

DAX

PARA TODOS

Volumen 1

Lenguaje de Análisis de
Datos en Power BI

PARA: RAMON ZAPATA



Didier Atehortúa Morales · Christian Tamayo Alfonso

DAX para todos

Expresiones de análisis de datos
y lenguaje de consulta para
modelos tabulares

©Didier Atehortua Morales, 2022

©Christian Tamayo Alfonso, 2022

ISBN: 978-958-53268-2-8

Sello editorial: DATA ICE S.A.S (978-958-53268)

Reservados todos los derechos. No se permite la reproducción total o parcial de esta obra, ni su incorporación a un sistema informático, ni su transmisión en cualquier forma o por cualquier medio (electrónico, mecánico, fotocopia, grabación u otros) sin autorización previa y por escrito de los titulares del copyright. La infracción de dichos derechos puede constituir un delito contra la propiedad intelectual.

DAX para todos
2022

El único y mejor método de aprender DAX en español

Primera Edición

Contenido Resumido

Capítulo 1: ¿Por qué aprender DAX?.....	13
Capítulo 2: Modelo de datos	21
Capítulo 3: Tipos de cálculos en DAX	34
Capítulo 4: Funciones	54
Capítulo 5: Manejo de variables	83
Capítulo 6: Principales funciones de tabla.....	93
Capítulo 7: Funciones y operadores lógicos.....	115
Capítulo 8: RELATED, RELATEDTABLE y LOOKUPVALUE	140
Capítulo 9: Evaluación de contextos	161
Capítulo 10: CALCULATE y funciones de filtro.....	182
Capítulo 11: CALCULATETABLE vs FILTER.....	201
Capítulo 12: Inteligencia de tiempo	209
Capítulo 13: Recetas.....	233

Contenido Específico

Contenido Resumido	5
Contenido Específico	6
Acerca de los autores.....	9
Dedicatorias	10
Introducción	11
Capítulo 1: ¿Por qué aprender DAX?	13
¿Por qué aparece DAX?	14
DAX requiere fundamentos teóricos.....	15
Resumen de tipos de datos y operadores	15
Operadores de comparación	17
Capítulo 2: Modelo de datos	21
Qué son los modelos Tabulares	21
¿Por qué la necesidad de un modelo tabular?	22
Comprensión del modelo de datos	23
Relaciones.....	24
Cardinalidad.....	25
Dirección de filtro cruzado	26
Recomendaciones	30
Capítulo 3: Tipos de cálculos en DAX	34
Tablas	36
Columnas.....	39
¿Cuándo crear columnas?	44
Medidas.....	44
Medidas Implícitas.....	45
Medidas explícitas	49
Capítulo 4: Funciones.....	54
Funciones de agregación.....	54
Clasificación de medidas	55
Medidas con formato	58
SUM	59
AVERAGE	61
COUNTROWS	62
MAX	63
MIN	64
DISTINCTCOUNT	65
Funciones de iteración.....	65
SUMX	66
AVERAGEGEX	72
MAXX	73
MINX	74
Funciones de texto	74
LEFT	74
RIGHT.....	76
CONCATENATE	76
UPPER	77
LOWER.....	77

LEN	78
SEARCH	78
SUBSTITUTE.....	81
Capítulo 5: Manejo de variables	83
Declarar variables en DAX	83
Comentar o documentar tu código.....	87
Formatear códigos DAX	87
Comprender que las variables se calculan una sola vez	89
Capítulo 6: Principales funciones de tabla.....	93
VALUES.....	93
ALL.....	94
Diferencia entre VALUES y ALL en uso de medidas	95
CROSSJOIN	97
SUMMARIZE	98
ADDCOLUMNS.....	100
Caso práctico ADDCOLUMNS y SUMMARIZE.....	101
UNION.....	103
DISTINCT.....	105
INTERSECT	107
EXCEPT	108
FILTER	112
Capítulo 7: Funciones y operadores lógicos.....	115
Funciones lógicas.....	115
IF	115
IF anidados	116
IF en medidas	118
SWITCH	119
SWITCH en medidas.....	122
AND	126
OR.....	130
Operadores lógicos.....	131
&& (AND)	131
(OR).....	132
Combinación , && y manejo de paréntesis.....	134
IN.....	135
IN con listas de Power Query	136
IN y NOT	137
Capítulo 8: RELATED, RELATEDTABLE y LOOKUPVALUE	140
RELATED	140
RELATED en medidas	144
Ejercicio práctico con RELATED.....	146
Ejercicios propuestos	149
RELATEDTABLE	149
RELATEDTABLE en medidas	151
Ejercicios propuestos	152
LOOKUPVALUE	153
Columna llave para relacionar tablas.....	157
Capítulo 9: Evaluación de contextos	161
Introducción a la evaluación de contextos.....	161
Comprendiendo los contextos de filtro	162
Comprendiendo los contextos de fila	171
Comprendiendo la transición de contexto	176
Capítulo 10: CALCULATE y funciones de filtro.....	182

Introducción a CALCULATE.....	182
ALL	187
ALLSELECTED	193
ALLEXCEPT	195
REMOVEFILTERS	197
KEEPFILTERS	198
Capítulo 11: CALCULATETABLE vs FILTER	201
Diferencias entre CALCULATETABLE y FILTER	201
CALCULATETABLE	201
Capítulo 12: Inteligencia de tiempo	209
Variaciones año anterior.....	210
SAMEPERIODLASTYEAR	210
Acumulados.....	215
DATESYTD	215
DATESQTD	217
DATESMTD.....	218
Comparativos mismo periodo.....	220
DATEADD	220
Medidas móviles (rangos de fecha).....	223
DATESINPERIOD.....	223
Introducción Inteligencia de tiempo personalizado	229
Comparativo semanas	229
Capítulo 13: Recetas	233
Clientes recurrentes.....	234
Clientes nuevos	236
Clientes perdidos	237
Porcentajes de participación columnas y filas.....	239
Participación a nivel de columna.....	239
Participación a nivel de fila.....	240
Proyección lineal	241
Días hábiles.....	242
Día actual.....	242
Días corridos	242
Días faltantes.....	243
Venta promedio día.....	243
Proyección	244
Estados financieros (PyG)	246

Acerca de los autores



Ingeniero financiero y de negocios, labora como consultor empresarial ayudando a decenas de empresas en soluciones de inteligencia de negocios, consultoría financiera y enseñando la mejor herramienta de análisis de datos como es Power BI.

Cofundador de la empresa Dataice.

Coautor de los libros Inteligencia de Negocios con Excel y Power BI, Estados Financieros en Excel y Power BI y los mejores 70 trucos de Microsoft Excel.

Didier Atehortúa Morales



Ingeniero de datos, con más de 10 años de experiencia en diseño de bodegas de datos, ETLs/ELTs, Bases de datos, integración y todo lo relacionado con inteligencia de negocios. Conoce y utiliza DAX desde 2015 con la salida de los modelos tabulares de Analysis Services. Apasionado por los datos y por aprender siempre nuevas tecnologías.

Cuenta con 3 certificaciones Microsoft:

-Querying Microsoft SQL Server 2012/2014

Christian Tamayo Alfonso

-Microsoft SQL Server 2008, Business Intelligence Development and Maintenance

-Programming in HTML5 with JavaScript and CSS3

Dedicatorias

Primero darle las gracias a Dios que nos da los talentos y dones para escribir libros como estos, a mi esposa Vanessa que siempre acepta el reto cuando le digo, voy a escribir otro libro, ella sabe que es un sacrificio no personal, sino familiar, ya que por ese tiempo solo me concentro en escribir y ellos me alientan detrás de cámaras, gracias, hijos: Sofía, Miguel y Simón todo es por ustedes.

Agradezco a mi gran amigo, socio y pana Julio César Rendón, no estuvo escribiendo, pero hace parte del equipo de poder sacar este libro a flote, un libro no solo es escribir, hay todo un equipo trabajando en diseños, edición, formatos, ventas etc., y entre ellos mi hermano Yibson Atehortúa que le toca un trabajo pesado, pero siempre la saca adelante.

También quiero hacer mención al gran Pablo Moreno, como siempre le digo este tipo es más conocido que la plata y con lo que sabe parece que viene de otro planeta, pero no es así, cuando lo conoces lo admirás más por la persona que es, que por lo que sabe, gracias por apoyarnos siempre y a su empresa <https://mlbi.io/> por editar y sacar nuestros libros físicos y que estos puedan llegar a cualquier lugar del mundo.

--- Didier Atehortúa Morales

Antes que nada, agradecer a Dios por presentarme la oportunidad de hacer parte de este equipo, este es mi primero libro y lo dedico especialmente a mi esposa porque sin ella muchas de las cosas que hago o emprendo no serían posibles, ya que es un tiempo que dejo de compartir con ella para sacar adelante muchos proyectos y a pesar de esto siempre está ahí apoyándome y alentándome constantemente, finalmente quiero agradecer a Didier por tenerme en cuenta para este libro, por insistirme y convencerme de esta gran oportunidad.

--Christian Tamayo Alfonso

Introducción

En los últimos 5 años se han generado más datos que los que se han producido 20 o 30 años atrás, la era digital ha impulsado el crecimiento exponencial de la información y por ende se crea la necesidad de aprender el manejo de herramientas tecnológicas para su debida manipulación, el problema de las empresas años atrás era donde guardar los datos que generaba la operación día a día, ahora el problema se trasladó y la nueva pregunta es ¿Qué hacemos con tantos datos? Microsoft siempre buscando soluciones empresariales crea un lenguaje de analítica de datos llamado DAX. DAX significa expresiones para el **Análisis de Datos**, en este libro lo explicaremos usando Power BI ya que es un lenguaje que podría aplicar a diferentes programas.

Somos conscientes del potencial que hay en DAX, por eso nos atrevimos a escribir este primer volumen, sabemos que necesitas dominar esta herramienta al máximo, seas de la profesión que seas, contador, financiero, administrador, ingeniero, abogado, profesor etc. Ya que hay algo en común que nos une, y es que al final todos trabajamos con datos para luego convertirlos en información relevante para la toma de decisiones.

Hoy en día los datos se convirtieron en el activo más importante de las compañías, sin embargo, muchas de ellas no lo saben, cuando hacemos un desarrollo o prestamos alguna de nuestras consultorías a una empresa, hacemos muchas preguntas como cualquier buen consultor haría, pero entre ellas hay preguntas claves e intencionales con el ánimo de llegar a puntos específicos y saber si en verdad están haciendo buen uso de sus datos, algunas de las preguntas “trampa” son: ¿Sabe usted que clientes le compraron el año anterior y no han vuelto a comprar este año?, ¿Cuánto le representa estos clientes en el cumplimiento de su presupuesto?, ¿Con qué periodicidad se mueve cada uno de sus productos?, ¿Cuáles son las líneas, canales, departamentos, ciudades o categorías que han tenido mayor impacto en sus ventas (crecimientos o decrecimientos)?, ¿Qué unidad de negocio le representa mejor beneficio? Estas y muchas más preguntas surgen para comprobar si los datos son tan importantes como lo que hacen en la empresa, si la respuesta es NO SÉ, le damos la bienvenida a DAX.

Power BI es la mejor herramienta de Inteligencia de Negocios que hoy existe en el mercado y cuando iniciamos a crear informes en esta, parece muy intuitivo, muy fácil, como si todo se generara por arte de magia, pero cuando necesitamos crear indicadores que requieren mayor programación, la cosa cambia, nos frustramos y hasta podemos llegar a desistir de continuar con Power BI, pero te tenemos

muy buenas noticias, aunque hay un viejo refrán muy conocido en el mundo de los datos que dice “DAX es simple pero no fácil” hemos hecho nuestro mejor esfuerzo para que tengas todo un paso a paso para lograr dominar DAX.

Así que bienvenido o bienvenida a este maravilloso mundo de DAX y esperamos que llegues hasta el final.

En el siguiente link puedes acceder al material y archivos de trabajo que usaremos durante los capítulos a estudiar

https://bit.ly/DAX_V1

Capítulo 1: ¿Por qué aprender DAX?

En la era de la información en la que vivimos actualmente en los que la digitalización de procesos empresariales y cotidianos de la vida es ahora más que una necesidad, las empresas han empezado a tener a su alcance cantidades de datos que hace unos años eran impensables, los cuales pueden generarse de la operación propia del negocio u obtenerse desde muchas fuentes externas, sobre todo con los grandes aportes de tecnologías como Big data, IoT (Internet de las cosas) entre muchos otros al sector de generación constante de datos.

Cosas tan sencillas y cotidianas para nosotros ahora como un celular, un reloj inteligente o la domótica generan datos constantemente que son subidos a internet, en sectores industriales podríamos mencionar por ejemplo, sensores de temperatura o humedad en plantas de producción dónde los dispositivos y sensores conectados a la red permiten analizar los datos y generar alarmas y mensajes que son enviados a los distintos usuarios para que tomen las acciones necesarias o incluso iniciar protocolos de forma automática, sin necesidad de la interacción humana, para corregir o tratar dichas alarmas.

Estos son sólo unos cuantos ejemplos de cómo ahora el mundo genera datos cada segundo con solo “existir” y es en este punto donde nos debemos hacer una pregunta a responder y esta sería: ¿Qué hacen las empresas con los datos? ¿Obtienen algún beneficio de ellos? ¿Se consideran para la toma de decisiones?

Es fácil crear un libro en Excel e importar cierta cantidad de datos en él. Se pueden incluso crear tablas o gráficos dinámicos que muestren información importante para muchos análisis iniciales sin utilizar las fórmulas de DAX. Pero ¿Qué sucede si necesita analizar datos críticos de ventas en varias categorías de productos y para distintos períodos de tiempo? ¿O si necesita combinar datos importantes del inventario de varias tablas y de diferentes orígenes de datos?

Llegados a este punto es donde las fórmulas de DAX nos proporcionan esta capacidad y muchas otras funciones importantes. Aprender a crear fórmulas DAX eficaces le ayudará a sacar el máximo partido a sus datos. Cuando se obtienen los datos que necesita estudiar, se manipulan y se exponen de tal forma que puedan ser analizados para arrojar resultados de valor para su operación, puede empezar a

solucionar las preguntas y los problemas empresariales reales que afectan directamente los cimientos de su negocio. Se trata de Business Intelligence y DAX le ayudará a entrar en esa materia.

¿Por qué aparece DAX?

Podríamos decir que DAX aparece por un motivo fundamental y es que necesitamos tratar o manipular nuestros datos de alguna forma que nos permita llevar a cabo análisis de valor, ya que estos en su forma original nunca tendrán toda la información que requerimos e inevitablemente tienen que ser tratados para lograr este objetivo.

Entrando en materia, DAX, significa **Data Analysis eXpressions** y es un lenguaje de fórmulas desarrollado por Microsoft que se utiliza o se ha utilizado principalmente en herramientas como Power BI, Microsoft Analysis Services y Microsoft Power Pivot para Excel y fue creado en 2010, con el primer lanzamiento de PowerPivot para Microsoft Excel 2010.

Desde entonces, DAX ha ganado popularidad, tanto dentro de la comunidad de Excel, que usa DAX para crear modelos de datos de Power Pivot en Excel, como dentro de la comunidad de Business Intelligence (BI), que usa DAX para construir modelos con Power BI y Analysis Services. DAX está presente en muchas herramientas diferentes, todas compartiendo el mismo motor interno llamado Tabular. Por esta razón, a menudo nos referimos a modelos tabulares, que incluyen todas estas herramientas diferentes en una sola palabra.

Aunque en un principio el término DAX puede llegar a sonar algo intimidante, no permita que su nombre y terminología lo acobarde. Los fundamentos de DAX son muy fáciles de comprender en realidad. En primer lugar: NO es un lenguaje de programación. DAX es un lenguaje de fórmulas o funcional. Puede usar DAX para definir cálculos personalizados para columnas calculadas y para medidas (también conocidas como campos calculados). DAX incluye algunas de las funciones que se usan en las fórmulas de Excel, así como funciones adicionales diseñadas para trabajar con datos relacionales y realizar agregaciones dinámicas.

DAX es un lenguaje simple. Dicho esto, DAX es diferente de la mayoría de los lenguajes de programación principalmente por lo que ya mencionamos anteriormente DAX es un lenguaje de fórmulas y no tanto un lenguaje de programación, por lo que conocer algunos de sus nuevos conceptos puede requerir algo de tiempo. Aprender los conceptos más básicos de DAX en la mayoría de los casos es sencillo y podrá

comenzar a usarlo en cuestión de horas, con el tiempo y la práctica descubrirá que DAX es, de hecho, un lenguaje fácil, solo se necesita un poco de tiempo para acostumbrarse.

DAX requiere fundamentos teóricos

Debemos ser claros en que el hecho de que DAX requiere de estudio y teoría, esto no lo diferencia de muchos lenguajes de programación. Es posible que esté acostumbrado a buscar en internet fórmulas complejas y patrones de solución para algunos de los escenarios que son propios de su negocio en busca de alguna solución que se ajuste a ellos, no obstante, si hablamos de Excel, es muy probable que encuentre una fórmula que haga prácticamente lo que usted necesita. En cuyo caso puede copiar la fórmula y ajustarla a sus necesidades y luego aplicarla en sus cálculos sin preocuparse demasiado por cómo funciona.

Sin embargo, esta práctica, que funciona en Excel, no le funcionará de igual forma con DAX. Es muy importante estudiar la teoría de DAX, comprender e interiorizar cómo funcionan los contextos de evaluación antes de poder escribir un buen código DAX. Si no tiene una base teórica adecuada, empezará a darse cuenta de que en muchas ocasiones DAX calcula los valores mágicamente o por el contrario calcula resultados extraños que no tienen ningún sentido. En este escenario el problema no es DAX, sino el hecho de que aún no ha comprendido exactamente cómo funciona DAX.

Afortunadamente, la teoría detrás de DAX se limita a un par de conceptos importantes, que veremos en detalle en el Capítulo 9, Evaluación de contextos. Cuando llegue a ese capítulo, se dará cuenta que allí se encuentra el pilar fundamental para la comprensión del funcionamiento de DAX, en otras palabras, DAX prácticamente ya no tendrá misterios para usted, básicamente estará listo para codificar expresiones mucho más eficaces y precisas para dar solución a muchos de los problemas que se le presenten en su día a día. Por lo demás será ir adquiriendo cada vez más competencias en este sentido que irá obteniendo con la práctica para así alcanzar niveles más avanzados.

Resumen de tipos de datos y operadores

En la tabla siguiente se enumeran los tipos de datos admitidos en un modelo de datos. Al importar datos o usar un valor en una fórmula, incluso si el origen de datos original contiene un tipo de datos

diferente, los datos se convierten en uno de estos tipos de datos. Los valores resultantes de las fórmulas también usan estos tipos de datos.

Tipo de datos en Excel	Tipo de datos en DAX	Descripción
Número entero	Un valor entero de 64 bits (ocho bytes) 1, 2	Números que no tienen posiciones decimales. Los números enteros pueden ser números positivos o negativos, pero deben ser números enteros entre -9.223.372.036.854.775.808 (-2^63) y 9.223.372.036.854.775.807 (2^63-1).
Número decimal	Un número real de 64 bits (ocho bytes) 1, 2	Los números reales son números que pueden tener posiciones decimales. Los números reales abarcan un amplio rango de valores: Valores negativos de -1,79E +308 a -2,23E -308 Cero Valores positivos de 2,23E -308 a 1,79E + 308 Sin embargo, el número de dígitos significativos se limita a 15 dígitos decimales.
VERDADERO/FALSO	Boolean	Un valor Verdadero o Falso.
Texto	String	Una cadena de datos de caracteres Unicode. Pueden ser cadenas, números o fechas representados en un formato de texto.

		La longitud máxima de cadena es de 268.435.456 caracteres Unicode (256 mega caracteres) o 536.870.912 bytes.
Fecha	Fecha y hora	Fechas y horas en una representación de fecha y hora aceptada. Las fechas válidas son todas después del 1 de enero de 1900.
Moneda	Moneda	El tipo de datos moneda permite valores entre -922.337.203.685.477.5808 a 922.337.203.685.477.5807 con cuatro dígitos decimales de precisión fija.
N/D	En blanco	Un blanco es un tipo de datos de DAX que representa y reemplaza valores nulos. Puede crear un blanco con la función BLANK y probar si hay espacios en blanco con la función lógica ISBLANK.

Operadores de comparación

En las expresiones de comparación, los valores booleanos se consideran mayores que los valores de cadena y los valores de cadena se consideran mayores que los valores numéricos o de fecha y hora; se considera que los números y los valores de fecha y hora tienen la misma jerarquía. No se realizan conversiones implícitas para valores booleanos o de cadena; BLANK o un valor en blanco se convierte en 0//false según el tipo de datos del otro valor comparado.

Las siguientes expresiones de DAX ilustran este comportamiento:

=SI (FALSO ()>"verdadero", "Expresión es verdadera", "Expresión es falsa"), devuelve "Expresión es verdadera"

=SI ("12">12,"Expresión es verdadera", "Expresión es falsa"), devuelve "Expresión es verdadera".

=SI ("12"=12,"Expresión es verdadera", "Expresión es falsa"), devuelve "Expresión es falsa"

Las conversiones se realizan implícitamente para tipos numéricos o de fecha y hora, como se describe en la tabla siguiente:

Operador de comparación	INTEGER	CURRENCY	REAL	Fecha y hora
INTEGER	INTEGER	CURRENCY	REAL	REAL
CURRENCY	CURRENCY	CURRENCY	REAL	REAL
REAL	REAL	REAL	REAL	REAL
Fecha y hora	REAL	REAL	REAL	Fecha y hora

- Otros Operadores:

- Aritméticos: (+ Suma, - Resta, * Multiplicación, / División, ^ Exponenciación)
- Operador de concatenación de cadenas: &

- Operadores lógicos:

Puede usar los operadores lógicos (&&) y (||) para combinar expresiones y obtener un único resultado.

- && (doble ampersand): Crea una condición de disyunción entre dos expresiones que devuelven un valor booleano, es decir, verdadero o falso, si ambas expresiones retornan verdadero, la combinación de las expresiones también retorna verdadero; de lo contrario la combinación devuelve falso. Ejemplo: (([Región] = "Francia") && ([Comprador] = "Yes"))
- || (doble símbolo pipe): Crea una condición de conjunción entre dos expresiones lógicas. Si alguna de las dos expresiones retorna verdadero, el resultado será verdadero; solamente cuando ambas expresiones son falsas el resultado será falso. Ejemplo: (([Región] = "Francia") || ([Comprador] = "Yes"))

- IN: Crea una condición de conjunción lógica entre cada fila que se compara con una tabla.
La sintaxis del constructor de tablas se usa con llaves. Ejemplo: ‘Producto’[Color] IN {“Rojo”, “Azul”, “Negro”}.
- Convenciones de nombres:
 - En DAX el nombre de las tablas va entre comillas simples, ejemplo: ‘Ventas último año’. Si el nombre de la tabla no tiene caracteres prohibidos como el espacio estas comillas se pueden omitir, por ejemplo: Ventas
 - El nombre de las columnas entre corchetes ej. ‘Venta’[Cantidad].



Nota: Entraremos en mayor detalle acerca de los operadores y otros aspectos interesantes ya mencionados en los capítulos del 4 al 7.

Capítulo 2: Modelo de datos

Desde la liberación de SQL Server 2012 (Un motor de base de datos desarrollado por Microsoft) Microsoft introdujo los modelos de datos tabulares como una alternativa a los ya existentes modelos multidimensionales. Los modelos tabulares son un nuevo tipo de modelo de datos que se introdujo específicamente en SSAS (SQL Server Analysis Services) los cuales en su momento eran principalmente utilizados como modelos relacionales dentro de proyectos de Power Pivot en Excel.

En ese entonces muchas personas se preguntaron ¿por qué Microsoft lanzó este nuevo modelo cuando ya tenían la facilidad de seguir trabajando y mejorando la tecnología de los modelos multidimensionales o bases de datos relacionales. ¿Cuáles son las ventajas y desventajas de usar los modelos tabulares en comparación con los modelos multidimensionales? Microsoft también incluyó un nuevo motor de análisis en memoria llamado xVelocity dentro de SQL Server 2012, no obstante, Microsoft ya había probado suerte en 2010 con la tecnología de este motor y demostró ser un gran punto de inflexión en los servicios de análisis que ya ofrecían hasta la fecha. Podríamos decir que los modelos tabulares son un punto medio entre los modelos multidimensionales y las bases de datos relacionales y más adelante aclararemos el por qué.

Qué son los modelos Tabulares

Según la definición de Microsoft en MSDN del modelo tabular en SSAS "los modelos tabulares son bases de datos de Analysis Services que se ejecutan In Memory (En memoria) o en modo DirectQuery (Consultas directas), accediendo a los datos directamente desde fuentes de datos relacionales de backend".

Un modelo tabular es una base de datos columnar con alto nivel de rendimiento y relación de compresión que admite dos modos "In Memory" y "DirectQuery".

En el modo "In Memory", los modelos tabulares son una base de datos que incluye o carga todos los datos desde sus diferentes orígenes en la memoria. El modelo tabular logra hacer esto con la ayuda de

su motor xVelocity. xVelocity permite que el modelo tabular coloque todos los objetos y sus datos en la memoria con algoritmos de compresión de última generación y le proporciona a este una funcionalidad de ejecución de consultas de subprocesos múltiples.

El modo "Direct Query" funciona parcialmente con el modo "In Query" (En consulta). La razón detrás de proporcionar este modo es poder manipular grandes cantidades de datos que no cabrían en memoria o cuando no hay un plan adecuado de procesamiento de datos debido a la imprevisibilidad o inestabilidad de los datos. Ambos modelos manejan tablas y columnas calculadas, seguridad a nivel de fila y optimizaciones de consultas.

Los modelos tabulares se pueden implementar en Power BI Premium, Azure Analysis Services o una instancia de SQL Server Analysis Services configurada para el modo de servidor tabular. Los modelos tabulares implementados se pueden administrar a través de SQL Server Management Studio en el caso de tener una instancia de servidor en modo tabular en Analysis Services o mediante Power BI Desktop.

¿Por qué la necesidad de un modelo tabular?

En el momento de la introducción de los modelos tabulares en SQL 2012, volvemos a la pregunta que muchas personas tenían entonces "¿cuál es la necesidad de un modelo tabular en el mundo actual?" La razón detrás de esto es muy simple, para acomodar requerimientos que quedan en el aire entre los modelos multidimensionales y las bases de datos relacionales. Las características faltantes de un modelo multidimensional y base de datos relacional crean un vacío que da lugar al modelo tabular. El modelo tabular tiene una combinación de funciones de MOLAP (multi-dimensional online analytical processing) y de bases de datos relacionales. Una base de datos relacional admite tablas y relaciones, pero al mismo tiempo no admite indicadores clave de rendimiento o en inglés KPI (Keep Performance Indicator) ni medidas. Por lo tanto, el usuario debe pasar a los modelos multi dimensionales, pero esta es una estructura muy compleja, necesita una persona experta para implementarla y, al mismo tiempo, es muy cierto decir que es costosa de implementar. Básicamente, MOLAP es complejo, costoso, consume mucho tiempo y tiene problemas con el recuento de valores distintos, al mismo tiempo, una base de datos relacional no proporciona un acceso rápido a consultas complejas como MOLAP y, en la mayoría de los casos, su diseño rara vez está enfocado para responder rápidamente a las consultas de análisis.

Un modelo tabular es muy efectivo y simple. Proporciona un modelo semántico con un sistema de memoria caché que proporciona rendimiento y escalabilidad. Microsoft marcó la diferencia entre el modelo Tabular y MOLAP, lo que diferencia su lenguaje y las herramientas utilizadas para definir los modelos. Las principales razones para elegir el modelo tabular son proporcionar un mejor rendimiento, un bajo costo de mantenimiento y propiedad; Más allá de esto, proporciona simplicidad y flexibilidad en el modelado de datos. En pocas palabras un modelo tabular es un modelo potenciado por Microsoft con lenguaje DAX y motor xVelocity.

Comprensión del modelo de datos

En la carpeta de archivos encontrarás este modelo el cual explicaremos detalle a detalle en cuanto a relaciones, tablas, columnas y medidas.

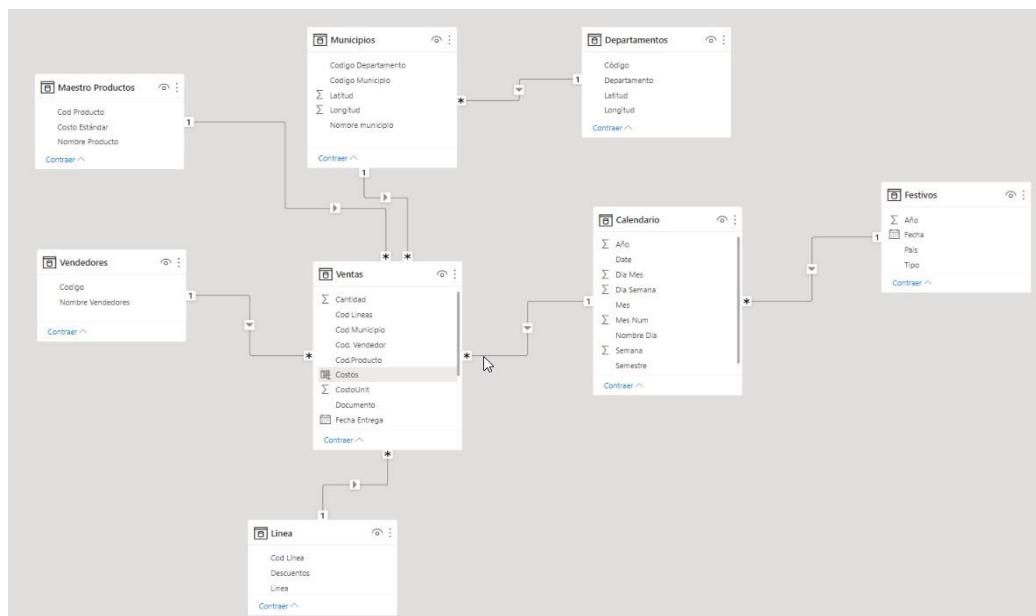


Figura 2.1 Modelo de datos.

DAX está específicamente diseñado para calcular fórmulas de negocio sobre un modelo de datos. Una definición de modelo de datos podría ser: una manera de estructurar y organizar los datos para que estos puedan ser fácilmente utilizados y estas estructuras pueden ser un conjunto de tablas vinculadas por relaciones, estaríamos hablando entonces de un modelo relacional. Todos sabemos qué es una tabla: Es un objeto que contiene muchos datos y estos están organizados en filas y columnas, similar al de una hoja de cálculo de Excel. Cada fila representa un registro único y cada columna un campo dentro del registro. Por ejemplo: si nos centramos en la tabla “Maestro Productos” de la figura 2.1 podríamos inferir que hay una fila para cada producto y distintas columnas en las que figuran detalles de estos,

como son el código del producto, costo estándar y nombre del producto. Las tablas son una forma conveniente de organizar los datos. Podríamos decir que una tabla es un modelo de datos en sí misma, aunque en su forma más simple. Por lo tanto, cuando escribimos nombres y números en un libro de Excel, estamos creando un modelo de datos.

Relaciones

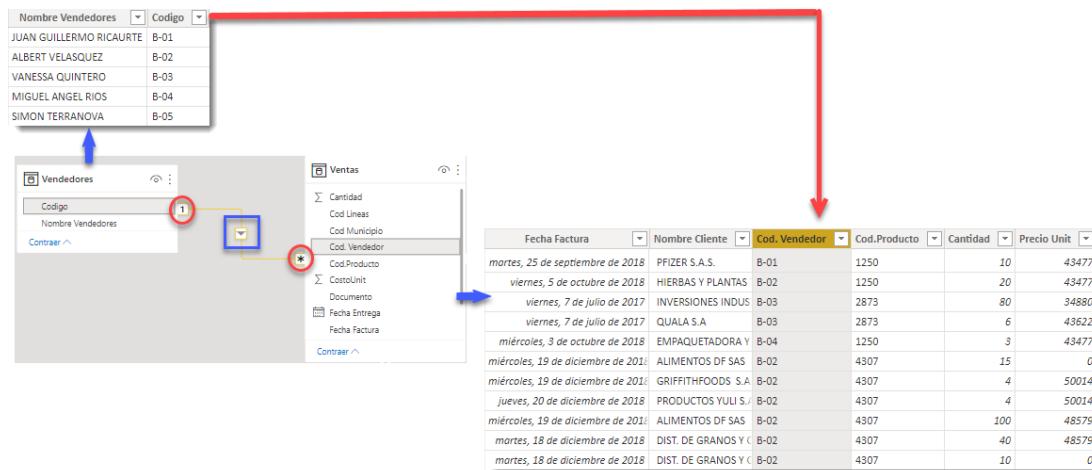


Figura 2.2 Relaciones y cardinalidad.

Cuando un modelo de datos contiene más de una tabla, es muy probable que estas estén vinculadas por medio de relaciones. Una relación es simplemente un enlace entre dos tablas. Nos referimos a que dos tablas están relacionadas cuando entre ellas existe un vínculo, que gráficamente está representado por una línea que conecta las dos tablas como se muestra en la figura 2.1, todas las líneas que conectan las tablas de la imagen son las relaciones. Para crear una relación es necesario tener una columna que exista en ambas tablas, que por lo general tienen el mismo nombre y por ende los mismos valores, a estas columnas se les conoce como llaves de relación, como vemos en la figura 2.2 tenemos la columna “Código” de la tabla Vendedores y la columna “Cod. Vendedor” de la tabla Ventas, las cuales en este caso son las llaves de relación entre estas dos tablas. Cuando una columna tiene un valor único para cada fila dentro de una tabla, se denomina “llave” para esa tabla, por ejemplo, en la figura 2.2 podemos ver que la llave de la tabla vendedores sería la columna “Código” ya que esta tiene un valor único que identifica cada vendedor.

Ahora, estas relaciones, tienen ciertas características dentro de Power BI y DAX, que veremos a continuación:

Cardinalidad

En palabras simples la cardinalidad es el número de entidades con las cuales una tabla, en este caso, se puede asociar a otra mediante una relación. Si nos centramos en la tabla Vendedores y Ventas nuevamente de la figura 2.2 podemos ver que la cardinalidad se expresa en el extremo de la relación de cada tabla (los valores encerrados en círculos).

Existen cuatro tipos de cardinalidad los cuales se escriben, por lo general, con el símbolo asterisco (*) y el número (1) e indican que de un lado de la relación los valores no se repiten (1) o que por el contrario pueden llegar a hacerlo (*), y se expresan con las siguientes etiquetas para cada uno de ellos:

- (1:*) uno a varios: Esto significa que en una de las dos tablas relacionadas existe un registro único para cada uno de los valores del conjunto y en la otra tabla pueden existir más de una vez dentro de ese conjunto, por ejemplo, volviendo a la imagen 2.2 vemos que la tabla Vendedores tiene un (1) en la cardinalidad de su relación con Ventas, esto nos indica que en dicha tabla un vendedor existe sólo una vez en la tabla vendedores, es decir, que no se repiten, pero, en la tabla Ventas vemos que esta tiene un asterisco (*) por lo tanto quiere decir que allí puede existir un vendedor varias veces, porque un vendedor puede tener muchas ventas.
- (*:1) varios a 1: Esta cardinalidad es prácticamente la explicada con anterioridad, pero a la inversa, tomando nuevamente el ejemplo de las tablas Vendedores y Ventas si leemos a la inversa, nos está diciendo que puede haber muchas ventas que pertenezcan a un vendedor en la tabla Vendedores.
- (1:1) uno a uno: Esta cardinalidad nos estaría indicando que tanto de un lado como del otro los registros existen una sola vez en cada tabla y que un registro de un lado está relacionado únicamente con un registro del otro lado. Supongamos un caso hipotético en el que a cada vendedor se le asigna solamente una ciudad en la cual operar, y que estas ciudades están en otra tabla que tiene el nombre de la ciudad y el código del vendedor, allí tendríamos un claro ejemplo en el que podríamos tener una relación de uno a uno.
- (*:*) varios a varios: Aquí tenemos un caso especial y poco frecuente, significa que un registro puede existir varias veces de un lado y varias veces del otro. Un ejemplo con en el que podríamos

ilustrar este escenario podría ser: un vendedor puede vender varios productos y un producto puede ser vendido por varios vendedores, sin embargo, esta relación conlleva mucha complejidad dentro de un modelo de datos y se pueden usar las demás alternativas de cardinalidad para evitar un caso complejo en el que debamos tener una relación de varios a varios.

Dirección de filtro cruzado

Las relaciones en DAX y Power BI poseen una dirección, (ver figura 2.2 en donde se encierra la dirección de la relación dentro de un cuadrado azul) y el propósito de esta es propagar los filtros entre las tablas a través de las relaciones.

Existen dos tipos de dirección de filtro cruzado Unidireccional y Bidireccional que en Power BI se llaman “Única” y “Ambas” básicamente, la relación unidireccional propaga los filtros en una sola dirección a través de las relaciones y la bidireccional pude propagarlos en ambas direcciones, veámoslo en detalle.

- Unidireccional o “única”:

En el caso de la figura 2.2 vemos que la dirección de la relación va de Vendedores hacia Ventas, por lo tanto, cualquier filtro que se haga en la tabla Vendedores automáticamente se replicará hacia la tabla de Ventas, sin embargo, no sucede a la inversa, cualquier filtro que se haga en la tabla Ventas no se propagará a la tabla vendedores, ya que la propagación del filtro de esa forma iría en contra de la dirección de la relación. Ilustremos este concepto con un ejemplo para comprenderlo mejor con la figura 2.3.

Nombre Vendedores	Cantidad
ALBERT VELASQUEZ	237.775
JUAN GUILLERMO RICAURTE	14.257.462
MIGUEL ANGEL RIOS	315.847
SIMON TERRANOVA	25.597
VANESSA QUINTERO	714.856
Total	15.551.537

Figura 2.3 Efecto de filtrado “único”.

“Nombre vendedores” es una columna que pertenece a la tabla Vendedores. Como el nombre está en un lado de la relación con las ventas como se observa en la figura 2.2, el motor filtra las

ventas según el nombre del vendedor. Por esta razón la cantidad mostrada se filtra por vendedor.

- Bidireccional o “ambas”: Para comprender fácilmente esta relación veremos nuevamente un ejemplo de cómo se verían los datos con un filtro unidireccional y luego, veremos cómo se verían aplicándolo de forma bidireccional.

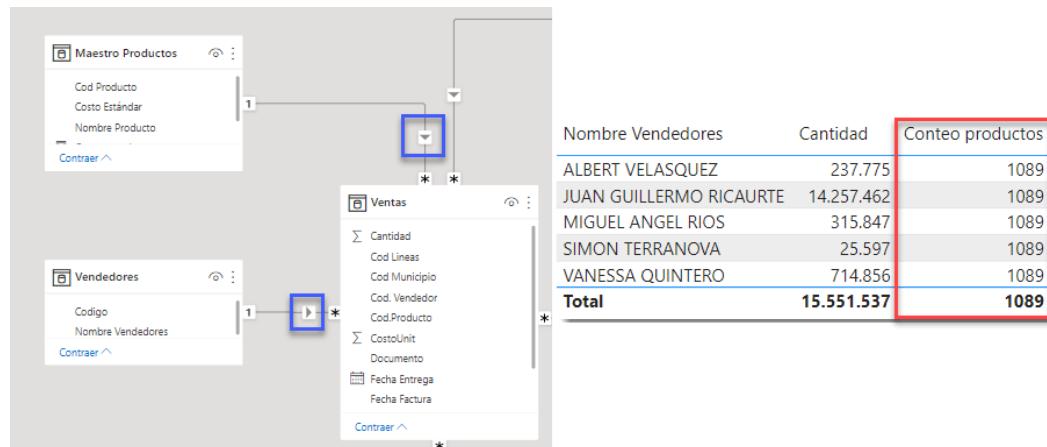


Figura 2.4 Esta imagen muestra que, si el filtro bidireccional no está activo, la tabla Maestro Productos no se filtra por vendedor.

El filtro en esta figura es el nombre del vendedor de la tabla vendedores, por lo tanto como la cantidad está en la tabla ventas y el filtro cruzado de la relación entre Vendedores por su dirección propaga el filtro hacia la tabla ventas, se calcula correctamente la cantidad de ventas por vendedor, sin embargo, el cálculo errado es Conteo productos ya que nos está mostrando el mismo valor para cada vendedor, es decir, está mostrando la cantidad total de productos que hay en la tabla Maestro products y no la cantidad de productos que ha vendido cada vendedor. El filtro que procede de la columna Nombre vendedores no se propaga a Maestro Productos porque la relación entre Ventas y Maestro Productos es unidireccional y en la dirección de Productos hacia Ventas. Por esta razón, aunque Ventas tiene el filtro de nombre de vendedor activo este no se puede propagar hacia productos porque la dirección de la relación lo impide. Si cambiamos la dirección entre Ventas y Maestro de productos a bidireccional, esto permitirá que el filtro que viene de Vendedores se propague de Ventas hacia Productos como se muestra en la figura 2.5.

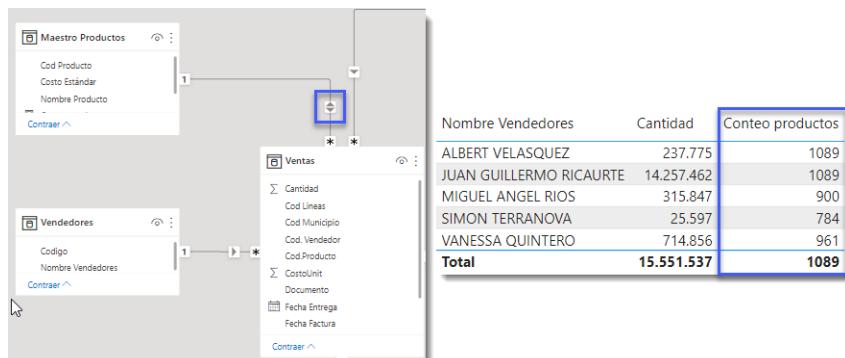


Figura 2.5 Con el filtro bidireccional habilitado, la tabla Maestro productos recibe el filtro que proviene de Vendedores.

El cálculo Conteo productos ahora refleja la cantidad de productos que vendió cada vendedor. Probablemente esté pensando que todas las relaciones deberían definirse como bidireccionales, para así propagar los filtros en cualquier dirección y siempre devolver resultados que sean coherentes, sin embargo, diseñar un modelo de datos de esta manera en muy pocos escenarios es apropiado y se debe evitar en la medida de lo posible, hablaremos más acerca de esto en el apartado de recomendaciones dentro de este mismo capítulo.

Para crear una relación en Power BI, puede hacerlo de varias formas:

- Puede dar clic sostenido a la columna Llave de la tabla y arrastrarla hacia la columna Llave de la tabla que desea vincular:

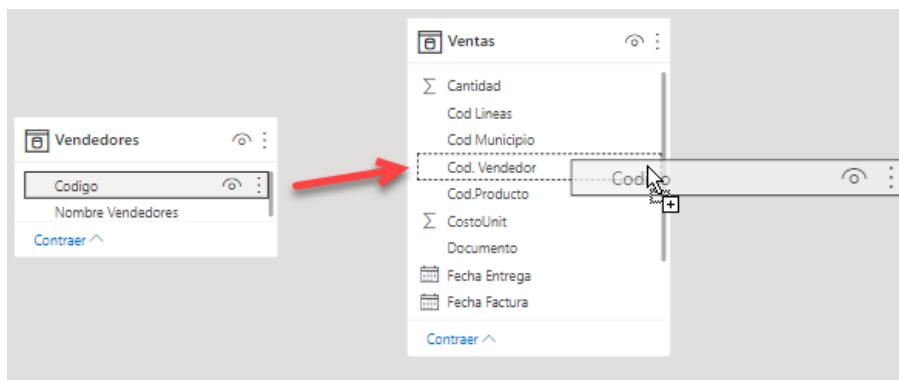


Figura 2.6 Crear relación arrastrando campo.

Automáticamente Power BI identificará el tipo de relación y la creará automáticamente.

- Otra forma más específica es dando clic derecho al nombre cualquier tabla y seleccionar la opción “Administrar relaciones” o también puede hacer clic sobre esta misma opción a través de la pestaña “Inicio” de la barra de menú:

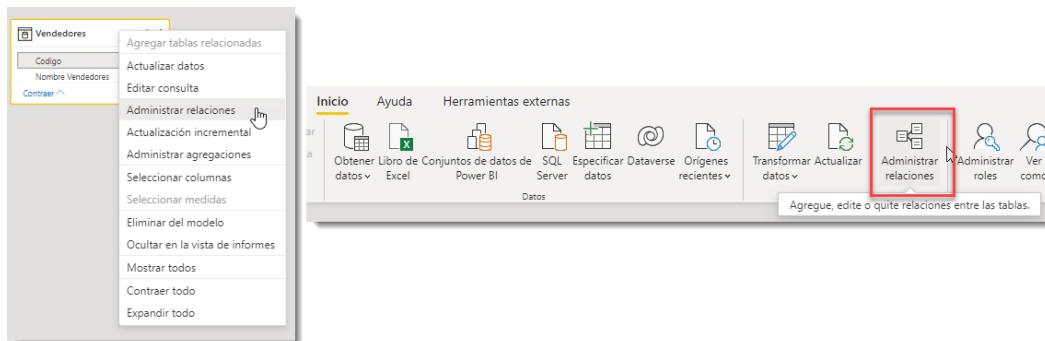


Figura 2.7 Crear relación arrastrando, administrar relaciones.

En ambos casos se abrirá la ventana de administración de relaciones, aquí aparecen las relaciones previamente creadas en todo el modelo, seleccionamos la opción “Nuevo...”:

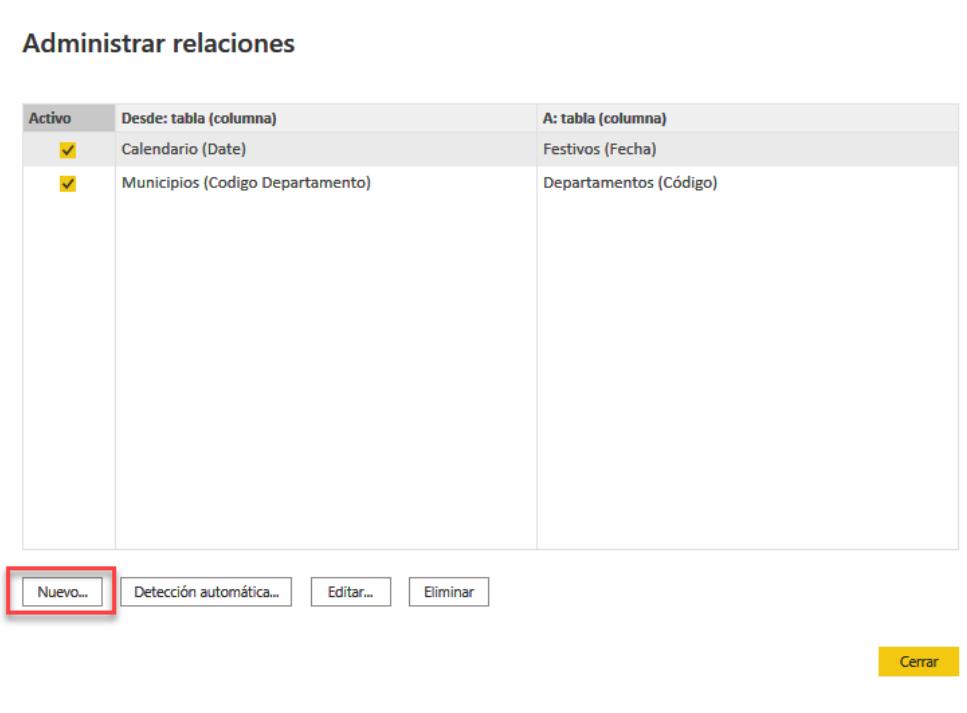


Figura 2.8 Crear relación arrastrando, administrar relaciones.

Posteriormente se abrirá la ventana de creación de relaciones en la que verá todas las características que ya hemos definido previamente, en esta ventana deberá seguir estos pasos para crear la nueva relación:

1. Seleccione la primera tabla que desea relacionar.
2. Seleccione la columna llave de esta tabla por medio de la cual creará el vínculo.
3. Seleccione la segunda tabla que desea relacionar con la primera.
4. Seleccione la columna llave de la segunda tabla.

5. Elija la cardinalidad que desea para la nueva relación de acuerdo con los datos que poseen ambas tablas.
6. Elija la dirección del filtro cruzado para la nueva relación
7. De clic en “Aceptar”

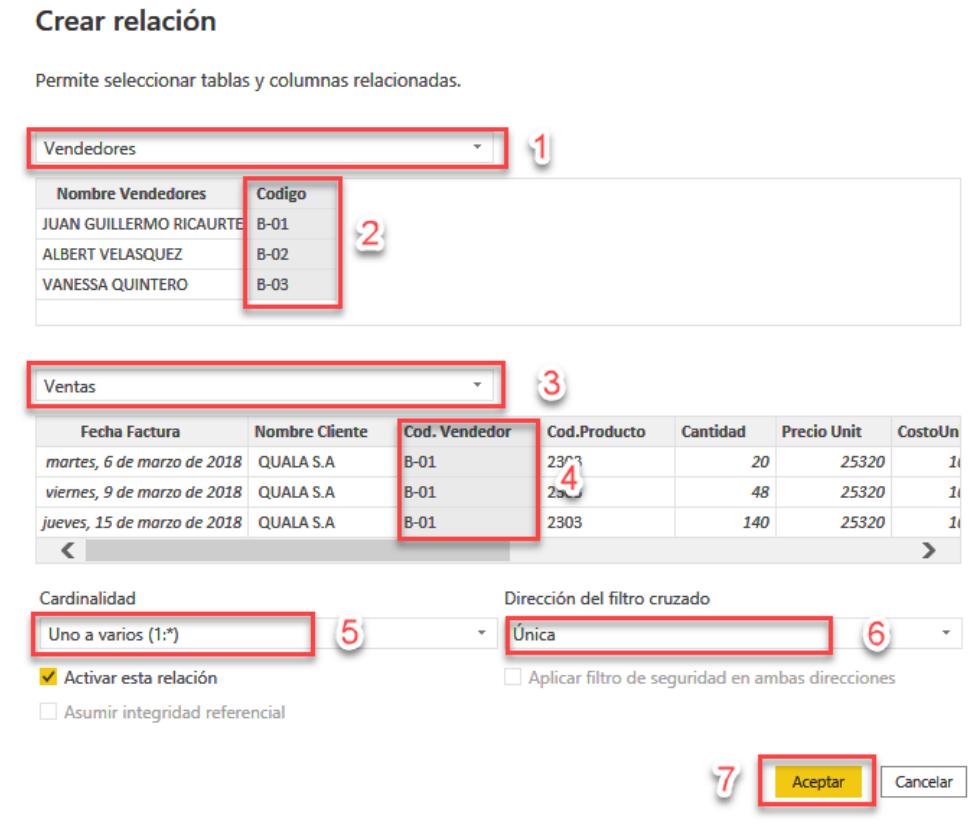


Figura 2.9 Editar relación arrastrando, administrar relaciones.

Una vez de clic en aceptar regresará a la ventana de “Administrar relaciones” en donde podrá visualizar ahora la nueva relación que ha creado, en esa misma ventana puede editar o eliminar la nueva relación o cualquier otra que haya creado previamente, cuando haya terminado, puede dar clic en el botón cerrar.

Recomendaciones

En este apartado le sugeriremos algunas recomendaciones que a nuestro criterio son de utilidad al momento de diseñar un modelo de datos:

- Cuando pudiésemos tener una relación de 1 a 1 lo ideal en este escenario sería unificar estas dos tablas en una sola para evitar utilizar esta relación dado que si de un lado tengo registros que sólo se relacionan con sólo un registro del otro lado podríamos entonces unir estas dos

tablas conservando los campos comunes entre ambas y adicionalmente agregando las columnas diferentes en un solo conjunto de datos, lo que facilitará las demás relaciones con las otras tablas del modelo.

- Evitar las relaciones de varios a varios, como lo mencionamos anteriormente esta cardinalidad (*:*) es para escenarios muy complejos y conlleva muchos resultados inesperados en los datos, principalmente porque en una relación de varios a varios no se puede garantizar que todos los valores de A existan en B por lo tanto los valores mostrados no incluyen una fila en blanco dedicada a las filas que no coinciden en la otra tabla. Además, los valores no se aplican a las filas donde la columna usada en la relación de la otra tabla es nula.
- Evitar tanto como sea posible las relaciones bidireccionales. La presencia de ese filtro cruzado bidireccional podría crear en muchos escenarios rápidamente nuestra peor pesadilla, porque introduce ambigüedad en el modelo. ¿Qué es la ambigüedad? Un modelo es ambiguo cuando hay varias rutas entre las tablas para resolver una consulta. En un modelo ambiguo, el motor tiene múltiples opciones al transferir un filtro de una tabla a otra. Por lo tanto, termina encontrando una forma preferida de transferir el filtro o genera un error. En el caso de que el motor encuentre una ruta preferida para propagar los filtros, se generarán resultados indeseados en los datos y el modelo se tornará mucho más difícil de comprender para cálculos o métricas que sin esta ambigüedad serían muy sencillas, no obstante, en casos muy específicos el filtro bidireccional podría llegar a ser útil, aun así, nosotros como diseñadores del modelo debemos estar conscientes de los riesgos que esto conlleva ya que seremos los responsables de que los datos tengan sentido y sean coherentes en los diferentes análisis.
- Es ideal trabajar con tablas relacionadas que segmenten la información a analizar, en lugar de trabajar con toda la información en una sola tabla, lo que comúnmente a nivel técnico se llama topología de “Estrella”. En este libro no abordaremos el tema de esta topología, sin embargo, para dar claridad sobre la importancia de este punto es necesario tocar muy por encima algunos conceptos (si desea saber más al respecto lo invitamos a buscar el término “topología estrella” en internet y encontrará mucha información al respecto). Como lo define Microsoft, en esta topología o estilo de modelamiento tenemos tablas denominadas dimensiones y otras llamadas hechos. Las tablas de dimensiones describen entidades empresariales (las cosas que se modelan). Las entidades pueden incluir productos, personas, lugares y conceptos, incluido el

propio tiempo. Por otro lado, las tablas de hechos pueden almacenar observaciones o eventos, y pueden ser pedidos de ventas, existencias, tasas de cambio, temperaturas, etc. Una tabla de hechos contiene columnas de clave de dimensiones relacionadas con las tablas de dimensiones y columnas de medida numéricas. La importancia de este tipo de modelos estrella en Power BI radica fundamentalmente en que el diseño de esquema de estrella y muchos otros conceptos relacionados son muy importantes para desarrollar modelos de Power BI optimizados para el rendimiento y la facilidad de uso.

Tenga en cuenta que cada objeto visual de informe de Power BI genera una consulta que se envía al modelo de Power BI (lo que el servicio Power BI denomina un conjunto de datos). Estas consultas se usan para filtrar, agrupar y resumir los datos del modelo. Por tanto, un modelo bien diseñado es aquel que proporciona tablas para filtrar y agrupar y tablas para resumir. Este diseño se ajusta bien a los principios de los esquemas de estrella:

- Las tablas de dimensiones admiten el filtrado y la agrupación
- Las tablas de hechos admiten el resumen.

Un diseño de modelo bien estructurado debe incluir tablas de tipo de dimensiones o de tipo de hechos. Evite mezclar los dos tipos en una sola tabla. También se recomienda intentar ofrecer el número correcto de tablas con las relaciones adecuadas aplicadas. Además, es importante que las tablas de tipo de hechos siempre carguen datos en un nivel de detalle coherente.

Capítulo 3: Tipos de cálculos en DAX

DAX es uno de estos lenguajes como hemos mencionado anteriormente de carácter obligatorio para cualquier analista o persona que labore con datos en Power BI, entender su comportamientos y lógica nos hace la vida más fácil para nuestros desarrollos y análisis; podemos crear 3 tipos de cálculos en DAX:

Tablas, Columnas y Medidas

Antes de entrar en materia con estos cálculos reiteramos la importancia de un buen modelo de datos, de esto depende que nuestros cálculos sean más simples de formular y no tener que incurrir en complejidades, ya que como empresa nos hemos caracterizado por nuestra teoría el poder de lo simple, y consiste en lo complejo volverlo simple y lo simple dejarlo así. Como cualquier lenguaje de programación tiene sus profundidades, podemos avanzar cuanto queramos, pero nunca olvide lo esencial, los conceptos, las bases necesarias para llegar a ser este experto deseado y vamos paso a paso.

Es importante que escuches esto por nuestra experiencia, la mayoría o muchos analistas de datos venimos del mundo del Excel y te confesamos nunca iniciamos creando macros o fórmulas complejas, se inicia creando una simple SUMA o un condicional y luego pasamos a una función de búsqueda, pero nunca se llega a la perfección o manejo total de una herramienta, ya que con el solo hecho de sus actualizaciones o novedades tenemos algo nuevo que aprender.

En este capítulo vamos a la base de todo, los tipos de cálculos que se pueden desarrollar en DAX, aunque ya tengas conocimientos en el lenguaje o apenas estas ingresando en este mundo es importante conocer que se puede crear en DAX, ya que al final todo se resume en medidas, debemos conocer los cálculos de tablas y de columnas, vas a escuchar por todo medio que no debes crear columnas calculadas o llenarte de tablas ya que esto afecta el rendimiento en los modelos y no te mentimos, es una verdad absoluta, pero si NO conoces las funciones de tablas o cuando sí o cuando no se debe crear una columna no entenderás los resultados que van arrojar las medidas.



Nota: Los cálculos se van a crear con el modelo que se explicó en el capítulo anterior

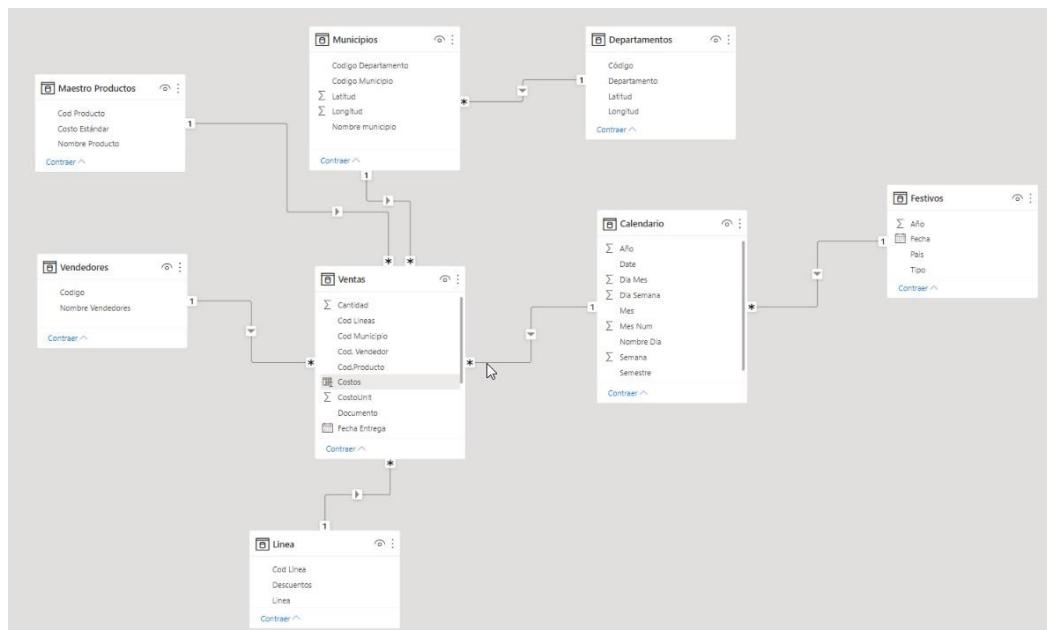


Figura 3.1 Modelo de datos.

Los tipos de cálculos que DAX nos ofrece los podemos encontrar en diferentes partes de Power BI.

En el lienzo en la pestaña Modelado podemos ver las opciones:

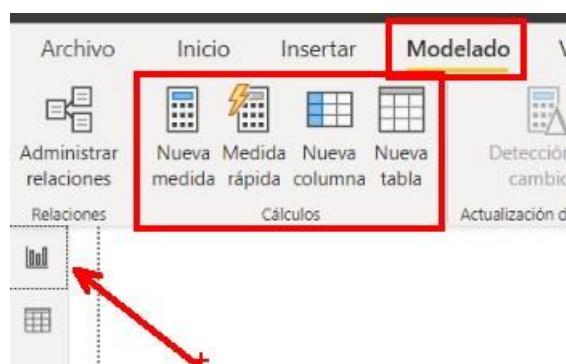


Figura 3.2 Tipos de cálculos pestaña modelado.

También podemos encontrar estas opciones en Datos → Inicio

Date	Año	Mes Num	Mes	Trimestre	Semestre	Semana	Día Mes	Día Semana	Nombre Día
1/07/2020 12:00:00 a. m.	2020	7	Julio	T3	Sem 2	27	1	3	miércoles
2/07/2020 12:00:00 a. m.	2020	7	Julio	T3	Sem 2	27	2	4	jueves
3/07/2020 12:00:00 a. m.	2020	7	Julio	T3	Sem 2	27	3	5	viernes
4/07/2020 12:00:00 a. m.	2020	7	Julio	T3	Sem 2	27	4	6	sábado

Figura 3.3 Tipos de cálculos en Datos.

En la pestaña datos vamos a crear nuestras tablas y columnas para ver los resultados.

Tablas

Cuando pensamos en tablas, nos imaginamos una serie de filas y columnas, pero no necesariamente debe ser así, una tabla puede estar compuesta de un único registro con un encabezado y esto es importante porque los filtros en DAX son tablas, analizaremos esto en capítulos posteriores.

DAX tiene la posibilidad de crear tablas por medio de sus funciones, este lenguaje permite crear tablas de manera rápida, fácil y muy intuitivamente, todas sus funciones son en inglés y el separador de los argumentos por defecto es la coma “,”.

Si deseamos usar los separadores que tengamos por defecto en nuestro sistema, ejemplo si usas Excel y deseas usar el mismo separador como es punto y coma “;” debes seguir la siguiente ruta:

Archivo → Opciones y configuración → Opciones → Configuración regional → Separadores DAX



Figura 3.4 Cambiar separadores DAX.

Te recomendamos que uses el separador que viene por defecto en Power BI.

En nuestro modelo contamos con una tabla *Calendario* la cual explicaremos a detalle más adelante, por el momento vamos a usar este ejemplo de crear una tabla calendario con funciones DAX.

Datos → Inicio → Nueva tabla

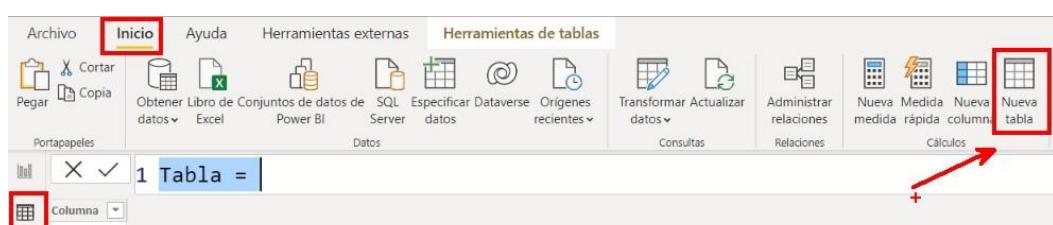


Figura 3.5 Crear una nueva tabla.

De inmediato se habilita la barra de fórmulas con un nombre de Tabla y el signo igual, lo que significa que ya podemos ingresar la función DAX.

El nombre de la tabla es opcional, para nuestro caso se va a llamar *Calendario 2*.

La función que nos devuelve una tabla calendario de manera automática es: *CALENDAR AUTO ()*. Cuando iniciamos a escribir la palabra CALENDAR... DAX nos trae el intellisense o autocompletado de las fórmulas que tienen estos prefijos.

Veamos la figura 3.6

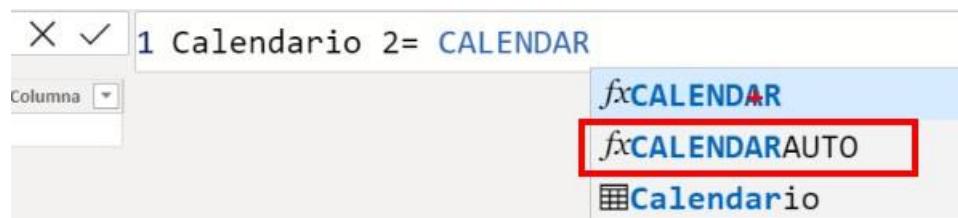


Figura 3.6 Autocompletado de funciones.



Nota: Todas las funciones tienen el prefijo *fx*, lo demás son tablas, variables, medidas etc. que veremos en el camino

Se puede escribir por completo la función o con la función seleccionada presionamos la tecla TAB, por último, cerramos paréntesis y obtenemos un conjunto de fechas.

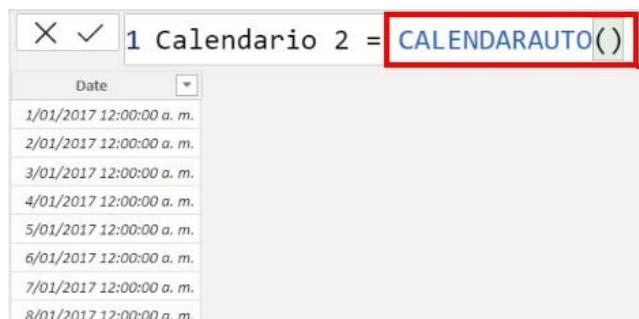


Figura 3.7 Tabla de fechas.

Con esta sencilla fórmula, DAX crea de manera automática un conjunto de fechas, que va desde 1 enero de 2017 hasta el 31 de diciembre de 2021 (con la tecla CTRL y cursor abajo te desplazas hacia la última fecha).

¿Como lo hizo? DAX revisa en todo el modelo de datos los campos que existen como fecha y toma como referencia las fechas mínimas y máximas, pero no necesariamente toma la fecha exacta, solo referencia el año, es decir, si en todo el modelo la fecha mínima es 2/4/2017, DAX toma como fecha o valor mínimo

el 1/1/2017, esto por una razón, para los cálculos de inteligencia de tiempo que veremos en los últimos capítulos, el mismo proceso hace con la fecha máxima, siempre será el 31 de diciembre del año mayor en alguna de las fechas.

CALENDARAUTO también cuenta con un argumento en su sintaxis, y es útil cuando se trabajan con períodos fiscales, si para una compañía el año inicia el 1 de julio, deberías usar este argumento o escribir la función de la siguiente manera:

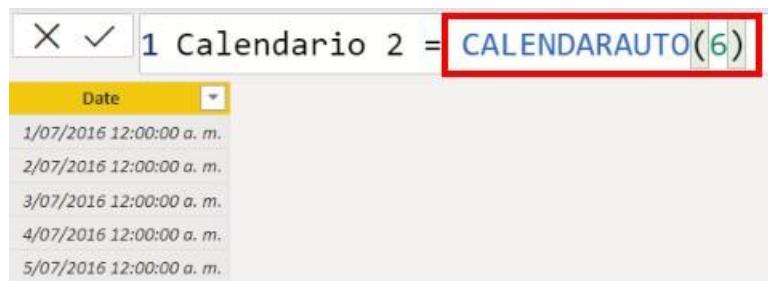


Figura 3.8 Tabla fechas para año fiscal.



Calendario 2 = **CALENDARAUTO (6)**

Observamos que el primer día del año corresponde a 1 de julio de 2017.

Para nuestros casos prácticos trabajaremos con períodos tradicionales.

Este tipo de calendarios son simples de crear, todo modelo de datos debe contar con una tabla calendario por diferentes motivos:

- Relacionar tablas transaccionales
- Crear funciones de inteligencia de tiempo
- Ordenación cronológica para objetos visuales

Que sucede si en nuestro modelo encontramos fechas erróneas, por ejemplo, es muy común encontrar una fecha 1/1/1900, *CALENDARAUTO* devolverá un conjunto de fechas desde 1900, lo cual no sería de agrado en nuestros filtros o segmentadores tener fechas desde allí, podemos crear un conjunto de fechas de manera manual con la función *CALENDAR ()*.

CALENDAR tiene dos argumentos, fecha inicial y fecha final, la cual podemos fijar o quemar en nuestro modelo.

The screenshot shows the DAX Studio interface with the following details:

- Left pane:** A date picker set to "Date" and "Fecha y hora". Below it is a dropdown menu labeled "Estructura".
- Top right:** The function signature **CALENDAR(StartDate, EndDate)**.
- Description:** "Devuelve una tabla con una columna† de todas las fechas entre StartDate y EndDate ."
- Bottom:** The formula **1 Calendario 2 = CALENDAR()**.

Figura 3.9 Argumentos función CALENDAR.

Estos dos argumentos podemos completarlos de manera manual, simplemente diligenciando la fecha de inicio y la fecha final entre comillas.

The screenshot shows the DAX Studio interface with the following details:

- Left pane:** A date picker set to "Date". Below it is a dropdown menu labeled "Estructura".
- Top right:** The formula **1 Calendario 2 = CALENDAR("01/01/2017", "31/12/2020")**.
- Result pane:** A table showing the generated dates from January 1, 2017, to December 31, 2020.

Date
1/01/2017 12:00:00 a. m.
2/01/2017 12:00:00 a. m.
3/01/2017 12:00:00 a. m.
4/01/2017 12:00:00 a. m.

Figura 3.10 Tabla calendario manual.



Calendario 2 = CALENDAR ("01/01/2017", "31/12/2020")

De esta forma tenemos un grupo de fechas quemadas o constante, pero es una manera de segmentarlas para no traer fechas basuras o que nos dañen nuestro modelo y reportes. A medida que vamos avanzando en este viaje de DAX vamos a poder automatizar cada vez más este tipo de cálculos, por ejemplo, que la fecha mínima y máxima dependa de una columna que destinemos en nuestro modelo.

Columnas

Las famosas columnas o columnas calculadas, siempre han sido un tema de discusión para los que estamos en el mundo de los datos en Power BI, la pregunta que surge con frecuencia es si se crean en Power Query o con DAX, la verdad todo depende, aunque este libro es 100% DAX, Power Query es una buena opción para crear columnas calculadas, ya que solo afectan el rendimiento cuando se refrescan o actualizan los datos, pero en DAX el rendimiento se afecta constantemente cuando se itera con el reporte, pero este no será nuestro tema de discusión por el momento, vamos a crear varias columnas como ejemplo, porque debemos entender y visualizar sus resultados para comprender mejor el caso de las medidas.

Cuando creamos una columna con DAX de inmediato se genera un contexto de fila (profundizaremos en el capítulo de evaluación de contextos), cada registro itera para crear un cálculo fila a fila.

Continuando con la tabla calendario, debemos complementarla para extraerle el año, el mes, el trimestre, semana etc.

Nos ubicamos en la pestaña Datos → Seleccionamos tabla Calendario 2 → Herramientas de tabla → Nueva Columna



Figura 3.11 Insertar nueva columna.

Se nos habilita la barra de fórmula y tal cual como nombramos la tabla calendario, seguido del signo igual llamamos la función que nos va a extraer el año de la fecha.

La función es YEAR ()

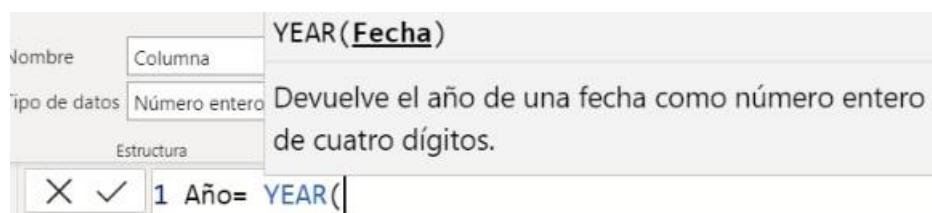


Figura 3.12 Función extraer año.

A diferencia de las fórmulas en Excel, DAX no hace referencia a celdas para extraer el año, DAX tiene una forma muy diferente y es que se debe hacer referencia a tablas y campos, ya que recuerda que son modelos tabulares y esto tiene demasiadas ventajas, ya que trabaja con datos agregados, lo que no hace Excel, ejemplo, en Excel se debe arrastrar las fórmulas (si no se trabajan con tablas) en DAX identifica si hay más registros y siempre se calcula hasta el último dato.

Para extraer el año debemos llamar primero la tabla *Calendario 2* y luego el campo *Date*.

Si la tabla tiene el nombre compuesto, el nombre está encerrado entre comillas simples, se puede identificar las tablas por su ícono, y los campos están entre corchete, como se ve en la imagen siguiente.

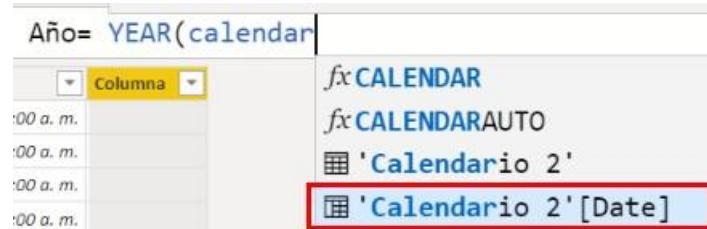


Figura 3.13 Seleccionar tabla y campo.

Seleccionamos la tabla y el campo correspondiente → Cerramos paréntesis → Enter

The screenshot shows the Power BI formula editor with the formula "1 Año = YEAR('Calendario 2'[Date])". The result table has three columns: Date, Año, and Mes Num. The "Año" column contains the value 2017 for all three rows. An arrow points to the "Año" column header.

Date	Año	Mes Num
1/01/2017 12:00:00 a. m.	2017	1
2/01/2017 12:00:00 a. m.	2017	1
3/01/2017 12:00:00 a. m.	2017	1
4/01/2017 12:00:00 a. m.	2017	1

Figura 3.14 Año calculado a partir de una fecha.

Obtenemos el resultado deseado. De esta forma podemos desgranar la columna fecha, ya que es necesaria para los diferentes análisis en diferentes períodos.

Vamos a añadir el mes en número y el trimestre para ver el ejemplo de agregar más columnas.

Con la tabla *Calendario 2* seleccionada → Herramientas de tabla → Nueva Columna

Ingresamos el nombre de la columna “Mes Numero” y la función, que intuitivamente deducimos que sería *MONTH()*, por la secuencia de la columna anterior.

The screenshot shows the Power BI formula editor with the formula "1 Mes Num = MONTH('Calendario 2'[Date])". The result table has three columns: Date, Año, and Mes Num. The "Mes Num" column contains the value 1 for all three rows. An arrow points to the "Mes Num" column header.

Date	Año	Mes Num
1/01/2017 12:00:00 a. m.	2017	1
2/01/2017 12:00:00 a. m.	2017	1
3/01/2017 12:00:00 a. m.	2017	1

Figura 3.15 Calcular el mes número.

De la misma manera que el cálculo anterior, hacemos referencia a la tabla y campo.

Para calcular el trimestre, la función es *QUARTER()*

Con la tabla *Calendario 2* seleccionada → Herramientas de tabla → Nueva Columna

Ver figura 3.11

Figura 3.16 Extraer el trimestre de una fecha.

Podemos completar la tabla con las demás columnas de semana, día mes, día semana, nombre del mes, nombre del día etc.

A continuación, el listado de las funciones para desagregar la columna fecha.



```

Año = YEAR ('Calendario 2'[Date])
Mes Num = MONTH ('Calendario 2'[Date])
Trimestre = QUARTER ('Calendario 2'[Date])
Mes = FORMAT ('Calendario 2'[Date], "mmmm")
Semana = WEEKNUM ('Calendario 2'[Date], 2)
Día Mes = DAY ('Calendario 2'[Date])
Nombre Día = FORMAT ('Calendario 2'[Date], "dddd")
    
```

Así se ve la tabla *Calendario 2* en la figura 3.17

Date	Año	Mes Num	Trimestre	Mes	Semana	Día Mes	Nombre Día
1/01/2017 12:00:00 a. m.	2017	1	1	enero	1	1	domingo
2/01/2017 12:00:00 a. m.	2017	1	1	enero	2	2	lunes
3/01/2017 12:00:00 a. m.	2017	1	1	enero	2	3	martes
4/01/2017 12:00:00 a. m.	2017	1	1	enero	2	4	miércoles
5/01/2017 12:00:00 a. m.	2017	1	1	enero	2	5	jueves
6/01/2017 12:00:00 a. m.	2017	1	1	enero	2	6	viernes

Figura 3.17 Tabla calendario.

No solo podemos crear columnas para una tabla calendario, podemos crear columnas que involucren operaciones aritméticas, por ejemplo, vamos a crear una columna llamada Ingresos en la tabla *Ventas*, multiplicamos Precio Unit. por cantidad.

Seleccionamos la tabla *Ventas* → Herramientas de tablas → Nueva columna

The screenshot shows the Microsoft Power BI desktop interface. The top navigation bar includes 'Archivo', 'Inicio', 'Ayuda', 'Herramientas externas', and 'Herramientas de tablas'. The 'Herramientas de tablas' tab is active, indicated by a red box. Below it, there are several icons: 'Marcar como tabla de fechas' (highlighted in red), 'Administrador relaciones', 'Relaciones', 'Nueva medida rápida' (highlighted in red), 'Nueva columna' (highlighted in red), and 'Cálculos'. A red arrow points from the text 'Figura 3.18 Crear nueva columna en tabla ventas.' to the 'Nueva columna' icon. The main area displays a table titled 'Ventas' with columns: Fecha Factura, Nombre Cliente, Cod. Vendedor, Cod.Producto, Cantidad, Precio Unit, and a calculated column 'Ingresos'.

Figura 3.18 Crear nueva columna en tabla ventas.

La barra de fórmulas se habilita para ingresar el cálculo correspondiente.

Llamamos el cálculo *Ingresos*, hacemos el llamado a los campos Precio Unit y Cantidad, recuerda seleccionar cada campo con el nombre de la tabla.

The screenshot shows the Power BI desktop interface with the formula bar at the top containing the formula 'Ingresos = Ventas[Precio Unit]*Ventas[Cantidad]'. A red box highlights this formula. Below the formula bar is a table with columns: Fecha, Nombre Cliente, Cod. Vendedor, Cod.Producto, Cantidad, Precio Unit, CostoUnit, Cod Lineas, Documento, Cod Municipio, Fecha Pedido, Fecha Entrega, and Ingresos. The 'Ingresos' column is highlighted with a red box. A red arrow points from the formula bar to the 'Ingresos' column header in the table.

Figura 3.19 Columna Ingresos.



Advertencia: Este tipo de columnas afectan el rendimiento en el modelo de datos, pero es necesario crearlas para comprender las medidas, no es recomendable crear estas columnas calculadas en los modelos de campo.

Po último y para ejercicios posteriores vamos a crear una columna calculada llamada “Costos”, que es la multiplicación de *Cantidad * Costo Unit*.

El resultado en la siguiente imagen.

The screenshot shows the Power BI desktop interface with the formula bar at the top containing the formula '. Costos = Ventas[Cantidad]*Ventas[CostoUnit]'. A red box highlights this formula. Below the formula bar is a table with columns: Fecha, Nombre Cliente, Cod. Vendedor, Cod.Producto, Cantidad, Precio Unit, CostoUnit, Cod Lineas, Documento, Cod Municipio, Fecha Pedido, Fecha Entrega, Ingresos, and Costos. The 'Costos' column is highlighted with a red box. A red arrow points from the formula bar to the 'Costos' column header in the table.

Figura 3.20 Columna Costos.

¿Cuándo crear columnas?

Esta es una pregunta muy común como lo hablamos al inicio de esta sesión, pero seremos lo más claro posible, debemos crear columnas siempre que debamos segmentar la información, por ejemplo, extraer el año de la fecha, es obligatorio crear una columna para esto, lo mismo sucede si necesitamos por ejemplo clasificar los precios de ventas en bajos, medio y altos, es necesario crear una columna, pero si la tarea es crear un cálculo aritmético esto se debe hacer con medidas, lo veremos posteriormente.

Medidas

Las medidas como las columnas calculadas tienen en común que al final son cálculos aritméticos, la gran diferencia está en el rendimiento, cuando creamos columnas siempre se almacenan en memoria del modelo de datos, las medidas no se almacenan en el modelo de datos, solo consume memoria a la hora de ejecutarse la consulta, por ejemplo, cuando se arroja una medida a una tabla con el campo año, esto es una consulta, las medidas se usan para calcular valores agregados.

Hay dos tipos de medidas (implícitas y explícitas), abundaremos en esto en los siguientes textos, por el momento veremos que es una medida en Power BI.

Con tabla *Ventas* seleccionada → Inicio → grupo cálculos → Nueva medida

The screenshot shows the Power BI ribbon with the 'Inicio' tab selected. Below the ribbon, a table named 'Ventas' is displayed with columns: Fecha Factura, Nombre Cliente, Cod. Vendedor, Cod.Producto, Cantidad, Precio Unit, CostoUnit, Cod Lineas, and Documento. The 'Cálculos' tab is highlighted with a red box, and the 'Nueva medida' button is also highlighted with a red box. A red arrow points from the text 'Crear nueva medida.' to the 'Nueva medida' button.

Fecha Factura	Nombre Cliente	Cod. Vendedor	Cod.Producto	Cantidad	Precio Unit	CostoUnit	Cod Lineas	Documento
martes, 6 de marzo de 2018	QUALA S.A.	B-01	2303	20	25320	16254	AL - 01	6977376
iernes, 9 de marzo de 2018	QUALA S.A.	B-01	2303	48	25320	16270	AL - 01	1378167
eves, 15 de marzo de 2018	QUALA S.A.	B-01	2303	140	25320	16275	AL - 01	7078105
sdes, 14 de febrero de 2018	QUALA S.A.	B-01	2303	400	25320	16178	AL - 01	0375155

Figura 3.21 Crear nueva medida.

Se nos habilita la barra de fórmula para ingresar el cálculo correspondiente.

Nombramos la medida como *Ventas*, vamos a sumar la columna que creamos como *Ingresos*.

Tal como hemos indicado anteriormente, las funciones son en inglés y se debe hacer el llamado de la tabla y el campo, ver el resultado en la siguiente imagen

The screenshot shows a Power BI interface. On the left is a table with columns: Fecha Factura, Nombre Cliente, Cod. Vendedor, Cod. Producto, and Cantidad. A red box highlights the formula bar at the top, which contains the DAX formula: `. Ventas = SUM(Ventas[Ingresos])`. To the right is a 'Campos' (Fields) pane with a search bar. A red arrow points from the formula bar down to the 'Ventas' entry in the list of fields, which is also highlighted with a red box.

Figura 3.22 Medida ventas.

Si damos Enter la medida la podemos identificar con el ícono de una calculadora y se guarda en la tabla donde estábamos ubicados, en este caso ventas, pero no vemos ningún resultado, ¿dónde está la suma de los ingresos?, esta es la gran diferencia de las medidas, que solo se ejecutan en una consulta, en palabras coloquiales cuando son consumidas por un objeto visual o llamadas desde una tabla o columna.

En la figura 3.20 vemos un ícono llamado *Medida rápida*, son medidas que se crean fácilmente, para evitar crear nosotros el código, pero no lo recomendamos ya que si aún no dominamos DAX sus resultados no son tan comprensibles a simple vista, paso a paso vamos a crear medidas para entender todo el panorama.

Hay dos tipos de medidas que podemos usar en nuestros reportes: implícitas y explícitas.

Medidas Implícitas

Las medidas implícitas son las que podemos usar directamente en el lienzo arrastrando a cualquier objeto visual, son cálculos que Power BI trae por defecto, implícitamente, pero que no es recomendable usar y esto lo veremos a detalle los pro y contra de usar medidas implícitas.

Procedamos de que trata esto, vamos al lienzo de Power BI o pestaña informe y demos clic en el objeto visual matriz.

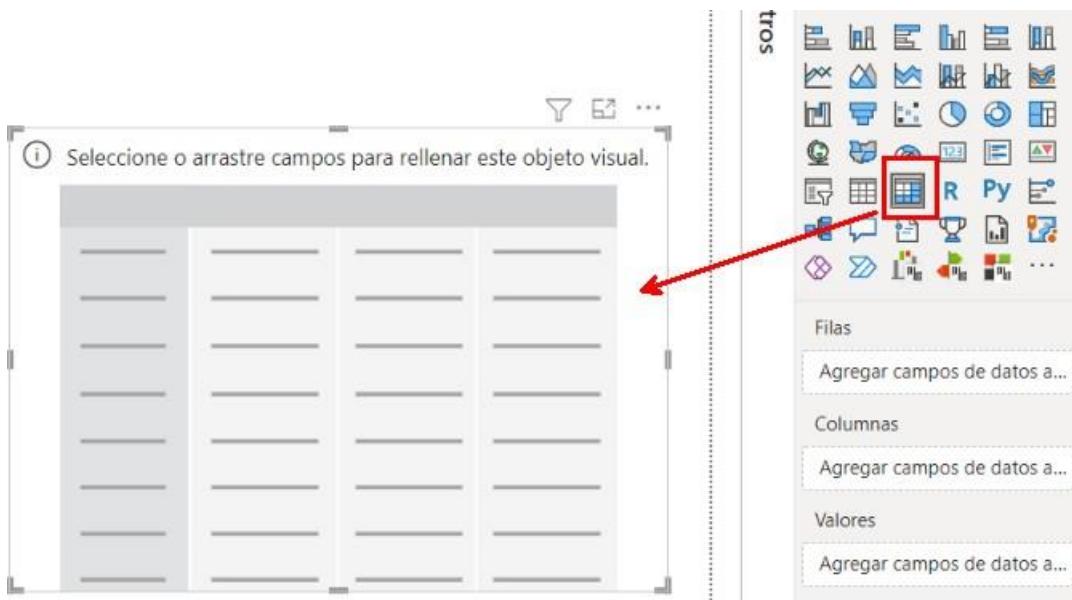


Figura 3.23 Insertar objeto visual matriz.

Al campo filas vamos a llevar la columna *Año* de la tabla *Calendario* (Para los ejemplos vamos a usar los campos de la tabla *Calendario* la cual fue creada y suministrada para este modelo de datos) y en el campo valores, arrojamos la columna *Ingresos*.

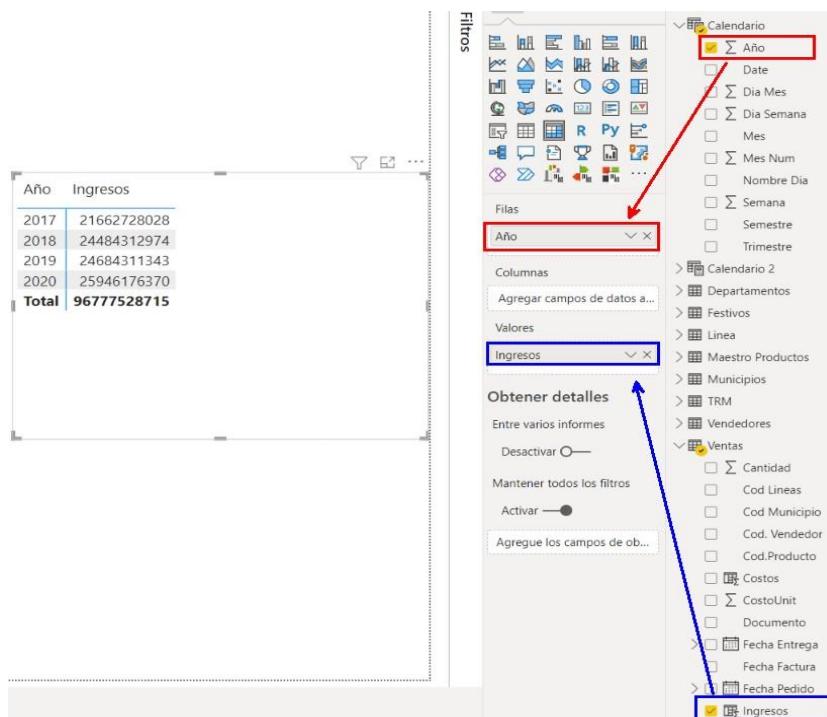


Figura 3.24 Ingresos por año.

De manera automática, Power BI consolida los ingresos por año en la matriz, ¿pero cuando le indicamos que hiciera esta SUMA?, si la columna ingresos fue simplemente una multiplicación de las cantidades por el precio unitario, es decir, que implícitamente se creó este cálculo.

Veamos que tipos de cálculos trae por defecto las medidas implícitas.

En el campo *Ingresos* desplegamos las opciones que nos brinda este tipo de medidas.

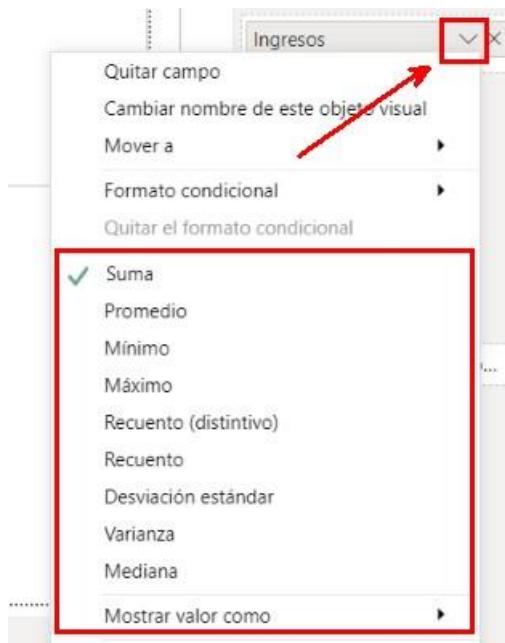


Figura 3.25 Operaciones medidas implícitas.

Por defecto vemos que Suma es la operación que realiza, pero podemos escoger un promedio, mínimo, máximo y hasta indicarle que nos dé el porcentaje de participación por año (Mostrar valor como).

La única ventaja de usar este tipo de medidas es su sencillez y la manera rápida de ver resultados, pero son más las desventajas que tiene, veamos.

Llevemos el campo *Costos* a la matriz

The screenshot shows the Power BI matrix editor. On the left is a table with columns 'Año', 'Ingresos', and 'Costos'. The data rows show values for 2017, 2018, 2019, 2020, and a total row. In the center, there is a matrix structure with columns for 'Año', 'Ingresos', and 'Costos'. On the right, there are two panes: 'Columns' and 'Values'. The 'Values' pane contains fields 'Ingresos' and 'Costos', with 'Costos' being the selected field, indicated by a red box and a red arrow pointing to it from the caption.

Figura 3.26 Llevar campo costos a la matriz.

Contamos con dos campos que suman los ingresos y costos por año, que tal si deseamos calcular la utilidad por año.

Vamos a la tabla *Ventas* → Clic derecho nueva medida (Otra manera de insertar medidas)

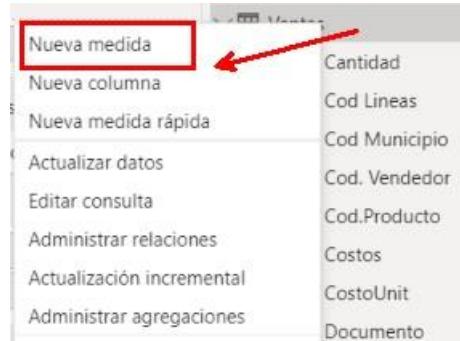


Figura 3.27 Insertar nueva medida.

En la parte superior derecha se habilita la barra de fórmula → Nombramos la medida como *Utilidad* y hacemos referencia a la resta de *Ingresos - Costos*

Pero sucede algo inusual, cuando iniciamos a escribir la palabra *Ingresos*, en el autocompletado no aparece ni el nombre de la tabla (*Ventas*) ni el campo *ingresos*.



Figura 3.28 Intellisense deshabilitado.

Porque cuando creamos una medida y llamamos directamente la columna no aparece en el intellisense, contrario cuando creamos una columna, que el autocompletado lo reconoce.

Queremos resaltar esto y es uno de los temas que le cuesta comprender a cada usuario de DAX, recuerda que las medidas se usan para calcular valores agregados, es decir, para que podamos escoger o ver la tabla y el campo *ingresos* en la barra de fórmulas, necesitamos ingresar una función que nos agregue dicho cálculo.

Que sucede si forzamos este cálculo, vamos a escribir manualmente la tabla *Ventas [Ingresos] - Ventas[costos]* → Enter

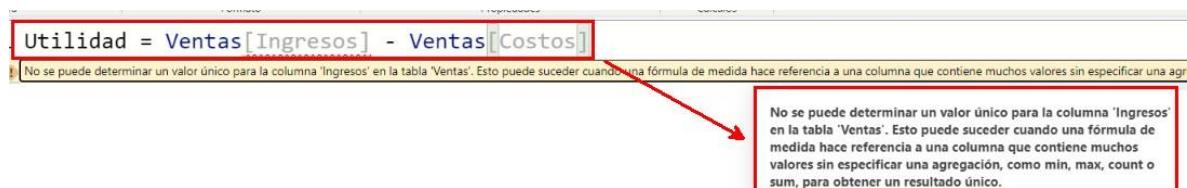


Figura 3.29 Error al llamar el campo *ingresos* y *costos* en una medida.

Lea detenidamente el error, DAX no puede arrojar un valor único realizando esta operación directa entre columnas, necesita de una función para hacerlo, al final el objetivo de cada medida es un valor escalar.

Y otra razón es que una columna tiene múltiples valores, lo cual necesita de una función de agregación para arrojar un único registro.

Esta es una de las grandes y mayores desventajas de usar medidas implícitas, que no podemos hacer uso de ellas, otra gran desventaja es que, si solo nos dedicamos a crear reportes con este tipo de medidas, vamos a caer en la probabilidad de tener que repetir cada gráfico, ya que, si el nombre de la tabla o el campo cambian, todo el reporte se dañará, por ejemplo, si tenemos un reporte con *Ventas [Ingresos]* y la columna cambia a *Ventas [Ventas]* tocar ir gráfico por gráfico reajustar el tablero.



Advertencia: Tener claro que las medidas implícitas es una de las peores metodologías que podemos usar a la hora de crear reportes, tienes más desventajas que ventajas, siempre usa medidas explícitas

De esta forma no podemos crear una medida para la *Utilidad*, debemos recurrir a las medidas explícitas.

Medidas explícitas

Las medidas explícitas o conocidas como medidas manuales es la que crea cada usuario desarrollador, son las que directamente creamos en la barra de fórmulas con DAX y las vamos guardando en una tabla "X".

Recordamos que estas medidas solo consumen memoria a la hora de ejecutarse las consultas o cuando se usan en algún objeto visual, vamos a destinar solo un capítulo para explicar las diferentes categorías o principales medidas en DAX, por el momento debemos solucionar el caso anterior, como calcular la *utilidad*.

Regresamos a Power BI, y lo que debemos hacer es encerrar *Ventas [Ingresos]* en SUM lo mismo para *Ventas[costos]* y hacer la operación.

1 Utilidad = `SUM(Ventas[Ingresos]) - SUM(Ventas[Costos])`

Figura 3.30 Medida Utilidad.

Si presionamos Enter, ya no arroja error dicha medida, preste mucha atención al cálculo, primero se realiza una suma para los ingresos menos la suma para los costos, ya que la función SUM solo acepta una única columna como argumento.

Por último, llevamos la medida la matriz.

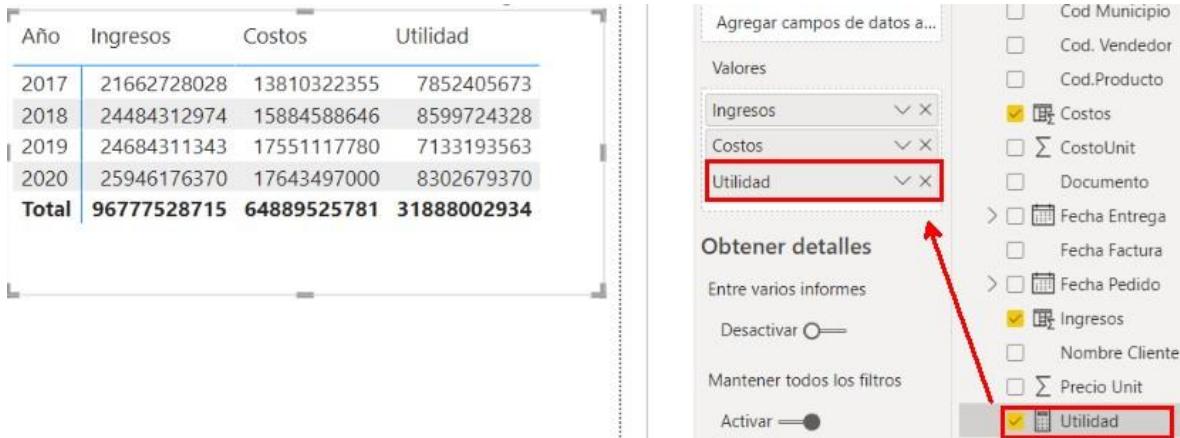


Figura 3.31 Medida Utilidad en la matriz.

Efectivamente el cálculo ya está realizado, la mejor metodología que debemos aprender para el buen manejo de DAX es crear todas las medidas de los campos numéricos como son: Cantidad, Ventas, Costos, Precio Unitario etc.

Para crear la medida *Utilidad* incurrimos en la resta de dos sumas, que sucede si necesitamos usar la medida *Ventas* para crear otro cálculo, nos tocaría volver a realizar la SUM (Ventas [*Ingresos*]), lo mismo en el caso de los costos, aprovechamos las ventajas de las medidas explícitas y su rendimiento.

Al inicio de esta sesión creamos una medida llamada [*Ventas*], llevémosla a nuestro informe por año.

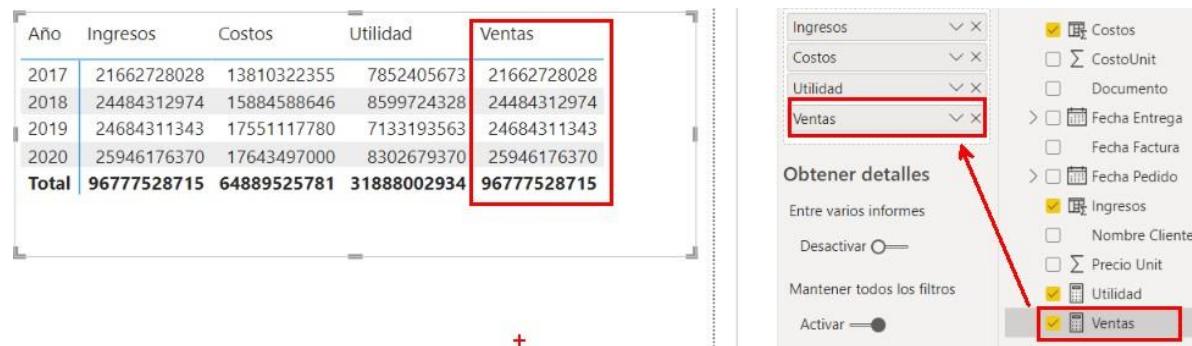


Figura 3.32 Medida Ventas en la matriz.

Observamos que es el mismo resultado del campo *Ingresos* (Medida implícita), seguido vamos a crear una medida llamada [*Costos Totales*]= SUM (*Ventas [Costos]*)

Clic derecho en la tabla Ventas → Nueva Medida → Ingresamos el siguiente cálculo

```
1 Costos Totales = SUM(Ventas[Costos])
```

Figura 3.33 Medida costos totales.

Seguido la llevamos a nuestra matriz

The screenshot shows a matrix table with columns: Año, Ingresos, Costos, Utilidad, Ventas, and Costos Totales. The last column is highlighted with a red box. To the right is a context pane with sections for Ingresos, Costos, Utilidad, Ventas, and Costos Totales. The Costos Totales section is also highlighted with a red box. A dropdown menu is open over the Costos Totales section, showing options like Costos, Costos Totales, and other measures.

Figura 3.34 Medida costos totales en la matriz.

Genera el mismo resultado de la medida implícita de *Costos*. Aunque sea un proceso repetitivo es la mejor práctica que debes adoptar en tus desarrollos, crear todas las medidas de manera explícita.

La gran ventaja la veremos si creamos una nueva medida en donde va a estar involucrada *Ventas* y *Costos*, por ejemplo, creamos una medida llamada *[Beneficio]*, sería la resta entre la medida *[Ventas]* – *[Costos]*

Clic derecho en la tabla *Ventas* → Nueva Medida → Beneficio =

The screenshot shows the 'New Measure' dialog with the formula '1 Beneficio= venta'. Below it is a list of three measures: 'Ventas[Costos Totales]', 'Ventas[Utilidad]', and 'Ventas[Ventas]'. The 'Ventas[Ventas]' option is highlighted with a red box.

Figura 3.35 Inicio medida Beneficio.

Cuando iniciamos con el cálculo el autocomplete reconoce de inmediato las medidas, pero antepone el nombre de la tabla donde se encuentran estas medidas, las medidas no pertenecen o no deben pertenecer a ninguna tabla, se guardan allí pero no debemos permitir el nombre de la tabla como prefijo, el capítulo siguiente veremos una forma ordenada de guardar cada medida.



Nota: Las medidas no hacen parte de ninguna tabla

Para evitar seleccionar la tabla, antes de escribir el nombre de una medida iniciamos con el “*[“corchete*, y el nombre de la tabla desaparece.

The screenshot shows the 'New Measure' dialog with the formula 'Beneficio=['. A red circle highlights the opening bracket '['. Below it is a list of three measures: '[Costos Totales]', '[Utilidad]', and '[Ventas]'. The '[Ventas]' option is highlighted with a red box.

Figura 3.36 Llamar medidas con el corchete.

La imagen muestra solo y únicamente las medidas disponibles en nuestro modelo, procedemos realizar el cálculo.

$$1 \text{ Beneficio} = [\text{Ventas}] - [\text{Costos Totales}]$$

Figura 3.37 Medida Beneficio.

Si llevamos esta medida a la matriz que estamos creando el resultado debe ser igual al de utilidad.

Año	Ingresos	Costos	Utilidad	Ventas	Costos Totales	Beneficio
2017	21662728028	13810322355	7852405673	21662728028	13810322355	7852405673
2018	24484312974	15884588646	8599724328	24484312974	15884588646	8599724328
2019	24684311343	17551117780	7133193563	24684311343	17551117780	7133193563
2020	25946176370	17643497000	8302679370	25946176370	17643497000	8302679370
Total	96777528715	64889525781	31888002934	96777528715	64889525781	31888002934

Ingresos

Costos

Utilidad

Ventas

Costos Totales

Beneficio

Figura 3.38 Medida Beneficio en la matriz.

El resultado es exactamente igual, pero la gran ventaja es que ya contamos con las medidas *Ventas* y *Costos Totales* y las podemos usar en cualquier otro cálculo.

Capítulo 4: Funciones

Las funciones DAX vienen en diferentes categorías: agregación, fecha y hora, texto, filtro, financieras, lógicas etc. En este capítulo vamos a profundizar en las funciones más usadas y relevantes que debes dominar para iniciar en el mundo de inteligencia de negocios con Power BI.

Las medidas como lo hablamos en el capítulo anterior es el resumen de lo que es DAX, en este punto se resumen las tablas y columnas convertidas en una medida y su objetivo es el resultado de un único valor presentado en un objeto visual. Con las medidas vemos los principales indicadores, KPI's y reglas de negocio concisas en un tablero o Dashboards listos para dar información para la toma de decisiones.



Referencia: En el siguiente enlace puedes ver todas las categorías del lenguaje DAX presentado por Microsoft: <https://docs.microsoft.com/pt-br/dax/>

Todas las medidas que vamos a usar van a hacer de manera explícita, vamos a crear una a una explicando sus bondades, usos y aplicaciones.

Funciones de agregación

Las funciones de agregación son aquellas que nos permiten agrupar un resultado devolviendo un único valor analizado por diferentes campos de una base de datos.

Para crear medidas con este tipo de funciones en DAX, debemos primero validar el tipo de dato de las columnas que se van a agregar, esto es de suma importancia ya que deben estar como número entero, número decimal o número decimal fijo.

Por ejemplo, si seleccionamos la columna *Cantidad* de la tabla *Ventas*, se habilita herramientas de columnas y allí podemos ver qué tipo de dato tiene asignado.

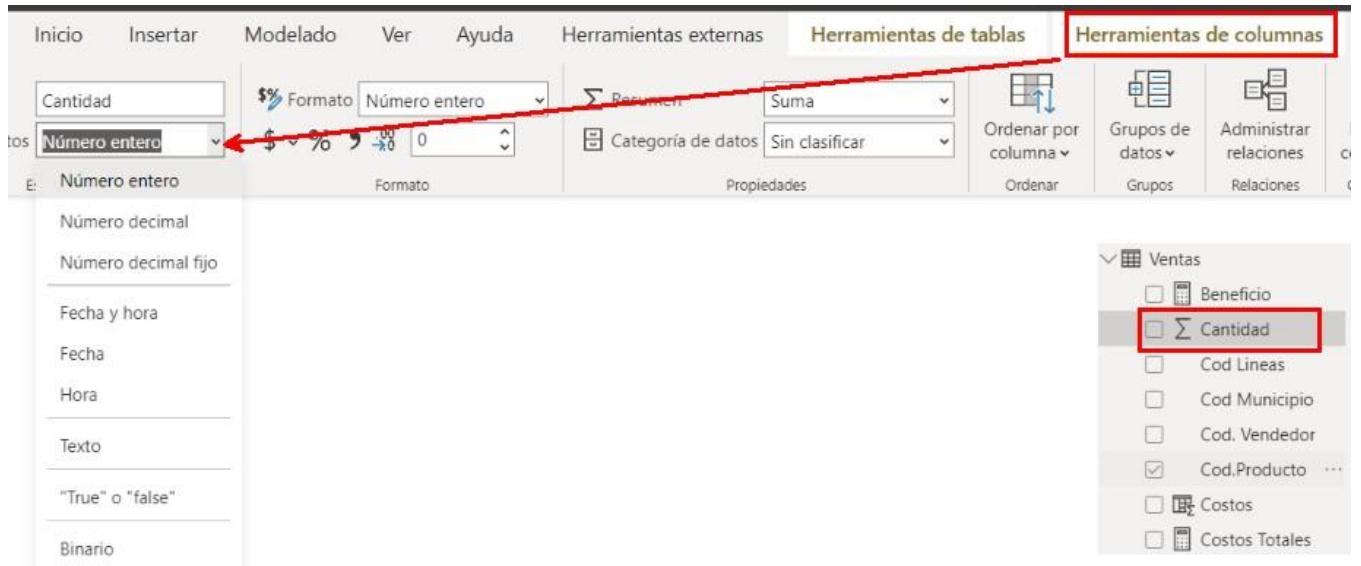


Figura 4.1 Validar tipo de dato.

Luego de haber validado el tipo de dato de las columnas especialmente numéricas debemos clasificar en diferentes tablas las principales funciones de DAX, esta metodología también se debe adoptar en los proyectos que realices, pues te dará un orden específico y vas a encontrar de manera rápida tus cálculos.

Clasificación de medidas

Esto es una metodología que se debe seguir, si queremos crear modelos ordenados, y es clasificar en cada tabla los tipos de medidas que se van creando.

Vamos a la pestaña Inicio → Datos → Especificar datos →



Figura 4.2 Crear tabla medidas.

Nombramos la tabla como *Medidas Agregación* y Cargar.

Se deja en blanco la columna1.

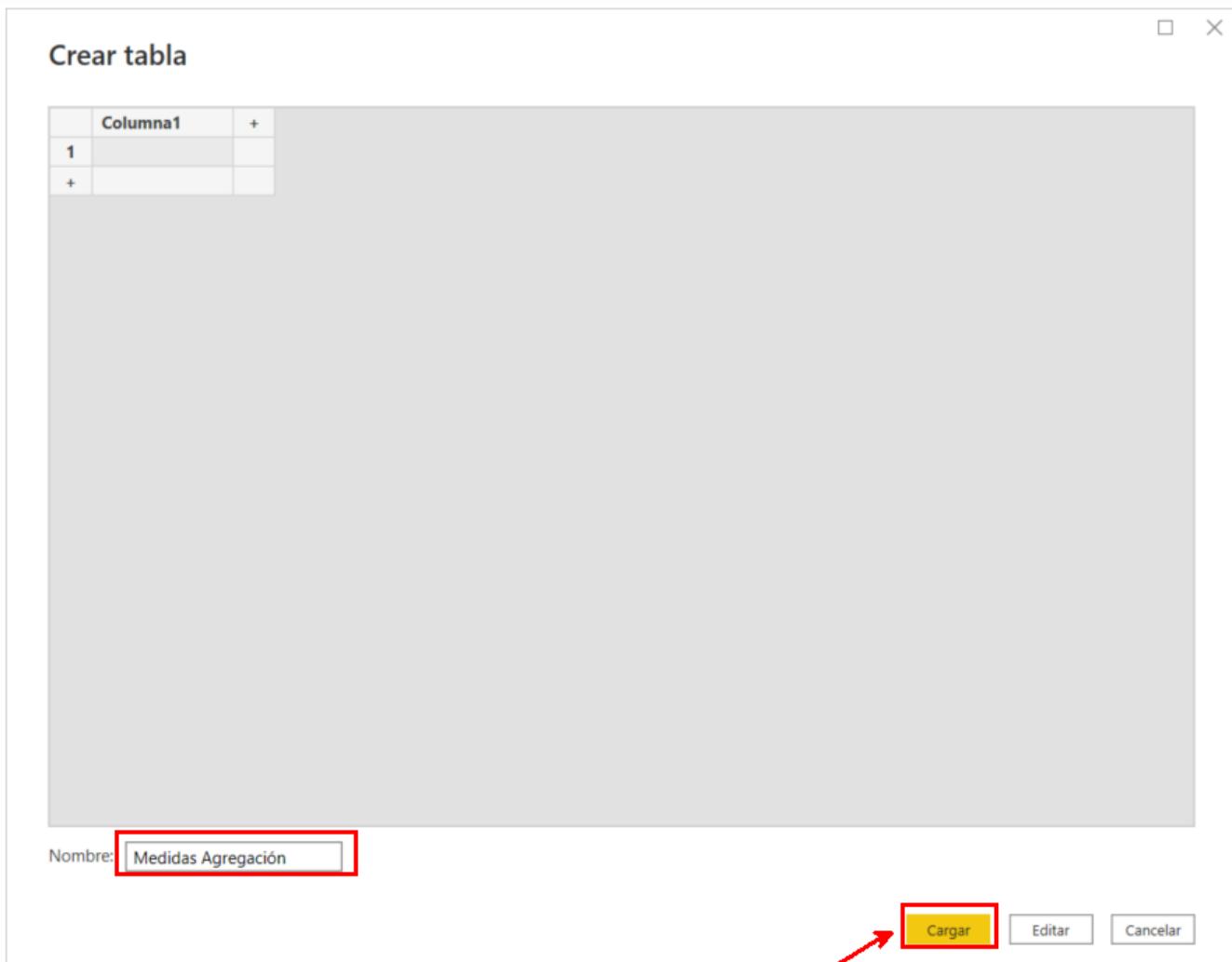


Figura 4.3 Nombrar la tabla medidas.

Power BI agrega esta tabla como si hiciera parte del modelo, pero es una tabla vacía.

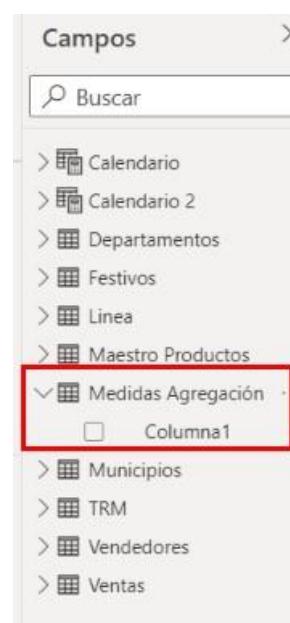


Figura 4.4 Tabla agregada.

El siguiente paso es llevar las medidas que hemos creado con anterioridad a esta tabla.

Movamos la medida [Ventas] a tabla *Medidas Agregación*.

Seleccionamos la medida [Ventas] → Herramienta de medición → Tabla Inicial → Seleccionamos la tabla a mover.

The screenshot shows the Power BI ribbon with the 'Herramientas de medición' tab highlighted. In the 'Tabla inicial' dropdown, 'Ventas' is selected. A red arrow points from the 'Medidas Agregación' table in the dropdown to the 'Herramientas de medición' tab. Another red arrow points from the 'Ventas' measure in the table to the 'Nueva medida rápida' button in the ribbon.

Figura 4.5 Mover medida de tabla.

Si volvemos a la tabla *Medidas Agregación*, dicha medida se incorporó ya.

Pero este no es el verdadero truco que queremos, el objetivo es que esta tabla quede marcada con el ícono de una tabla medida y siempre se encuentre en la parte superior de los campos.

Seleccionamos la tabla *Medidas Agregación* → Clic derecho en Columna1 → Eliminar del modelo

The screenshot shows the context menu for 'Columna1' in the 'Medidas Agregación' table. The 'Eliminar del modelo' option is highlighted with a red box.

Figura 4.6 Eliminar column1.

Si volvemos a revisar nuestros campos, esta tabla se ubicó en la parte superior y tiene un ícono que la diferencia de las demás tablas.

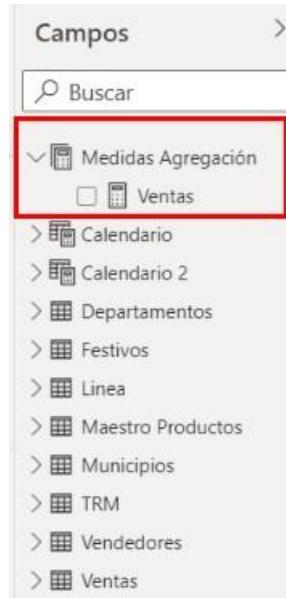


Figura 4.7 Tabla medidas agregación ordenada.

Este mismo proceso debemos ejecutarlo cada que se quiera crear diferentes tipos de funciones.

Vamos a mover las medidas de [Costos Totales], [Utilidad] y [Beneficio] a esta nueva tabla. Ver figura 4.5.

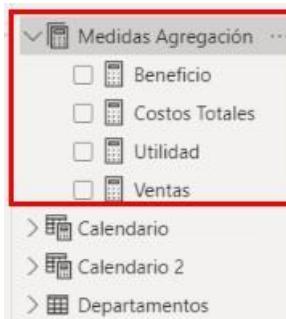


Figura 4.8 Medidas de agregación.

Estas medidas son funciones de SUMA que resumen los datos por año según el informe creado anteriormente.

Medidas con formato

Una siguiente práctica que te recomendamos es el formato a cada una de las medidas, por ejemplo, la medida [Ventas] vamos a ponerle separadores de miles.

Seleccionamos la medida → Herramientas de medición → Formato → Separadores de miles
58

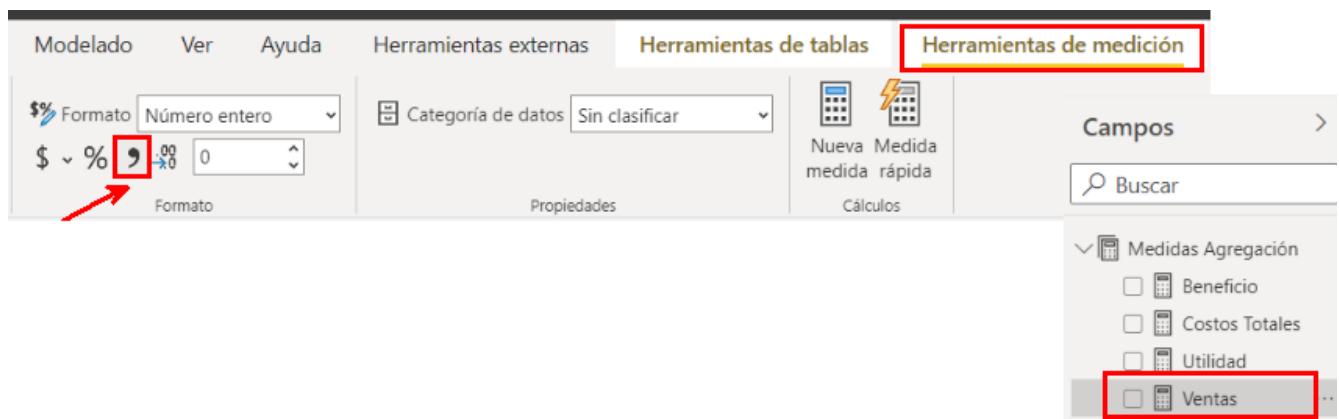


Figura 4.9 Asignar separadores de miles a Ventas.

Este mismo proceso lo repetimos para las demás medidas, también puedes poner tipo de moneda, porcentajes y demás.

Por último, ajustamos nuestra matriz, dejando solo las medidas de: *[Ventas]*, *[Costos Totales]* y *[Utilidad]*, tu informe se debe ver la siguiente manera.

Año	Ventas	Costos Totales	Utilidad
2017	21.662.728.028	13.810.322.355	7.852.405.673
2018	24.484.312.974	15.884.588.646	8.599.724.328
2019	24.684.311.343	17.551.117.780	7.133.193.563
2020	25.946.176.370	17.643.497.000	8.302.679.370
Total	96.777.528.715	64.889.525.781	31.888.002.934

Figura 4.10 Ventas. Costos y Utilidad por año.

SUM

La figura 4.10 muestra un claro resumen de tres cálculos agrupados por año, la función SUM (Suma) es una función que solo tiene en cuenta el contexto de filtro, pero no el contexto de fila, los contextos de evaluación son la clave de todo en DAX, por eso no te apresures ni te preocupes si aún no los comprendes, hemos separado un capítulo entero para esto.

Esta es una de las funciones más usadas en DAX, solo contiene un único argumento y es una columna.

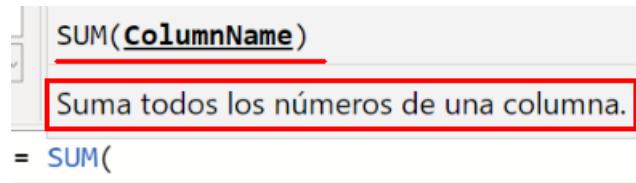


Figura 4.11 Argumento función SUM.

Creemos la medida `[Cantidad]`, para completar la agrupación de los principales campos (ventas, costos, cantidad).

1 Cantidad = `SUM(Ventas[Cantidad])`

Figura 4.12 Medida cantidad.

Si llevamos esta medida a nuestra matriz, el resultado es la suma de las cantidades por año, la pregunta que surge es: ¿Cuándo le indicamos a DAX que nos sumara las cantidades por año?

Año	Ventas	Costos Totales	Utilidad	Cantidad
2017	21.662.728.028	13.810.322.355	7.852.405.673	3.169.951
2018	24.484.312.974	15.884.588.646	8.599.724.328	3.715.929
2019	24.684.311.343	17.551.117.780	7.133.193.563	4.545.938
2020	25.946.176.370	17.643.497.000	8.302.679.370	4.119.719
Total	96.777.528.715	64.889.525.781	31.888.002.934	15.551.537

Figura 4.13 Cantidad en la matriz.

En ningún momento la fórmula dice que nos sumara las cantidades por año, esto sucede porque el campo año funciona como un filtro, pero año hace parte de la tabla calendario, la cuál por medio del campo fecha está relacionada con la tabla Ventas y el filtro se propaga hacia esta. La siguiente imagen nos ilustra mejor esta operación detrás de cámaras. Analicemos el resultado de 2019.

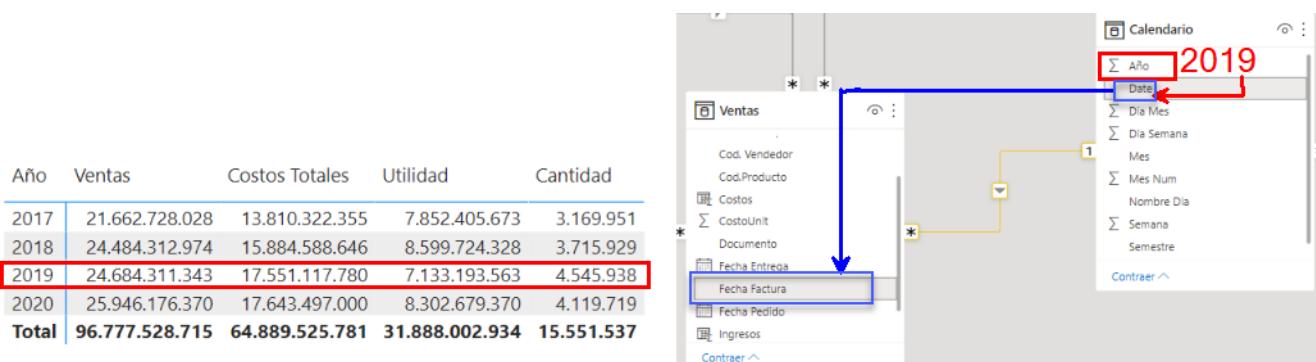


Figura 4.14 Propagación de filtro.

La imagen ilustra claramente el proceso del resultado del cálculo, en la tabla calendario se filtra el año 2019, ya está tabla solo queda con las fechas de este año, por medio de la relación estas fechas viajan hacia la tabla Ventas, Ventas ya está internamente contraída y solo suma las cantidades de las fechas 2019, siempre ten en cuenta esta operación, la cual es conocida como propagación de filtro.

AVERAGE

AVERAGE, da como resultado el promedio de un conjunto de datos, hace parte de esta familia de funciones de agregación, como cualquier otro promedio toma la suma de datos y lo divide sobre el número de registros, esta función solo admite un único argumento.

Calculemos el promedio de cantidades por año.

Clic derecho en la tabla medidas de agregación → Nueva medida

1 Prom Cantidad= AVERAGE(Ventas[Cantidad])

Figura 4.15 Promedio cantidades.

Si llevamos esta medida a nuestra matriz, se verá así:

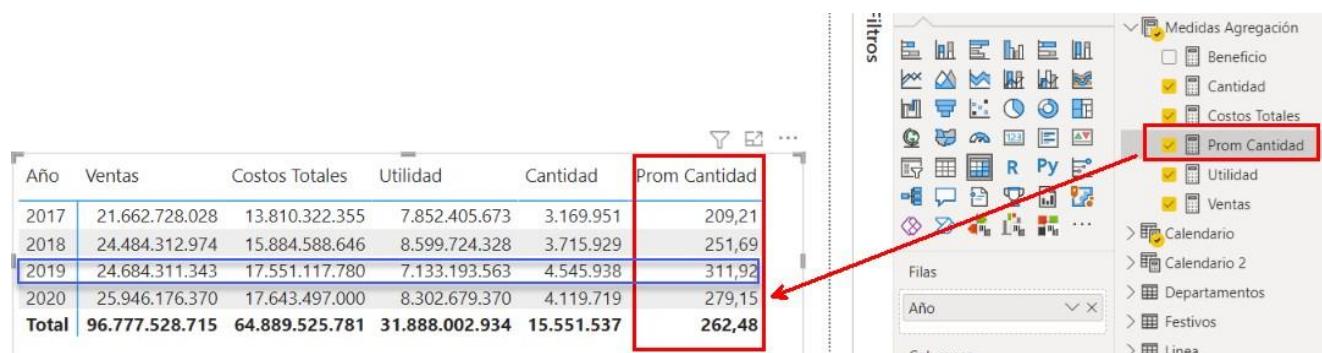


Figura 4.16 Medida promedio en la matriz.

Si analizamos 2019, significa que las cantidades promedio vendidas para este año es 311,92. Si conocemos nuestros datos es un resultado raro, pues el total acumulado para 2019 es 4.545.938, recuerda que es un promedio, es el total dividido el número de registros, es decir, 4.545.938 sobre el total de movimientos que hubo en 2019.



Advertencia: Si buscamos crear promedios para datos agrupados este no es el camino, ya que la tabla ventas está a nivel de días, en capítulos posteriores analizaremos como crear promedios con datos agrupados

Si queremos comprobar el resultado de esta fórmula, veamos la siguiente función.

COUNTROWS

Cuenta el número de registros o filas de una tabla en el contexto donde se evalúe la medida, es de las únicas funciones que acepta como argumento una tabla y arroja un valor escalar o único valor.

Por ejemplo, contemos el número de filas de la tabla *Ventas* por año.

Clic derecho en la tabla medidas de agregación → Nueva medida → Ingresamos la siguiente función

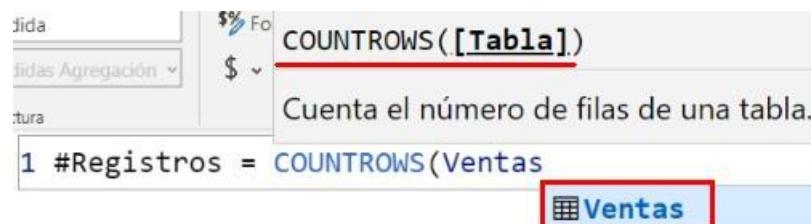


Figura 4.17 Medida número de registros tabla ventas.

Si llevamos esta medida a nuestra matriz, el resultado es el siguiente; no olvides asignar separadores de miles.

Año	Ventas	Costos Totales	Utilidad	Cantidad	Prom Cantidad	#Registros
2017	21.662.728.028	13.810.322.355	7.852.405.673	3.169.951	209,21	15.152
2018	24.484.312.974	15.884.588.646	8.599.724.328	3.715.929	251,69	14.764
2019	24.684.311.343	17.551.117.780	7.133.193.563	4.545.938	311,92	14.574
2020	25.946.176.370	17.643.497.000	8.302.679.370	4.119.719	279,15	14.758
Total	96.777.528.715	64.889.525.781	31.888.002.934	15.551.537	262,48	59.248

Figura 4.18 Medida número de registros en la matriz.

2019 se asentaron 14.574 registros, si dividimos $[Cantidad]/[#Registros] = 311,92$. Comprobemos, creamos esta medida.

✓	1 Prom2_Cantidad = [Cantidad] / [#Registros]						
Año	Ventas	Costos Totales	Utilidad	Cantidad	Prom Cantidad	#Registros	Prom2_Cantidad
2017	21.662.728.028	13.810.322.355	7.852.405.673	3.169.951	209,21	15.152	209,21
2018	24.484.312.974	15.884.588.646	8.599.724.328	3.715.929	251,69	14.764	251,69
2019	24.684.311.343	17.551.117.780	7.133.193.563	4.545.938	311,92	14.574	311,92
2020	25.946.176.370	17.643.497.000	8.302.679.370	4.119.719	279,15	14.758	279,15
Total	96.777.528.715	64.889.525.781	31.888.002.934	15.551.537	262,48	59.248	262,48

Figura 4.18 Medida promedio 2 cantidades.

Claramente el resultado es el mismo calculado con AVERAGE o con COUNTROWS, hemos comprobado cómo funciona el promedio detrás de cámaras.

MAX

Calcula el valor máximo de un conjunto de datos evaluados en un contexto de filtro, en palabras coloquiales, valida cuál fue el valor máximo que se generó en una columna afectada por un filtro u objeto visual.

Por ejemplo, queremos calcular cual fue la venta máxima que se generó en cada uno de los años.

Clic derecho en la tabla medidas de agregación → Nueva medida → Ingresamos la siguiente función

Arrojamos esta función a la matriz.

The screenshot shows a Power BI interface. At the top, there is a formula bar with a checkmark and the text "1 Venta MAX = MAX(Ventas[Ingresos])". Below the formula bar is a table with the following data:

Año	Ventas	Costos Totales	Utilidad	Cantidad	Prom Cantidad	#Registros	Prom2_Cantidad	Venta MAX
2017	21.662.728.028	13.810.322.355	7.852.405.673	3.169.951	209,21	15.152	209,21	31.868.100
2018	24.484.312.974	15.884.588.646	8.599.724.328	3.715.929	251,69	14.764	251,69	37.592.000
2019	24.684.311.343	17.551.117.780	7.133.193.563	4.545.938	311,92	14.574	311,92	40.273.200
2020	25.946.176.370	17.643.497.000	8.302.679.370	4.119.719	279,15	14.758	279,15	57.553.866
Total	96.777.528.715	64.889.525.781	31.888.002.934	15.551.537	262,48	59.248	262,48	57.553.866

Figura 4.19 Medida venta máxima.

El valor significa que para cada año esta fue la venta máxima evaluada en todos los registros o filas, siempre que deseamos comprobar un resultado, como empresa tenemos una metodología y es tomar solo un resultado de la matriz o gráfico, en este caso continuamos con 2019, filtramos este valor o conjunto de fechas en la tabla *Ventas*, nos llevamos la tabla a Excel y allí lo hacemos de manera manual y es sacar el valor máximo de estos datos para este caso.

The screenshot shows an Excel spreadsheet with a table of sales data. The first column is labeled "Fecha Factura". A dropdown arrow next to it is highlighted with a red box and the number 1. To the right of the table is a "Filtros de fecha" (Date Filters) dialog box. It contains fields for filtering by date range, with "Entre..." selected. Two date inputs are shown: "tiene lugar a partir del" with the value "01/01/2019" and "tiene lugar hasta el" with the value "31/12/2019". A yellow arrow points to the "Entre..." button. To the right of the dialog box is a context menu for the "Fecha Factura" column header, with several options like "Orden ascendente", "Copiar", and "Copiar tabla" listed. A yellow arrow points to the "Copiar tabla" option, which is highlighted with a red box and the number 3.

Figura 4.20 Proceso validar datos en Excel.

Copiamos la tabla en Excel y simplemente hacemos el cálculo

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
Fecha Factur	Nombre Clie	Cod.	Vended	Cod.Proc	Cantidad	Precio Unit	CostoUnit	Cod Lineas	Documento	Cod Municipi	Fecha Pedi	Fecha En	Ingresos	Costos	=MAX(M2:M14575)	40.273.200
martes, 26 d	QUALA S.A	B-01	8602	288	33408	27128 CA - 01	6412155	5088 lunes, 11 di	9.621.504	7812864						
sábado, 27 d	QUALA S.A	B-01	8602	72	33408	28679 CA - 01	8512339	5088 sábado, 23 lunes, 6	2.405.376	2064888						
martes, 23 d	QUALA S.A	B-01	8602	324	33408	29195 CA - 01	9312388	5088 lunes, 15 di doming	10.824.192	9459180						
jueves, 4 de	QUALA S.A	B-01	8602	324	33408	28365 CA - 01	2812299	5088 domingo, 1 martes, 9	10.824.192	9190260						
martes, 9 de	QUALA S.A	B-01	8602	144	33408	29210 CA - 01	312335	5088 domingo, 3 martes, 2	4.810.752	4206240						
miércoles, 1º	QUALA S.A	B-01	8602	180	33408	29484 CA - 01	9812153	5088 lunes, 8 de jueves, 1	6.013.440	5307120						

Figura 4.21 Resultado validado en Excel para 2019.

MIN

Por intuición realiza el cálculo contrario al ejemplo anterior, tirando como resultado el valor mínimo evaluado en un contexto de filtro.



Nota: Todos los objetos visuales funcionan como un filtro, es decir, si tenemos en una matriz el campo Vendedor, cada vendedor funciona como un filtro a la hora de llevar una medida a dicho objeto

Calculemos el valor mínimo en venta en cada uno de los años.

Clic derecho en la tabla medidas de agregación → Nueva medida → Ingresamos la siguiente función

1 Venta MIN = MIN(Ventas[Ingresos])									
Año	Ventas	Costos Totales	Utilidad	Cantidad	Prom Cantidad	#Registros	Prom2_Cantidad	Venta MAX	Venta MIN
2017	21.662.728.028	13.810.322.355	7.852.405.673	3.169.951	209,21	15.152	209,21	31.868.100	0
2018	24.484.312.974	15.884.588.646	8.599.724.328	3.715.929	251,69	14.764	251,69	37.592.000	0
2019	24.684.311.343	17.551.117.780	7.133.193.563	4.545.938	311,92	14.574	311,92	40.273.200	0
2020	25.946.176.370	17.643.497.000	8.302.679.370	4.119.719	279,15	14.758	279,15	57.553.866	0
Total	96.777.528.715	64.889.525.781	31.888.002.934	15.551.537	262,48	59.248	262,48	57.553.866	0

Figura 4.22 Valor mínimo por año.

Parece un resultado extraño, pero no, si vuelves hacer la validación como se ve la figura 4.21 y manualmente calculas el valor mínimo en Excel, debe dar cero, en los datos de ventas es muy común que se hagan descuentos, obsequios y no se cobre o tenga venta cero, se registran porque si tienen costos.



Advertencia: Las funciones vistas hasta este punto, solo admiten un único argumento, y es una columna, si ingresas una medida como argumento, nos trae un error, es decir, MAX ([Medida Ventas], genera error

DISTINCTCOUNT

Esta función permite contar los valores únicos de una columna evaluada en un contexto de filtro, por ejemplo, deseamos contar los clientes únicos que compraron en cada uno de los años esta función es la indicada.

Clic derecho en la tabla medidas de agregación → Nueva medida → Ingresamos la siguiente función

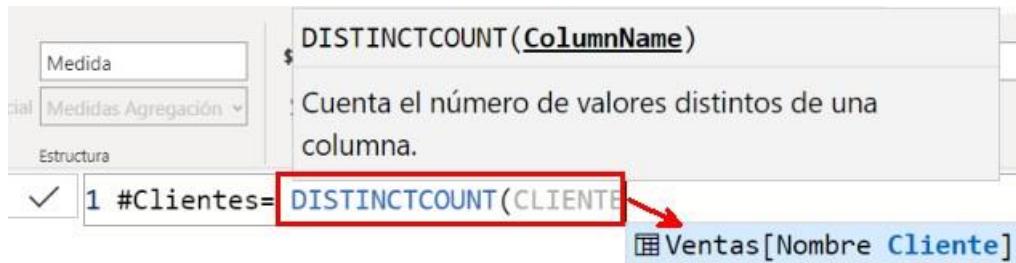


Figura 4.23 Función DISTINCTCOUNT.

Seleccionamos la columna Nombre Clientes de la tabla Ventas y llevamos la medida a la matriz.

Año	Ventas	Costos Totales	Utilidad	Cantidad	Prom Cantidad	#Registros	Prom2_Cantidad	Venta MAX	Venta MIN	#Clientes
2017	21.662.728.028	13.810.322.355	7.852.405.673	3.169.951	209,21	15.152	209,21	31.868.100	0	60
2018	24.484.312.974	15.884.588.646	8.599.724.328	3.715.929	251,69	14.764	251,69	37.592.000	0	60
2019	24.684.311.343	17.551.117.780	7.133.193.563	4.545.938	311,92	14.574	311,92	40.273.200	0	59
2020	25.946.176.370	17.643.497.000	8.302.679.370	4.119.719	279,15	14.758	279,15	57.553.866	0	60
Total	96.777.528.715	64.889.525.781	31.888.002.934	15.551.537	262,48	59.248	262,48	57.553.866	0	60

Figura 4.24 Resultado número de clientes por año.

Es una función muy útil, ya que trae un resultado para medir impactos de productos, de clientes, de cualquier campo que se quiera medir en cuanto a datos únicos.

Funciones de iteración

Las funciones de iteración son todas aquellas que tienen como sufijo o terminan con la letra "X" y hacen parte de la familia de las funciones de agregación pero se estudian por separado; como su palabra lo dice son funciones que repiten operaciones en ciertos registros visibles para arrojar un resultado según el cálculo indicado.

Uno de sus objetivos principales es evitar crear columnas calculadas y crear todo como medidas, así cuidar el rendimiento de tu modelo. Las funciones de iteración toman mucho potencial cuando se combinan con otras funciones y con las mismas medidas, temas que abordamos más adelante.

Pongamos en contexto lo que llevamos hasta este punto; hemos creado una serie de funciones de agregación para resumir datos por año, ejemplo, para crear el total de ventas por año, primero creamos una columna calculada en la tabla *Ventas*, multiplicamos *Cantidad * Precio Unit*, esto genera una serie de iteraciones de manera automática y luego con la función *SUM* creamos la medida [Ventas] que nos agrupara dicha información en el contexto de filtro año.

Para no tener que acudir a crear columnas adicionales, podemos iterar sobre la columna *Cantidad* y *Precio Unit* fila por fila, que almacene dicho resultado y por último sume las iteraciones totales.

SUMX

Esta función contiene dos argumentos obligatorios, una tabla y la expresión, es decir, primero necesitamos el nombre de la tabla sobre la que va a iterar para aplicarle un cálculo deseado, para nuestro caso sería multiplicar *Cantidad * Precio Unit*, esta sería la expresión.

La función primero itera y pues como su prefijo (SUM) lo dice, luego suma, veamos esto en acción.

Con los ejemplos anteriores, vamos a crear una nueva hoja en Power BI con la siguiente matriz.

Año	Ventas	Costos Totales	Utilidad
2017	21.662.728.028	13.810.322.355	7.852.405.673
2018	24.484.312.974	15.884.588.646	8.599.724.328
2019	24.684.311.343	17.551.117.780	7.133.193.563
2020	25.946.176.370	17.643.497.000	8.302.679.370
Total	96.777.528.715	64.889.525.781	31.888.002.934

Figura 4.25 Matriz ejemplo.

Ya lo hemos repetido, pero queremos ser reiterativos en esto, para que puedas tener claro que todo va en medidas y sus beneficios, [Ventas] es la suma de la siguiente columna calculada:

Ingresos = Ventas[Precio Unit]*Ventas[Cantidad]													
a	Nombre Cliente	Cod. Vendedor	Cod. Producto	Cantidad	Precio Unit	CostoUnit	Cod Lineas	Documento	Cod Municipio	Fecha Pedido	Fecha Entrega	Ingresos	
1ero de 2018	QUALA S.A	B-01	2303	20	25320	16254	AL - 01	6977376	25214	jueves, 25 de enero de 2018	domingo, 25 de marzo de 2018	506400	
1ero de 2018	QUALA S.A	B-01	2303	48	25320	16270	AL - 01	1378167	25214	jueves, 15 de febrero de 2018	miércoles, 14 de marzo de 2018	1215360	
1ero de 2018	QUALA S.A	B-01	2303	140	25320	16275	AL - 01	7078105	25214	jueves, 8 de marzo de 2018	martes, 10 de abril de 2018	3544800	
brero de 2018	QUALA S.A	B-01	2303	400	25320	16178	AL - 01	0375155	25214	martes, 13 de febrero de 2018	sábado, 24 de febrero de 2018	10128000	
brero de 2018	QUALA S.A	B-01	2303	100	25320	16147	AL - 01	6674103	25214	martes, 30 de enero de 2018	viernes, 23 de febrero de 2018	2532000	
julio de 2018	QUALA S.A	B-01	2303	140	25320	18659	AL - 01	4692192	25214	lunes, 14 de mayo de 2018	domingo, 5 de agosto de 2018	3544800	
julio de 2018	QUALA S.A	B-01	2303	20	25320	18661	AL - 01	5093150	25214	martes, 26 de junio de 2018	jueves, 9 de agosto de 2018	506400	

Figura 4.26 Columna calculada para ingresos.

Para nuestro modelo, esta tabla solo cuenta con 59.248 registros, algo muy insignificante para Power BI, que sucede si tenemos millones de filas, tu modelo quedaría modo tortuga, ya que las columnas

calculadas se almacenan en memoria dentro del modelo, en sí, SUMX es nuestro salvavidas, esta función tiene la posibilidad de recorrer esta tabla, creando este cálculo de manera implícito para solo ser ejecutado en el contexto que se analice la medida.

Creemos una nueva tabla para guardar estas nuevas medidas, la nombramos → Medidas Iteración.

Vamos a la pestaña Inicio → Datos → Especificar datos → Ver figura 4.2, 4.3 y 4.4



Figura 4.27 Tabla medidas iteración.

Antes de eliminar la columna1 agreguemos y veamos la sintaxis de SUMX:



Figura 4.28 Sintaxis SUMX.

1. **Table:** recibe una tabla, y esta sería sobre la que va a iterar, para nuestro caso es *Ventas*, esta tabla también puede estar filtrada o condicionada, funciones que veremos posteriormente.
2. **Expresión:** es la operación que la función realiza en diferentes columnas, por ejemplo, *Cantidad * Precio Unit*, incluso puede contener una sola columna, porque no puedes perder de vista que la función siempre va a ser una SUMA.

Con estos argumentos claros, creamos la medida → [Ventas X].

1 Ventas X = **SUMX(Ventas, Ventas[Cantidad]*Ventas[Precio Unit])**

Figura 4.29 Medida Ventas X.

Ponemos separadores de miles y llevamos a nuestra tabla matriz.

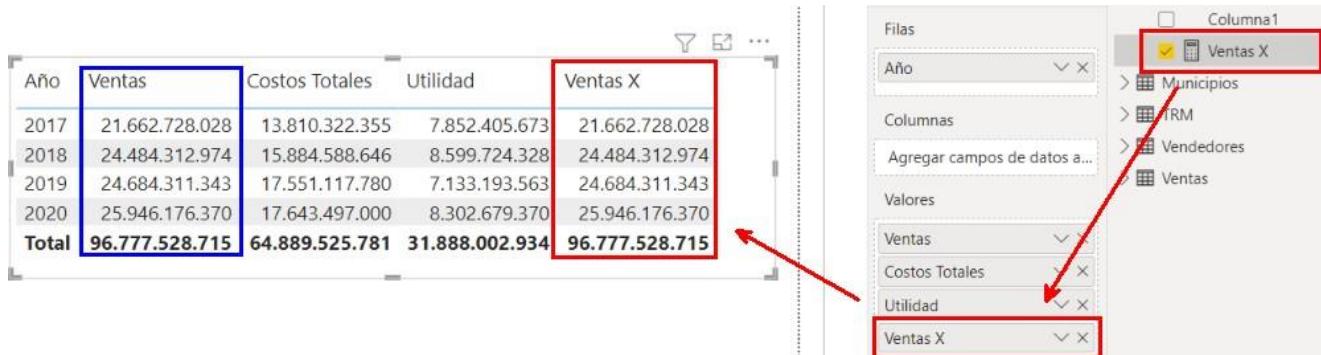


Figura 4.30 Medida Ventas X en la matriz.

El resultado es exactamente igual a la medida [Ventas] (apoyada de columna calculada).

Ya podemos eliminar la columna1 de la tabla *Medida Iteración*. Ver figura 4.6

Veamos el resultado detrás de cámaras, analizando el valor que hay en 2019. Vamos a la pestaña Datos y filtramos la fecha factura entre 1/1/2019 hasta 31/12/2019.

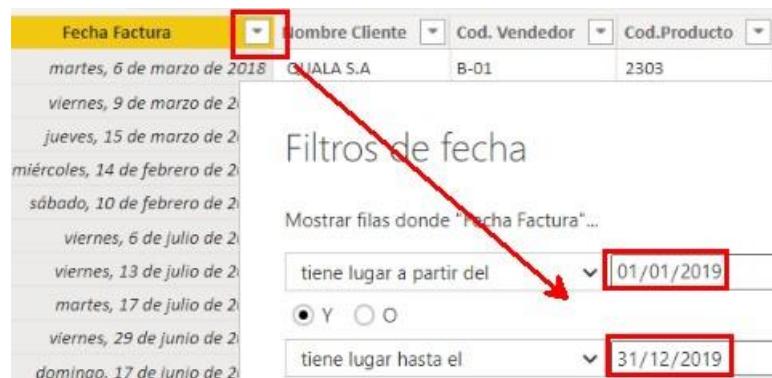


Figura 4.31 Filtrar tabla ventas.

Internamente la función recorre o itera sobre la columna *Cantidad* y *Precio Unit*, cada uno de estos cálculos se almacenan y al final se suman.

Fecha Factura	Nombre Cliente	Cod. Vendedor	Cod.Producto	Cantidad	Precio Unit		
martes, 26 de marzo de 2019	QUALA S.A.	B-01	8602	284	X 33408	= 9.621.504	
sábado, 27 de abril de 2019	QUALA S.A.	B-01	8602	72	X 33408	= 2.405.376	
martes, 23 de abril de 2019	QUALA S.A.	B-01	8602	324	X 33408	= 10.824.192	
jueves, 4 de abril de 2019	QUALA S.A.	B-01	8602	324	33408		
martes, 9 de abril de 2019	QUALA S.A.	B-01	8602	144	33408		
miércoles, 17 de abril de 2019	QUALA S.A.	B-01	8602	180	33408		
domingo, 16 de junio de 2019	QUALA S.A.	B-01	8602	216	33408		
sábado, 8 de junio de 2019	QUALA S.A.	B-01	8602	108	33408		
martes, 16 de julio de 2019	QUALA S.A.	B-01	8602	180	33408	= SUMA TOTAL	

Figura 4.32 Proceso interno SUMX.

Esto es lo que sucede en cada una de las filas iteradas por SUMX, pero la gran ventaja es que esta medida no se almacena en memoria a no ser que sea consultada o consumida en algún objeto visual.

En tu caminar con DAX te vas a encontrar con retos en donde no puedes ni usar o crear columnas calculadas, es decir, sí o sí tiene que crear medidas sin usar columnas calculadas en medio, por ejemplo, que sucede si queremos sacar el margen bruto en % de cada uno de los años, el beneficio o utilidad ya lo tenemos, que es la resta entre *Ventas* y *Costos*, incluso podríamos haber recurrido a crear una columna y luego llevar a una medida con SUM.

Pero si deseamos obtener el % margen en cada uno de los años, crear columnas antes del cálculo, sería un grave error, veámoslo.

Creemos una columna calculada en la tabla *Ventas* para este resultado.

%Margen Bruto = (*Ingresos* – *Costos*) / *Ingresos*. Anexamos formato porcentaje.

	Nombre Cliente	Cod. Vendedor	Cod.Producto	Cantidad	Precio Unit.	CostoUnit	Cod Lineas	Documento	Cod Municipio	Fecha Pedido	Fecha Entrega	Ingresos	Costos	%Margen Bruto
1. 6 de marzo de 2018	QUALA S.A.	B-01	2303	20	25320	16254	AL-01	6977376	25214	jueves, 8 de enero de 2018	domingo, 25 de marzo de 2018	506400	325080	35,81 %
1. 9 de marzo de 2018	QUALA S.A.	B-01	2303	48	25320	16270	AL-01	1378167	25214	jueves, 15 de febrero de 2018	miércoles, 14 de marzo de 2018	1215380	780560	35,74 %
15 de marzo de 2018	QUALA S.A.	B-01	2303	140	25320	16275	AL-01	7078105	25214	jueves, 8 de marzo de 2018	miércoles, 10 de abril de 2018	3544800	2278500	35,72 %
14 de febrero de 2018	QUALA S.A.	B-01	2303	400	25320	16178	AL-01	0375155	25214	martes, 13 de febrero de 2018	sábado, 24 de febrero de 2018	10128000	6471200	36,11 %
10 de febrero de 2018	QUALA S.A.	B-01	2303	100	25320	16147	AL-01	6674103	25214	martes, 30 de enero de 2018	vierres, 23 de febrero de 2018	2532000	1614700	36,23 %
1. 6 de julio de 2018	QUALA S.A.	B-01	2303	140	25320	18659	AL-01	4692192	25214	jueves, 14 de mayo de 2018	domingo, 5 de agosto de 2018	3544800	2612260	26,31 %

Figura 4.33 Columna margen bruto.

A nivel de fila el resultado es perfecto, si revisas cada fila tiene el margen correcto, que sucede si llevamos esto en una medida y sumamos dichos márgenes.

Clic derecho en la tabla Medidas Iteración → Nueva Medida → Ingresamos la siguiente medida y la llevamos a la matriz → Formato porcentaje

Año	Ventas	Costos Totales	Utilidad	Ventas X	Margen %
2017	21.662.728.028	13.810.322.355	7.852.405.673	21.662.728.028	-Infinito
2018	24.484.312.974	15.884.588.646	8.599.724.328	24.484.312.974	-Infinito
2019	24.684.311.343	17.551.117.780	7.133.193.563	24.684.311.343	-Infinito
2020	25.946.176.370	17.643.497.000	8.302.679.370	25.946.176.370	-Infinito
Total	96.777.528.715	64.889.525.781	31.888.002.934	96.777.528.715	-Infinito

Figura 4.34 Medida margen bruto en matriz.

El resultado es infinito, es decir tu margen bruto no tiene techo lo que es mentira, si conocemos nuestros datos sabemos que no es verdad, ¿Por qué pasa esto? Sigue porque Power BI está sumando

cada una de las filas, sin tener en cuenta el contexto de fila, es decir, si tomamos nuestro ejemplo de 2019, SUM filtra las fechas de 2019 y suma toda la columna %*Margen Bruto*, por consiguiente, un resultado erróneo.

En este tipo de ejercicios es donde necesitas aplicar funciones de iteración, o simplemente puedes dividir el total *Utilidad /Ventas*, pero recuerda que el cálculo para *Utilidad* necesito de una columna calculada, el objetivo es eliminar por completo esto.

Desarrollemos una medida llamada [*Utilidad 2*], donde vamos a iterar fila a fila para realizar esta resta, como lo muestra la imagen siguiente:

The screenshot shows the Power BI interface. At the top, there is a formula bar with the following DAX code:

```

1 Utilidad 2 = SUMX(Ventas, (Ventas[Precio Unit]*Ventas[Cantidad])-  

2 (Ventas[CostoUnit]*Ventas[Cantidad]))  

3

```

A red box highlights the part of the formula from 'SUMX' to the closing parenthesis. An arrow points down to a table visual below. The table has columns: Año, Ventas, Costos Totales, Utilidad (highlighted with a blue border), Ventas X, and Utilidad 2 (highlighted with a red border). The data is as follows:

Año	Ventas	Costos Totales	Utilidad	Ventas X	Utilidad 2
2017	21.662.728.028	13.810.322.355	7.852.405.673	21.662.728.028	7.852.405.673
2018	24.484.312.974	15.884.588.646	8.599.724.328	24.484.312.974	8.599.724.328
2019	24.684.311.343	17.551.117.780	7.133.193.563	24.684.311.343	7.133.193.563
2020	25.946.176.370	17.643.497.000	8.302.679.370	25.946.176.370	8.302.679.370
Total	96.777.528.715	64.889.525.781	31.888.002.934	96.777.528.715	31.888.002.934

Figura 4.35 Utilidad 2 creada con iteradores.



Nota: Otra recomendación, es haber creado la medida Costos con SUMX y luego una simple resta entre las dos medidas (Ventas – Costos).

Podemos ver el potencial que contiene SUMX, el resultado es exactamente igual a la medida *Utilidad*, creada con columnas calculadas.



Utilidad 2 = SUMX (Ventas,

$$(Ventas[Precio Unit]*Ventas[Cantidad])-
(Ventas[CostoUnit]*Ventas[Cantidad]))$$

Para ingresar nuevas líneas en la barra de fórmulas lo hacemos con la combinación de ALT + enter o SHIF + enter, estudiaremos como formatear los códigos DAX.

Volvamos a nuestra medida [Margen %] y dividimos [Utilidad 2] / [Ventas X], dos medidas creadas sin usar columnas calculadas con anterioridad.

Para obtener mejores prácticas vamos a usar la función **DIVIDE**, tiene una ventaja en caso de haber un error no trae la palabra infinito, solo trae blanco.

Tiene 3 argumentos: Numerador, Denominador y el último argumento como opcional, en caso de que el cálculo genere un error se puede poner blanco, un guion, un cero etc. Usar la función DIVIDE es una muy buena práctica que se pueda adoptar.

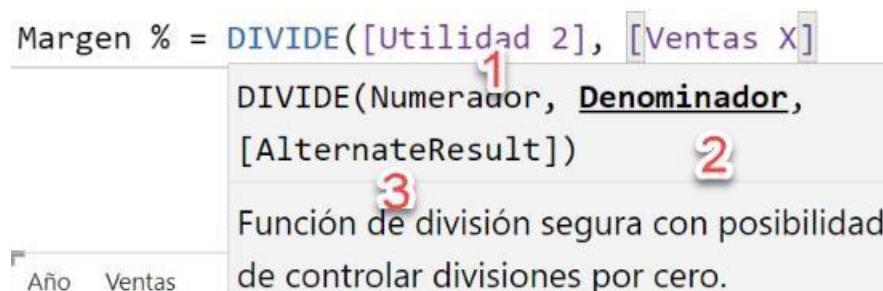


Figura 4.36 Argumentos función DIVIDE.

Cerramos paréntesis, y llevamos de nuevo esta medida a nuestra matriz, el resultado es el correcto ahora.

Año	Ventas	Costos Totales	Utilidad	Ventas X	Utilidad 2	Margen %
2017	21.662.728.028	13.810.322.355	7.852.405.673	21.662.728.028	7.852.405.673	36,25 %
2018	24.484.312.974	15.884.588.646	8.599.724.328	24.484.312.974	8.599.724.328	35,12 %
2019	24.684.311.343	17.551.117.780	7.133.193.563	24.684.311.343	7.133.193.563	28,90 %
2020	25.946.176.370	17.643.497.000	8.302.679.370	25.946.176.370	8.302.679.370	32,00 %
Total	96.777.528.715	64.889.525.781	31.888.002.934	96.777.528.715	31.888.002.934	32,95 %

Figura 4.37 Margen % calculado correctamente.



Advertencia: Por el momento no debes usar medidas en el argumento expresión (segundo parámetro) de las funciones de iteración, en este caso en SUMX, hasta no entender y haber explicado transición de contextos.



Nota: En resumen, la función SUM es una Syntax Sugar o abreviación de la función SUMX, por lo tanto, si usas cualquiera de las dos no hay diferencia en el rendimiento.

Este resultado del Margen % es un promedio ponderado, donde se suma el total de ventas sobre el total de utilidad, si deseamos calcular la medida de *Precio Unit* debemos recurrir a DIVIDE, entre el total de ventas sobre la suma de cantidades, el resultado es el precio promedio ponderado.

SUMX va acumulando cada iteración que realiza, por eso es necesario y repetimos que en el argumento expresión se haga alguna operación aritmética, por ejemplo, si deseamos crear la suma de las cantidades con SUMX sería así:

Medida = **SUMX(Ventas, Ventas[Cantidad])**

Figura 4.38 Medidas cantidades con SUMX.

Porque es importante que comprendas esto, porque cuando hablamos de transición de contextos y cálculos un poco más avanzado tengas claro el panorama de lo que sucede detrás de cámaras.

AVERAGEX

Esta función de la familia de iteradores ejecuta el promedio iterando fila a fila según la expresión indicada evaluada en un contexto de filtro.

Si usamos esta función para calcular el promedio de ventas o cantidad por año, pues el resultado va el mismo que la función de agregación AVERAGE, porque todo depende del contexto.

Pero si podemos sacar provecho de esta función si por ejemplo deseamos saber el promedio de días por entrega de pedidos, en nuestro modelo contamos con la columna Fecha Pedido y Fecha Entrega, si no queremos crear una columna adicional donde me genere la resta entre estas dos fechas y luego sacar el promedio, pues AVERAGEX es la apropiada.

Al grano, vamos a crear una nueva medida en la tabla Medidas Iteración → La nombramos [Prom Días Entrega] → Ingresamos la siguiente función arrojada en la matriz

Prom Días Entrega = **AVERAGEX(Ventas, Ventas[Fecha Entrega]-Ventas[Fecha Pedido])**

Año	Ventas	Costos Totales	Utilidad	Ventas X	Utilidad 2	Margen %	Prom Días Entrega
2017	21.662.728,028	13.810.322,355	7.852.405,673	21.662.728,028	7.852.405,673	36,25 %	43,10
2018	24.484.312,974	15.884.588,646	8.599.724,328	24.484.312,974	8.599.724,328	35,12 %	45,58
2019	24.684.311,343	17.551.117,780	7.133.193,563	24.684.311,343	7.133.193,563	28,90 %	45,99
2020	25.946.176,370	17.643.497,000	8.302.679,370	25.946.176,370	8.302.679,370	32,00 %	45,38
Total	96.777.528,715	64.889.525,781	31.888.002,934	96.777.528,715	31.888.002,934	32,95 %	45,00

Figura 4.39 Promedio días entrega.

La medida recorre la tabla ventas iterando sobre la columna Fecha Entrega menos Fecha Pedido, y este es el resultado de días entrega promedio en cada uno de los años.

MAXX

Calcula el valor máximo del resultado de una iteración, esta función es objetiva dependiendo del contexto donde se evalúa, si creamos una medida calculando el valor máximo iterando la columna ingresos, el resultado va a ser el mismo que crear la medida con MAX, por el contexto, para crear ejemplos prácticos con esta función debemos estudiar primero las funciones de tabla y la transición de contexto, ya que esto nos dará más potencial en su aplicabilidad.

Por ejemplo, si queremos saber en qué mes o día se generó la mayor venta o cual fue la venta máxima por año en alguno de los meses, debemos recurrir a otras funciones que nos ayuden a esto, solo por poner el ejemplo queremos que veas el resultado de esta operación, pero no te asustes ni te apresures, recuerda que apenas vamos en el camino hacia dominar, pero sobre todo aplicar DAX a tus informes.

Le medida sería la siguiente:

1 Venta MAX Mes = **MAXX(VALUES(Calendar[Mes]),[Ventas])**

Año	Ventas	Costos Totales	Utilidad	Ventas X	Utilidad 2	Margen %	Prom Días Entrega	Venta MAX Mes
2017	21.662.728,028	13.810.322,355	7.852.405,673	21.662.728,028	7.852.405,673	36,25 %	43,10	2.475.851.186
2018	24.484.312,974	15.884.588,646	8.599.724,328	24.484.312,974	8.599.724,328	35,12 %	45,58	2.455.011.214
2019	24.684.311,343	17.551.117,780	7.133.193,563	24.684.311,343	7.133.193,563	28,90 %	45,99	2.286.797.916
2020	25.946.176,370	17.643.497,000	8.302.679,370	25.946.176,370	8.302.679,370	32,00 %	45,38	2.544.078.418
Total	96.777.528,715	64.889.525,781	31.888.002,934	96.777.528,715	31.888.002,934	32,95 %	45,00	9.267.922.320

Figura 4.40 Venta máxima en alguno de los meses.

Recuerda que esto es un viaje o una curva de conocimiento que debemos recorrer, aún las cosas las podemos volver un poco más complejas, por ejemplo, cómo podemos encontrar el nombre del mes en que se generó esta venta, te vamos a dar la medida ya ejecutada, pero en capítulos posteriores debemos

Llegar a estos resultados complejos, vas a terminar este libro con los conocimientos necesarios para este tipo de medidas:

```
1 Encontrar Mes =
2 VAR Mayor_Ingreso = [Venta MAX Mes]
3 VAR EncontrarMes =
4     FILTER ( VALUES ( Calendario[Mes] ), [Ventas] = Mayor_Ingreso )
5 VAR Resultado =
6     IF ( COUNTROWS ( EncontrarMes ) = 1, EncontrarMes, BLANK () )
7 RETURN
8 Resultado
```

Año	Ventas	Costos Totales	Utilidad	Ventas X	Utilidad 2	Margen %	Prom Días Entrega	Venta MAX Mes	Encontrar Mes
2017	21.662.728,028	13.810.322,355	7.852.405,673	21.662.728,028	7.852.405,673	36,25 %	43,10	2.475.851,186	Agosto
2018	24.484.312,974	15.884.588,646	8.599.724,328	24.484.312,974	8.599.724,328	35,12 %	45,58	2.455.011,214	Agosto
2019	24.684.311,343	17.551.117,780	7.133.193,563	24.684.311,343	7.133.193,563	28,90 %	45,99	2.286.797,916	Mayo
2020	25.946.176,370	17.643.497,000	8.302.679,370	25.946.176,370	8.302.679,370	32,00 %	45,38	2.544.078,418	Octubre
Total	96.777.528,715	64.889.525,781	31.888.002,934	96.777.528,715	31.888.002,934	32,95 %	45,00	9.267.922,320	Agosto

Figura 4.41 Encontrar el nombre del mes con la venta máxima.

Si aún estas iniciando en DAX como la mayoría de los usuarios que pasan por estos lados, no te preocupes, olvida esta imagen te garantizamos que la comprenderás más adelante.

MINX

Trae el valor mínimo de una iteración evaluada en un contexto de filtro, vamos a profundizar en los iteradores cuando expliquemos funciones de tabla y transición de contexto, allí hay mucho poder reprimido que estamos seguros vas a sacar provecho en tus informes.

Funciones de texto

Las funciones DAX operan muy distinto a las fórmulas tradicionales de Excel, eso debe quedar claro, Excel no trabaja con datos agregados, como si lo hace Power BI, DAX tiene en cuenta los contextos y en esto se encuentra el secreto, pero las funciones de texto en el lenguaje DAX son muy similares a sus argumentos a las de Excel y si vienes o manejas Excel te vas a identificar.

LEFT

Es la misma función que usamos en Excel llamada IZQUIERDA, es útil para extraer los primeros caracteres de un texto, esta función se usa especialmente para extraer, validar o filtrar los primeros caracteres de una columna dentro de nuestro modelo de datos.

Para ver un ejemplo claro, nos dirigimos a la pestaña Datos → Seleccionamos la tabla *Calendario 2* → y vamos a extraer los primeros 3 letras del mes texto.

Inicio → Cálculos → Nueva columna → Nombramos la medida Fx Left→

The screenshot shows the Power BI Data Editor with the 'Calendario 2' table selected. A new column named 'Fx Left' has been created, containing the formula `LEFT('Calendario 2'[Mes],3)`. Below the table, the DAX documentation for the `LEFT` function is displayed, stating: "Devuelve el número de caracteres especificados desde el principio de una cadena de texto".

Figura 4.42 Función LEFT.

Esta función cuenta con dos argumentos en su sintaxis, Texto y Número de caracteres, este último es opcional, si se omite, DAX internamente trae solo 1 carácter.



Nota: Si un argumento de una función se encuentra entre corchetes significa que es un argumento opcional, pero recomendamos revisar siempre su resultado.

Por último, cerramos paréntesis y obtenemos el siguiente resultado.

The screenshot shows the Power BI Data Editor with the 'Calendario 2' table. The 'Fx Left' column now contains the first three letters of each month name. A context menu is open over the 'Fx Left' column header, with the 'Filtros de texto' option highlighted. A red arrow points from this option to a list of month abbreviations ('abr', 'ago', 'dic', 'ene', 'feb', 'jul', 'jun', 'mar') which are all checked, indicating they are being used as filters.

Figura 4.43 Resultado función LEFT.

RIGHT

Es de las pocas o únicas funciones que se ejecutan de derecha a izquierda extrayendo el número de caracteres indicado.

Para nuestro ejemplo, vamos a extraer los últimos dos dígitos del año.

Inicio → Cálculos → Nueva columna → Nombramos la medida Fx Right→

The screenshot shows a Power BI data view with a table containing columns: Año, Mes Num, Trimestre, Mes, Semana, Día Mes, Nombre Día, Fx Left, and Fx Right. The Fx Right column is highlighted with a red box and contains the formula `RIGHT('Calendario 2'[Año], 2)`. A context menu is open over the Fx Right column, with the 'Orden ascendente' option selected. To the right of the table, a red box highlights the 'Buscar' section of the context menu, which includes options like '(Seleccionar todo)', '17', '18', '19', and '20', all of which are checked.

	Año	Mes Num	Trimestre	Mes	Semana	Día Mes	Nombre Día	Fx Left	Fx Right
10 a. m.	2017	1		1 enero		1	domingo	ene	17
10 a. m.	2017	1		1 enero		2	lunes	ene	17
10 a. m.	2017	1		1 enero		3	martes	ene	17
10 a. m.	2017	1		1 enero		4	miércoles	ene	17
10 a. m.	2017	1		1 enero		5	jueves	ene	17
10 a. m.	2017	1		1 enero		6	viernes	ene	17
10 a. m.	2017	1		1 enero		7	sábado	ene	17
10 a. m.	2017	1		1 enero		8	domingo	ene	17
10 a. m.	2017	1		1 enero		9	lunes	ene	17
10 a. m.	2017	1		1 enero		10	martes	ene	17
10 a. m.	2017	1		1 enero		11	miércoles	ene	17
10 a. m.	2017	1		1 enero		12	jueves	ene	17
10 a. m.	2017	1		1 enero		13	viernes	ene	17
10 a. m.	2017	1		1 enero		14	sábado	ene	17

Figura 4.44 Últimos 2 caracteres del año.

CONCATENATE

Une varias cadenas de texto, pero esta función en DAX a diferencia de Excel solo admite dos argumentos, mientras que en Excel admite hasta 255 caracteres.

Unamos el mes con el año. → Crear una nueva columna en la tabla que estamos trabajando.

The screenshot shows a Power BI data view with a table containing columns: Año, Mes, and Año_Mes. The Año_Mes column is highlighted with a red box and contains the formula `CONCATENATE('Calendario 2'[Año], 'Calendario 2'[Mes])`. The formula bar also shows the formula `CONCATENATE(Text1, Text2)` with `Text1` and `Text2` underlined. A context menu is open over the Año_Mes column, with the 'Ordenar por columna' option selected.

Año	Mes	Año_Mes
2017	enero	2017 enero
2017	febrero	2017 febrero
2017	marzo	2017 marzo
2017	abril	2017 abril
2017	mayo	2017 mayo
2017	junio	2017 junio
2017	julio	2017 julio
2017	agosto	2017 agosto
2017	septiembre	2017 septiembre
2017	octubre	2017 octubre
2017	noviembre	2017 noviembre
2017	diciembre	2017 diciembre

Figura 4.45 Unir año y mes.

El resultado es una columna con el año y el mes juntos

Si deseamos unir el año, seguido de un slash más el mes, debemos usar dos funciones por la limitante de los argumentos, solo admite dos campos, podemos ayudarnos del signo de concatenación “&” ampersand.

La función la podemos modificar y obtenemos el resultado.

CONCATENATE('Calendario 2'[Año],"/") &'Calendario 2'[Mes]

Mes Num	Trimestre	Mes	Semana	Día Mes	Nombre Dia	Fx Left	Fx Right	Año_Mes
1	1	enero	1	1	domingo	ene	17	2017/enero
1	1	enero	2	2	lunes	ene	17	2017/enero
1	1	enero	2	3	martes	ene	17	2017/enero
1	1	enero	2	4	miércoles	ene	17	2017/enero
1	1	enero	2	5	jueves	ene	17	2017/enero
1	1	enero	2	6	viernes	ene	17	2017/enero
1	1	enero	2	7	sábado	ene	17	2017/enero

Figura 4.46 Año, slash y mes concatenado.

UPPER

Convierte a mayúscula una cadena de texto, solo contiene un único argumento.

Si queremos convertir en mayúscula el nombre del mes, encerramos esta columna en UPPER.

Continuamos en la tabla *Calendario 2* → Nueva Columna → Fx Upper

Fx Upper = UPPER('Calendario 2'[Mes])								
Año	Mes Num	Trimestre	Mes	Semana	Día Mes	Nombre Dia	Fx Left	Fx Right
O a. m.	2017	1	1	enero	1	domingo	ene	17
O a. m.	2017	1	1	enero	2	lunes	ene	17
O a. m.	2017	1	1	enero	2	martes	ene	17
O a. m.	2017	1	1	enero	2	miércoles	ene	17
O a. m.	2017	1	1	enero	2	jueves	ene	17
O a. m.	2017	1	1	enero	2	viernes	ene	17

Figura 4.47 Convertir mes en mayúscula.

Una función muy sencilla de usar.

LOWER

Convierte en minúscula una cadena de texto, solo contiene un único argumento.

Reversemos el paso anterior, esta última columna Fx Upper la convertimos en minúscula.

Fx Lower = LOWER('Calendario 2'[Fx Upper])								
Año	Mes Num	Trimestre	Mes	Semana	Día Mes	Nombre Dia	Fx Left	Fx Right
O a. m.	2017	1	1	enero	1	domingo	ene	17
O a. m.	2017	1	1	enero	2	lunes	ene	17
O a. m.	2017	1	1	enero	2	martes	ene	17
O a. m.	2017	1	1	enero	2	miércoles	ene	17
O a. m.	2017	1	1	enero	2	jueves	ene	17
O a. m.	2017	1	1	enero	2	viernes	ene	17

Figura 4.48 Convertir mes en minúscula.

LEN

Devuelve como resultado el número de caracteres que compone un texto.

Extraemos el número de caracteres que tiene la columna mes.

The screenshot shows a Power BI report with a calendar table. The table has columns: Año, Mes Num, Trimestre, Mes, Semana, Día Mes, Nombre Día, Fx Left, Fx Right, Año_Mes, Fx Upper, Fx Lower, and Fx Len. The Fx Len column displays the length of the month names. A red arrow points from the Fx Len column to a context menu. The menu includes: Orden ascendente, Orden descendente, Borrar orden, Borrar filtro, Borrar todos los filtros, and Filtros de número. A sub-menu for 'Filtros de número' is open, showing checkboxes for (Seleccionar todo), 4, 5, 6, 7, 9, and 10, with checkboxes for 4, 5, 6, 7, 9, and 10 being checked. A red box highlights the 'Filtros de número' section.

Año	Mes Num	Trimestre	Mes	Semana	Día Mes	Nombre Día	Fx Left	Fx Right	Año_Mes	Fx Upper	Fx Lower	Fx Len
2017	1		enero	1	1	domingo	ene	17	2017/enero	ENERO	enero	3
2017	1		enero	2	2	lunes	ene	17	2017/enero	ENERO	enero	4
2017	1		enero	2	3	martes	ene	17	2017/enero	ENERO	enero	5
2017	1		enero	2	4	miércoles	ene	17	2017/enero	ENERO	enero	6
2017	1		enero	2	5	jueves	ene	17	2017/enero	ENERO	enero	7
2017	1		enero	2	6	viernes	ene	17	2017/enero	ENERO	enero	8
2017	1		enero	2	7	sábado	ene	17	2017/enero	ENERO	enero	9
2017	1		enero	2	8	domingo	ene	17	2017/enero	ENERO	enero	10
2017	1		enero	3	9	lunes	ene	17	2017/enero	ENERO	enero	11
2017	1		enero	3	10	martes	ene	17	2017/enero	ENERO	enero	12
2017	1		enero	3	11	miércoles	ene	17	2017/enero	ENERO	enero	13
2017	1		enero	3	12	jueves	ene	17	2017/enero	ENERO	enero	14
2017	1		enero	3	13	viernes	ene	17	2017/enero	ENERO	enero	15
2017	1		enero	3	14	sábado	ene	17	2017/enero	ENERO	enero	16

Figura 4.49 Número de caracteres columna mes.



Nota: Todas estas funciones toman poder cuando se combinan entre ellas, podemos sacar resultados asombrosos.

Vamos a ver un poco más de aplicabilidad a estas funciones, en nuestro modelo cargamos una tabla de correos.

The screenshot shows a Power BI report with a 'Correo' table containing a list of emails. To the right, a context menu is open for 'Tabla Correos', with the option 'Tabla Correos' highlighted by a red box. Other options in the menu include: Lista Países, Maestro Productos, Municipios, TRM, Vendedores, and Ventas.

Correo
didier.morales@hotmail.com
adriana.acevedo@gmail.com
cesar.higuita@yahoo.com
claudia.valencia@servicios.co
delsy.castro@iflexo.co
gloria.olaya@gmail.com
ian.movilla@excel.com
jenifer.sanchez@dataice.co
jheyson.sanchez@gmail.com

Figura 4.50 Tabla correos cargada.

SEARCH

Aunque es esta función hace parte de las categorías de texto, parece una función de búsqueda, pues esto es lo que hace, determina en qué posición se encuentra un determinado carácter.

Esta función tiene 4 argumentos, de los cuales 2 son obligatorios y 2 opcionales.

Pongámonos un reto, si deseamos extraer el nombre de cada correo antes del signo “@” de la tabla anterior, no podemos usar simplemente la función LEFT por una simple razón, el número de caracteres de cada nombre es diferente, por ende, necesitamos combinar varias funciones vistas hasta este punto.

Primero con SEARCH identificamos hasta el signo “@” cuantos caracteres hay.

Seleccionamos la Tabla Correos → Nueva columna y diligenciamos los argumentos de esta función.

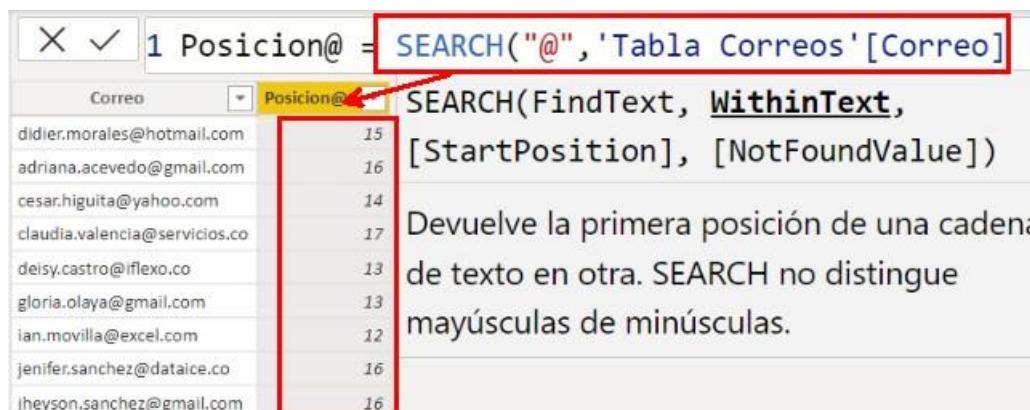


Figura 4.51 Argumentos función SEARCH.

El primer argumento es el carácter buscado “@”, va entre comillas por ser un texto y el segundo es donde, en que columna se va a buscar dicho carácter.

Si analizamos el primer resultado, el número 15 es el número de caracteres que hay hasta el @. Este resultado es dinámico y nos va a servir para extraer el nombre del correo, ya que si usamos este resultado como el segundo argumento de la función LEFT el resultado es:

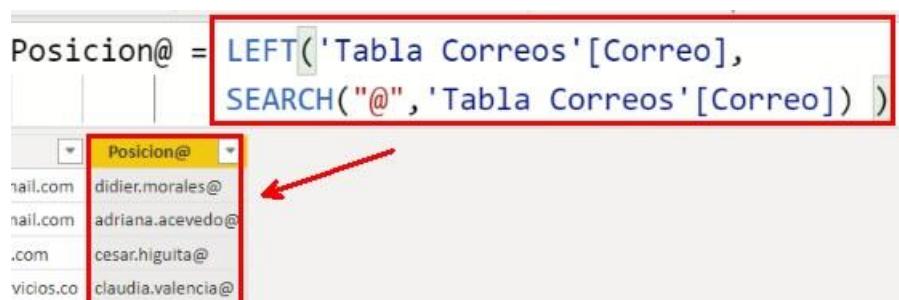


Figura 4.52 Traer datos hasta el @.

El nombre aun contiene el símbolo @, para eliminarlo simplemente le restamos 1 a la función SEARCH.

Posicion@ =	<code>LEFT('Tabla Correos'[Correo], SEARCH("@",'Tabla Correos'[Correo])-1)</code>										
	<table border="1"> <tr> <td>Posicion@</td> <td></td> </tr> <tr> <td>nail.com</td> <td>didier.morales</td> </tr> <tr> <td>nail.com</td> <td>adriana.acevedo</td> </tr> <tr> <td>.com</td> <td>cesar.higuita</td> </tr> <tr> <td>'vicios.co</td> <td>claudia.valencia</td> </tr> </table>	Posicion@		nail.com	didier.morales	nail.com	adriana.acevedo	.com	cesar.higuita	'vicios.co	claudia.valencia
Posicion@											
nail.com	didier.morales										
nail.com	adriana.acevedo										
.com	cesar.higuita										
'vicios.co	claudia.valencia										

Figura 4.53 Extraer nombre del correo.



```
Posicion@ = LEFT ('Tabla Correos'[Correo],  
SEARCH ("@",'Tabla Correos'[Correo])-1)
```

Antes que continúes queremos que realices el siguiente ejercicio:

Ejercicio: Extraer el dominio de cada correo, es decir, el primer resultado sería Hotmail.com

Antes que continúes con tu lectura, realiza el ejercicio.

Solución: Para realizar este ejercicio, debes saber que la función LEFT no nos sirve, ya que necesitamos extraer datos de derecha a izquierda, pero debemos crear una función que nos dé el número de caracteres después del @, esta función así de específica no existe en DAX, contar el número de registros después de un carácter no está, pero si combinamos LEN con SEARCH lo podemos lograr.

El objetivo es saber cuántos caracteres tiene cada correo, luego le restamos el valor que se encuentre hasta el @, la función sería la siguiente.

Dominio =	<code>RIGHT('Tabla Correos'[Correo], LEN('Tabla Correos'[Correo])- SEARCH("@",'Tabla Correos'[Correo]))</code>																		
	<table border="1"> <tr> <td>Posicion@</td> <td>Dominio</td> </tr> <tr> <td>mail.com</td> <td>didier.morales</td> </tr> <tr> <td>mail.com</td> <td>adriana.acevedo</td> </tr> <tr> <td>.com</td> <td>cesar.higuita</td> </tr> <tr> <td>'vicios.co</td> <td>claudia.valencia</td> </tr> <tr> <td></td> <td>hotmail.com</td> </tr> <tr> <td></td> <td>gmail.com</td> </tr> <tr> <td></td> <td>yahoo.com</td> </tr> <tr> <td></td> <td>servicios.co</td> </tr> </table>	Posicion@	Dominio	mail.com	didier.morales	mail.com	adriana.acevedo	.com	cesar.higuita	'vicios.co	claudia.valencia		hotmail.com		gmail.com		yahoo.com		servicios.co
Posicion@	Dominio																		
mail.com	didier.morales																		
mail.com	adriana.acevedo																		
.com	cesar.higuita																		
'vicios.co	claudia.valencia																		
	hotmail.com																		
	gmail.com																		
	yahoo.com																		
	servicios.co																		

Figura 4.54 Extraer el dominio de un correo.

Estas funciones son de mucha utilidad aprenderlas en DAX ya que a medida que avancemos vamos a poder crear cálculos de mucho potencial para nuestro análisis de datos.



```
Dominio = RIGHT( 'Tabla Correos'[Correo],  
LEN('Tabla Correos'[Correo])-  
SEARCH("@",'Tabla Correos'[Correo]))
```

SUBSTITUTE

Es la función que usamos en Excel como sustituir, útil para reemplazar un carácter por otro dentro de una cadena de texto, por ejemplo, la columna llamada *Posicion@* podemos organizar esta columna reemplazando el punto por un espacio.

Creemos una columna nueva, esta función tiene 4 argumentos de los cuales 3 son obligatorios.

Texto: El texto o a columna donde se va a ser el reemplazo

Texto antiguo: El carácter que se va a reemplazar

Nuevo texto: El nuevo carácter que va a ir en el texto

La función quedaría así:

Nombre Ajustado = SUBSTITUTE('Tabla Correos'[Posicion@], "."," ")		
	Posicion@	Dominio
@mail.com	didier.morales	hotmail.com
gmail.com	adriana.acevedo	gmail.com
.com	cesar.higuita	yahoo.com
ervicios.co	claudia.valencia	servicios.co

Figura 4.55 Reemplazar un carácter.

Podemos darle un mejor formato, encerrando esta función en UPPER.

UPPER(SUBSTITUTE('Tabla Correos'[Posicion@], "."," "))		
	Dominio	Nombre Ajustado
hotmail.com		DIDIER MORALES
gmail.com		ADRIANA ACEVEDO
yahoo.com		CESAR HIGUITA
servicios.co		CLAUDIA VALENCIA

Figura 4.56 Convertir en mayúscula el nuevo nombre.



Nombre Ajustado = UPPER(
SUBSTITUTE('Tabla Correos'[Posicion@], "."," "))



Nota: Aunque DAX no es una herramienta para limpieza de datos, es muy importante entender estas funciones, ya que si se requiere una medida que sume solo las cuentas contables que inician por "41" debes obligatoriamente usar un LEFT, de esta misma forma nos vamos a encontrar con cientos de casos.

Capítulo 5: Manejo de variables

Las variables en el lenguaje DAX son básicamente para que el código sea más legible, evitar el uso repetitivo de expresiones y ayudar al rendimiento, las variables se definen con la palabra VAR, luego se debe proporcionar una sección RETURN que defina el valor resultado de la expresión.

Las variables solo se pueden usar dentro de la misma expresión, no existe variables en DAX de manera global, es decir, cuando creamos una variable esta no se puede usar en todo el modelo, ni hacer referencia a ella desde otro cálculo que no sea en el mismo ambiente que se creó.

Las variables se ejecutan utilizando una evaluación retrasada, es decir, si definimos una variable que por alguna razón no se calcula dentro del código la variable en sí misma nunca será evaluada, por eso las variables son de suma importancia cuando creamos cálculos complejos usando una expresión varias veces.

También podemos definir múltiples variables dentro del mismo código y podemos anidar múltiples variables dentro de una de ellas, tienen una gran ventaja es que las variables solo se evalúan una vez, aunque se usen varias veces dentro del mismo código DAX garantiza que se evalúe una sola vez.

Debemos tener claro que al final las variables son constantes en DAX, ya que se evalúan una sola vez después de la indicación VAR y no cuando se usa su valor.

Declarar variables en DAX

Cuando ingresamos la palabra VAR en DAX estamos indicado una sentencia que debe estar seguida del nombre de esta variable, la cual no debe contener espacios y evitar nombres que tengan conflictos con nombres de tablas, columnas o medidas, luego se ingresa el signo igual y se termina con el RETURN seguido de la expresión a evaluar.

Vamos a nuestro modelo y seleccionamos la tabla *Calendario 2* → Columna mes (Archivo Capítulo 5)

	Año	Mes Num	Trimestre	Mes	Semana	Día Mes	Nombre Día
10 a. m.	2017	1	1	1 enero	1	1	domingo
10 a. m.	2017	1	1	1 enero	2	2	lunes
10 a. m.	2017	1	1	1 enero	2	3	martes
10 a. m.	2017	1	1	1 enero	2	4	miércoles
10 a. m.	2017	1	1	1 enero	2	5	jueves
10 a. m.	2017	1	1	1 enero	2	6	viernes
10 a. m.	2017	1	1	1 enero	2	7	sábado

Figura 5.1 Tabla calendario 2, columna mes.

Uno de los primeros retos con nuestros estudiantes cuando impartimos nuestras capacitaciones, es que cuando tenemos Power BI en español y extraemos el nombre del mes de la fecha el resultado es todo en minúscula y en DAX hasta este punto no existe una función como en Excel llamada NOMPROPIO, que convierta la primera letra en mayúscula, toca crearla de manera personalizada mediante la combinación de varias funciones y nos va servir de ejemplo para explicar el uso y ventajas de manejar variables.



Sintaxis Variable: **VAR Nombre_Variable = Calculo**

RETURN

Nombre_Variable

Ingresemos este cálculo de la columna Mes en una variable según la sintaxis anterior dada, para efectos de estudio, vamos a nombrar la variable como "X".

Mes = VAR X =FORMAT('Calendario 2'[Date],"mmmm")	↓
RETURN	
X	

Figura 5.2 Calculo nombre mes con variable.

El resultado es exactamente igual, hasta este punto no hay ciencia ni ninguna diferencia, como el objetivo es que la primera letra quede en mayúscula y el resto del mes en minúscula, vamos a realizarlo paso a paso.

Primero creamos una variable para extraer el primer carácter y luego convertimos esta letra en mayúscula. Como indicamos al inicio de este capítulo podemos anidar varias variables en el mismo código.

Agregamos una nueva línea e ingresamos el cálculo indicado.

La variable la nombramos “Y” y esta es la que calculamos después del RETURN.

The screenshot shows the Power BI Data Editor interface. At the top, there is DAX code:

```
1 Mes = VAR X =FORMAT('Calendario 2'[Date],"mmmm")
2 VAR Y= UPPER(LEFT(X))
3 RETURN
4 Y
```

The cell containing 'Y' is highlighted with a red box. A red arrow points from this cell down to the table below. The table has columns: Año, Mes Num, Trimestre, Mes, Semana, Dia Mes, and Nombre. All rows show '2017' in the Año column and '1' in the Mes Num column. The Mes column shows 'E' for Enero. The Nombre column shows 'Enero' for all rows. To the right of the table is a context menu and a search bar:

- Orden ascendente
- Orden descendente
- Borrar orden
- Borrar filtro
- Borrar todos los filtros
- Filtros de texto

Below the menu is a 'Buscar' input field. To the right is a list of filters:

- (Seleccionar todo)
- A
- D
- E
- F
- J
- M

Figura 5.3 Extraer la primera letra en mayúscula.

Ya empezamos a ver ventajas de usar variables, para extraer el primer carácter en mayúscula evitamos repetir la función FORMAT y simplemente llamamos la variable.

Debemos crear otra variable para extraer el resto del texto, es decir, para el caso de enero sería extraer después de la letra e “nero”, usaremos la función MID, es similar a la función EXTRAE en Excel, esta función tiene la capacidad de extraer datos a partir de cierta posición. La variable la nombramos “Z” y será calculada después del RETURN.

```

1 Mes = VAR X =FORMAT('Calendario 2'[Date],"mmmm")
2 VAR Y= UPPER(LEFT(X))
3 VAR Z= MID(X,2, LEN(X) )
4 RETURN
5 Z

```

	Año	Mes Num	Trimestre	Mes	Semana	Día Mes	
00:00 a. m.	2017	1	1	nero	Orden ascendente		
00:00 a. m.	2017	1	1	nero			
00:00 a. m.	2017	1	1	nero			
00:00 a. m.	2017	1	1	nero			
00:00 a. m.	2017	1	1	nero			
00:00 a. m.	2017	1	1	nero			
00:00 a. m.	2017	1	1	nero			
00:00 a. m.	2017	1	1	nero			
00:00 a. m.	2017	1	1	nero			
00:00 a. m.	2017	1	1	nero			

Figura 5.4 Extraer el resto del texto del mes.

De nuevo las variables toman poder, para extraer el resto del texto usamos dos veces la variable "X".

De esta forma casi tenemos resuelto nuestro ejercicio, solo es unir la variable "Y" y la variable "Z" después del RETURN.

```

Mes = VAR X =FORMAT('Calendario 2'[Date],"mmmm")
VAR Y= UPPER(LEFT(X))
VAR Z= MID(X,2, LEN(X) )
RETURN
Y&Z

```

	Año	Mes Num	Trimestre	Mes	Semana	Día Mes	Nombre
00 a. m.	2017	1	1	Enero	Orden ascendente		
00 a. m.	2017	1	1	Enero			
00 a. m.	2017	1	1	Enero			
00 a. m.	2017	1	1	Enero			
00 a. m.	2017	1	1	Enero			
00 a. m.	2017	1	1	Enero			
00 a. m.	2017	1	1	Enero			
00 a. m.	2017	1	1	Enero			
00 a. m.	2017	1	1	Enero			
00 a. m.	2017	1	1	Enero			

Figura 5.5 Columna mes primer carácter en mayúscula.

Comentar o documentar tu código

DAX admite comentarios en nuestros códigos, podemos escribir textos que no hacen parte de la ejecución de una expresión, de suma importancia para describir las operaciones o variables plasmadas.

Por ejemplo, podemos describir cada una de las variables que hemos creado, para comentar una sola línea solo es agregar doble slash // o doble guion --

```
1 Mes = VAR X =FORMAT('Calendario 2'[Date],"mmmm") //Extraer el nombre mes de la fecha
2 VAR Y= UPPER(LEFT(X)) -- El primero caracter en mayúscula
3 VAR Z= MID(X,2, LEN(X) ) --Extrae el resto del texto
4 RETURN
5 Y&Z
```

Figura 5.6 Comentar código DAX.

También podemos realizar comentarios de varias líneas de código iniciando con /* Comentario */

```
1 Mes = VAR X =FORMAT('Calendario 2'[Date],"mmmm") //Extraer el nombre mes de la fecha
2 VAR Y= UPPER(LEFT(X)) -- El primero caracter en mayúscula
3 VAR Z= MID(X,2, LEN(X) ) --Extrae el resto del texto
4 /* Crear comentarios de múltiples líneas
5 | para la documentación de tu código */
6 RETURN
7 Y&Z
```

Figura 5.7 Comentar múltiples líneas.

Formatear códigos DAX

Aunque hemos tratado de ser muy ordenados escribiendo DAX, debemos garantizar la legibilidad de cada medida, tabla o columna creada, hay un dicho muy famoso en este mundo, que si no está formateado no es DAX, muy cierto ya que esto da un orden y entendimiento más claro a la hora de analizar una expresión.

Existe una herramienta muy conocida llamada DAX FORMATTER, creada por el equipo de SQLBI y aprovechamos este espacio para agradecer a sus líderes **Marcos Russo y Alberto Ferrari** por todo el aporte y conocimiento que han brindado para avanzar en este maravilloso viaje de los datos especialmente en DAX.



Referencia: <https://www.daxformatter.com/>

Abrimos la página de DAX FORMATTER y copiamos todo el código anterior creado y pegamos.

Esta herramienta puede identificar si los separadores de los argumentos son coma o punto y coma, validar que tu navegador siempre esté en inglés .

The screenshot shows the DAX FORMATTER website interface. On the left, there's a logo for 'DAX FORMATTER'. The main area contains a red-bordered code editor with the following DAX code:

```
Mes = VAR X =FORMAT('Calendario 2'[Date], "mmmm") //Extaer el nombre mes de la fecha
VAR Y= UPPER(LEFT(X)) -- El primero caracter en mayúscula
VAR Z= MID(X,2, LEN(X) ) --Extrae el resto del texto
/* Crear comentarios de múltiples líneas
   para la documentación de tu código */
RETURN
Y&Z
```

Below the code editor, there are two buttons: 'BUG REPORT' and 'FORMAT'. A red arrow points from the 'FORMAT' button towards the bottom right of the interface.

Figura 5.8 Pegar código en DAX FORMMATER.

Damos clic en FORMAT y el código está listo para copiar y pegar de nuevo en Power BI.

The screenshot shows the DAX FORMATTER website interface after clicking the 'FORMAT' button. The code has been processed and now includes line numbers on the left. A red arrow points from the 'COPY' button towards the bottom right of the interface.

```
1 Mes =
2 VAR X =
3   FORMAT ( 'Calendario 2'[Date], "mmmm" ) //Extaer el nombre mes de la fecha
4 VAR Y =
5   UPPER ( LEFT ( X ) ) -- El primero caracter en mayúscula
6 VAR Z =
7   MID ( X, 2, LEN ( X ) ) --Extrae el resto del texto
8 /* Crear comentarios de múltiples líneas
   para la documentación de tu código */
9 RETURN
10 Y & Z
```

Below the code editor, there are several buttons: 'COPY' (highlighted with a red box), 'COPY HTML', 'SAVE AS DOCX', 'EDIT', and 'NEW'.

Figura 5.9 Copiar y pegar el nuevo código formateado.

Esta herramienta acepta comentarios.

El nuevo código en Power BI se ve así.

The screenshot shows the Power BI DAX code visual. The code is identical to the one shown in Figure 5.9, with line numbers and comments. A red arrow points from the 'COPY' button towards the bottom right of the interface.

```
1 Mes =
2 VAR X =
3   FORMAT ( 'Calendario 2'[Date], "mmmm" ) //Extaer el nombre mes de la fecha
4 VAR Y =
5   UPPER ( LEFT ( X ) ) -- El primero caracter en mayúscula
6 VAR Z =
7   MID ( X, 2, LEN ( X ) ) --Extrae el resto del texto
8 /* Crear comentarios de múltiples líneas
   para la documentación de tu código */
9 RETURN
10 Y & Z
```

Figura 5.10 Código formateado.



Nota: DAX FORMATTER no es una herramienta para arreglar errores de códigos ni mejorar rendimientos, solo es un tema visual.

Veamos la gran diferencia si creamos esta misma función sin variables y sin formateo.

```
= UPPER( LEFT( FORMAT ( 'Calendario 2'[Date], "mmmm" ) ) & MID( FORMAT ( 'Calendario 2'[Date], "mmmm" ),2,LEN( FORMAT ( 'Calendario 2'[Date], "mmmm" ) ))
```

Figura 5.11 Código sin variable y sin formato.

Ves la diferencia, muy difícil de comprender un cálculo así.

Comprender que las variables se calculan una sola vez

Al inicio hicimos algún comentario que las variables solo se ejecutan una sola vez, y es cuando se evalúa en la definición VAR, y esto no es fácil de identificar cuando estamos escribiendo medidas que nos arrojan resultados erróneos.

Suena extraño, pero las variables se convierten en una constante después de ser indicadas por VAR, por ejemplo, queremos realizar un comparativo de las ventas, donde cada año se compare con el año máximo de la tabla calendario, para esto primero debemos indicarle a DAX cuál es el año máximo que tenemos en ventas y luego que este año se repita en cada uno de los registros.

Vamos a paso a paso, primero creamos una medida dentro de una variable que nos calcule el año máximo de la matriz que venimos trabajando.

Creamos la medida y la nombramos [Año Max] y la arrojamos a la matriz, el resultado es el siguiente.

The screenshot shows a DAX editor window with the following code:

```
1 Año Max =
2 VAR Year_Max =
3     MAX ( Calendario[Año] )
4 RETURN
5 Year_Max
```

A red arrow points from the code line '5 Year_Max' to a table visualization below. The table has three columns: 'Año', 'Ventas', and 'Año Max'. The 'Año Max' column contains the value '2017' for all rows, indicating a calculation issue.

Año	Ventas	Año Max
2017	21.662.728.028	2017
2018	24.484.312.974	2018
2019	24.684.311.343	2019
2020	25.946.176.370	2020
Total	96.777.528.715	2020

Figura 5.12 Año máximo.

El resultado es correcto, MAX es una función de agregación y tiene en cuenta el contexto de filtro, queremos que el año 2020 se repita en cada uno de los registros para crear el cálculo comparativo de cada año con el año máximo, para esto usaremos CALCULATE.



Nota: Tenemos un capítulo entero dedicado a CALCULATE y evaluación de contextos, así que no te preocupes sino entiendes la siguiente fórmula.

Creamos una nueva variable para hacer referencia a la variable anterior y modificar el contexto de años.

```
1 Año_Max =
2 VAR Year_Max =
3     MAX ( Calendario[Año] )
4 VAR Repetir_Year_Max =
5     CALCULATE ( Year_Max, ALL ( Calendario[Año] ) )
6 RETURN
7     Repetir_Year_Max
```

Año	Ventas	Año Max
2017	21.662.728.028	2017
2018	24.484.312.974	2018
2019	24.684.311.343	2019
2020	25.946.176.370	2020
Total	96.777.528.715	2020

Figura 5.13 Omitir contexto del año.

¿Por qué el resultado es el mismo?, se supone que ALL omite el contexto para que se repita el año en todos los registros, la explicación es sencilla, CALCULATE no puede procesar esta variable que ya fue ejecutada una única vez anteriormente.

Si modificamos el cálculo y en vez de la variable llamada por CALCULATE, insertamos directamente MAX (Calendario [Año]).

```
1 Año_Max =
2 VAR Repetir_Year_Max =
3     CALCULATE ( MAX ( Calendario[Año] ), ALL ( Calendario[Año] ) )
4 RETURN
5     Repetir_Year_Max
```

Año	Ventas	Año Max
2017	21.662.728.028	2020
2018	24.484.312.974	2020
2019	24.684.311.343	2020
2020	25.946.176.370	2020
Total	96.777.528.715	2020

Figura 5.14 Año máximo repetido en cada registro.

Ahora el cálculo es el correcto, y la explicación es que ya CALCULATE pudo procesar el MAX (Calendario [Año]), antes estaba siendo ejecutada por la variable VAR Year_Max.

Si deseamos ingresar este cálculo del valor máximo del año, lo podemos primero ingresar en una medida y luego llamamos esta medida en CALCULATE.

Creemos primero una medida llamada [Año Máximo]

The screenshot shows the Power BI Data View interface. At the top, there is a formula bar with the text "Año Máximo = MAX(Calendario[Año])". Below the formula bar, the "Medidas Variables" pane is open, displaying two measures: "Año Max" and "Año Máximo". The "Año Máximo" measure is highlighted with a red border. The "Año Max" measure is shown with its formula: "MAX(Calendario[Año])".

Figura 5.15 Medida año máximo.

Luego volvemos a nuestro cálculo, y modificamos el código de tal manera que CALCULATE va a ser referencia a la medida.

The screenshot shows the Power BI Data View interface. On the left, the DAX code is displayed:

```
1 Año Max =
2 VAR Repetir_Year_Max =
3     CALCULATE ( [Año Máximo], ALL ( Calendario[Año] ) )
4 RETURN
5     Repetir_Year_Max
```

A red arrow points from the line "5 Repetir_Year_Max" down to a table visual on the right. The table has three columns: "Año", "Ventas", and "Año Max". The data is as follows:

Año	Ventas	Año Max
2017	21.662.728.028	2020
2018	24.484.312.974	2020
2019	24.684.311.343	2020
2020	25.946.176.370	2020
Total	96.777.528.715	2020

Figura 5.16 Año máximo con medida.

Para concluir siempre acostúmbrate a crear variables y saber cómo usarlas y no te olvides de formatear tu código DAX.

Capítulo 6: Principales funciones de tabla

DAX tiene la posibilidad de crear tablas como lo vimos en los tipos de cálculo, hay algunas funciones que lo permiten, en este capítulo veremos las principales y las más usadas para crear cálculos de alto valor, el objetivo primordial de esta sesión es que entendamos el manejo y resultado que arroja cada una de estas funciones y su aplicabilidad, pero no debemos perder de vista que todo debe ir en medidas.

No importa si al inicio de tu modelo estas lleno de una cantidad de tablas creadas por medio de estas funciones, pues es lo normal en la curva de aprendizaje, ya que cuando creamos medidas al principio no entendemos lo que está detrás de cámaras y más aún si las medidas tienen implícitas funciones de tabla.

En estas funciones hay mucho potencial para realizar análisis que darán mucha información para la toma de decisiones, por ejemplo, saber que clientes no han vuelto a comprar, que clientes son más recurrentes, clientes nuevos, que productos componen cada línea y mucho más.

VALUES

Esta función tiene un único argumento, recibe una tabla o una columna, se usa para devolver los valores únicos de una columna, si en los datos la columna contiene valores en blanco, VALUES los tiene en cuenta y lo agrega como parte de los registros.

Creemos una tabla que nos arroje como resultado los clientes únicos de la tabla ventas.

Pestaña Datos → Inicio → Cálculos → Nueva tabla.



Figura 6.1 Crear nueva tabla.

Tal como lo venimos estudiando, ingresamos la siguiente función, nombrando la tabla como *Fx Values*.

Hacemos referencia a la tabla *Ventas* y columna *Nombre Cliente*.

The screenshot shows the Power BI Data View interface. At the top, there is a formula bar with the text "1 Fx Values = VALUES(Ventas[Nombre Cliente])". Below the formula bar is a table with a single column labeled "Nombre Cliente". The table lists 60 unique company names, starting with "LABORATORIOS FRANCO COLOMBIANO LAFRANCOL" and ending with "PAPELES NACIONALES S.A.". The table has a red border around its content area. At the bottom of the table, there is a footer bar with the text "Fx Values (60 filas)". A red arrow points from the text "VALUES(Ventas[Nombre Cliente])" in the formula bar down to the "Nombre Cliente" column header in the table.

Nombre Cliente
LABORATORIOS FRANCO COLOMBIANO LAFRANCOL
HIERBAS Y PLANTAS TROPICALES S.A.S
INTERNACIONAL GIL S.A.S
JOHN RESTREPO A. Y CIA S.A
NUTRIMENTI DE COLOMBIA S A S
EMBOTELLADORA DE BEBIDAS DEL TOLIMA S.A
MOLINOS DEL ATLANTICO S.A
CRUJIAREPAS
COMERCIALIZADORA QFC S A S
INDUSTRIA DE ASEO LIMPIAHOGAR SAS
LA FABRIL S.A.
ILUMAX S.A.
CIPA S.A
LABORATORIO MAXPROTEC S.A.
CONSERMAR S.A.S
COLOMBIA CIPE S.A.S
TEAM FOODS COLOMBIA S.A.
PAPELES NACIONALES S.A.

Figura 6.2 Clientes únicos.

El resultado son 60 clientes únicos que han comprado en toda la historia de la empresa.

Cuando el parámetro es una tabla en la función VALUES, devuelve todos los registros de la misma tabla.

ALL

Es una de estas funciones que sirve como tabla y como modificador (tema que abarcaremos más adelante).

En el ejemplo anterior se evidenció los clientes únicos, si queremos consultar cada cliente que códigos de línea compra, VALUES no nos sirve, solo admite una columna, con ALL podemos resolver este inconveniente ya que puede traer las combinaciones existentes para este caso entre el cliente y el código de línea.

Creamos una nueva tabla y la llamamos Fx ALL → Con ALL hacemos referencia a las columnas Cliente y código línea.

X	✓	1 Fx ALL = ALL(Ventas[Nombre Cliente],Ventas[Cod Lineas])
		Nombre Cliente Cod Lineas
LABORATORIOS FRANCO COLOMBIANO LAFRANCOL	CA - 01	
HIERBAS Y PLANTAS TROPICALES S.A.S	CA - 01	
INTERNACIONAL GIL SAS	CA - 01	
JOHN RESTREPO A. Y CIA S.A.	CA - 01	

Figura 6.3 Combinaciones existentes entre cliente y código de línea.

Ya se registran 291 filas



Advertencia: ALL solo admite columnas de la misma tabla, en caso de intentar hacer combinaciones entre columnas de diferentes tablas, arrojará un error.

En caso de necesitar combinación entre columnas de diferentes tablas vamos a usar otro tipo de funciones.

Como empresa usamos mucho esta función para crear maestros y poder relacionar tablas transaccionales.

Diferencia entre VALUES y ALL en uso de medidas

Como el objetivo es llevar estos cálculos a medidas, en nuestro modelo tenemos un cálculo que cuenta el número de clientes únicos por año o por cualquier campo llevado a la matriz, eso lo hicimos con DISTINCTCOUNT, pero si deseamos ver la diferencia entre VALUES y ALL operando en medidas, podemos crear este cálculo con estas dos funciones.

Analicemos de nuevo el caso, creamos una matriz con el nombre de la línea de la tabla Línea, y la medida [#Clientes]

Linea	#Clientes
ALIMENTO	58
ASEO	57
BEBIDAS	59
CAFÉ	60
CONDIMENTOS	57
Total	60

Filas

Columnas

Valores

Figura 6.4 Número de clientes por línea.

El análisis es el número de clientes que hay en cada línea. Creemos la medida que nos dará el mismo resultado usando VALUES.

```
1 #Values = COUNTROWS( VALUES(Ventas[Nombre Cliente]) )
```

Linea	#Clientes	#Values
ALIMENTO	58	58
ASEO	57	57
BEBIDAS	59	59
CAFÉ	60	60
CONDIMENTOS	57	57
Total	60	60

Figura 6.5 # clientes con VALUES.

Es exactamente el mismo resultado que usar DISTINCTCOUNT, incluso DISTINCTCOUNT es la evolución de la medida creada.



Advertencia: VALUES debe estar encerrada entre COUNTROWS ya que el resultado de una medida debe ser un único valor o un valor escalar

Desarrollemos esta misma medida, pero en vez de usar VALUES usamos ALL.

```
1 #ALL = COUNTROWS( ALL(Ventas[Nombre Cliente]) )
```

Linea	#Clientes	#Values	#ALL
ALIMENTO	58	58	60
ASEO	57	57	60
BEBIDAS	59	59	60
CAFÉ	60	60	60
CONDIMENTOS	57	57	60
Total	60	60	60

Figura 6.6 # clientes con ALL.

El resultado es totalmente diferente, repite el mismo valor en cada registro, esta es la gran diferencia entre estas dos funciones, ALL omite el contexto de filtro, VALUES lo tiene en cuenta, por eso el resultado es el total clientes en cada línea.

Recomendamos usar ALL como tabla o como modificador de CALCULATE, en otras funciones genera resultados como estos.

CROSSJOIN

CROSSJOIN realiza todas las combinaciones existentes entre tablas, muy usado para crear cruces de información que no necesariamente se encuentran en la misma consulta, ALL solo puede crear combinaciones existentes de la misma tabla, si necesitamos por ejemplo el nombre del cliente y el nombre de la línea (no el código), CROSSJOIN puede hacerlo.

El nombre de la línea está en la tabla LINEA y el nombre del cliente en VENTAS.

CROSSJOIN tiene solo un argumento obligatorio, los otros son las tablas opcionales para crear el cruce.

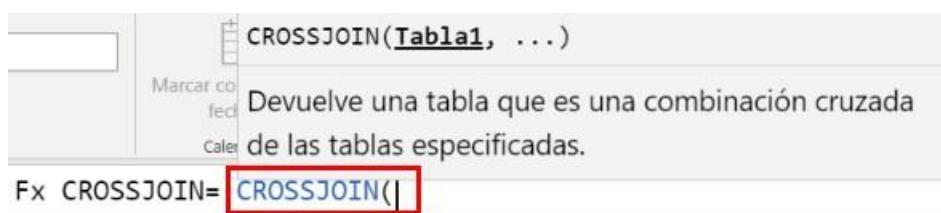


Figura 6.7 Argumentos de CROSSJOIN.

Como recibe una tabla completa y solo deseamos es el nombre cliente y nombre línea, nos apoyamos con ALL para extraer solo esta columna de cada tabla, la función sería así.

The screenshot shows the Power BI formula bar with the following text:
Fx CROSSJOIN = CROSSJOIN(**ALL(Ventas[Nombre Cliente]), ALL(Linea[Linea])**)
The arguments 'ALL(Ventas[Nombre Cliente])' and 'ALL(Linea[Linea])' are highlighted with red and blue boxes respectively. Below the formula bar, a preview table is shown:

Línea	Nombre Cliente
CAFÉ	LABORATORIOS FRANCO COLOMBIANO LAFRANCOL
CAFÉ	HIERBAS Y PLANTAS TROPICALES S.A.S
CAFÉ	INTERNACIONAL GIL SAS
CAFÉ	JOHN RESTREPO A. Y CIA S.A
CAFÉ	NUTRIMENTI DE COLOMBIA S A S

Figura 6.8 Combinaciones existentes entre nombre cliente y línea.

Este resultado es un producto cartesiano, es decir, si en la data hay 5 tipo de líneas, a cada cliente le asigna una de ellas, podemos agregar más campos para crear los posibles cruces entre ellos.

Es útil estos resultados cuando es necesario aplicar filtros en CALCULATE que impliquen campos de diferentes tablas.

SUMMARIZE

Esta función de tabla nos permite crear totales resumidos para un conjunto de datos.

Es una de estas funciones que hacen parte de la colección de DATAICE ya que nos permite crear cálculos de suma importancia combinadas con otras funciones. Iniciemos como es su uso.

SUMMARIZE tiene un solo argumento obligatorio y es una tabla, este es el más importante no porque es el que debe ir obligadamente, sino porque debemos escoger que tipo de tabla llevar, de esto depende el resumen agrupado por los otros campos.

Si volvemos a nuestro modelo relacional, todas las tablas apuntan hacia la tabla VENTAS, lo que en el segundo capítulo de este libro explicamos la importancia de este tipo de modelos ya que implícitamente la tabla VENTAS contiene todas las columnas de las tablas con las que se encuentran relacionadas y no debes perder esto de vista.

Gracias a estas ventajas de las relaciones, podemos crear un resumen de los datos con SUMMARIZE de las ventas por año y línea.

Creamos una nueva tabla → La nombramos Fx SUMMARIZE y observemos la sintaxis de la función.

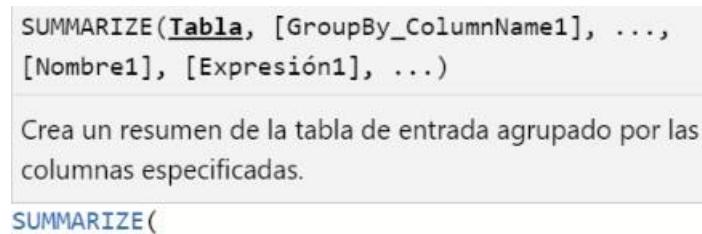


Figura 6.9 Argumentos de SUMMARIZE.

Después de recibir la tabla como argumento obligatorio para este caso sería VENTAS, hay varios argumentosopcionales, que son las columnas por las que se va a agrupar para nuestro ejercicio es año y línea.

The screenshot shows the Power BI Data View. At the top, there is a formula bar with the following text:

```
X ✓ 1 Fx SUMMARIZE = SUMMARIZE(Ventas, Calendario[Año], Linea[Linea])
```

Below the formula bar is a table with two columns: "Año" and "Linea". The data in the table is:

Año	Linea
2017	CAFÉ
2017	ALIMENTO
2017	BEBIDAS
2017	CONDIMENTOS
2017	ASEO

A red arrow points from the formula bar down towards the table, indicating the relationship between the function and its data source.

Figura 6.10 Resumen año y línea.

Hasta este punto es muy parecido al resultado que nos arroja CROSSJOIN, pero SUMMARIZE tiene una gran diferencia y es que resume estas combinaciones, pero existentes, es decir, solo trae las combinaciones entre año y línea pero que tuvieron venta, ejemplo, tenemos 5 líneas, pero si en un año una de ellas no tuvo transacción no la trae a la agrupación.

También podemos agregarle los otros argumentos de esta función [Nombre1] y [Expresión1], es el cálculo que deseamos añadirle a este resumen.

En [Nombre1] es un nombre opcional, por ejemplo “Total Ventas”, y la [Expresión1] es la operación aritmética, SUM (Ventas [ingresos]). No olvides formatear el código.

The screenshot shows the Power BI formula editor with the following code:

```
X ✓ 1 Fx SUMMARIZE =  
2 SUMMARIZE (  
3     Ventas,  
4     Calendario[Año],  
5     Linea[Linea],  
6     "Total Ventas", SUM ( Ventas[Ingresos] )  
7 )
```

Below the editor is a table visualization with three columns: Año, Linea, and Total Ventas. The data is as follows:

Año	Linea	Total Ventas
2017	CAFÉ	2344526313
2017	ALIMENTO	6036663201
2017	BEBIDAS	8639627296
2017	CONDIMENTOS	1471172824
2017	ASEO	3170738394
2018	CAFÉ	1914242953

Figura 6.11 Total ventas por año y línea.



Advertencia: Este proceso de ver los datos agrupados, en este caso “Total Ventas” es ambigua, no la recomendamos, solo es útil para hacer consultas rápidas o validaciones.

Para ejemplos prácticos, mostramos como totalizar estos datos, pero no es recomendable hacer esta operación para ser procesados en una medida, solo hasta las agrupación y combinación de columnas es recomendable.

SUMMARIZE tiene en cuenta lo que se conoce como tablas expandidas, aunque el argumento inicial fue la tabla VENTAS, recibe campos de diferentes tablas como es AÑO y LÍNEA, hay mucho potencial allí.

SUMMARIZE es de nuestras funciones de tabla favoritas ya que solucionamos problemas de datos agrupados que necesitamos calcular en medidas, promedios consolidados por año, mes o cualquier

periodo, guarde esta función como favorita, y más adelante en esta sesión la vamos a combinar con ADDCOLUMNS para solucionar problemas del día a día.

ADDCOLUMNS

Su resultado es una tabla con nuevas columnas, a partir de una tabla como argumento añade más columnas con nombres opcionales.

El modelo que estamos trabajando contiene una tabla llamada *Calendario*, la cual fue construida previamente para los ejemplos explicativos, esta tabla es un resumen de lo que es ADDCOLUMNS, en este punto la entenderás.

The screenshot shows the Power BI Data Editor interface. On the left, there is a code editor window with the following DAX code:

```
X ✓
1 Calendario =
2 ADDCOLUMNS (
3     CALENDAR ("01/01/2017","31/12/2020"),
4     "Año", YEAR ( [Date] ),
5     "Mes Num", MONTH ( [Date] ),
6     "Mes",
7         VAR X =
8             FORMAT ( [Date], "mmmm" ) // Variables para poner en mayúscula la primera letra del mes
9         VAR Y =
10            UPPER ( LEFT ( X ) ) //Se extrae la primera letra y se convierte en mayúscula
11         VAR Z =
12            MID ( X, 2, LEN ( X ) ) //Se extraer el resto del texto
13         RETURN
14             Y & Z // Se concatena la primera letra y el resto del texto
15 , "Trimestre", "T"&QUARTER([Date])
16 , "Semestre", IF(MONTH([Date])<=6,"Sem 1","Sem 2")
17 , "Semana", WEEKNUM([Date],2)
18 , "Dia Mes", DAY([Date])
19 , "Dia Semana", WEEKDAY([Date],2)
20 , "Nombre Dia", FORMAT([Date],"dddd")
21 )
```

To the right of the code editor is a preview of the generated table. The table has the following columns and data:

Date	Año	Mes Num	Mes	Trimestre	Semestre	Semana	Dia Mes	Dia Semana	Nombre Dia
1/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	27	1	6	sábado
2/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	27	2	7	domingo
3/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	28	3	1	lunes
4/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	28	4	2	martes
5/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	28	5	3	miércoles
6/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	28	6	4	jueves
7/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	28	7	5	vier
8/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	28	8	6	jueves

Figura 6.12 Tabla calendario.

ADDCOLUMNS inicia recibiendo una tabla, que es CALENDAR y luego se van añadiendo columnas adicionales.

ADDCOLUMNS tiene tres argumentos obligatorios: Tabla, Nombre1 y Expresión1.... y el ciclo del Nombre y Expresión se repite.

Creamos una nueva tabla → La nombramos Fx ADDCOLUMNS → La tabla inicial puede ser un CALENDARAUTO y le añadimos las columnas para desgranar esta tabla, iniciemos calculando el año.

X	✓	1 Fx ADDCOLUMNS =
		2 ADDCOLUMNS (CALENDARAUTO (), "Año", YEAR ([Date]))
	3	
		Date Año
		1/01/2020 12:00:00 a. m. 2020
		2/01/2020 12:00:00 a. m. 2020
		3/01/2020 12:00:00 a. m. 2020
		4/01/2020 12:00:00 a. m. 2020
		5/01/2020 12:00:00 a. m. 2020

Figura 6.13 Tabla calendario con año.

La columna *Date* es el resultado de CALENDARAUTO, vamos nombrando las columnas que se van a ir añadiendo, en este caso “Año” y la función que extraerá el año, si notas YEAR solo tiene como argumento [Date] lo que significa que para este punto no se referencia tabla y columna, solo columna porque la tabla es implícita.

Tienes una tarea de terminar la tabla añadiendo el mes número, mes nombre, trimestre, semana etc. Puedes aplicar y practicar en este punto variables y documentar tu código.

Caso práctico ADDCOLUMNS y SUMMARIZE

Retomemos el caso que estudiamos en el capítulo 4, como calcular el promedio de venta mes en cada uno de los años, es decir, si el total de ventas en un año fue de 2.400 el promedio venta mes son 200.

Si creamos una medida AVERAGE (Ventas [Ingresos]) y la llevamos a una matriz por año el resultado es el siguiente.

1 Venta Promedio = AVERAGE(Ventas[Ingresos])		
Año	Ventas	Venta Promedio
2017	21.662.728.028	1.429.694
2018	24.484.312.974	1.658.379
2019	24.684.311.343	1.693.722
2020	25.946.176.370	1.758.109
Total	96.777.528.715	1.633.431

Figura 6.14 Medida venta promedio.

Analizemos 2019, el total de ventas son 24.684.311.343 si dividimos esto entre 12 meses el resultado es: 2.057.025.945 y si observamos el resultado de la medida promedio es 1.693.722 totalmente erróneo como lo explicamos anteriormente.

El reto se encuentra en indicarle a DAX que primero me agrupe las ventas por mes y luego realice el promedio de estos meses agrupados, pues SUMMARIZE con ADDCOLUMNS es una buena opción.

Primero creamos una nueva tabla con SUMMARIZE por mes.

Pestaña Datos → Inicio → Nueva tabla

The screenshot shows the Power BI Data Editor interface. A red arrow points from the text "2 SUMMARIZE(Ventas,Calendario[Mes])" in the formula bar down to the "Mes" column in the table below. The table has columns for Mes (Month) and Ventas Mes (Sales Month). The rows show data for Enero, Febrero, Marzo, Abril, and Mayo.

Mes	Ventas Mes
Enero	
Febrero	
Marzo	
Abril	
Mayo	

Figura 6.15 Tabla mes nombre resumen.

El siguiente paso es añadir una columna con ADDCOLUMNS, llamada “@Total Ventas”, que nos acumule las ventas por mes, hacemos referencia a la medida [Ventas] no al cálculo SUM.

Recomendamos usar el símbolo @ para nombrar columnas en tablas virtuales que serán usadas en medidas, es una práctica que recomiendan también expertos para evitar conflictos en nombres de otros campos.

The screenshot shows the Power BI Data Editor interface. A red box highlights the formula bar where the code "3 SUMMARIZE(Ventas,Calendario[Mes]),[@Total Ventas],[Ventas])" is entered. Another red box highlights the "Mes" and "@Total Ventas" columns in the table below. The table shows monthly sales data for Enero through Mayo.

Mes	@Total Ventas
Enero	5826850303
Febrero	7441745623
Marzo	8145130232
Abril	8171430456
Mayo	8540042641

Figura 6.16 Ventas agrupadas por mes.

Ya tenemos el mayor proceso solucionado, lo único que nos queda es crear una medida con que itere dicha columna y realice el promedio de los meses recorridos, ¿Qué función crees que lo puede hacer?

Creemos una medida nueva e ingresamos este código que genera la tabla agrupada dentro de una variable y retornamos esta variable con el iterador recorriendo “@Total Ventas”.

Detallar la siguiente imagen con la medida y el resultado.

```
1 Ventas Prom Real =
2 VAR VentasMes =
3     ADDCOLUMNS ( SUMMARIZE ( Ventas, Calendario[Mes] ), "@Total Ventas", [Ventas] )
4 RETURN
5     AVERAGEX ( VentasMes, [@Total Ventas] )
```

Año	Ventas	Venta Promedio	Ventas Prom Real
2017	21.662.728.028	1.429.694	1.805.227.336
2018	24.484.312.974	1.658.379	2.040.359.415
2019	24.684.311.343	1.693.722	2.057.025.945
2020	25.946.176.370	1.758.109	2.162.181.364
Total	96.777.528.715	1.633.431	8.064.794.060

Figura 6.17 Promedio mensual real.

Analicemos 2019, el resultado es el mismo que debe arrojar si dividimos total 2019 entre los 12 meses.



```
Ventas Prom Real =
VAR VentasMes =
    ADDCOLUMNS ( SUMMARIZE ( Ventas, Calendario[Mes] ),
"@Total Ventas", [Ventas] )
RETURN
    AVERAGEX ( VentasMes, [@Total Ventas] )
```



Nota: Hacemos referencia la medida [Ventas] en ADDCOLUMNS, ya que hace una operación llamada transición de contexto, tema que explicaremos más adelante, para estos ejercicios debemos hacer referencia a las medidas y no crear los cálculos directamente a no ser que se encuentre encerrado en un CALCULATE.

UNION

Hemos agregado a nuestro modelo una tabla de clientes que tienen presupuesto, en esta tabla se encuentran clientes que sí y no se le han generado ventas. UNION permite unir varias tablas, pero con el mismo número de columnas, es decir, cada tabla debe tener la misma estructura, no necesariamente el mismo nombre en las columnas, sino exactamente el número de columnas.

Clientes
IMPRESOS FLEXOGRAFICOS
HIERBAS Y PLANTAS TROPICALES S.A.S
INTERNACIONAL GIL SAS
JOHN RESTREPO A. Y CIA S.A
NUTRIMENTI DE COLOMBIA S A S
DATAICE SAS
MOLINOS DEL ATLANTICO S.A
CRUJIAREPAS
DEPOSITOS MIRANDA
INDUSTRIA DE ASEO LIMPIAHOGAR SAS
LA FABRIL S.A.
ILUMAX S.A.
CIPA S.A

> Medidas Agregación
> Medidas Iteración
> Medidas Variables
> Calendario
> Calendario 2
> Clientes_Presupuesto ⓘ ..
Clientes
> Departamentos

Figura 6.18 Tabla clientes presupuesto.

Esta tabla cuenta con 20 clientes, con UNION podemos unir las dos tablas con la misma estructura para que me consolide los clientes *Ventas* y *Presupuesto*.

Creamos una nueva tabla nombrada Total Clientes e ingresamos el siguiente cálculo.

Clientes
IMPRESOS FLEXOGRAFICOS
HIERBAS Y PLANTAS TROPICALES S.A.S
INTERNACIONAL GIL SAS
JOHN RESTREPO A. Y CIA S.A
NUTRIMENTI DE COLOMBIA S A S
DATAICE SAS
MOLINOS DEL ATLANTICO S.A
CRUJIAREPAS
DEPOSITOS MIRANDA
INDUSTRIA DE ASEO LIMPIAHOGAR SAS
LA FABRIL S.A.
ILUMAX S.A.
CIPA S.A

Total Clientes (90 filas)

Figura 6.19 Total clientes.

El resultado son 90 clientes según indica la imagen, la explicación es, los 60 clientes en ventas y los 30 clientes de presupuesto.

Presta mucha atención a la estructura de la fórmula, la tabla de clientes presupuesto solo contaba con una única columna, los clientes de la tabla VENTAS tiene múltiples columnas, para cumplir con la sintaxis de UNION usamos VALUES que devuelve un único campo.



Total Clientes =

```
UNION ( Clientes_Presupuesto, VALUES ( Ventas[Nombre Cliente] ) )
```

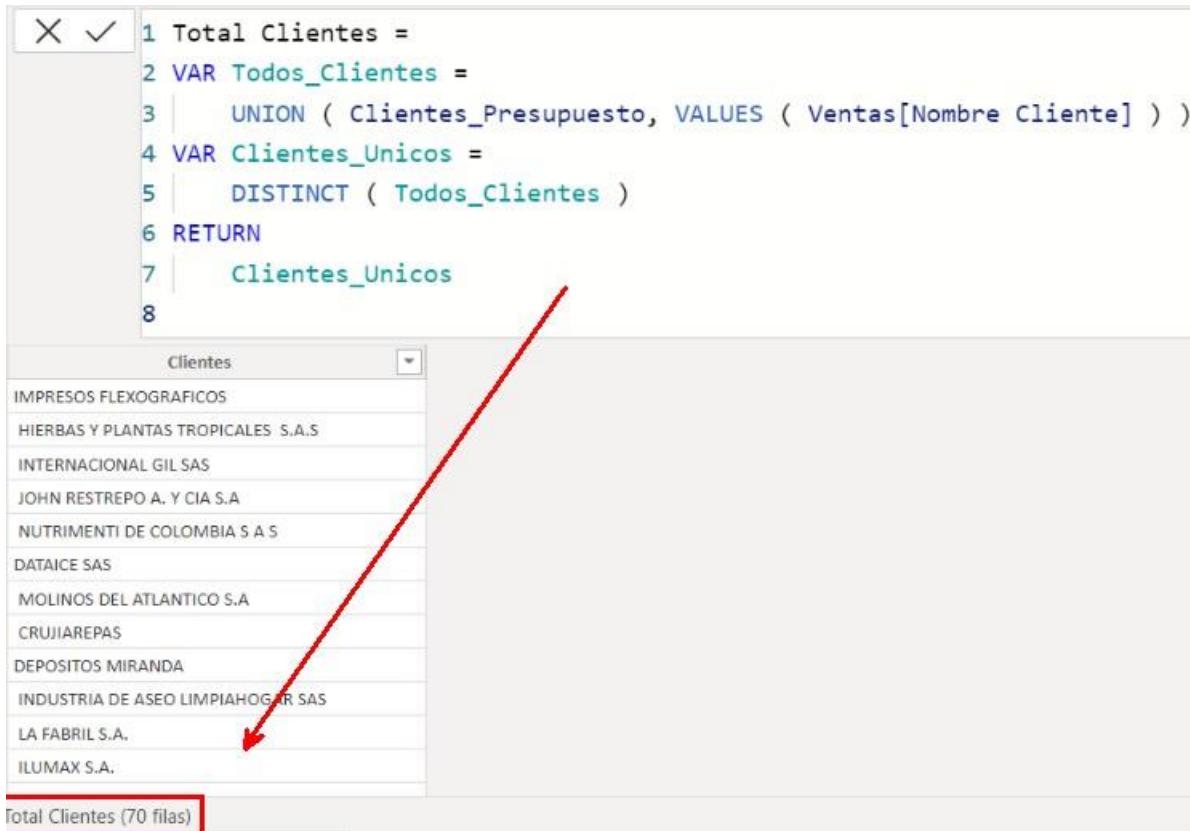
DISTINCT

En el ejercicio anterior tenemos 90 clientes, pero entre este resultado puede haber registros duplicados, DISTINCT recibe una única tabla o columna como argumento y devuelve los valores únicos de dicha tabla o columna.

Podemos crear el cálculo en la misma tabla anterior usando variables.

Ingresamos en una variable el cálculo de UNION, y en otra variable el cálculo DISTINCT haciendo referencia a la anterior variable para que así solo traiga los valores únicos de dicha tabla.

La siguiente figura nos muestra el proceso.



```

X ✓ 1 Total Clientes =
2 VAR Todos_Clientes =
3 UNION ( Clientes_Presupuesto, VALUES ( Ventas[Nombre Cliente] ) )
4 VAR Clientes_unicos =
5 DISTINCT ( Todos_Clientes )
6 RETURN
7 Clientes_unicos
8

```

Cuentas

Clientes
IMPRESOS FLEXOGRAFICOS
HIERBAS Y PLANTAS TROPICALES S.A.S
INTERNACIONAL GIL S.A.S
JOHN RESTREPO A. Y CIA S.A
NUTRIMENTI DE COLOMBIA S A S
DATAICE SAS
MOLINOS DEL ATLANTICO S.A
CRUJIAREPAS
DEPOSITOS MIRANDA
INDUSTRIA DE ASEO LIMPIAHOGAR SAS
LA FABRIL S.A.
ILUMAX S.A.

Total Clientes (70 filas)

Figura 6.20 Cliente únicos.

Si analizamos el resultado, estos 70 clientes corresponden a 60 clientes en ventas y 10 clientes en presupuesto, lo que significa que 20 clientes que estaban en presupuesto también correspondían a la tabla ventas.

Este proceso es muy común en los desarrollos que realizamos como empresa consultora que somos, cuando se presentan este tipo de casos, donde debemos sacar los valores únicos de dos columnas para crear una tabla maestra y poder generar la relación entre presupuestos y ventas y que se encuentren todos los clientes o registros únicos este es el camino.



```

Total Clientes =
VAR Todos_Clientes =
    UNION ( Clientes_Presupuesto, VALUES ( Ventas[Nombre Cliente] ) )
VAR Clientes_unicos =
    DISTINCT ( Todos_Clientes )
RETURN
    Clientes_unicos

```

INTERSECT

Devuelve el intercepto entre dos tablas y se conservan los valores duplicados en caso de existir entre ellas. Continuando con el ejemplo anterior, esta función nos permite saber que clientes están presupuestados y cuáles de ellos han generado ventas, es decir, los 20 clientes faltantes de la tabla anterior.

Creamos una nueva tabla → Nombrada Clientes Interceptados → Primero creamos dos variables → Primera para extraer los clientes únicos de las ventas → Segunda extraer los clientes únicos del presupuesto.

Como muestra la imagen 6.21

```
1 Clientes Interceptados =  
2 VAR Clientes_Ventas= VALUES(Ventas[Nombre Cliente])  
3 VAR Clientes_Ppto= VALUES(Clientes_Presupuesto[Clientes])
```

Figura 6.21 Declarar variables de clientes.

Por último, creamos una variable para la función INTERSECT, la cual creará la intercepción entre las dos variables anteriores de *Clientes_Ventas* y *Clientes_Ppto*.

```
1 Clientes Interceptados =  
2 VAR Clientes_Ventas =  
3 | INTERSECT(LeftTable, RightTable)  
4 VAR  
5 | Devuelve las filas de la tabla de la izquierda que aparecen en  
6 | la tabla de la derecha  
7 | INTERSECT ( Clientes_Ventas,| Clientes_Ppto
```

Figura 6.22 Sintaxis función INTERSECT.

INTERSECT cuenta con dos argumentos obligatorios, para este caso el orden es indiferente, si primero van los clientes de las ventas o del presupuesto o viceversa, ya que contamos con valores únicos en cada tabla.

Completamos nuestra función y el resultado es el siguiente.

The screenshot shows a Power BI query editor window. At the top, there is a toolbar with a close button (X), a checkmark button, and a refresh button. Below the toolbar is a code editor containing the following DAX query:

```

1 Clientes Interceptados =
2 VAR Clientes_Ventas =
3     VALUES ( Ventas[Nombre Cliente] )
4 VAR Clientes_Ppto =
5     VALUES ( Clientes_Presupuesto[Clientes] )
6 VAR Interseccion =
7     INTERSECT ( Clientes_Ventas, Clientes_Ppto )
8 RETURN
9
10 Interseccion

```

Below the code editor is a table view with a header "Nombre Cliente". The table lists 20 company names:

- HIERBAS Y PLANTAS TROPICALES S.A.S
- INTERNACIONAL GIL S.A.S
- JOHN RESTREPO A. Y CIA S.A
- NUTRIMENTI DE COLOMBIA S.A.S
- MOLINOS DEL ATLANTICO S.A.
- CRUJIAREPAS
- INDUSTRIA DE ASEO LIMPIAHOJAS S.A.S
- LA FABRIL S.A.
- ILUMAX S.A.
- CIPA S.A
- LABORATORIO MAXPROTEC S.A.

A red arrow points from the text "Cuentas interceptadas" in the caption below to the "Interseccion" row in the table. The entire table area is highlighted with a red border.

Figura 6.23 Cuentas interceptadas.



```

Clientes Interceptados =

VAR Clientes_Ventas =
    VALUES ( Ventas[Nombre Cliente] )
VAR Clientes_Ppto =
    VALUES ( Clientes_Presupuesto[Clientes] )
VAR Interseccion =
    INTERSECT ( Clientes_Ventas, Clientes_Ppto )
RETURN
    Interseccion

```

EXCEPT

Algún día te han preguntado por datos que se encuentran en una base pero que no existen en la otra o viceversa, pongamos esto en mejor contexto, queremos saber que clientes del año 2021 no han comprado en 2022, o que clientes de 2022 no compraron en 2021 es decir, clientes nuevos o que

cuentas contables no se han movido de un periodo a otro y cuantos posibles escenarios y análisis nos piden realizar.

EXCEPT es la función que te salvará la vida y puedes entrar en el siguiente nivel de tus informes, por ejemplo, continuando con los ejercicios anteriores, si deseamos saber que clientes están presupuestados que aún no se les ha generado ventas o que clientes tienen venta que no se presupuestaron, son análisis diferentes que podemos concretar.

EXCEPT tiene dos argumentos obligatorios, tabla izquierda y tabla derecha, el orden en esta ocasión si importa.

Vamos a Power BI y creamos una nueva tabla → creando las mismas dos variables anteriores de clientes ventas y clientes presupuesto, ver figura 6.21.

Y definimos la variable que guardará los datos que se encuentran en una tabla y no en la otra.

```
1 Clientes No Encontrados =
2 VAR Clientes_Ventas =
3 |    VALUES ( Ventas[Nombre Cliente] )
4 VAR Clientes_Ppto =
5 |    VALUES ( Clientes_Presupuesto[Clientes] )
6 VAR No_Existentes =
7 |    EXCEPT( Clientes_Ventas, Clientes_Ppto )
```

Figura 6.24 Variables definidas.

Antes de tirar el RETURN y sacar la variable No_Existentes, piensa cual será el resultado final.

La primera tabla que usa EXCEPT es clientes ventas, es decir, revisa los clientes que se encuentran en ventas y cuáles de ellos no tienen un presupuesto asignado y el resultado es el siguiente:

The screenshot shows the Power BI Data View interface. At the top, there is a DAX query:

```

1 Clientes No Encontrados =
2 VAR Clientes_Ventas =
3     VALUES ( Ventas[Nombre Cliente] )
4 VAR Clientes_Ppto =
5     VALUES ( Clientes_Presupuesto[Clientes] )
6 VAR No_Existentes =
7     EXCEPT( Clientes_Ventas, Clientes_Ppto )
8 RETURN
9 No_Existentes

```

Below the DAX code is a table titled "Nombre Cliente" with the following data:

Nombre Cliente
LABORATORIOS FRANCO COLOMBIANO LAFRANCOL
EMBOTTELLADORA DE BEBIDAS DEL TOLIMA S.A.
COMERCIALIZADORA QFC S A S
CONSERMAR S.A.S
PAPELES NACIONALES S.A.
GROUPE SEB COLOMBIA S.A.
ABURRA LTDA.
EL CARBONERO LA MARIA J.C.R.M S.A.S.
INVERSIONES INDUSTRIALES IMPERIAL SAS
BODEGAS SANTA LUCIA S.A.S
VALENCIA VICTORIA GUILLERMO
PROCESADOS LHM S.A.S
Clientes No Encontrados (40 filas)

A red arrow points from the DAX code to the last row of the table, which is highlighted with a red box.

Figura 6.25 Clientes no encontrados en tabla presupuesto.



```

Cuentas No Encontradas =
VAR Clientes_Ventas =
    VALUES ( Ventas[Nombre Cliente] )
VAR Clientes_Ppto =
    VALUES ( Clientes_Presupuesto[Clientes] )
VAR No_Existentes =
    EXCEPT( Clientes_Ventas, Clientes_Ppto )
RETURN
No_Existentes

```

40 clientes que tienen venta no tienen aún presupuesto asignado.

El análisis puede ser también, al contrario, que DAX nos informe que clientes tienen un presupuesto que aún no se ha efectuado la venta.

Solo con cambiar de posición de las variables de *Clientes_Ventas* y *Clientes_Ppto* obtenemos el análisis

```

1 Clientes No Encontrados =
2 VAR Clientes_Ventas =
3     VALUES ( Ventas[Nombre Cliente] )
4 VAR Clientes_Ppto =
5     VALUES ( Clientes_Presupuesto[Clientes] )
6 VAR No_Existentes =
7     EXCEPT( Clientes_Ppto, Clientes_Ventas )
8 RETURN
9 No_Existentes

```

Clientes

IMPRESOS FLEXOGRAFICOS
DATAICE SAS
DEPOSITOS MIRANDA
TERPEL COLOMBIA
AUTECO
PRODUCTOS FAMILIA
INVERSIONES MEDELLIN
EXTRACTORA BPD
INDUCASCOS SAS
AGROTES
Clients No Encontrados (10 filas)

Figura 6.26 Clientes no encontrados en tabla ventas.



```

Clientes No Encontrados =
VAR Clientes_Ventas =
    VALUES ( Ventas[Nombre Cliente] )
VAR Clientes_Ppto =
    VALUES ( Clientes_Presupuesto[Clientes] )
VAR No_Existentes =
    EXCEPT( Clientes_Ppto, Clientes_Ventas )
RETURN
No_Existentes

```

10 clientes han sido presupuestados sin que tengan venta.



Nota: A nuestros estudiantes y clientes siempre les reiteramos demasiado en cuanto a estas herramientas y lenguajes, puedes tener un muy buen dominio de ellas, pero si no tienes imaginación y resolver las preguntas del negocio solo tienes conocimiento y no productividad, que es lo que las empresas necesitan y resolver sus necesidades para la buena toma de decisiones

Estas últimas funciones y sus resultados sirven como filtros en CALCULATE, cuando estemos allí vas a ver el BOOM que estas combinaciones pueden arrojar.

FILTER

FILTER es una de estas funciones del diario vivir cuando entras al mundo de DAX, aunque Microsoft la clasifica en la categoría de funciones de filtro, creemos que encaja perfectamente en este capítulo ya que su resultado final es una tabla segmentada. Esta función las verás constantemente de la mano de CALCULATE, y en algunos casos de iteradores para crear análisis de gran valor.

FILTER cuenta con dos argumentos obligatorios, tabla y la expresión filtrada, antes de entrar en materia analicemos algunas características de DAX en cuanto a las tablas.

Si creamos una nueva tabla y simplemente hacemos referencia a una de ellas sin anteponer ninguna función se crea una copia de esta.

1 Fx FILTER = Ventas														
Fecha Factura	Nombre Cliente	Cod. Vendedor	Cod. Producto	Cantidad	Precio Unit	CostoUnit	Cod Lineas	Documento	Cod Municipio	Fecha Pedido	Fecha Entrega			
17/04/2018 12:00:00 a. m.	CONSERMAR S.A.S	B-02	8928	2	0	21277	AL - 01	6582162	05360	5/03/2018 12:00:00 a. m.	2/05/2018 12:00:00 a. m.			
23/05/2018 12:00:00 a. m.	CONSERMAR S.A.S	B-02	8928	2	0	19936	AL - 01	7687393	05360	22/04/2018 12:00:00 a. m.	20/06/2018 12:00:00 a. m.			
11/05/2018 12:00:00 a. m.	URREGO LEZCANO DANIEL	B-02	8928	2	0	20042	AL - 01	0486216	05360	3/05/2018 12:00:00 a. m.	8/06/2018 12:00:00 a. m.			
14/09/2018 12:00:00 a. m.	URREGO LEZCANO DANIEL	B-02	8928	2	0	22959	AL - 01	7410121	05360	9/08/2018 12:00:00 a. m.	6/10/2018 12:00:00 a. m.			
26/09/2018 12:00:00 a. m.	URREGO LEZCANO DANIEL	B-02	8928	2	0	22903	AL - 01	1110188	05360	17/09/2018 12:00:00 a. m.	5/10/2018 12:00:00 a. m.			
29/09/2018 12:00:00 a. m.	URREGO LEZCANO DANIEL	B-02	8928	2	0	22872	AL - 01	5810210	05360	12/09/2018 12:00:00 a. m.	29/10/2018 12:00:00 a. m.			
29/09/2018 12:00:00 a. m.	DIST. DE GRANOS Y CEREALES LA CASCADA S.	B-02	8928	2	0	22872	AL - 01	6010122	05360	9/08/2018 12:00:00 a. m.	3/10/2018 12:00:00 a. m.			
23/08/2018 12:00:00 a. m.	CONSERMAR S.A.S	B-02	8928	2	0	22818	AL - 01	5398368	05360	24/07/2018 12:00:00 a. m.	31/08/2018 12:00:00 a. m.			
7/09/2018 12:00:00 a. m.	VALENCIA VICTORIA GUILLERMO	B-02	8928	2	0	22953	AL - 01	8110111	05360	19/07/2018 12:00:00 a. m.	18/09/2018 12:00:00 a. m.			
25/07/2018 12:00:00 a. m.	COMERCIALIZADORA QFC S A S	B-02	8928	2	0	22717	AL - 01	9494396	05360	31/05/2018 12:00:00 a. m.	9/08/2018 12:00:00 a. m.			
5/12/2018 12:00:00 a. m.	INVERSIONES POMAR ROA & CIA S.A.S	B-02	8928	2	0	22882	AL - 01	4711234	05360	19/11/2018 12:00:00 a. m.	9/12/2018 12:00:00 a. m.			

Figura 6.27 Copia tabla Ventas.

Trae exactamente las mismas filas y columnas de la tabla Ventas, pero FILTER tiene la capacidad de iterar fila a fila e ir validando alguna de las condiciones expresadas en el segundo argumento.

Si queremos reducir la tabla, que solo agregue los registros cuyo *Precio Unit* sea mayor a cero. Probemos con FILTER.

Ventas, sería la tabla como el primer argumento de FILTER y en el segundo argumento validamos esta columna indicando que sea mayor que cero.

Fx FILTER = FILTER(Ventas, Ventas[Precio Unit]>0)

The table shows 15 rows of data with columns: Nombre Cliente, Cod. Vendedor, Cod.Producto, Cantidad, Precio Unit, CostoUnit, Cod Lineas, Documento. The Precio Unit column has values ranging from 80 to 139. A context menu is open over the Precio Unit column, specifically the dropdown for filtering numbers.

	Nombre Cliente	Cod. Vendedor	Cod.Producto	Cantidad	Precio Unit	CostoUnit	Cod Lineas	Documento
00 a. m.	DRYPERS ANDINA S.A	B-01	1317	80				
00 a. m.	DRYPERS ANDINA S.A	B-01	1317	115				
00 a. m.	DRYPERS ANDINA S.A	B-01	1317	55				
00 a. m.	DRYPERS ANDINA S.A	B-01	1317	124				
00 a. m.	DRYPERS ANDINA S.A	B-01	1317	91				
00 a. m.	DRYPERS ANDINA S.A	B-01	1317	231				
00 a. m.	DRYPERS ANDINA S.A	B-01	1317	236				
00 a. m.	DRYPERS ANDINA S.A	B-01	1317	102				
00 a. m.	DRYPERS ANDINA S.A	B-01	1317	49				
00 a. m.	DRYPERS ANDINA S.A	B-01	1317	72				
00 a. m.	DRYPERS ANDINA S.A	B-01	1317	33				
00 a. m.	DRYPERS ANDINA S.A	B-01	1317	85				
00 a. m.	DRYPERS ANDINA S.A	B-01	1317	50				
00 a. m.	DRYPERS ANDINA S.A	B-01	1317	110				
00 a. m.	DRYPERS ANDINA S.A	B-01	1317	139				

Figura 6.28 Tabla con precios mayores que cero.

Ya el número de registros de dicha tabla se redujo y lo podemos comprobar en el filtro de *Precio Unit*.

El número cero no necesita estar entre comillas, pero si necesitamos filtrar la tabla que solo me lleve el código de producto que sea igual a “1250” debe estar entre comillas.

= FILTER(Ventas, Ventas[Cod.Producto] = "1250")

The table shows 10 rows of data with columns: Cod. Vendedor, Cod.Producto, Cantidad, Precio Unit, CostoUnit, Cod Lineas, Documento. The Cod.Producto column has values all equal to 1250. A context menu is open over the Cod.Producto column, specifically the dropdown for filtering text.

	Cod. Vendedor	Cod.Producto	Cantidad	Precio Unit	CostoUnit	Cod Lineas	Documento
	B-01	1250					
	B-01	1250					
	B-01	1250					
	B-01	1250					
	B-01	1250					
	B-01	1250					
	B-01	1250					
	B-01	1250					
	B-01	1250					
	B-01	1250					

Figura 6.29 Tabla filtrada solo producto “1250”.

Es muy importante antes de crear cualquier tabla filtrada revisar el tipo de dato.

Piensa en este ejercicio, queremos crear un cálculo, donde solo me sume los productos que inician por el número 2, calculemos esta tabla. Piensa por un momento como sería la solución.

Figura 6.30 Productos solo que inicien por el número "2".



Fx `FILTER = FILTER(Ventas,
LEFT(Ventas[Cod.Producto])="2")`

También podemos crear diferentes expresiones filtradas entre dos o más columnas, veremos diferentes ejemplos en el capítulo de operadores lógicos.

FILTER también recibe medidas como filtros de expresión



Advertencia: FILTER solo puede operar en las columnas que estén incluidas en la tabla que se defina o indique en el primer argumento.

Capítulo 7: Funciones y operadores lógicos

Las funciones lógicas no pueden faltar en cualquier lenguaje de programación, incluso son de las primeras fórmulas que aprendemos en Excel o en cualquier otro ambiente sistemático. Estas funciones permiten crear una o múltiples validaciones para devolver un resultado de acuerdo con lo esperado. DAX no es la excepción, incluso en cualquier modelo de datos en Power BI mínimo tenemos una de estas funciones.

Estar comparando o validando resultados es muy común en nuestros análisis, por ejemplo, si el margen esperado no supera el valor asignado en las metas nos devuelva una cadena de texto, un cálculo, un color o cualquier otra alerta, lo mismo sucede si queremos hacer clasificaciones con nuestros datos, en donde dependiendo el valor de la venta o el número de cantidades vendidas esto se clasifica como bajo, medio o alto. Todo este tipo de escenarios se puede realizar gracias a este conjunto de funciones que veremos a continuación.

Microsoft ha venido trabajando muy fuerte en el rendimiento de estas funciones ya que son tan usadas que deben garantizar un buen funcionamiento.

Funciones lógicas

IF

Démosle el primer paso a esta función en español es SI, pero no un sí de respuesta sino un si condicionado. Esta función se compone de 3 argumentos, de los cuales dos son obligatorios y 1 es opcional, comprueba una condición y devuelve un valor verdadero o de lo contrario un valor falso.

1

2

3

```
IF(LogicalTest, ResultIfTrue, [ResultIfFalse])
```

Comprueba si se cumple una condición y devuelve un valor si se evalúa como TRUE y otro valor si se evalúa como FALSE.

```
IF(
```

Figura 7.1 Sintaxis función IF.

Saquemos uso de esta función, en nuestra tabla *Calendario 2*, no contamos con el semestre, ya que no hay una función directa que me calcule esto, pero gracias a IF lo podemos lograr, simplemente creamos una columna nueva en esta tabla llamada Semestre y validamos si el mes en número es menor o igual a 6, lo definimos como semestre 1, de lo contrario hacer parte del semestre 2.

	Año	Mes Num	Trimestre	Mes	Semana	Día Mes	Nombre Día	Columna	Semestre	
00 a. m.	2017	1	1	Enero	1	1	domingo	Enero	Sem 1	Orden ascendente
00 a. m.	2017	1	1	Enero	2	2	lunes	Enero	Sem 1	Orden descendente
00 a. m.	2017	1	1	Enero	2	3	martes	Enero	Sem 1	Borrar orden
00 a. m.	2017	1	1	Enero	2	4	miércoles	Enero	Sem 1	Borrar filtro
00 a. m.	2017	1	1	Enero	2	5	jueves	Enero	Sem 1	Borrar todos los filtros
00 a. m.	2017	1	1	Enero	2	6	viernes	Enero	Sem 1	Filtros de texto
00 a. m.	2017	1	1	Enero	2	7	sábado	Enero	Sem 1	
00 a. m.	2017	1	1	Enero	2	8	domingo	Enero	Sem 1	
00 a. m.	2017	1	1	Enero	3	9	lunes	Enero	Sem 1	
00 a. m.	2017	1	1	Enero	3	10	martes	Enero	Sem 1	
00 a. m.	2017	1	1	Enero	3	11	miércoles	Enero	Sem 1	
00 a. m.	2017	1	1	Enero	3	12	jueves	Enero	Sem 1	

Figura 7.2 Semestre con IF.

Para validar el mes número el 6 no va entre comillas, contrario a “Sem 1” y “Sem 2”, ya que todo texto validado en una función debe ir entre comillas.

El segundo argumento de la función IF como es opcional, si no se agrega el resultado es blanco.

IF anidados

También podemos crear IF anidados para evaluar diferentes resultados, como por ejemplo clasificar los precios de ventas en diferentes categorías, si el precio es cero, se debe clasificar como “Bonificación”, si es menor que 2.000 “Bajo”, menor que 30.000 “Medio” de lo contrario “Alto”

Seleccionamos la tabla *Ventas* → Nueva columna y vamos validando cada resultado como lo muestra la función.

```
1 Clas. Precios =
2 IF (
3     Ventas[Precio Unit] = 0, 1
4     "Bonificación",
5     IF (
6         Ventas[Precio Unit] < 2000,
7         "Bajo",
8         IF ( Ventas[Precio Unit] < 30000, "Medio", "Alto" )
9     )
10 ) 3
```

Figura 7.3 IF anidados.

Se hacen 3 condicionales y cada uno con su debida respuesta, la última clasificación “Alto” no se ingresa en un IF ya que es por defecto, si no se encuentra el resultado en los anteriores condicionales, es “Alto”.

1	Clas. Precios =
2	IF (
3	Ventas[Precio Unit] = 0,
4	"Bonificación",
5	IF (
6	Ventas[Precio Unit] < 2000,
7	"Bajo",
8	IF (Ventas[Precio Unit] < 30000, "Medio", "Alto")
9)
10)

Orden ascendente
Orden descendente
Borrar orden
Borrar filtro
Borrar todos los filtros
Filtros de texto
Buscar
(Seleccionar todo)
Alto
Bajo
Bonificación
Medio

Figura 7.4 Clasificación precios.



Advertencia: El orden en que se definen los IF es importante, si primero se evalúa los precios menores que 30.000 nunca se va a clasificar los precios menores de 2.000 ya que este primer condicional evaluado lo arropa o sobreponde

IF solo cuenta con un argumento lógico, aunque se puede crear varias pruebas lógicas agrupando varios IF, en este capítulo cubriremos otras funciones lógicas y operadores que nos darán la posibilidad de hacer varios testeos lógicos.

IF en medidas

Las funciones IF también son muy usadas en medidas, en nuestro primer ejercicio creamos una columna calculada para clasificar cada fecha en su debido semestre, este tipo de columnas es de toda confianza crearlas, así como clasificar los precios, es la única forma de segmentar la información a no ser que se cree desde Power Query o ya venga desde la fuente de información. IF es muy común verlo en columnas, ya que se permite crear fácilmente por el contexto de fila que se genera.

Un error que cometen los usuarios de DAX es tratar de usar IF en medidas como si estuvieran ejecutándolo en una columna, recuerda que el objetivo de una medida es arrojar un único valor o escalar.

Contamos con una columna en la tabla *Ventas* llamada *% Margen Bruto*, si queremos validar esta columna en un IF donde la regla sería, si el margen es menor a 35% “Bajo”, de lo contrario “Alto”, nos va a arrojar un error, procedamos y analicemos.

Creemos una nueva tabla para estas nuevas medidas, la podemos llamar Medidas Condicionales.

Luego allí vamos a agregar la medida IF Medida y vamos analizando paso a paso.

El primer paso es revisar porque después de iniciar con IF el intellisense o autocompletado no nos trae la columna *% Margen Bruto*, y es lo que explicamos anteriormente en el tipo de medidas, es imposible que DAX evalúe toda una columna dentro de una medida sin una expresión de agregación.

Figura 7.5 Autocompletado deshabilitado para columnas.

Esto es difícil a veces de digerir, pero al terminar este libro estamos completamente seguros de que lo entenderás. Aun si terminamos de evaluar la medida anterior, el resultado sería el siguiente.

Figura 7.6 Error en medida con IF.

El error es claro, no se puede determinar un único valor para esta evaluación.

La recomendación es hacer referencia a otras medidas que ya están bien formulados o dentro del mismo IF hacerlo con funciones de agregación para este caso.

Modifiquemos la condición, y en vez de hacer referencia a la columna, hacemos un llamado a la medida [Margen %], el resultado sería el siguiente.

The screenshot shows a Power BI table with two columns: 'Año' and 'IF Medida'. The 'Año' column has values 2017, 2018, 2019, 2020, and Total. The 'IF Medida' column contains the results of the formula IF([Margen %]<0.35,"Bajo","Alto"). The rows for 2017, 2018, and 2020 have 'Alto' in the 'IF Medida' column, while 2019 and Total have 'Bajo'. A red arrow points from the text above to the 'IF Medida' column header in the table.

Año	IF Medida
2017	Alto
2018	Alto
2019	Bajo
2020	Bajo
Total	Bajo

Figura 7.7 Medida con IF ajustada.

Para la evaluación del margen, el condicional fue <0.35 , DAX no admite como en Excel 35%, debe ser 0.35 o 0,35 dependiendo tu configuración regional de decimales.

SWITCH

Si hacemos la traducción al español, es interrumpir, acá en Colombia nuestros padres tienen una costumbre que cuando quieren que se apague un bombillo o lámpara, su expresión: Apague el "SWITCHE", algo erróneo, pero en si hacían referencia a que se interrumpiera la energía. Hablando ya de DAX con esta función esto es lo que hace, evalúa una lista de valores dentro de una expresión y devuelve posibles resultados, en resumen, son IF anidados, pero mejor estructurados y con diferentes formas de trabajar.

Analicemos la figura 7.4, donde se realizaron varios IF anidados para llegar al resultado, esto lo podemos mejorar con un SWITCH ya que permite en una sola función evaluar la condición y sus posibles resultados.

Veamos su sintaxis.

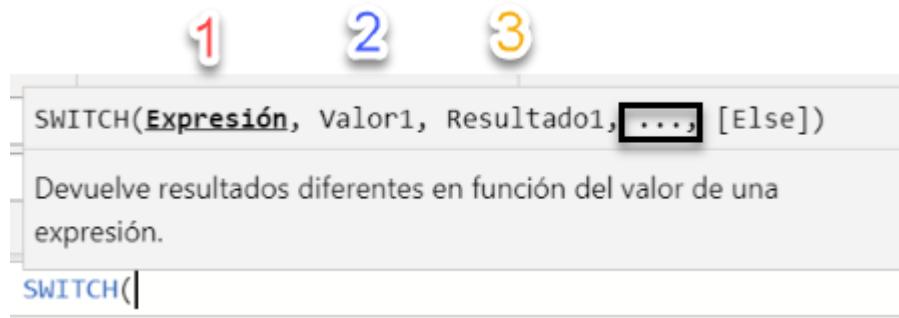


Figura 7.8 Sintaxis función SWITCH.

Tiene 3 argumentos obligatorios y uno opcional, y los argumentos *Valor1* y *Resultado1* son ciclos que se pueden repetir para permitir la evaluación de varios resultados.

Para llegar al resultado de la figura 7.4, reemplazado por SWICHT, en el argumento expresión, debemos indicarle a la función que se interrumpe y evalúe el resultado cuando sea verdadero, para esto DAX trae una función lógica llamada TRUE () .

Insertemos una nueva columna en la tabla *Ventas*.

```
1 FX SWITCH =
2 SWITCH (
3     TRUE (),
4     Ventas[Precio Unit] = 0, "Bonificación",
5     Ventas[Precio Unit] < 2000, "Bajo",
6     Ventas[Precio Unit] < 30000, "Medio",
7     "Alto"
8 )
```

Figura 7.9 Fórmula con SWITCH.

Si analizamos, el primer argumento valida que siempre sea verdadero, y si en uno de los registros cumple esta condición se interrumpe, para y devuelve el resultado indicado, cuando se evalúa cada condición y si ninguno es verdadero el resultado es “Alto”, o se puede dejar en blanco este argumento, ya que es opcional.

Si revisamos la columna resultante debe ser exactamente igual a la creada con IF.

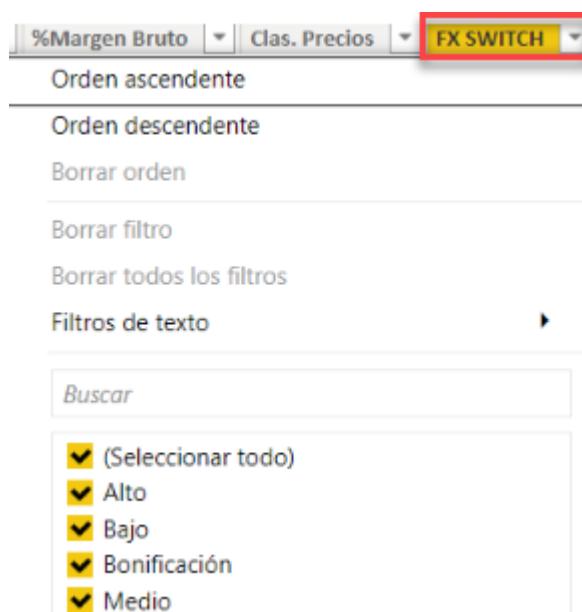


Figura 7.10 Clasificación precios con SWITCH.

Hay otra forma de usar SWITCH y todo depende del contexto donde aplique, en el ejercicio anterior el ciclo se interrumpe cuando la prueba lógica evaluada es verdadera allí se aplica el resultado, y esto gracias a la expresión indicada TRUE (), pero hay otra manera de que opere SWITCH, y en vez de definir que sea TRUE () evaluamos una expresión que puede ser una columna, un ítem, un registro etc.

Nos dirigimos a la tabla *Calendario 2* y vamos a crear una nueva columna para calcular el bimestre, para esto vamos a evaluar el mes número, y dependiendo de su número se va a clasificar.

La expresión para este caso va a ser *Mes Num*, y según el valor evaluado va a ser el resultado.

```
1 Bimestre =
2 SWITCH (
3     'Calendario 2'[Mes Num],
4     1, "Bim 1",
5     2, "Bim 1",
6     3, "Bim 2",
7     4, "Bim 2",
8     5, "Bim 3",
9     6, "Bim 3",
10    7, "Bim 4",
11    8, "Bim 4",
12    9, "Bim 5",
13    10, "Bim 5",
14    11, "Bim 6",
15    12, "Bim 6"
16 )
```

Figura 7.11 Clasificación bimestre.

La columna *Mes Num* contiene 12 registros únicos, esta columna se almacena en SWITCH fila a fila y luego hace el respectivo recorrido validando el valor almacenado y clasifica el bimestre según el número del mes, en esta parte no es necesario crear una prueba lógica para cada mes, ya esto es realizado al inicio, almacenando el *Mes Num* como primer argumento, en caso de que fuera con IF tocaría hacer demasiadas anidaciones para llegar al mismo resultado.



Figura 7.12 Resultado bimestre.



Bimestre =

```
SWITCH (
    'Calendario 2'[Mes Num],
    1, "Bim 1",
    2, "Bim 1",
    3, "Bim 2",
    4, "Bim 2",
    5, "Bim 3",
    6, "Bim 3",
    7, "Bim 4",
    8, "Bim 4",
    9, "Bim 5",
    10, "Bim 5",
    11, "Bim 6",
    12, "Bim 6"
)
```

SWITCH en medidas

Cuando combinamos SWITCH con diferentes medidas podemos solucionar problemas que a simple vista son bien complejos de tratar, como por ejemplo crear informes con tablas y ordenes personalizados, como lo son KPI's y estados financieros. Antes de entrar en materia creamos una matriz y llevamos el campo Línea de la tabla Línea.

Linea
ALIMENTO
ASEO
BEBIDAS
CAFÉ
CONDIMENTOS

Figura 7.13 Matriz por línea.

Queremos recorrer cada registro y validar cada una de las líneas para crear un porcentaje de descuento y saber esto en cuanto afecta la rentabilidad.

La regla es la siguiente:

- ALIMENTO 2%
- ASEO 3%
- BEBIDAS 5%
- CAFÉ 4%
- CONDIMENTO 10%

Este análisis de escenario lo vamos a crear dentro de una medida acompañado de la siguiente función que nos ayudará a realizar el ejercicio.

SELECTEDVALUE

Lo primero que debemos saber es el nombre del registro que se encuentra en la tabla, una medida que me indique el nombre de la línea cuando sea llevada a la matriz.

SELECTEDVALUE es la solución, tiene un argumento obligatorio y es una columna, valida si hay un único argumento, identifica su nombre y lo aplica, si hay varios registros en el contexto actual, trae el argumento opcional que si no se parametriza el resultado es blanco.

Realicemos una medida % DCTO, y evaluamos el campo Línea en SELECTEDVALUE, por último, la arrojamos en la matriz.

% Dcto = SELECTEDVALUE([Linea[Linea]])	
Línea	% Dcto
ALIMENTO	ALIMENTO
ASEO	ASEO
BEBIDAS	BEBIDAS
CAFÉ	CAFÉ
CONDIMENTOS	CONDIMENTOS
Total	

Figura 7.14 SELECTEDVALUE en medida.

SELECTEDVALUE valida el nombre de cada una de las líneas y como resultado es su nombre, parece algo sin sentido y sin mucha lógica si apenas inicias en el mundo de DAX , pero no, es una de las maneras de evaluar los registros de un contexto de filtro con medida, en una columna no hay problema, podemos hacer referencia a la misma columna *Línea [Línea]*, pero en una matriz como en este caso no podemos hacer referencia simplemente a la columna como hemos explicado anteriormente, sino que se debe usar una medida que nos envuelva dicha columna y pueda ser evaluada en el contexto.

Ya con esto claro creo que puedes imaginar el siguiente paso, según lo explicado anteriormente, y es combinar SWITCH con SELECTEDVALUE, SELECTEDVALUE (*Línea [Línea]*) será la expresión para SWITCH y luego se recorrerá registro por registro y se le asignará la regla correspondiente.

Creamos el primer ejemplo para la línea ALIMENTO.

```
% Dcto =  
SWITCH ( SELECTEDVALUE ( Línea[Línea] ), "ALIMENTO", 0.02, "-" )
```

Línea	% Dcto
ALIMENTO	0,02
ASEO	-
BEBIDAS	-
CAFÉ	-
CONDIMENTOS	-

Figura 7.15 Regla para la línea ALIMENTO.

SWITCH guarda el nombre devuelto por SELECTEDVALUE y ejecuta cada condición, en este caso cicla en toda la matriz y donde dice ALIMENTO aplica el 2%, de lo contrario agrega un guion a las demás

Con esta lógica podemos terminar las otras reglas para las demás líneas.

1	% Dcto =
2	SWITCH (
3	SELECTEDVALUE (Linea[Linea]),
4	"ALIMENTO", 0.02,
5	"ASEO", 0.03,
6	"BEBIDAS", 0.05,
7	"CAFÉ", 0.04,
8	"CONDIMENTOS", 0.1
9)



Linea	% Dcto
ALIMENTO	0,02
ASEO	0,03
BEBIDAS	0,05
CAFÉ	0,04
CONDIMENTOS	0,10
Total	

Figura 7.16 Reglas para cada línea.



```
% Dcto =
SWITCH (
    SELECTEDVALUE ( Linea[Linea] ),
    "ALIMENTO", 0.02,
    "ASEO", 0.03,
    "BEBIDAS", 0.05,
    "CAFÉ", 0.04,
    "CONDIMENTOS", 0.1
)
```

Esta es otra metodología adoptada por nuestra empresa para crear informes a la medida, un ejemplo claro de estos son los informes financieros, que llevan una estructura y orden específico, y adicional no se puede relacionar con ninguna otra tabla ya que contiene cálculos a nivel de fila como lo es un PyG o estado de pérdidas y ganancias.

Primero se carga la tabla sin relacionar.

GRUPO	NOMBRE GRUPO	ORDEN
41	VENTAS	1
61	COSTO DE VENTA	2
	UTILIDAD BRUTA	3
51	GASTOS ADMON	4
52	GASTOS VENTAS	5
	TOTAL GASTOS	6
	UTILIDAD OPERACIONAL	7
42	INGRESOS NO OPERACIONALES	8
53	GASTOS NO OPERACIONALES	9
	UTILIDAD ANTES IMPTO	10

Figura 7.17 Tabla personalizada.

Se ordena el nombre grupo según la columna orden y se crea la siguiente función para ir validando fila a fila su respectivo resultado.

El uso de variables dará una mejor legibilidad del código.



Nota: Si quieras profundizar más en crear informes de estados financieros en Power BI puedes adquirir nuestro libro físico o digital desde cualquier lugar del mundo.

<https://bit.ly/3rJ9GC7>

<https://fliptml5.com/ohsap/rmhd>

```

1 Valor
2 VAR F
3 |
4 VAR I
5 |
6 VAR C
7 |
8 VAR L
9 VAR C
10 |
11 VAR C
12 | CALCULATE ( [Valor], FILTER ( ALL ( BD[GRUPO] ), BD[GRUPO] = "52" ) )
13 VAR TotalGastos = GastosAdmon + GastosVenta
14 VAR U_Operativa = Utilidad_Bruta - GastosAdmon - GastosVenta
15 VAR Margen_operativo =
16 | DIVIDE ( U_Operativa, Ingresos )
17 VAR IngresosNoOperacionales =
18 | CALCULATE ( [Valor], FILTER ( ALL ( BD[GRUPO] ), BD[GRUPO] = "42" ) )
19 VAR GastosNoOperacionales =
20 | CALCULATE ( [Valor], FILTER ( ALL ( BD[GRUPO] ), BD[GRUPO] = "53" ) )
21 VAR UtilidadAntesImpsto = U_Operativa + IngresosNoOperacionales - GastosNoOperacionales
22 |
23 RETURN
24 SWITCH (
25   Rubro,
26   "VENTAS", Ingresos,
27   "COSTO DE VENTA", Costos,
28   "UTILIDAD BRUTA", Utilidad_Bruta,
29   "GASTOS ADMON", GastosAdmon,
30   "GASTOS VENTAS", GastosVenta,
31   "TOTAL GASTOS", TotalGastos,
32   "UTILIDAD OPERACIONAL", U_Operativa,
33   "% UTILIDAD OPERACIONAL", Margen_operativo,
34   "INGRESOS NO OPERACIONALES", IngresosNoOperacionales,
35   "GASTOS NO OPERACIONALES", GastosNoOperacionales,
36   "UTILIDAD ANTES IMPTO", UtilidadAntesImpsto,
37   |
38   BLANK ()
39 )
  
```

= UTILIDAD BRUTA - GASTOS ADMON - GASTOS VENTAS - TOTAL GASTOS	225,3M 99,6M 165,5M 265,1M
= UTILIDAD OPERACIONAL INGRESOS NO OPERACIONALES - GASTOS NO OPERACIONALES	-39,8M 16,8M 52,5M
= UTILIDAD ANTES IMPTO	-75,5M

Figura 7.18 Ejemplo con SWITCH.

Esto es un ejemplo práctico de todo lo que puedes crear gracias a estas funciones.

AND

Evalúa dos pruebas lógicas y si ambas se cumplen devuelve VERDADERO, de lo contrario FALSO, a diferencia de la función Y en Excel que se pueden comprobar hasta 255 condiciones, en DAX solo permite dos.

Esta función por si sola no es muy común analizarla en DAX, más que un resultado con la palabra VERDADERO o FALSO necesitamos traer un resultado en caso de que las condiciones se cumplan o no.

Para explicar esta función y su uso, creamos una columna calculada en la tabla Ventas, para calcular el número de días en la entrega de cada pedido.

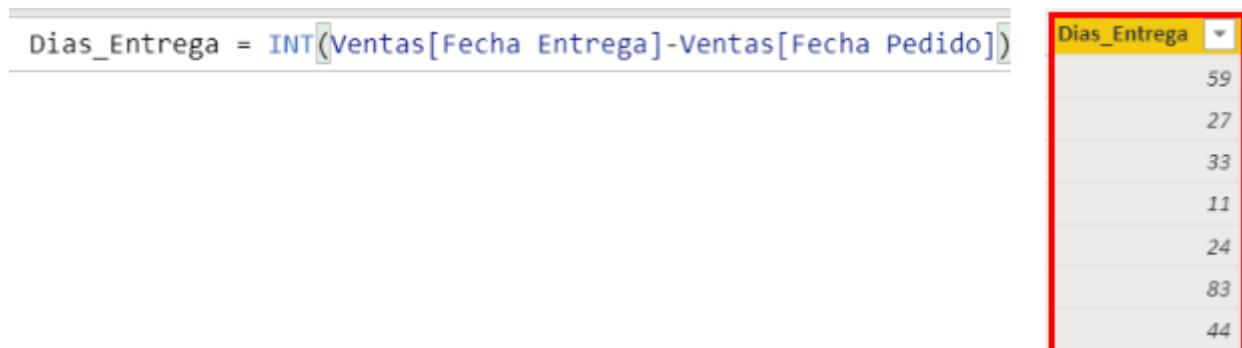


Figura 7.19 Días entrega.

Con la función AND podemos validar fila a fila que pedido fue entregado entre el rango de los cero a los 30 días.

Insertamos una nueva columna y hacemos estas dos validaciones.

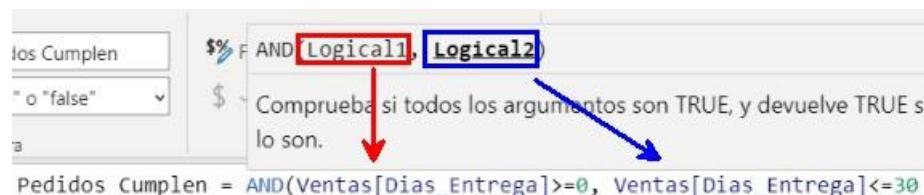


Figura 7.20 Condición pedidos cumplen.

Podemos revisar el resultado y es una columna con True y False (Verdaderos y falsos).



Figura 7.21 True y false resultados.

Y con esta columna podemos crear medidas que nos cuente el número de pedidos que cumplan con la condición. En DAX los resultados VERDADEROS los podemos convertir en número 1 y los FALSOS en 0, solo anteponiendo la función INT.

The screenshot shows the Power BI Data Editor interface. At the top, there is a formula bar with the DAX code: `INT (AND(Ventas[Dias_Entrega]>=0, Ventas[Dias_Entrega]<=30))`. Below the formula bar, there are two columns of context filters:

- Left Column (Initial State):**
 - FX SWITCH ▾ Dias_Entrega ▾ Pedidos Cumplen ▾ (highlighted with a red box)
 - Orden ascendente
 - Orden descendente
 - Borrar orden
 - Borrar filtro
 - Borrar todos los filtros
 - (Seleccionar todo)
 - False
 - True
- Right Column (Final State):**
 - FX SWITCH ▾ Dias_Entrega ▾ Pedidos Cumplen ▾ (highlighted with a red box)
 - Orden ascendente
 - Orden descendente
 - Borrar orden
 - Borrar filtro
 - Borrar todos los filtros
 - Filtros de número ▾ (highlighted with a red arrow pointing to the right)
 - (Seleccionar todo)
 - 0
 - 1

Figura 7.22 Convertir verdadero en 1 y falso en 0.



Pedidos Cumplen =

```
INT ( AND ( Ventas[Dias_Entrega] >= 0, Ventas[Dias_Entrega] <= 30 ) )
```

También AND es útil en funciones de tabla, en la figura 6.28 del capítulo de funciones de tabla, segmentamos la tabla ventas solo los precios mayores que cero, con AND podemos agregarle más filtros, que sean mayores que cero los precios y que el código del municipio sea igual a “05001”.

Insertamos una nueva tabla y realizamos esta segmentación:

The screenshot shows a data filtering interface with two dropdown menus. The left menu, labeled 'Precio Unit', has options: Orden ascendente, Orden descendente, Borrar orden, Borrar filtro, Borrar todos los filtros, Filtros de número, (Seleccionar todo), 35, 67, 68, 69, 84, 126, and 127. The right menu, labeled 'Cod Municipio', has options: Orden ascendente, Orden descendente, Borrar orden, Borrar filtro, Borrar todos los filtros, Filtros de texto, Buscar, (Seleccionar todo), and 05001. Red arrows point from the code in the figure to these menus.

```

1 Fx AND_FILTER =
2 FILTER (
3   Ventas,
4   AND ( Ventas[Precio Unit] > 0, Ventas[Cod Municipio] = "05001" )
5 )

```

Figura 7.23 Tabla filtrada por precio y municipio.



```

Fx AND_FILTER =
FILTER (
  Ventas,
  AND ( Ventas[Precio Unit] > 0, Ventas[Cod Municipio] = "05001" )
)

```

Si deseamos agregar más condiciones, como por ejemplo que el código del producto sea igual a “1250”, debemos añadir un AND dentro de otro AND y allí parece ser complejo la cosa.

```

1 Fx AND_FILTER =
2 FILTER (
3   Ventas,
4   AND (
5     AND ( Ventas[Precio Unit] > 0, Ventas[Cod Municipio] = "05001" ),
6     Ventas[Cod.Producto] = "1250"
7   )
8 )

```

Figura 7.24 AND anidados.

Parece enredado o complejo el código si aun estas iniciando en este lenguaje, formatea la imagen anterior porque que sucede si tambien queremos filtrar la tabla por un vendedor o cliente en especial, seria anidar AND sobre AND, te tenemos una buena noticia, para esto existen los operadores y va a ser mas facil de lo que crees.

OR

Evalúa dos pruebas lógicas y si alguna de las dos cumple una condición su resultado es verdadero, de lo contrario falso.

Esta función se usa para evaluar varios criterios de la misma columna, por ejemplo, crear una tabla a partir de la tabla ventas que solo tenga los productos "1250" y "2204", aunque el enunciado puede confundir porque dice "y", pues si iteramos sobre esta columna validando con AND el resultado, en una misma celda o fila jamás se va a encontrar este "1250 2204".

Muy importante leer cada problema planteado cuando usamos OR o AND.

Insertamos una nueva tabla y con ayuda de FILTER definimos:

The screenshot shows a Power BI interface with a table of sales data. The columns are: Hora, Nombre Cliente, Cod. Vendedor, Cod.Producto, Cantidad, Precio Unit, CostoUnit, and Cod Lin. The 'Cod.Producto' column is highlighted in yellow. A context menu is open over this column, with the 'Filtros de texto' option selected. A red box highlights the 'Filtros de texto' section of the menu, which contains three checked filter items: '(Seleccionar todo)', '1250', and '2204'.

```

1 Fx OR_FILTER =
2 FILTER (
3   Ventas,
4   OR ( Ventas[Cod.Producto] = "1250", Ventas[Cod.Producto] = "2204" )
5 )

```

ra	Nombre Cliente	Cod. Vendedor	Cod.Producto	Cantidad	Precio Unit	CostoUnit	Cod Lin
10:00 a. m.	CIPA S.A	B-01	1250				
10:00 a. m.	CIPA S.A	B-01	1250				
10:00 a. m.	CIPA S.A	B-01	1250				
10:00 a. m.	CIPA S.A	B-01	1250				
10:00 a. m.	CIPA S.A	B-01	1250				
10:00 a. m.	CIPA S.A	B-01	1250				
10:00 a. m.	CIPA S.A	B-01	1250				
10:00 a. m.	CIPA S.A	B-01	1250				
10:00 a. m.	CIPA S.A	B-01	1250				
10:00 a. m.	CIPA S.A	B-01	1250				
10:00 a. m.	CIPA S.A	B-01	1250				
10:00 a. m.	CIPA S.A	B-01	1250				
10:00 a. m.	CIPA S.A	B-01	1250				

Figura 7.25 Tabla ventas con productos 1250 y 2204.



```

Fx OR_FILTER =
FILTER (
  Ventas,
  OR ( Ventas[Cod.Producto] = "1250", Ventas[Cod.Producto] = "2204" )
)

```

Podemos también combinar AND con OR, pero lo veremos con operadores y así facilitarte tu código.

Operadores lógicos

En el capítulo 1, dimos una breve introducción a los operadores, que se dividen en lógicos, comparativos, concatenación de texto, aritméticos y de precedencia. Los operadores en DAX son un tipo de atajos que podemos usar para evitar grandes líneas de códigos y mejorar el entendimiento de nuestro código.

&& (AND)

El doble ampersand es el operador que puede reemplazar la prueba lógica AND, y es muy útil ya que permite anidar varias condiciones sin necesidad de usar funciones adicionales.

Como podemos reemplazar esta imagen 7.24 con este tipo de operadores

```
1 Fx AND_FILTER =
2 FILTER (
3     Ventas,
4     AND (
5         AND ( Ventas[Precio Unit] > 0, Ventas[Cod Municipio] = "05001" ),
6         Ventas[Cod.Producto] = "1250"
7     )
8 )
```

Insertemos una nueva tabla y con doble ampersand podemos eliminar los dos AND de la función sin paréntesis ni separadores de coma.

```
1 Fx && =
2 FILTER (
3     Ventas,
4     Ventas[Precio Unit] > 0
5     && Ventas[Cod Municipio] = "05001"
6     && Ventas[Cod.Producto] = "1250"
7 )
```

Figura 7.26 Reemplazar AND por &&.

El resultado es el mismo, pero con un código más fácil de digerir, adicional podemos agregarle más condiciones y de manera muy intuitiva, como terminar de segmentar que solo muestre los pedidos menos a 30 días.

The screenshot shows the Power BI Query Editor. At the top, there is a code block:

```

1 Fx && =
2 FILTER (
3   Ventas,
4     Ventas[Precio Unit] > 0
5     && Ventas[Cod Municipio] = "05001"
6     && Ventas[Cod.Producto] = "1250"
7     && Ventas[Dias_Entrega] < 30
8 )

```

Below the code is a table with columns: Fecha Factura, Nombre Cliente, Cod. Vendedor, and another partially visible column. The table contains 42 rows, each representing a sales record. A red arrow points from the bottom of the table back up to the code block. A red box highlights the status bar at the bottom of the editor which reads: "Fx && (42 filas Columna: Fecha Factura (39 valores distintos))".

Figura 7.27 Tabla filtrada por múltiples condiciones.



```

Fx && =
FILTER (
  Ventas,
  Ventas[Precio Unit] > 0
    && Ventas[Cod Municipio] = "05001"
    && Ventas[Cod.Producto] = "1250"
    && Ventas[Dias_Entrega] < 30
)

```

Ya la tabla solo cuenta con 42 filas, y esto puede ser el resultado para una medida.

|| (OR)

El doble símbolo de tubo reemplaza el condicional OR, con la misma estructura del ejercicio anterior podemos lograr esta sustitución. Este símbolo se encuentra en el teclado al lado izquierdo del número 1 parte superior izquierda.

`FILTER (`

```

Ventas,
OR ( Ventas[Cod.Producto] = "1250", Ventas[Cod.Producto] = "2204" )
)

```

Si tomamos la imagen 7.25, podemos sustituirla con el doble símbolo de tubo, simplemente quitando el OR y sus paréntesis y en vez de la coma va ||

La función se vería así:

```

1 Fx || =
2 FILTER (
3   Ventas,
4   Ventas[Cod.Producto] = "1250"
5   || Ventas[Cod.Producto] = "2204"
6 )

```

Figura 7.28 Reemplazar OR por ||.

Si validas esta nueva tabla creada, el resultado a nivel de filas y columnas es exactamente igual.

Gracias a este operador podemos agregar otro código de producto sin ningún problema, por ejemplo, el “3607”. La función sería así:

	Nombre Cliente	Cod. Vendedor	Cod.Producto	Cantidad	Precio Unit
00:00 a. m.	PFIZER S.A.S.	B-01	2204		
00:00 a. m.	PFIZER S.A.S.	B-01	2204		
00:00 a. m.	PFIZER S.A.S.	B-01	3607		
00:00 a. m.	PFIZER S.A.S.	B-01	2204		
00:00 a. m.	PFIZER S.A.S.	B-01	1250		
00:00 a. m.	HAVELLS SYLVANIA COLOMBIA S.A	B-01	3607		
00:00 a. m.	EMPAQUETADORA Y COMERCIALIZADORA JDY S.A	B-01	3607		
00:00 a. m.	EMPAQUETADORA Y COMERCIALIZADORA JDY S.A	B-01	3607		
00:00 a. m.	EMPAQUETADORA Y COMERCIALIZADORA JDY S.A	B-01	3607		
00:00 a. m.	HAVELLS SYLVANIA COLOMBIA S.A	B-01	3607		
00:00 a. m.	PFIZER S.A.S.	B-01	2204		
00:00 a. m.	VALENCIA VICTORIA GUILLERMO	B-01	2204		
00:00 a. m.	INDUSTRIA DE ASEO LIMPIAHOGAR SAS	B-01	3607		
00:00 a. m.	INDUSTRIA DE ASEO LIMPIAHOGAR SAS	B-01	3607		

Orden ascendente
Orden descendente
Borrar orden
Borrar filtro
Borrar todos los filtros
Filtros de texto

 (Seleccionar todo)
 1250
 2204
 3607

Figura 7.29 Filtro con 3 productos.



```
Fx || =  
FILTER (  
    Ventas,  
    Ventas[Cod.Producto] = "1250"  
    || Ventas[Cod.Producto] = "2204"  
    || Ventas[Cod.Producto] = "3607"  
)
```

Combinación ||, && y manejo de paréntesis

El buen uso de paréntesis permitirá el buen resultado cuando queremos combinar los operadores || y &&. En el ejemplo anterior la tabla esta filtrada por 3 productos, si adicional queremos añadir a esta condición, que sean solo los precios mayores que cero.

Encerramos entre paréntesis la primera condición de los productos y como se va a condicionar con otra columna añadimos &&, veamos.

The screenshot shows the Power BI Data Editor interface. At the top, there is a code editor window with the following DAX query:

```
1 Fx || y && =  
2 FILTER (  
3     Ventas,  
4     ( Ventas[Cod.Producto] = "1250"  
5         || Ventas[Cod.Producto] = "2204"  
6         || Ventas[Cod.Producto] = "3607" )  
7     && Ventas[Precio Unit] > 0  
8 )
```

Two red arrows point to specific parts of the query: one points to the opening parenthesis '(', and another points to the condition 'Ventas[Precio Unit] > 0'. Below the code editor are two tables:

Cod.Producto	Cantidad	Precio Unit
1250	Orden ascendente	500
3607	Orden descendente	500
3607	Borrar orden	500
3607	Borrar filtro	500
3607	Borrar todos los filtros	500
3607	Filtros de texto	500
3607	Buscar	500
3607	(Seleccionar todo)	500
3607	1250	500
3607	2204	500
3607	3607	500

Precio Unit	CostoUnit	Cod.Linesas
500	Orden ascendente	500
500	Orden descendente	500
500	Borrar orden	500
500	Borrar filtro	500
500	Borrar todos los filtros	500
500	Filtros de número	500
500	(Seleccionar todo)	500
500	257	500
500	298	500
500	332	500
500	337	500
500	503	500

Figura 7.30 Combinación &&, || y paréntesis.

Y con esta lógica podemos añadir varios criterios hasta generar análisis potentes.



Nota: Con la misma sintaxis podemos aplicar estos operadores en medidas, recordando el buen uso de ellas para ver resultados exitosos.



```
Fx || y && =  
FILTER (  
    Ventas,  
    ( Ventas[Cod.Producto] = "1250"  
        || Ventas[Cod.Producto] = "2204"  
        || Ventas[Cod.Producto] = "3607" )  
        && Ventas[Precio Unit] > 0  
)
```

IN

Este operador es una concatenación de varios OR o ||, de gran uso para no repetir varios OR cuando se requiera evaluar varias condiciones de un único campo. Por ejemplo, la figura 7.29 tiene filtrados los productos “1250”, “2204” y “3607”, esta misma función puede ser mejorada en su legibilidad del código con IN. Creamos una nueva tabla, y luego de nombrar el campo código del producto anexamos el operador IN y entre llaves separados por coma los productos a filtrar.

```
1 Fx IN =  
2 FILTER ( Ventas, Ventas[Cod.Producto] IN { "1250", "2204", "3607" } )
```

Figura 7.31 Función con operador IN.

Esta función devuelve exactamente el mismo resultado que con la función evaluada con || y con la ventaja de seguir añadiendo más códigos de productos, adicionando una coma y el código entre comillas.



```
Fx IN =  
FILTER ( Ventas, Ventas[Cod.Producto] IN { "1250", "2204", "3607" } )
```

```
1 Fx || =  
2 FILTER (  
3     Ventas,  
4     Ventas[Cod.Producto] = "1250"  
5     || Ventas[Cod.Producto] = "2204" → 1 Fx IN =  
6     || Ventas[Cod.Producto] = "3607" → 2 FILTER ( Ventas, Ventas[Cod.Producto] IN { "1250", "2204", "3607" } )  
7 )
```

Figura 7.32 Comparativo de funciones con || e IN.

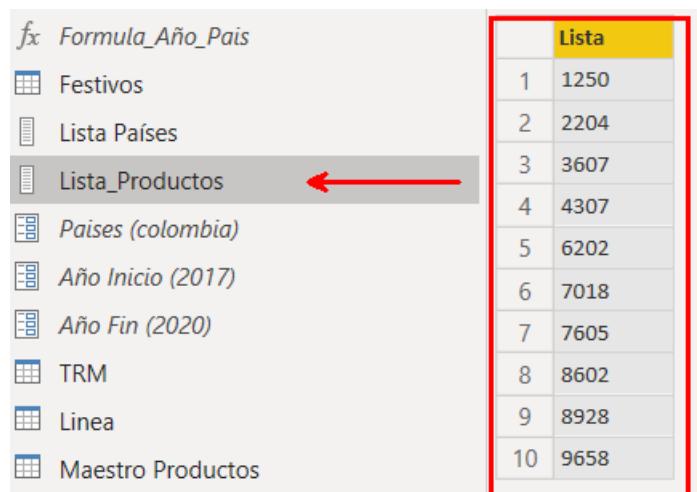
Cada vez más mejoramos nuestros códigos DAX

IN con listas de Power Query

En este caminar con DAX te vas a encontrar con cientos de escenarios, que incluso no estén como ejemplo en este libro, es lo normal cuando aplicamos cualquier lenguaje de programación a nuestra realidad, pero tratamos de cubrir los conceptos y ejercicios más relevantes que te sirvan como base para enfrentar tus retos diarios, uno de ellos lo queremos plantear, que sucede si se desea realizar una tabla o medida que nos execute una serie de productos provenientes de una lista, es decir, que el cálculo quede dinámico y no quemado como ejercicios anterior, pues te contamos que esto se puede realizar y de la manera más sencilla posible.

La lista puede ser de una fuente de SQL, Excel, OneDrive, txt etc.

Para nuestro ejemplo la cargamos como tabla temporal



The screenshot shows the Power BI Data View interface. On the left, there is a list of tables and queries under the heading 'Formula_Año_Pais'. The table 'Lista_Productos' is highlighted with a red arrow pointing to it from the left. To the right of the list is a table preview titled 'Lista' with 10 rows of data:

	Lista
1	1250
2	2204
3	3607
4	4307
5	6202
6	7018
7	7605
8	8602
9	8928
10	9658

Figura 7.33 Lista de código de productos en Power Query.

Con esta lista de códigos de productos cargadas en Power BI, lo único que debemos hacer para crear una tabla o medida que solo contenga estos códigos, es con el operador IN escribir el nombre de esta lista en comillas simples 'Lista_Productos'.

Insertamos una nueva tabla y la función sería:

ira	Nombre Cliente	Cod. Vendedor	Cod.Producto	Cantidad	Precio Unit
00:00 a. m.	CONSERMAR S.A.S	B-02	8928		
00:00 a. m.	CONSERMAR S.A.S	B-02	8928		
00:00 a. m.	URREGO LEZCANO DANIEL	B-02	8928		
00:00 a. m.	URREGO LEZCANO DANIEL	B-02	8928		
00:00 a. m.	URREGO LEZCANO DANIEL	B-02	8928		
00:00 a. m.	URREGO LEZCANO DANIEL	B-02	8928		
00:00 a. m.	DIST. DE GRANOS Y CEREALES LA CASCADA S.	B-02	8928		
00:00 a. m.	CONSERMAR S.A.S	B-02	8928		
00:00 a. m.	VALENCIA VICTORIA GUILLERMO	B-02	8928		
00:00 a. m.	COMERCIALIZADORA QFC S A S	B-02	8928		
00:00 a. m.	INVERSIONES POMAR ROA & CIA S.A.S	B-02	8928		
00:00 a. m.	PROCESADOS LHM S.A.S	B-02	8928		
00:00 a. m.	DIST. DE GRANOS Y CEREALES LA CASCADA S.	B-02	8928		
00:00 a. m.	CONSERMAR S.A.S	B-02	8928		
00:00 a. m.	COMERCIALIZADORA QFC S A S	B-02	8928		
00:00 a. m.	PAPELES PRIMAVERA S.A.	B-02	8928		
00:00 a. m.	INVERSIONES POMAR ROA & CIA S.A.S	B-02	8928		
00:00 a. m.	CONSERMAR S.A.S	B-02	8928		
00:00 a. m.	URREGO LEZCANO DANIEL	B-02	8928		

Figura 7.34 Función IN con lista.

El resultado de la tabla es los 10 productos que solo se encuentran en la lista.

 Fx IN Listas =
FILTER (Ventas, Ventas[Cod.Producto] IN 'Lista_Productos')

No te parece asombroso todo lo que podemos crear con DAX, y no bajes la toalla, esto apenas inicia a los siguientes capítulos que tenemos preparados para ti.

IN y NOT

IN puede combinar perfectamente con la función lógica NOT, no se explicó en este capítulo ya que se debe aclarar en un contexto para su mejor entendimiento, por ejemplo, si deseamos traer los días hábiles (lunes a viernes) de la tabla *Calendario* podemos hacerlo con IN en caso de que sea con los nombres de los días. La respuesta sería:

```

1 Días Hábiles =
2 FILTER (
3   CALENDARIO,
4   Calendario[Nombre Dia]
5   IN { "lunes", "martes", "miércoles", "jueves", "viernes" }
6 )

```

Figura 7.35 Tabla días hábiles con nombre de los días.

Algo que podemos modificar es diciéndole a DAX que no traiga los días sábado y domingo, en vez de escribir todos los días de la semana, esto gracias a la combinación con NOT

Anteponemos la función lógica NOT antes de la columna Nombre Día y validamos.

Ructura	Formato	Propiedades	Ordenar
1 Días Hábiles = 2 FILTER (CALENDARIO, NOT Calendario[Nombre Dia] IN { "sábado", "domingo" })	↓		

Figura 7.36 Tabla días hábiles con nombre de los días con NOT.

Las dos funciones traen el mismo resultado, pero aprendemos a siempre escribir menos códigos y otras posibilidades.

1 Días Hábiles = 2 FILTER (CALENDARIO, 3 Calendario[Nombre Dia] 4 IN { "lunes", "martes", "miércoles", "jueves", "viernes" } 5 6)	→	1 Días Hábiles = 2 FILTER (CALENDARIO, NOT Calendario[Nombre Dia] IN { "sábado", "domingo" })
--	---	--

Figura 7.37 Comparativo funciones con y sin NOT.



Días Hábiles =
FILTER (CALENDARIO, NOT Calendario[Nombre Dia]
IN { "sábado", "domingo" })

Capítulo 8: RELATED, RELATEDTABLE y LOOKUPVALUE

En el capítulo 2 hicimos mucho énfasis en la importancia de tener un buen modelo relacional y la dirección del filtro cruzado, que el mejor modelo que podemos crear es un modelo de uno a muchos, que las tablas maestras o de dimensión siempre viajan hacia las tablas transaccionales o de hechos con una única dirección. Si sigues estas buenas prácticas te harás la vida más fácil en DAX y no tendrás confusión en los resultados arrojados.

En este capítulo vamos a ver materializado la importancia de las relaciones y que funciones podemos usar para llevar campos de una tabla a otra, medidas que permitan iterar sobre una tabla creando operaciones junto con otra, todo este tipo de ejecuciones vamos a poder comprenderlos gracias a las funciones RELATED y RELATEDTABLE.

Aunque en el título ves la función LOOKUPVALUE, no hace parte de esta categoría de funciones de relación; primero queremos enseñarte técnicas cuando no podemos crear relaciones entre las tablas, es decir, como trabajar con tablas desconectadas y usar esos campos, pero también crear técnicas de relación usando llaves o campos únicos para crear menos código y mayor rendimiento.

RELATED

Trabajar con campos de otras tablas es lo que hace poderoso los modelos tabulares, poder crear análisis con diferentes variables es lo que nos da verdadero valor para la toma de decisiones. RELATED es una de estas funciones que permite traer campos o columnas de una tabla a otra, pero solo y únicamente campos de las tablas de los uno a los muchos, es decir, solo podemos contar con los campos que tenemos en las tablas maestras, por ejemplo, campos de la tabla Vendedores y llevarlos a la tabla transaccional, en este caso Ventas.

Si volvemos a revisar nuestro modelo, todas las tablas apuntan o viajan hacia la tabla *Ventas*, es allí donde RELATED puede desplazarse y usar todas las columnas que contiene cada una de estas tablas y gracias a la relación física que existe se pueden ver en la tabla de hechos (*Ventas*)

Solo hay una tabla en el modelo que no se encuentra relacionada **TRM**, daremos un tratado especial.

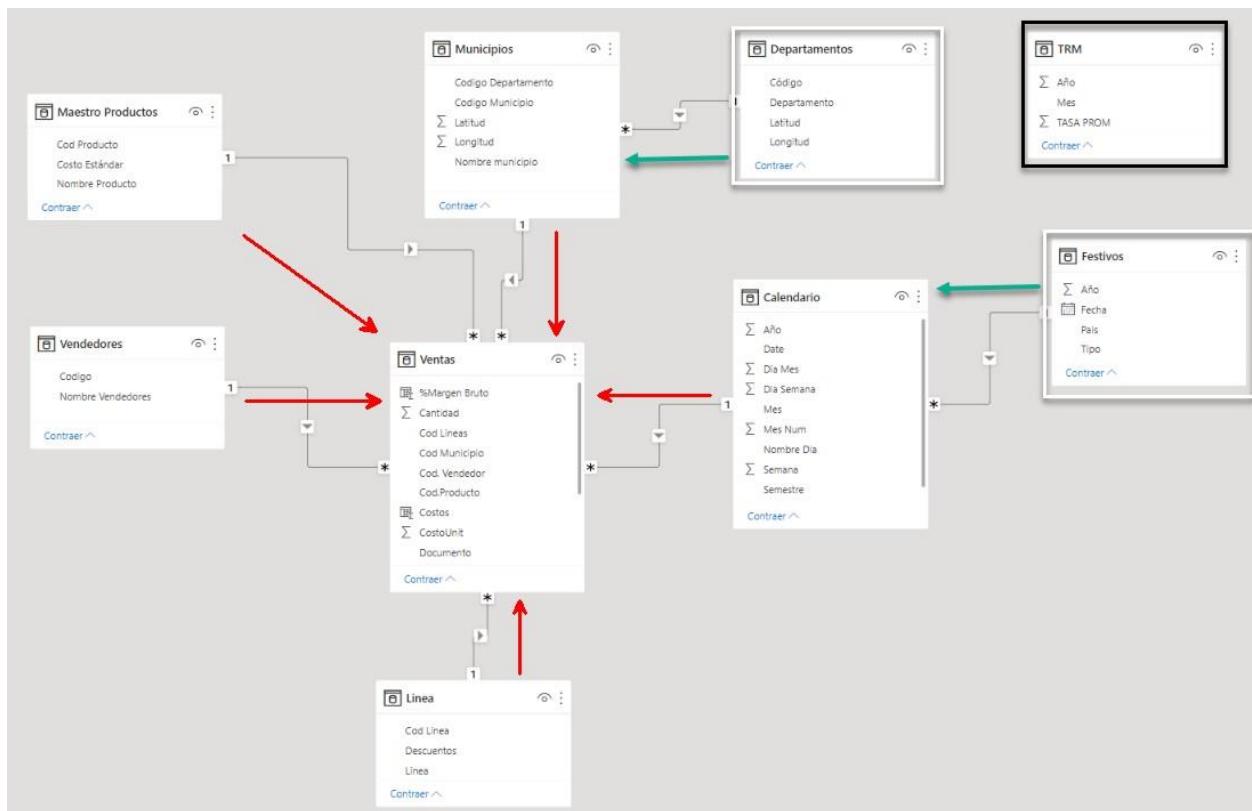


Figura 8.1 Modelo relacional.

Veamos un ejemplo como se usa RELATED.

En la tabla *Ventas* solo contamos con el código del vendedor, pero el nombre se encuentra en la tabla *Vendedores*, insertamos una nueva columna en la tabla ventas y usamos RELATED.

RELATED solo cuenta con un argumento en su sintaxis y es una columna.

Al referenciar RELATED nos trae todas las columnas en el autocompletado que podamos usar.

The screenshot shows the Power BI ribbon with the 'Data Model' tab selected. In the center, there's a table with columns 'actura' and 'Nombre Cliente'. Below the table, the formula bar shows '1 Vendedor = RELATED('. A red box highlights the dropdown menu that appears when clicking on 'RELATED(', listing various options like 'Maestro Productos', 'Calendario', etc., with 'Calendario' being the selected item.

Figura 8.2 Autocompletado en RELATED.

Seleccionamos la columna que deseamos llevar o completar en nuestra tabla ventas, para este caso sería *Nombre Vendedores*.

The screenshot shows the Power BI ribbon with the 'Vendedores' column selected. The formula bar displays 'Vendedor = RELATED(Vendedores[Nombre Vendedores])'. A red box highlights the formula bar. Below it, the column settings dropdown is open, showing options like 'Orden ascendente', 'Orden descendente', 'Borrar orden', 'Borrar filtro', 'Borrar todos los filtros', and 'Filtros de texto'. At the bottom, a list of names is shown with checkboxes, all of which are checked: '(Seleccionar todo)', 'ALBERT VELASQUEZ', 'JUAN GUILLERMO RICAURTE', 'MIGUEL ANGEL RIOS', 'SIMON TERRANOVA', and 'VANESSA QUINTERO'.

Figura 8.3 Columna nombre vendedores agregada.

Advertencia: Para completar columnas en tabla *Ventas* es obligatorio que exista una relación física.



Podemos comprobar el anterior texto, si deseamos llevar la tasa de cambio o TRM (Tasa representativa del mercado) a la tabla *Ventas*, no es posible, ya que no existe relación.

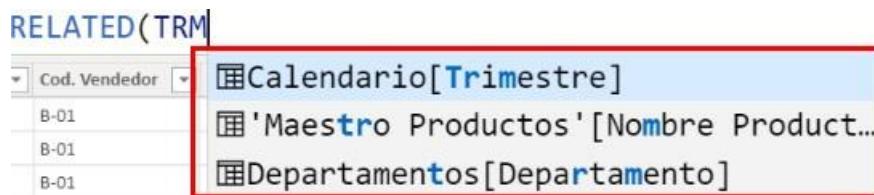


Figura 8.4 Relación inexistente entre la tabla *Ventas* y TRM.

El intellisense no encuentra ningún campo de la tabla *TRM*.



Advertencia: RELATED solo es aplicado en la tabla de hechos o tablas transaccionales.

Podemos comprobar que RELATED solo es aplicado en la tabla *Ventas* para este caso.

Si nos ubicamos en la tabla *Vendedores* e insertamos una nueva columna, RELATED no puede identificar ningún campo disponible.

X ✓ 1 Columna = RELATED()		
Nombre Vendedores	Codigo	Columna
JUAN GUILLERMO RICAURTE	B-01	
ALBERT VELASQUEZ	B-02	
VANESSA QUINTERO	B-03	
MIGUEL ANGEL RIOS	B-04	
SIMON TERRANOVA	B-05	

Figura 8.5 RELATED no se puede usar en la tabla vendedores.

Esta solución y explicación la veremos con RELATEDTABLE.

De esta forma podemos completar en la tabla *Ventas* con todas las columnas de las posibles relaciones existentes, pero esto no tiene sentido realizarlo hasta este momento, ya que simplemente podemos usar campos en cualquier informe de ventas y de cualquier otra tabla relacionada sin crear columnas calculadas adicionales.

Gracias a los modelos tabulares RELATED tiene la capacidad de usar campos de tablas que directamente no tienen una relación con ella, pero que sí hay una tabla relacionada en el medio (copo de nieve) como lo son los municipios y departamentos.

Si deseamos traer el nombre del departamento a la tabla *Ventas* parece a simple vista que no es posible, ya que si revisamos nuestro modelo la tabla *Departamentos* está relacionada con la tabla *Municipios* por medio del campo *Código Departamento* y la tabla *Municipios* está relacionada con la tabla *Ventas* enlazada con el *Código Municipio*. Ver figura 8.6

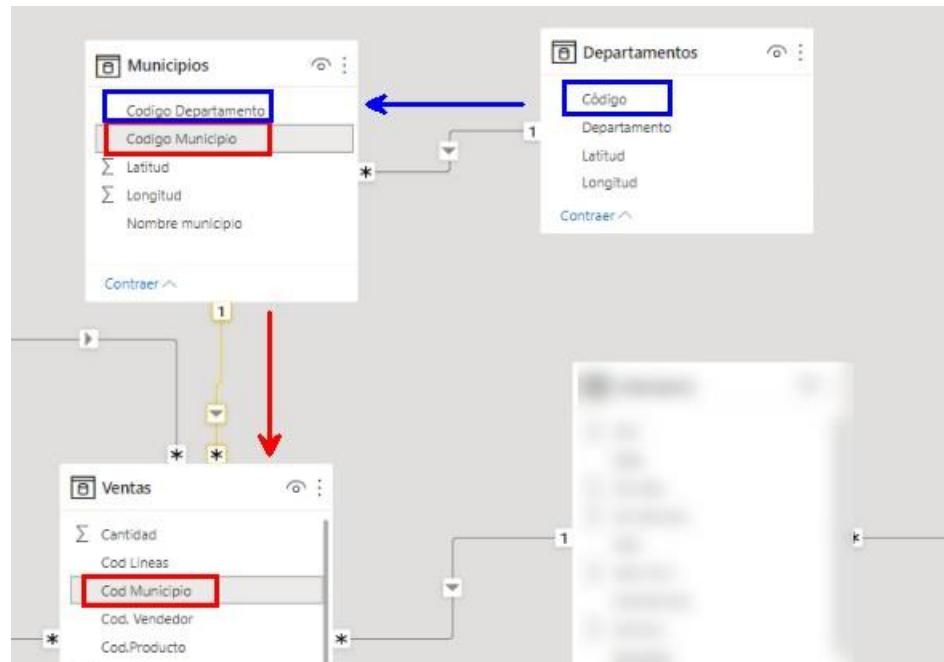


Figura 8.6 Relación tabla departamentos, municipios y ventas.

Gracias a los modelos tabulares, RELATED puede taladrar hasta la tabla *Departamentos* y hacer uso de ellos en la tabla *Ventas*.

Insertemos una nueva columna en la tabla *Ventas* y acompañada de RELATED traemos el nombre del departamento.

1 Departamento = RELATED(Departamentos)				
Nombre Cliente	Cod. Vendedor	Cod. Producto	Cantidad	
I8 QUALA S.A	B-01	2303	20	
I8 QUALA S.A	B-01	2303	48	
I8 QUALA S.A	B-01	2303	140	
I8 QUALA S.A	B-01	2303	400	

Departamentos [Código]
Departamentos [Departamento]
Departamentos [Latitud]
Departamentos [Longitud]

Figura 8.7 Campos tabla departamentos.

Si iniciamos escribiendo la palabra *Departamentos*, el autocomplete nos trae todos los campos que están allí y poder agregarlos en la tabla *Ventas*, seleccionamos la columna *Departamento* y Enter.

RELATED en medidas

Desde que iniciamos el mundo de DAX reiteramos que al final todo se debe resumir en medidas, principalmente por su rendimiento, pero que creamos columnas y tablas para poder ver todo el proceso

y resultados que estas funciones nos arrojan ya que las medidas solo son evidentes cuando se llevan a una consulta o un objeto visual.

No solo RELATED es útil para completar tablas con otras columnas, sino que gracias a su combinación con funciones de iteración podemos crear medidas que nos ayuden a generar valor a nuestros reportes. Por ejemplo, en la tabla *Maestro Productos* contamos con una columna que contiene el *Costos Estándar* de cada producto y en la tabla *Ventas*, está registrado el costo real, si quisieramos realizar un comparativo año a año o producto por producto cual fue la variación con respecto a los costos que se estimaron y se ejecutaron, esta columna *Costos Estándar* se debe llevar a la tabla *Ventas* y luego crear el comparativo adecuado.

The screenshot shows a Power BI interface. On the left is a table with columns: 'Cod Producto', 'Nombre Producto', and 'Costo Estándar'. The table contains 17 rows of product information. To the right is a navigation pane titled 'Buscar' (Search) with a list of items. The item 'Maestro Productos' is highlighted with a red box. Other items listed include 'Medidas Iteración', 'Medidas Variables', 'Calendario', 'Calendario 2', 'Departamentos', 'Festivos', 'Línea', 'Lista Paises', 'Municipios', and 'Ventas'.

Cod Producto	Nombre Producto	Costo Estándar
8928	CORN FLAKES CEREAL INTEGRAL x 800 gr	20110
1738	NESTUM 5 CEREALES x 200 gr	19096
2303	QUIPITOS SOBRE x 24 x 8gr	15215
9180	TOSTADAS X 145 GR	1247
1986	POPETAS CARAMELO-QUESO FAMILIAR x 135 gr	556
2204	NACHOS X 180 GRS. FRESCAMPO	1711
7902	NUTRIBELA NUTRICION FRASCO x 300 ml	11973
3607	BIOEXPERT CHAMPU CAP.GUSANO SEDAx18x15ml	31473
7018	NUTRIBELA REPOLARIZACION x27 mlx12	24986
2129	SAVITAL CHAMPU BIOTINA x 550 ml.	1350
2873	EGO GEL EXTREME MAX MEGA x15 x27 ml.	8145
4086	ELITE SERVILLETA PRACTICA x 320	59703
2438	MILO TETRA PACK x 180 ml	2702

Figura 8.8 Tabla maestro de productos.

Vamos al lienzo y creamos una matriz con el nombre del producto y la medida costo total.

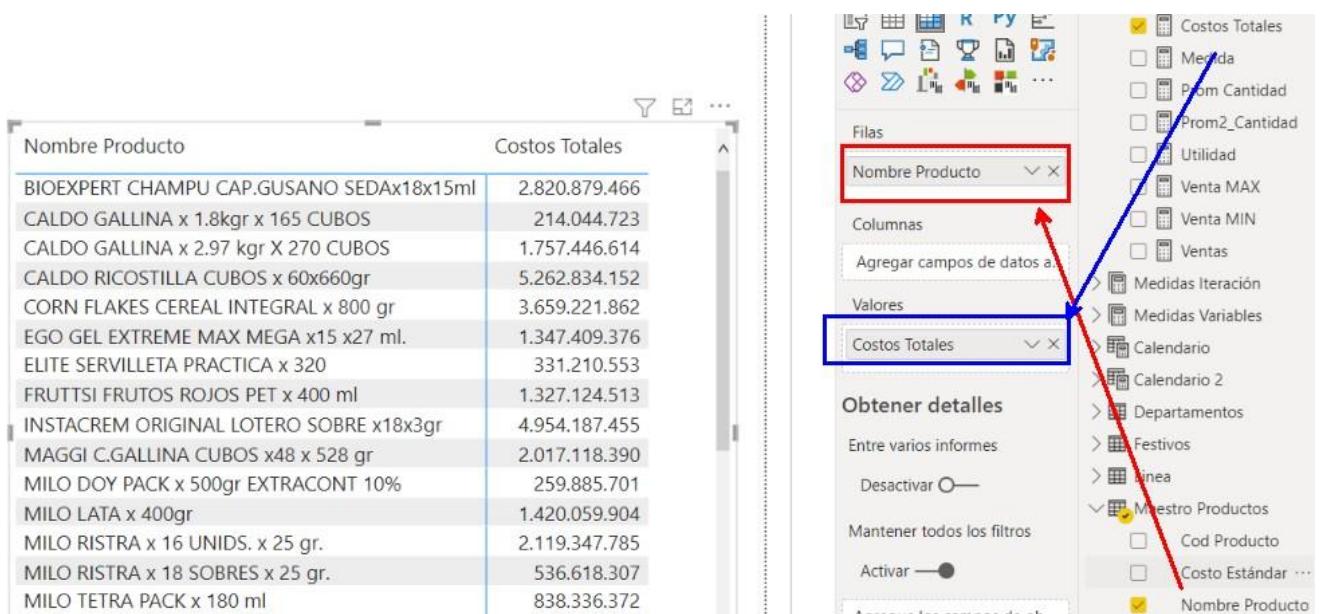


Figura 8.9 Matriz costo real producto.

Generemos una nueva medida que me calcule cual sería los costos estándar según el valor definido en la tabla *Maestro Productos*. Necesitamos crear una función que itere sobre la tabla *Ventas* en la columna *Cantidad* y luego multiplique esa cantidad vendida por los costos estándar de la tabla *productos*.

SUMX me permite generar esta iteración sobre la tabla *Ventas* en la columna *Cantidad* y con ayuda de RELATED podemos multiplicar por los costos estándar. La medida sería:

```

1 Costos Estándar =
2 SUMX (
3   Ventas,
4   Ventas[Cantidad] * RELATED ( 'Maestro Productos'[Costo Estándar] )
5 )

```

Nombre Producto	Costos Totales	Costos Estándar
BIOEXPERT CHAMPU CAP.GUSANO SEDAx18x15ml	2.820.879.466	2.826.306.873
CALDO GALLINA x 1.8kgr x 165 CUBOS	214.044.723	204.203.488
CALDO GALLINA x 2.97 kgr X 270 CUBOS	1.757.446.614	1.763.990.097
CALDO RICOSTILLA CUBOS x 60x660gr	5.262.834.152	5.284.302.716
CORN FLAKES CEREAL INTEGRAL x 800 gr	3.659.221.862	3.653.946.780
EGO GEL EXTREME MAX MEGA x15 x27 ml.	1.347.409.376	1.348.812.000
ELITE SERVILLETA PRACTICA x 320	331.210.553	331.232.244
FRUTTSI FRUTOS ROJOS PET x 400 ml	1.327.124.513	1.325.689.890
INSTACREM ORIGINAL LOTERO SOBRE x18x3gr	4.954.187.455	4.980.624.240
MAGGI C.GALLINA CUBOS x48 x 528 gr	2.017.118.390	2.017.013.132
MILO DOY PACK x 500gr EXTRACONT 10%	259.885.701	258.064.560
MILO LATA x 400gr	1.420.059.904	1.411.525.692
Total	**64.889.525.781**	**64.593.736.964**

Figura 8.10 Costo estándar en la matriz.

Una operación muy similar como cuando llevamos campos de las tablas maestras hacia ventas, pero ya en este caso hacemos referencia en la iteración a RELATED, detrás de cámara sería como si la columna *Costos Estándar* ya hiciera parte como columna en la tabla *Ventas*.

Sin necesidad de crear más columnas calculadas y afectar el rendimiento del informe, podemos hacer referencia a otras columnas de diferentes tablas en medidas.

Ejercicio práctico con RELATED

En el capítulo funciones de tabla, indicamos que podemos generar tablas a partir de una condición, como traer solo los días hábiles de la tabla calendario (lunes a viernes). En el modelo tenemos una tabla con festivos, en este caso para Colombia (también puedes escoger diferentes países que contamos en Power Query).

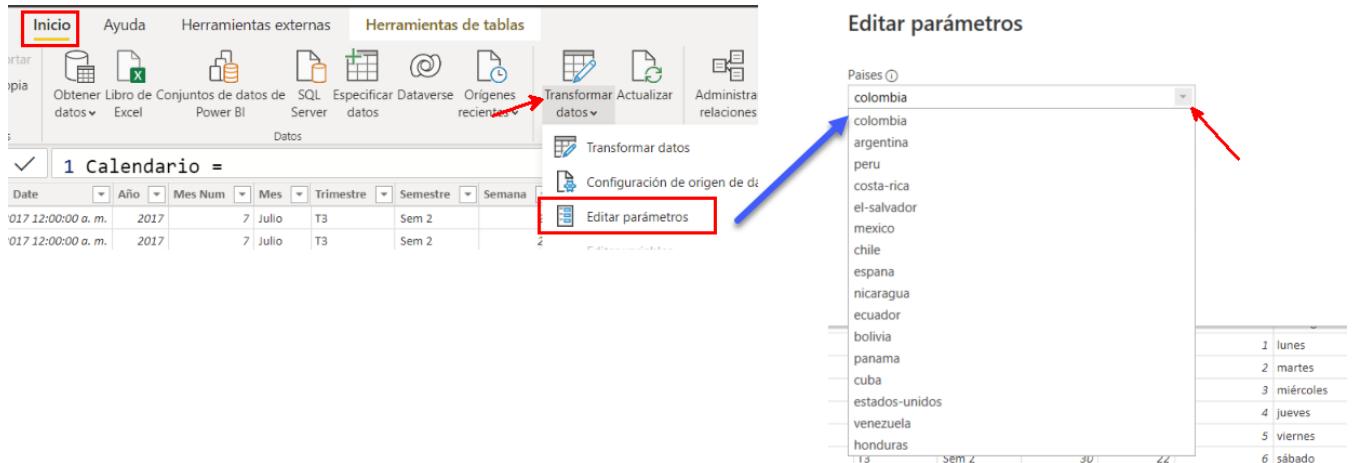


Figura 8.11 Editar país para festivos.

Adicional de la tabla que se creó con FILTER para contener solo días hábiles, necesitamos contarlos y excluir también los días feriados en una medida.

Para de ir de menos a más, llevemos a nuestra tabla calendario, las fechas donde hay festivos.

Seleccionamos la tabla *Calendario* → Nueva columna y con RELATED agregamos la columna Fecha de la tabla *Festivos*.

Se crea una nueva columna con los feriados, hay filas blancas y filas con fechas según la figura 8.12

Date	Año	Mes Num	Mes	Trimestre	Semestre	Semana	Dia Mes	Dia Semana	Nombre Día	Feriados
1/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	27	1		6 sábado	
2/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	27	2		7 domingo	
3/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	28	3	1	lunes	3/07/2017 12:00:00 a. m.
4/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	28	4	2	martes	
5/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	28	5	3	miércoles	
6/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	28	6	4	jueves	
7/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	28	7	5	viernes	
8/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	28	8	6	sábado	
9/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	28	9	7	domingo	
10/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	29	10	1	lunes	
11/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	29	11	2	martes	
12/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	29	12	3	miércoles	
13/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	29	13	4	jueves	
14/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	29	14	5	viernes	
15/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	29	15	6	sábado	
16/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	29	16	7	domingo	
17/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	30	17	1	lunes	
18/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	30	18	2	martes	
19/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	30	19	3	miércoles	
20/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 2	30	20	4	jueves	20/07/2017 12:00:00 a. m.
21/07/2017 12:00:00 a. m.	2017	7	Julio	T3	Sem 3	31	21	5	viernes	

Figura 8.12 Columna feriados.

Ahora simplemente es añadirle a la condición de los días hábiles que excluya las filas que contiene datos, ósea los festivos.

FILTER nos genera la tabla fragmentada, COUNTROWS nos da el número de registros.

```
1 # Días Hábiles =
2   FILTER (
3     Calendario,
4     Calendario[Dia Semana] <= 6
5     && Calendario[Feriados] = BLANK ()
6   )
```

Figura 8.13 Días hábiles sin feriados.

La imagen anterior tiene la tabla para los días hábiles sin feriados, para que la medida no genere error, la encerramos en COUNTROWS ya que debe generar como resultado un valor escalar.

Este resultado la llevamos a una matriz de mes por año.

El resultado es el esperado, nos cuenta cada mes el número de días hábiles que hay, esto es útil para hacer cálculo de proyecciones lineales.

```
1 # Días Hábiles =
2 COUNTROWS (
3   FILTER (
4     Calendario,
5     Calendario[Dia Semana] <= 6
6     && Calendario[Feriados] = BLANK ())
7 )
8 )
```



Año	Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio	Agosto	Septiembre	Octubre	Noviembre	Diciembre	Total
2017	25	24	26	23	25	24	24	25	26	25	24	24	295
2018	25	24	24	25	25	24	24	25	25	26	24	24	295
2019	25	24	25	24	26	23	25	25	25	26	24	25	297
2020	25	25	25	24	24	23	26	24	26	26	23	25	296
Total	100	97	100	96	100	94	99	99	102	103	95	98	1183

Figura 8.14 Matriz Días hábiles sin feriados.

Si observamos el cálculo, antes de llegar a él, se procedió a crear una columna calculada en la tabla *Calendario* y no está mal, pero si queremos mejorar el rendimiento del modelo, este proceso lo podemos omitir, gracias a que RELATED puede operar en las medidas como si estuviéramos creando columnas adicionales.

Cambiamos *Calendario [Feriados]* por RELATED (Festivos [Fecha]).

```

1 # Días Hábiles =
2   FILTER (
3     Calendario,
4     Calendario[Dia Semana] <= 6
5     && Calendario[Feriados] = BLANK ()
6   )
7
8 
```

```

1 # Días Hábiles =
2 COUNTROWS (
3   FILTER (
4     Calendario,
5     Calendario[Dia Semanal] <= 6
6     && RELATED( Festivos[Fecha]) = BLANK ())
7
8 ) 
```

Figura 8.15 RELATED en medida días hábiles.



```

# Días Hábiles =
COUNTROWS (
  FILTER (
    Calendario,
    Calendario[Dia Semana] <= 6
    && RELATED( Festivos[Fecha]) = BLANK ())
) 
```

Ejercicios propuestos

- Te proponemos que crees una medida que cuente el número de feriados por mes existentes en tu país.
- Crees una medida que calcule las nuevas ventas teniendo en cuenta los descuentos que cada línea tiene en la tabla LINEA.

RELATEDTABLE

A medida que avanzamos en DAX las funciones se van correlacionando una con otra, a diferencia de la anterior función, RELATEDTABLE devuelve una tabla como resultado y es usada especialmente en la tabla de los unos o tablas maestras cambiando el contexto en el que se filtran los datos y se evalúa la expresión en el nuevo contexto.

En resumen, RELATEDTABLE es un atajo de la función CALCULATETABLE sin expresiones lógicas, tema que veremos en el capítulo 11.

No puedes perder de vista que RELATEDTABLE devuelve una tabla como resultado, allí se encuentra todavía confusión en esto, veamos.

Nos ubicamos en la tabla *Línea* → Insertamos una nueva columna → y hacemos llamado a la función RELATEDTABLE a la tabla *Ventas*.

The screenshot shows a Power BI interface with a table editor. At the top, there is a formula bar with the text "1 Columna = RELATEDTABLE(Ventas)". Below it, a warning message says: "La expresión hace referencia a varias columnas. No se pueden convertir varias columnas a un valor escalar." (The expression refers to multiple columns. Multiple columns cannot be converted to a scalar value.) A dropdown menu labeled "Columna" is open, showing five entries: CAFÉ, ALIMENTO, BEBIDAS, CONDIMENTOS, and ASEO, each followed by "#ERROR".

Línea	Cod Línea	Descuentos	Columna
CAFÉ	CA - 01	0,06	#ERROR
ALIMENTO	AL - 01	0,08	#ERROR
BEBIDAS	BE - 01	0,05	#ERROR
CONDIMENTOS	CO - 01	0,02	#ERROR
ASEO	AS - 01	0,03	#ERROR

Figura 8.16 Error RELATEDTABLE.

Si leemos detalladamente el error, la función hace referencia a varias columnas y esto es imposible procesar para DAX, ya que el contexto donde se está evaluando la expresión es una columna, un contexto de fila, y en una sola celda DAX no puede ingresar todas columnas de la tabla *Ventas*.

La verdad RELATEDTABLE es una de estas funciones que siempre deben ir acompañadas de otras funciones para devolver un valor escalar y cambiar estos contextos, tema que debes leer cuidadosamente en el siguiente capítulo.

Algo que si podemos realizar es contar el número de registros que cada línea tiene en la tabla *Ventas*.

Simplemente encerramos RELATEDTABLE entre la función COUNTROWS.

The screenshot shows a Power BI interface with a table editor. The formula bar at the top has the text "1 #Registros = COUNTROWS(RELATEDTABLE(Ventas))". The table below has four columns: Cod Línea, Descuentos, and two additional columns labeled "#Registros" which contain the values 10569, 11327, 18673, 6810, and 11869 respectively. The last column is highlighted with a red box.

Cod Línea	Descuentos	#Registros
CA - 01	0,06	10569
AL - 01	0,08	11327
BE - 01	0,05	18673
CO - 01	0,02	6810
AS - 01	0,03	11869

Figura 8.17 Número registros por línea.

COUNTROWS su resultado es un valor escalar. Si sumamos la columna #Registros debe ser exactamente al número de filas totales que se encuentran en la tabla *Ventas*, solo que en este caso está filtrado por línea.



#Registros = COUNTROWS(RELATEDTABLE(Ventas))

En vez de crear un conteo, también podemos crear una suma por línea.

Insertar columna nueva → Total Línea → Reemplazamos COUNTROWS por SUMX.

Cod Linea	Descuentos	#Registros	Total Línea
CA - 01	0,06	10569	13.410.774,469
AL - 01	0,08	11327	21.653.792,830
BE - 01	0,05	18673	32.452.118,734
CO - 01	0,02	6810	13.485.438,855
AS - 01	0,03	11869	15.775.403,827

Figura 8.18 Total venta por línea.



Total Línea =

SUMX (RELATEDTABLE (Ventas), Ventas[Cantidad] * Ventas[Precio Unit])

RELATEDTABLE en medidas

Podemos crear análisis interesantes gracias a esta función, que permite cambiar los contextos y generar medidas de gran valor. Nos piden saber cuál es última fecha de compra de cada cliente para luego filtrar aquellos que tienen más de 60 o 90 días sin venta y poder establecer una estrategia.

Lo primero es saber que debemos usar una función que encaje con RELATEDTABLE, que reciba como parámetro una tabla y calcule el valor o fecha máxima.

Si analicemos el ejercicio anterior, nos puede dar algunas pistas para esta posible solución, ya que las funciones de agregación que pueden iterar reciben como argumento una tabla y se ejecuta el cálculo indicado en el prefijo. MAXX es la solución, recibe una tabla como argumento en este caso sería RELATEDTABLE y como expresión estaría definida la columna fecha factura.

Generamos una matriz con el nombre del cliente y escribimos la siguiente medida.

1 Fecha Ultima Compra =	
2 MAXX (RELATEDTABLE (Ventas), Ventas[Fecha Factura])	
Nombre Cliente	Fecha Ultima Compra
ABSORBENTES DE COLOMBIA S.A.	21/11/2020 12:00:00 a.m.
ABURRA LTDA.	20/12/2020 12:00:00 a.m.
ALIMENTOS DF SAS	26/12/2020 12:00:00 a.m.
ALIMENTOS Y VINOS DE ESPAÑA S.A.S.	09/11/2020 12:00:00 a.m.
AZUL K S.A	24/12/2020 12:00:00 a.m.
BODEGAS SANTA LUCIA S.A.S	26/12/2020 12:00:00 a.m.
CIPA S.A	27/12/2020 12:00:00 a.m.
COLGATE PALMOLIVE CIA	17/12/2020 12:00:00 a.m.

Figura 8.19 Fecha última compra.

MAXX itera sobre la columna cliente hasta traer la última fecha de compra, esta medida también se puede crear como columna calculada en el maestro de clientes, de productos, de línea etc.



Fecha Ultima Compra =

MAXX (RELATEDTABLE (Ventas), Ventas[Fecha Factura])

Para completar la diferencia entre el número de días y la última fecha de compra, solo es hacer la resta con INT (TODAY () – [Fecha Ultima Compra]).



Advertencia: Si se agrega un filtro por año y siempre se quiere ver la fecha de la última compra omitiendo este filtro, lo estudiaremos en el capítulo 10, con los modificadores de CALCULATE



Nota: RELADTETABLE es una expresión corta de lo que es CALCULATETABLE, pero sin los argumentos de filtros

Ejercicios propuestos

- En la tabla Maestro Productos, crear una columna calculada para llevar la primera fecha de venta
- En la tabla Maestro Productos, crear una columna calculada para llevar la última fecha de venta
- Calcular el número de días entre la primera y la última fecha de venta de cada producto

LOOKUPVALUE

En el desarrollo de nuevos modelos, nos vamos a encontrar con tablas desconectadas (no relacionadas), y allí no podríamos usar RELATED ni RELATEDTABLE porque es de carácter obligatorio que exista una relación física, aunque LOOKUPVALUE no hace parte de las funciones de relación, encaja perfectamente en este ejercicio, porque en esta sesión vamos a demostrar la importancia de las relaciones, del modelo y todas las buenas prácticas antes mencionadas.

En la figura 8.1 tenemos la tabla *TRM* que no se encuentra relacionada con ninguna otra tabla, ya que en el estado que se encuentra no se puede relacionar, TRM es la tasa representativa del mercado para nuestro país Colombia o tasa de cambio, si la relacionamos sería una relación de muchos a muchos y en el capítulo 2 indicamos el riesgo que se corre al usar este tipo de relación.

Mes	Año	TASA PROM
Diciembre	2020	3468,50380952381
Noviembre	2020	3675,2075
Octubre	2020	3834,21863636364
Septiembre	2020	3752,10666666667
Agosto	2020	3788,09894736842
Julio	2020	3660,59909090909
Junio	2020	3691,56666666667
Mayo	2020	3856,116
Abril	2020	3986,561
Marzo	2020	3886,515

Figura 8.20 Tabla TRM.

La tabla cuenta con el mes que se repite y el año de igual forma y el ejercicio es convertir las Ventas que tenemos en pesos colombianos según la tasa de cambio promedio por el año y mes. LOOKUPVALUE tiene la capacidad de realizar búsquedas en otras tablas que no se encuentra relación alguna.

Veamos su sintaxis antes de:

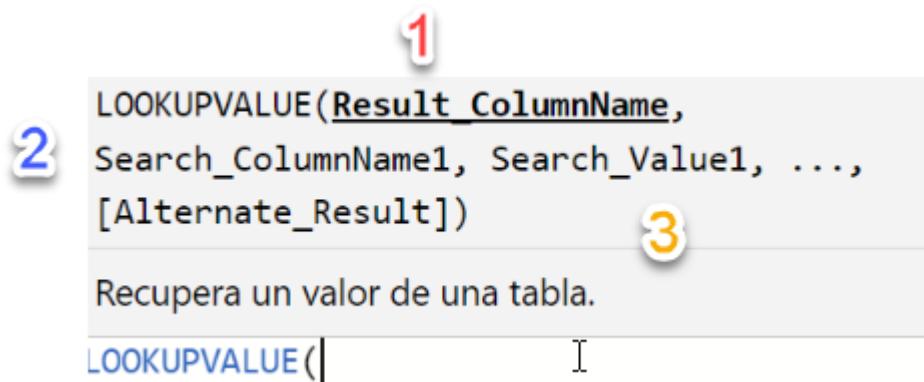


Figura 8.21 Sintaxis LOOKUPVALUE.

La función tiene 3 argumentos obligatorios:

1. Resultado de la columna, es el resultado que deseamos ver en dicha operación, para este caso sería la columna TASA PROM.
2. Columna buscada 1, es el primer valor para buscar en la tabla donde está la tasa de cambio, para este caso puede ser año o mes.
3. Valor buscado, es el valor para buscar en la columna buscada 1.

Los dos últimos argumentos se pueden repetir, ya que LOOKUPVALUE tiene la capacidad de hacer búsquedas con varios argumentos.

El objetivo es identificar en la tabla *Ventas* de la fecha de factura el mes y el año, y con estos dos registros buscarlos en la tabla *TRM* y llevar con que tasa de cambio se va a hacer la operación de ese registro.

Si se generó una venta en 3 de diciembre de 2020 la tasa de cambio para ese día sería de 3.468,50, según la figura 8.20

Insertemos una nueva columna en la tabla *Ventas* → Llamada *TRM* → Y completamos la sintaxis indicada.

Hay que recordar que en la tabla *Ventas* no contamos con el mes y el año desagregado de la fecha factura, pero implícitamente la podemos añadir.

Veamos esto paso a paso:

El primer argumento es el objetivo, sería *TRM PROM*, luego de que año va a ser esta *TRM*, la columna año de la misma tabla.

```
1 TRM = LOOKUPVALUE(TRM[TASA PROM], TRM[Año])
```

Figura 8.22 Valor objetivo y primer valor a buscar.

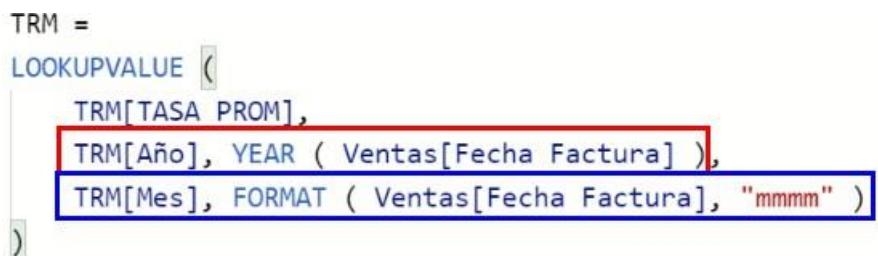
Tenemos el objetivo definido y la primera columna de este objetivo, luego el valor a buscar en la primera columna objetivo, como es año, debemos extraer el año de la fecha factura.



```
LOOKUPVALUE(TRM[TASA PROM], TRM[Año],  
YEAR(Ventas[Fecha Factura]))
```

Figura 8.23 Año buscado.

Si lo dejamos hasta allí, generaría un error, ya que el valor objetivo debe ser un único valor, y hay muchas tasas de cambio para el mismo año. Por consiguiente, debemos buscar también el mes de la fecha factura en la tabla *TRM*. Seguimos la misma lógica de la imagen 8.24



```
TRM =  
LOOKUPVALUE (  
    TRM[TASA PROM],  
    TRM[Año], YEAR ( Ventas[Fecha Factura] ),  
    TRM[Mes], FORMAT ( Ventas[Fecha Factura], "mmmm" ))
```

Figura 8.24 Función traer TRM promedio año y mes.



```
TRM =  
LOOKUPVALUE (  
    TRM[TASA PROM],  
    TRM[Año], YEAR ( Ventas[Fecha Factura] ),  
    TRM[Mes], FORMAT ( Ventas[Fecha Factura], "mmmm" ))
```

Si damos Enter, el resultado es el correcto en la columna calculada, solo nos queda crear una medida donde se divida el valor de la venta sobre la TRM promedio. Llevamos esta medida a la matriz año y mes.

1	Ventas USD =								
2	SUMX (Ventas, (Ventas[Precio Unit] * Ventas[Cantidad]) / Ventas[TRM])								
Y E ...									
Año	Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio	Agosto	Septi
2017	348.788,74	493.964,57	578.537,72	662.630,01	594.593,87	619.831,30	532.335,87	832.883,99	72
2018	549.532,72	659.455,62	726.389,27	746.027,50	719.842,82	680.407,19	733.078,41	829.516,18	70
2019	508.706,06	656.568,71	713.779,25	666.066,33	691.254,20	622.655,12	618.919,03	663.966,81	62
2020	488.563,71	611.232,80	550.709,86	526.995,60	636.743,83	553.803,60	642.048,99	546.636,27	58
Total	1.895.591,23	2.421.221,70	2.569.416,09	2.601.719,44	2.642.434,71	2.476.697,21	2.526.382,31	2.873.003,26	2.63

Figura 8.25 Ventas en USD por año y mes.



Ventas USD =

SUMX (Ventas, (Ventas[Precio Unit] * Ventas[Cantidad]) / Ventas[TRM])

Hemos llegado a la solución, pero usando columnas calculadas de por medio como es **Ventas [TRM]**, como el objetivo es omitir al máximo la creación de columnas, te damos la medida que puedes usar y generar el mismo resultado sin generar columnas adicionales. En este punto puedes entender dicha medida.

```

1 USD_Sin_Columnas =
2 VAR Resumen =
3   ADDCOLUMNS (
4     SUMMARIZE ( Ventas, Calendario[Año], Calendario[Mes] ),
5     "@ValorCOP", [Ventas]
6   )
7 VAR TRM_PROM =
8   ADDCOLUMNS (
9     Resumen,
10    "@TRMPROM",
11    LOOKUPVALUE (
12      TRM[TASA PROM],
13      TRM[Año], Calendario[Año],
14      TRM[Mes], Calendario[Mes]
15    )
16  )
17 RETURN
18  SUMX ( TRM_PROM, [@ValorCOP] / [@TRMPROM] )

```

Figura 8.26 Ventas en USD sin columnas calculadas.



```
USD_Sin_Columnas =  
VAR Resumen =  
    ADDCOLUMNS (  
        SUMMARIZE ( Ventas, Calendario[Año], Calendario[Mes] ),  
        "@ValorCOP", [Ventas]  
    )  
VAR TRM_PROM =  
    ADDCOLUMNS (  
        Resumen,  
        "@TRMPROM",  
        LOOKUPVALUE (   
            TRM[TASA PROM],  
            TRM[Año], Calendario[Año],  
            TRM[Mes], Calendario[Mes]  
        )  
    )  
RETURN  
    SUMX ( TRM_PROM, [@ValorCOP] / [@TRMPROM] )
```

Analiza bien esta medida, y es la combinación de todo lo visto hasta este punto. Creamos una tabla con SUMMARIZE para llevar las ventas por año y mes, luego hacemos uso de esta tabla en otra variable para llevar la TRM promedio e iteramos por último este resultado con un SUMX.

Columna Llave para relacionar tablas

Hasta este punto queríamos llegar, cuando no tenemos las tablas relacionadas nos toca acudir a esta serie de procesos intermedios o crear medidas un poco más complejas, pero si pudiésemos relacionar la tabla *TRM* con *Calendario* por medio del año y del mes nuestra medida sería mas fácil de crear y obtenemos mejor rendimiento.

Generamos una copia de la tabla *TRM* y concatenamos la columna Año y Mes en una sola.

Mes	Año	TASA PROM	Llave
Diciembre	2020	3468,5038095238	2020Diciembre
Noviembre	2020	3675,2075	2020Noviembre
Octubre	2020	3834,2186363636	2020Octubre
Septiembre	2020	3752,1066666666	2020Septiembre
Agosto	2020	3788,0989473684	2020Agosto
Julio	2020	3660,5990909090	2020Julio
Junio	2020	3691,5666666666	2020Junio

Figura 8.27 Columna Concatenada en tabla TRM 2.

Esta columna también la podemos crear desde Power Query.

El mismo proceso lo realizamos en la tabla *Calendario*, enlazamos año y mes.

Llave = Calendario[Año]&Calendario[Mes]													
	Año	Mes Num	Mes	Trimestre	Semestre	Semana	Dia Mes	Dia Semana	Nombre Día	Feriados	Llave		
00 a. m.	2017	7	Julio	T3	Sem 2		27	1	6 sábado		2017Julio		
00 a. m.	2017	7	Julio	T3	Sem 2		27	2	7 domingo		2017Julio		
00 a. m.	2017	7	Julio	T3	Sem 2		28	3	1 lunes	3/07/2017 12:00:00 a. m.	2017Julio		
00 a. m.	2017	7	Julio	T3	Sem 2		28	4	2 martes		2017Julio		
00 a. m.	2017	7	Julio	T3	Sem 2		28	5	3 miércoles		2017Julio		
00 a. m.	2017	7	Julio	T3	Sem 2		28	6	4 jueves		2017Julio		
00 a. m.	2017	7	Julio	T3	Sem 2		28	7	5 viernes		2017Julio		
00 a. m.	2017	7	Julio	T3	Sem 2		28	8	6 sábado		2017Julio		

Figura 8.28 Columna Concatenada en tabla Calendario.

Con ayuda de estas columnas llaves procedemos a crear la relación de las dos tablas.

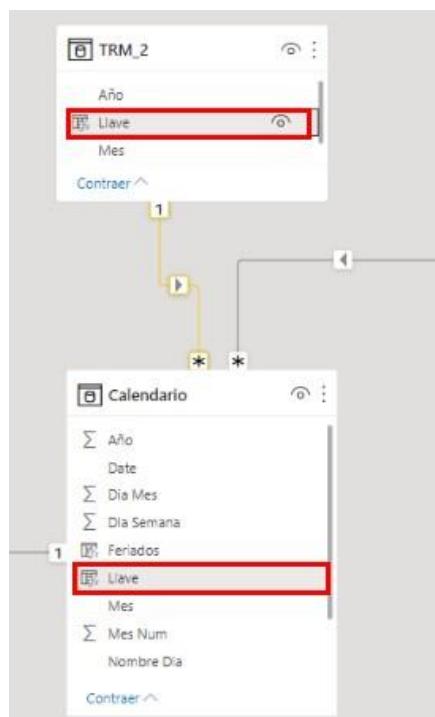


Figura 8.29 Relación TRM 2 con tabla calendario.

Con esta relación física y con los conocimientos adquiridos en este capítulo podemos realizar una medida que me convierta en USD las ventas.

```

1 USD Con Relación =
2 SUMX (
3     Ventas,
4     ( Ventas[Precio Unit] * Ventas[Cantidad] )
5     / RELATED ( TRM_2[TASA PROM] )
6 )

```

Año	Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio	Agosto
2017	348.788,74	493.964,57	578.537,72	662.630,01	594.593,87	619.831,30	532.335,87	832.883,99
2018	549.532,72	659.455,62	726.389,27	746.027,50	719.842,82	680.407,19	733.078,41	829.516,18
2019	508.706,06	656.568,71	713.779,25	666.066,33	691.254,20	622.655,12	618.919,03	663.966,81
2020	488.563,71	611.232,80	550.709,86	526.995,60	636.743,83	553.803,60	642.048,99	546.636,27
Total	1.895.591,23	2.421.221,70	2.569.416,09	2.601.719,44	2.642.434,71	2.476.697,21	2.526.382,31	2.873.003,26

Figura 8.30 Medida con tablas relacionadas.



```
USD Con Relación =  
SUMX (   
    Ventas,  
    ( Ventas[Precio Unit] * Ventas[Cantidad] )  
        / RELATED ( TRM_2[TASA PROM] )  
)
```

Solo por el hecho de existir una relación entre las tablas, nos evitamos crear medidas complejas y proceso intermedio que nos afecten el rendimiento del modelo.



Nota: En lo posible relacione todas las tablas, evite tener tablas desconectadas y así podrá generar cálculos más rápidos y de la manera fácil, use el poder de lo simple.

Capítulo 9: Evaluación de contextos

Llegados a este punto, ya hemos aprendido los conceptos básicos de DAX. Hemos adquirido habilidades de modelamiento y relaciones, como también aprendido a crear columnas y medidas calculadas, conocemos ya las funciones más básicas de DAX, condiciones, variables, operadores y otros conceptos importantes.

Con los conocimientos que ha logrado hasta ahora a través de este libro, ya está preparado para crear muchos informes interesantes, sin embargo, como lo mencionábamos al inicio del libro en el capítulo 1 aprender acerca de la evaluación de contextos es la clave para conocer todos los secretos que esconde DAX y es esta la base sobre la que podrá fundamentar todas las características avanzadas de DAX que aprenda en el futuro.

Adicionalmente, los contextos de evaluación juegan un papel fundamental cuando se utiliza la función CALCULATE, esta función es quizá la función más potente que tiene DAX e igualmente quizá la más difícil de aprender para muchos. Por esta razón dedicamos un apartado entero para esta función en el capítulo 10 y algunos conceptos adicionales en el capítulo 11. No sin previamente haber estudiado y comprendido los contextos de evaluación ya que entender la función CALCULATE antes de esto sería problemático. En dichos capítulos terminará de convencerse del porqué de la importancia de la evaluación de contextos utilizando la función CALCULATE.

Introducción a la evaluación de contextos

En el lenguaje DAX existen dos contextos de evaluación, el contexto de filtro y el contexto de fila. Antes de describirlos, es necesario aclarar que son dos conceptos muy diferentes con funcionalidades y aplicaciones completamente diferentes.

No se deben confundir los dos contextos como si uno fuera la variación del otro. El contexto de filtro filtra datos y el contexto de fila itera tablas. Por lo tanto, si DAX está iterando, no está filtrando; y cuando está filtrando, no está iterando. Aunque escrito es un concepto simple de entender materializarlo en la mente puede tornarse difícil. En palabras simples, el contexto de filtro filtra, el contexto de fila itera, jamás serán lo mismo y en ninguna circunstancia debe confundirlos o pensar que son similares.

Teniendo en cuenta que los dos contextos son diferentes, es importante destacar que se deben tener en cuenta ambos contextos si queremos entender correctamente el comportamiento de una expresión o fórmula, ya que ambos actúan conjuntamente, estos representan el entorno sobre el cuál las expresiones DAX actuarán sobre nuestro modelo, pudiendo proporcionarnos diferentes resultados en función del contexto en el que se ejecuten, de ahí la importancia de comprenderlos correctamente.

Comprendiendo los contextos de filtro

Para empezar a comprender el contexto de filtro de una manera práctica, comencemos por analizar una medida como la siguiente:



Total Ventas = `SUMX (Ventas, Ventas[Cantidad] * Ventas[Precio Unit])`

El resultado que arrojaría esta fórmula es la suma de la cantidad multiplicada por el precio, ambos valores pertenecen a la tabla Ventas, si utilizamos esta métrica en una tabla para ver los resultados, se vería como en la Figura 9.1

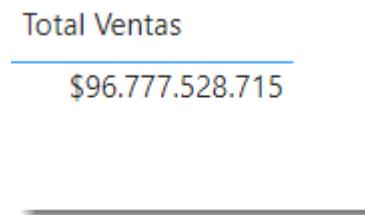


Figura 9.1 La métrica Total Ventas, sin ningún contexto, muestra el gran total de las ventas

Este resultado por sí sólo no nos arroja mucha información sobre la cual se pudiera hacer análisis, no obstante, viéndolo objetivamente la métrica muestra lo que debería: la suma de todos los valores de ventas. Ahora si lo vemos desde otra perspectiva y a este resultado lo cruzamos con los productos, la matriz resultante ya comienza a mostrarnos valores con conclusiones mucho más interesante como lo podemos ver en la Figura 9.2

Nombre Producto	Total Ventas
BIOEXPERT CHAMPU CAP.GUSANO SEDAx18x15ml	\$4.256.483.181
CALDO GALLINA x 1.8kgr x 165 CUBOS	\$329.762.199
CALDO GALLINA x 2.97 kgr X 270 CUBOS	\$2.692.701.294
CALDO RICOSTILLA CUBOS x 60x660gr	\$7.259.735.576
CORN FLAKES CEREAL INTEGRAL x 800 gr	\$4.548.499.194
EGO GEL EXTREME MAX MEGA x15 x27 ml.	\$2.280.306.567
ELITE SERVILLETA PRACTICA x 320	\$914.725.625
FRUTTSI FRUTOS ROJOS PET x 400 ml	\$2.178.092.924
INSTACREM ORIGINAL LOTERO SOBRE x18x3gr	\$6.694.513.472
MAGGI C.GALLINA CUBOS x48 x 528 gr	\$3.203.239.786
MILO DOY PACK x 500gr EXTRACONT 10%	\$460.687.466
MILO LATA x 400gr	\$2.202.462.729
MILO RISTRA x 16 UNIDS. x 25 gr.	\$3.706.246.914
MILO RISTRA x 18 SOBRES x 25 gr.	\$992.592.502
MILO TETRA PACK x 180 ml	\$1.372.798.912
Total	\$96.777.528.715

Figura 9.2 El total de ventas se ve segmentado por producto.

Si observamos nuevamente la Figura 9.2, podemos ver que el gran total es el mismo de la Figura 9.1, sin embargo, ya es la suma de los valores vendidos de cada producto, el conjunto de todos los valores nos da ahora información más detallada, no obstante, seamos más críticos al respecto, la fórmula ya no está calculando el total de todas las ventas, que, al fin y al cabo, fue lo que codificamos, de hecho, la métrica la llamamos “Total Ventas”. En contraste, la métrica está calculando las ventas de un producto específico para cada fila. Aun así, en ninguna parte del código le estamos especificando que la métrica debería funcionar en subconjuntos de datos. Este filtro está ocurriendo por fuera de la fórmula.

Entonces, ¿por qué los valores de cada fila son diferentes si la fórmula es exactamente la misma?

Nombre Producto	Total Ventas	
BIOEXPERT CHAMPU CAP.GUSANO SEDAx18x15ml	\$4.256.483.181	Total Ventas = SUMX(Ventas, Ventas[Cantidad] * Ventas[Precio Unit])
CALDO GALLINA x 1.8kgr x 165 CUBOS	\$329.762.199	Total Ventas = SUMX(Ventas, Ventas[Cantidad] * Ventas[Precio Unit])
CALDO GALLINA x 2.97 kgr X 270 CUBOS	\$2.692.701.294	Total Ventas = SUMX(Ventas, Ventas[Cantidad] * Ventas[Precio Unit])
CALDO RICOSTILLA CUBOS x 60x660gr	\$7.259.735.576	Total Ventas = SUMX(Ventas, Ventas[Cantidad] * Ventas[Precio Unit])
CORN FLAKES CEREAL INTEGRAL x 800 gr	\$4.548.499.194	Total Ventas = SUMX(Ventas, Ventas[Cantidad] * Ventas[Precio Unit])
EGO GEL EXTREME MAX MEGA x15 x27 ml.	\$2.280.306.567	Total Ventas = SUMX(Ventas, Ventas[Cantidad] * Ventas[Precio Unit])
ELITE SERVILLETA PRACTICA x 320	\$914.725.625	Total Ventas = SUMX(Ventas, Ventas[Cantidad] * Ventas[Precio Unit])
FRUTTSI FRUTOS ROJOS PET x 400 ml	\$2.178.092.924	Total Ventas = SUMX(Ventas, Ventas[Cantidad] * Ventas[Precio Unit])
INSTACREM ORIGINAL LOTERO SOBRE x18x3gr	\$6.694.513.472	Total Ventas = SUMX(Ventas, Ventas[Cantidad] * Ventas[Precio Unit])
MAGGI C.GALLINA CUBOS x48 x 528 gr	\$3.203.239.786	Total Ventas = SUMX(Ventas, Ventas[Cantidad] * Ventas[Precio Unit])
MILO DOY PACK x 500gr EXTRACONT 10%	\$460.687.466	Total Ventas = SUMX(Ventas, Ventas[Cantidad] * Ventas[Precio Unit])
MILO LATA x 400gr	\$2.202.462.729	Total Ventas = SUMX(Ventas, Ventas[Cantidad] * Ventas[Precio Unit])
MILO RISTRA x 16 UNIDS. x 25 gr.	\$3.706.246.914	
MILO RISTRA x 18 SOBRES x 25 gr.	\$992.592.502	
MILO TETRA PACK x 180 ml	\$1.372.798.912	
Total	\$96.777.528.715	

Figura 9.3 Se muestra que a pesar de que en cada fila de la matriz el valor Total Ventas se calcula utilizando la misma fórmula, los valores son diferentes por fila.

Cada fila calcula un valor distinto debido al contexto de evaluación bajo el cuál en este escenario DAX está ejecutando la fórmula. Para comprenderlo mejor veámoslo de la siguiente manera: Cada valor en el área de valores de la matriz (En este caso tenemos sólo la métrica Total Ventas en esta área de valores) tiene un contexto diferente, es decir, cada celda en la que hay un valor para Total Ventas posee un contexto de filtro diferente. Tomemos entonces la posición del primer valor de la Figura 9.2 que corresponde a \$4.256.483.181. ¿Cuáles son los filtros? Una forma fácil de imaginarse el contexto de evaluación en este caso puede ser todo lo que rodea la celda del valor en cuestión, es decir, todos los campos que no están en el área de valores.

Para este caso puntual el filtro sería:

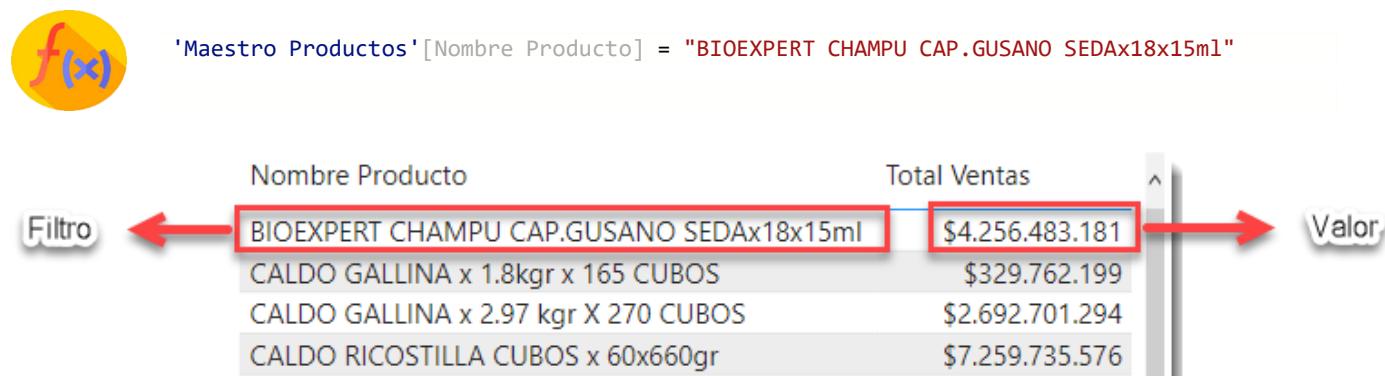


Figura 9.4 Descripción gráfica del contexto de filtro para el valor de la primera celda.



DAX evalúa todas las fórmulas dentro de un contexto respectivo. Aunque la fórmula es la misma, el resultado es diferente porque DAX ejecuta el mismo código en diferentes subconjuntos de datos.

Ahora, vayamos mucho más al detalle para entenderlo mejor, si vamos directamente a las tablas Ventas y Maestro Productos y visualizamos los valores de ambas tablas en Power BI tendríamos algo como lo siguiente:

Fecha Factura	Nombre Cliente	Cod. Vendedor	Cod. Producto	Cantidad	Precio Unit.	CostoUnit	Cod Lineas	Documento
domingo, 22 de enero de 2017	ILUMAX S.A.	B-01	3607	2	42990	23906	AS -01	7924355
domingo, 22 de enero de 2017	ILUMAX S.A.	B-01	7018	2	42990	23257	AS -01	7924251
miércoles, 8 de febrero de 2017	ILUMAX S.A.	B-01	3607	2	42990	23911	AS -01	7726194
miércoles, 8 de febrero de 2017	ILUMAX S.A.	B-01	4307	2	42990	24189	BE -01	7726245
jueves, 23 de febrero de 2017	ILUMAX S.A.	B-01	7018	2	42990	23277	AS -01	6728254
sábado, 11 de marzo de 2017	ILUMAX S.A.	B-01	4307	2	42990	24208	BE -01	0931292
sábado, 11 de marzo de 2017	ILUMAX S.A.	B-01	3607	2	42990	23914	AS -01	0931142
sábado, 15 de abril de 2017	ILUMAX S.A.	B-01	4307	2	42990	24209	BE -01	9534143
sábado, 15 de abril de 2017	ILUMAX S.A.	B-01	3607	2	42990	23914	AS -01	9534278
sábado, 15 de abril de 2017	ILUMAX S.A.	B-01	7018	2	42990	23278	AS -01	9534226
viernes, 28 de abril de 2017	ILUMAX S.A.	B-01	4307	2	42990	24209	BE -01	8136316
sábado, 13 de mayo de 2017	ILUMAX S.A.	B-01	7018	2	42990	23278	AS -01	8938112
viernes, 26 de mayo de 2017	ILUMAX S.A.	B-01	3607	2	42990	25674	AS -01	8940223
viernes, 26 de mayo de 2017	ILUMAX S.A.	B-01	7018	2	42990	23634	AS -01	8940382
viernes, 9 de junio de 2017	ILUMAX S.A.	B-01	4307	2	42990	24954	BE -01	4642137
viernes, 16 de junio de 2017	ILUMAX S.A.	B-01	3607	2	42990	24553	AS -01	4443342
viernes, 28 de febrero de 2020	ILUMAX S.A.	B-01	1317	2	35719	23313	BE -01	6716244

Cod Producto	Nombre Producto	Costo Estándar
8928	CORN FLAKES CEREAL INTEGRAL x 800 gr	20110
1738	NESTUM 5 CEREALES x 200 gr	19096
2303	QUIPITOS SOBRE x 24 x 8gr	15215
9180	TOSTADAS X 145 GR	1247
1986	POPETAS CARAMELO-QUESO FAMILIAR x 135 gr	556
2204	NACHOS X 180 GR. FRESCAMPO	1711
7902	NUTRIBELA NUTRICION FRASCO x 300 ml	11973
3607	BIOEXPERT CHAMPU CAP.GUSANO SEDAx18x15ml	31473
7018	NUTRIBELA REPOLARIZACION x27 mlx12	24986

Figura 9.5 Registros de las tablas Ventas y Maestro Productos

Ahora es necesario tomar los filtros identificados y aplicarlos a la tabla Ventas, en este caso tomaremos el valor de la columna Cod producto para filtrar, ya que esta es la llave de relación en ambas tablas, por lo tanto, es la columna por la cual también se está propagando el filtro de Maestro Productos hacia Ventas, entonces tenemos que 3607 es el código del producto identificado anteriormente para “BIOEXPERT CHAMPU CAP.GUSANO SEDAx18x15ml”

Fecha Factura	Nombre Cliente	Cod. Vendedor	Cod. Producto	Cantidad	Precio Unit.	CostoUnit	Cod Lineas	Documento
domingo, 22 de enero de 2017	ILUMAX S.A.	B-01	3607	2	42990	23906	AS -01	7924355
domingo, 22 de enero de 2017	ILUMAX S.A.	B-01	7018	2	42990	23257	AS -01	7924251
miércoles, 8 de febrero de 2017	ILUMAX S.A.	B-01	3607	2	42990	23911	AS -01	7726194
miércoles, 8 de febrero de 2017	ILUMAX S.A.	B-01	4307	2	42990	24189	BE -01	7726245
jueves, 23 de febrero de 2017	ILUMAX S.A.	B-01	7018	2	42990	23277	AS -01	6728254
sábado, 11 de marzo de 2017	ILUMAX S.A.	B-01	4307	2	42990	24208	BE -01	0931292
sábado, 11 de marzo de 2017	ILUMAX S.A.	B-01	3607	2	42990	23914	AS -01	0931142
sábado, 15 de abril de 2017	ILUMAX S.A.	B-01	4307	2	42990	24209	BE -01	9534143
sábado, 15 de abril de 2017	ILUMAX S.A.	B-01	3607	2	42990	23914	AS -01	9534278
sábado, 15 de abril de 2017	ILUMAX S.A.	B-01	7018	2	42990	23278	AS -01	9534226
viernes, 28 de abril de 2017	ILUMAX S.A.	B-01	4307	2	42990	24209	BE -01	8136316
sábado, 13 de mayo de 2017	ILUMAX S.A.	B-01	7018	2	42990	23278	AS -01	8938112
viernes, 26 de mayo de 2017	ILUMAX S.A.	B-01	3607	2	42990	25674	AS -01	8940223
viernes, 26 de mayo de 2017	ILUMAX S.A.	B-01	7018	2	42990	23634	AS -01	8940382
viernes, 9 de junio de 2017	ILUMAX S.A.	B-01	4307	2	42990	24954	BE -01	4642137
viernes, 16 de junio de 2017	ILUMAX S.A.	B-01	3607	2	42990	24553	AS -01	4443342
viernes, 28 de febrero de 2020	ILUMAX S.A.	B-01	1317	2	35719	23313	BE -01	6716244

Cod Producto	Nombre Producto	Costo Estándar
8928	CORN FLAKES CEREAL INTEGRAL x 800 gr	20110
1738	NESTUM 5 CEREALES x 200 gr	19096
2303	QUIPITOS SOBRE x 24 x 8gr	15215
9180	TOSTADAS X 145 GR	1247
1986	POPETAS CARAMELO-QUESO FAMILIAR x 135 gr	556
2204	NACHOS X 180 GR. FRESCAMPO	1711
7902	NUTRIBELA NUTRICION FRASCO x 300 ml	11973
3607	BIOEXPERT CHAMPU CAP.GUSANO SEDAx18x15ml	31473
7018	NUTRIBELA REPOLARIZACION x27 mlx12	24986

Figura 9.6 Aplicación del filtro de producto en la tabla Ventas

Básicamente lo que estaría haciendo DAX en el escenario específico de la celda que tomamos para el ejemplo es filtrar en la tabla Ventas los valores del producto en cuestión, y, por lo tanto, haciendo la operación que le indicamos en la fórmula: [Cantidad] * [Precio Unit], al final el resultado de la primera celda de la Figura 9.4 sería la sumatoria del resultado de esta multiplicación para todas las filas de la tabla Ventas con el filtro aplicado.

Este contexto se denomina Contexto de filtro y como su nombre indica, es un contexto que filtra las tablas. Cualquiera que sea la fórmula creada tendrá un valor diferente según el contexto de filtro utilizado para realizar la evaluación. Este comportamiento, aunque intuitivo, debe entenderse bien porque oculta muchas complejidades. El motor puede realizar algún nivel de optimización interna para mejorar la velocidad del cálculo, pero debe tener en cuenta que cada uno tiene una evaluación diferente y propia de la expresión DAX involucrada. Por tanto, el cálculo de la fila Total en la Figura 9.2 no se calcula sumando las otras filas del informe. Este se calcula agregando todas las filas de la tabla Ventas, aunque esto significa que ya se calcularon otras iteraciones para las otras filas en el mismo informe. Por consiguiente, dependiendo de la expresión DAX, el resultado en la fila Total podría mostrar un resultado diferente, no relacionado con las otras filas en el mismo informe.



Para los ejemplos mostrados, estamos utilizando una matriz por simplicidad. También es posible definir un contexto de evaluación en otros tipos de visualización como gráficos, sin embargo, utilizamos matrices para mantener una comprensión visual simplificada de los conceptos

Ahora bien, en los ejemplos anteriores tenemos el contexto de filtro por producto para cada celda, pero podemos aumentar la complejidad de la matriz agregando otras variables, adicionemos ahora en columnas, el año de la tabla Calendario de nuestro modelo ejemplo para obtener el resultado de la Figura 9.7

Nombre Producto	2017	2018	2019	2020	Total
BIOEXPERT CHAMPU CAP.GUSANO SEDAx18x15ml	\$515.538.576	\$401.234.029	\$1.939.338.301	\$1.400.372.275	\$4.256.483.181
CALDO GALLINA x 1.8kgr x 165 CUBOS	\$51.966.300	\$63.470.776	\$54.730.785	\$159.594.338	\$329.762.199
CALDO GALLINA x 2.97 kgr X 270 CUBOS	\$862.119.874	\$649.583.653	\$510.634.813	\$670.362.954	\$2.692.701.294
CALDO RICOSTILLA CUBOS x 60x660gr	\$207.267.365	\$247.877.867	\$4.099.822.979	\$2.704.767.365	\$7.259.735.576
CORN FLAKES CEREAL INTEGRAL x 800 gr	\$1.512.330.993	\$974.772.069	\$1.037.117.955	\$1.024.278.177	\$4.548.499.194
EGO GEL EXTREME MAX MEGA x15 x27 ml.	\$509.096.689	\$551.845.367	\$601.272.245	\$618.092.266	\$2.280.306.567
ELITE SERVILLETA PRACTICA x 320	\$12.028.006	\$177.381.245	\$415.293.207	\$310.023.167	\$914.725.625
FRUTTSI FRUTOS ROJOS PET x 400 ml	\$639.671.769	\$542.617.671	\$448.950.371	\$546.853.113	\$2.178.092.924
INSTACREM ORIGINAL LOTERO SOBRE x18x3gr	\$416.529.633	\$446.874.840	\$3.185.848.805	\$2.645.260.194	\$6.694.513.472
MAGGI C.GALLINA CUBOS x48 x 528 gr	\$349.819.285	\$777.866.087	\$1.070.140.797	\$1.005.413.617	\$3.203.239.786
MILO DOY PACK x 500gr EXTRACONT 10%	\$91.476.177	\$79.260.944	\$99.060.722	\$190.889.623	\$460.687.466
MILO LATA x 400gr	\$731.459.066	\$549.235.208	\$361.248.920	\$560.519.535	\$2.202.462.729
MILO RISTRA x 16 UNIDS. x 25 gr.	\$1.000.160.635	\$977.591.288	\$884.868.617	\$843.626.374	\$3.706.246.914
MILO RISTRA x 18 SOBRES x 25 gr.	\$261.018.437	\$198.636.775	\$243.393.127	\$289.544.163	\$992.592.502
MILO TETRA PACK x 180 ml	\$414.413.627	\$389.708.848	\$301.415.139	\$267.261.298	\$1.372.798.912
NACHOS X 180 GRS. FRESCAMPO	\$211.197.319	\$184.504.269	\$193.049.063	\$197.879.091	\$786.629.742
NESCAFE TRADIC.x135 SOBRE GRTIS 9BOTADER	\$215.729.760	\$251.847.945	\$341.585.468	\$295.822.985	\$1.104.986.158
NESCAFE TRADICION FRASCO x170gr	\$144.421.227	\$180.488.234	\$171.640.949	\$164.710.225	\$661.260.635
Total	\$21.662.728.028	\$24.484.312.974	\$24.684.311.343	\$25.946.176.370	\$96.777.528.715

Figura 9.7 El total de ventas ahora se ve filtrado por Producto y Año.

En esta figura podemos observar ahora que cada celda posee un subconjunto de datos compuesto por un producto y un año, esto significa que ahora el contexto de filtro de cada una de las celdas ahora está filtrando por producto y por año, siendo más detallistas podemos observar también que para la fila Total, el filtro sólo está dado por Producto, mientras que en la columna Total el filtro está dado por el año, es decir, para la fila Total los valores sumados son los de cada producto de arriba hacia abajo, mientras que para la columna total, los valores sumados son los de cada año de izquierda a derecha. Y por último el gran Total es la única celda que calcula la suma de todas las ventas porque allí, el contexto de filtro no aplica ningún filtro.

Nombre Producto	2017	2018	2019	2020	Total
BIOEXPERT CHAMPU CAP.GUSANO SEDAx18x15ml	\$515.538.576	\$401.234.029	\$1.939.338.301	\$1.400.372.275	\$4.256.483.181
CALDO GALLINA x 1.8kgr x 165 CUBOS	\$51.966.300	\$63.470.776	\$54.730.785	\$159.594.338	\$329.762.199
CALDO GALLINA x 2.97 kgr X 270 CUBOS	\$862.119.874	\$649.583.653	\$510.634.813	\$670.362.954	\$2.692.701.294
CALDO RICOSTILLA CUBOS x 60x660gr	\$207.267.365	\$247.877.867	\$4.099.822.979	\$2.704.767.365	\$7.259.735.576
CORN FLAKES CEREAL INTEGRAL x 800 gr	\$1.512.330.993	\$974.772.069	\$1.037.117.955	\$1.024.278.177	\$4.548.499.194
EGO GEL EXTREME MAX MEGA x15 x27 ml.	\$509.096.689	\$551.845.367	\$601.272.245	\$618.092.266	\$2.280.306.567
ELITE SERVILLETA PRACTICA x 320	\$12.028.006	\$177.381.245	\$415.293.207	\$310.023.167	\$914.725.625
FRUTTSI FRUTOS ROJOS PET x 400 ml	\$639.671.769	\$542.617.671	\$448.950.371	\$546.853.113	\$2.178.092.924
INSTACREM ORIGINAL LOTERO SOBRE x18x3gr	\$416.529.633	\$446.874.840	\$3.185.848.805	\$2.645.260.194	\$6.694.513.472
MAGGI C.GALLINA CUBOS x48 x 528 gr	\$349.819.285	\$777.866.087	\$1.070.140.797	\$1.005.413.617	\$3.203.239.786
MILO DOY PACK x 500gr EXTRACONT 10%	\$91.476.177	\$79.260.944	\$99.060.722	\$190.889.623	\$460.687.466
MILO LATA x 400gr	\$731.459.066	\$549.235.208	\$361.248.920	\$560.519.535	\$2.202.462.729
MILO RISTRA x 16 UNIDS. x 25 gr.	\$1.000.160.635	\$977.591.288	\$884.868.617	\$843.626.374	\$3.706.246.914
MILO RISTRA x 18 SOBRES x 25 gr.	\$261.018.437	\$198.636.775	\$243.393.127	\$289.544.163	\$992.592.502
MILO TETRA PACK x 180 ml	\$414.413.627	\$389.708.848	\$301.415.139	\$267.261.298	\$1.372.798.912
NACHOS X 180 GRS. FRESCAMPO	\$211.197.319	\$184.504.269	\$193.049.063	\$197.879.091	\$786.629.742
NESCAFE TRADIC.x135 SOBRE GRTIS 9BOTADER	\$215.729.760	\$251.847.945	\$341.585.468	\$295.822.985	\$1.104.986.158
NESCAFE TRADICION FRASCO x170gr	\$144.421.227	\$180.488.234	\$171.640.949	\$164.710.225	\$661.260.635
Total	\$21.662.728.028	\$24.484.312.974	\$24.684.311.343	\$25.946.176.370	\$96.777.528.715

Figura 9.8 Contexto de filtro en los totales

Llegados a este punto ya podríamos inferir entonces que: cuantas más variables usemos como filas o columnas para cruzar, más filas y columnas serán filtradas por el contexto de filtro en cada celda de la matriz. Agreguemos entonces la columna Nombre Vendedor de la tabla Vendedores, tendremos un resultado diferente como se muestra en la Figura 9.9.

Nombre Producto	2017	2018	2019	2020	Total
■ BIOEXPERT CHAMPU CAP.GUSANO SEDAx18x15ml	\$515.538.576	\$401.234.029	\$1.939.338.301	\$1.400.372.275	\$4.256.483.181
ALBERT VELASQUEZ	\$181.038.714	\$189.775.750	\$181.570.666	\$192.918.957	\$745.304.087
JUAN GUILLERMO RICAURTE	\$141.382.897	\$75.422.538	\$1.631.559.393	\$1.035.219.493	\$2.883.584.321
MIGUEL ANGEL RIOS	\$191.827.265	\$134.746.938	\$124.993.767	\$171.553.719	\$623.121.689
SIMON TERRANOVA	\$1.289.700	\$1.288.803	\$1.214.475	\$680.106	\$4.473.084
■ CALDO GALLINA x 1.8kgr x 165 CUBOS	\$51.966.300	\$63.470.776	\$54.730.785	\$159.594.338	\$329.762.199
ALBERT VELASQUEZ	\$6.831.970	\$4.044.063	\$2.003.815	\$40.771.776	\$53.651.624
JUAN GUILLERMO RICAURTE	\$10.009.448	\$42.373.317	\$41.195.728	\$56.661.161	\$150.239.654
MIGUEL ANGEL RIOS	\$22.771.756	\$8.180.994	\$8.889.652	\$37.029.649	\$76.872.051
SIMON TERRANOVA		\$166.880		\$291.474	\$458.354
VANESSA QUINTERO	\$12.353.126	\$8.705.522	\$2.641.590	\$24.840.278	\$48.540.516
■ CALDO GALLINA x 2.97 kgr X 270 CUBOS	\$862.119.874	\$649.583.653	\$510.634.813	\$670.362.954	\$2.692.701.294
ALBERT VELASQUEZ	\$76.963.640	\$56.348.402	\$59.081.927	\$87.591.909	\$279.985.878
JUAN GUILLERMO RICAURTE	\$324.033.982	\$281.615.012	\$177.330.604	\$222.842.342	\$1.005.821.940
MIGUEL ANGEL RIOS	\$461.122.252	\$311.620.239	\$274.222.282	\$359.636.671	\$1.406.601.444
SIMON TERRANOVA				\$292.032	\$292.032
■ CALDO RICOSTILLA CUBOS x 60x660gr	\$207.267.365	\$247.877.867	\$4.099.822.979	\$2.704.767.365	\$7.259.735.576
ALBERT VELASQUEZ	\$1.273.176	\$8.795.482	\$3.060.476	\$8.191.274	\$21.320.408
JUAN GUILLERMO RICAURTE	\$46.009.224	\$41.587.774	\$3.983.118.727	\$2.533.286.005	\$6.604.001.730
MIGUEL ANGEL RIOS	\$495.750			\$16.983.782	\$17.479.532
SIMON TERRANOVA	\$358.461	\$1.575.245	\$8.911.386	\$11.026.715	\$21.871.807
VANESSA QUINTERO	\$159.130.754	\$195.919.366	\$104.732.390	\$135.279.589	\$595.062.099
■ CORN FLAKES CEREAL INTEGRAL x 800 gr	\$1.512.330.993	\$974.772.069	\$1.037.117.955	\$1.024.278.177	\$4.548.499.194
ALBERT VELASQUEZ	\$721.877.758	\$619.865.450	\$503.003.788	\$485.908.069	\$2.330.655.065
JUAN GUILLERMO RICAURTE	\$114.342.780	\$140.140.778	\$199.176.887	\$160.217.340	\$613.877.785
MIGUEL ANGEL RIOS	\$675.984.542	\$207.813.356	\$313.025.400	\$366.797.868	\$1.563.621.166
SIMON TERRANOVA		\$4.990.260	\$6.351.240	\$10.585.400	\$21.926.900
VANESSA QUINTERO	\$125.913	\$1.962.225	\$15.560.640	\$769.500	\$18.418.278
Total	\$21.662.728.028	\$24.484.312.974	\$24.684.311.343	\$25.946.176.370	\$96.777.528.715

Figura 9.9 El Contexto de filtro está definido por el conjunto de campos en las filas y columnas

Vemos aquí que ahora el contexto de filtro de cada celda está definido por producto, vendedor y año. Dicho de otra manera, el contexto de filtro contiene el conjunto completo de campos que se utilizan como filas y columnas dentro de la matriz.



Nota: Debe tener en cuenta que cualquier campo que esté en las filas o columnas del elemento visual, así como también en un filtro de segmentación de datos, en un filtro de página, de informe, en un filtro visual o en cualquier otro tipo de filtro que podamos crear en el informe, es indiferente. Todos los filtros contribuyen a definir el contexto de filtro único, que DAX utiliza para evaluar las fórmulas. No importa si el campo se visualiza en filas o columnas, en un gráfico de barras o en un gráfico de líneas, esto no cambia la forma en la que DAX calcula los valores.

Las interacciones visuales en Power BI también hacen parte de la composición del contexto de filtro, esto da como resultado una combinación bastante amplia de diferentes elementos que interactúan no sólo gráficamente sino también internamente por medio del contexto de filtro. Por lo tanto, el contexto de filtro de una celda estará calculado por la fusión de todos los filtros que provienen del informe, sean filas, columnas, segmentaciones, barras de un gráfico o cualquier otro elemento visual utilizado dentro de Power BI para filtrar los datos. Para verlo gráficamente observe la Figura 9.10.

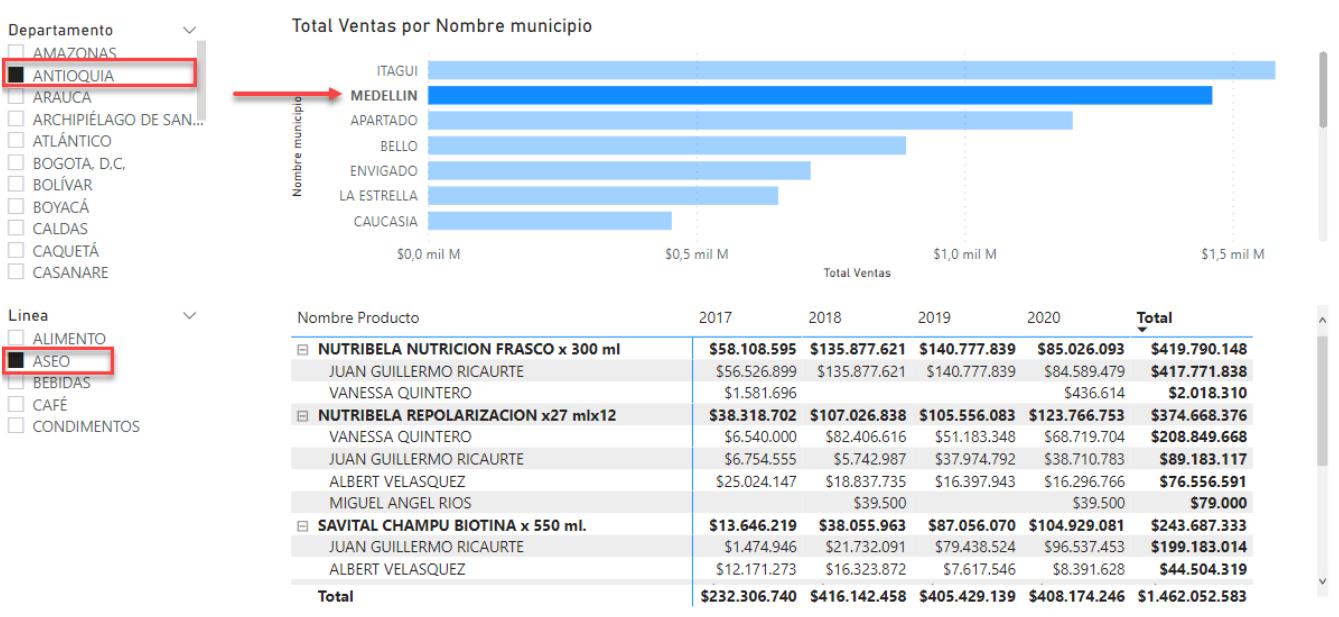


Figura 9.10 El Contexto de filtro está definido por las segmentaciones, los filtros y las gráficas

El contexto de filtro de las celdas de la matriz, en este ejemplo no estaría dado sólo por los elementos que ya describimos en las filas y las columnas, sino que también está definido por el departamento “Antioquia” y la línea “Aseo”, que están llegando de las dos segmentaciones de filtro en la página, adicionalmente también se está filtrando el municipio “Medellín” que se seleccionó en la gráfica de barras, todo este conjunto de filtros están contribuyendo a la definición de un contexto de filtro único y válido para una celda, que DAX está aplicando a todo el modelo de datos antes de evaluar la fórmula.

Antes de finalizar el concepto de contexto de filtro analicemos nuevamente la fórmula de la métrica “Total Ventas” que definimos al inicio:



$$\text{Total Ventas} = \text{SUMX}(\text{Ventas}, \text{Ventas[Cantidad]} * \text{Ventas[Precio Unit]})$$

La manera correcta de leer esta métrica sería: La suma de la cantidad multiplicada por el precio unitario para todas las filas de la tabla Ventas que son visibles en el contexto del filtro actual.

Esto mismo aplica para las funciones de agregación más simples, por ejemplo, miremos esta métrica:



Total Cantidad = `SUM(Ventas[Cantidad])`

Esta métrica suma la columna cantidad de todas las filas en la tabla Ventas que están visibles dentro del contexto del filtro actual. Podemos comprender mejor su funcionamiento considerando la versión correspondiente de SUMX:



Total Cantidad = `SUMX (Ventas, Ventas[Cantidad])`

Si observamos la definición dentro de SUMX, podríamos decir que el contexto del filtro afectará la evaluación del contexto de la expresión en la tabla de Ventas, puesto que estamos definiendo que Ventas será la tabla para iterar con esta expresión, es decir, que solo devolverá las filas de la tabla Ventas que estén visibles en el contexto del filtro actual. Sin embargo, debemos considerar que este contexto de filtro también se aplica a métricas que no tienen definido un iterador correspondiente como las siguientes:



Cantidad Vendedores = `DISTINCTCOUNT (Ventas[Cod. Vendedor])`

Esta métrica estaría contando los vendedores dentro del contexto de filtro



```
Productos =
VAR ListaProductos = DISTINCT('Maestro Productos'[Nombre Producto])
RETURN
COUNTRROWS(ListaProductos)
```

Esta, por otro lado, cuenta los nombres únicos de los productos dentro del contexto de filtro y los almacena en una variable y posteriormente retorna el conteo de productos diferentes.

En estos ejemplos anteriores, también aplica el contexto de filtro a estas métricas a pesar de no tener un iterador definido, por lo tanto, nuevamente estamos diciendo que el contexto de filtro siempre está activo y afectará el resultado de la fórmula, básicamente, es lo que hemos repetido en cada uno de los ejemplos de este concepto de diferentes formas, y la razón de tanto énfasis en esto es porque DAX requiere que seamos extremadamente precisos. Lo complejo de DAX no está tanto en aprender nuevas funciones, sino más bien, en la presencia de muchos conceptos sutiles, cuando todos estos conceptos se mezclan, lo que observamos es un escenario complejo.

Comprendiendo los contextos de fila

En esta sección aprenderemos el segundo contexto de evaluación: el contexto de fila. Recordemos que a pesar de que el contexto de filtro y el contexto de fila son contextos de evaluación, son conceptos completamente distintos. Como vimos en el tema anterior, el propósito del contexto de filtro es el de filtrar tablas. Por otro lado, como veremos en este tema el contexto de fila no es una herramienta para filtrar tablas. Es un contexto que se utiliza para iterar sobre tablas u evaluar valores de columna.

Ahora utilizaremos una fórmula diferente para analizar las características del contexto de fila, definiremos una columna calculada en la tabla de Ventas para computar el margen bruto:



Margen bruto = `Ventas[Cantidad] * (Ventas[Precio Unit] - Ventas[CostoUnit])`

Tendremos entonces un valor diferente como resultado para cada fila en la nueva columna calculada, como se muestra en la figura 9.11

Cantidad	CostoUnit	Precio Unit	Margen bruto
1	53.634	82.154	28.520
1	57.633	95.100	37.467
1	62.986	100.704	37.718
1	62.986	116.106	53.120
1	67.829	180.000	112.171
1	70.255	110.730	40.475
1	73.866	110.730	36.864
1	75.614	110.730	35.116
1	82.274	157.850	75.576
1	87.457	282.624	195.167
1	88.584	157.850	69.266
1	115.041	220.578	105.537
2	400	619	438
2	418	699	562
2	421	699	556
2	429	699	540
2	429	754	650
2	442	608	332

Figura 9.11 Se observa un valor diferente para cada fila de Margen bruto como resultado de la operación con las demás variables.

Debido a que hay valores dispuestos en cada fila para las tres columnas utilizadas en la expresión, es una consecuencia natural que la expresión final calcule diferentes valores. Como sucede con el contexto de filtro, la razón es la presencia de un contexto de evaluación, sin embargo, esta vez el contexto no filtra la tabla, sino que identifica cada fila para que se realice el cálculo correspondiente.



Advertencia: El contexto de fila hace referencia a una fila en el resultado de una expresión de tabla en DAX. No confundir con una fila en el informe, es decir, con una fila en un componente visual como una matriz. Los valores que se muestran en una matriz en Power BI y en una tabla dinámica en Excel son el resultado de métricas DAX calculadas en un contexto de filtro, o son valores almacenados en la tabla como columnas nativas o calculadas.

Dicho de otro modo, sabemos que una columna calculada se computa fila por fila, pero ¿cómo DAX logra saber qué fila está iterando actualmente? Conoce la fila porque hay otro contexto de evaluación que proporciona la fila, y este es el contexto de fila del que estamos hablando. Cuando creamos una columna calculada sobre una tabla con una cantidad X de filas, DAX crea un contexto de fila que evalúa la expresión iterando sobre la tabla fila por fila, utilizando el contexto de fila como el punto de guía para la iteración. A su vez, cuando se crea una columna calculada, DAX crea un contexto de fila por defecto. En ese caso, no sería necesario crear un contexto de fila manualmente, iniciando una iteración. De hecho, uno puede escribir el margen bruto como una métrica, como en el siguiente código:



```
Margen bruto =  
SUMX ( Ventas, Ventas[Cantidad] * ( Ventas[Precio Unit] - Ventas[CostoUnit] ) )
```

En este caso, dado que este código es para una métrica, no hay contexto de fila automático. SUMX, al ser un iterador, crea un contexto de fila que comienza a iterar sobre la tabla Ventas, fila por fila. Durante cada iteración, se ejecuta la segunda expresión de SUMX dentro del contexto de fila. De esta manera, durante cada paso de dicha iteración, DAX sabe qué valor usar para cada una de las tres columnas que se utilizaron para el cálculo en la expresión.

El contexto de fila se establece cuando se crea una columna calculada o cuando se calcula una expresión dentro de una iteración. En DAX no existe otra forma de crear un contexto de fila. Adicionalmente, puede ayudarnos a comprenderlo mejor, pensar que se necesita un contexto de fila cada vez que queramos obtener el valor de una columna para una fila determinada. Por ejemplo, veamos la siguiente definición de la métrica Margen bruto:



```
Margen bruto =  
Ventas[Cantidad] * ( Ventas[Precio Unit] - Ventas[CostoUnit] )
```

Esta definición no es válida, ya que le estamos pidiendo a DAX que calcule el valor de Ventas [Precio Unit] y no hay ningún contexto de fila que le proporcione a DAX la fila para la que debe ejecutarse el cálculo, no obstante, esta expresión si es válida cuando se ejecuta como una columna calculada. La razón no es porque las métricas y las columnas calculadas tengan diferentes formas de usar DAX. La razón se encuentra en que una columna calculada tiene un contexto de fila automático, que está dado por la tabla en la que se crea dicha columna, mientras que una métrica no. Si se quiere evaluar una expresión fila por fila dentro de una métrica, es necesario comenzar una iteración para crear el contexto de fila.



Nota: Una referencia de columna requiere un contexto de fila para devolver el valor de la columna de una tabla. Una referencia de columna también se puede utilizar como argumento para varias funciones DAX sin un contexto de fila. Como, por ejemplo, DISTINCT y DISTINCTCOUNT pueden tener una referencia de columna como parámetro, sin definir un contexto de fila. Sin embargo, una referencia de columna en una expresión DAX requiere que se evalúe un contexto de fila.

Llegados a este punto, es necesario aclarar nuevamente que un contexto de fila no es un tipo especial o una variación del contexto de filtro que filtra una fila. El contexto de fila no está filtrando el modelo de ninguna forma, simplemente está indicando a DAX qué fila usar fuera de una tabla. Si se quiere aplicar un filtro al modelo, la herramienta a utilizar es el contexto de filtro. Por otro lado, si el usuario desea evaluar una expresión fila por fila, entonces el contexto de fila hará este trabajo.

Veamos otro ejemplo de contexto de fila directamente en la tabla Calendario de nuestro modelo, crearemos la columna calculada “Día Año” con la siguiente fórmula:



Día Año =
DATEDIFF (DATE (YEAR ('Calendario'[Date]), 1, 1), 'Calendario'[Date], DAY) + 1

Hecho esto, tenemos entonces la tabla Calendario con una columna adicional que corresponde al día del año de acuerdo con la fecha de cada fila en la columna Date

Date	Año	Mes Num	Mes	Trimestre	Semestre	Semana	Día Mes	Día Semana	Nombre Día	Día Año
24/01/2017 12:00:00 a. m.	2017	1	Enero	T1	Sem 1	5	24	2	martes	24
25/01/2017 12:00:00 a. m.	2017	1	Enero	T1	Sem 1	5	25	3	miércoles	25
26/01/2017 12:00:00 a. m.	2017	1	Enero	T1	Sem 1	5	26	4	jueves	26
27/01/2017 12:00:00 a. m.	2017	1	Enero	T1	Sem 1	5	27	5	viernes	27
28/01/2017 12:00:00 a. m.	2017	1	Enero	T1	Sem 1	5	28	6	sábado	28
29/01/2017 12:00:00 a. m.	2017	1	Enero	T1	Sem 1	5	29	7	domingo	29
30/01/2017 12:00:00 a. m.	2017	1	Enero	T1	Sem 1	6	30	1	lunes	30
31/01/2017 12:00:00 a. m.	2017	1	Enero	T1	Sem 1	6	31	2	martes	31
1/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	6	1	3	miércoles	32
2/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	6	2	4	jueves	33
3/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	6	3	5	viernes	34
4/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	6	4	6	sábado	35
5/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	6	5	7	domingo	36
6/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	7	6	1	lunes	37
7/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	7	7	2	martes	38
8/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	7	8	3	miércoles	39
9/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	7	9	4	jueves	40
10/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	7	10	5	viernes	41
11/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	7	11	6	sábado	42
12/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	7	12	7	domingo	43
13/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	8	13	1	lunes	44
14/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	8	14	2	martes	45
15/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	8	15	3	miércoles	46
16/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	8	16	4	jueves	47
17/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	8	17	5	viernes	48
18/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	8	18	6	sábado	49
19/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	8	19	7	domingo	50
20/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	9	20	1	lunes	51
21/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	9	21	2	martes	52
22/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	9	22	3	miércoles	53
23/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	9	23	4	jueves	54
24/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	9	24	5	viernes	55
25/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	9	25	6	sábado	56
26/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	9	26	7	domingo	57
27/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	10	27	1	lunes	58
28/02/2017 12:00:00 a. m.	2017	2	Febrero	T1	Sem 1	10	28	2	martes	59
1/03/2017 12:00:00 a. m.	2017	3	Marzo	T1	Sem 1	10	1	3	miércoles	60
2/03/2017 12:00:00 a. m.	2017	3	Marzo	T1	Sem 1	10	2	4	jueves	61

Figura 9.12 Fila actual como contexto de fila

En el campo día del año podemos ver un poco más claro el contexto de fila. Le indicamos que tome el valor de la columna Date y calcule el día del año con respecto a la fecha. De esta manera, para cada fila DAX considerará el campo Date y realizará el cálculo indicado en la expresión.

En este ejemplo hemos creado una columna calculada con un solo campo, pero como vimos anteriormente con la columna calculada Margen Bruto podemos utilizar varios campos de la misma tabla, o incluso de otra tabla que esté relacionada.

En este ejemplo, explicaremos el contexto de fila entre dos tablas relacionadas utilizando campos de la tabla Maestro productos.

Retomemos nuevamente la fórmula de la columna calculada Margen bruto, pero con una ligera modificación para usar los valores del campo “Costo Estándar” de la tabla de productos en vez de la de “Costo Unit” que se encuentra en la tabla Ventas, tendríamos entonces una fórmula como la siguiente:



Margen bruto modificado =
`Ventas[Cantidad] * (Ventas[Precio Unit] - RELATED('Maestro Productos'[Costo Estándar]))`

Como vemos utilizando la función RELATED vista en el capítulo 8, estamos calculando un nuevo margen bruto restándole esta vez el costo estándar que se encuentra en la tabla de productos. Entonces, en la tabla de ventas tendríamos una nueva columna con un resultado diferente como se ve en la Figura 9.13

Cod. Vendedor	Cod.Producto	Cantidad	Precio Unit	CostoUnit	Cod Lineas	Documento	Cod Municipio	Fecha Pedido	Fecha Entrega	Margen bruto	Margen bruto modificado
B-01	2303	20	25.320	16.254 AL-01	697376	25214	jueves, 25 de enero de 2018	domingo, 25 de marzo de 2018	181.320	202.100	
B-01	2303	48	25.320	16.270 AL-01	1978167	25214	jueves, 15 de febrero de 2018	miércoles, 14 de marzo de 2018	434.400	485.040	
B-01	2303	140	25.320	16.275 AL-01	7078105	25214	jueves, 8 de marzo de 2018	martes, 10 de abril de 2018	1.266.300	1.414.700	
B-01	2303	400	25.320	16.178 AL-01	0375155	25214	miércoles, 13 de febrero de 2018	sábado, 24 de febrero de 2018	3.656.800	4.042.000	
B-01	2303	100	25.320	16.147 AL-01	6674108	25214	miércoles, 20 de febrero de 2018	viernes, 25 de febrero de 2018	917.300	1.010.500	
B-01	2303	140	25.320	18.659 AL-01	4692192	25214	jueves, 14 de mayo de 2018	domingo, 5 de agosto de 2018	832.540	1.414.700	
B-01	2303	20	25.320	18.651 AL-01	5093150	25214	miércoles, 26 de junio de 2018	jueves, 5 de agosto de 2018	133.180	202.100	
B-01	2303	120	25.320	18.434 AL-01	0894288	25214	miércoles, 5 de junio de 2018	jueves, 16 de agosto de 2018	826.320	1.212.600	
B-01	2303	120	25.320	18.582 AL-01	7191311	25214	jueves, 31 de mayo de 2018	viernes, 20 de julio de 2018	808.560	1.212.600	
B-01	2303	48	25.320	18.505 AL-01	6690194	25214	jueves, 14 de junio de 2018	viernes, 6 de julio de 2018	328.928	485.040	
B-01	2303	48	25.320	18.592 AL-01	0291131	25214	jueves, 23 de abril de 2018	jueves, 5 de julio de 2018	822.944	485.040	
B-01	2303	60	25.320	16.107 AL-01	0188101	25214	viernes, 20 de enero de 2018	jueves, 28 de mayo de 2018	552.780	606.300	
B-01	2303	24	25.320	16.311 AL-01	4487244	25214	miércoles, 15 de mayo de 2018	jueves, 4 de junio de 2018	216.216	242.520	
B-01	2303	50	25.320	16.416 AL-01	1487326	25214	sábado, 12 de mayo de 2018	domingo, 20 de mayo de 2018	445.200	505.250	
B-01	2303	36	25.320	16.928 AL-01	9788262	25214	viernes, 4 de mayo de 2018	domingo, 10 de junio de 2018	302.112	363.780	
B-01	2303	90	25.320	16.293 AL-01	7788156	25214	sábado, 7 de abril de 2018	jueves, 21 de junio de 2018	812.430	909.450	
B-01	2303	100	25.320	18.491 AL-01	3890295	25214	sábado, 21 de abril de 2018	jueves, 28 de junio de 2018	682.900	1.010.500	
B-01	2303	60	25.320	18.381 AL-01	9389399	25214	miércoles, 18 de abril de 2018	jueves, 2 de julio de 2018	416.340	606.300	
B-01	2303	24	25.320	18.294 AL-01	5989281	25214	miércoles, 6 de junio de 2018	jueves, 11 de junio de 2018	168.624	242.520	
B-01	2303	50	25.320	16.280 AL-01	1882340	25214	domingo, 25 de marzo de 2018	miércoles, 1 de mayo de 2018	452.000	505.250	
B-01	2303	50	25.320	16.280 AL-01	6681205	25214	miércoles, 27 de marzo de 2018	domingo, 22 de abril de 2018	452.000	505.250	
B-01	2303	100	25.320	16.279 AL-01	8380288	25214	jueves, 23 de marzo de 2018	jueves, 5 de abril de 2018	904.100	1.010.500	
B-01	2303	130	25.320	16.279 AL-01	1680245	25214	miércoles, 31 de enero de 2018	miércoles, 18 de abril de 2018	1.175.330	1.313.650	
B-01	2303	35	25.320	16.277 AL-01	2579110	25214	domingo, 18 de febrero de 2018	miércoles, 18 de abril de 2018	316.505	353.675	
B-01	2303	20	25.320	16.369 AL-01	7684160	25214	miércoles, 4 de abril de 2018	domingo, 20 de mayo de 2018	179.020	202.100	
B-01	2303	100	25.320	16.426 AL-01	6185356	25214	jueves, 3 de mayo de 2018	jueves, 17 de mayo de 2018	889.400	1.010.500	
B-01	2303	80	25.320	16.530 AL-01	3886239	25214	viernes, 23 de marzo de 2018	viernes, 8 de junio de 2018	703.200	808.400	
B-01	2303	6	25.320	16.429 AL-01	0386562	25214	miércoles, 13 de marzo de 2018	jueves, 31 de mayo de 2018	53.346	60.630	
B-01	2303	80	25.320	16.281 AL-01	6883349	25214	miércoles, 21 de marzo de 2018	jueves, 24 de mayo de 2018	723.120	808.400	
B-01	2303	100	25.320	16.281 AL-01	9282397	25214	viernes, 2 de marzo de 2018	jueves, 19 de abril de 2018	903.900	1.010.500	
B-01	2303	100	25.320	16.281 AL-01	3384385	25214	jueves, 29 de marzo de 2018	domingo, 6 de mayo de 2018	903.900	1.010.500	
B-01	2303	150	25.320	18.147 AL-01	6998102	25214	jueves, 25 de junio de 2018	jueves, 30 de agosto de 2018	1.075.950	1.515.750	
B-01	2303	180	25.320	18.182 AL-01	8210351	25214	miércoles, 17 de julio de 2018	jueves, 8 de octubre de 2018	1.284.840	1.818.900	
B-01	2303	30	25.320	18.246 AL-01	7210362	25214	sábado, 18 de agosto de 2018	viernes, 28 de septiembre de 2018	212.220	303.150	

Figura 9.13 Se observa un valor diferente Margen bruto modificado resultado de la operación con campos de otra tabla

Reglas de evaluación

Cuando se evalúa una expresión DAX hay una serie de reglas que se ejecutan en orden. Forman la base de cómo se usarán las funciones DAX.

Las reglas para la evaluación de una expresión DAX son:

- Evaluación del contexto de filtro Inicial/Entrante de las segmentaciones de datos.
- Campos de un objeto visual (por ejemplo, filas y columnas de una matriz)
- Filtros de página o informe
- Filtros de otro componente visual
- Si se utiliza la función CALCULATE, se verifica la modificación del filtro
- Comprobación del filtro aplicado a la(s) tabla(s) subyacente(s)
- Interpretar la propagación del filtro por relación(es)
- Evaluación del resultado

Hay muchas excepciones y variaciones a estas reglas, sin embargo, es una forma de verlo de manera más simple y genérica. Esto le permitirá comprender inicialmente los fundamentos más importantes sin entrar en detalles muy complejos.

Comprendiendo la transición de contexto

En varias ocasiones hemos comentado que el contexto de fila y el contexto de filtro son conceptos diferentes. Esto seguirá siendo así. Sin embargo, hay una operación realizada por la función CALCULATE que puede transformar un contexto de fila en un contexto de filtro. Es la operación de transición de contexto, esta sería la definición dada por Microsoft en su documentación:

” CALCULATE invalida cualquier contexto de fila. Agrega automáticamente como argumentos de filtro todas las columnas que se están iterando actualmente en cualquier contexto de fila. Filtrando su valor real en la fila que se está iterando.

Antes de pasar al detalle hagamos una breve comparación entre ambos contextos:

Contexto de filtro	Contexto de fila
Proporcionado por las coordenadas de un objeto visual, por ejemplo, una matriz.	Existe en las columnas calculadas o las funciones iterativas de DAX
La función CALCULATE se puede utilizar para modificar un contexto de filtro	La función CALCULATE puede convertir un contexto de fila en un contexto de filtro
Automáticamente sigue las relaciones de 1 a varios (1:*) desde el lado 1 hacia el lado varios.	No sigue relaciones automáticamente ni crea contexto de filtro.

Figura 9.14 Se muestra la misma cantidad de productos vendidos para todas las filas

Para dar una explicación sencilla de la transición de contexto vamos a crear una columna calculada en la tabla Maestro productos que muestre la cantidad vendida para cada producto:



Cantidad de productos vendidos =
`SUM(Ventas[Cantidad])`

Con esta fórmula encontraríamos la cantidad total vendida para cada uno de los productos de la tabla Maestro productos, sin embargo, fijémonos en el resultado arrojado para cada fila en la tabla de productos de la figura 9.14.

Cod Producto	Nombre Producto	Costo Estándar	Cantidad de productos vendidos
8928	CORN FLAKES CEREAL INTEGRAL x 800 gr	20.110	15551537
1738	NESTUM 5 CEREALES x 200 gr	19.096	15551537
2303	QUIPITOS SOBRE x 24 x 8gr	15.215	15551537
9180	TOSTADAS X 145 GR	1.247	15551537
1986	POPETAS CARAMELO-QUESO FAMILIAR x 135 gr	556	15551537
2204	NACHOS X 180 GRS. FRESCAMPO	1.711	15551537
7902	NUTRIBELA NUTRICION FRASCO x 300 ml	11.973	15551537
3607	BIOEXPERT CHAMPU CAP.GUSANO SEDAx18x15ml	31.473	15551537
7018	NUTRIBELA REPOLARIZACION x27 mlx12	24.986	15551537
2129	SAVITAL CHAMPU BIOTINA x 550 ml.	1.350	15551537

Figura 9.15 Resultado incorrecto

El resultado no es correcto porque no se vendió la misma cantidad para cada uno de los productos, realmente lo que DAX nos está mostrando aquí es la cantidad total vendida para todos los productos, ya que al estar la fórmula en una columna calculada nos encontramos en un contexto de fila con respecto a la tabla de Ventas. Sin embargo, CALCULATE nos permitirá obtener el resultado deseado, ya que, CALCULATE convierte el contexto de fila en contexto de filtro. Esto se llama transición de contexto.



Nota: CALCULATE se analizará con más detalle en el capítulo 10 CALCULATE y funciones de filtro.

La transición de contexto en este escenario lo que nos está permitiendo es que cada producto filtre de manera individual hacia la tabla Ventas, ya que existe una relación entre ambas tablas, sin embargo, sin usar la función CALCULATE nos encontramos en un contexto de fila, por esta razón, no se están filtrando los valores de manera correcta.

Lo que haremos entonces con la fórmula es agregar la función CALCULATE, resultando de la siguiente manera:



Cantidad de productos vendidos =
CALCULATE(SUM(Ventas[Cantidad]))

El contexto de filtro de la métrica Cantidad de productos vendidos queda entonces modificado por la función CALCULATE y de esta manera, obtenemos el siguiente resultado para cada fila en la tabla de Maestro productos como se ve en la Figura 9.16.

Cod Producto	Nombre Producto	Costo Estándar	Cantidad de productos vendidos
8928	CORN FLAKES CEREAL INTEGRAL x 800 gr	20.110	181698
1738	NESTUM 5 CEREALES x 200 gr	19.096	86193
2303	QUIPITOS SOBRE x 24 x 8gr	15.215	218820
9180	TOSTADAS X 145 GR	1.247	866776
1986	POPETAS CARAMELO-QUESO FAMILIAR x 135 gr	556	8082613
2204	NACHOS X 180 GR5. FRESCAMPO	1.711	214156
7902	NUTRIBELA NUTRICION FRASCO x 300 ml	11.973	278810
3607	BIOEXPERT CHAMPU CAP.GUSANO SEDAx18x15ml	31.473	89801
7018	NUTRIBELA REPOLARIZACION x27 mlx12	24.986	51184
2129	SAVITAL CHAMPU BIOTINA x 550 ml.	1.350	557429
2873	EGO GEL EXTREME MAX MEGA x15 x27 ml.	8.145	165600
4086	ELITE SERVILLETA PRACTICA x 320	59.703	5548
2438	MILO TETRA PACK x 180 ml	2.702	311045
1677	MILO RISTRAS x 16 UNIDS. x 25 gr.	30.722	68858
1317	MILO LATA x 400gr	10.809	130588
1964	MILO RISTRAS x 18 SOBRES x 25 gr.	3.842	137268
1250	MILO DOY PACK x 500gr EXTRACONT 10%	5.584	46215
3512	PULPIFRUTA MORA 12 x160gr.	2.800	66152
6202	SUNTEA DURAZNO x 4.5 LITROS x 266 gr.	20.668	301618
6905	SAVILOE AGUA DE ALOE x 270 ml	34.023	241319
4307	FRUTTSI FRUTOS ROJOS PET x 400 ml	28.830	45983
5731	SUNTEA LIMON x9 LITROS x 532 gr	2.133	340572
7228	NESCAFE TRADIC.x135 SOBRE GRTIS 9BOTADER	13.211	60147
9376	NESCAFE TRADICION x12 sobres x25gr	7.110	126760
5617	NESCAFE TRADICION FRASCO x170gr	359	796111
6264	NESCAFE TRADICION FRASCO x50 gr.	22.723	32504
1246	NESCAFE TRADICION x12 sobres x10gr.	1.446	175442
2950	NESCAFE TRADICION FRASCO x85gr	1.815	599877
8602	INSTACREM ORIGINAL LOTERO SOBRE x18x3gr	9.840	506161
1977	MAGGI C.GALLINA CUBOS x48 x 528 gr	12.094	166778
9658	CALDO GALLINA x 2.97 kgr X 270 CUBOS	16.931	104187
7599	CALDO GALLINA x 1.8kgr x 165 CUBOS	848	240806
7605	CALDO RICOSTILLA CUBOS x 60x660gr	20.762	254518

Figura 9.16 Se muestra la cantidad total vendida para cada uno de los productos.

Veamos otro ejemplo de transición de contexto trayendo a esta misma tabla de productos la fecha del último pedido de cada producto, para esto utilizaremos la siguiente fórmula:



Fecha último pedido=
`CALCULATE(MAX(Ventas[Fecha]))`

En esta expresión la función MAX encontrará la última fecha de pedido en la tabla de Ventas y la función CALCULATE hará la transición del contexto de fila hacia un contexto de filtro para que se pueda filtrar la fecha del último pedido para cada producto. Como resultado obtendremos la fecha del último de cada producto como se ve en la Figura 9.17.

Cod Producto	Nombre Producto	Costo Estándar	Cantidad de productos vendidos	Fecha último pedido
8928	CORN FLAKES CEREAL INTEGRAL x 800 gr	20.110	181698	22/12/2020 12:00:00 a. m.
1738	NESTUM 5 CEREALES x 200 gr	19.096	86193	25/12/2020 12:00:00 a. m.
2303	QUIPITOS SOBRE x 24 x 8gr	15.215	218820	10/12/2020 12:00:00 a. m.
9180	TOSTADAS X 145 GR	1.247	865776	25/12/2020 12:00:00 a. m.
1986	POPETAS CARAMELO-QUESO FAMILIAR x 135 gr	556	8082613	20/12/2020 12:00:00 a. m.
2204	NACHOS X 180 GRS. FRESCAMPO	1.711	214156	20/12/2020 12:00:00 a. m.
7902	NUTRIBELA NUTRICION FRASCO x 300 ml	11.973	278810	3/12/2020 12:00:00 a. m.
3607	BIOEXPERT CHAMPU CAP.GUSANO SEDAx18x15ml	31.473	89801	23/12/2020 12:00:00 a. m.
7018	NUTRIBELA REPOLARIZACION x27 mlx12	24.986	51184	15/12/2020 12:00:00 a. m.
2129	SAVITAL CHAMPU BIOTINA x 550 ml.	1.350	557429	14/12/2020 12:00:00 a. m.
2873	EGO GEL EXTREME MAX MEGA x15 x27 ml.	8.145	165600	20/12/2020 12:00:00 a. m.
4086	ELITE SERVILLETA PRACTICA x 320	59.703	5548	15/12/2020 12:00:00 a. m.
2438	MILO TETRA PACK x 180 ml	2.702	311045	13/12/2020 12:00:00 a. m.
1677	MILO RISTRAS x 16 UNIDS. x 25 gr.	30.722	68858	19/12/2020 12:00:00 a. m.
1317	MILO LATA x 400gr	10.809	130588	16/12/2020 12:00:00 a. m.
1964	MILO RISTRAS x 18 SOBRES x 25 gr.	3.842	137268	20/12/2020 12:00:00 a. m.
1250	MILO DOY PACK x 500gr EXTRACONT 10%	5.584	46215	9/12/2020 12:00:00 a. m.
3512	PULPIFRUTA MORA 12 x160gr.	2.800	66152	13/12/2020 12:00:00 a. m.
6202	SUNTEA DURAZNO x 4.5 LITROS x 266 gr.	20.668	301618	13/12/2020 12:00:00 a. m.
6905	SAVILOE AGUA DE ALOE x 270 ml	34.023	241319	26/12/2020 12:00:00 a. m.
4307	FRUTTSI FRUTOS ROJOS PET x 400 ml	28.830	45983	15/12/2020 12:00:00 a. m.
5731	SUNTEA LIMON x9 LITROS x 532 gr	2.133	340572	22/12/2020 12:00:00 a. m.
7228	NESCAFE TRADIC.x135 SOBRE GRTIS 9BOTADER	13.211	60147	17/12/2020 12:00:00 a. m.
9376	NESCAFE TRADICION x12 sobres x25gr	7.110	126760	16/12/2020 12:00:00 a. m.
5617	NESCAFE TRADICION FRASCO x170gr	359	796111	25/12/2020 12:00:00 a. m.
6264	NESCAFE TRADICION FRASCO x50 gr.	22.723	32504	22/12/2020 12:00:00 a. m.
1246	NESCAFE TRADICION x12 sobres x10gr.	1.446	175442	23/12/2020 12:00:00 a. m.
2950	NESCAFE TRADICION FRASCO x85gr	1.815	599877	30/11/2020 12:00:00 a. m.
8602	INSTACREM ORIGINAL LOTERO SOBRE x18x3gr	9.840	506161	23/12/2020 12:00:00 a. m.
1977	MAGGI C.GALLINA CUBOS x48 x 528 gr	12.094	166778	26/12/2020 12:00:00 a. m.
9658	CALDO GALLINA x 2.97 kgr X 270 CUBOS	16.931	104187	21/12/2020 12:00:00 a. m.
7599	CALDO GALLINA x 1.8kgr x 165 CUBOS	848	240806	15/12/2020 12:00:00 a. m.
7605	CALDO RICOSTILLA CUBOS x 60x660gr	20.762	254518	25/12/2020 12:00:00 a. m.

Figura 9.17 Se obtiene la fecha en la que se hizo el último pedido para cada producto.

Comprender la transición de contexto es muy importante debido a otro aspecto importante de DAX.

Cada referencia a una medida o métrica siempre tiene un CALCULATE implícito a su alrededor.

Debido a CALCULATE, una referencia de medida genera una transición de contexto implícita si se ejecuta en presencia de cualquier contexto de fila. Esta es la razón por la que en DAX es importante usar la convención de nomenclatura correcta al escribir referencias de columna (siempre incluyendo el nombre de la tabla) y referencias de medida (siempre sin el nombre de la tabla). Desea estar al tanto de cualquier transición de contexto implícita escribiendo y leyendo una expresión DAX.

Como conclusión tenemos entonces que la transición de contexto es un concepto importante en DAX. Este concepto se basa en el poder de la función CALCULATE. La transición de contexto se refiere a la conversión del contexto de fila en contexto de filtro usando CALCULATE.

Consideraciones

- La transición de contexto es clara. Si la transición de contexto se usa durante una iteración en una tabla con 10 columnas y un millón de filas, CALCULATE necesita aplicar 10 filtros, un millón de veces. No importa qué, será una operación lenta. Esto no quiere decir que deba evitarse depender de la transición de contexto. Sin embargo, hace que CALCULATE sea una característica que debe usarse con cuidado.
- La transición de contexto crea un contexto de filtro a partir de un contexto de fila. Quizás recuerde el mantra del contexto de evaluación, "el contexto de fila itera una tabla, mientras que el contexto de filtro filtra el modelo". Una vez que la transición de contexto transforma un contexto de fila en un contexto de filtro, cambia la naturaleza del filtro. En lugar de iterar una sola fila, DAX filtra todo el modelo; las relaciones se vuelven parte de la ecuación. En otras palabras, la transición de contexto que ocurre en una tabla podría propagar sus efectos de filtrado lejos de la tabla de la que se originó el contexto de fila.
- La transición de contexto se invoca siempre que hay un contexto de fila. Por ejemplo, si se usa CALCULATE en una columna calculada, se produce la transición de contexto. Hay un contexto de fila automático dentro de una columna calculada, y esto es suficiente para que ocurra la transición de contexto.

- La transición de contexto invalida los contextos de fila. Aunque hemos repetido este concepto varias veces, vale la pena llamar su atención nuevamente. Ninguno de los contextos de la fila externa es válido dentro de la expresión evaluada por CALCULATE. Todos los contextos de filas externas se transforman en contextos de filtro equivalentes.

Capítulo 10: CALCULATE y funciones de filtro

En el capítulo anterior aprendimos los dos contextos de evaluación que hacen parte del lenguaje DAX, el contexto de filtro y el contexto de fila. Sabemos entonces que el contexto de fila se genera automáticamente para una columna calculada, y se puede crear también un contexto de fila codificando a través de las funciones que utilizan un iterador, las cuáles son fácilmente identificables en DAX por que terminan con una X, por ejemplo: SUMX, AVERAGEX o COUNTX. El contexto de filtro por su parte es creado por el informe y las interacciones generadas a través de los objetos visuales en él, sin embargo, así como el contexto de fila podemos generarla codificando, el contexto de filtro no es la excepción y las funciones CALCULATE y CALCULATETABLE son las únicas funciones necesarias para lograr generar un contexto de filtro de manera programática y poder operar en él.

CALCULATE es la función más importante, útil y a la vez compleja de DAX, por esta razón dedicamos un capítulo completo para trabajar con ella, no obstante, la función por sí sola es bastante sencilla de aprender, ya que posee unas funcionalidades muy concretas. CALCULATE empieza a complejizarse por el hecho de que esta función junto con CALCULATETABLE son las únicas funciones en DAX que pueden crear nuevos contextos de filtro, por lo que al ser usadas dentro de una expresión aumenta su complejidad.



Nota: En este capítulo nos enfocaremos en la función CALCULATE y otras funciones complementarias a esta, la función CALCULATETABLE la veremos en el capítulo 11 CALCULATETABLE Vs FILTER.

Introducción a CALCULATE

Existen muchas razones por las que se hace necesaria la creación de nuevos contextos de filtro una de ellas es poder mejorar la manera en que codificamos las expresiones DAX para obtener ciertos resultados y que estas a su vez sean más claras de entender. Veámoslo con un ejemplo sencillo con base en nuestro modelo. Supongamos que el departamento de ventas nos solicita poder ver en un informe en el que se comparan los márgenes brutos en cantidad y porcentaje de los productos de la línea de Café en la ciudad de Medellín con el resto del país. Para esto necesitaríamos recurrir nuevamente a las fórmulas ya utilizadas en capítulos anteriores, Total Ventas y Margen Bruto y adicionalmente una métrica para calcular este Margen bruto como porcentaje. Tendríamos entonces las 3 formulas siguientes:



Total Ventas = `SUMX (Ventas, Ventas[Cantidad] * Ventas[Precio Unit])`



Margen bruto = `Ventas[Cantidad] * (Ventas[Precio Unit] - Ventas[CostoUnit])`



Margen bruto % = `DIVIDE([Margen bruto],[Total Ventas])`

Como podemos ver en la expresión Margen bruto % estamos reutilizando las dos métricas anteriores lo cual es una característica muy interesante de DAX que nos permite construir cálculos más complejos con las métricas ya existentes, esto nos evita reescribir código para métricas que ya calculan valores que necesitamos reutilizándolas dentro de otra expresión. Si ponemos estas tres métricas en una matriz tendríamos el resultado que muestra la Figura 10.1.

Linea	Nombre Producto	Total Ventas	Margen bruto	Margen bruto %
<input type="checkbox"/> ALIMENTO <input type="checkbox"/> ASEO <input type="checkbox"/> BEBIDAS <input checked="" type="checkbox"/> CAFÉ <input type="checkbox"/> CONDIMENTOS	INSTACREM ORIGINAL LOTERO SOBRE x18x3gr	\$6.694.513.472	\$1.740.326.017	26,00 %
	NESCAFE TRADICION FRASCO x85gr	\$1.681.381.194	\$537.328.165	31,96 %
	NESCAFE TRADICION x12 sobres x25gr	\$1.511.515.268	\$599.130.810	39,64 %
	NESCAFE TRADICION FRASCO x50 gr.	\$1.397.865.463	\$660.277.186	47,23 %
	NESCAFE TRADIC.x135 SOBRE GRTIS 9BOTADER	\$1.104.986.158	\$313.259.695	28,35 %
	NESCAFE TRADICION FRASCO x170gr	\$661.260.635	\$374.554.358	56,64 %
	NESCAFE TRADICION x12 sobres x10gr.	\$359.252.279	\$117.982.929	32,84 %
Total		\$13.410.774.469	\$4.342.859.160	32,38 %

Figura 10.1 Las tres métricas nos proporcionan un caso inicial de lo que nos solicitaron.

El siguiente paso sería poder visualizar para cada producto de esta línea, los márgenes de la ciudad de Medellín en dinero y porcentaje. Con los conocimientos adquiridos hasta ahora, deberíamos ser capaces de codificar estas dos métricas. Sin embargo, veamos como CALCULATE podría facilitarnos la vida en este escenario específico.

Sabemos que el contexto de filtro filtra el modelo. Por lo tanto, un contexto de filtro puesto en la columna Municipio [Nombre Municipio] filtrará la tabla Ventas debido a la relación que vincula ambas tablas. Usando el contexto de filtro, se puede filtrar una tabla indirectamente por que el contexto de filtro opera en todo el modelo.

Teniendo esto claro, si pudiéramos hacer que DAX calcule la medida del margen bruto mediante la creación de un contexto de filtro codificado en una expresión que sólo filtre los valores de la ciudad de Medellín, tendríamos resuelto el requerimiento. Y si, es lo que está pensando, esto es posible utilizando la función CALCULATE.

Antes que nada, examinemos la sintaxis de la función CALCULATE:

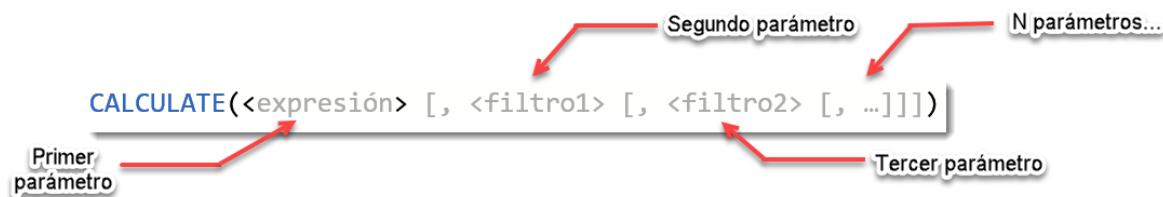


Figura 10.2 Sintaxis función CALCULATE

El primer parámetro es la expresión que CALCULATE evaluará. Antes de evaluar la expresión, CALCULATE calcula los argumentos de filtro y los usa para manipular el contexto del filtro. CALCULATE puede aceptar cualquier cantidad de parámetros. El único parámetro que es obligatorio es el primero, es decir, la expresión a evaluar. Los demás parámetros siguientes al primero se denominan argumentos de filtro y son opcionales. CALCULATE opera en el siguiente orden: crea un nuevo contexto de filtro con base a todo el conjunto de argumentos de filtro, posteriormente CALCULATE aplica este nuevo contexto al modelo y por último procede con la evaluación de la expresión.

Comencemos entonces con nuestro requerimiento, como nos están solicitando únicamente los márgenes de la ciudad de Medellín, podemos utilizar los argumentos de filtro de la función CALCULATE para restringir el cálculo de los márgenes a sólo esta ciudad:



Margen bruto Medellin =
`CALCULATE([Margen bruto], Municipios[Nombre municipio] = "MEDELLIN")`



Margen bruto % Medellin =
`CALCULATE([Margen bruto %], Municipios[Nombre municipio] = "MEDELLIN")`

Adicionando estas dos nuevas métricas en la matriz de la Figura 10.1 tendríamos el resultado mostrado en la Figura 10.3.

Nombre Producto	Total Ventas	Margen bruto	Margen bruto %	Margen bruto Medellin	Margen bruto % Medellin
INSTACREM ORIGINAL LOTERO SOBRE x18x3gr	\$6.694.513.472	\$1.740.326.017	26,00 %	\$545.336.743	46,25 %
NESCAFE TRADICION FRASCO x85gr	\$1.681.381.194	\$537.328.165	31,96 %	\$35.823.422	20,89 %
NESCAFE TRADICION x12 sobres x25gr	\$1.511.515.268	\$599.130.810	39,64 %	\$189.047.750	43,35 %
NESCAFE TRADICION FRASCO x50 gr.	\$1.397.865.463	\$660.277.186	47,23 %	\$35.395.381	91,45 %
NESCAFE TRADIC.x135 SOBRE GRTIS 9BOTADER	\$1.104.986.158	\$313.259.695	28,35 %	\$20.044.676	42,11 %
NESCAFE TRADICION FRASCO x170gr	\$661.260.635	\$374.554.358	56,64 %	\$123.836.688	87,36 %
NESCAFE TRADICION x12 sobres x10gr.	\$359.252.279	\$117.982.929	32,84 %	\$11.164.148	20,34 %
Total	\$13.410.774.469	\$4.342.859.160	32,38 %	\$960.648.808	46,41 %

Figura 10.3 Las dos nuevas columnas en la matriz muestran el margen en dinero y porcentaje para la ciudad de Medellín.

Al crear un contexto de filtro que obliga a la ciudad a ser Medellín, se pueden utilizar las métricas ya existentes para modificar su comportamiento sin tener que reescribir el código original de las mismas. Ahora que hemos tenido una aproximación inicial a esta función, empecemos a comprender los detalles de esta. Como lo mencionamos al inicio de este capítulo CALCULATE y CALCULATETABLE, son las únicas

funciones de DAX que pueden crear nuevos contextos de filtro. CALCULATE no modifica un contexto de filtro, crea un nuevo contexto de filtro fusionando sus argumentos de filtro con el contexto de filtro existente. Por ejemplo, si regresamos a observar la Figura 10.1 podemos ver que al lado izquierdo de la matriz hay una segmentación de datos en la que está seleccionada la línea “CAFÉ” que era parte del requerimiento inicial, sin embargo, nótese que los valores de margen para Medellín en la matriz siguen estando filtrados para todos los productos de esta línea, allí vemos un claro ejemplo de que CALCULATE fusiona el contexto de filtro existente en el informe con los argumentos de filtro de la función.

La primera cosa importante para tener en cuenta sobre CALCULATE es que los argumentos del filtro no son condiciones booleanas (Evalúan verdadero o falso), **son tablas**. *Siempre que utilice una condición booleana como argumento de filtro de CALCULATE, DAX lo traducirá a una tabla de valores*. Cuando esté aprendiendo CALCULATE por primera vez, le recomendamos interpretar los argumentos de filtro como tablas. Esto hace que el comportamiento de CALCULATE en primera instancia sea más evidente.

The screenshot shows a Power BI report. At the top, there is a calculated table with columns: Nombre Producto, Total Ventas, Margen bruto, Margen bruto %, Margen bruto Medellin, and Margen bruto % Medellin. The last two columns are highlighted with a red border. Below the table is a visualization of filter context, showing dropdown menus for Código Departamento (05), Nombre municipio (MEDELLIN), Código Municipio (05001), Latitud (6,2913889), and Longitud (-75,5361111). An arrow points from the highlighted table cells down to the filter context visualization.

Nombre Producto	Total Ventas	Margen bruto	Margen bruto %	Margen bruto Medellin	Margen bruto % Medellin
INSTACREM ORIGINAL LOTERO SOBRE x18x3gr	\$6.694.513.472	\$1.740.326.017	26,00 %	\$545.336.743	46,25 %
NESCAFE TRADICION FRASCO x85gr	\$1.681.381.194	\$537.328.165	31,96 %	\$35.823.422	20,89 %
NESCAFE TRADICION x12 sobres x25gr	\$1.511.515.268	\$599.130.810	39,64 %	\$189.047.750	43,35 %
NESCAFE TRADICION FRASCO x50 gr.	\$1.397.865.463	\$660.277.186	47,23 %	\$35.395.381	91,45 %
NESCAFE TRADIC.x135 SOBRE GRTIS 9BOTADER	\$1.104.986.158	\$313.259.695	28,35 %	\$20.044.676	42,11 %
NESCAFE TRADICION FRASCO x170gr	\$661.260.635	\$374.554.358	56,64 %	\$123.836.688	87,36 %
NESCAFE TRADICION x12 sobres x10gr.	\$359.252.279	\$117.982.929	32,84 %	\$11.164.148	20,34 %
Total	\$13.410.774.469	\$4.342.859.160	32,38 %	\$960.648.808	46,41 %

Figura 10.4 Se muestra gráficamente como los argumentos de filtro son en realidad tablas

En la sección anterior usamos esta fórmula:



Margen bruto Medellin =
`CALCULATE([Margen bruto], Municipios[Nombre municipio] = "MEDELLIN")`

El uso de una condición booleana es sólo un atajo para la sintaxis CALCULATE completa. Esto se conoce como *syntax sugar*. DAX internamente ejecutaría la siguiente expresión:



Margen bruto Medellin =
`CALCULATE (`
 `[Margen bruto],`
 `FILTER (`
 `ALL (Municipios[Nombre municipio]),`
 `Municipios[Nombre municipio] = "MEDELLIN"`
 `)`
`)`

Las dos sintaxis son equivalentes y no existen diferencias semánticas o de rendimiento entre ellas. Un argumento de filtro dentro de CALCULATE es una tabla, es decir, una lista de valores. La tabla proporcionada como argumento de filtro define la lista de valores que será visibles para la columna durante la evaluación de la expresión. En las dos métricas anteriores en las que utilizamos CALCULATE,

el argumento de filtro en la función estaría devolviendo una tabla con una sola fila, que corresponde al valor “MEDELLIN” de la columna Municipios [Nombre municipio]. Por esta razón, CALCULATE filtra el modelo donde se incluyan productos de la ciudad de Medellín.

Otra cosa importante es que CALCULATE no sobrescribe todo el contexto del filtro original. Solo reemplaza los filtros existentes previamente en las columnas contenidas en el argumento del filtro, es decir, fusiona los filtros existentes con los argumentos de filtro, de hecho, si se cambia la matriz de la Figura 10.3 para que ahora se muestre por la línea de producto, el resultado será el mostrado en la Figura 10.5.

Linea	Total Ventas	Margin bruto	Margin bruto %	Margin bruto Medellin	Margin bruto % Medellin
BEBIDAS	\$32.452.118.734	\$10.542.877.496	32,49 %	\$788.884.585	40,12 %
ALIMENTO	\$21.653.792.830	\$6.896.501.342	31,85 %	\$679.694.469	34,00 %
ASEO	\$15.775.403.827	\$5.871.769.960	37,22 %	\$568.358.431	38,87 %
CONDIMENTOS	\$13.485.438.855	\$4.233.994.976	31,40 %	\$386.816.506	38,79 %
CAFÉ	\$13.410.774.469	\$4.342.859.160	32,38 %	\$960.648.808	46,41 %
Total	\$96.777.528.715	\$31.888.002.934	32,95 %	\$3.384.402.799	39,84 %

Figura 10.5 A pesar de que la matriz filtra por la línea de producto, el filtro de Ciudad se fusionará y no se sobreescrivirá

Vemos entonces que la matriz ahora filtra por la línea de producto, mientras que CALCULATE aplica un filtro en Municipios [Nombre municipio] para evaluar las métricas Margen bruto Medellín y Margen bruto % Medellín. Los dos filtros no funcionan en la misma columna, puesto que están en tablas diferentes, incluso si estuviesen en la misma tabla, pero en columnas diferentes, en ambos escenarios no se sobreescriviría y los dos filtros funcionan juntos como un nuevo contexto de filtro. Como resultado la matriz nos muestra los márgenes de las dos últimas columnas para la ciudad de Medellín para la línea de producto en cada fila. Por lo tanto, haciendo una aclaración sobre este último concepto, CALCULATE sobreescrive los filtros si están en la misma columna y los fusiona si están en columnas diferentes.

Como ha visto hasta ahora, CALCULATE en sí mismo no es una función compleja. Su comportamiento es sencillo de describir. Al mismo tiempo, tan pronto como empiece a usar CALCULATE, la complejidad del código se vuelve mucho mayor. Lo importante es centrarse en el contexto de filtro y comprender exactamente como CALCULATE genera nuevos contextos de filtro. La clave para dominar todo el poder del lenguaje DAX está en dominar los contextos de evaluación.

Ahora que hemos analizado las características básicas de CALCULATE, podemos resumir su semántica:

- CALCULATE realiza una copia del contexto de filtro actual.
- CALCULATE evalúa cada argumento de filtro y produce, para cada condición, la lista de valores válidos para las columnas especificadas.
- Si dos o más argumentos de filtro afectan a la misma columna, se fusionan usando un operador AND (o usando la intersección establecida en términos matemáticos).

- CALCULATE utiliza la nueva condición para reemplazar los filtros existentes en las columnas del modelo. Si una columna ya tiene un filtro, el nuevo filtro reemplaza al existente. Por otro lado, si la columna no tiene un filtro, CALCULATE agrega el nuevo filtro al contexto del filtro.
- Una vez que el nuevo contexto de filtro está listo, CALCULATE aplica el contexto de filtro al modelo y calcula el primer argumento: la expresión. Al final, CALCULATE restaura el contexto del filtro original y devuelve el resultado calculado.
- CALCULATE acepta filtros de dos tipos:
 - 1. Listas de valores, en forma de expresión de tabla. En ese caso, proporciona la lista exacta de valores que desea hacer visible en el nuevo contexto de filtro. El filtro puede ser una tabla con cualquier número de columnas. En el filtro solo se considerarán las combinaciones de valores existentes en diferentes columnas.
 - 2. Condiciones booleanas, como Producto [Color] = "Blanco". Estos filtros deben funcionar en una sola columna porque el resultado debe ser una lista de valores para una sola columna. Este tipo de argumento de filtro también se conoce como predicado.



Nota: CALCULATE realiza otra tarea muy importante que vimos en el capítulo 9 y es que transforma cualquier contexto de fila existente en un contexto de filtro equivalente. Esto es a lo que se le llama transición de contexto. Entonces recuerde: CALCULATE crea un contexto de filtro a partir de los contextos de fila existentes.

A partir de aquí vamos a empezar a ver algunas funciones de filtro, pero desde la perspectiva y aplicación de la función CALCULATE que podrían denominarse también como modificadores de CALCULATE.

ALL

Ya vimos la función ALL en el Capítulo 6: Principales funciones de tabla. No obstante, en este capítulo lo estudiaremos ya no como tabla sino como argumento de filtro, por lo tanto, analizaremos algunas particularidades que se generan al utilizar esta función como argumento de filtro dentro de CALCULATE.

Un comportamiento que aparece a menudo con esta función es el de los porcentajes. Al trabajar con porcentajes, es muy importante definir exactamente el cálculo requerido. En los siguientes ejemplos vamos a ver como algunos usos diferentes de las funciones CALCULATE y ALL proporcionan resultados diferentes.

Comencemos entonces con un simple cálculo de porcentaje. Crearemos una nueva métrica que nos devuelva el total de las ventas, pero esta vez en porcentaje. Para esto será necesario dividir el valor de la métrica que ya creamos previamente “Total Ventas” en un contexto de filtro específico entre el valor de “Total Ventas” en un contexto de filtro que ignore el filtro existente, tratemos de entenderlo mejor con la Figura 10.6.

The diagram illustrates a context filter and a denominator for calculating percentages. On the left, a callout labeled "Contexto de filtro actual" points to the row for "ALIMENTO" in a table. On the right, another callout labeled "Denominador para calcular %" points to the "Total" row, specifically the value "\$96.777.528.715". Red arrows indicate the flow from the context filter to the specific row and from the total row to the denominator.

Linea	Total Ventas
ALIMENTO	\$21.653.792.830
ASEO	\$15.775.403.827
BEBIDAS	\$32.452.118.734
CAFÉ	\$13.410.774.469
CONDIMENTOS	\$13.485.438.855
Total	\$96.777.528.715

Figura 10.6 Total Ventas dado por la línea de producto como contexto de filtro actual y gran total como denominador a utilizar para calcular el porcentaje de las ventas.

De acuerdo con la Figura 10.6 lo que queremos lograr es tomar el valor de cada fila y dividirlo por el gran total, es decir, para la fila uno de la línea ALIMENTO tomar el valor 21.653.792.830 del contexto actual y dividirlo entre 96.777.528.715 ignorando el contexto existente dado por la línea de producto.

En cada fila del informe, el contexto de filtro ya contiene la línea actual. Por lo tanto, para Total Ventas, el resultado se filtra automáticamente por la línea dada. El denominador de la relación debe ignorar el contexto de filtro actual, de tal forma que evalúe el total general. Debido a que los argumentos de filtro de CALCULATE son tablas, es suficiente proporcionar una función de tabla que ignore el contexto de filtro actual para la línea de producto y siempre devuelva todas las líneas, independientemente de cualquier filtro. En el capítulo 6 aprendimos que esta función es ALL. Por lo tanto, definiremos la siguiente fórmula para ver cómo actuaría en el cálculo del denominador para calcular dicho porcentaje:



Ventas todas las líneas =
`CALCULATE([Total Ventas], ALL('Linea'[Linea]))`

ALL elimina el filtro en la columna Linea [Linea] del contexto de filtro. Por lo tanto, en cualquier celda de la matriz, ignora cualquier filtro existente en las líneas de producto. El efecto es que se elimina el filtro de línea de producto aplicado por la fila de la matriz. En la Figura 10.7 podemos apreciar como en cada fila de la matriz la métrica “Ventas todas las líneas” devuelve el mismo valor hasta el final, es decir, el gran total de las ventas.

Linea	Total Ventas	Ventas todas las líneas
ALIMENTO	\$21.653.792.830	96.777.528.715
ASEO	\$15.775.403.827	96.777.528.715
BEBIDAS	\$32.452.118.734	96.777.528.715
CAFÉ	\$13.410.774.469	96.777.528.715
CONDIMENTOS	\$13.485.438.855	96.777.528.715
Total	\$96.777.528.715	96.777.528.715

Figura 10.7 CALCULATE define un contexto de filtro sin ningún filtro de línea gracias a que ALL ha eliminado dicho filtro para la línea de productos.

Como necesitamos calcular el porcentaje de las ventas, esta métrica que vemos en la Figura 10.7 por sí sola no nos proporciona gran utilidad, sería extraño ver en un informe una métrica que muestre los mismos valores para todas las filas, sin embargo, esta métrica contiene el valor para cada fila que es exactamente el que necesitamos como denominador para calcular el porcentaje de las ventas. Entonces utilizándola en una expresión podremos calcular el porcentaje que nos están solicitando, una manera de hacerlo sería con la siguiente fórmula:



Total ventas % =
`DIVIDE([Total Ventas], [Ventas todas las líneas],)`

Y el resultado de esta métrica utilizado en la matriz sería el que muestra la Figura 10.8.

Linea	Total Ventas	Total ventas %
ALIMENTO	\$21.653.792.830	22,37 %
ASEO	\$15.775.403.827	16,30 %
BEBIDAS	\$32.452.118.734	33,53 %
CAFÉ	\$13.410.774.469	13,86 %
CONDIMENTOS	\$13.485.438.855	13,93 %
Total	\$96.777.528.715	100,00 %

Figura 10.8 Se calcula el porcentaje correctamente para cada línea de producto, resultando el 100% en el gran total.

Como vemos en este ejemplo, el uso combinado de funciones de tabla como ALL con CALCULATE permite crear métricas útiles de manera sencilla. Sin embargo, hasta ahora todo ha sido color de rosa, no hemos visto los “resultados diferentes” que pueden resultar del uso de ALL con CALCULATE como lo mencionamos al principio del capítulo. Prestemos atención a este pequeño detalle a continuación al momento de crear porcentajes de esta manera:

El porcentaje se calcula bien si la matriz está dada por la línea de producto, ya que el código elimina el filtro de la línea de producto y no toca ningún otro filtro existente. Pero ¿Qué sucede si se agrega otro filtro a la matriz o se cambia el filtro de línea? Veamos la Figura 10.9 para averiguarlo.

Linea	Total Ventas	Total ventas %	Departamento	Total Ventas	Total ventas %
ALIMENTO	\$21.653.792.830	22,37 %	AMAZONAS	\$970.154.187	100,00 %
CORN FLAKES CEREAL INTEGRAL x 800 gr	\$4.548.499.194	100,00 %	ANTIOQUIA	\$43.331.521.758	100,00 %
NACHOS X 180 GRS. FRESCAMPO	\$786.629.742	100,00 %	ATLÁNTICO	\$65.710.642	100,00 %
NESTUM 5 CEREALES x 200 gr	\$2.029.910.962	100,00 %	BOGOTA, D.C.	\$8.638.030.031	100,00 %
POPETAS CARAMELO-QUESO FAMILIAR x 135 gr	\$7.552.098.972	100,00 %	BOLÍVAR	\$692.746.774	100,00 %
QUIPITOS SOBRE x 24 x 8gr	\$4.884.741.089	100,00 %	BOYACÁ	\$59.959.143	100,00 %
TOSTADAS X 145 GR	\$1.851.912.871	100,00 %	CALDAS	\$511.878.776	100,00 %
ASEO	\$15.775.403.827	16,30 %	CAQUETÁ	\$304.566.085	100,00 %
BIOEXPERT CHAMPU CAP.GUSANO SEDAx18x15ml	\$4.256.483.181	100,00 %	CAUCA	\$33.549.695	100,00 %
EGO GEL EXTREME MAX MEGA x15 x27 ml.	\$2.280.306.567	100,00 %	CESAR	\$919.872.458	100,00 %
ELITE SERVILLETAS PRACTICA x 320	\$914.725.625	100,00 %	CHOCÓ	\$262.048.076	100,00 %
NUTRIBELA NUTRICION FRASCO x 300 ml	\$4.954.065.292	100,00 %	CORDOBA	\$1.141.314.673	100,00 %
NUTRIBELA REPOLARIZACION x27 mlx12	\$2.135.553.681	100,00 %	CUNDINAMARCA	\$18.816.843.364	100,00 %
SAVITAL CHAMPU BIOTINA x 550 ml.	\$1.234.269.481	100,00 %	HUILA	\$1.110.546.737	100,00 %
BEBIDAS	\$32.452.118.734	33,53 %	LA GUAJIRA	\$54.669.400	100,00 %
FRUTTSI FRUTOS ROJOS PET x 400 ml	\$2.178.092.924	100,00 %	MAGDALENA	\$1.211.069.853	100,00 %
MILLO DOY PACK x 500gr EXTRACONT 10%	\$460.687.466	100,00 %	META	\$482.555.470	100,00 %
MILLO LATA x 400gr	\$2.202.462.729	100,00 %	NARIÑO	\$1.169.139.394	100,00 %
MILLO RISTRAS x 16 UNIDS. x 25 gr.	\$3.706.246.914	100,00 %	NORTE DE SANTANDER	\$1.215.302.011	100,00 %
MILLO RISTRAS x 18 SORRES x 25 gr.	\$992.992.592	100,00 %	RISARALDA	\$4.080.642.580	100,00 %
Total	\$96.777.528.715	100,00 %	Total	\$96.777.528.715	100,00 %

Figura 10.9 En ambas matrices se ven valores inconsistentes para el porcentaje de ventas.

Efectivamente, el resultado es completamente diferente para la métrica Total ventas % y genera valores inconsistentes a nuestro objetivo, en el lado izquierdo de la Figura 10.9 tenemos un nivel adicional dentro de línea que sería cada producto perteneciente a esa línea, si bien el porcentaje se sigue calculando correctamente a nivel de línea, a nivel de producto arroja resultados inconsistentes. Por otro lado, en la parte derecha tenemos un cambio de filtro, ya no estamos cruzando con línea de producto sino con departamento y nuevamente aquí los resultados no son los esperados. En ambos casos cambió el contexto de filtro y la métrica Total Ventas % ya no cubre dichos escenarios.

Para comprender exactamente lo que sucedió debemos analizar los contextos de filtro. El contexto de filtro original, es decir, el de la Figura 10.5 tiene sólo un filtro a nivel de línea de producto, el nuevo contexto de filtro del lado izquierdo de la Figura 10.9 ahora contiene un filtro tanto de línea como de producto. CALCULATE no sobrescribe el filtro de Maestro Productos [Nombre producto], sólo elimina el filtro de Linea [Linea] por lo que, como resultado, el contexto de filtro final sólo contiene el producto. De esta manera el denominador de la relación contiene las ventas de todos los productos de la línea correspondiente y de todas las demás líneas a su vez. Si arrastramos nuevamente la métrica que calcula el denominador a la matriz, lo veremos más claramente como en la Figura 10.10.

Linea	Total Ventas	Total ventas %	Ventas todas las líneas
ALIMENTO	\$21.653.792.830	22,37 %	96.777.528.715
BIOEXPERT CHAMPU CAP.GUSANO SEDAx18x15ml			4.256.483.181
CALDO GALLINA x 1.8kgr x 165 CUBOS			329.762.199
CALDO GALLINA x 2.97 kgr X 270 CUBOS			2.692.701.294
CALDO RICOSTILLA CUBOS x 60x660qr			7.259.735.576
CORN FLAKES CEREAL INTEGRAL x 800 gr	\$4.548.499.194	100,00 %	4.548.499.194
EGO GEL EXTREME MAX MEGA x15 x27 ml.			2.280.306.567
ELITE SERVILLETA PRACTICA x 320			914.725.625
FRUTTSI FRUTOS ROJOS PET x 400 ml			2.178.092.924
INSTACREM ORIGINAL LOTERO SOBRE x18x3gr			6.694.513.472
MAGGI C.GALLINA CUBOS x48 x 528 gr			3.203.239.786
MILO DOY PACK x 500gr EXTRACONT 10%			460.687.466
MILO LATA x 400gr			2.202.462.729
MILO RISTRA x 16 UNIDS. x 25 gr.			3.706.246.914
MILO RISTRA x 18 SOBRES x 25 gr.			992.592.502
MILO TETRA PACK x 180 ml			1.372.798.912
NACHOS X 180 GRS. FRESCAMPO	\$786.629.742	100,00 %	786.629.742
NESCAFE TRADIC.x135 SOBRE GRTIS 9BOTADER			1.104.986.158
NESCAFE TRADICION FRASCO x170gr			661.260.635
NESCAFE TRADICION FRASCO x50 gr.			1.397.865.463
NESCAFE TRADICION FRASCO x85gr			1.681.381.194
NESCAFE TRADICION x12 sobres x25gr			1.511.515.268
NESCAFE TRADICION x12 sobres x10gr.			359.252.279
NESTUM 5 CEREALES x 200 gr	\$2.029.910.962	100,00 %	2.029.910.962
NUTRIBELA NUTRICION FRASCO x 300 ml			4.954.065.292
NUTRIBELA REPOLARIZACION x27 mlx12			2.135.553.681
POPETAS CARAMELO-QUESO FAMILIAR x 135 qr	\$7.552.098.972	100,00 %	7.552.098.972
Total	\$96.777.528.715	100,00 %	96.777.528.715

Figura 10.10 Se observan seleccionados a la izquierda productos que no corresponden a la línea de alimentos y a la derecha los valores de todos los productos sin importar la línea.

El cálculo incorrecto no es un comportamiento inesperado de CALCULATE, simplemente es que la fórmula de la métrica fue creada para trabajar específicamente con el filtro de línea de producto. Entonces el objetivo es corregir el cálculo para que se calcule para este nuevo escenario. La métrica quedaría entonces así:



```

Total ventas % =
VAR TotalVentas = [Total Ventas]
VAR VentasLineayProducto =
CALCULATE (
    [Total Ventas],
    ALL ( Linea[Linea] ),
    ALL ( 'Maestro Productos'[Nombre Producto] )
)
RETURN
DIVIDE ( TotalVentas, VentasLineayProducto )

```

En esta ocasión almacenamos el resultado de la métrica Total Ventas en una variable que sería nuestro numerador, y el denominador lo calculamos en la variable VentasLineayProducto adicionando esta vez un ALL para el contexto de filtro de los productos, posteriormente hacemos la división entre ambas variables y retornamos el resultado de esa operación.

Y el resultado en la matriz sería el siguiente:

Línea	Total Ventas	Total ventas %
ALIMENTO	\$21.653.792.830	22,37 %
CORN FLAKES CEREAL INTEGRAL x 800 gr	\$4.548.499.194	4,70 %
NACHOS X 180 GR. FRESCAMPO	\$786.629.742	0,81 %
NESTUM 5 CEREALES x 200 gr	\$2.029.910.962	2,10 %
POPETAS CARAMELO-QUESO FAMILIAR x 135 gr	\$7.552.098.972	7,80 %
QUIPITOS SOBRE x 24 x 8gr	\$4.884.741.089	5,05 %
TOSTADAS X 145 GR	\$1.851.912.871	1,91 %
ASEO	\$15.775.403.827	16,30 %
BIOEXPERT CHAMPU CAP.GUSANO SEDAx18x15ml	\$4.256.483.181	4,40 %
EGO GEL EXTREME MAX MEGA x15 x27 ml.	\$2.280.306.567	2,36 %
ELITE SERVILLETA PRACTICA x 320	\$914.725.625	0,95 %
NUTRIBELA NUTRICION FRASCO x 300 ml	\$4.954.065.292	5,12 %
NUTRIBELA REPOLARIZACION x27 mlx12	\$2.135.553.681	2,21 %
SAVITAL CHAMPU BIOTINA x 550 ml.	\$1.234.269.481	1,28 %
BEBIDAS	\$32.452.118.734	33,53 %
FRUTTSI FRUTOS ROJOS PET x 400 ml	\$2.178.092.924	2,25 %
MILO DOY PACK x 500gr EXTRACONT 10%	\$460.687.466	0,48 %
MILO LATA x 400gr	\$2.202.462.729	2,28 %
MILO RISTRA x 16 UNIDS. x 25 gr.	\$3.706.246.914	3,83 %
MILO RISTRA x 18 SOBRES x 25 gr.	\$992.592.502	1,03 %
MILO TETRA PACK x 180 ml	\$1.372.798.912	1,42 %
PULPIFRUTA MORA 12 x160gr.	\$351.151.083	0,36 %
SAVILOE AGUA DE ALOE x 270 ml	\$11.953.022.626	12,35 %
SUNTEA DURAZNO x 4.5 LITROS x 266 gr.	\$7.979.938.469	8,25 %
SUNTEA LIMON x9 LITROS x 532 gr	\$1.255.125.109	1,30 %
CAFÉ	\$13.410.774.469	13,86 %
INSTACREM ORIGINAL LOTERO SOBRE x18x3qr	\$6.694.513.472	6,92 %
Total	\$96.777.528.715	100,00 %

Figura 10.11 Los porcentajes se calculan correctamente ya que se están teniendo en cuenta todos los contextos de filtro involucrados en la matriz.

Hasta ahora, hemos visto que al usar CALCULATE y ALL juntos en una expresión, puede eliminar filtros de una o varias columnas en la misma tabla o en diferentes tablas, sin embargo, también es posible eliminar filtros de una tabla completa, es decir, que se eliminan los filtros de cualquier columna de una tabla que se utilice en la matriz, simplemente usando la función ALL y especificando únicamente la tabla sin columnas.

El objetivo de esta sección es explicar algunas técnicas básicas para manipular el contexto de filtro, sin embargo, si tuviéramos un escenario en el que hay muchos filtros que están definiendo el contexto de filtro, sería muy tedioso codificarlos para tenerlos todos en cuenta con la función ALL. Por esta razón existe un enfoque más fácil para resolver este requisito específico de calcular un porcentaje dividiendo sobre el gran total visible y esto es posible por medio de la función ALLSELECTED.

ALLSELECTED

ALLSELECTED puede utilizarse como una función de tabla o como un modificador de CALCULATE dependiendo del número de parámetros y del contexto donde se utilice, sin embargo, en este capítulo nos enfocaremos en verlo como lo segundo, un modificador de CALCULATE.

ALLSELECTED actúa como un modificador de CALCULATE cuando se usa como un argumento de filtro dentro de CALCULATE. De hecho, ALLSELECTED se puede usar con tres parámetros diferentes: una tabla, una columna o ningún parámetro, como en los siguientes ejemplos:



```
AllSelectedColumna =  
CALCULATE ( [Total Ventas], ALLSELECTED ( 'Maestro Productos'[Nombre Producto] ) )
```



```
AllSelectedTabla =  
CALCULATE ( [Total Ventas], ALLSELECTED ( 'Maestro Productos' ) )
```



```
AllSelectedTodo =  
CALCULATE ( [Total Ventas], ALLSELECTED () )
```

En la siguiente Figura podemos ver el resultado de estas tres métricas cuando las utilizamos sobre una matriz que contiene el nombre del municipio, nombre del producto y el código del producto:

Nombre Producto

- NESCAFE TRADICION FRASCO x50 gr.
- NESCAFE TRADICION FRASCO x85gr
- NESCAFE TRADICION x12 sobres x2...
- NESCAFE TRADICION x12 sobres x1...
- NESTUM 5 CEREALES x 200 gr
- NUTRIBELA NUTRICION FRASCO x 3...
- NUTRIBELA REPOLARIZACION x27 ...
- POPETAS CARAMELO-QUESO FAMI...
- PULPIFRUTA MORA 12 x160gr.
- QUIPITOS SOBRE x 24 x 8gr
- SAVILOE AGUA DE ALOE x 270 ml
- SAVITAL CHAMPU BIOTINA x 550 ml.
- SUNTEA DURAZNO x 4.5 LITROS x 2...
- SUNTEA LIMON x9 LITROS x 532 gr
- TOSTADAS X 145 GR

Nombre municipio

- AGUACHICA
- APARTADO

Buscar

	Nombre municipio	Total Ventas	AllSelectedColumna	AllSelectedTabla	AllSelectedTodo
<input checked="" type="checkbox"/> AGUACHICA	<input checked="" type="checkbox"/> NESCAFE TRADICION FRASCO x50 gr.	\$51.442.269	51.442.269	51.442.269	449.226.729
	6264	\$6.297.765	6.297.765	51.442.269	449.226.729
<input checked="" type="checkbox"/> APARTADO	<input checked="" type="checkbox"/> NESCAFE TRADICION FRASCO x85gr	\$485.790	51.442.269	51.442.269	449.226.729
	2950	\$485.790	485.790	51.442.269	449.226.729
<input checked="" type="checkbox"/> SUNTEA DURAZNO x 4.5 LITROS x 266 gr.	6202	\$17.282.151	51.442.269	51.442.269	449.226.729
	5731	\$17.282.151	17.282.151	51.442.269	449.226.729
<input checked="" type="checkbox"/> SUNTEA LIMON x9 LITROS x 532 gr	5731	\$27.376.563	51.442.269	51.442.269	449.226.729
	5731	\$27.376.563	27.376.563	51.442.269	449.226.729
<input checked="" type="checkbox"/> SUNTEA DURAZNO x 4.5 LITROS x 266 gr.	6264	\$397.784.460	397.784.460	397.784.460	449.226.729
<input checked="" type="checkbox"/> SUNTEA LIMON x9 LITROS x 532 gr	2950	\$26.542.586	397.784.460	397.784.460	449.226.729
<input checked="" type="checkbox"/> SUNTEA DURAZNO x 4.5 LITROS x 266 gr.	6202	\$95.126.018	397.784.460	397.784.460	449.226.729
<input checked="" type="checkbox"/> SUNTEA LIMON x9 LITROS x 532 gr	5731	\$78.765.905	397.784.460	397.784.460	449.226.729
Total		\$197.349.951	397.784.460	397.784.460	449.226.729
		\$449.226.729	449.226.729	449.226.729	449.226.729

Figura 10.12 Se muestran los valores de Total Ventas para cada una de las columnas en el contexto de filtro de columna, tabla y todo el modelo.

Básicamente lo que hace ALLSELECTED en cada uno de los 3 casos de la Figura 10.12 es eliminar el contexto de filtro de la celda actual de acuerdo con el nivel de filtro dado en la fórmula, nivel de columna, de tabla y de modelo, al mismo tiempo que conserva todos los demás filtros del contexto externo, es decir, los filtros que están por fuera de la matriz (Las segmentaciones de datos por Nombre producto y Nombre municipio).

Por lo tanto, para la métrica **AllSelectedColumna** el contexto de filtro está dado para la columna Maestro productos [Nombre producto] por esta razón los valores se muestran detallados para cada producto, mientras que la métrica **AllSelectedTabla** el contexto de filtro está dado para toda la tabla de Maestro productos, debido a esto los valores que se muestran en dicha columna se ven detallados sólo a nivel de los productos, si observamos bien, cada que hay un nuevo municipio o ciudad es cuando los valores cambian, es decir que para cada ciudad están mostrándose los valores de todos los productos por que el contexto de filtro de toda la tabla se eliminó.

Y por último para la métrica **AllSelectedTodo** se eliminan todos los contextos de filtro que hay en la matriz, es decir, los de Ciudad, Nombre producto y código producto, por lo que los valores son siempre el total de todos los productos.

Sin embargo, hay que tener en cuenta que, para los 3 casos, si se tienen en cuenta los contextos de filtro externos, es decir, para la Figura 10.12, se están teniendo en cuenta o restaurando dentro del contexto de la fórmula los filtros de las segmentaciones de datos de Nombre Producto y Nombre municipio. Básicamente se eliminan los contextos de filtro correspondientes dentro de la matriz y se aplican los externos, donde Nombre producto contenga los valores seleccionados en la segmentación de datos “Nombre producto” y a su vez Nombre municipio contenga los valores seleccionados en la segmentación de datos “Nombre municipio”.

Si regresáramos al ejemplo del cálculo del porcentaje de ventas, utilizando la función ALLSELECTED sin parámetros, para retornar siempre en el denominador el valor total de ventas, ignorando el contexto de filtro de la matriz, tendríamos una forma sencilla de calcular este porcentaje teniendo en cuenta todos los posibles escenarios de cruce dentro de la matriz y conservando los filtros externos,

básicamente, tendríamos un cálculo de porcentaje “a prueba de todo” sin tener que codificar todos los posibles escenarios como con la función ALL. Veamos cómo sería la métrica:



```
Total ventas % AllSelected =  
DIVIDE ( [Total Ventas],  
CALCULATE ( [Total Ventas], ALLSELECTED () )  
)
```

Y para el mismo escenario de la Figura 10.9 de la sección ALL tendremos valores correctos en ambos escenarios con la misma métrica como se ve en la siguiente Figura:

Línea	Total Ventas	Total ventas % AllSelected	Departamento	Total Ventas	Total ventas % AllSelected
ALIMENTO	\$21.653.792.830	22,37 %	AMAZONAS	\$970.154.187	1,00 %
CORN FLAKES CEREAL INTEGRAL x 800 gr	\$4.548.499.194	4,70 %	ANTIOQUIA	\$43.331.521.758	44,77 %
NACHOS X 180 GRS. FRESCAMPO	\$786.629.742	0,81 %	ATLÁNTICO	\$65.710.642	0,07 %
NESTUM 5 CEREALES x 200 gr	\$2.029.910.962	2,10 %	BOGOTÁ, D.C.	\$8.638.030.031	8,93 %
POQUITOS CARAMELO-QUESO FAMILIAR x 135 gr	\$7.552.098.972	7,80 %	BOLÍVAR	\$692.746.774	0,72 %
QUIPITOS SOBRE x 24 x 8gr	\$4.884.741.089	5,05 %	BOYACÁ	\$59.959.143	0,06 %
TOSTADAS X 145 GR	\$1.851.912.871	1,91 %	CALDAS	\$511.878.776	0,53 %
ASEO	\$15.775.403.827	16,30 %	CAQUETÁ	\$304.566.085	0,31 %
BIOEXPERT CHAMPU CAP.GUSANO SEDAx18x15ml	\$4.256.483.181	4,40 %	CAUCA	\$33.549.695	0,03 %
EGO GEL EXTREME MAX MEGA x15 x27 ml.	\$2.280.306.567	2,36 %	CESAR	\$919.872.458	0,95 %
ELITE SERVILLETAS PRACTICA x 320	\$914.725.625	0,95 %	CHOCÓ	\$262.048.076	0,27 %
NUTRIBELA NUTRICION FRASCO x 300 ml	\$4.954.065.292	5,12 %	CÓRDOBA	\$1.141.314.673	1,18 %
NUTRIBELA REPOLARIZACION x27 mlx12	\$2.135.553.681	2,21 %	CUNDINAMARCA	\$18.816.843.364	19,44 %
SAVITAL CHAMPU BIOTINA x 550 ml.	\$1.234.269.481	1,28 %	HUILA	\$1.110.546.737	1,15 %
BEBIDAS	\$32.452.118.734	33,53 %	LA GUAJIRA	\$54.669.400	0,06 %
FRUTTSI FRUTOS ROJOS PET x 400 ml	\$2.178.092.924	2,25 %	MAGDALENA	\$1.211.069.853	1,25 %
MILO DOY PACK x 500gr EXTRACONT 10%	\$460.687.466	0,48 %	META	\$482.555.470	0,50 %
MILO LATA x 400gr	\$2.202.462.729	2,28 %	NARIÑO	\$1.169.139.394	1,21 %
MILO RISTRA x 16 UNIDS. x 25 gr.	\$3.706.246.914	3,83 %	NORTE DE SANTANDER	\$1.215.302.011	1,26 %
MILO RISTRA x 18 SOBRES x 25 gr.	\$992.592.502	1,03 %	RISARALDA	\$4.080.642.580	4,22 %
MILO TETRA PACK x 180 ml	\$1.372.798.912	1,42 %	SANTANDER	\$1.906.382.785	1,97 %
PULPIFRUTA MORA 12 x160gr.	\$351.151.083	0,36 %	SUCRE	\$201.542.056	0,21 %
SAVILCE AGUA DE ALOE x 270 ml	\$11.953.022.626	12,35 %	TOLIMA	\$706.110.324	0,73 %
SUNTEA DURAZNO x 4,5 LITROS x 266 gr.	\$7.979.938.469	8,25 %	VALLE DEL CAUCA	\$8.841.128.507	9,14 %
SUNTEA LIMON x9 LITROS x 532 gr	\$1.255.125.109	1,30 %	VAUPÉS	\$50.243.936	0,05 %
CAFÉ	\$13.410.774.469	13,86 %	Total	\$96.777.528.715	100,00 %
INSTACREM ORIGINAL LOTERO SOBRE x18x3ar	\$6.694.513.472	6,92 %			
Total	\$96.777.528.715	100,00 %			

Figura 10.13 Se muestran los valores de Total Ventas % con valores correctos en ambos escenarios con contextos de filtro distintos y utilizando la misma métrica.

Comprobamos entonces que para la matriz de la izquierda la función ALLSELECTED sin parámetros está forzando a que se ignoren los filtros de filas y columnas, se consideren en el cálculo de la métrica las ventas de todas las líneas y productos en su respectivo nivel y se considere el contexto impuesto “externamente” que, en este caso, no existe, pues no hemos realizado ninguna selección en las segmentaciones de datos. Lo mismo sucede en la matriz de la derecha se ignoran los filtros de las filas y columnas y se consideran en el cálculo las ventas de todos los municipios.

ALLEXCEPT

Cómo las dos funciones anteriores que hemos visto pertenecientes a la familia ALL*, ALLEXCEPT puede ofrecer dos comportamientos diferentes, uno como función de tabla y otro como modificador de CALCULATE, nuevamente lo que nos interesa abordar en este capítulo es el segundo comportamiento,

el de modificador de CALCULATE. Básicamente ALLEXCEPT se usa principalmente junto con CALCULATE para eliminar todos los filtros de una tabla, excepto los de algunas columnas.

Veámoslo con un ejemplo, supongamos que nos solicitan el % del total de ventas que cada mes representa en el año. Para calcular esto necesitamos la métrica Total Ventas dada por los meses del año dividido entre la misma métrica en un contexto de filtro diferente donde eliminamos todos los filtros de la tabla Calendario, excepto los de la columna Año. De esta forma obtendremos en el denominador el total de ventas para cada año. Una posible implementación de dicha métrica sería la siguiente:



```
Total Ventas Mes % =  
DIVIDE ( [Total Ventas],  
CALCULATE ( [Total Ventas], ALLEXCEPT ( Calendario, Calendario[Año] ) )  
)
```

Y en una matriz obtendríamos el siguiente resultado:

Año	Total Ventas	Ventas Año	Total Ventas Mes %
2017	\$21.662.728.028	\$21.662.728.028	100,00 %
Abril	\$1.906.689.262	\$21.662.728.028	8,80 %
Agosto	\$2.475.851.186	\$21.662.728.028	11,43 %
Diciembre	\$1.980.040.968	\$21.662.728.028	9,14 %
Enero	\$1.025.054.850	\$21.662.728.028	4,73 %
Febrero	\$1.422.899.788	\$21.662.728.028	6,57 %
Julio	\$1.617.294.343	\$21.662.728.028	7,47 %
Junio	\$1.833.686.293	\$21.662.728.028	8,46 %
Marzo	\$1.702.922.633	\$21.662.728.028	7,86 %
Mayo	\$1.737.880.439	\$21.662.728.028	8,02 %
Noviembre	\$2.020.198.624	\$21.662.728.028	9,33 %
Octubre	\$1.830.178.290	\$21.662.728.028	8,45 %
Septiembre	\$2.110.031.352	\$21.662.728.028	9,74 %
2018	\$24.484.312.974	\$24.484.312.974	100,00 %
Abril	\$2.062.940.986	\$24.484.312.974	8,43 %
Agosto	\$2.455.011.214	\$24.484.312.974	10,03 %
Diciembre	\$2.146.891.479	\$24.484.312.974	8,77 %
Enero	\$1.572.862.712	\$24.484.312.974	6,42 %
Febrero	\$1.886.277.688	\$24.484.312.974	7,70 %
Julio	\$2.112.506.352	\$24.484.312.974	8,63 %
Junio	\$1.969.848.554	\$24.484.312.974	8,05 %
Marzo	\$2.069.384.595	\$24.484.312.974	8,45 %
Mayo	\$2.060.006.221	\$24.484.312.974	8,41 %
Noviembre	\$1.980.032.506	\$24.484.312.974	8,09 %
Octubre	\$2.043.894.060	\$24.484.312.974	8,35 %
Septiembre	\$2.124.656.607	\$24.484.312.974	8,68 %
2019	\$24.684.311.343	\$24.684.311.343	100,00 %
Abril	\$2.100.900.104	\$24.684.311.343	8,51 %
Agosto	\$2.266.347.628	\$24.684.311.343	9,18 %
Diciembre	\$1.967.434.591	\$24.684.311.343	7,97 %
Enero	\$1.607.238.862	\$24.684.311.343	6,51 %
Febrero	\$2.044.402.923	\$24.684.311.343	8,28 %
Julio	\$1.985.671.439	\$24.684.311.343	8,04 %
Junio	\$2.025.726.818	\$24.684.311.343	8,21 %
Marzo	\$2.232.480.889	\$24.684.311.343	9,04 %
Total	\$96.777.528.715	\$96.777.528.715	100,00 %

Figura 10.14 Se muestran los valores de Total Ventas Mes % distribuidos para cada año.

Como vemos en la Figura 10.14 obtenemos el cálculo del porcentaje que cada mes representa en ventas para cada año. La métrica Ventas Año simplemente está allí para darnos una idea de cómo funciona el denominador de la fórmula para cada uno de los valores de la fila, como observamos se está eliminando

el contexto de filtro del mes y conservando el del año, por esta razón para cada mes del año visualizamos el mismo total de ventas.

REMOVEFILTERS

REMOVEFILTERS simplemente es un alias de la función ALL cuando esta se usa como argumento de filtro dentro de un CALCULATE, es decir, cuando se usa como un modificador de CALCULATE, por esta razón no profundizaremos mucho en ella, simplemente es una forma de leer más fácilmente una expresión, recordemos que ALL como argumento de filtro de CALCULATE elimina o remueve todos los contextos de filtro, por lo tanto usar REMOVEFILTERS simplemente hace que la lectura del código se haga un poco más sencilla de leer y entender.

Veamos un ejemplo sencillo, con base al ejemplo anterior utilizado para ALLEXCEPT.

Replicaremos el mismo comportamiento de la métrica Ventas Año, pero esta vez utilizando la función REMOVEFILTERS por lo tanto tendremos la siguiente fórmula:



Ventas Año =

CALCULATE ([Total Ventas], REMOVEFILTERS (Calendario[Mes]))

Año	Total Ventas	Ventas Año
2017	\$21.662.728.028	\$21.662.728.028
	Abril	\$1.906.689.262
	Agosto	\$2.475.851.186
	Diciembre	\$1.980.040.968
	Enero	\$1.025.054.850
	Febrero	\$1.422.899.788
	Julio	\$1.617.294.343
	Junio	\$1.833.686.293
	Marzo	\$1.702.922.633
	Mayo	\$1.737.880.439
	Noviembre	\$2.020.198.624
	Octubre	\$1.830.178.290
2018	\$24.484.312.974	\$24.484.312.974
	Abril	\$2.062.940.986
	Agosto	\$2.455.011.214
	Diciembre	\$2.146.891.479
	Enero	\$1.572.862.712
	Febrero	\$1.886.277.688
	Julio	\$2.112.506.352
	Junio	\$1.969.848.554
	Marzo	\$2.069.384.595
	Mayo	\$2.060.006.221
	Noviembre	\$1.980.032.506
	Octubre	\$2.043.894.060
	Septiembre	\$2.124.656.607

Figura 10.15 Se muestran los valores Ventas Año con el filtro de Mes removido.

Como vemos simplemente tenemos para cada mes el valor total del año, es decir, con REMOVEFILTER, removimos el filtro de mes del contexto de filtro.

KEEPFILTERS

Hasta ahora hemos visto que los argumentos de filtro de CALCULATE sobrescriben cualquier filtro previamente existente sobre una columna. Sin embargo, KEEPFILTERS puede conservar algún filtro existente que deseemos. Por ejemplo, si deseáramos mostrar el total de ventas de la línea de productos "CAFÉ" cuando esta esté presente en el contexto de filtro y un blanco cuando no esté presente, podríamos utilizar una métrica como la siguiente:



Ventas CAFÉ Keepfilter =
CALCULATE ([Total Ventas], KEEPFILTERS (Linea[Linea] = "CAFÉ"))

Y el resultado sería el siguiente:

Linea	Total Ventas	Ventas CAFÉ Keepfilter
ALIMENTO	\$21.653.792.830	
ASEO	\$15.775.403.827	
BEBIDAS	\$32.452.118.734	
CAFÉ	\$13.410.774.469	\$13.410.774.469
CONDIMENTOS	\$13.485.438.855	
Total	\$96.777.528.715	\$13.410.774.469

Figura 10.16 Se muestran valores sólo para la línea de producto CAFÉ y para el total general.

KEEPFILTERS altera la manera en que CALCULATE aplica un filtro al nuevo contexto de filtro. En lugar de sobrescribir un filtro existente en la misma columna, KEEPFILTERS agrega el nuevo filtro a los existentes. Por lo tanto, sólo las celdas donde la categoría filtrada ya estaba incluida en el contexto del filtro producirán un resultado visible.

Veamos una comparación del comportamiento de CALCULATE con y sin KEEPFILTERS y para esto utilizaremos la siguiente métrica para comparar:



Ventas CAFÉ =
CALCULATE ([Total Ventas], Linea[Linea] = "CAFÉ")

Y el resultado de la comparación en una matriz sería el siguiente:

Linea	Total Ventas	Ventas CAFÉ Keepfilter	Ventas CAFÉ
ALIMENTO	\$21.653.792.830		\$13.410.774.469
ASEO	\$15.775.403.827		\$13.410.774.469
BEBIDAS	\$32.452.118.734		\$13.410.774.469
CAFÉ	\$13.410.774.469	\$13.410.774.469	\$13.410.774.469
CONDIMENTOS	\$13.485.438.855		\$13.410.774.469
Total	\$96.777.528.715	\$13.410.774.469	\$13.410.774.469

Figura 10.17 Ventas CAFÉ, KEEPFILTERS muestra valores sólo para la línea de producto CAFÉ y para el total general mientras que Ventas CAFÉ, para todas las filas.

Mientras que CALCULATE sin KEEPFILTER sobrescribe los filtros existentes en las columnas donde se aplica un nuevo filtro y las demás columnas restantes del contexto de filtro se dejan intactas, con KEEPFILTER hace exactamente lo que su nombre implica, conservar el filtro. En lugar de sobrescribir el filtro existente, mantiene el filtro y agrega el nuevo filtro al contexto del filtro.

Dado que KEEPFILTERS evita la sobreescritura, el nuevo filtro generado por el argumento de filtro de CALCULATE se agrega al contexto. Si nos centramos por ejemplo en la fila ALIMENTO y en la métrica Ventas CAFÉ Keepfilter, allí el contexto de filtro resultante estaría teniendo dos filtros, uno filtra ALIMENTO y el otro CAFÉ, por que nuevamente, KEEPFILTERS evita la sobreescritura, por lo tanto, conserva el filtro CAFÉ y añade el de ALIMENTO. La intersección de las dos condiciones da como resultado un conjunto vacío, que produce un resultado en blanco.

Linea	Total Ventas	Ventas CAFÉ Keepfilter	
ALIMENTO	\$21.653.792.830		ALIMENTO + CAFÉ = Vacío
ASEO	\$15.775.403.827		ASEO + CAFÉ = Vacío
BEBIDAS	\$32.452.118.734		BEBIDAS + CAFÉ = Vacío
CAFÉ	\$13.410.774.469	\$13.410.774.469	
CONDIMENTOS	\$13.485.438.855		CONDIMENTOS + CAFÉ = Vacío
Total	\$96.777.528.715	\$13.410.774.469	

Figura 10.18 La intersección de los dos filtros da como resultado un valor en blanco

Veamos un ejemplo más claro con varios elementos seleccionados en una columna. Utilizaremos las siguientes métricas:



Alimento y Café =
`CALCULATE ([Total Ventas], Linea[Linea] IN { "ALIMENTO", "CAFÉ" })`



Alimento y Café Keepfilter =
`CALCULATE ([Total Ventas],
KEEPFILTERS (Linea[Linea] IN { "ALIMENTO", "CAFÉ" })
)`

Y el resultado de la comparación sería el siguiente:

Linea	Total Ventas	Alimento y Café	Alimento y Café Keepfilter
ALIMENTO	\$21.653.792.830	\$35.064.567.299	\$21.653.792.830
ASEO	\$15.775.403.827	\$35.064.567.299	
BEBIDAS	\$32.452.118.734	\$35.064.567.299	
CAFÉ	\$13.410.774.469	\$35.064.567.299	\$13.410.774.469
CONDIMENTOS	\$13.485.438.855	\$35.064.567.299	
Total	\$96.777.528.715	\$35.064.567.299	\$35.064.567.299

Figura 10.19 Utilizando KEEPFILTERS, el contexto de filtro original se fusiona con el nuevo.

En la Figura 10.19 vemos que la versión de la métrica con KEEPFILTERS solo calcula los valores del total de las ventas para ALIMENTO y CAFÉ, dejando todas las demás categorías en blanco. La fila Total solo tiene en cuenta nuevamente ALIMENTO y CAFÉ.

Capítulo 11: CALCULATETABLE vs FILTER

En el capítulo anterior aprendimos muchos de los elementos significativos de la función CALCULATE y algunos de sus modificadores más importantes, en este capítulo nos centraremos en la función CALCULATETABLE y en como esta devuelve el mismo resultado que una función FILTER con algunas particularidades específicas entre ambas funciones. Veremos algunas aplicaciones de la función FILTER, pero desde la utilización en compañía de CALCULATETABLE. Las mismas consideraciones que explicamos para la función CALCULATE en el capítulo 10: CALCULATE y funciones de filtro, son aplicables y válidas para CALCULATETABLE.

Diferencias entre CALCULATETABLE y FILTER

Para poder explicar las diferencias entre la función de CALCULATETABLE y FILTER primero definamos algunos conceptos importantes de la función CALCULATETABLE.

CALCULATETABLE

CALCULATETABLE funciona de la misma manera que CALCULATE. La única diferencia está en el tipo de resultado, CALCULATE devuelve un valor escalar, mientras que CALCULATETABLE evalúa una expresión de tabla que devuelve una tabla como resultado.

CALCULATETABLE también tiene la misma sintaxis de CALCULATE, el primer parámetro es obligatorio y es la expresión para evaluar, la cual debe ser una tabla del modelo o una función que retorne una tabla. Los demás parámetros son los argumentos de filtro los cuales pueden ser opcionales y ser la cantidad que necesitemos.

Cómo usar CALCULATETABLE

Como CALCULATETABLE devuelve una tabla como resultado, inicialmente vamos a ver un caso práctico de su utilización. Vamos a crear una nueva tabla en Power BI dando clic en la opción *Modelado* del menú principal y posteriormente seleccionaremos la opción *Nueva Tabla*.



Figura 11.1 Crear nueva tabla

En la barra de fórmulas utilizaremos la siguiente fórmula para generar una nueva tabla por medio de la función CALCULATETABLE:



```
Productos alto costo =
CALCULATETABLE (
    'Maestro Productos',
    'Maestro Productos'[Costo Estándar] > 15000
)
```

Utilizaremos esta función para generar una nueva tabla que contenga solamente los productos que tengan un costo superior a los 15.000, el resultado de la tabla es el mostrado en la figura 11.2.

La captura de pantalla muestra la barra de fórmulas de Power BI con la siguiente expresión DAX:

```
1 Productos alto costo =
2 CALCULATETABLE (
3     'Maestro Productos',
4     'Maestro Productos'[Costo Estándar] > 15000
5 )
6
```

Abajo de la barra de fórmulas se muestra la tabla resultante:

Cod Producto	Nombre Producto	Costo Estándar	Cantidad de productos vendidos	Fecha último pedido
8928	CORN FLAKES CEREAL INTEGRAL x 800 gr	20110	181698	22/12/2020 12:00:00 a. m.
1738	NESTUM 5 CEREALES x 200 gr	19096	86193	25/12/2020 12:00:00 a. m.
2303	QUIPITOS SOBRE x 24 x 8gr	15215	218820	10/12/2020 12:00:00 a. m.
3607	BIOEXPERT CHAMPU CAP.GUSANO SEDAx18x15ml	31473	89801	23/12/2020 12:00:00 a. m.
7018	NUTRIBELA REPOLARIZACION x27 mlx12	24986	51184	15/12/2020 12:00:00 a. m.
4086	ELITE SERVILLETA PRACTICA x 320	59703	5548	15/12/2020 12:00:00 a. m.
1677	MILO RISTRAS x 16 UNIDS. x 25 gr.	30722	68858	19/12/2020 12:00:00 a. m.
6202	SUNTEA DURAZNO x 4.5 LITROS x 266 gr.	20658	301618	13/12/2020 12:00:00 a. m.
6905	SAVIOLE AGUA DE ALOE x 270 ml	34023	241319	26/12/2020 12:00:00 a. m.
4307	FRUTTSI FRUTOS ROJOS PET x 400 ml	28830	45983	15/12/2020 12:00:00 a. m.
6264	NESCAFE TRADICION FRASCO x50 gr.	22723	32504	22/12/2020 12:00:00 a. m.
9658	CALDO GALLINA x 2.97 kgr X 270 CUBOS	16931	104187	21/12/2020 12:00:00 a. m.
7605	CALDO RICOSTILLA CUBOS x 60x660gr	20762	254518	25/12/2020 12:00:00 a. m.

Figura 11.2 Tabla con productos cuyo costo es superior a 15.000.

Como vemos en la Figura 11.2 DAX creó una nueva tabla que existe ahora en el modelo y que contiene únicamente los productos cuyo costo estándar es superior a los 15.000. Sin embargo, este no es el único uso que tiene CALCULATETABLE, también podemos utilizar CALCULATETABLE como argumento de filtro dentro de la función CALCULATE.

Veamos un ejemplo práctico para ilustrar esto. Supongamos que ahora queremos saber cuáles fueron las ventas totales de estos productos que tienen un alto costo, sin necesidad de crear una tabla en el modelo, para esto podemos utilizar la misma expresión dentro de una función CALCULATE para crear una nueva métrica que me devuelva las ventas totales de estos productos, para esto utilizaremos entonces la siguiente fórmula:



```
Ventas productos alto costo =  
CALCULATE (  
    Ventas[Total Ventas],  
    CALCULATETABLE (  
        'Maestro Productos',  
        'Maestro Productos'[Costo Estándar] > 15000  
    )  
)
```

El resultado de esta métrica si lo visualizamos dentro de una matriz y cruzamos con la columna Maestro productos [Nombre producto] sería el siguiente:

Nombre Producto	Ventas productos alto costo
BIOEXPERT CHAMPU CAP.GUSANO SEDAx18x15ml	\$4.256.483.181
CALDO GALLINA x 2.97 kgr X 270 CUBOS	\$2.692.701.294
CALDO RICOSTILLA CUBOS x 60x660gr	\$7.259.735.576
CORN FLAKES CEREAL INTEGRAL x 800 gr	\$4.548.499.194
ELITE SERVILLETA PRACTICA x 320	\$914.725.625
FRUTTSI FRUTOS ROJOS PET x 400 ml	\$2.178.092.924
MILO RISTRAS x 16 UNIDS. x 25 gr.	\$3.706.246.914
NESCAFE TRADICION FRASCO x50 gr.	\$1.397.865.463
NESTUM 5 CEREALES x 200 gr	\$2.029.910.962
NUTRIBELA REPOLARIZACION x27 mlx12	\$2.135.553.681
QUIPITOS SOBRE x 24 x 8gr	\$4.884.741.089
SAVILOE AGUA DE ALOE x 270 ml	\$11.953.022.626
SUNTEA DURAZNO x 4.5 LITROS x 266 gr.	\$7.979.938.469
Total	\$55.937.516.998

Figura 11.3 Matriz con las ventas totales de los productos cuyo costo es superior a 15.000.

Como vemos en la Figura 11.3 la métrica *Ventas productos alto costo* está calculando el total de ventas únicamente para los productos que tienen un costo estándar superior a los 15.000, esto significa que la función CALCULATETABLE dentro del argumento de filtro de la función CALCULATE para la expresión a evaluar de la métrica *Total Ventas*, creó un nuevo contexto de filtro en el que se calculan únicamente los productos que le especificamos, y para esto no fue necesario dejar creada la tabla que genera CALCULATETABLE dentro del modelo.

Cuál es el propósito de CALCULATETABLE

Podemos usar CALCULATETABLE siempre que deseemos aplicar un filtro a una columna dentro del modelo y siempre que exista un requisito para la transición de contexto o de los modificadores del contexto de filtro.

Significa entonces que CALCULATETABLE se puede aplicar a una columna que esté presente en el modelo de datos. No se puede aplicar a una medida como la función FILTER. Como es posible usar

diferentes modificadores de contexto de filtro como CROSSFILTER, ALL, ALLEXCEPT entre otros, esto ayuda a realizar la transición de contexto. Por lo tanto, la función CALCULATETABLE se vuelve más poderosa que FILTER.

Según el escenario que se requiera, tendremos que decidir que función utilizar si CALCULATETABLE o FILTER.

Diferencias entre CALCULATETABLE y FILTER

En DAX las dos funciones que más se suelen mezclar conceptualmente son CALCULATETABLE y FILTER. Esto se debe al hecho de que en realidad producen el mismo resultado en ciertos escenarios. Por ejemplo, cuando se crea una matriz en la que se aplica un filtro a una tabla y el filtro en la fórmula DAX se aplica a otra tabla. Por ejemplo, consideremos las siguientes fórmulas:



Conteo productos CAFÉ CALCULATETABLE =
`COUNTROWS (CALCULATETABLE (Ventas, Linea[Linea] = "CAFÉ"))`



Conteo productos CAFÉ FILTER =
`COUNTROWS (FILTER (Ventas, RELATED ('Linea'[Linea]) = "CAFÉ"))`

La sintaxis es similar, y los resultados también, veámoslos en una matriz:

Año	Conteo productos CAFÉ CALCULATETABLE	Conteo productos CAFÉ FILTER
2017	2708	2708
2018	2324	2324
2019	2814	2814
2020	2723	2723
Total	10569	10569

Figura 11.4 Matriz con la cantidad de productos vendidos de la línea de Café con CALCULATETABLE y FILTER

A pesar de que los resultados son iguales, el motor evalúa estas dos fórmulas de manera diferente la que tiene el FILTER se ejecuta en un contexto de fila anidado proporcionado por el mismo FILTER dentro de un contexto de filtro generado por la medida. No obstante, la fórmula que posee el CALCULATETABLE se agrega al contexto de filtro inicial que se completa sólo con el nombre del año que proviene de la columna año.

Veamos otro ejemplo para apreciar mejor las diferencias, revisemos las siguientes dos fórmulas:



Conteo Ventas Julio CALCULATETABLE =
COUNTRWS (CALCULATETABLE (Ventas, Calendario[Mes] = "Julio"))



Conteo Ventas Julio FILTER =
COUNTRWS (FILTER (Ventas, RELATED (Calendario[Mes]) = "Julio"))

Si las utilizamos dentro de una matriz cruzando por el nombre del mes tendríamos el siguiente resultado:

Mes	Conteo Ventas Julio FILTER	Conteo Ventas Julio CALCULATETABLE
Abril		5012
Agosto		5012
Diciembre		5012
Enero		5012
Febrero		5012
Julio	5012	5012
Junio		5012
Marzo		5012
Mayo		5012
Noviembre		5012
Octubre		5012
Septiembre		5012
Total	5012	5012

Figura 11.5 Matriz con la cantidad de ventas del mes de Julio con CALCULATETABLE y FILTER

Como vemos en la Figura 11.5 la métrica con FILTER simplemente filtra la tabla de ventas para el mes de julio, si el contexto de filtro es evaluado, por ejemplo, para el mes de abril, se aplica un filtro para abril y se adiciona el de julio, la intersección de los dos filtros da como resultado un valor en blanco. Por otro lado, para la métrica con CALCULATETABLE se sobrescribe el nombre del mes que proviene del contexto de filtro que en este caso es la columna Mes, por lo que siempre se devolverá el valor para el mes de Julio en todas las filas independientemente del mes.

Analicemos ahora una diferencia más profunda entre ambas funciones. Para este caso analicemos las dos nuevas fórmulas siguientes:



Conteo ventas mes anterior CALCULATETABLE =
COUNTRWS (CALCULATETABLE (Ventas, PREVIOUSMONTH (Calendario[Date])))



Conteo ventas mes anterior FILTER =
COUNTRROWS (FILTER (Ventas, PREVIOUSMONTH (Calendario[Date])))

Primero que todo la fórmula con FILTER no puede sobrescribir el contexto de filtro inicial, por lo que intentar visualizarlo en la matriz daría un error. Sin embargo, para la métrica con CALCULATETABLE tendríamos el siguiente resultado en la matriz:

Año	Conteo Ventas	Conteo ventas mes anterior CALCULATETABLE
2017	15152	
Enero	870	
Febrero	1163	870
Marzo	1271	1163
Abril	1252	1271
Mayo	1448	1252
Junio	1209	1448
Julio	1129	1209
Agosto	1460	1129
Septiembre	1389	1460
Octubre	1253	1389
Noviembre	1332	1253
Diciembre	1376	1332
Total	15152	

Figura 11.6 Matriz con la cantidad de ventas del mes anterior con CALCULATETABLE para el año 2017.

Dado que para esta fórmula estamos utilizando CALCULATETABLE el contexto de filtro inicial se puede sobrescribir con lo que se especifica en el argumento de filtro, es decir, PREVIOUSMONTH (Calendario [Date]). En esta fórmula, DAX hace que el nombre del mes en el contexto de filtro inicial se sobre escriba con el mes anterior.

La última diferencia, sin embargo, no la menos importante, es el hecho de que CALCULATETABLE, como CALCULATE, provocan una transición de contexto. Por lo tanto, si una fórmula se evalúa inicialmente dentro del contexto de fila, ese contexto de fila se convertirá en un contexto de filtro. Lo que ya varias veces hemos mencionado que se conoce con el nombre de transición de contexto. Para mostrar este comportamiento, las columnas calculadas funcionan bien porque el contexto de evaluación de una columna calculada es el contexto de fila. Analicemos las dos siguientes fórmulas que utilizaremos para crear dos nuevas columnas calculadas en la tabla Vendedores:



Conteo ventas CAFÉ CALCULATETABLE =
COUNTRROWS (CALCULATETABLE (Ventas, Linea[Linea] = "CAFÉ"))



Conteo ventas CAFÉ FILTER =
COUNTRROWS (FILTER (Ventas, RELATED (Linea[Linea]) = "CAFÉ"))

El resultado directamente en la tabla de Vendedores se ve de la siguiente forma:

Nombre Vendedores	Codigo	Conteo ventas CAFÉ FILTER	Conteo ventas CAFÉ CALCULATETABLE
JUAN GUILLERMO RICAURTE	B-01	10569	7815
ALBERT VELASQUEZ	B-02	10569	961
VANESSA QUINTERO	B-03	10569	851
MIGUEL ANGEL RIOS	B-04	10569	683
SIMON TERRANOVA	B-05	10569	259

Figura 11.7 Resultados de las columnas calculadas con FILTER y CALCULATETABLE en la tabla vendedores

La métrica con FILTER no realiza ninguna transición de contexto, el contexto de evaluación sigue siendo el contexto de fila. El contexto de fila no filtra, por lo que sólo obtenemos un recuento de todas las ventas de la línea de productos *CAFÉ* para todos los vendedores.

No obstante, la métrica con CALCULATETABLE, realizará la transición de contexto. Entonces, primero se aplica un filtro al vendedor en el modelo de datos y luego se realiza la evaluación de la fórmula. Por lo tanto, esto da como resultado el recuento de las ventas de la línea de producto *CAFÉ* para cada vendedor.

Por estas razones es importante poder identificar que según el escenario que se nos presente, decidir que función utilizar de acuerdo con el resultado que necesitemos, si CALCULATETABLE o FILTER.

Capítulo 12: Inteligencia de tiempo

En la presentación de cualquier tablero o informe en Power BI casi siempre debe existir uno de esos indicadores que compara o mide si un resultado es bueno o no, es la única forma de saber cómo estamos frente a un dato y DAX es en lo que se especializa, crear métricas que permitan emitir información de alto valor para la toma de decisiones y rastreo de indicadores.

En DAX contamos con un alto portafolio de funciones que nos permiten crear este tipo de análisis como, por ejemplo, hacer un comparativo de enero 2022 con enero 2021, o un acumulado a cierta fecha, o crear medidas móviles y múltiples combinaciones posibles de funciones que nos lleva al siguiente nivel.

Todo modelo que requiera de inteligencia de tiempo es de carácter obligatorio contar con una tabla calendario, ya que con esta podemos relacionar tablas transaccionales entre ellas, ordenar períodos de manera cronológicos y sacar provecho a todo este set de funciones de tiempo de manera fácil.

Con estas funciones que vamos a estudiar a continuación solucionamos la mayoría de los casos en las consultorías que hacemos como empresa, en todo requerimiento que levantamos están allí incluidas para salir a flote y mostrar resultados sorprendentes.

No se olvide del PODER DE LO SIMPLE, nuestro lema como empresa, ya que las funciones de tiempo que vamos a usar ya están prediseñadas en DAX, no tiene necesidad a hoy de saber cómo es que se calcularon, con el tiempo, a medida que avances en este mundo vas a poder crearlas todas de manera nativa, lo más importante de cualquier modelo BI es su análisis y la capacidad decisiva que usted posee al ver cada resultado, un ejemplo que aprendimos es cuando inicias clases de conducción de un vehículo, en el momento solo quieras aprender su manejo, con el tiempo y pasando horas en él, aprendes más sobre vehículos, cambios de aceite, temperaturas, baterías y demás, pero en tu primer día de viaje solo quieras que te lleve a tu destino y así es DAX.

Antes de iniciar, no te olvides de lo aplicado y recomendado hasta este punto, iniciemos creando una tabla para estas medidas de tiempo y allí las arrojaremos todas.

Si es posible volver a leer el capítulo 9 y 10, para entender un poco más sobre los contextos y los modificadores de CALCULATE.

Variaciones año anterior

SAMEPERIODLASTYEAR

Creo que te has dado cuenta de que DAX no trabaja de igual forma que las funciones bases de Excel, aunque comparten cierta lógica y mucho parecido en alguna de las funciones, los contextos es su gran diferencial, algo que podemos hacer en Excel es hacer referencia a una celda en específico para calcular o comparar entre periodos, en DAX no, debemos siempre tener en cuenta los contextos y hacemos referencia es a tablas y columnas completas, por ejemplo, en el lienzo insertamos una matriz que contenga el año y mes, analizado por la medida Ventas.

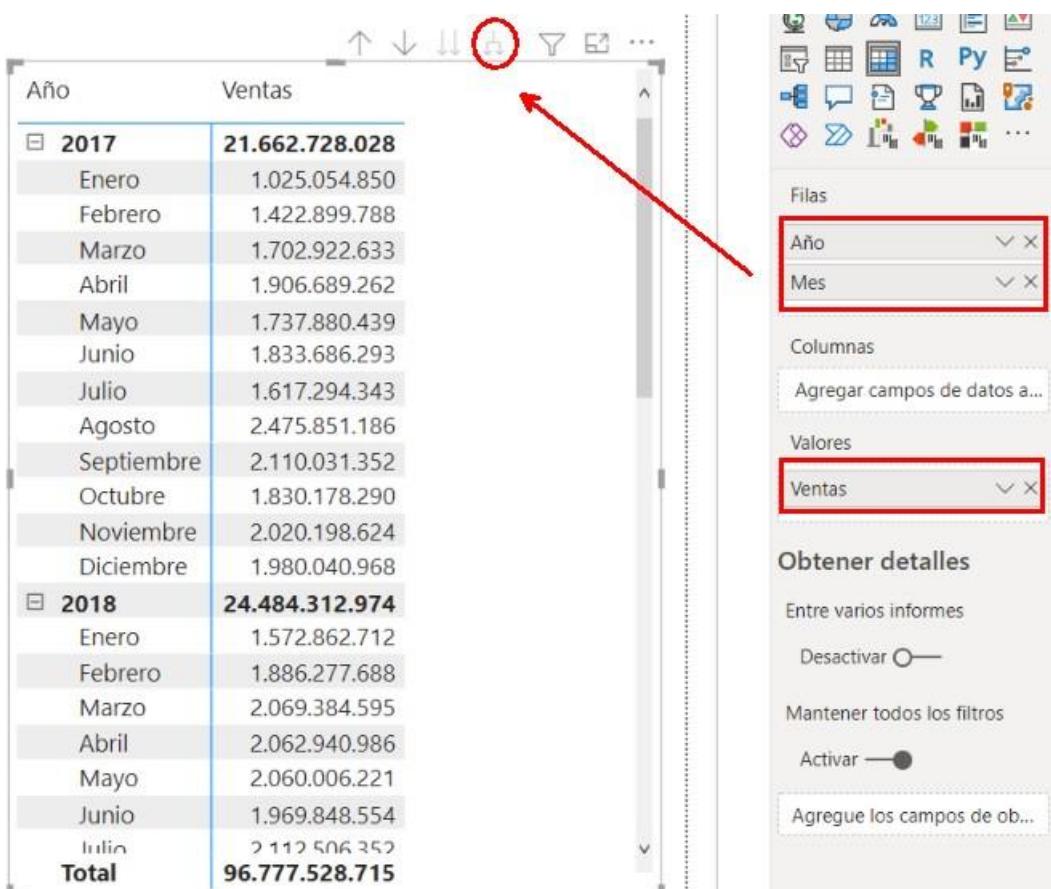


Figura 12.1 Ventas año y mes.

Si queremos calcular la variación de enero 2018 con enero 2017 en DAX no es posible hacerlo directamente o decirle que el valor se desplace un año atrás sino es con funciones que ya están creadas por Microsoft o de manera nativas.

SAMEPERIODLASTYEAR permite hacer este desplazamiento de año y queda dinámico, es decir, compara no solo enero 2018 con enero 2017, sino que también lo hace con cada uno de los meses.

SAMEPERIODLASTYEAR tiene un único argumento y es la columna fecha de la tabla *Calendario* y debe ir obligadamente acompañada de CALCULATE, ya que en resumidas cuentas esto cambia el contexto.

El objetivo es primero llevar el valor de enero 2017 (1.025.054.850) al lado de enero 2018 para poder crear las variaciones absolutas y relativas, siempre se debe pensar así en DAX, leer bien los contextos y con una pequeña muestra de los datos analizamos el problema.

Ver figura 12.2

Año	Ventas
2017	21.662.728.028
Enero	1.025.054.850
Febrero	1.422.899.788
Marzo	1.702.922.633
Abril	1.906.689.262
Mayo	1.737.880.439
Junio	1.833.686.293
Julio	1.617.294.343
Agosto	2.475.851.186
Septiembre	2.110.031.352
Octubre	1.830.178.290
Noviembre	2.020.198.624
Diciembre	1.980.040.968
2018	24.484.312.974
Enero	1.572.862.712
Febrero	1.886.277.688
Marzo	2.069.384.595

Figura 12.2 Proceso desplazar valor.

Para hacer este desplazamiento debemos acudir a nuestra primera función que enumera las funciones de inteligencia de tiempo.

Creamos una nueva medida en la tabla *Medidas Tiempo* y la llevamos a la matriz.

. Año Anterior = CALCULATE([Ventas], SAMEPERIODLASTYEAR(Calendar[Date]))

Año	Ventas	Año Anterior
2017	21.662.728.028	
Enero	1.025.054.850	
Febrero	1.422.899.788	
Marzo	1.702.922.633	
Abril	1.906.689.262	
Mayo	1.737.880.439	
Junio	1.833.686.293	
Julio	1.617.294.343	
Agosto	2.475.851.186	
Septiembre	2.110.031.352	
Octubre	1.830.178.290	
Noviembre	2.020.198.624	
Diciembre	1.980.040.968	
2018	24.484.312.974	21.662.728.028
Enero	1.572.862.712	1.025.054.850
Febrero	1.886.277.688	1.422.899.788
Marzo	2.069.384.595	1.702.922.633
Abril	2.062.940.986	1.906.689.262

Figura 12.3 Valor desplazado.

Solo con un argumento y la ayuda de CALCULATE, SAMEPERIODLASTYEAR desplazó el valor de cada mes un año atrás.

Esta es la manera correcta siempre de pensar en DAX, contextos, fue el trabajo más difícil que tuvimos que realizar, los otros cálculos son restas y divisiones.

Las celdas en blanco son porque 2017 no tiene valor contra que compararse.

Variación absoluta

Es la diferencia de un valor a otro, en este caso sería restar las medidas [Ventas] menos la medida [Año Anterior]. Nueva medida → Variación \$ → Llevamos a la matriz

Variación \$ = [Ventas] - [Año Anterior]

Año	Ventas	Año Anterior	Variación \$
2017	21.662.728.028		21.662.728.028
Enero	1.025.054.850		1.025.054.850
Febrero	1.422.899.788		1.422.899.788
Marzo	1.702.922.633		1.702.922.633
Abril	1.906.689.262		1.906.689.262
Mayo	1.737.880.439		1.737.880.439
Junio	1.833.686.293		1.833.686.293
Julio	1.617.294.343		1.617.294.343
Agosto	2.475.851.186		2.475.851.186
Septiembre	2.110.031.352		2.110.031.352
Octubre	1.830.178.290		1.830.178.290
Noviembre	2.020.198.624		2.020.198.624
Diciembre	1.980.040.968		1.980.040.968
2018	24.484.312.974	21.662.728.028	2.821.584.946
Enero	1.572.862.712	1.025.054.850	547.807.862
Febrero	1.886.277.688	1.422.899.788	463.377.900
Marzo	2.069.384.595	1.702.922.633	366.461.962

Figura 12.4 Variación absoluta.

El cálculo para cada mes de 2018 es correcto, pero en 2017 trae el mismo valor, ya que no hay valor para 2016, año anterior. Podemos usar un condicional, si Año Anterior este vacío devuelva blanco, sino realice el cálculo.



Variación \$ =
IF ([Año Anterior] = BLANK (), BLANK (), [Ventas] - [Año Anterior])

2017	21.662.728.028		
Enero	1.025.054.850		
Febrero	1.422.899.788		
Marzo	1.702.922.633		
Abril	1.906.689.262		
Mayo	1.737.880.439		
Junio	1.833.686.293		
Julio	1.617.294.343		
Agosto	2.475.851.186		
Septiembre	2.110.031.352		
Octubre	1.830.178.290		
Noviembre	2.020.198.624		
Diciembre	1.980.040.968		
2018	24.484.312.974	21.662.728.028	2.821.584.946
Enero	1.572.862.712	1.025.054.850	547.807.862
Febrero	1.886.277.688	1.422.899.788	463.377.900
Marzo	2.069.384.595	1.702.922.633	366.461.962

Figura 12.5 Variación absoluta validando solo datos para comparar.

Variación relativa

Es el cambio de un valor a otro expresado en porcentaje, la fórmula es tomar el valor actual (Ventas) y dividirlo con el periodo anterior (Año Anterior) y restarle 1, si el resultado es positivo, significa que hubo crecimiento, de lo contrario, decrecimiento.

Variación % = DIVIDE([Ventas], [Año Anterior])-1				
Año	Ventas	Año Anterior	Variación \$	Variación %
2017	21.662.728.028			-100,00 %
	Enero	1.025.054.850		-100,00 %
	Febrero	1.422.899.788		-100,00 %
	Marzo	1.702.922.633		-100,00 %
	Abril	1.906.689.262		-100,00 %
	Mayo	1.737.880.439		-100,00 %
	Junio	1.833.686.293		-100,00 %
	Julio	1.617.294.343		-100,00 %
	Agosto	2.475.851.186		-100,00 %
	Septiembre	2.110.031.352		-100,00 %
	Octubre	1.830.178.290		-100,00 %
	Noviembre	2.020.198.624		-100,00 %
2018	Diciembre	1.980.040.968		-100,00 %
		24.484.312.974	21.662.728.028	2.821.584.946
	Enero	1.572.862.712	1.025.054.850	53,44 %
	Febrero	1.886.277.688	1.422.899.788	32,57 %
	Marzo	2.069.384.595	1.702.922.633	21,52 %

Figura 12.6 Variación relativa.

Si volvemos a retomar enero 2018, las ventas crecieron un 53,44% con respecto a enero 2017.

Vemos unos resultados extraños para 2017 ya explicados anteriormente, como no hay datos para 2016 trae -100%, validamos con el mismo condicional explicado en el cálculo anterior.



```
Variación % =  
IF (  
    [Año Anterior] = BLANK (),  
    BLANK (),  
    DIVIDE ( [Ventas], [Año Anterior] ) - 1  
)
```

Luego de tener tus métricas creadas, la imaginación juega un papel muy importante de ahora en adelante para mostrar estos datos en los gráficos correctos que transmitan una historia que contar.

Acumulados

DATESYTD

DATESYTD realiza un acumulado del año a la fecha evaluado en el contexto actual. Esta función es usada especialmente para hacer acumulados de meses, aunque diga que a la fecha puede confundir un poco, ya que es una función que acumula los datos mes a mes.

Continuamos con la matriz anterior, las ventas por año y mes, si queremos ir acumulando los datos mes a mes como lo muestra la siguiente figura.

Año	Ventas
2017	21.662.728.028
Enero	1.025.054.850
Febrero	1.422.899.788
Marzo	1.702.922.633
Abril	1.906.689.262
Mayo	1.737.880.439
Junio	1.833.686.293
Julio	1.617.294.343
Agosto	2.475.851.186

Figura 12.7 Matriz ventas por año y mes.

Si estamos ubicados en abril 2017, esté sería nuestro contexto actual, y desde el 1 de enero de 2017 hasta el 30 de abril de 2017 sería el resultado acumulado y así, si nos ubicamos en el cualquier mes.

DATESYTD tiene dos argumentos, el último es opcional, lo veremos enseguida cómo funciona, el primer argumento es la columna fecha de la tabla *Calendario*.

Recordemos que estas funciones van acompañadas de CALCULATE para poder realizar el desfase y cambiar los contextos.

Insertamos una nueva medida y veamos el resultado.

Acumulado Año = CALCULATE([Ventas],DATESYTD(Calendar[Date]))

Año	Ventas	Acumulado Año
2017	21.662.728.028	21.662.728.028
Enero	1.025.054.850	1.025.054.850
Febrero	1.422.899.788	2.447.954.638
Marzo	1.702.922.633	4.150.877.271
Abril	1.906.689.262	6.057.566.533
Mayo	1.737.880.439	7.795.446.972
Junio	1.833.686.293	9.629.133.265
Julio	1.617.294.343	11.246.427.608
Agosto	2.475.851.186	13.722.278.794
Septiembre	2.110.031.352	15.832.310.146
Octubre	1.830.178.290	17.662.488.436
Noviembre	2.020.198.624	19.682.687.060
Diciembre	1.980.040.968	21.662.728.028
2018	24.484.312.974	24.484.312.974

Figura 12.8 Acumulado año.

Si sumamos manualmente de enero a abril, el resultado debe dar 6.057.566.533.

Notemos que el valor de diciembre 2017 es el mismo valor total de las ventas 2017, y en cada año vuelve y empieza acumular.

Año fiscal

Muchas compañías trabajan con periodos fiscales, es decir, cada empresa define su inicio y fin de año, gracias al segundo argumento de esta función tenemos la posibilidad de tratar estos períodos especiales, simplemente indicándole hasta qué fecha debe acumular, ejemplo, si el año inicia el 1 de julio de cada año, en este argumento definimos el día de terminación o hasta donde debe sumar “30/06”.

Año Fiscal =

CALCULATE ([Ventas], DATESYTD (Calendario[Date], "30/06"))

Año	Ventas	Acumulado Año	Año Fiscal
2017	21.662.728.028	21.662.728.028	12.033.594.763
Enero	1.025.054.850	1.025.054.850	1.025.054.850
Febrero	1.422.899.788	2.447.954.638	2.447.954.638
Marzo	1.702.922.633	4.150.877.271	4.150.877.271
Abril	1.906.689.262	6.057.566.533	6.057.566.533
Mayo	1.737.880.439	7.795.446.972	7.795.446.972
Junio	1.833.686.293	9.629.133.265	9.629.133.265
Julio	1.617.294.343	11.246.427.608	1.617.294.343
Agosto	2.475.851.186	13.722.278.794	4.093.145.529
Septiembre	2.110.031.352	15.832.310.146	6.203.176.881
Octubre	1.830.178.290	17.662.488.436	8.033.355.171
Noviembre	2.020.198.624	19.682.687.060	10.053.553.795
Diciembre	1.980.040.968	21.662.728.028	12.033.594.763
2018	24.484.312.974	24.484.312.974	12.862.992.218

Figura 12.9 Acumulado año fiscal.

Los datos se van acumulando hasta junio y luego en julio se resetea y se acumula de nuevo hasta junio del siguiente año, es poco común este tipo de cálculos, pero conocemos empresas que laboran así.



Año Fiscal =

CALCULATE ([Ventas], DATESYTD (Calendario[Date], "30/06"))

DATESQTD

Realiza el acumulado del trimestre, todas estas funciones devuelven una tabla que son las usados por CALCULATE como filtros, tener cuenta los contextos para no confundir nuestros resultados. Para ver el uso de esta función agreguemos el trimestre de la tabla *Calendario* a la matriz y expandimos.

Año	Ventas
2017	21.662.728.028
T1	4.150.877.271
Enero	1.025.054.850
Febrero	1.422.899.788
Marzo	1.702.922.633
T2	5.478.255.994
Abril	1.906.689.262
Mayo	1.737.880.439
Junio	1.833.686.293
T3	6.203.176.881

Figura 12.10 Matriz con trimestre.

El objetivo de esta función es dentro de cada trimestre ir acumulando los meses según el contexto que muestra la figura 12.10, si estamos ubicados en febrero de 2017, acumula enero y febrero, pero si se está ubicado en mayo, ya el cálculo se reinició y solo acumula abril y mayo, y con esta lógica realiza el cálculo en cada trimestre.

DATESQTD solo tiene un argumento y es la misma columna fecha de la tabla *Calendario* que venimos usando, insertemos esta nueva medida para acumular las ventas por trimestre y la llevamos a la matriz.

Acum Trim = CALCULATE([Ventas],DATESQTD(Calendario[Date]))		
Año	Ventas	Acum Trim
2017	21.662.728.028	5.830.417.882
T1	4.150.877.271	4.150.877.271
Enero	1.025.054.850	1.025.054.850
Febrero	1.422.899.788	2.447.954.638
Marzo	1.702.922.633	4.150.877.271
T2	5.478.255.994	5.478.255.994
Abril	1.906.689.262	1.906.689.262
Mayo	1.737.880.439	3.644.569.701
Junio	1.833.686.293	5.478.255.994
T3	6.203.176.881	6.203.176.881

Figura 12.11 Acumulado trimestre.

Notamos que el acumulado se reinicia cada trimestre, y el total de cada año en esta medida es el valor del último trimestre.



Acum Trim = CALCULATE([Ventas],DATESQTD(Calendario[Date]))

DATESMTD

Genera el acumulado de fechas del mes en el contexto que se evalúe. Es una medida que se usó mucho en pandemia para mirar el comportamiento de la aceleración de los contagios por COVID 19, día a día presentaban esta gráfica acumulada día a día para generar alertas si la curva sí o no se aplanaba. Como es un acumulado la gráfica nunca desciende, siempre al alza o constante.

Podemos insertar una nueva matriz año, mes y día y expandimos.

Año	Ventas
2017	21.662.728.028
Enero	1.025.054.850
4	91.298.138
5	99.104.196
6	20.626.940
7	42.775.941
8	38.874.238
12	10.019.294
13	35.203.680
14	108.572.637
15	51.154.652
18	26.388.720

Figura 12.12 Ventas año, mes y día.

Creamos una medida para ver el acumulado día a día en cada uno de los meses.

Acum Mes = CALCULATE([Ventas],DATESMTD(Calendar[Date]))			
Año	Ventas	Acum Mes	
2017	21.662.728.028	1.980.040.968	
Enero	1.025.054.850	1.025.054.850	
4	91.298.138	91.298.138	
5	99.104.196	190.402.334	
6	20.626.940	211.029.274	
7	42.775.941	253.805.215	
8	38.874.238	292.679.453	
9		292.679.453	
10		292.679.453	
11		292.679.453	

Figura 12.13 Acumulado mes.

Y aunque no se registraron ventas en ciertos días el resultado sigue igual para ellos, ya que va acumulando.



Acum Mes = CALCULATE([Ventas],DATESMTD(Calendar[Date]))

Para crear acumulados DAX también cuenta con otro paquete de funciones como lo son:

- TOTALYTD
- TOTALQTD
- TOTALMTD

Su resultado es completamente el mismo, solo cambia la sintaxis de uso.



Referencia: Podemos estudiar un poco más sobre estas funciones.

<https://docs.microsoft.com/en-us/dax/totalmtd-function-dax>

Comparativos mismo periodo

DATEADD

Esta función nos brinda la posibilidad de desplazar un conjunto de fechas hacia adelante o hacia atrás, ya sea a nivel de año, trimestre, mes o día. Un análisis muy común en los informes es crear variaciones entre el mismo periodo, por ejemplo, mes a mes, enero con febrero, febrero con marzo, 2020 con 2021 y así sucesivamente.

DATEADD se compone de 3 argumento y todos son obligatorios:

1. **Fechas:** La columna fechas de la tabla calendario
2. **Número de intervalos:** el número entero que determina como se va a desplazar las fechas, si es negativo se desplaza hacia atrás, positivo hacia adelante.
3. **Intervalo:** el nivel de periodicidad con el cual se va a mover, si a nivel de año, trimestre, mes o día.

Comparativo mes a mes

Crear un análisis que permita mostrar la variación mes a mes desfasado un periodo anterior, pero antes de iniciar debemos preparar el ambiente, el contexto para que el análisis sea efectivo y transmitido de la mejor manera.

Solo en la matriz dejaremos las ventas por mes y aparte tendremos un segmentador o filtro por año, ya que permitirá mostrar varios escenarios al cambiar de periodo.

Ver figura 12.14

Año	Mes	Ventas
2017	Enero	1.621.693.879
2018	Febrero	2.088.166.224
2019	Marzo	2.140.342.115
2020	Abri	2.100.900.104
	Mayo	2.455.358.065
	Junio	2.044.402.923
	Julio	2.350.283.948
	Agosto	2.070.712.292
	Septiembre	2.187.487.529
	Octubre	2.544.078.418
	Noviembre	2.232.480.889
	Diciembre	2.110.269.984
	Total	25.946.176.370

Figura 12.14 Ventas por mes con filtro por año.

El objetivo es que el valor de enero se mueva y quede en febrero y el de febrero se desfase hacia marzo y así con cada mes para poder comparar un mes con otro. Agreguemos una nueva medida y junto con CALCULATE completamos los argumentos de DATEADD para este resultado.

1 Comparativo Mes =

2 CALCULATE ([Ventas], DATEADD (Calendario[Date], -1, MONTH))

Año	Mes	Ventas	Comparativo Mes
2017	Enero	1.621.693.879	1.967.434.591
2018	Febrero	2.088.166.224	1.621.693.879
2019	Marzo	2.140.342.115	2.088.166.224
2020	Abri	2.100.900.104	2.140.342.115
	Mayo	2.455.358.065	2.100.900.104
	Junio	2.044.402.923	2.455.358.065
	Julio	2.350.283.948	2.044.402.923
	Agosto	2.070.712.292	2.350.283.948
	Septiembre	2.187.487.529	2.070.712.292
	Octubre	2.544.078.418	2.187.487.529
	Noviembre	2.232.480.889	2.544.078.418
	Diciembre	2.110.269.984	2.232.480.889
	Total	25.946.176.370	25.803.340.977

Figura 12.15 Ventas movidas un mes anterior.

Efectivamente el valor se corre un mes antes por el argumento en el intervalo de -1 a nivel de mes.

¿A qué corresponde el valor de enero de la medida Comparativo Mes 1.967.434.591? si analizamos el contexto, el filtro del año es 2020, como esta medida se mueve un periodo antes para este caso, este valor hacer parte de las ventas de diciembre 2019.

Solo queda crear las variaciones absolutas y relativas como lo estudiamos al inicio de este capítulo.



Comparativo Mes =

`CALCULATE ([Ventas], DATEADD (Calendario[Date], -1, MONTH))`

Comparativo año a año

De la misma forma que realicemos el comparativo con el mes anterior, podemos crear la medida para comparar año con año, incluso este análisis es de gran utilidad ya que nos muestra las variaciones de un mes o de varios meses como ha sido el comportamiento en cada uno de los años.

Primero debemos crear una matriz de solo el año y un filtro para el mes.

Mes	Año	Ventas
Enero	2017	21.662.728.028
Febrero	2018	24.484.312.974
Marzo	2019	24.684.311.343
Abril	2020	25.946.176.370
Mayo	Total	96.777.528.715
Junio		
Julio		
Agosto		
Septiembre		
Octubre		
Noviembre		
Diciembre		

Figura 12.16 Matriz por año y filtro por mes.

El objetivo es hacer el desplazamiento de las ventas un año atrás → Insertamos una nueva medida → solo modificamos el intervalo de la medida anterior, que sea a nivel de año.

Comparativo Año = CALCULATE([Ventas], DATEADD(Calendario[Date],-1,YEAR))		
Año	Ventas	Comparativo Año
2017	21.662.728.028	
2018	24.484.312.974	21.662.728.028
2019	24.684.311.343	24.484.312.974
2020	25.946.176.370	24.684.311.343
Total	96.777.528.715	70.831.352.345

Figura 12.17 Ventas desplazadas un año anterior.

La imagen muestra claramente que el desplazamiento se realizó correctamente, solo nos queda crear las variaciones absolutas y relativas.

Pero el potencial está en cuando empieces a filtrar por mes o agregues más Segmentadores por cliente, producto, línea y todos los demás campos que contamos en el modelo para ver el comportamiento año a año de dicha información.



Comparativo Año =
`CALCULATE([Ventas], DATEADD(Calendar[Date], -1, YEAR))`

De esta misma forma podemos crear comparativos a nivel de trimestre y día.

Medidas móviles (rangos de fecha)

Hasta este punto hemos estudiado como crear variaciones entre periodos del año anterior, acumulados y comparativos; no siempre los cálculos llegan hasta ahí cuando de análisis de inteligencia de tiempo se trata, las medidas móviles en DAX son las que permiten movernos en ciertos números de periodos ya sea hacia atrás o hacia adelante para crear acumulados o promedios que permitan ver cambios en un corto plazo.

DATESINPERIOD

Esta es una de las funciones que podemos indicarle un conjunto de fechas entre periodos, por ejemplo, siempre queremos que el resultado nos muestre el acumulado de los últimos 3 meses independiente donde estemos ubicados, si estamos en abril, el resultado sería la suma de febrero, marzo y abril.

DATESINPERIOD tiene esta capacidad de acumular o promediar cierto número de periodos indicados en el número de intervalos, esta función cuenta con 4 argumentos obligatorios:

1. **Fechas:** la misma columna fechas de la tabla calendario
2. **Fecha inicio:** se le indica la fecha desde donde va a traer datos, este argumento puede ser de confusión para algunos, pero siempre revisemos el contexto, si lo que deseamos es traer el acumulado 3 meses atrás, y si estamos ubicados en abril, la fecha de inicio sería el último día de abril, para que tome abril, marzo y febrero, usamos la función MAX.
3. **Número de intervalos:** el número que se especifica a moverse entre el conjunto de fechas
4. **Intervalo:** la periodicidad con la que se va a trabajar, año, trimestre, mes o día.

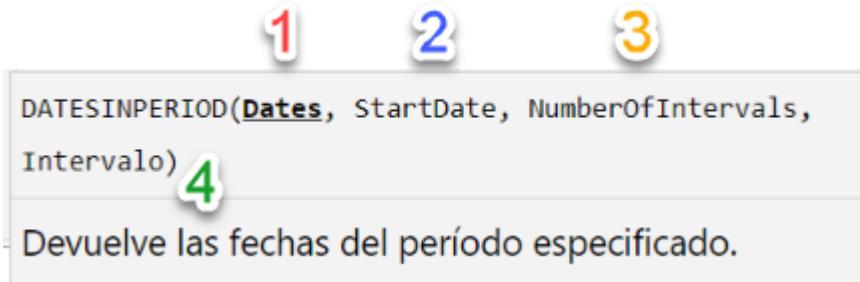


Figura 12.18 Argumentos función DATESTINPERIOD.

Generamos una nueva medida como siempre, acompañada de CALCULATE ya que DATESTINPERIOD arroja una tabla de fechas y sirve como filtro para CALCULATE, según lo estudiado en el capítulo 10.

El contexto para arrojar la medida es la matriz por año y mes.

El resultado sería el siguiente si llevamos la medida a la matriz.

```
1 Acum 3 Meses =  

2 CALCULATE (  

3     [Ventas],  

4     DATESTINPERIOD ( Calendario[Date], MAX ( Calendario[Date] ), -3, MONTH )  

5 )
```

30/4/2017 → DATESTINPERIOD (Calendario[Date], MAX (Calendario[Date]), -3, MONTH)

Año	Ventas	Acum 3 Meses
2017	21.662.728.028	5.830.417.882
Enero	1.025.054.850	1.025.054.850
Febrero	1.422.899.788	2.447.954.638
Marzo	1.702.922.633	4.150.877.271
Abril	1.906.689.262	5.032.511.683
Mayo	1.737.880.439	5.347.492.334
Junio	1.833.686.293	5.478.255.994
Julio	1.617.294.343	5.188.861.075

Figura 12.19 Acumulado 3 meses atrás.

```
Acum 3 Meses =  

CALCULATE (  

    [Ventas],  

    DATESTINPERIOD ( Calendario[Date], MAX ( Calendario[Date] ), -3, MONTH )  

)
```

Si nos concentramos en abril de 2017, el cálculo inicia exactamente desde el 30 de abril hasta el 1 de febrero de 2017, si sumas manualmente estos 3 meses, el resultado debe ser 5.032.511.683.

Promedio móvil

En el ejercicio anterior realizamos un acumulado de los últimos 3 meses, los promedios móviles permiten suavizar los datos para ver comportamientos en cortos períodos de tiempos, esto ayuda a identificar mejores patrones y tener análisis más certeros y con otros escenarios.

En nuestro modelo contamos con una medida que realiza el promedio mes.



Promedio Mes = AVERAGEX(VALUES(Calendario[Mes]), [Ventas])

Como lo vimos en transición de contextos y en medidas de iteración, esta función agrupa las ventas por mes y luego itera creando un promedio de ellos.

El reto es crear un promedio móvil, que nos permita ver el comportamiento de los datos 3 meses atrás.

Simplemente en vez de usar la medida [Ventas], hacemos referencia a la medida [Promedio Mes] y la llevamos a una matriz de mes con filtro por año.

El resultado sería el siguiente.

```
1 Prom Móvil =
2 CALCULATE (
3     [Promedio Mes],
4     DATESINPERIOD ( Calendario[Date], MAX ( Calendario[Date] ), -3, MONTH )
5 )
```

Año	Mes	Ventas	Prom Móvil
2017	Enero	1.607.238.862	1.911.387.616
2018	Febrero	2.044.402.923	1.932.844.421
2019	Marzo	2.232.480.889	1.961.374.225
2019	Abril	2.100.900.104	2.125.927.972
2020	Mayo	2.286.797.916	2.206.726.303
	Junio	2.025.726.818	2.137.808.279
	Julio	1.985.671.439	2.099.398.724

Figura 12.20 Promedio móvil.

El resultado para abril 2019 es el promedio de las ventas de los meses de febrero, marzo y abril.

Esto se vería mejor si cambiamos la matriz por un gráfico de líneas.

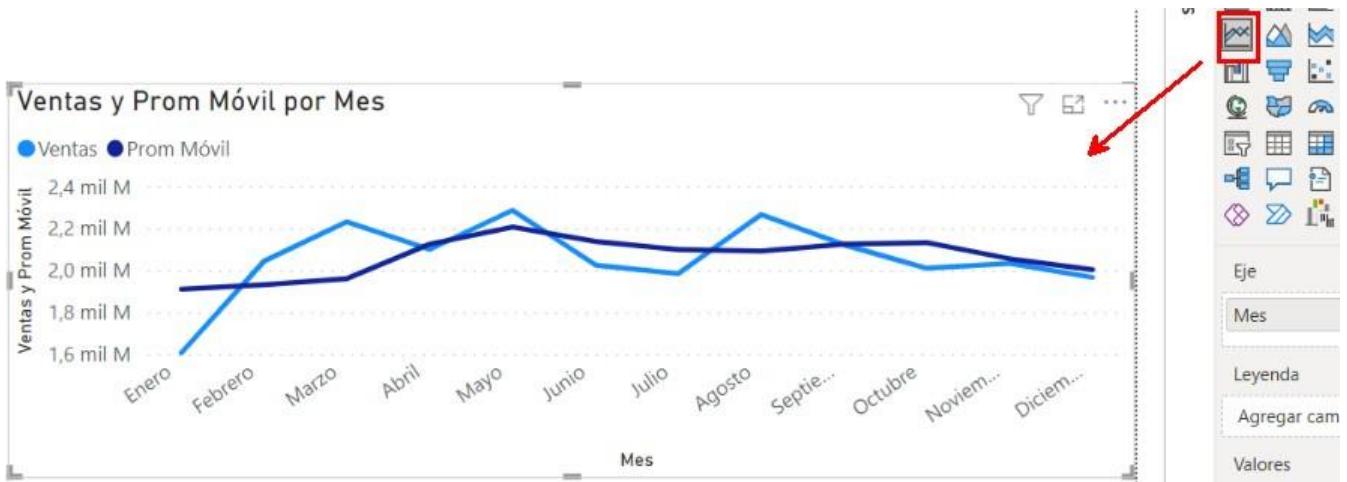


Figura 12.21 Promedio móvil en gráfico de líneas.

Ahora es más claro identificar y comparar los datos de las ventas con el promedio 3 meses atrás.



```
Prom Móvil =  
CALCULATE (  
    [Promedio Mes],  
    DATESINPERIOD ( Calendario[Date], MAX ( Calendario[Date] ), -3,  
    MONTH )  
)
```

Parámetros

La solución anterior es muy estática, es decir, el usuario final solo podría ver siempre el promedio a 3 meses, si quisiera cambiar este argumento a otro valor, tendrías que ser desde Power BI Desktop, algo no recomendable, ya que es un método ambiguo, como desarrolladores de tableros debemos garantizar la mejor experiencia al usuario final.

Power BI cuenta con una herramienta llamada parámetros, es muy útil ya que permite crear escenarios y analizar la información de manera dinámica, que en lugar de poner un número 3 constante, el usuario defina el valor.

Para crear un parámetro siga los siguientes pasos:

Modelado → What if → Parámetro nuevo



Figura 12.22 Crear un parámetro nuevo.

El siguiente paso es parametrizarlo.

Asignamos un nombre, es opcional → Indicamos el número de rango a mover, para este caso sería de 1 a 12 meses (opcional) → Incremento sea de 1 en 1 → y que si no escogen un rango o número a analizar siempre sea 3.



Figura 12.23 Parámetro de hipótesis.

Si damos aceptar en el lienzo hay un desplazador numérico con una nueva medida que se crea por defecto.

The screenshot shows a Power BI interface. On the left, there is a parameter named 'Rango' with a dropdown arrow and a circular selection button. On the right, a context menu is open with the following options:

- Campo: Rango
- Obtener detalles
- Entre varios informes
- Desactivar
- Mantener todos los filtros

A red arrow points from the text '1 Valor Rango = SELECTEDVALUE('Rango'[Rango], 3)' in the top bar to the 'Valor Rango' option in the menu, which is highlighted with a red border.

Figura 12.24 Nueva medida del parámetro.

Solo es hacer un pequeño cambio en la medida [Promedio Móvil], reemplazar el número 3 que es constante por la nueva medida que es dinámica

```
1 Prom Móvil =
2 CALCULATE (
3     [Promedio Mes],
4     DATESINPERIOD ( Calendario[Date], MAX ( Calendario[Date] ), -3, MONTH )
5 )
```

```
1 Prom Móvil =
2 CALCULATE (
3     [Promedio Mes],
4     DATESINPERIOD ( Calendario[Date], MAX ( Calendario[Date] ), -[Valor Rango], MONTH )
5 )
```

Figura 12.25 Editar medida promedio móvil.

Ahora tienes un informe totalmente dinámico y el usuario puede seleccionar el número de periodos que se pueden desplazar los datos hacia atrás.

Si notas el resultado entre más número de periodos se le asigne al rango, mayor suavidad toma la gráfica.



```
Prom Móvil =
CALCULATE (
    [Promedio Mes],
    DATESINPERIOD ( Calendario[Date], MAX ( Calendario[Date] ),
    -[Valor Rango], MONTH )
)
```



Figura 12.26 Promedio móvil dinámico.



Nota: Te vas a encontrar en DAX que hay funciones de inteligencia tiempo que realizan la misma operación o arrojan el mismo resultado, te invitamos a que estudies un poco más sobre estas funciones, nunca olvides el contexto.

<https://docs.microsoft.com/en-us/dax/time-intelligence-functions-dax>

Introducción Inteligencia de tiempo personalizado

Desafortunadamente en DAX no contamos con todas las medidas de inteligencia de tiempo que por defecto vienen y vimos anteriormente, como por ejemplo si queremos comparar los datos a nivel de semana, no existe esta función de manera rápida.

Por ende, debemos recurrir a crear nuestras propias funciones y queremos hacer una pequeña introducción a este tema de crear medidas nativas o personalizadas que será nuestro tema para el próximo volumen 2 de esta serie de DAX.

Comparativo semanas

Vamos a poner esto en un contexto real, como empresa tenemos muchos clientes del sector agrícola que todo lo trabajan a nivel de semanas y un análisis muy normal que nos solicitan es comparar cada

semana con el año anterior, como siempre el objetivo es lograr desplazar el valor de la semana 1 del primer año al lado de la semana 2 del último año, según el contexto.

La imagen nos ilustra cual debe ser el objetivo:

Semana	Año	Ventas
1	2017	2.779.444.175
2		292.679.453
3		204.950.263
4		296.996.655
5		230.428.479
6		332.490.636
7		379.818.463
8		299.901.286
9		410.689.403
10		331.489.537
11	2018	4.264.908.245
12		332.571.531
13		198.301.993
14		416.613.438
15		459.838.870
16		333.120.625
17		467.336.667
18		569.267.257

Figura 12.27 Objetivo comparativo semanas.

Seleccionamos solo 10 semanas en el filtro para acotar un poco los datos y pintar mejor el objetivo.

Solucionemos esto por partes, vamos a paso a paso como la canción.

Debemos crear una medida de manera nativa que me desplace el año un valor atrás, es decir que si arrojamos esta medida a la matriz, para el contexto 2018, va a salir 2017. Es una medida muy fácil, solo es extraer el valor MÁXIMO de la columna año y restarle 1

Comparar Semana = MAX(Calendar[Año])-1

Año	Ventas	Comparar Semana
4	296.996.655	2016
5	230.428.479	2016
6	332.490.636	2016
7	379.818.463	2016
8	299.901.286	2016
9	410.689.403	2016
10	331.489.537	2016
2018	4.264.908.245	2017
1	332.571.531	2017
2	198.301.993	2017
3	416.613.438	2017

Figura 12.28 Desplazar un año atrás.

El siguiente paso ya es validar este filtro dentro de CALCULATE, donde le indicamos que las ventas de cada año por semana deben ser igual al año que desplazamos un periodo atrás.

La medida sería la siguiente:

Año	Ventas	Comparar Semana
2017	2.779.444.175	
	292.679.453	
	204.950.263	
	296.996.655	
	230.428.479	
	332.490.636	
	379.818.463	
	299.901.286	
	410.689.403	
	331.489.537	
2018	4.264.908.245	2.779.444.175
	332.571.531	292.679.453
	198.301.993	204.950.263
	416.613.438	296.996.655
	459.838.870	230.428.479

Figura 12.29 Desplazar ventas por semana un año atrás.



Comparar Semana =
CALCULATE ([Ventas], Calendario[Año] = MAX (Calendario[Año]) - 1)

Los valores se desplazaron correctamente, semana 1 del año 2018 está en blanco porque no se registraron ventas en semana 1 de 2017.

Solo es calcular las respectivas variaciones, absolutas y relativas estudiadas anteriormente.

Capítulo 13: Recetas

Es muy común encontramos en internet, videos tutoriales, redes o blogs de estudio fórmulas de DAX que realizan cálculos especiales en un modelo determinado, al inicio indicamos no es muy recomendable hacer esto, pero si lo has hecho y te ha ayudado a resolver algún problema es lo normal mientras no conocíamos las bases bajo la cual opera DAX ahora es posible hacerlo; a última hora te confesamos hemos decidido hacer este anexo ya que creemos muy necesario porque vas a tener la capacidad de entender las recetas que vamos a brindarte para que puedas aplicar en tus informes.

Estamos seguros de que si has leído detenidamente el capítulo 9 y 10 o si es necesario volver allí y darle una nueva repasada vas a poder comprender las fórmulas que vamos a aplicar en los modelos que hemos creado para que practiques todo lo aprendido.

Los cálculos que te vamos a presentar a continuación los usamos constantemente en los desarrollos que hacemos en DATAICE para múltiples empresas, hemos creado dos modelos totalmente diferentes al que venimos explicando (comercial y financiero), son modelos reales de empresas que asesoramos solo que con datos totalmente cambiados por obvias razones.

El primer modelo para trabajar será el “Modelo Ventas Recetas”, allí haremos unos cálculos bien interesantes que seguro van a hacer de gran provecho si los usas en tus datos.

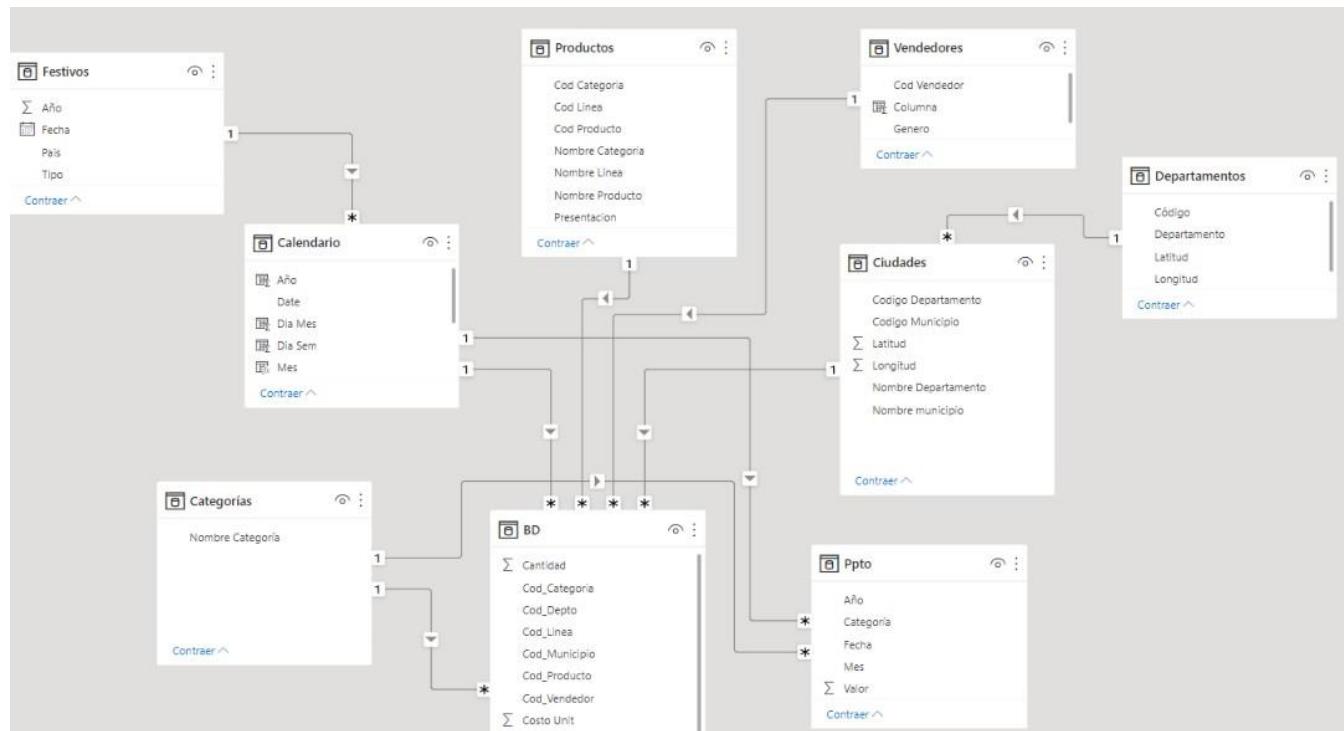


Figura 13.1 Modelos comercial ventas y presupuesto.

En este modelo ya contamos con algunas medidas básicas creadas y vistas en capítulos anteriores.

El área comercial debe analizar el comportamiento de compra de los clientes o por lo menos saber que clientes son fieles o recurrentes de un año a otro, que clientes son nuevos y sobre todo cuales de ellos no han vuelto a comprar comparado con el año anterior.

Clients recurrentes

Si tenemos un filtro de año y seleccionamos cualquiera de ellos, el informe nos deberá mostrar que clientes de este año segmentado han venido comprando o han sido “fieles” con respecto al año anterior, es decir, si tenemos 2022 que nos muestre los clientes que compran 2022 y 2021 consecutivamente, la receta es la siguiente si la medida la llevamos a una matriz de clientes con el filtro de 2022:

```
1 Clientes Recurrentes =
2 VAR Year_Max =
3     MAX ( Calendario[Año] )
4 VAR Clientes_Actuales =
5     CALCULATETABLE (
6         SUMMARIZE ( BD, BD[Nombre Cliente] ),
7         Calendario[Año] = Year_Max
8     )
9 VAR Clientes_YearAnterior =
10    CALCULATETABLE (
11        SUMMARIZE ( BD, BD[Nombre Cliente] ),
12        Calendario[Año] = Year_Max - 1
13    )
14 VAR Clientes_Interceptados =
15     INTERSECT ( Clientes_Actuales, Clientes_YearAnterior )
16 VAR Resultado =
17     CALCULATE ( [Ventas], Clientes_Interceptados )
18 RETURN
19 Resultado
```

Nombre Cliente	Clientes Recurrentes
DISTRIBUIDORA PUNTO SEIS	732.264
5 ESQUINAS	273.892
61PRADO EUROPEAN GUESTHOUSE	1.339.409
ABARROTES ALFER	194.154
ABARROTES CUCUTA	345.749
ABARROTES DORA	272.814
ABARROTES EL ABASTO	317.194
ABARROTES EL CAFETERO SAS	163.860
ABARROTES EL CONDE	204.505
Total	1.345.414.453

Figura 13.2 Medida clientes recurrentes en la matriz.

Si analizas cada línea del código, cada una de estas funciones fueran explicadas en los capítulos anteriores, es solo cuestión de ponerlas en contexto, la receta calculó el valor de los clientes que compraron en 2022 y en 2021, los que no están en este intercepto, la saca de la matriz, ya que están en blanco.

Si llevamos la medida [Ventas] a una tarjeta, podemos observar cuento fue la venta de 2022 comparada con la venta de sus clientes más fieles.

1.463.371.734	
Ventas	
Nombre Cliente	
DISTRIBUIDORA PUNTO SEIS	732.264
5 ESQUINAS	273.892
61PRADO EUROPEAN GUESTHOUSE	1.339.409
ABARROTES ALFER	194.154
ABARROTES CUCUTA	345.749
ABARROTES DORA	272.814
ABARROTES EL ABASTO	317.194
ABARROTES EL CAFETERO SAS	163.860
ABARROTES EL CONDE	204.505
Total	1.345.414.453

Figura 13.3 Medida clientes recurrentes y ventas totales 2022.

La tarjeta muestra que en 2022 las ventas son 1.463 millones, lo cual se han vendido a los mismos clientes en dos años consecutivos 1.345 millones, si haces esta división tenemos un 91,9 % de efectividad en clientes fieles.



```

Clientes Recurrentes =
VAR Year_Max =
    MAX ( Calendario[Año] )
VAR Clientes_Actuales =
    CALCULATETABLE (
        SUMMARIZE ( BD, BD[Nombre Cliente] ),
        Calendario[Año] = Year_Max
    )
VAR Clientes_YearAnterior =
    CALCULATETABLE (
        SUMMARIZE ( BD, BD[Nombre Cliente] ),
        Calendario[Año] = Year_Max - 1
    )
VAR Clientes_Interceptados =
    INTERSECT ( Clientes_Actuales, Clientes_YearAnterior )
VAR Resultado =
    CALCULATE ( [Ventas], Clientes_Interceptados )
RETURN
    Resultado

```

Si analizas la medida también puedes crear un cálculo no solo que te sume las ventas de los clientes recurrentes, sino que te cuente el número de clientes, en la variable Resultados, cambias [Ventas] por DISTINCTCOUNT (BD [Nombre Cliente])

Clients nuevos

Saber que clientes han entrado nuevos en un año específico comparado con el año anterior es el objetivo, también se puede considerar clientes nuevos aquellos que alguna vez o años anteriores compraron y regresaron después de 1 año.

La receta es muy similar a la anterior, solo es hacer un pequeño cambio en la función de tabla INTERSECT por EXCEPT, si llevamos esta medida a una matriz de cliente obtendremos el siguiente resultado.

```
1 Clientes_Nuevos =
2 VAR Year_Max =
3     MAX ( Calendario[Año] )
4 VAR Clientes_Actuales =
5     CALCULATETABLE (
6         SUMMARIZE ( BD, BD[Nombre Cliente] ),
7         Calendario[Año] = Year_Max
8     )
9 VAR Clientes_YearAnterior =
10    CALCULATETABLE (
11        SUMMARIZE ( BD, BD[Nombre Cliente] ),
12        Calendario[Año] = Year_Max - 1
13    )
14 VAR Clientes_Interceptados =
15     EXCEPT ( Clientes_Actuales, Clientes_YearAnterior )
16 VAR Resultado =
17     CALCULATE ( [Ventas], Clientes_Interceptados )
18 RETURN
19     Resultado
```

	Nombre Cliente	Clientes Nuevos
	ABARROTES EL BARATILLO	429.080
	ABARROTES EL CLAN	372.177
	ABARROTES EL FILTRO	89.067
	ABARROTES JUAN MAZO	208.756
	ABORROTES MAFELA	274.144
	ACOSTA LUGO ALFONSO	150.241
	Total	117.957.281

Figura 13.4 Medida clientes nuevos.

Con un pequeño cambio tendremos el resultado esperado, ¿cómo podemos comprobar que el valor arrojado es correcto?, simplemente si sumas el valor de los clientes recurrentes **1.343.414.453** y el valor de los clientes nuevos **117.957.281** nos debe dar el total ventas 2022 =**1.463.371.734**

WOW DAX es asombroso, ¿no te parece?

Y de la misma forma puedes contar el número de clientes nuevos y compararlos con el total de clientes que tiene la empresa y así crear KPI's de seguimiento con respecto a esta medida.



```
Clientes_Nuevos =
VAR Year_Max =
    MAX ( Calendario[Año] )
VAR Clientes_Actuales =
    CALCULATETABLE (
        SUMMARIZE ( BD, BD[Nombre Cliente] ),
        Calendario[Año] = Year_Max
    )
VAR Clientes_YearAnterior =
    CALCULATETABLE (
        SUMMARIZE ( BD, BD[Nombre Cliente] ),
        Calendario[Año] = Year_Max - 1
    )
VAR Clientes_Interceptados =
    EXCEPT( Clientes_Actuales, Clientes_YearAnterior )
VAR Resultado =
    CALCULATE ( [Ventas], Clientes_Interceptados )
RETURN
    Resultado
```

Clientes perdidos

Este es uno de los indicadores más relevantes que no puede faltar en el área comercial, tener en un tablero que clientes no han vuelto comprar de un periodo a otro, o que se le han generado alguna venta, pero ha pasado determinado tiempo sin volverse a impactar.

En esta parte debemos hacer dos ajustes:

1. En la variable **Clientes_Interceptados** debemos cambiar la posición, primero van los clientes del año anterior y luego los clientes del año actual.
2. En la variable **Resultado** se cambia la medida [Ventas] por las ventas del año anterior, ya que es el valor que necesitamos, que clientes no han vuelto a comprar.

Si llevamos esta nueva medida de clientes perdidos con los ajustes correspondientes a la matriz por clientes tendremos el resultado buscado, según la figura 13.5

1	Clientes Perdidos =
2	VAR Year_Max =
3	MAX (Calendario[Año])
4	VAR Clientes_Actuales =
5	CALCULATETABLE (
6	SUMMARIZE (BD, BD[Nombre Cliente]),
7	Calendario[Año] = Year_Max
8)
9	VAR Clientes_YearAnterior =
10	CALCULATETABLE (
11	SUMMARIZE (BD, BD[Nombre Cliente]),
12	Calendario[Año] = Year_Max - 1
13)
14	VAR Clientes_Interceptados =
15	EXCEPT(Clientes_YearAnterior, Clientes_Actuales)
16	VAR Resultado =
17	CALCULATE ([Ventas Año Anterior], Clientes_Interceptados)
18	RETURN
19	Resultado

Nombre Cliente	Clientes Perdidos
HOTEL PLAZA 70	726.611
INVERSIONES EURO S.A.	25.601.175
MERCAMAX	898.835
PATIO DEL MUNDO	1.098.488
Total	28.325.109

Figura 13.5 Medida clientes perdidos.

Observamos que son pocos clientes los que no han vuelto a comprar en 2022 comparado con 2021.



```

Clientes Perdidos =
VAR Year_Max =
    MAX ( Calendario[Año] )
VAR Clientes_Actuales =
    CALCULATETABLE (
        SUMMARIZE ( BD, BD[Nombre Cliente] ),
        Calendario[Año] = Year_Max
    )
VAR Clientes_YearAnterior =
    CALCULATETABLE (
        SUMMARIZE ( BD, BD[Nombre Cliente] ),
        Calendario[Año] = Year_Max - 1
    )
VAR Clientes_Interceptados =
    EXCEPT( Clientes_YearAnterior, Clientes_Actuales )
VAR Resultado =
    CALCULATE ( [Ventas Año Anterior], Clientes_Interceptados )
RETURN
    Resultado

```

Pero incluso me atrevo a decir que esto no es todo, porque puedes crear un tablero con tanta imaginación gráfica y añadir los filtros que sean necesarios para ver el comportamiento de estos clientes por vendedor, ciudad, línea, producto etc.



Nota: Puedes aumentar un poco la complejidad en las recetas, por ejemplo, en las variables de Clientes Actuales y años anteriores, puedes añadir a la función SUMMARIZE que no solo te tome el nombre del cliente, sino el cliente y la sucursal, y así determinar cliente por diferentes zonas, o crear un filtro que no te tenga en cuentas los clientes con ventas en valor cero o negativos, todo esto y mucho más según tu necesidad.

Porcentajes de participación columnas y filas

Crear medidas que muestren la participación de un ítem en una tabla o gráfica es muy común en Power BI, incluso lo vimos en el capítulo 10 de CALCULATE y funciones de filtro, pero en ocasiones necesitamos crear medidas no solo que me calculen la participación a nivel de fila, sino a nivel de columna, pongamos esto en contexto.

Antes de entrar en materia y dar la receta o varias recetas que te ayudarán a solucionar estos problemas, una manera sencilla de crear un porcentaje de participación y resumir lo visto en capítulos anteriores es esta medida:



% Partici/ =
DIVIDE ([Ventas], CALCULATE ([Ventas], ALLSELECTED ()))

Con esta medida puedes calcular el porcentaje de participación de cualquier variable o campo que lleves a una matriz u objeto visual como lo muestra la siguiente imagen.

Nombre Categoría	Ventas	% Partici/
BEBIDAS	74.202.651	5,1 %
CARNICA	130.512.207	8,9 %
HELADO	39.729.454	2,7 %
LACTEA	1.218.927.421	83,3 %
Total	1.463.371.734	100,0 %

Nombre Linea	Ventas	% Partici/
CARNES FRIAS	130.512.207	8,9 %
ESPARCIBLES	218.036.582	14,9 %
FIT	162.885.068	11,1 %
HELADOS	39.729.454	2,7 %
JUGOS Y REFRESCOS	74.202.651	5,1 %
KIDS	54.932.442	3,8 %
LACTEOS	350.795.841	24,6 %
Total	1.463.371.734	100,0 %

Nombre Producto	Ventas	% Partici/
AGUA CON GAS	4.191.361	0,3 %
AGUA SIN GAS	4.809.789	0,3 %
AMERICANO	43.803.841	3,0 %
AREQUIPE	17.305.898	1,2 %
AREQUIPE MINI	37.896.455	2,6 %
AREQUIPE PER	90.061.607	6,2 %
AVENA	22.851.062	1,6 %
Total	1.463.371.734	100,0 %

Figura 13.6 Medida de participación para diferentes campos.

Participación a nivel de columna

Que sucede si en la matriz por categoría agregamos el año en columna y borramos el filtro de año.

Año	2018			2019			2020			2021			2022			Total	
	Nombre Categoría	Ventas	% Partici/	Ventas	% Partici/	Ventas	% Partici/	Ventas	% Partici/	Ventas	% Partici/	Ventas	% Partici/	Ventas	% Partici/	Ventas	% Partici/
BEBIDAS	25.552.710	0,5 %	45.822.517	0,8 %	76.051.276	1,4 %	61.977.335	1,1 %	74.202.651	1,4 %	283.606.489						
CARNICA	83.145.844	1,5 %	62.662.093	1,1 %	149.867.347	2,7 %	157.892.171	2,9 %	130.512.207	2,4 %	584.079.663						
HELADO	34.918.820	0,6 %	18.059.868	0,3 %	5.540.910	0,1 %	14.282.051	0,3 %	39.729.454	0,7 %	112.531.104						
LACTEA	505.661.735	9,3 %	761.506.036	13,9 %	1.008.989.831	18,5 %	986.408.885	18,1 %	1.218.927.421	22,3 %	4.481.493.908						
Total	649.279.109	11,9 %	888.050.515	16,3 %	1.240.449.364	22,7 %	1.220.560.442	22,3 %	1.463.371.734	26,8 %	5.461.711.164						

Figura 13.7 % Participación por categoría y año.

Analicemos son un dato en el contexto de la matriz → HELADO → 2018

El resultado 0,6% es el resultado de dividir las ventas de Helado de 2018, 34.918.820 entre el total de totales, o sea entre 5.461.711.164, si revisamos la medida, es lo que hace ALLSELECTED, omitir cualquier contexto de filtro que se encuentre en la matriz, para este caso Categoría y Año, pero esto no es un análisis pertinente, ya que no queremos ver como participa cada categoría en el total de todos los años, sino el en total de cada año, para este caso, como es la participación de helado en el 2018.

Como ALLSELECTED omite cualquier contexto, solo debemos devolver el filtro para año, es decir, en CALCULATE, debemos agregar una función que devuelva el filtro del año o lo tenga en cuenta en la matriz, usamos la función VALUES haciendo referencia al año.

Creamos otra medida para que tengas la historia de las medidas o recetas generadas en este capítulo.

```

1 % Partici/ Columna =
2 DIVIDE (
3     [Ventas],
4     CALCULATE ( [Ventas], ALLSELECTED (), VALUES ( Calendario[Año] ) )
5 )
6

```

Año	2018	2019	2020			
Nombre Categoría	Ventas	% Partici/ Columna	Ventas	% Partici/ Columna	Ventas	% Partici/ Columna
BEBIDAS	25.552.710	3,94 %	45.822.517	5,16 %	76.051.276	6,13 %
CARNICA	83.145.844	12,81 %	62.662.093	7,06 %	149.867.347	12,08 %
HELADO	34.918.820	5,38 %	18.059.868	2,03 %	5.540.910	0,45 %
LACTEA	505.661.735	77,88 %	761.506.036	85,75 %	1.008.989.831	81,34 %
Total	649.279.109	100,00 %	888.050.515	100,00 %	1.240.449.364	100,00 %

Figura 13.8 % Participación devolviendo el filtro del año.

Ahora la imagen muestra que cada categoría se divide por cada uno de los años, es decir, es un porcentaje de participación a nivel de columna.



```

% Partici/ Columna =
DIVIDE (
    [Ventas],
    CALCULATE ( [Ventas], ALLSELECTED (), VALUES ( Calendario[Año] ) )
)

```

Participación a nivel de fila

Con el anterior cálculo ya podemos intuir todo lo que podemos realizar y los respectivos análisis que este tipo de recetas nos pueden dar, si queremos mostrar en el informe NO el porcentaje de participación de cada categoría por año, sino enseñar como cada año ha participado en cada categoría, simplemente es cambiar en VALUES el campo año por el campo categoría, es decir, devolvemos el filtro de categoría en CALCULATE.

```

1 % Partici/ Fila =
2 DIVIDE (
3     [Ventas],
4     CALCULATE ( [Ventas], ALLSELECTED (), VALUES ( 'Categorías'[Nombre Categoría]) )
5 )

```

Año	2018	2019	2020	2021	2022	Total	Ventas	% Partici/ Fila
Nombre Categoría	Ventas	% Partici/ Fila	Ventas	% Partici/ Fila	Ventas	% Partici/ Fila	Ventas	% Partici/ Fila
BEBIDAS	25.552.710	9,01 %	45.822.517	16,16 %	76.051.276	26,82 %	61.977.335	21,85 %
CARNICA	83.145.844	14,24 %	62.662.093	10,73 %	149.867.347	25,66 %	157.892.171	27,03 %
HELADO	34.918.820	31,03 %	18.059.868	16,05 %	5.540.910	4,92 %	14.282.051	12,69 %
LACTEA	505.661.735	11,28 %	761.506.036	16,99 %	1.008.989.831	22,51 %	986.408.885	22,01 %
Total	649.279.109	11,89 %	888.050.515	16,26 %	1.240.449.364	22,71 %	1.220.560.442	22,35 %
							1.463.371.734	26,79 %
							5.461.711.164	100,00 %

Figura 13.9 % Participación devolviendo el filtro de categoría.

El análisis cambia, ya el porcentaje de participación es a nivel de fila, es cada año como participa en el total de cada categoría.



```

% Partici/ Fila =
DIVIDE (
    [Ventas],
    CALCULATE ( [Ventas], ALLSELECTED (), VALUES ( 'Categorías'[Nombre
    Categoría]) )
)

```

Proyección lineal

Muchos líderes y gerentes comerciales analizan datos a diario, si no tienen esta cultura es porque no cuentan con las herramientas adecuadas para hacerlo, por esto sabemos que es una oportunidad que tienes para implementar un análisis lineal en tu compañía, y es de manera sencilla determinar el comportamiento de las ventas y si se continua a este ritmo proyectar un cierre mes y como compararlo con un presupuesto, cumplimos o no cumplimos.

Antes de darte esta receta vamos a ir paso a paso para que comprendamos juntos mejor el resultado final:

1. Debemos crear una medida que nos brinde el número de días hábiles de cada mes.
2. Determinar el número de día actual, según el mes seleccionado, ejemplo, si estamos en el mes de febrero, saber el día actual que se está viendo el reporte (HOY).
3. Calcular el número de días transcurridos del mes actual.
4. Determinar el número de días faltantes de cierre para el mes de análisis.
5. Dividir las ventas del mes en curso sobre el número de días transcurridos para tener el promedio venta día
6. Por último, multiplicar el promedio de venta día por el número de días hábiles mes corriente.

Antes de iniciar preparamos el ambiente en Power BI, filtramos el año 2022 y el mes de febrero (mes actual en el que se terminó el libro).

Días hábiles

Para calcular los días hábiles solo es definir si es de lunes a viernes o lunes a sábado y no tener en cuenta los días festivos o feriados según el país, recuerda cambiar el parámetro del país en el modelo.

The screenshot shows the Power BI formula editor with the following code:

```
1 Días Hábiles =  
2 CALCULATE (  
3     COUNTROWS ( Calendario ),  
4     Calendario[Dia Sem] <= 5  
5     && Calendario[Feriado] = BLANK ()  
6 )
```

Below the code is a table visual with the following data:

Mes	Días Hábiles
Febrero	20
Total	20

Figura 13.10 Medida días hábiles.



```
Días Hábiles =  
CALCULATE (  
    COUNTROWS ( Calendario ),  
    Calendario[Dia Sem] <= 5  
    && Calendario[Feriado] = BLANK ()  
)
```

Día actual

Creamos la siguiente medida para obtener el día actual y la llevamos a la matriz.

The screenshot shows the Power BI formula editor with the following code:

```
1 Día Actual = TODAY()
```

A red arrow points from the formula editor down to the table visual below.

Below the code is a table visual with the following data:

Mes	Días Hábiles	Día Actual
Febrero	20	04/02/2022 12:00:00 a.m.
Total	20	04/02/2022 12:00:00 a.m.

Figura 13.11 Medida día actual.



Día Actual = TODAY()

Días corridos

Debemos calcular cuantos días a transcurridos a la fecha del mes evaluado.

1 Días Corridos = Day([Día Actual])			
2022			
Mes	Días Hábiles	Día Actual	Días Corridos
Febrero	20	04/02/2022 12:00:00 a.m.	4
Total	20	04/02/2022 12:00:00 a.m.	4

Figura 13.12 Medida días corridos.

Extraemos el día de la fecha.



$$\text{Días Corridos} = \text{Day}(\text{[Día Actual]})$$

Días faltantes

Determinar cuántos días hábiles faltan para que se termine el mes, solo es una resta.

Días Faltantes = [Días Hábiles]-[Días Corridos]			
2022	Febrero		
Días Hábiles	Día Actual	Días Corridos	Días Faltantes
20	04/02/2022 12:00:00 a.m.	4	16
20	04/02/2022 12:00:00 a.m.	4	16

Figura 13.13 Medida días faltantes.



$$\text{Días Faltantes} = \text{[Días Hábiles]} - \text{[Días Corridos]}$$

Venta promedio día

Dividimos las ventas que se llevan del mes entre el número de días corridos.

Venta Prom Día = DIVIDE([Ventas],[Días Corridos])				
2022	Febrero			
Días Hábiles	Día Actual	Días Corridos	Días Faltantes	Ventas
20	04/02/2022 12:00:00 a.m.	4	16	94.657.405
20	04/02/2022 12:00:00 a.m.	4	16	94.657.405

Figura 13.14 Medida venta promedio día.



$$\text{Venta Prom Día} = \text{DIVIDE}(\text{[Ventas]}, \text{[Días Corridos]})$$

Proyección

Por último, multiplicamos la venta promedio diaria por el número de días hábiles del mes y tenemos el resultado con un seguimiento más periódico del comportamiento de las ventas.

Proyección = [Venta Prom Día]*[Días Hábiles]					
	2022	▼	Febrero	▼	
Días Hábiles	Día Actual	Días Corridos	Días Faltantes	Ventas	Venta Prom Día
20	04/02/2022 12:00:00 a.m.	4	16	94.657.405	23.664.351
20	04/02/2022 12:00:00 a.m.	4	16	94.657.405	23.664.351
					473.287.027
					473.287.027

Figura 13.15 Proyección mes.



Proyección = [Venta Prom Día]*[Días Hábiles]

Ahora podemos hacer cálculos comparativos contra presupuestos, mes anterior o segmentar la información para ver la proyección por vendedor, línea, producto etc.



Nota: Este cálculo es aplicado solo al mes corriente que se está analizando, no es útil para meses anteriores

Te sugerimos un paso a paso para llegar a este tipo de soluciones, ya cuando lleves más horas en DAX puedes crear los siguientes cálculos para llegar al mismo resultado, pero ahorrando medidas intermedias.

Días corridos todos los meses

La siguiente medida te puede servir si quieres ver los días corridos de todos los meses incluso el mes corriente.

Seleccionamos enero y febrero para el ejemplo, aunque el mes de enero ya es pasado, calcula el número de días que tuvo enero, y así la proyección sería la misma venta real del mes para los meses que ya pasaron.

```

1 Dias Corridos 2 =
2 VAR Hoy =
3   TODAY ()
4 VAR Mes =
5   MAX ( Calendario[Mes Num] )
6 VAR Dias_Hoy =
7   CALCULATE (
8     COUNTROWS ( Calendario ),
9     FILTER (
10       ALL (
11         Calendario[Dia Sem],
12         Calendario[Feriado],
13         Calendario[Date],
14         Calendario[Mes Num]
15       ),
16       Calendario[Dia Sem] <= 6
17       && Calendario[Feriado] = BLANK ()
18       && Calendario[Mes Num] = Mes
19     ),
20     FILTER ( Calendario, Calendario[Date] < Hoy )
21   )
22 RETURN
23 COALESCE ( Dias_Hoy, 0 )

```

Mes	Dias Corridos 2
Enero	24
Febrero	3
Total	3

Figura 13.16 Medida días corridos para todos los meses.



```

Dias Corridos 2 =
VAR Hoy =
  TODAY ()
VAR Mes =
  MAX ( Calendario[Mes Num] )
VAR Dias_Hoy =
  CALCULATE (
    COUNTROWS ( Calendario ),
    FILTER (
      ALL (
        Calendario[Dia Sem],
        Calendario[Feriado],
        Calendario[Date],
        Calendario[Mes Num]
      ),
      Calendario[Dia Sem] <= 6
      && Calendario[Feriado] = BLANK ()
      && Calendario[Mes Num] = Mes
    ),
    FILTER ( Calendario, Calendario[Date] < Hoy )
  )
RETURN
  COALESCE ( Dias_Hoy, 0 )

```

Con esta nueva medida puedes volver a calcular la proyección para cada mes aun si ya es pasado y para el mes que está en curso.

Estados financieros (PyG)

DATAICE es una empresa muy conocida por crear estados financieros en Power BI, ya que fuimos los primeros en proponer la mejor metodología de llevar los resultados de las empresas a esta herramienta, pero no solo nos dedicamos a esto, hemos creado todo tipo de desarrollos para el área de recursos humanos, producción, logística, agronomía, contabilidad, comisiones, cartera etc. Pero no podíamos dejar por fuera una de nuestras recetas favoritas y es como creamos una fórmula que realice el cálculo de cada uno de los rubros los cuáles esta compuesto un PyG o estado de pérdidas y ganancias.

El gran reto se encuentra en que el estado de resultados tiene cálculos especiales a nivel de fila, como lo es por ejemplo la utilidad bruta y demás utilidades, adicional se debe presentar en un formato u orden específico por temas legales y costumbre gerencial, esto es lo que lo hace tener un tratamiento único y se debe cargar una tabla al modelo con esta estructura y NO se puede relacionar, veamos.

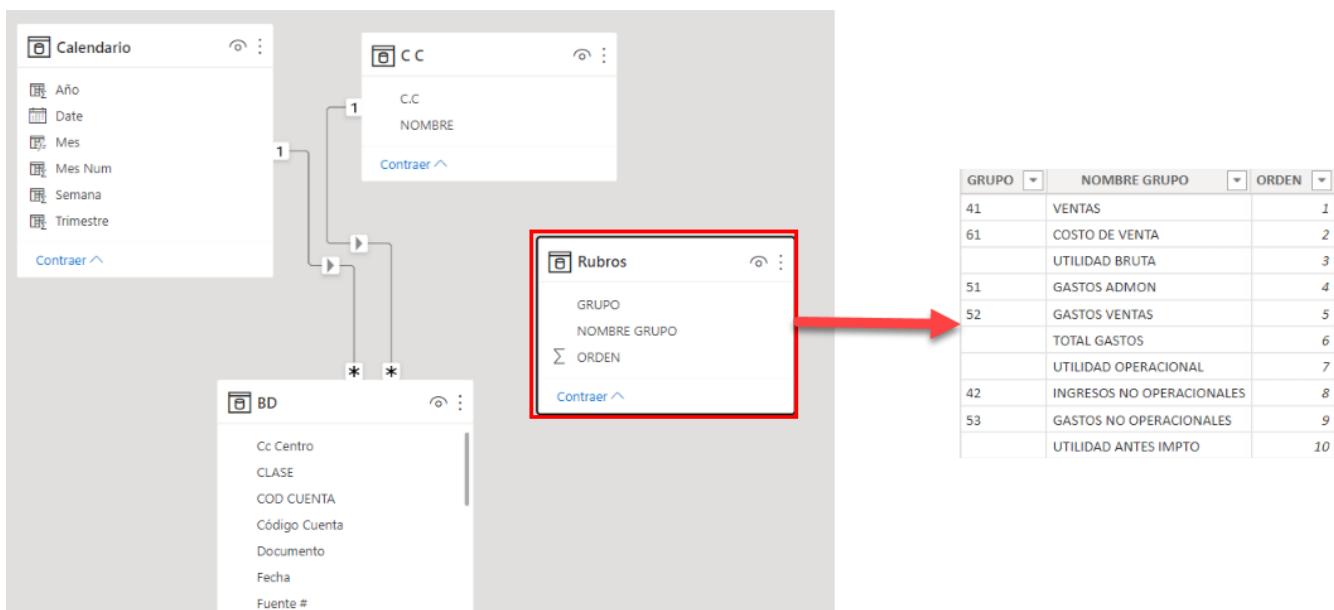


Figura 13.17 Modelo estado de resultados.

La tabla Rubros no se encuentra relacionada y tampoco se puede hacer, ya que hay filas que nunca se van a encontrar en la tabla BD como lo son las utilidades, estos son cálculos aritméticos que dependen de rubros anteriores.

En el capítulo 7 explicamos la función SWITCH, y es la que nos va a ayudar a resolver este tipo de problemas, cuando se necesita crear cálculos para estructuras personalizadas.

Previamente se creó una medida que suma el valor de los movimientos contables (Ingresos, Costos y Gastos), es una medida simple.

Luego se ordenó el campo NOMBRE GRUPO de la tabla Rubros por la columna ORDEN, este es un gran truco, tal como se hace con la columna mes nombre en la tabla calendario, debe ir ordenada según el mes número.

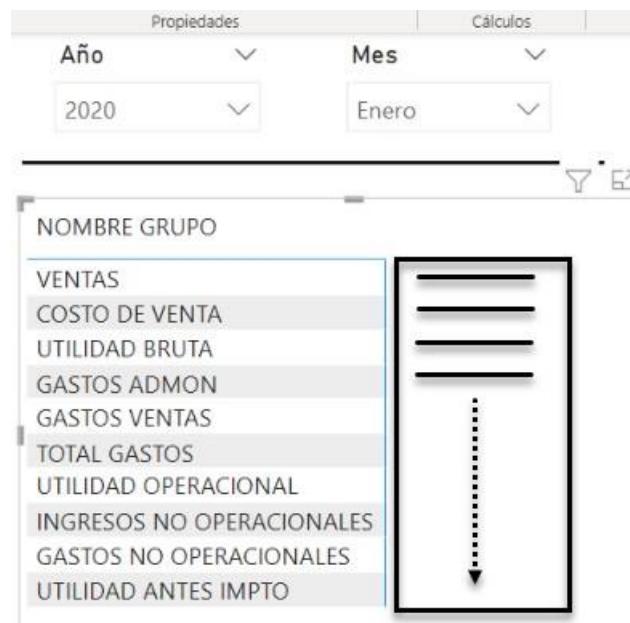


Figura 13.18 Matriz por nombre grupo filtrada por año 2020 y mes enero.

La figura 13.18 muestra el diagrama que es lo que debemos realizar, una función que recorra fila a fila la matriz e identifique el nombre del rubro y aplique el cálculo indicado y pues SWITCH lo puede hacer.

La siguiente función es la solución, si la vez de primera vez puede parecer un poco intimidante, pero si la detallas solo es un proceso repetido.

```

1 Valor_Rubro =
2 VAR Rubro =
3 |> SELECTEDVALUE ( Rubros[NOMBRE GRUPO] )
4 VAR Ingresos =
5 |> CALCULATE ( [Valor], BD[GRUPO] = "41" )
6 VAR Costos =
7 |> CALCULATE ( [Valor], BD[GRUPO] = "61" )
8 VAR Utilidad_Bruta = Ingresos - Costos
9 VAR GastosAdmon =
10 |> CALCULATE ( [Valor], BD[GRUPO] = "51" )
11 VAR GastosVenta =
12 |> CALCULATE ( [Valor], BD[GRUPO] = "52" )
13 VAR TotalGastos = GastosAdmon + GastosVenta
14 VAR U_Operativa = Utilidad_Bruta - GastosAdmon - GastosVenta
15 VAR Margen_operativo =
16 |> DIVIDE ( U_Operativa, Ingresos )
17 VAR IngresosNoOperacionales =
18 |> CALCULATE ( [Valor], BD[GRUPO] = "42" )
19 VAR GastosNoOperacionales =
20 |> CALCULATE ( [Valor], BD[GRUPO] = "53" )
21 VAR UtilidadAntesImpo = U_Operativa + IngresosNoOperacionales + GastosNoOperacionales
22
23 RETURN
24 |> SWITCH (
25 |>     Rubro,
26 |>     "VENTAS", Ingresos,
27 |>     "COSTO DE VENTA", Costos,
28 |>     "UTILIDAD BRUTA", Utilidad_Bruta,
29 |>     "GASTOS ADMON", GastosAdmon,
30 |>     "GASTOS VENTAS", GastosVenta,
31 |>     "TOTAL GASTOS", TotalGastos,
32 |>     "UTILIDAD OPERACIONAL", U_Operativa,
33 |>     "% UTILIDAD OPERACIONAL", Margen_operativo,
34 |>     "INGRESOS NO OPERACIONALES", IngresosNoOperacionales,
35 |>     "GASTOS NO OPERACIONALES", GastosNoOperacionales,
36 |>     "UTILIDAD ANTES IMPTO", UtilidadAntesImpo,
37
38 |>     BLANK ()
39 )
--
```

Figura 13.19 Medida para calcular cada rubro.

Si la analizamos con detenimiento no hay nada que no hemos visto en este libro y de esta manera podemos tener los resultados para cualquier empresa, no importa la estructura, no importa que cálculos se tengan que realizar a nivel de fila, este es el método.

NOMBRE GRUPO	Valor Rubro
VENTAS	575.186.736
COSTO DE VENTA	349.884.130
UTILIDAD BRUTA	225.302.606
GASTOS ADMON	99.562.875
GASTOS VENTAS	165.500.098
TOTAL GASTOS	265.062.973
UTILIDAD OPERACIONAL	-39.760.367
INGRESOS NO OPERACIONALES	16.831.634
GASTOS NO OPERACIONALES	52.523.152
UTILIDAD ANTES IMPTO	-75.451.885
Total	

Figura 13.20 Estado de resultados solucionado.



```

Valor Rubro =
VAR Rubro =
    SELECTEDVALUE ( Rubros[NOMBRE GRUPO] )
VAR Ingresos =
    CALCULATE ( [Valor], BD[GRUPO] = "41" )
VAR Costos =
    CALCULATE ( [Valor], BD[GRUPO] = "61" )
VAR Utilidad_Bruta = Ingresos - Costos
VAR GastosAdmon =
    CALCULATE ( [Valor], BD[GRUPO] = "51" )
VAR GastosVenta =
    CALCULATE ( [Valor], BD[GRUPO] = "52" )
VAR TotalGastos = GastosAdmon + GastosVenta
VAR U_Operativa = Utilidad_Bruta - GastosAdmon - GastosVenta
VAR Margen_operativo =
    DIVIDE ( U_Operativa, Ingresos )
VAR IngresosNoOperacionales =
    CALCULATE ( [Valor], BD[GRUPO] = "42" )
VAR GastosNoOperacionales =
    CALCULATE ( [Valor], BD[GRUPO] = "53" )
VAR UtilidadAntesImpo = U_Operativa + IngresosNoOperacionales - GastosNoOperacionales

RETURN
SWITCH (
    Rubro,
    "VENTAS", Ingresos,
    "COSTO DE VENTA", Costos,
    "UTILIDAD BRUTA", Utilidad_Bruta,
    "GASTOS ADMON", GastosAdmon,
    "GASTOS VENTAS", GastosVenta,
    "TOTAL GASTOS", TotalGastos,
    "UTILIDAD OPERACIONAL", U_Operativa,
    "% UTILIDAD OPERACIONAL", Margen_operativo,
    "INGRESOS NO OPERACIONALES", IngresosNoOperacionales,
    "GASTOS NO OPERACIONALES", GastosNoOperacionales,
    "UTILIDAD ANTES IMPTO", UtilidadAntesImpo,

    BLANK ()
)

```

También puedes encontrar en el modelo otras medidas como el análisis vertical o comparar resultados con períodos anteriores.



Nota: Si quieres avanzar más en estos temas financieros, te recomendamos nuestro libro Estados Financieros en Excel y Power BI.

Creo que podemos crear un libro de solo recetas y patrones en DAX, pero lo dejaremos para un próximo tomo, por el momento esto es lo que creemos puedes aplicar en tus informes.

Esperamos sigas avanzando y nos cuentes en nuestras redes que tal te ha parecido este primer volumen de DAX.

“Tarde que temprano la disciplina vencerá la inteligencia”

Proverbio Japones.

Espero hayas disfrutado tanto como nosotros en este primer tomo de DAX, estamos seguro de que si aplicas bien los conceptos y recomendaciones dadas en este libro, vas a recoger frutos muy pronto y llevaras tus análisis a otro nivel, te esperamos en el siguiente volumen, donde abarcaremos temas más profundos, pero no queremos despedirnos sin antes agradecerte tu inmenso apoyo y creer que material como este puede ayudarte en tu vida profesional.

¡GRACIAS MIL GRACIAS!

Si estas iniciando en Power BI o ya interactúas con la herramienta, pero aún te cuesta comprender los fundamentos bajo los cuáles están basadas las principales funciones de DAX, este libro es para ti. DAX es el lenguaje de programación de análisis de datos que opera Power BI y otras herramientas como Analysis Services y Power Pivot en Excel, que muchos de los que empiezan a desarrollar tableros tropiezan tratando de entender cálculos y medidas donde sus resultados son erróneos o simplemente se ejecutan sin entender todo lo que sucede internamente en el motor que opera DAX, si estás buscando una introducción simple al lenguaje DAX, comenzando desde cero y alcanzar un nivel básico de codificación en el mismo, este es el libro que necesitas. Con toda nuestra experiencia creamos un paso a paso donde creemos y estamos completamente seguros de que te llevarán a dominar los principales conceptos de DAX, adicionalmente estarás en la capacidad de crear indicadores de alto valor para la toma de decisiones.

Por eso no te preocupes si no tienes experiencia o nunca has comprendido cómo funciona DAX, en este libro quitaremos todo velo para que puedas percibir un panorama más claro de ahora en adelante al codificar tus propias fórmulas.

Nuestra intención al escribir este libro no es sólo que lo leas, sino que también encuentres en él un instrumento de apoyo para tu estudio del lenguaje, tratamos de incluir muchos ejemplos simples y prácticos para poner a prueba todos los conocimientos que irás adquiriendo a lo largo de todo el contenido. Si eres un usuario que apenas está familiarizándose con DAX te sugerimos que vayas en el orden en el que están escritos los capítulos, ya que saltarse a los temas finales sin haber digerido los más básicos puede llevar a una comprensión incompleta de los conceptos. Esperamos que puedas disfrutar de cada capítulo a la vez que obtienes todo este nuevo conocimiento.

DAX PARA TODOS

Volumen 1

