EPICODE

# Power BI

## Giorno 3

# Agenda

**Cosa vedremo**

- ❏ Introduzione alla DAX
- ❏ Misure
- ❏ Colonne Calcolate

# Agenda

**Obiettivi di apprendimento**

- ❏ Scrivere semplici espressioni DAX
- ❏ Saper scegliere quando implementare una colonna o misura
- ❏ Ottimizzare il report

EPICODE

# Introduzione alla DAX

## Che cos'è la DAX?

DAX è l'acronimo di Data Analysis Expressions.

La DAX è un insieme di funzioni, sviluppato da Microsoft, utile a calcolare:
- Colonne calcolate
- Tabelle calcolate
- Misure

**Qual è la differenza tra misura calcolata e colonna calcolata?**

La colonna calcolata è un'estensione di una tabella:
- Crea un valore per ogni riga della tabella
- I valori sono memorizzati nel file .pbix

La misura è un'espressione logica!
- Il risultato della misura dipende dai filtri applicati (contesto)
- La misura è un'aggregazione
- La misura è on-demand per cui viene eseguita quando 'richiesta'

## Misura


L'output della misura **TotalSales** dipende dal contesto cioè dai filtri applicati.

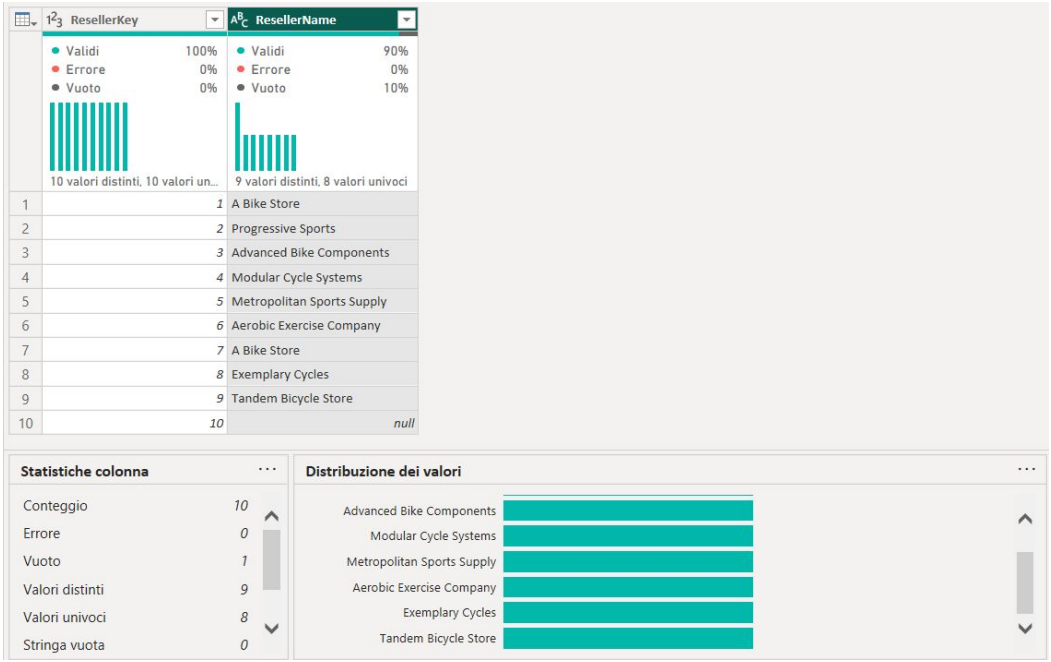
Filtri impliciti:
- Month

Filtri espliciti:
- Year
- Category

In questo caso, si parla di **filter-context!**

```
TotalSales =
SUM(Sales[SalesAmount])
```

## …altri esempi di misure di aggregazione e funzioni DAX: COUNTROWS



| | ResellerKey | ResellerName |
|---|---|---|
| | Validi 100% | Validi 90% |
| | Errore 0% | Errore 0% |
| | Vuoto 0% | Vuoto 10% |
| | 10 valori distinti, 10 valori un… | 9 valori distinti, 8 valori univoci |
| 1 | 1 | A Bike Store |
| 2 | 2 | Progressive Sports |
| 3 | 3 | Advanced Bike Components |
| 4 | 4 | Modular Cycle Systems |
| 5 | 5 | Metropolitan Sports Supply |
| 6 | 6 | Aerobic Exercise Company |
| 7 | 7 | A Bike Store |
| 8 | 8 | Exemplary Cycles |
| 9 | 9 | Tandem Bicycle Store |
| 10 | 10 | null |



La profilazione della *Reseller* tabella in figura ci fornisce queste informazioni:

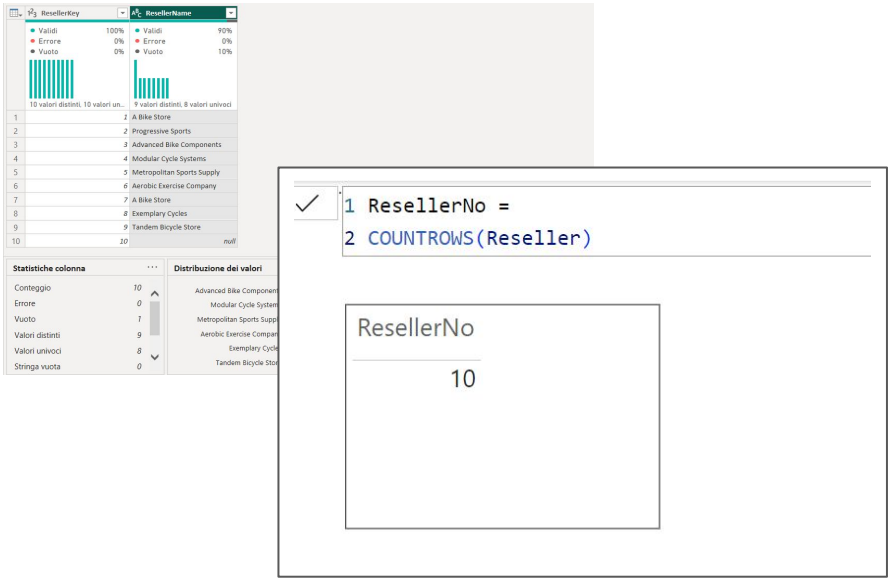
- il numero di record della tabella è 10
- il numero di record vuoti è 1
- il numero di record univoci è 8
- il numero di record diversi (distinti) è 9

Supponiamo di dover implementare una misura DAX per esporre in un report il numero totale di reseller presente in anagrafica

```
ResellerNo =
COUNTROWS(Reseller)
```

La funzione COUNTROWS conta il numero di record nella tabella specificata.

## …altri esempi di misure di aggregazione e funzioni DAX: COUNTROWS




La profilazione della *Reseller* tabella in figura ci fornisce queste informazioni:

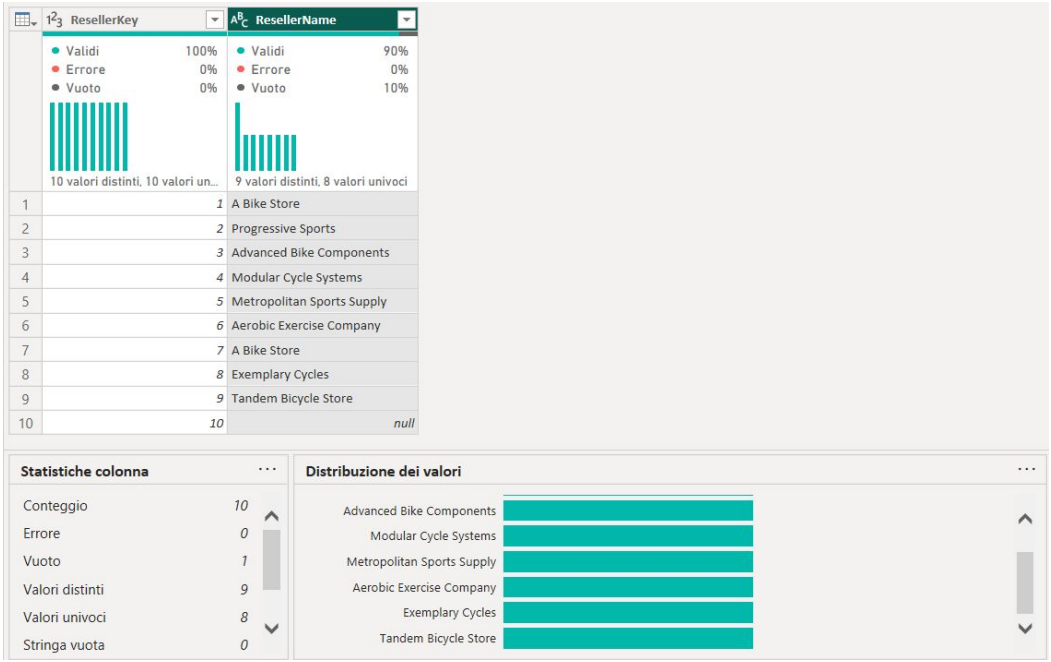
- il numero di record della tabella è 10
- il numero di record vuoti è 1
- il numero di record univoci è 8
- il numero di record diversi (distinti) è 9

Supponiamo di dover implementare una misura DAX per esporre in un report il numero totale di reseller presente in anagrafica

```
ResellerNo =
COUNTROWS(Reseller)
```

La funzione COUNTROWS conta il numero di righe nella tabella specificata.

## …altri esempi di misure di aggregazione e funzioni DAX: COUNTBLANK



| | ResellerKey | ResellerName |
|---|---|---|
| | Validi 100% | Validi 90% |
| | Errore 0% | Errore 0% |
| | Vuoto 0% | Vuoto 10% |
| | 10 valori distinti, 10 valori un… | 9 valori distinti, 8 valori univoci |
| 1 | 1 | A Bike Store |
| 2 | 2 | Progressive Sports |
| 3 | 3 | Advanced Bike Components |
| 4 | 4 | Modular Cycle Systems |
| 5 | 5 | Metropolitan Sports Supply |
| 6 | 6 | Aerobic Exercise Company |
| 7 | 7 | A Bike Store |
| 8 | 8 | Exemplary Cycles |
| 9 | 9 | Tandem Bicycle Store |
| 10 | 10 | null |



La profilazione della *Reseller* tabella in figura ci fornisce queste informazioni:

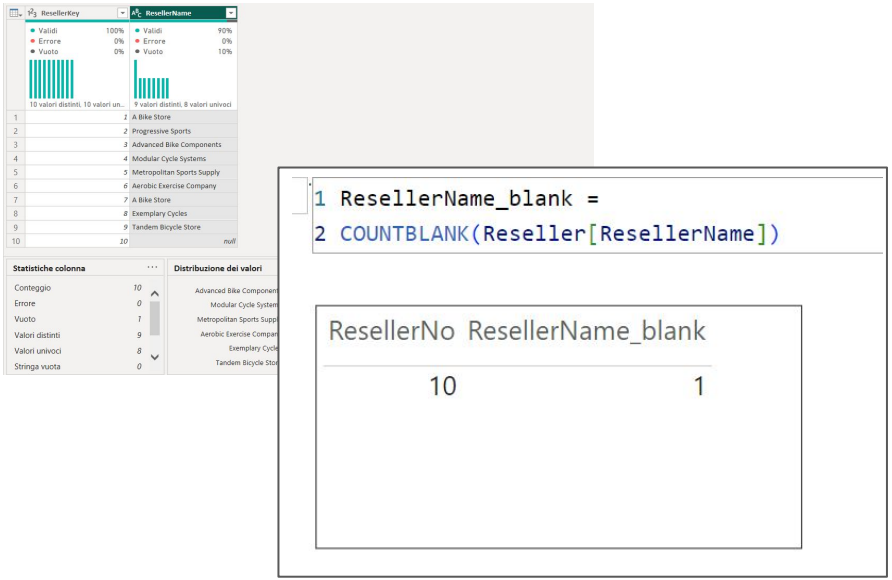
- il numero di record della tabella è 10
- il numero di record vuoti è 1
- il numero di record univoci è 8
- il numero di record diversi (distinti) è 9

Supponiamo di dover implementare una misura DAX per esporre in un report il numero totale di record della colonna *ResellerName* vuoti

```
ResellerNo =
COUNTBLANK(Reseller[ResellerName])
```

La funzione COUNTBLANK conta il numero di celle vuote presenti in una colonna

## …altri esempi di misure di aggregazione e funzioni DAX: COUNTBLANK




La profilazione della *Reseller* tabella in figura ci fornisce queste informazioni:

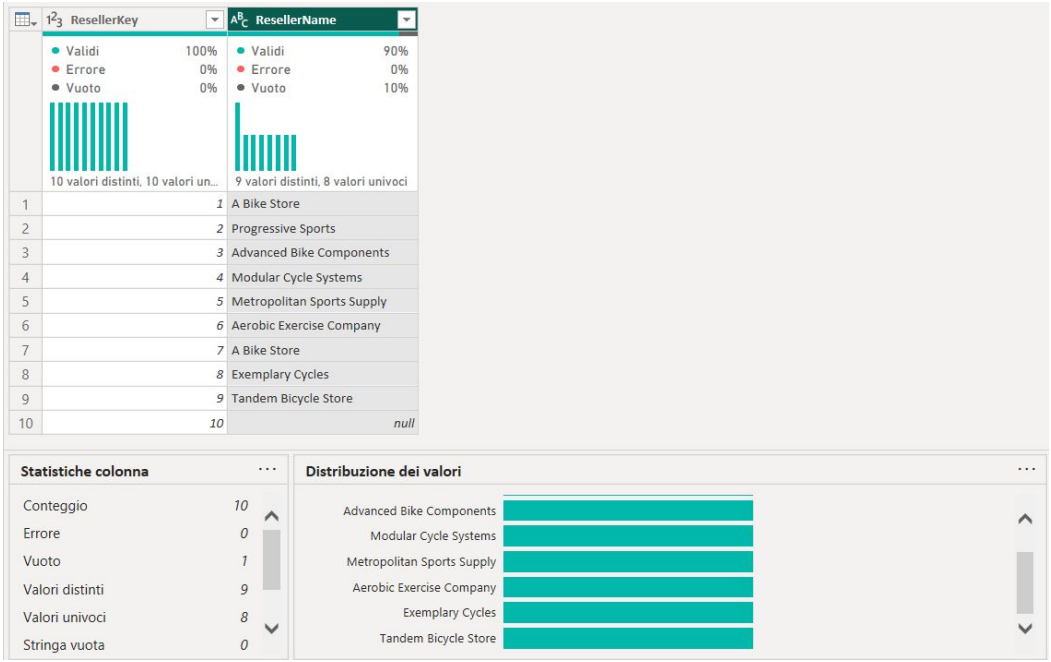
- il numero di record della tabella è 10
- il numero di record vuoti è 1
- il numero di record univoci è 8
- il numero di record diversi (distinti) è 9

Supponiamo di dover implementare una misura DAX per esporre in un report il numero totale di record della colonna *ResellerName* vuoti

```
ResellerNo =
COUNTBLANK(Reseller[ResellerName])
```

La funzione COUNTBLANK conta il numero di celle vuote presenti in una colonna

## …altri esempi di misure di aggregazione e funzioni DAX: COUNTA



| | ResellerKey | ResellerName |
|---|---|---|
| | Validi 100% | Validi 90% |
| | Errore 0% | Errore 0% |
| | Vuoto 0% | Vuoto 10% |
| | 10 valori distinti, 10 valori un… | 9 valori distinti, 8 valori univoci |
| 1 | 1 | A Bike Store |
| 2 | 2 | Progressive Sports |
| 3 | 3 | Advanced Bike Components |
| 4 | 4 | Modular Cycle Systems |
| 5 | 5 | Metropolitan Sports Supply |
| 6 | 6 | Aerobic Exercise Company |
| 7 | 7 | A Bike Store |
| 8 | 8 | Exemplary Cycles |
| 9 | 9 | Tandem Bicycle Store |
| 10 | 10 | null |



La profilazione della *Reseller* tabella in figura ci fornisce queste informazioni:

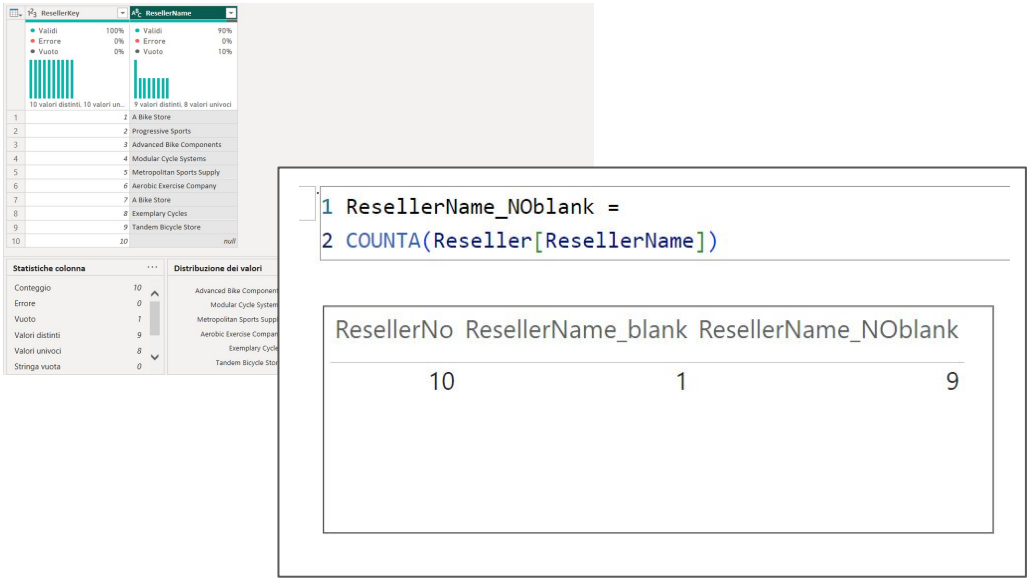
- il numero di record della tabella è 10
- il numero di record vuoti è 1
- il numero di record univoci è 8
- il numero di record diversi (distinti) è 9

Supponiamo di dover implementare una misura DAX per esporre in un report il numero totale di record della colonna *ResellerName* senza vuoti

```
ResellerNo =
COUNTA(Reseller[ResellerName])
```

La funzione COUNTA conta il numero di righe nella colonna specificata che contengono valori non vuoti

## …altri esempi di misure di aggregazione e funzioni DAX: COUNTA




La profilazione della *Reseller* tabella in figura ci fornisce queste informazioni:

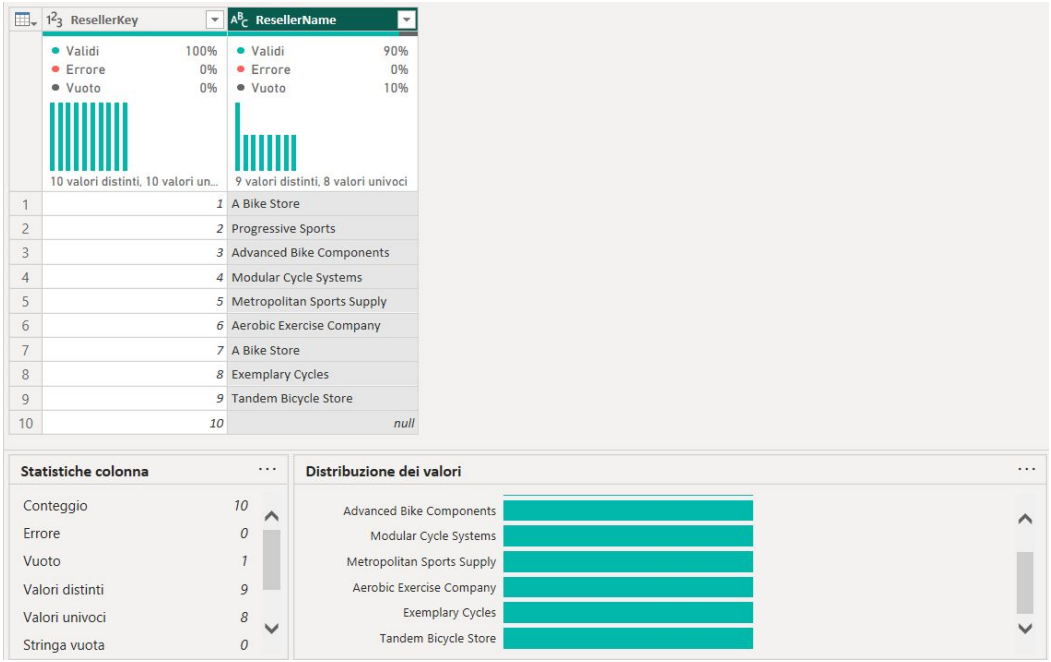
- il numero di record della tabella è 10
- il numero di record vuoti è 1
- il numero di record univoci è 8
- il numero di record diversi (distinti) è 9

Supponiamo di dover implementare una misura DAX per esporre in un report il numero totale di record della colonna *ResellerName* senza vuoti

```
ResellerNo =
COUNTA(Reseller[ResellerName])
```

La funzione COUNTA conta il numero di righe nella colonna specificata che contengono valori non vuoti

## …altri esempi di misure di aggregazione e funzioni DAX: DISTINCTCOUNT



| | ResellerKey | ResellerName |
|---|---|---|
| | Validi 100% | Validi 90% |
| | Errore 0% | Errore 0% |
| | Vuoto 0% | Vuoto 10% |
| | 10 valori distinti, 10 valori un… | 9 valori distinti, 8 valori univoci |
| 1 | 1 | A Bike Store |
| 2 | 2 | Progressive Sports |
| 3 | 3 | Advanced Bike Components |
| 4 | 4 | Modular Cycle Systems |
| 5 | 5 | Metropolitan Sports Supply |
| 6 | 6 | Aerobic Exercise Company |
| 7 | 7 | A Bike Store |
| 8 | 8 | Exemplary Cycles |
| 9 | 9 | Tandem Bicycle Store |
| 10 | 10 | null |



La profilazione della *Reseller* tabella in figura ci fornisce queste informazioni:

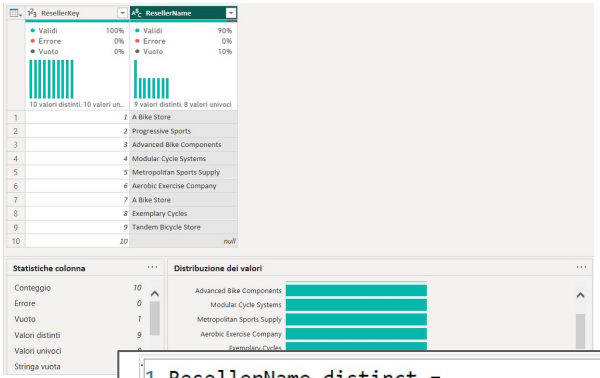
- il numero di record della tabella è 10
- il numero di record vuoti è 1
- il numero di record univoci è 8
- il numero di record diversi (distinti) è 9

Supponiamo di dover implementare una misura DAX per esporre in un report il conteggio dei diversi valori di *ResellerName*

```
ResellerNo =
DISTINCTCOUNT(Reseller[ResellerName])
```

La funzione DISTINCTCOUNT restituisce il numero di valori distinti della colonna specificata. La funzione DISTINCTCOUNT conta anche il BLANK.

## …altri esempi di misure di aggregazione e funzioni DAX: DISTINCTCOUNT




```
1 ResellerName_distinct =
2 DISTINCTCOUNT(Reseller[ResellerName])
```

| ResellerNo | ResellerName_blank | ResellerName_NOblank | ResellerName_distinct |
|---|---|---|---|
| 10 | 1 | 9 | 9 |

La profilazione della *Reseller* tabella in figura ci fornisce queste informazioni:

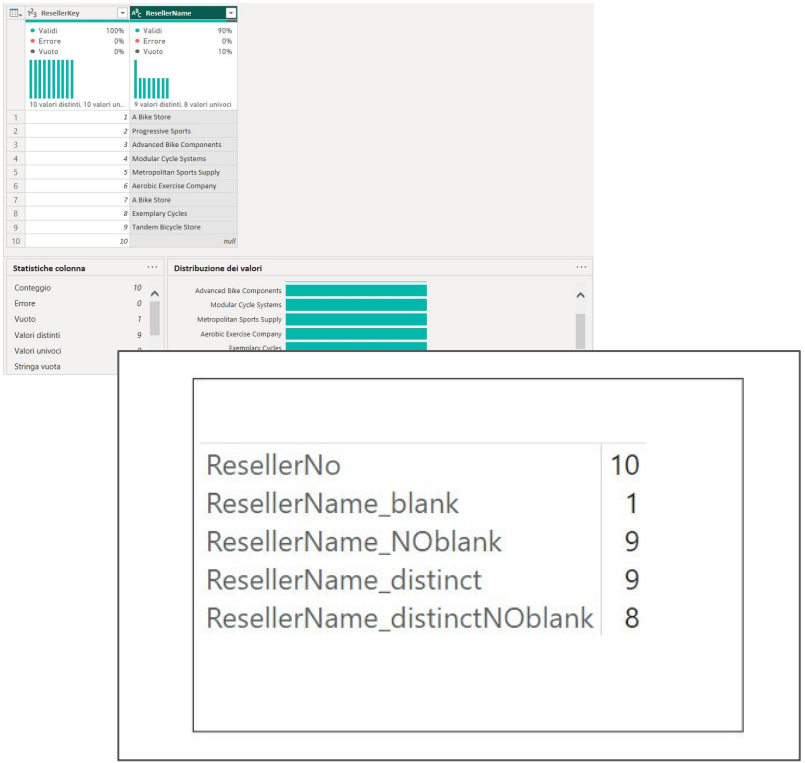
- il numero di record della tabella è 10
- il numero di record vuoti è 1
- il numero di record univoci è 8
- il numero di record diversi (distinti) è 9

Supponiamo di dover implementare una misura DAX per esporre in un report il conteggio dei diversi valori di *ResellerName*

```
ResellerNo =
DISTINCTCOUNT(Reseller[ResellerName])
```

La funzione DISTINCTCOUNT restituisce il numero di valori distinti della colonna specificata. La funzione DISTINCTCOUNT conta anche il BLANK.

## …altri esempi di misure di aggregazione e funzioni DAX: DISTINCTCOUNTNOBLANK




La profilazione della *Reseller* tabella in figura ci fornisce queste informazioni:

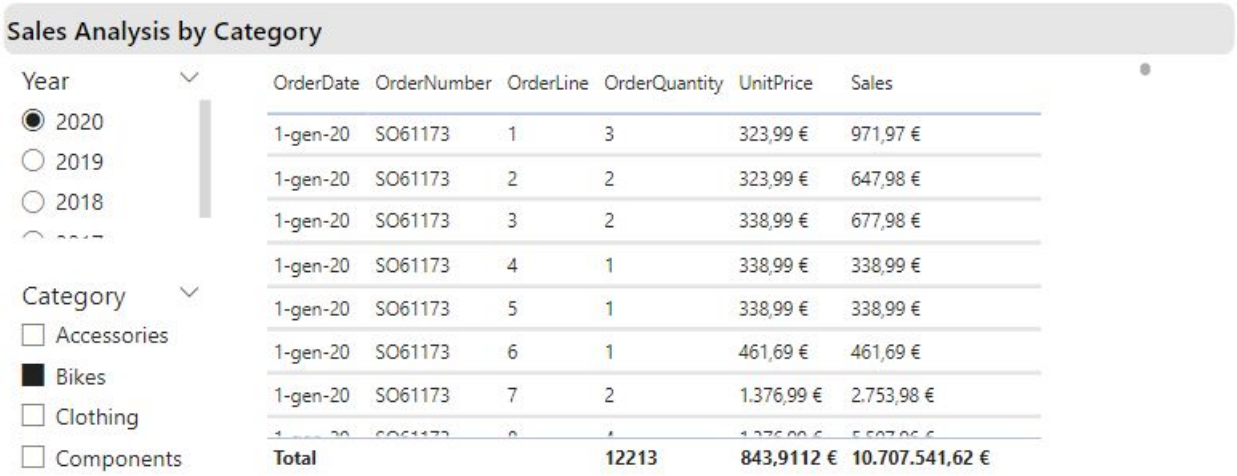
- il numero di record della tabella è 10
- il numero di record vuoti è 1
- il numero di record univoci è 8
- il numero di record diversi (distinti) è 9

Supponiamo di dover implementare una misura DAX per esporre in un report il conteggio dei diversi valori di *ResellerName* ignorando i BLANK.

```
ResellerNo =
DISTINCTCOUNTNOBLANK(Reseller[ResellerName])
```

La funzione DISTINCTCOUNT restituisce il numero di valori distinti della colonna specificata ignorando i BLANK.

## Colonna Calcolata




| OrderDate | OrderNumber | OrderLine | OrderQuantity | UnitPrice | Sales |
|---|---|---|---|---|---|
| 1-gen-20 | SO61173 | 1 | 3 | 323,99 € | 971,97 € |
| 1-gen-20 | SO61173 | 2 | 2 | 323,99 € | 647,98 € |
| 1-gen-20 | SO61173 | 3 | 2 | 338,99 € | 677,98 € |
| 1-gen-20 | SO61173 | 4 | 1 | 338,99 € | 338,99 € |
| 1-gen-20 | SO61173 | 5 | 1 | 338,99 € | 338,99 € |
| 1-gen-20 | SO61173 | 6 | 1 | 461,69 € | 461,69 € |
| 1-gen-20 | SO61173 | 7 | 2 | 1.376,99 € | 2.753,98 € |
| **Total** | | | **12213** | **843,9112 €** | **10.707.541,62 €** |

```
Sales =
Sales[OrderQuantity] * Sales[UnitPrice]
```

La colonna è un'estensione della tabella e, in quanto tale, scrive una valore row-by-row nella tabella stessa sulla quale è creata.

L'output dell'espressione è una nuova colonna denominata Sales.

L'espressione è valuta riga per riga.

In questo caso, si parla di **row-context!**

## X functions

```
1 Sales(measure) =
2 SUMX(
3     Sales
4     , Sales[OrderQuantity] * Sales[UnitPrice])
```

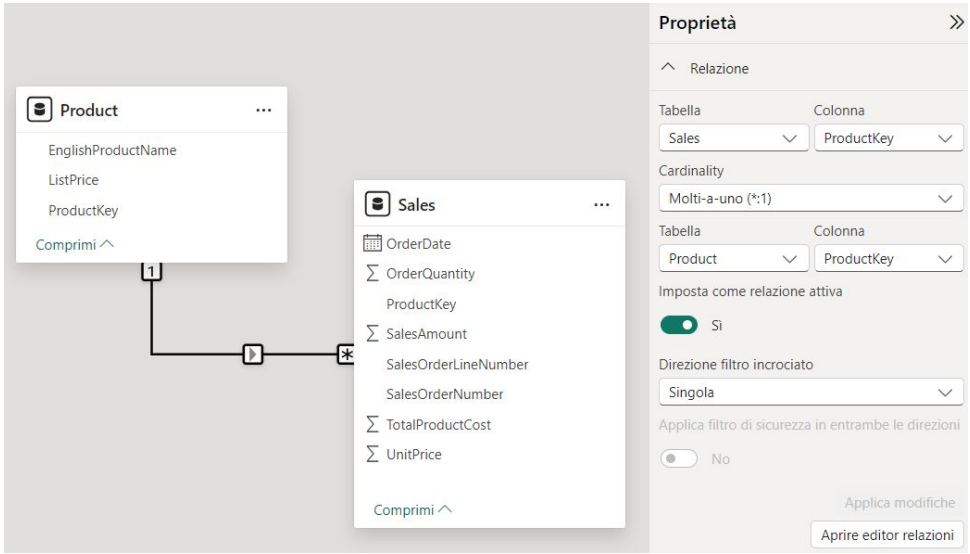
| SalesOrderNumber | SalesOrderLineNumber | Somma di OrderQuantity | Media di UnitPrice | Somma di Sales(columns) | Sales(measure) |
|---|---|---|---|---|---|
| SO43667 | 1 | 3 | 5,70 € | 17,1 € | 17,10 € |
| SO43667 | 2 | 1 | 2.039,99 € | 2.039,99 € | 2.039,99 € |
| SO43667 | 3 | 1 | 2.024,99 € | 2.024,99 € | 2.024,99 € |
| SO43667 | 4 | 1 | 2.024,99 € | 2.024,99 € | 2.024,99 € |
| **Totale** | | **6** | **1.523,92 €** | **6.107,07 €** | **6.107,07 €** |

In generale, bisognerebbe preferire le misure alle colonne calcolate.
La colonna calcolata scrive valori persistenti e, di conseguenza, contribuisce al volume dati e potrebbe inficiare l'aggiornamento del dataset.

Per inizializzare un contesto di riga mediante misure si possono utilizzare le *X functions*.

le ***X functions*** *consentono di iterare riga per riga una tabella (prima argomento della funzione) e valutare un'espressione (secondo argomento della funzione) riga per riga.*

## X functions

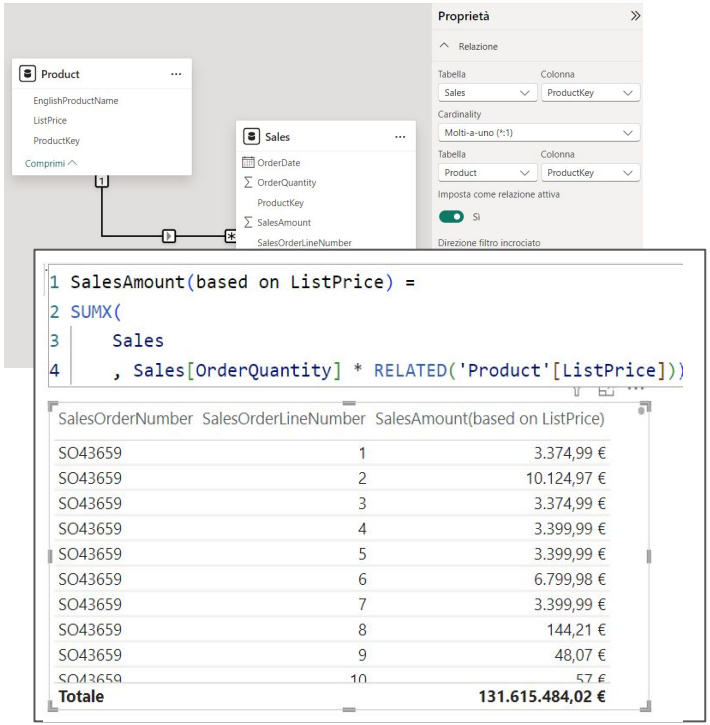



Supponiamo di avere uno schema come quello in figura. Data la cardinalità dei campi utilizzati per implementare la relazione, i filtri si propagano da *Product* verso *Sales*.

Dai requisiti emerge la necessità di calcolare l'importo totale di ciascuna transazione utilizzando il prezzo di listino (*ListPrice*) presente in anagrafica prodotto (*Product*).

Per soddisfare lo scopo, è necessario iterare riga per riga la tabella *Sales* (lato *n* della relazione) e per ciascuna ottenere il singolo valore di *ListPrice* dalla tabella *Product* correlata (lato *1* della relazione)

## X functions



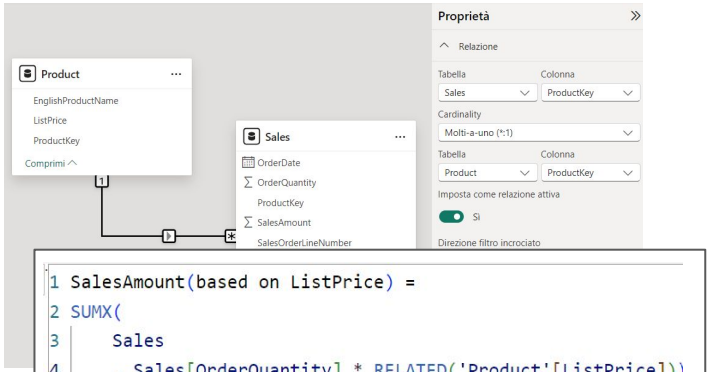

Dai requisiti emerge la necessità di calcolare l'importo totale di ciascuna transazione utilizzando il prezzo di listino (*ListPrice*) presente in anagrafica prodotto (*Product*).

Per soddisfare lo scopo, è necessario iterare riga per riga la tabella *Sales* (lato *n* della relazione) e per ciascuna ottenere il singolo valore di *ListPrice* dalla tabella *Product* correlata (lato *1* della relazione).

```
Sales =
SUMX(
      Sales
      , Sales[OrderQuantity] * RELATED(Product[ListPrice])
)
```

La funzione **RELATED** restituisce un valore correlato da un'altra tabella. L'argomento richiesto è la colonna contenente i valori che si vogliono recuperare.

# X functions




```
1 SalesAmount(based on ListPrice) =
2 SUMX(
3     Sales
4     , Sales[OrderQuantity] * RELATED('Product'[ListPrice]))
```

| SalesOrderNumber | SalesOrderLineNumber | SalesAmount(based on ListPrice) |
|---|---|---|
| SO43659 | 1 | 3.374,99 € |
| SO43659 | 2 | 10.124,97 € |
| SO43659 | 3 | 3.374,99 € |
| SO43659 | 4 | 3.399,99 € |
| SO43659 | 5 | 3.399,99 € |
| SO43659 | 6 | 6.799,98 € |
| SO43659 | 7 | 3.399,99 € |
| SO43659 | 8 | 144,21 € |
| SO43659 | 9 | 48,07 € |
| SO43659 | 10 | 57 € |
| **Totale** | | **131.615.484,02 €** |

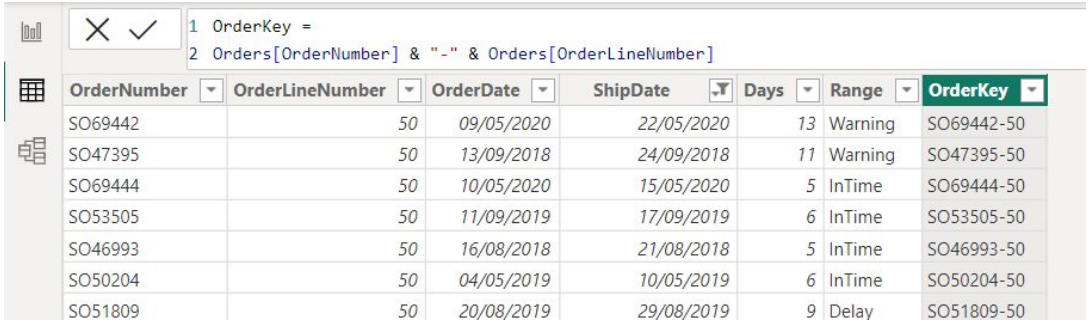
La funzione RELATED restituisce un valore correlato da un'altra tabella. L'argomento richiesto è la colonna contenente i valori che si vogliono recuperare.

Per la funzione RELATED è necessario che esista una relazione tra la tabella corrente (*Sales*) e la tabella con le informazioni correlate (*Product*). Si specifica la colonna che contiene i dati desiderati e la funzione naviga la relazione molti-a-uno esistente per recuperare il valore dalla colonna specificata (*Sales*) nella tabella correlata (*Product*). Se non esiste una relazione, è necessario crearne una.

RELATED itera la tabella dal lato molti e cerca il valore correlato in base alla relazione. Nell'esempio, cerca i valori della colonna *ProductKey* della tabella *Sales* nella colonna *ProductKey* di *Product*.

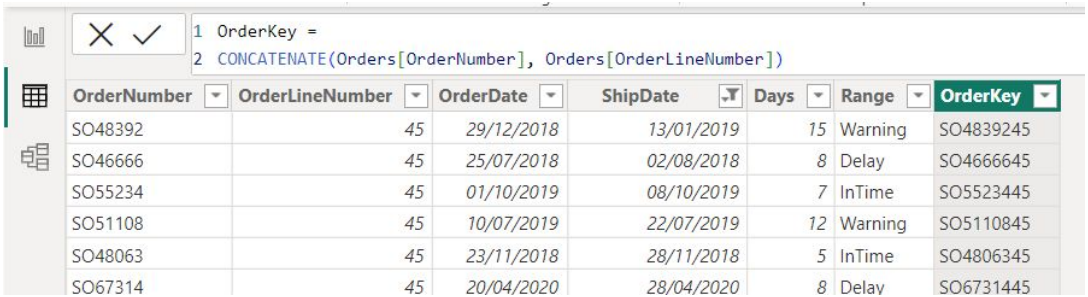
La funzione RELATED richiede un contesto di riga!

## Colonna Calcolata come campo chiave



```
1 OrderKey =
2 Orders[OrderNumber] & "-" & Orders[OrderLineNumber]
```

| OrderNumber | OrderLineNumber | OrderDate | ShipDate | Days | Range | OrderKey |
|---|---|---|---|---|---|---|
| SO69442 | 50 | 09/05/2020 | 22/05/2020 | 13 | Warning | SO69442-50 |
| SO47395 | 50 | 13/09/2018 | 24/09/2018 | 11 | Warning | SO47395-50 |
| SO69444 | 50 | 10/05/2020 | 15/05/2020 | 5 | InTime | SO69444-50 |
| SO53505 | 50 | 11/09/2019 | 17/09/2019 | 6 | InTime | SO53505-50 |
| SO46993 | 50 | 16/08/2018 | 21/08/2018 | 5 | InTime | SO46993-50 |
| SO50204 | 50 | 04/05/2019 | 10/05/2019 | 6 | InTime | SO50204-50 |
| SO51809 | 50 | 20/08/2019 | 29/08/2019 | 9 | Delay | SO51809-50 |



```
1 OrderKey =
2 CONCATENATE(Orders[OrderNumber], Orders[OrderLineNumber])
```

| OrderNumber | OrderLineNumber | OrderDate | ShipDate | Days | Range | OrderKey |
|---|---|---|---|---|---|---|
| SO48392 | 45 | 29/12/2018 | 13/01/2019 | 15 | Warning | SO4839245 |
| SO46666 | 45 | 25/07/2018 | 02/08/2018 | 8 | Delay | SO4666645 |
| SO55234 | 45 | 01/10/2019 | 08/10/2019 | 7 | InTime | SO5523445 |
| SO51108 | 45 | 10/07/2019 | 22/07/2019 | 12 | Warning | SO5110845 |
| SO48063 | 45 | 23/11/2018 | 28/11/2018 | 5 | InTime | SO4806345 |
| SO67314 | 45 | 20/04/2020 | 28/04/2020 | 8 | Delay | SO6731445 |

La funzione ***CONCATENATE*** in DAX accetta solo due argomenti, mentre la funzione *CONCATENA* di Excel accetta fino a 255 argomenti.

Se è necessario aggiungere altri argomenti, è possibile usare l'operatore e commerciale ***&***.

## Colonna Calcolata per ottenere filtri o range

Esempio di utilizzo della funzione ***DATEDIFF***

```
1 Days = DATEDIFF(Orders[OrderDate], Orders[ShipDate], DAY)
```

| OrderNumber | OrderLineNumber | OrderDate | ShipDate | Days |
|---|---|---|---|---|
| SO48063 | 45 | 23/11/2018 | 06/12/2018 | 13 |
| SO67314 | 45 | 20/04/2020 | 05/05/2020 | 15 |
| SO51157 | 45 | 26/07/2019 | 31/07/2019 | 5 |
| SO53567 | 45 | 22/09/2019 | 03/10/2019 | 11 |
| SO50727 | 45 | 21/06/2019 | 03/07/2019 | 12 |
| SO50259 | 45 | 18/05/2019 | 23/05/2019 | 5 |
| SO47975 | 45 | 03/11/2018 | 10/11/2018 | 7 |
| SO47718 | 45 | 26/10/2018 | 07/11/2018 | 12 |
| SO47450 | 45 | 27/09/2018 | 05/10/2018 | 8 |
| SO47395 | 45 | 13/09/2018 | 28/09/2018 | 15 |

```
1 Range =
2 IF(
3     Orders[Days] <= 7, "InTime"
4     , IF(AND(Orders[Days] > 7, Orders[Days] < 11), "Delay", "Warning"))
```

| OrderNumber | OrderLineNumber | OrderDate | ShipDate | Days | Range |
|---|---|---|---|---|---|
| SO53505 | 50 | 11/09/2019 | 21/09/2019 | 10 | Delay |
| SO46993 | 50 | 16/08/2018 | 28/08/2018 | 12 | Warning |
| SO50204 | 50 | 04/05/2019 | 15/05/2019 | 11 | Warning |
| SO51809 | 50 | 20/08/2019 | 26/08/2019 | 6 | InTime |
| SO47355 | 50 | 02/09/2018 | 17/09/2018 | 15 | Warning |
| SO47056 | 50 | 28/08/2018 | 11/09/2018 | 14 | Warning |
| SO47435 | 50 | 23/09/2018 | 28/09/2018 | 5 | InTime |

## Tabella Calendario

Disporre di una dimensione temporale è fondamentale per eseguire in maniera efficiente analisi nel tempo.
È possibile utilizzare due funzioni DAX per creare una tabella calendario:

CALENDAR(<start_date>, <end_date>)

*Restituisce una tabella con una sola colonna denominata Date. L'intervallo di date è compreso tra la data di inizio e di fine specificate.*

```
Calendar =
CALENDAR("01-01-2023", "31-12-2029")

Calendar =
CALENDAR(
    MIN(Sales[OrderDate])
    , MAX(Sales[OrderDate]))
```

CALENDARAUTO([fiscal_year_end_month])

*Restituisce una tabella con una sola colonna denominata Date. L'intervallo di date viene calcolato in automatico in base ai dati nel modello.*

```
Calendar =
CALENDARAUTO()

Calendar =
CALENDARAUTO(6)*
```

**esercizio fiscale 1-luglio, 30 Giugno*

## Tabella Calendario

... per completare la tabella calendario è necessario aggiungere colonne calcolate

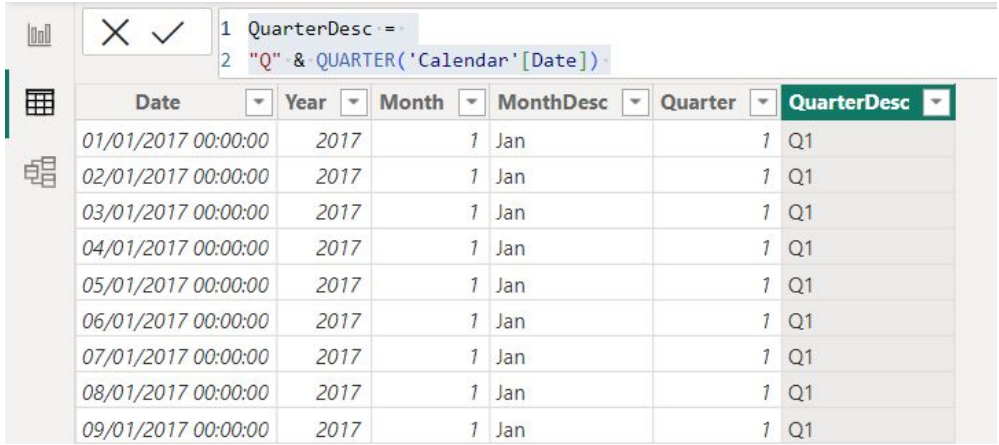
```
Year =
YEAR('Calendar'[Date]),

Quarter =
QUARTER('Calendar'[Date])

QuarterDesc =
"Q" & QUARTER('Calendar'[Date])

Month =
MONTH('Calendar'[Date])

MonthDesc =
FORMAT('Calendar'[Date], "MMM")
```




| Date | Year | Month | MonthDesc | Quarter | QuarterDesc |
|---|---|---|---|---|---|
| 01/01/2017 00:00:00 | 2017 | 1 | Jan | 1 | Q1 |
| 02/01/2017 00:00:00 | 2017 | 1 | Jan | 1 | Q1 |
| 03/01/2017 00:00:00 | 2017 | 1 | Jan | 1 | Q1 |
| 04/01/2017 00:00:00 | 2017 | 1 | Jan | 1 | Q1 |
| 05/01/2017 00:00:00 | 2017 | 1 | Jan | 1 | Q1 |
| 06/01/2017 00:00:00 | 2017 | 1 | Jan | 1 | Q1 |
| 07/01/2017 00:00:00 | 2017 | 1 | Jan | 1 | Q1 |
| 08/01/2017 00:00:00 | 2017 | 1 | Jan | 1 | Q1 |
| 09/01/2017 00:00:00 | 2017 | 1 | Jan | 1 | Q1 |

Esistono anche altri approcci DAX-based per ottenere una tabella calendario in maniera più efficiente o per gestire esercizi fiscali!

## Check Point!

**La colonna calcolata è on-demand.**

- ☐ Vero
- ☐ Falso

**La misura occupa spazio.**

- ☐ Vero
- ☐ Falso

**Chi dipende dai filtri applicati? La misura o la colonna calcolata?**

# Let's take a look!

EPICODE

# GRAZIE

Epicode