

Función RANKX

Compatibilidad




Microsoft Excel
Excel ≥ 2016

★★★★★

Power BI Desktop
PBI ≥ Nov 2016

★★★★★

SQL Analysis Services
SSAS ≥ 2012

DIRECTQUERY: C.Calculadas  Medidas 
ROW LEVEL SECURITY: 

Int. Contexto

Contexto de Filtro

Tiene en cuenta el contexto de filtro

★★★★★

Contexto de Fila

Ignora contextos de filas previos
Tiene en cuenta su contexto de fila

Categorías

Según Proceso Interno
Iteración

★★★★★

Según Resultado
Escalar

Recursos de Aprendizaje



MAGÍSTER EN LEGUAJE DAX

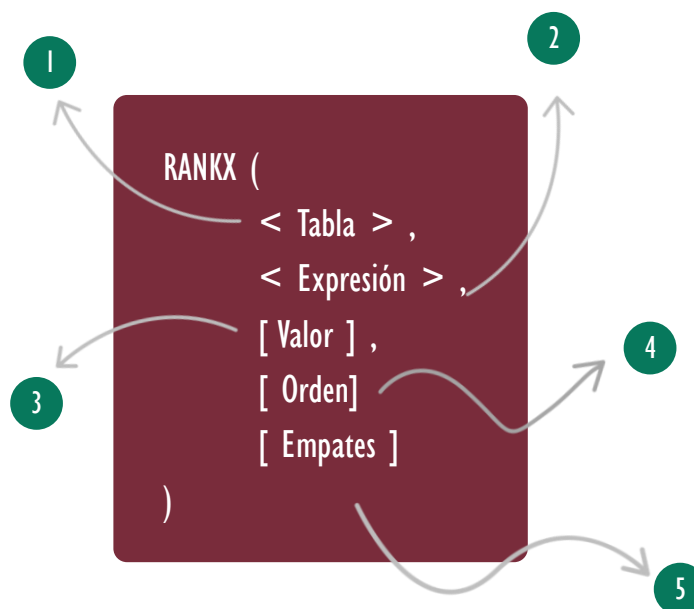
→ [Capacitación OnLine] ←

<https://bit.ly/3bzlkG0>

Descripción

La función **RANKX** retorna la clasificación (*1 a N*) de una expresión evaluada en el contexto original de acuerdo una *tabla de búsqueda (ordenada descendientemente por defecto)*, donde sus valores se crean mediante una expresión que se ejecuta fila a fila en la tabla del primer argumento.

Sintaxis



1 Tabla

Puede ser una referencia a una tabla nativa ejemplo: *Pedidos* o una expresión de tipo tabla, ejemplo: *ALL(Pedidos[País])*. Aquí es donde se ejecutará el recorrido fila a fila para determinar la tabla de búsqueda para el ranking.

 Iterador

 Expresión Tipo Tabla

Tipo

Obligatorio

Atributo

No Repetible 

2 Expresión

Una expresión escalar de tipo texto, aritmético, algebraico o booleana para ser evaluada en el recorrido fila a fila.

 Contexto de Fila

Tipo

Obligatorio

Atributo

No Repetible 

Videos Sobre ITERADORES



En el seminario grabado en directo se exponen los fundamentos de las funciones de iteración escalares, es decir, **SUMX**, **AVERAGEX**, **MEDIANX**, etc.

Con lo previo, en el seminario se crean medidas o cálculo de agregación con restricciones tipo O, tipo Y y combinaciones de los dos, donde, aprovechamos para ver las diversas formas de aplicar un O, con la función **OR**, el operador **|**, el operador **IN**, como implementar una operación tipo **NOTIN**, tanto con constructor de tabla y lista externa.

[→ Ver Video Aquí ←](#)

Capacitación Gratuita de DAX de 16 horas:

LEGAJE DAX DE CERO A GUERRERO

[VER AQUÍ](#)

3 Valor

Una expresión escalar que es evaluada en el contexto de evaluación original, cuyo resultado es el *valor de búsqueda* para determinar su posición o ranking en la *tabla de búsqueda* mediante el criterio: $Valor\ de\ Búsqueda \geq Valor\ Actual\ en\ la\ Tabla\ de\ Búsqueda$ si la ordenación de la tabla de búsqueda es descendente (**DESC**) o $Valor\ de\ Búsqueda \leq Valor\ Actual\ en\ la\ Tabla\ de\ Búsqueda$ si la ordenación de la tabla de búsqueda es ascendente (**ASC**).

Si este valor es omitido la expresión del segundo argumento es evaluada en el contexto de evaluación original y será el *valor de búsqueda*.

Tipo

Opcional

Atributo

No Repetible 

4 Orden

Orden para aplicar la tabla de búsqueda: **DESC** o **FALSE** o **0** para ordenación descendente. **ASC** o **TRUE** o **1** para ordenación ascendente. Si se omite le orden es **DESC**.

Tipo

Opcional

Atributo

No Repetible 

5 Empates (Duplicados)

Comportamiento en caso de valores duplicados en la tabla de búsqueda. **SKIP**: mantiene valores duplicados contando sus posiciones. **DENSE**: elimina valores duplicados. Si se omite utiliza **SKIP**.

Tipo

Opcional

Atributo

No Repetible 

 Valor Que Retorna 

Un valor único escalar de tipo **INTEGER** que representa el valor de la clasificación (1 a N) a todos los valores de la expresión evaluada en la tabla.

Para comprender la función **RANK** es necesario tener claro su algoritmo interno, aunque sigue los principios de las funciones de iteración, se debe tener presente un par detalles para no tener confusiones con su semántica.

1

ALGORITMO DE RANKX

i. La función **RANKX** construye una tabla de búsqueda iterando en la tabla de *primer argumento* y generando en cada iteración el valor asociado a la expresión del *segundo argumento*, luego los valores son ordenados descendientemente (**DESC**) por defecto, y ningún otro elementos o columna es heredado en la *tabla de búsqueda*.

• Ejemplo:

$$1. \text{ RankingDelIngresos} = \text{RANKX} (\text{ALL} (\text{Pedidos}[\text{País}]), [\text{Ingresos Tot}])$$

Tabla de Búsqueda

1	381 278
2	380 170
3	375 095
4	374 136
5	373 248
6	372 727
7	362 302
8	353 314
9	347 945

- 1 Por cada fila de la tabla de primer argumento: *ALL (Pedidos[País])* se evalúa la expresión del segundo argumento: *[Ingresos Tot]*, que internamente es *CALCULATE (SUM (Pedidos[Ingresos]))*, donde el *CALCULATE* es implícito por referenciación a una medida.
- 2 Luego de que todos los valores son generados, los mismos son ordenados descendientemente (*mayor a menor*), nótese que al final no queda la columna país ni nada similar (*los números 1 a 9 de la ilustración de la izquierda es una referencia visual para el ranking o posición*).
- 3 Si queremos que la table de búsqueda quede ordenado ascendentemente (*menor mayor*), debemos señalar *ASC* o *TRUE* o *1* en el cuarto argumento de la función *RANKX*, así: *RANKX (ALL (Pedidos[País]), [Ingresos Tot], , ASC)*

ii. La Función **RANKX** toma como *valor de búsqueda* el resultado de evaluar la expresión del segundo argumento en el contexto original.

Tomando la expresión del inciso anterior:

$$1. \text{ RankingDelIngresos} = \text{RANKX} (\text{ALL} (\text{Pedidos}[\text{País}]), [\text{Ingresos Tot}])$$

Y analizando en la matriz la celda correspondiente a Colombia, entonces el valor de búsqueda será los ingresos para Colombia, que para el caso corresponde a: 372 727.

Si se desear tomar otra expresión como *valor de búsqueda*, esta se puede señalar en el tercer argumento de la función **RANKX**, por ejemplo:

$$1. \text{ RankingDelIngresos} = \text{RANKX} (\text{ALL} (\text{Pedidos}[\text{País}]), [\text{Ingresos Tot}], [\text{Costos Tot}] * 9)$$

2

ALGORITMO DE RANKX

iii. Para determinar la posición o ranking la función **RANKX** ejecuta una pregunta que depende del orden de la *tabla búsqueda*, así:

a) Si el orden es descendente (*DESC* valor por defecto) la pregunta es:

Valor de Búsqueda ≥ Valor Actual en la Tabla de Búsqueda

Iniciando desde la primera fila de la *tabla de búsqueda*, y devuelve la posición donde la preguntara retorne verdadero (**TRUE**) por primera vez.

Si la condición nunca una se cumple, retorna: *posición máxima + 1*.

• Ejemplo:

1. **RankingDelIngresos** = **RANKX** (**ALL** (Pedidos[País]), [Ingresos Tot])

Que es lo mismo que

1. **RankingDelIngresos** = **RANKX** (**ALL** (Pedidos[País]), [Ingresos Tot], , **DESC**)

b) Si el orden es ascendente (*ASC*) la pregunta es:

Valor de Búsqueda ≤ Valor Actual en la Tabla de Búsqueda

Iniciando desde la primera fila de la *tabla de búsqueda*, y devuelve la posición donde la preguntara retorne verdadero (**TRUE**) por primera vez.

Si la condición nunca una se cumple, retorna: *posición máxima + 1*.

• Ejemplo:

1. **RankingDelIngresos** = **RANKX** (**ALL** (Pedidos[País]), [Ingresos Tot], , **ASC**)

Si queremos posiciones continuas **1, 2, 3...** indicamos **DENSE** en el quinto argumento, sino señalamos **SKIP** que es el valor por defecto.

1. **RankingDelIngresos** = **RANKX** (**ALL** (Pedidos[País]), [Ingresos Tot], , **ASC**, **DENSE**)

Véase el ejemplo 7 para más detalles.

Ejemplos

- Ejemplo 1 – Parámetro de Tabla Sencillo:

Para el modelo de DISPRODUCTOS LD encontramos una matriz con los países en el área de filas. Véase la página Ej-1 en el archivo .pbix asociado.

- ➔ Crear una medida para ser arrastrada a la matriz que muestre a cuál posición (como número entero) corresponde los ingresos del país que se lee en la matriz.

```

1. RankingDelIngresos = -- Estudie la presente expresión en compañía
2. RANKX ( -- Del algoritmo de RANKX de la página previa.
3.     ALL ( Pedidos[País] ),
4.     [Ingresos Tot]
5. )
    
```

Se debe utilizar **ALL** en el primer argumento de **RANK** dado que se debe generar los valores de todos los países para la tabla de búsqueda.

Una expresión que implemente **VALUES** o en general cualquier función que tenga en cuenta el contexto de filtro como **DISTINCT**, genera una tabla con un único elemento (País), por lo tanto, si no se utiliza el argumento *Valor* siempre devolverá uno, pues el único valor de la tabla de búsqueda se corresponde de forma idéntica con el valor de búsqueda.

- Ejemplo 2 – Parámetro de Tabla con Más Elementos:

El primer parámetro (*Tabla*) como se ha corroborado en el ejemplo anterior admite funciones de tabla, y puede ser una expresión tan sencilla o compleja como necesitemos.

- ➔ Por ejemplo, si deseamos el ranquin para la matriz de países pero que la tabla de búsqueda sólo tenga en cuenta para la tabla de búsqueda los valores de los países SEDES que corresponde a: Argentina, Colombia, Ecuador, Perú y Uruguay.

Un solución sería:

“ La presente ficha técnica cuenta con un archivo de Power BI asociado con el modelo de Disproductos, donde podrás ver cada medida y su expresión DAX. Descargar CartasDAX.Com ”

↓ Descargar Aquí



www.CartasDAX.Com

Los ejemplos 3, 4 y 5 utilizan diversas funciones alguna de ellas funciones de manipulación de tabla, cada una cuenta con su ficha técnica, además, la web www.CartasDAX.Com te brinda la opción de explorar las funciones por categorías.

Por lo que, si deseas explorar todas las funciones de manipulación, de tablas encontraras una página dedicada a ello:

→ Explorar Funciones de Tabla ←

Más Recursos de Aprendizaje



MAGÍSTER EN LENGUAJE DAX

Tres Niveles: Cada uno 33 horas

→ [Visitar Curso] ←

La capacitación **Magíster en Lenguaje DAX** es una capacitación brindada de forma presencial o virtual, la cual consta de **3 niveles, cada uno de 33 horas + sesiones de monitoria.**

En el magíster en lenguaje DAX se revelan **todos los secretos del lenguaje DAX** acompañado de una metodología altamente visual y aplicada, que se ha desarrollado y mejorado durante casi una década.

Si deseas dominar el arte marcial del **análisis de datos** para la tecnología número 1 en el mercado en inteligencia de negocios (self-services BI), esta es la capacitación correcta en tu idioma.

→ <https://bit.ly/3bzlkG0> ←

```

1. RankingDelIngresosSEDES =
2. RANKX (
3.     INTERSECT (
4.         ALL ( Pedidos[País] ),
5.         {
6.             "Argentina",
7.             "Colombia",
8.             "Ecuador",
9.             "Perú",
10.            "Uruguay"
11.        }
12.    ),
13.    [Ingresos Tot]
14. )
    
```

• Ejemplo 3 – Parámetro Expresión:

El argumento expresión tiene dos grandes papeles en la función **RANKX**, uno de ellos como se señala en el algoritmo para determinar los valores de la *tabla de búsqueda* mientras se itera en la tabla del primer argumento

☛ *Esta expresión en la gran mayoría de medidas debe ejecutar transición de contextos, de lo contrario los valores para la tabla de búsqueda se repetirán, por lo que, si no contamos la medida previamente para referenciar a ello y apoyarnos en el CALCULATE implícito, entonces, debemos encerrar la expresión en CALCULATE explícitamente.*

Por ejemplo, el ranquin respecto a las unidades vendidas

```

1. RankingDeUnidades =
2. RANKX (
3.     ALL ( Pedidos[País] ),
4.     CALCULATE (
5.         SUM ( Pedidos[Unidades] )
6.     )
7. )
    
```


Seminario Sobre Variables en DAX



La solución al ejemplo 5 puede parecer ser de alta complejidad, no obstante, es sencilla, pues es extensa en número de líneas, pero simple en su lógica de construcción.

La solución del ejemplo 5 utiliza variables para dividir en pasos el problema, con el apoyo de las funciones de texto del lenguaje DAX.

El estudio de variables es un aspecto fundamental, por lo que, si no tienes conocimiento de ellas o las manipulas a un nivel básico, es recomendable que dediques un espacio para estudiar la grabación 100% gratuita del seminario: [Guía Definitiva a Variables en Lenguaje DAX](#).

[→ Ver Vídeo Aquí ←](#)

• Ejemplo 4 – Parámetro Valor:

El valor de búsqueda no tiene que derivar del segundo argumento ya que puede ser cualquier otra expresión e incluso un valor constante.

☛ *Por ejemplo: Queremos el ranquin de los ingresos, sin embargo, cuando el país sea Colombia en la matriz no queremos sean los ingresos determinados por la medida [Ingresos Tot] si no el valor 500 000.*

Una alternativa de solución sería:

```

1. RankingDelIngresosConVariacionEnColombia =
2. RANKX (
3.     ALL ( Pedidos[País] ),
4.     [Ingresos Tot],
5.     IF (
6.         SELECTEDVALUE ( Pedidos[País] ) = "Colombia",
7.         500000,
8.         [Ingresos Tot]
9.     )
10. )
    
```

Es importante destacar que para esta medida tendríamos dos países con el valor 1, *Argentina* que es el país que más genero ingresos y *Colombia* que se le ha asignado el valor de 500000 un ingreso aún más grande que el de *Argentina*.

• Ejemplo 5 – Parámetro Tabla, Expresión y Valor:

Combinando nuestro conocimiento del *argumento valor*, podremos notar que es posible utilizar una tabla de búsqueda personalizada para ejecutar el ranking respecto a un *valor expresión* deseado,

☛ *Por ejemplo: Queremos el ranquin de los ingresos respecto a la siguiente tabla:*

```

1. { 500000, 380000, 370000, 360000, 350000 }
    
```

MAGÍSTER EN LENGUAJE DAX — NIVEL 2

La capacitación **Magíster en Lenguaje DAX del equipo de Excel Free Blog** cuenta con tres niveles, cada uno de 30 horas.

El nivel más popular es el 1, no obstante, si cuentas con un conocimiento amplio en los fundamentos, como: **contexto de fila, contexto de filtro, funciones de iteración, CALCULATE avanzado, transición de contextos y manejo de tablas de calendario con funciones de inteligencia de tiempo**, entonces, el nivel 2 es para ti, ya que parte de dicho conocimiento y explora otras temáticas y aplicaciones de carácter avanzado.

[\[Más Información Aquí\]](#)

La solución con **RANKX** y apoyado de la sintaxis de los constructores de tabla sería:

```

1. RankingConTablaPersonalizada =
2. RANKX (
3.     {
4.         500000,
5.         380000,
6.         370000,
7.         360000,
8.         350000
9.     },
10.    [Value], -- Value es el nombre de la columna
11.    IF (
12.        SELECTEDVALUE ( Pedidos[País] ) = "Colombia",
13.        500000,
14.        [Ingresos Tot]
15.    )
16. )
    
```

La construcción de la tabla personalizada de búsqueda no tiene que ser necesariamente con la sintaxis de los constructores, puede ser con funciones como **DATATABLE**, **ROW/UNION** y hasta creada con Power Query y/o Lenguaje M, incluso cargada de forma externa.

• Ejemplo 6 — Parámetro Orden

Hasta el momento con la función **RANKX** hemos conseguido indicar la posición del valor más grande al más pequeño según la tabla de búsqueda, es decir, el o los valores más grandes tendrá la posición 1.

No obstante, es posible señalar en el cuarto argumento de **RANKX** que el orden de la tabla de búsqueda se de menor a mayor, es decir, ascendentemente, esto se logra bien sea indicado: **ASC**, **TRUE** o **1** de forma explícita en el parámetro, ya que el valor por defecto es **DESC**.

☛ *Por ejemplo: Queremos la clasificación de los ingresos para cada país del más pequeño al más grande, es decir, el menor valor tendrá la posición 1, esto se consigue así:*



```

1. RankingDelIngresosIverso =
2. RANKX (
3.   ALL ( Pedidos[País] ),
4.   [Ingresos Tot],
5.   , -- NO es obligatorio señalar el tercer argumento.
6.   ASC
7. )
    
```

• Ejemplo 7 – Parámetro Empate/Ties (Valores Duplicados):

El quinto y último argumento de la función **RANKX** es útil en presencia de empates en los valores de la tabla de búsqueda.

Último Argumento de RANKX: SKIP y DENSE

1 Imaginemos una expresión hipotética para una medida con **RANKX** donde el resultado de la tabla de búsqueda es el siguiente:

1	380 000
2	380 000
3	380 000
4	370 000
5	370 000
6	370 000
7	360 000
8	350 000
9	350 000

2 Tomemos como valor de búsqueda: **372 000**, dado que es orden descendente la pregunta de **RANKX** es:

Valor de Búsqueda ≥ Valor Actual En La Tabla de Búsqueda

Entonces el resultado sería: **4**, si bien las **3** primeras posiciones corresponden al mismo valor, estas son tenidas en cuenta para el ranking como se observa en la ilustración de la tabla de búsqueda

3 El comportamiento descrito es la ejecución por defecto de **RANKX**, que está asociado al quinto argumento como **SKIP**.

4 El Si queremos que la tabla de búsqueda elimine duplicados, de tal forma que luzca así:

1	380 000
2	370 000
3	360 000
4	350 000

Por consiguiente, valor de búsqueda: **372 000** retorne: **2**

5 Esto se consigue señalando, en el último argumento de la función **RANKX**: **DENSE**.

SKIP hace referencia a que se omitirán los rangos que correspondan a elementos en empates. **DENSE**: todos los elementos de un empate se cuentan como uno, es decir, **DENSE** aplica **DISTINCT** en la tabla de búsqueda.

Para lograr valores repetidos en el modelo de Disproducto implementaremos la tabla con un constructor.



```

1. RankingDENSE =
2. RANKX (
3.     {
4.         380000,
5.         380000,
6.         380000,
7.         370000,
8.         370000,
9.         370000,
10.        360000,
11.        350000,
12.        350000
13.    },
14.    [Value],
15.    [Ingresos Tot],
16.    ,
17.    DENSE -- DENSE elimina los duplicados en la tabla de búsqueda
18. )        -- SKIP (que es el valor por defecto) mantiene los duplicados
    
```

• Ejemplo 8 – Manipulando en Total General

El total general como clasificación puede ser confuso, por ello una solución recurrente es utilizar [HASONEVALUE](#).

```

1. RankingTotalGeneral =
2. IF (
3.     HASONEVALUE ( Pedidos[País] ),
4.     RANKX (
5.         ALL ( Pedidos[País] ),
6.         [Ingresos Tot]
7.     )
8. )
    
```

En la práctica es buen hábito siempre utilizar [IF](#) y [HASONEVALUE](#) con [RANKX](#), puesto que sin ello tenemos el peligro de que arroje resultados inesperados, concretamente cuando se segmenta sin el campo utilizando en el segundo argumento de [RANKX](#).

Acerca de las Cartas DAX



Las cartas DAX del equipo de **Excel Free Blog** es un paquete de contenido de documentación y representación para un juego de todas las funciones en lenguaje DAX, compuesta por dos partes:

I. La Carta

Cada función en todo el lenguaje DAX contará con un **personaje representativo**, por ejemplo, la función SUMX será representada por el ser mitológico: el grifo.

II. La Ficha Técnica

La ficha técnica tiene **información de la función** para su manejo, consulta y entendimiento, en ella se documenta y explica: Descripción, sintaxis, parámetros y más. (Cómo la presente)

Más Información

→ <https://bit.ly/3aZiBqu> ←
→ www.CartasDax.Com ←

Última Actualización:
28 de mayo del 2021

Observaciones

- I. Si el argumento en expresión o valor en **RANKX** al ser evaluados retornan **BLANK**, entonces son tratados como **0** para todas las expresiones numéricas.
- II. Los argumentos opcionales se pueden saltar en su totalidad, simplemente se debe indicar la coma respectiva (*o separador de lista predeterminado*) y continuar con los argumentos necesarios.
- III. La función **RANK.EQ** en DAX es como su equivalente en Excel, retorna el ranking de acuerdo con una lista proporcionada, sin embargo, ofrece menos opciones con **RANKX**, por lo que es utilizada muy rara vez, puesto que **RANKX** es una opción mucho más potente, **RANK.EQ** se mantiene por compatibilidad.

BIBLIOGRAFÍA

Páginas Web:

- 1. DAX GUIDE: <https://dax.guide/rankx/>
- 2. MICROSOFT: <https://docs.microsoft.com/en-us/dax/rankx-function-dax>
- 3. SQLBI: <https://www.sqlbi.com/articles/use-of-rankx-in-power-bi-measures/>
- 4. SQLBI: <https://www.sqlbi.com/articles/displaying-nth-element-in-dax/>

Libros:

- Definitive Guide To DAX (2nd Edition) — Marco Russo y Alberto Ferrari [↗](#)
- Practical PowerPivot & DAX Formulas — Art Tennick [↗](#)

Creado por:

Miguel Caballero y Fabian Torres.

Cualquier Retroalimentación:

excelfreebymcs@gmail.com

Funciones Relacionadas:



RANK.EQ