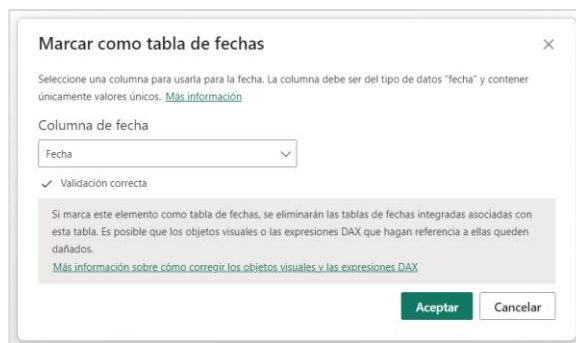
Funcions d'intel·ligència de temps

Hi ha un grup de funcions de DAX que treballen de conjunt amb CALCULATE per canviar la dimensió Data en el context de filtratge i, per tant, poder comparar els càlculs actuals amb dates passades o futures.

Aquestes funcions necessiten que en el model hi hagi una dimensió Data i a més cal identificar-lo en Power BI Desktop com una taula de dates.

Per a això cal seleccionar la taula de Dates, en el nostre cas és la taula Calendari i utilitzar el botó Marcar com a taula de dates, com es mostra en la imatge.





Veiem ara algunes de les funcions d'intel·ligència de temps més utilitzades.

SAMEPERIODLASTYEAR ens serveix per moure'ns un any enrere. Per exemple, podem fer servir la següent mesura per calcular els ingressos de l'any anterior.

Ingresos AA = **CALCULATE**([Ingresos],**SAMEPERIODLASTYEAR**(Calendario[Fecha]))

Com es veu, a la funció SAMEPERIODLASTYEAR se li passa un sol paràmetre que és la columna que conté les dates a la taula de dates. Al mateix temps, SAMEPERIODLASTYEAR es passa com a segon paràmetre de CALCULATE. El resultat és que es calcularan els ingressos amb un context de filtratge en què les dates es tenen un any abans.

En la imatge següent es mostra el resultat d'utilitzar aquesta mesura per veure els ingressos en un mes i comparar-lo amb els ingressos del mes anterior. Com que 2018 és el primer any, la mesura [Import AA] retorna valors buits, però el 2019 si retorna els valors del 2018.

Año	Ingresos	Ingresos AA
2018	€ 37.844.930,76	
enero	€ 3.011.626,09	
febrero	€ 3.338.730,14	
marzo	€ 3.857.192,67	
abril	€ 2.947.731,31	
mayo	€ 3.393.868,91	
junio	€ 3.113.491,33	
julio	€ 3.278.427,08	
agosto	€ 2.956.156,13	
septiembre	€ 3.342.235,74	
octubre	€ 3.060.518,16	
noviembre	€ 2.957.003,42	
diciembre	€ 2.587.949,78	
2019	€ 38.725.107,02	€ 37.844.930,76
enero	€ 3.417.723,11	€ 3.011.626,09
febrero	€ 3.430.735,86	€ 3.338.730,14
marzo	€ 3.471.379,61	€ 3.857.192,67
abril	€ 3.107.931,59	€ 2.947.731,31
mayo	€ 3.343.104,93	€ 3.393.868,91
junio	€ 3.482.362,41	€ 3.113.491,33
julio	€ 2.545.010,12	€ 3.278.427,08
agosto	€ 3.247.112,43	€ 2.956.156,13
septiembre	€ 3.315.489,60	€ 3.342.235,74
octubre	€ 3.122.213,43	€ 3.060.518,16
noviembre	€ 3.135.733,87	€ 2.957.003,42
diciembre	€ 3.106.310,06	€ 2.587.949,78

DATEADD ens permet canviar a diferents intervals de data (any, trimestre, mes, dia) en el passat o en el futur, per la qual cosa és més general que SAMEPERIODLASTYEAR.

Per exemple, podem modificar la mesura anterior per utilitzar DATEADD.

```
Ingresos AA = CALCULATE( [Ingresos],DATEADD(Calendario[Fecha], -1, YEAR )
```

En l'exemple es veu que a DATEADD se li passen tres paràmetres: la columna data de la taula de dates, el nombre d'intervals que ens volem desplaçar (negatiu cap enrere, positiu cap a davant) i el tipus d'interval (DAY, MONTH, QUARTER, YEAR).

TOTALYTD és per calcular l'acumulat anual. És molt senzilla d'utilitzar perquè només cal passar-li en el primer paràmetre l'expressió amb el càlcul que volem fer i en el segon paràmetre la columna data de la taula de dates. No cal fer servir CALCULATE.

```
Ingresos YTD = TOTALYTD( [Ingresos], Calendario[Fecha] )
```

En la imatge següent es mostra una taula on s'utilitza la mesura [Ingressos YTD] i on s'aprecia com es van acumulant els ingressos durant el 2018 i com el gener del 2019 comença de nou.

Año	Ingresos	Ingresos YTD
2018		
enero	€ 3.011.626,09	3.011.626,09 €
febrero	€ 3.338.730,14	6.350.356,23 €
marzo	€ 3.857.192,67	10.207.548,90 €
abril	€ 2.947.731,31	13.155.280,21 €
mayo	€ 3.393.868,91	16.549.149,12 €
junio	€ 3.113.491,33	19.662.640,45 €
julio	€ 3.278.427,08	22.941.067,53 €
agosto	€ 2.956.156,13	25.897.223,66 €
septiembre	€ 3.342.235,74	29.239.459,40 €
octubre	€ 3.060.518,16	32.299.977,56 €
noviembre	€ 2.957.003,42	35.256.980,98 €
diciembre	€ 2.587.949,78	37.844.930,76 €
2019		
enero	€ 3.417.723,11	3.417.723,11 €
febrero	€ 3.430.735,86	6.848.458,97 €
marzo	€ 3.471.379,61	10.319.838,58 €
abril	€ 3.107.931,59	13.427.770,17 €
mayo	€ 3.343.104,93	16.770.875,10 €
junio	€ 3.482.362,41	20.253.237,51 €
julio	€ 2.545.010,12	22.798.247,63 €
agosto	€ 3.247.112,43	26.045.360,06 €
septiembre	€ 3.315.489,60	29.360.849,66 €
octubre	€ 3.122.213,43	32.483.063,09 €
noviembre	€ 3.135.733,87	35.618.796,96 €
diciembre	€ 3.106.310,06	38.725.107,02 €

També podem acumular per trimestre o mesos utilitzant les funcions **TOTALQTD** i **TOTALMTD**.