|  |
| --- |
| **ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**  **TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**  **KHOA: CÔNG NGHỆ THÔNG TIN**  **A blue and white logo  Description automatically generated**  **BÁO CÁO ĐỒ ÁN SEMINAR**  **Đề tài: DAX in Power BI**  **MÔN: HỆ THỐNG THÔNG TIN**  **PHỤC VỤ TRÍ TUỆ KINH DOANH**  **NHÓM THỰC HIỆN – NHÓM 03:**  **MSSV: 20120049 – HỌ TÊN: Nguyễn Hải Đăng**  **MSSV: 20120113 – HỌ TÊN: Lê Nguyên Khang**  **MSSV: 20120269 – HỌ TÊN: Võ Văn Minh Đoàn**  **MSSV: 18120564 – HỌ TÊN: Lâm Hồng Thành**    **Giảng viên hướng dẫn: ThS. Hồ Thị Hoàng Vy**  **Giảng viên thực hành: ThS. Tiết Gia Hồng**  **Lớp lý thuyết: 20\_1**  **Học kỳ - Niên khoá: HK1 - 2023-2024** |

MỤC LỤC

[I. Danh sách thành viên – Phân công đồ án seminar 5](#_Toc151206141)

[II. Lý thuyết 5](#_Toc151206142)

[1. Tổng quan về DAX. 5](#_Toc151206143)

[2. Các công thức tính toán của DAX 5](#_Toc151206144)

[a. Measures 5](#_Toc151206145)

[b. Calculated columns 7](#_Toc151206146)

[c. Calculated tables 9](#_Toc151206147)

[d. Row-level security 11](#_Toc151206148)

[3. Các hàm tính toán thông dụng 13](#_Toc151206149)

[3.1. Aggregation Functions: 14](#_Toc151206150)

[**a.** **Average** 14](#_Toc151206151)

[**b.** **Sum** 14](#_Toc151206152)

[**c.** **Min** 15](#_Toc151206153)

[**d.** **Max** 15](#_Toc151206154)

[**e.** **Count** 16](#_Toc151206155)

[**f.** **CountRows** 16](#_Toc151206156)

[**g.** **DistinctCount** 17](#_Toc151206157)

[3.2. Date and Time Functions: 17](#_Toc151206158)

[**a.** **Now** 18](#_Toc151206159)

[**b.** **Today** 18](#_Toc151206160)

[**c.** **Date** 19](#_Toc151206161)

[**d.** **Datevalue** 19](#_Toc151206162)

[**e.** **Datediff** 20](#_Toc151206163)

[**f.** **Time** 20](#_Toc151206164)

[**g.** **Timevalue** 21](#_Toc151206165)

[**h.** **Calendar** 21](#_Toc151206166)

[3.3. Filter Functions: 22](#_Toc151206167)

[**a.** **Calculate** 22](#_Toc151206168)

[**b.** **All** 23](#_Toc151206169)

[**c.** **Filter** 23](#_Toc151206170)

[**d.** **KeepFilters** 24](#_Toc151206171)

[**e.** **RemoveFilters** 25](#_Toc151206172)

[3.4. Financial Functions: 26](#_Toc151206173)

[**a.** **NPER** 28](#_Toc151206174)

[**b.** **PV** 28](#_Toc151206175)

[**c.** **FV** 29](#_Toc151206176)

[**d.** **RATE** 30](#_Toc151206177)

[**e.** **PMT** 30](#_Toc151206178)

[3.5. Information Functions: 31](#_Toc151206179)

[**a.** **Contains** 31](#_Toc151206180)

[**b.** **ContainsString** 32](#_Toc151206181)

[**c.** **ContainsStringExact** 32](#_Toc151206182)

[**d.** **IsBlank** 33](#_Toc151206183)

[**e.** **IsEmpty** 34](#_Toc151206184)

[**f.** **IsNumber** 34](#_Toc151206185)

[3.6. Table Manipulation Functions: 35](#_Toc151206186)

[**a.** **SUMMARIZE** 35](#_Toc151206187)

[**b.** **SELECTCOLUMNS** 37](#_Toc151206188)

[**c.** **ADDCOLUMNS** 38](#_Toc151206189)

[**d.** **UNION** 39](#_Toc151206190)

[**e.** **VALUES** 39](#_Toc151206191)

[**f.** **TOPN** 40](#_Toc151206192)

[3.7. Logical functions: 41](#_Toc151206193)

[3.8. Mathematical and trigonometric functions: 43](#_Toc151206194)

[3.9. Relationship functions: 48](#_Toc151206195)

[3.10. Statistical functions: 50](#_Toc151206196)

[3.11. Text functions: 53](#_Toc151206197)

[3.12. Time intelligence functions: 54](#_Toc151206198)

[4. Variable 55](#_Toc151206199)

[4.1. Cú pháp và cấu trúc của biến: 56](#_Toc151206200)

[4.2. Lợi ích: 56](#_Toc151206201)

[III. Demo 56](#_Toc151206202)

[1. Bối cảnh 56](#_Toc151206203)

[2. Hướng giải quyết 57](#_Toc151206204)

[3. Thực hiện 58](#_Toc151206205)

[a. Tính điểm RFM: sử dụng DAX, chia dữ liệu thành 5 phần bằng cách dùng hàm Percentile tương ứng từ 1  5 điểm. 58](#_Toc151206206)

[b. Map điểm RFM vừa tính với phân khúc khách hàng tương ứng 60](#_Toc151206207)

[4. Kết quả 61](#_Toc151206208)

[IV. Tài liệu tham khảo 62](#_Toc151206209)

# Danh sách thành viên – Phân công đồ án seminar

|  |  |  |
| --- | --- | --- |
| **Họ và tên** | **Mã số sinh viên** | **Phân công** |
| Nguyễn Hải Đăng | 20120049 | Mục 1, 2 |
| Lê Nguyên Khang | 20120113 | Phần II. Lý thuyết:  Mục 3.7->3.12, mục 4. |
| Võ Văn Minh Đoàn | 20120269 | Phần II. Lý thuyết:  Mục 3.1, 3.2, 3.3, 3.4, 3.5, 3.6 |
| Lâm Hồng Thành | 18120564 | Phần III: Demo |

# Lý thuyết

## Tổng quan về DAX.

* Biểu thức phân tích dữ liệu (DAX) là ngôn ngữ biểu thức công thức được sử dụng trong Analysis Services, Power BI và Power Pivot của Excel. DAX bao gồm các hàm, toán tử và giá trị để thực hiện các phép tính và truy vấn nâng cao về dữ liệu trong các bảng và cột liên quan trong mô hình dữ liệu dạng bảng.
* DAX được sử dụng trong việc thống kê, tính toán trên cột, tính toán trên bảng và bảo mật trên dòng.

## Các công thức tính toán của DAX

### Measures

* Measures là các công thức tính toán động của DAX trong đó kết quả thay đổi tùy theo ngữ cảnh. Measures được sử dụng trong báo cáo hỗ trợ kết hợp và lọc dữ liệu bằng cách sử dụng nhiều thuộc tính như Power BI report hoặc Excel PivotTable hoặc PivotChart. Measures được tạo bằng cách sử dụng thanh công thức DAX (New Measures).
* Các công thức trong Measures có thể là hàm tổng hợp tiêu chuẩn, chẳng hạn như COUNT hoặc SUM hoặc bạn có thể xác định công thức của riêng mình bằng cách sử dụng thanh công thức DAX.
  + Để tạo Measures, sau khi đưa data model thành công vào Power BI, trong thẻ Data, nhấn vào dấu … cạnh tên model, chọn New Measure.

A screenshot of a computer

Description automatically generated

* + Sau đó định nghĩa Measures bằng cách thanh từ “Measure” dưới thành tên tùy ý.

A screenshot of a computer

Description automatically generated

* + Sau đó gõ công thức bạn muốn tính, ví dụ như hình bên dưới:

A screenshot of a computer

Description automatically generated

* + Quay lại thẻ Data, thấy “Total After Price” tức là đã tạo Measure thành công.

A screenshot of a checklist

Description automatically generated

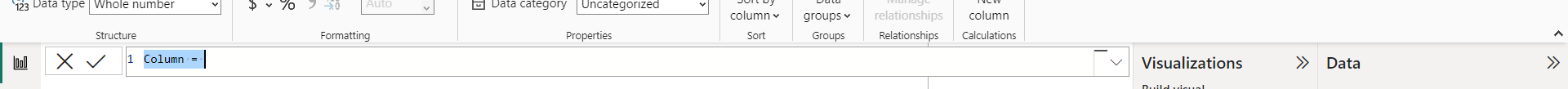
### Calculated columns

* Cột được tính toán (Calculated columns) là cột mà bạn thêm vào bảng hiện có (trong model designer) rồi tạo công thức DAX xác định các giá trị của cột. Khi một cột được tính toán chứa công thức DAX hợp lệ, các giá trị sẽ được tính cho mỗi hàng ngay khi công thức được nhập. Các giá trị sau đó được lưu trữ trong mô hình dữ liệu trong bộ nhớ.
  + Để tạo một Calculated columns, sau khi đưa data model thành công vào Power BI, trong thẻ Data, nhấn vào dấu … cạnh tên model, chọn New Column.

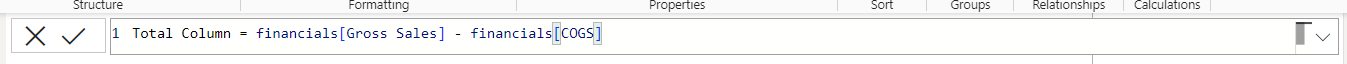
A screenshot of a computer

Description automatically generated

* + Sau khi tạo column, một công thức bắt đầu bằng “Column =” sẽ hiện lên trên thanh công thức của Power BI.



* + Thay đổi “Column” bằng tên column muốn đặt, ví dụ muốn thay đổi thành “Total Column” và công thức sẽ là financials[Gross Sales] - financials[COGS] thì ta gõ công thức như hình bên dưới.



* + Nhấn Enter để hoàn tất, nếu thành công thì trong thẻ Data sẽ có tên cột mới (trong ví dụ là “Total Column”) và khi mở Table View sẽ thấy cột mới ở bên phải cùng.

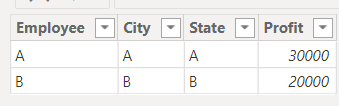
A screenshot of a checklist

Description automatically generatedA screenshot of a computer

Description automatically generated

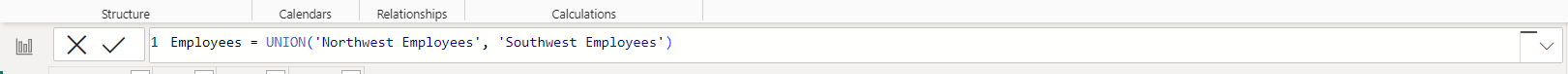
### Calculated tables

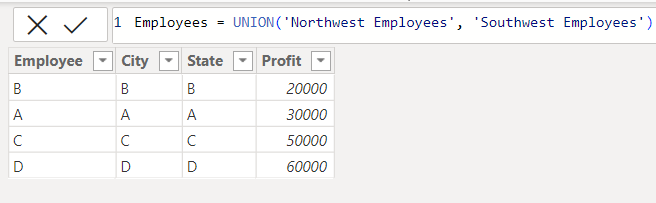
* Bảng được tính toán (Calculated tables) là một đối tượng được tính toán, dựa trên biểu thức công thức, được lấy từ tất cả hoặc một phần của các bảng khác trong cùng một mô hình. Thay vì truy vấn và tải các giá trị vào các cột của bảng mới từ nguồn dữ liệu, công thức DAX xác định các giá trị của bảng.
* Một ví dụ là bảng Ngày, dưới dạng OrderDate, ShipDate hoặc DueDate, tùy thuộc vào mối quan hệ khóa ngoại. Bằng cách tạo một bảng được tính toán cho ShipDate một cách rõ ràng, ta sẽ có được một bảng độc lập có sẵn cho các truy vấn, có thể hoạt động hoàn toàn như bất kỳ bảng nào khác. Các bảng được tính toán cũng hữu ích khi cấu hình tập hợp hàng được lọc hoặc tập hợp con hoặc tập hợp con của các cột từ các bảng hiện có khác. Điều này cho phép ta giữ nguyên bảng gốc trong khi tạo các biến thể của bảng đó để hỗ trợ các tình huống cụ thể.
  + Ví dụ, ta có 2 bảng là Northwest Employees và Southwest Employees, bây giờ ta muốn gộp 2 bảng này thành 1 bảng Employees duy nhất.

 A screenshot of a computer

Description automatically generated

* + Ta chọn Table View, sau đó trên thanh công cụ chọn Table Tools -> Chọn New Table, thay “Table” trên thanh công thức thành tên table muốn tạo, ở đây ta dùng hàm UNION để gộp dữ liệu 2 bảng lại.



* + Nhấn Enter và đây là kết quả.
* 
  + Ngoài ra, Power BI còn hỗ trợ một số hàm khác để Caculated Tables như: DISTINCT, VALUES, CROSSJOIN, NATURALINNERJOIN, NATURALLEFTOUTERJOIN, INTERSECT, CALENDAR, CALENDARAUTO

|  |  |  |
| --- | --- | --- |
|  | **Công thức** | **Nội dung tính toán** |
| *DISTINCT* | Table = DISTINCT(ColumnName Or Tableexpression) | Tạo ra một bảng chứa các giá trị duy nhất mà ColumnName của Table có thể có. |
| *VALUES* | Table = VALUES(Tablename or columnname) | Tạo ra một bảng chứa các giá trị duy nhất (kể cả dòng trống) mà ColumnName của Table có thể có. |
| CROSSJOIN | Table = CROSSJOIN(<table>, <table>[, <table>]…) | Tạo ra một bảng là tích Đề-các chứa tất các dòng của tất cả table (cách hoạt động tương tự việc FROM nhiều bảng trong SQL) |

### Row-level security

* Bảo mật trên dòng (Row-level security), DAX sẽ đánh giá theo điều kiện Boolean TRUE/FALSE, xác định những hàng nào có thể được trả về theo kết quả truy vấn của các thành viên có vai trò cụ thể (khá giống với chính sách bảo mật trên dòng của Oracle).
  + Đầu tiên (có thể bỏ qua), ta sẽ kích hoạt enhanced row-level security editor bằng cách nhấn vào Files > Options and Settings > Options > Preview features and turn on “Enhanced row-level security editor”.
  + Để sử dụng RLS, đầu tiên ta sẽ tạo role cho user: Tại Report View, trên thanh công cụ chọn Modeling -> Manage Roles.

A screenshot of a computer

Description automatically generated

* + Nhấn + New, sau đó nếu muốn thay đổi tên Role thì nhấn vào … bên cạnh Role -> Rename -> Gõ tên role.

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

* + Trong thẻ Filter Data, nhấn + Add và chọn điều kiện tương ứng.

A screenshot of a computer

Description automatically generated

* + Để kiểm tra RLS đã được kích hoạt cho role “User” chưa, trên Report View nhấn Modeling -> View as -> Tick vào User -> Nhấn OK.

A screenshot of a computer screen

Description automatically generated

* + Chọn Table View và ta thấy chỉ có các dòng có cột Segment khác “Government” hiện lên cho user có role “User” xem.

A screenshot of a computer

Description automatically generated

## Các hàm tính toán thông dụng

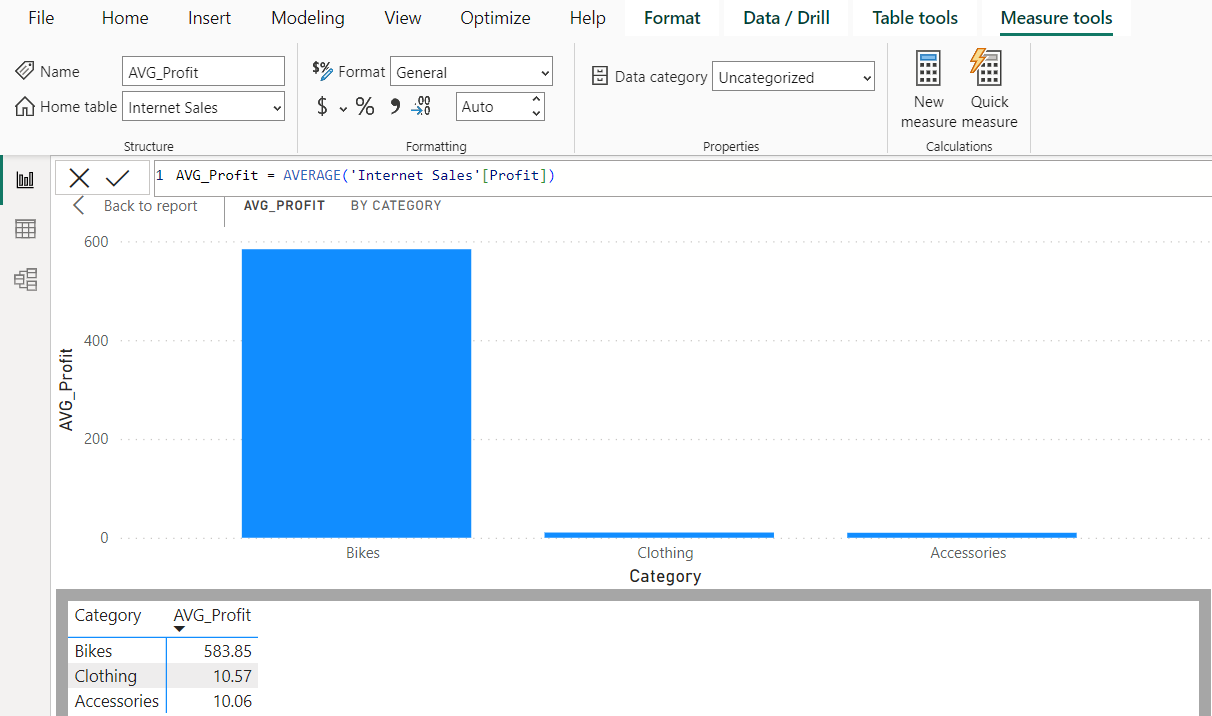
### Aggregation Functions:

Aggregation Functions là nhóm các hàm được sử dụng để thực hiện các phép tổ hợp (aggregation) trên dữ liệu. Những hàm này giúp tính toán giá trị tổ hợp từ một tập hợp các giá trị trong một cột hoặc bảng dữ liệu. Dưới đây là một số Aggregation Functions quan trọng:

#### **Average**

* Hàm này tính giá trị trung bình của các giá trị có trong cột được cung cấp làm đầu vào.
* Cú pháp: **AVERAGE(***<column>***)**
* Ví dụ: Tính trung bình lợi nhuận và thống kê theo danh mục sản phẩm

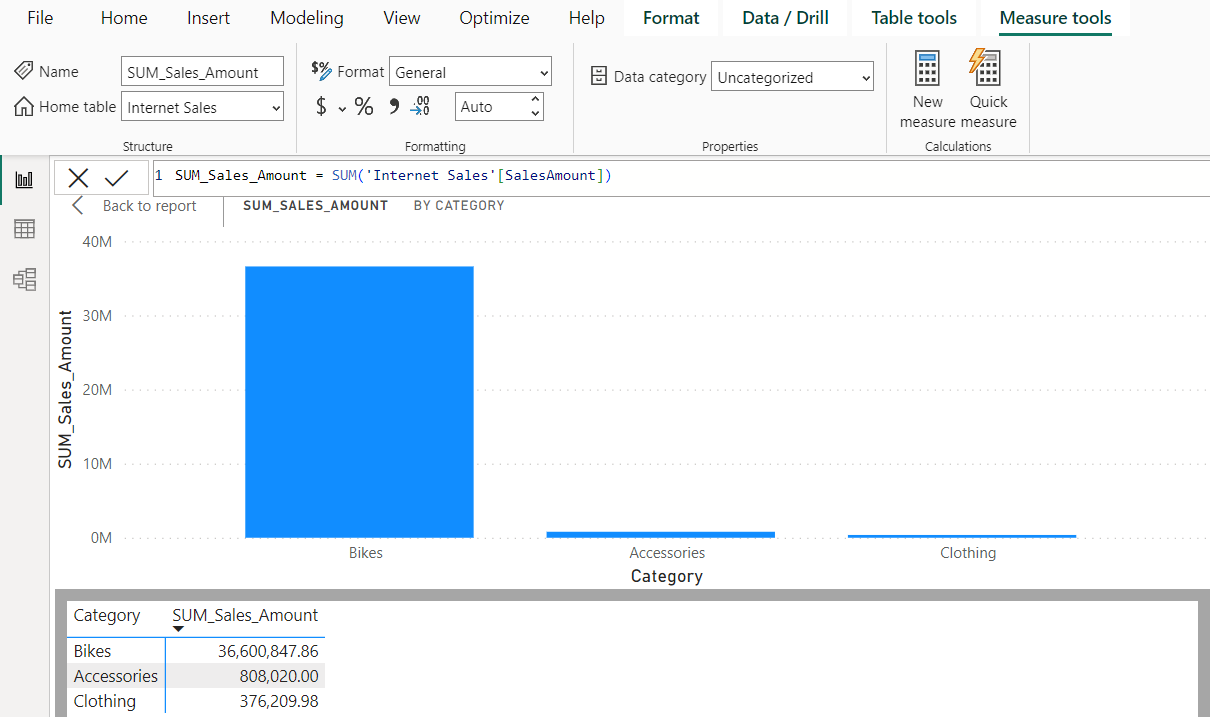
Formula: AVG\_Profit = AVERAGE('Internet Sales'[Profit])



#### **Sum**

* Hàm này tính tổng tất cả các số trong 1 cột.
* Cú pháp: **SUM(***<column>***)**
* Ví dụ: Tính tổng doanh số bán hàng và thống kê theo danh mục sản phẩm

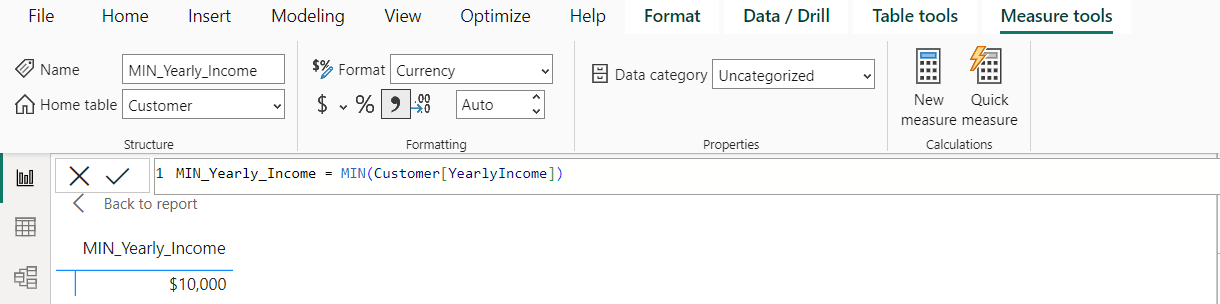
Formula: SUM\_Sales\_Amount = SUM('Internet Sales'[SalesAmount])



#### **Min**

* Hàm này trả về giá trị số nhỏ nhất trong một cột hoặc giữa hai biểu thức vô hướng.
* Cú pháp: **MIN(***<column>***)**;**MIN(***<expression1>, <expression2>***)**
* Ví dụ: Tìm mức thu nhập hằng năm thấp nhất của khách hàng

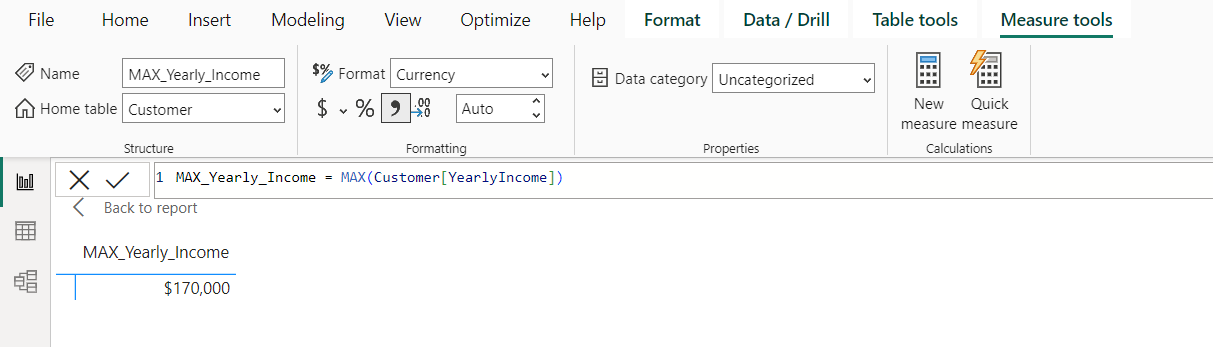
Formula: MIN\_Yearly\_Income = MIN(Customer[YearlyIncome])



#### **Max**

* Hàm này trả về giá trị số lớn nhất trong một cột hoặc giữa hai biểu thức vô hướng.
* Cú pháp: **MAX(***<column>***)**;**MAX(***<expression1>, <expression2>***)**
* Ví dụ: Tìm mức thu nhập hằng năm cao nhất của khách hàng

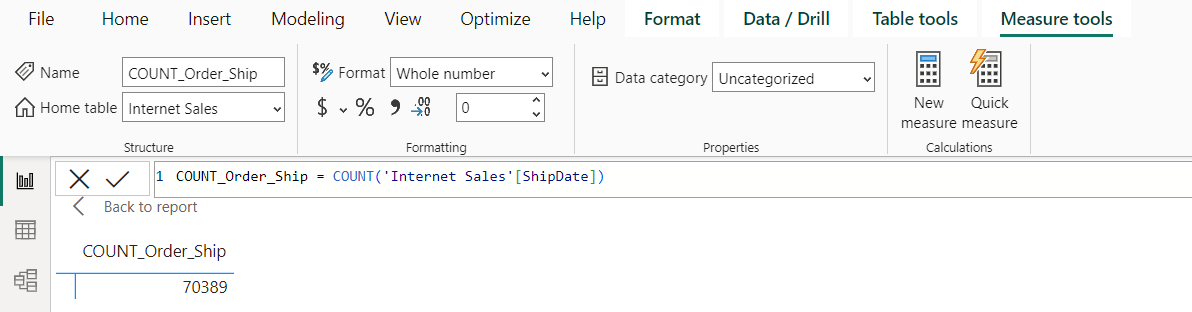
Formula: MAX\_Yearly\_Income = MAX(Customer[YearlyIncome])



#### **Count**

* Hàm này đếm số hàng trong 1 cột được chỉ định có chứa giá trị không trống. Hàm chỉ nhận 1 cột làm đối số và đếm các hàng chứa các loại giá trị: Numbers, Dates, Strings.
* Cú pháp: **COUNT(***<column>***)**
* Ví dụ: Đếm số lượng đơn hàng theo ngày giao hàng

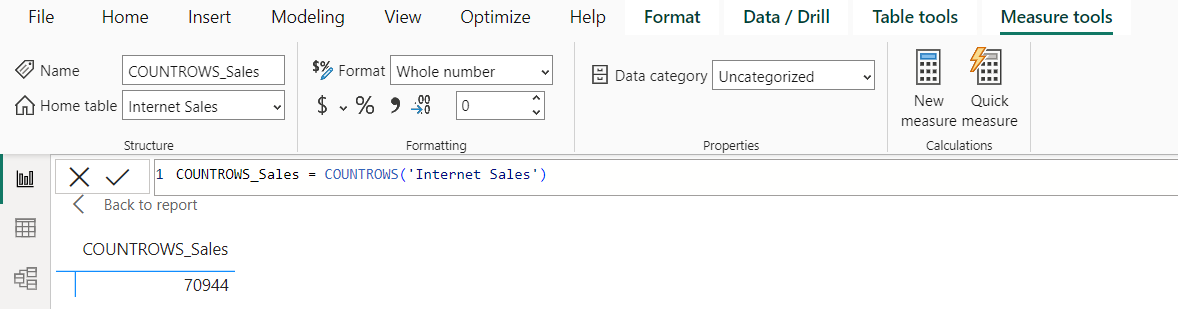
Formula: COUNT\_Order\_Ship = COUNT('Internet Sales'[ShipDate])



#### **CountRows**

* Hàm này xác định có bao nhiêu hàng trong bảng được cung cấp hoặc một bảng đã được xác định bằng biểu thức.
* Cú pháp: **COUNTROWS**(*[<table>]***)**
* Ví dụ: Đếm số lượng đơn hàng đã đặt

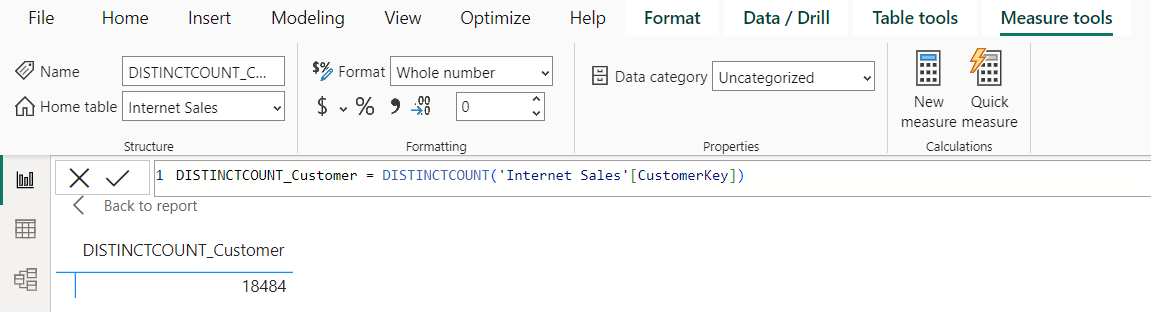
Formula: COUNTROWS\_Sales = COUNTROWS('Internet Sales')



#### **DistinctCount**

* Hàm này xác định có bao nhiêu giá trị khác nhau trong một cột. Hàm chỉ chấp nhận một cột làm đối số và có thể sử dụng các cột với bất kỳ loại dữ liệu nào trong đó. Hàm trả về BLANK nếu không có hàng nào để đếm; mặt khác, nó cho số lượng giá trị riêng biệt.
* Cú pháp: **DISTINCTCOUNT**(*<column>***)**
* Ví dụ: Đếm số lượng khách hàng khác nhau đã mua hàng

Formula: DISTINCTCOUNT\_Customer = DISTINCTCOUNT('Internet Sales'[CustomerKey])

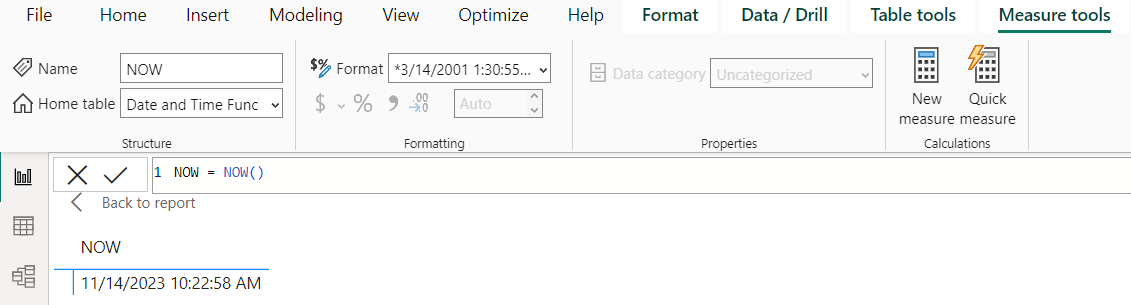


### Date and Time Functions:

Hàm Date and Time trong DAX tương tự như hàm Date and Time trong Excel. Tuy nhiên, các hàm Date and Time trong DAX dựa trên kiểu dữ liệu ngày giờ DAX. Dưới đây là một số Date and Time Functions quan trọng:

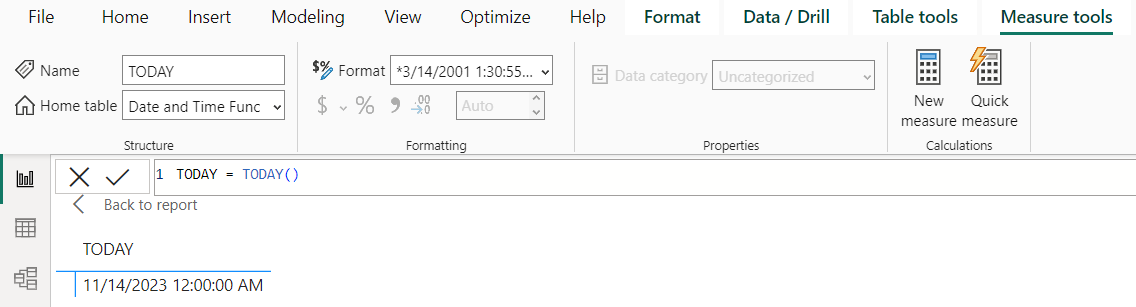
#### **Now**

* Hàm này trả về chuỗi định dạng ngày giờ chứa ngày và giờ hiện tại. Khi bạn cần hiển thị ngày và giờ hiện tại trên một trang tính hoặc tính một số dựa trên ngày và giờ hiện tại và cập nhật giá trị đó mỗi khi bạn xem worksheet, hàm NOW sẽ rất hữu ích.
* Cú pháp: **NOW**(**)**
* Ví dụ: Ngày và giờ hiện tại



#### **Today**

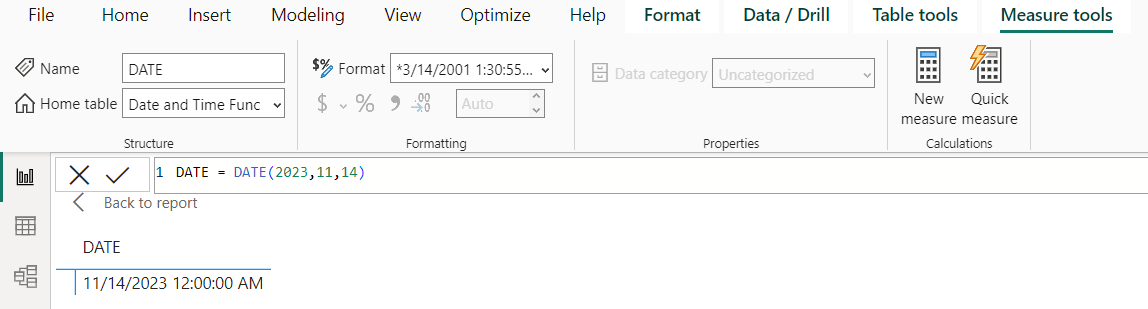
* Hàm này trả về ngày hiện tại. Hàm TODAY rất hữu ích khi bạn muốn ngày hiện tại được hiển thị trên worksheet bất kể sổ làm việc được mở khi nào. Nó cũng có thể được sử dụng để tính toán các khoảng.
* Cú pháp: **TODAY**(**)**
* Ví dụ: Ngày hiện tại



#### **Date**

* Hàm DATE trả về ngày đã được cung cấp ở định dạng DateTime. Nó tạo ngày tương ứng bằng cách sử dụng các số nguyên được cung cấp làm tham số. Khi năm, tháng và ngày được xác định bằng công thức thì hàm DATE sẽ hữu ích nhất.
* Cú pháp: **DATE**(*<year>, <month>, <day>***)**
* Ví dụ: Ngày 14/11/2023

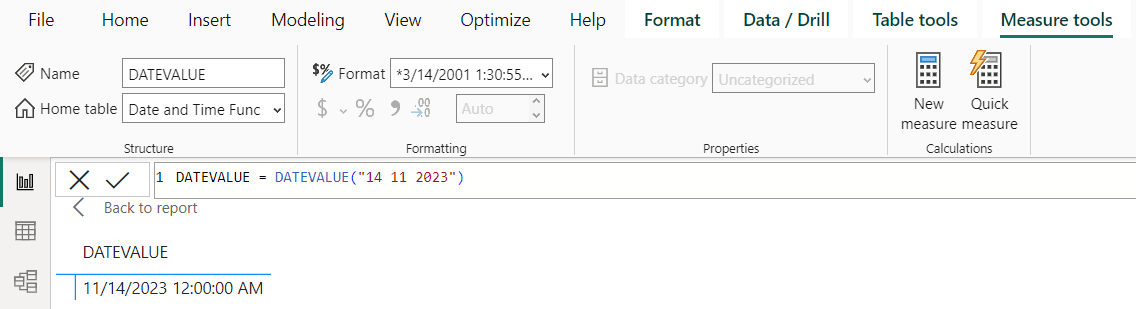
Formula: DATE = DATE(2023,11,14)



#### **Datevalue**

* Hàm DATEVALUE dịch ngày ở dạng text thành ngày ở định dạng DateTime. Cài đặt ngôn ngữ và ngày/giờ của client được hàm DATEVALUE sử dụng để diễn giải giá trị văn bản trong khi thực hiện chuyển đổi.
* Cú pháp: **DATEVALUE**(*date\_text***)**
* Ví dụ: Ngày 14/11/2023

Formula: DATEVALUE = DATEVALUE("14 11 2023")



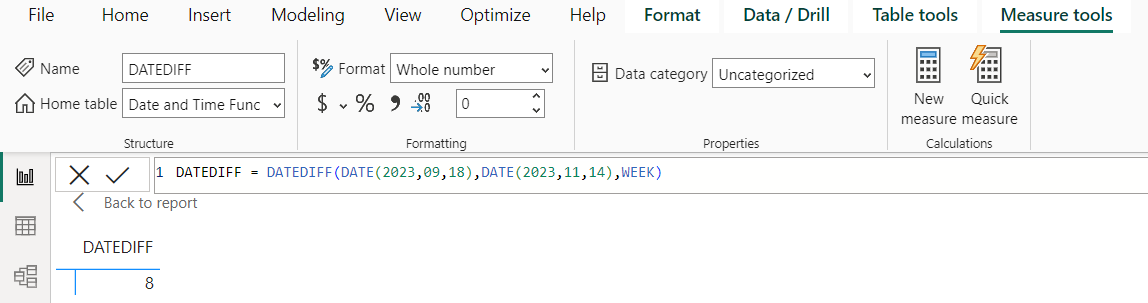
#### **Datediff**

* Hàm DATEDIFF trả về số ranh giới khoảng thời gian giữa hai ngày.
* Cú pháp: **DATEDIFF**(*<Date1>, <Date2>, <Interval>***)**

Trong đó, Interval: Second, Minute, Hour, Day, Week, Month, Quarter, Year

* Ví dụ: Tính số tuần trong khoảng từ ngày 18/09/2023 đến ngày 14/11/2023

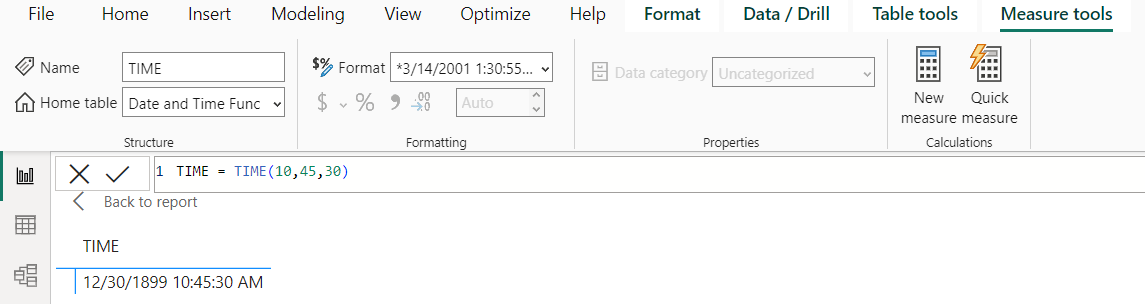
Formula: DATEDIFF = DATEDIFF(DATE(2023,09,18), DATE(2023,11,14), WEEK)



#### **Time**

* Hàm này giúp chuyển đổi giờ, phút và giây dưới dạng số thành thời gian ở định dạng datetime.
* Cú pháp: **TIME**(*<hour>, <minute>, <second>***)**
* Ví dụ: Thời gian 10:45:30

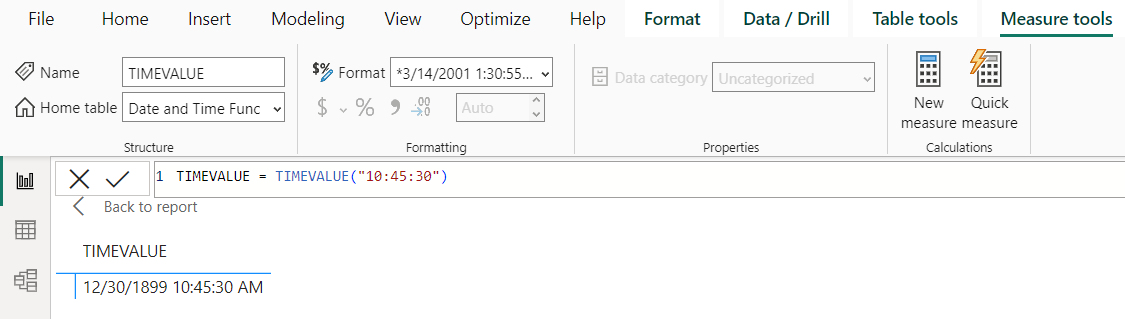
Formula: TIME = TIME(10,45,30)



#### **Timevalue**

* Hàm này giúp chuyển đổi thời gian dưới dạng văn bản thành thời gian ở định dạng datetime.
* Cú pháp: **TIMEVALUE**(*<time\_text>***)**
* Ví dụ: Thời gian 10:45:30

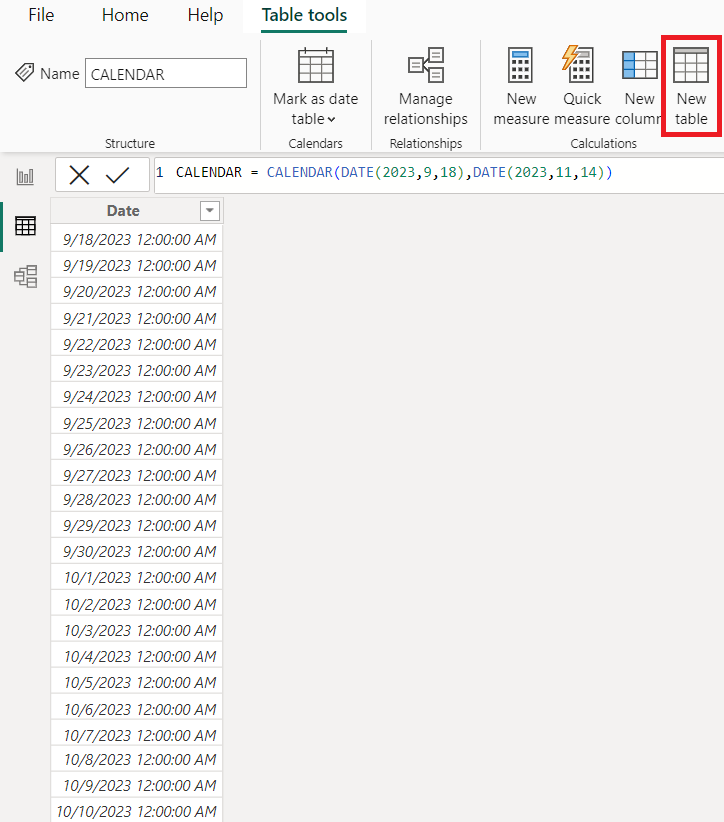
Formula: TIMEVALUE = TIMEVALUE("10:45:30")



#### **Calendar**

* Hàm này tạo ra một bảng có một cột "Date" duy nhất được tạo thành từ một phạm vi ngày liên tục. Phạm vi ngày bao gồm hai ngày đó và kéo dài từ ngày bắt đầu được chỉ định đến ngày kết thúc được chỉ định.
* Cú pháp: **CALENDAR**(*<start\_date>, <end\_date>***)**
* Ví dụ: Tạo 1 table mới chứa calendar từ 18/09/2023 đến 14/11/2023

Formula: CALENDAR = CALENDAR(DATE(2023,9,18), DATE(2023,11,14))



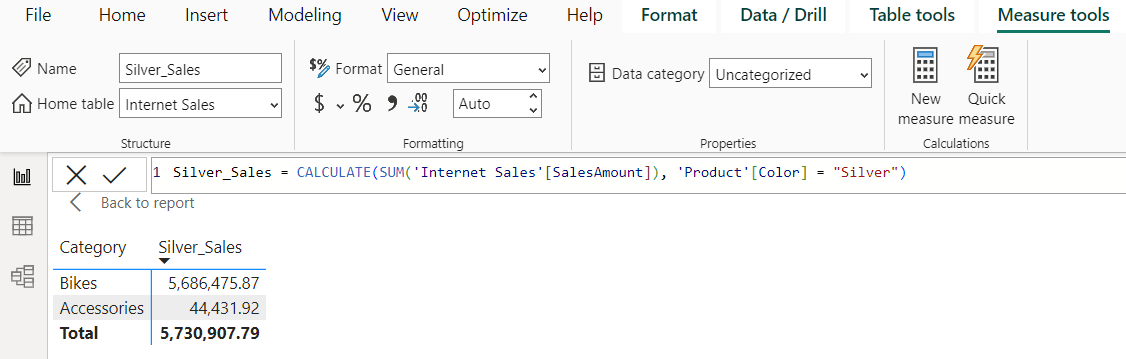
### Filter Functions:

Các hàm lọc trong DAX nằm trong số những hàm phức tạp và mạnh mẽ nhất, đồng thời chúng rất khác với các thao tác trên Excel. Các hàm tìm kiếm hoạt động tương tự như cơ sở dữ liệu bằng cách sử dụng các bảng và mối quan hệ. Bạn có thể tạo các phép tính động bằng cách thao tác bối cảnh dữ liệu bằng các hàm lọc. Dưới đây là một số Filter Functions quan trọng:

#### **Calculate**

* Hàm Calculate chủ yếu được sử dụng để tạo các bộ lọc mới hoặc thay đổi bộ lọc hiện tại trong các biểu thức DAX. Hàm Calculate được sử dụng để thực hiện các phép toán tính toán dựa trên các bộ lọc được áp dụng.
* Cú pháp: **CALCULATE**(*<expression>, [<filter1>[, <filter2>[,…]]]***)**
* Ví dụ: Thống kê tổng doanh thu bán hàng của các sản phẩm có màu bạc theo danh mục sản phẩm

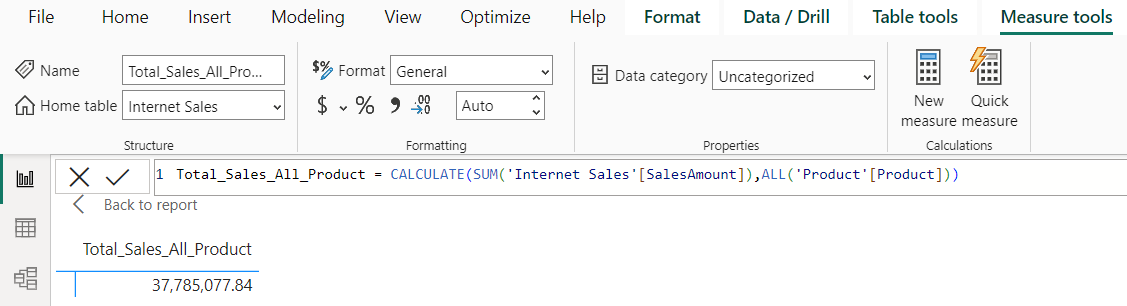
Formula: Silver\_Sales = CALCULATE(SUM('Internet Sales'[SalesAmount]), 'Product'[Color] = "Silver")



#### **All**

* Hàm All bỏ qua mọi bộ lọc có thể đã được sử dụng và trả về tất cả các hàng trong bảng hoặc tất cả các giá trị trong một cột. Chức năng này có thể được sử dụng để loại bỏ bộ lọc và thực hiện các phép tính trên tất cả các hàng của bảng.
* Cú pháp: **ALL**(*[<table> | <column>[, <column>[, <column>[,…]]]]***)**
* Ví dụ: Tính tổng doanh thu trên tất cả sản phẩm (không áp dụng bộ lọc nào)

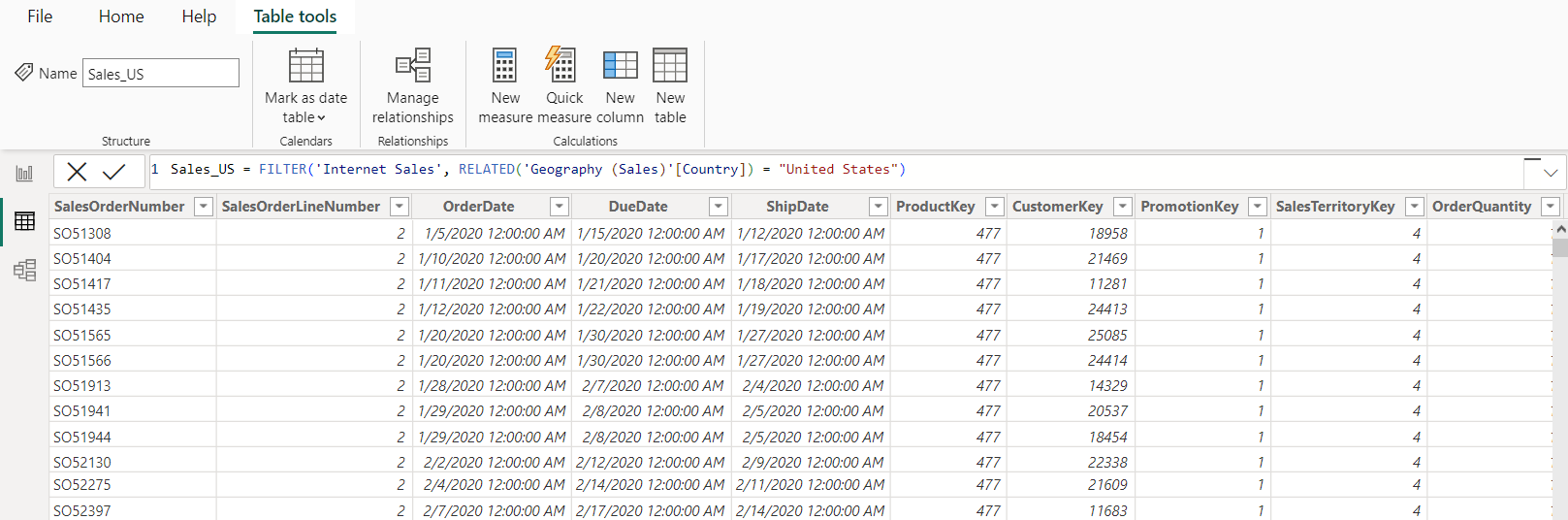
Formula: Total\_Sales\_All\_Product = CALCULATE(SUM('Internet Sales'[SalesAmount]), ALL('Product'[Product]))



#### **Filter**

* Hàm Filter lọc dữ liệu từ một bảng dựa trên một điều kiện được chỉ định. Hàm này sẽ duyệt qua từng hàng của bảng để kiểm tra điều kiện.
* Cú pháp: **FILTER**(*<table>, <filter>***)**
* Ví dụ: Tạo bảng Sale\_US mới dựa trên bộ lọc các đơn hàng tại Mỹ của table Internet Sales (dùng hàm RELATED để liên kết SalesTerritoryKey trong bảng Internet Sales với Country trong bảng Geography (Sales))

Formula: Sales\_US = FILTER('Internet Sales', RELATED('Geography (Sales)'[Country]) = "United States")



#### **KeepFilters**

* Hàm KeepFilters giữ nguyên bộ lọc hiện tại khi áp dụng một biểu thức đối với một cột.
* Cú pháp: **KEEPFILTERS(***<expression>***)**
* Ví dụ: Tính tổng doanh thu bán hàng tại Mỹ (dùng 2 bộ lọc lồng nhau và hàm KEEPFILTERS giúp giữ nguyên bộ lọc cũ khi áp dụng bộ lọc mới)

Formula: Total\_Sales\_US =

CALCULATE(

CALCULATE('Internet Sales'[SUM\_Sales\_Amount]

, KEEPFILTERS('Geography (Sales)'[Country]="United States"

|| 'Geography (Sales)'[Country]="Canada"

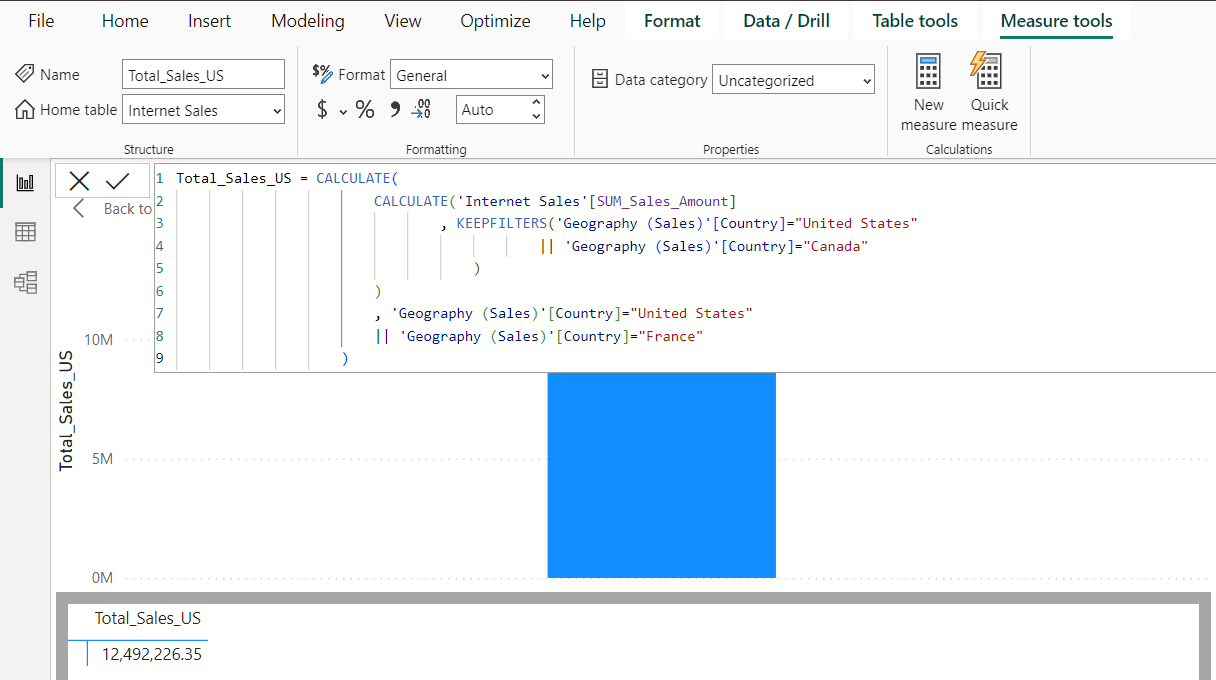
)

)

, 'Geography (Sales)'[Country]="United States"

|| 'Geography (Sales)'[Country]="France"

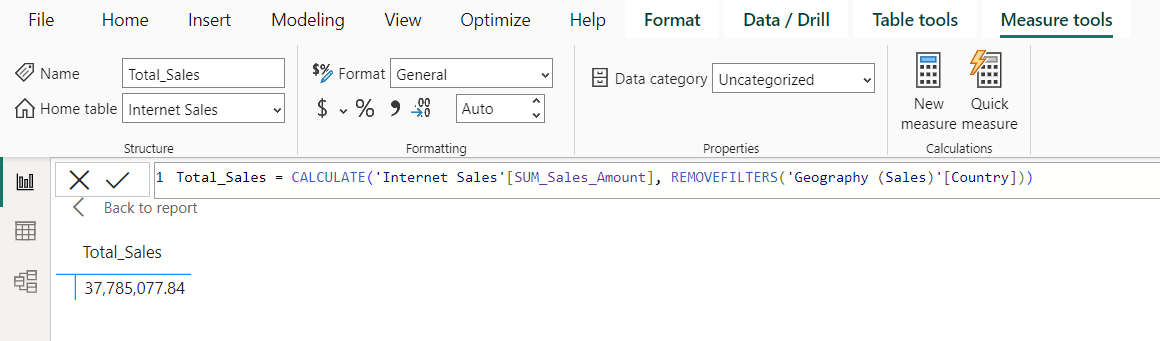
)



#### **RemoveFilters**

* Hàm RemoveFilters dùng để xóa bộ lọc khỏi các bảng hoặc cột được chỉ định.
* Cú pháp: **REMOVEFILTERS(***[<table> | <column>[, <column>[, <column>[,…]]]]***)**
* Ví dụ: Tính tổng doanh thu bán hàng (loại bỏ tất cả bộ lọc về khu vực bán hàng)

Formula: Total\_Sales = CALCULATE('Internet Sales'[SUM\_Sales\_Amount], REMOVEFILTERS('Geography (Sales)'[Country]))



### Financial Functions:

* Các hàm tài chính trong DAX được sử dụng trong các công thức thực hiện các phép tính tài chính, chẳng hạn như giá trị hiện tại ròng và tỷ suất lợi nhuận. Các hàm này tương tự như các hàm tài chính được sử dụng trong Microsoft Excel.
* Bài toán: Lập kế hoạch cho việc vay mua nhà

Giả sử bạn đang lập kế hoạch để mua một căn nhà và bạn muốn tính toán các thông số tài chính liên quan đến khoản vay mua nhà.

* Số tiền vay (PV): Bạn muốn biết số tiền bạn có thể vay dựa trên thu nhập và khả năng trả nợ hàng tháng.
* Tổng số kỳ trả nợ (NPER): Bạn muốn biết cần trả nợ trong bao nhiêu kỳ để hoàn trả hết khoản vay.
* Lãi suất hàng tháng (RATE): Bạn muốn biết lãi suất hàng tháng cần áp dụng cho khoản vay.
* Trả nợ hàng tháng (PMT): Bạn muốn biết số tiền cần trả mỗi tháng để hoàn trả khoản vay.
* Tổng số tiền trả lại (FV): Bạn muốn biết tổng số tiền bạn sẽ trả sau khi hoàn thành việc trả nợ theo kế hoạch.
* Bảng mô tả tham số của các hàm:

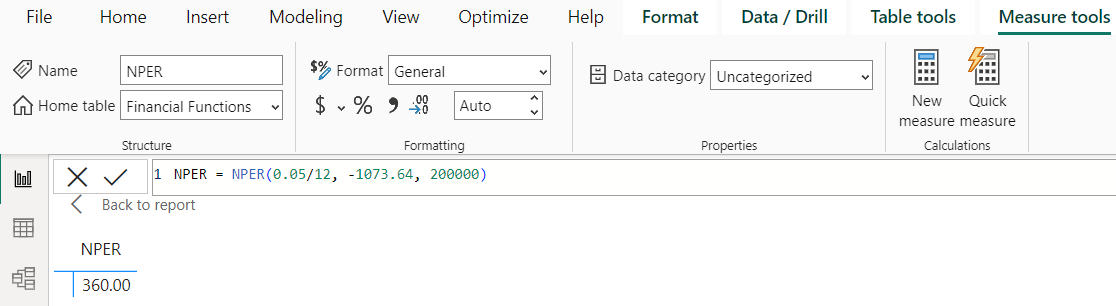
|  |  |
| --- | --- |
| **Tham số** | **Ý nghĩa** |
| rate | Lãi suất từng kỳ |
| nper | Tổng số kỳ thanh toán trong một năm. Ví dụ: nếu bạn vay tiền mua ô tô thời hạn 4 năm và thanh toán hàng tháng, khoản vay của bạn có kỳ hạn 4\*12 (hoặc 48). Bạn sẽ nhập 48 vào công thức cho nper |
| pmt | Việc thanh toán được thực hiện từng kỳ; nó không thể thay đổi trong suốt thời gian của khoản vay. Thông thường, pmt chứa tiền gốc và lãi nhưng không có phí hoặc thuế khác |
| pv | Giá trị hiện tại hoặc tổng số tiền mà một loạt khoản thanh toán trong tương lai có giá trị ở thời điểm hiện tại, pv mang giá trị âm khi được xem là khoản nợ và mang giá trị dương khi được xem là nguồn thu nhập |
| fv | (Optional) Giá trị tương lai hoặc số dư tiền mặt bạn muốn đạt được sau lần thanh toán cuối cùng được thực hiện. Nếu fv bị bỏ qua, nó được coi là giá trị trống |
| type | (Optional) Số 0 hoặc 1 và cho biết khi nào đến hạn thanh toán. Nếu loại bị bỏ qua, nó được coi là 0. Nếu là 0 hoặc bị bỏ qua: đến hạn vào cuối kỳ; nếu là 1: đến hạn vào đầu kỳ |
| guess | (Optional) Dự đoán về tỷ lệ:  - Nếu bỏ qua thì giả định là 10%  - Nếu RATE không hội tụ, thử các giá trị khác nhau để đoán. RATE thường hội tụ nếu dự đoán nằm trong khoảng từ 0 đến 1 |

* Dưới đây là một số Financial Functions quan trọng hỗ trợ cho bài toán:

#### **NPER**

* Hàm NPER (Number of Periods) tính toán số lượng chu kỳ thanh toán trong một loại khoản vay dựa trên lãi suất cố định và thanh toán hàng kỳ.
* Cú pháp: **NPER(***<rate>, <pmt>, <pv>[, <fv>[, <type>]]***)**
* Ví dụ:
* Lãi suất hàng tháng (RATE): 0.0041667 (lãi suất hàng năm là 0.05)
* Số tiền trả hàng tháng (PMT): $1,073.64
* Số tiền vay ban đầu (PV): $200,000
* Type: mặc định (thanh toán vào cuối kỳ)

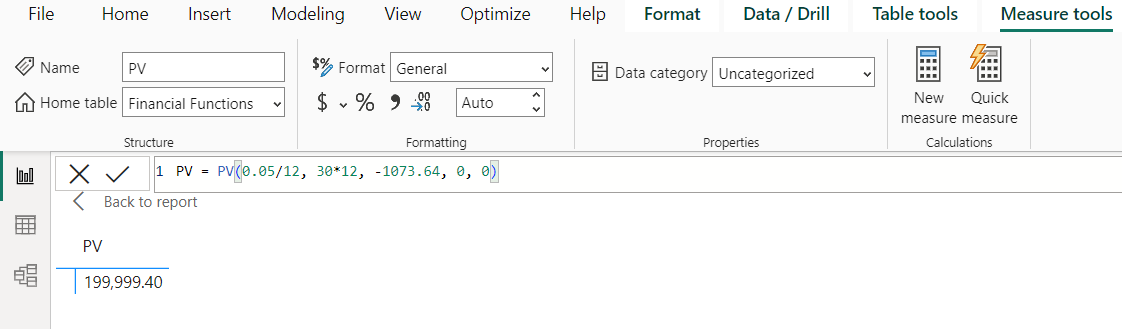
Formula: NPER = NPER(0.05/12, -1073.64, 200000)



#### **PV**

* Hàm PV (Present Value) tính giá trị hiện tại của khoản vay hoặc khoản đầu tư, dựa trên lãi suất không đổi. Bạn có thể sử dụng PV với các khoản thanh toán định kỳ, cố định (chẳng hạn như khoản thế chấp hoặc khoản vay khác) và/hoặc giá trị tương lai là mục tiêu đầu tư của bạn.
* Cú pháp: **PV(***<rate>, <nper>, <pmt>[, <fv>[, <type>]]***)**
* Ví dụ:
* Lãi suất hàng tháng (RATE): 0.0041667 (lãi suất hàng năm là 0.05)
* Số kỳ trả nợ (NPER): 360 tháng (30 năm)
* Số tiền trả hàng tháng (PMT): $1,073.64
* Type: mặc định (thanh toán vào cuối kỳ)

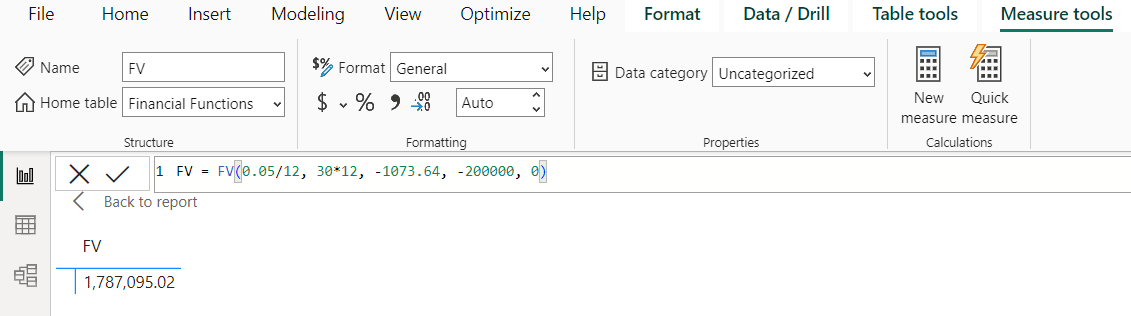
Formula: PV = PV(0.05/12, 30\*12, -1073.64, 0, 0)



#### **FV**

* Hàm FV (Future Value) tính toán giá trị tương lai của khoản đầu tư dựa trên lãi suất không đổi. Bạn có thể sử dụng FV với các khoản thanh toán định kỳ, cố định và/hoặc thanh toán một lần.
* Cú pháp: **FV(***<rate>, <nper>, <pmt>[, <pv>[, <type>]]***)**
* Ví dụ:
* Lãi suất hàng tháng (RATE): 0.0041667 (lãi suất hàng năm là 0.05)
* Số kỳ trả nợ (NPER): 360 tháng (30 năm)
* Số tiền trả hàng tháng (PMT): $1,073.64
* Số tiền vay ban đầu (PV): $200,000
* Type: mặc định (thanh toán vào cuối kỳ)

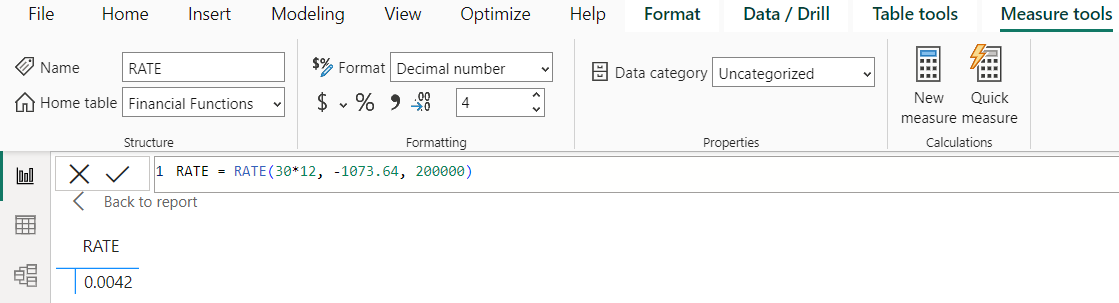
Formula: FV = FV(0.05/12, 30\*12, -1073.64, -200000, 0)



#### **RATE**

* Hàm RATE trả về lãi suất mỗi kỳ của một khoản vay. RATE được tính bằng phép lặp và có thể có 0 hoặc nhiều giải pháp. Nếu các kết quả liên tiếp của RATE không hội tụ về khoảng 0,0000001 sau 20 lần lặp thì lỗi sẽ được trả về.
* Cú pháp: **RATE(***<nper>, <pmt>, <pv>[, <fv>[, <type>[, <guess>]]]***)**
* Ví dụ:
* Số kỳ trả nợ (NPER): 360 tháng (30 năm)
* Số tiền trả hàng tháng (PMT): $1,073.64
* Số tiền vay ban đầu (PV): $200,000
* Type: mặc định (thanh toán vào cuối kỳ)
* Guess: giả định 10%

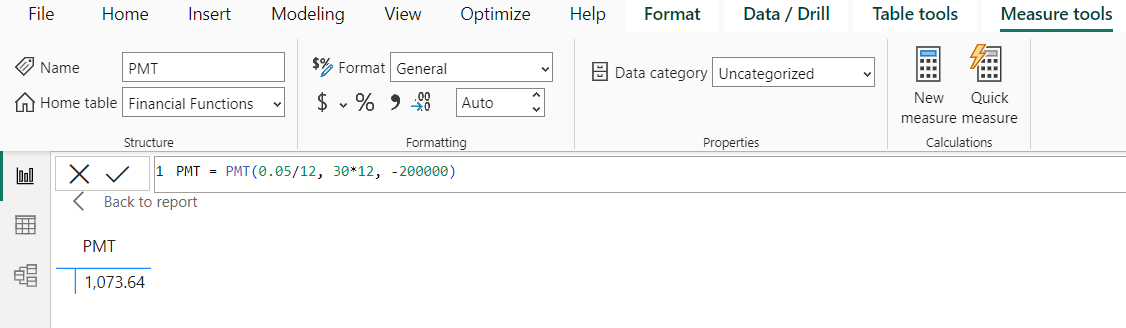
Formula: RATE = RATE(30\*12, -1073.64, 200000)



#### **PMT**

* Hàm này tính toán khoản thanh toán cho khoản vay dựa trên các khoản thanh toán cố định và lãi suất không đổi.
* Cú pháp: **PMT(***<rate>, <nper>, <pv>[, <fv>[, <type>]]***)**
* Ví dụ:
* Lãi suất hàng tháng (RATE): 0.0041667 (lãi suất hàng năm là 0.05)
* Số kỳ trả nợ (NPER): 360 tháng (30 năm)
* Số tiền vay ban đầu (PV): $200,000
* Type: mặc định (thanh toán vào cuối kỳ)

Formula: PMT = PMT(0.05/12, 30\*12, -200000)



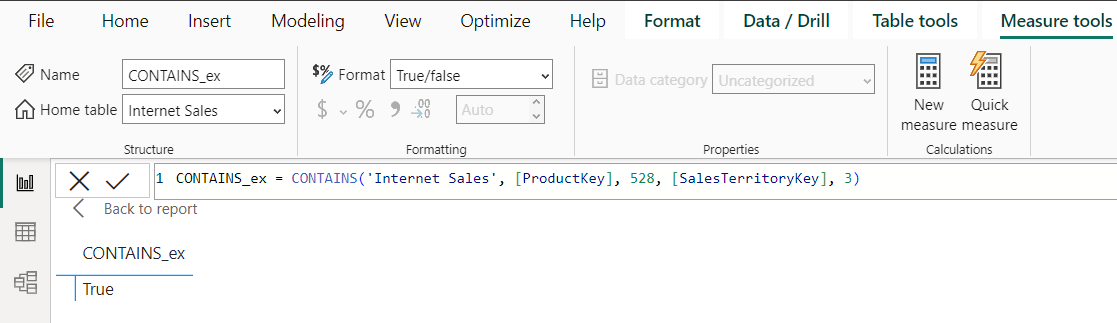
### Information Functions:

Khi được cung cấp một ô hoặc hàng làm đối số, các hàm thông tin DAX sẽ kiểm tra ô hoặc hàng đó và thông báo cho bạn về tính tương thích của giá trị với loại dự kiến. Ví dụ: nếu giá trị bạn tham chiếu có lỗi thì hàm ISERROR sẽ trả về TRUE. Dưới đây là một số Information Functions quan trọng:

#### **Contains**

* Hàm này trả về true nếu các giá trị cho tất cả các cột được tham chiếu tồn tại hoặc được chứa trong các cột đó; nếu không thì hàm trả về false.
* Cú pháp: **CONTAINS(***<table>, <columnName>, <value>[, <columnName>, <value>]…***)**
* Ví dụ: Tạo measure cho biết có đơn hàng của sản phẩm 528 và khu vực 3 không

Formula: CONTAINS\_ex = CONTAINS('Internet Sales', [ProductKey], 528, [SalesTerritoryKey], 3)



#### **ContainsString**

* Tùy thuộc vào việc một chuỗi có chứa chuỗi khác hay không, hàm này trả về true hoặc false. Hàm này khác ContainsStringExact là không phân biệt hoa thường hay thứ tự các ký tự trong chuỗi.
* Cú pháp: **CONTAINSSTRING(***<within\_text>, <find\_text>***)**
* Ví dụ: Tạo table với một số test cases

Formula: CONTAINSSTRING\_ex = ROW(

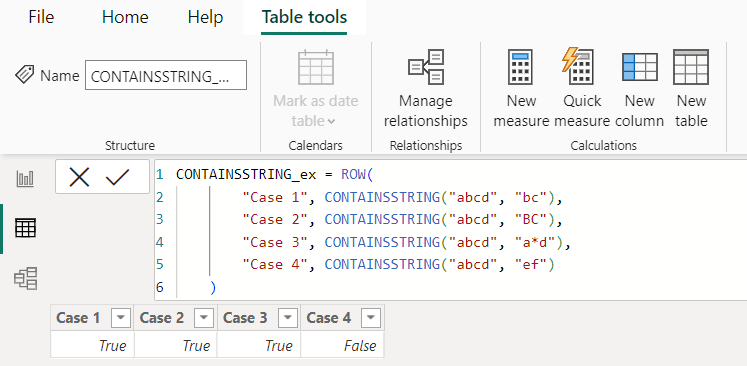
"Case 1", CONTAINSSTRING("abcd", "bc"),

"Case 2", CONTAINSSTRING("abcd", "BC"),

"Case 3", CONTAINSSTRING("abcd", "a\*d"),

"Case 4", CONTAINSSTRING("abcd", "ef")

)



#### **ContainsStringExact**

* Hàm này giúp xác định một chuỗi có chứa chuỗi khác hay không, có phân biệt hoa thường và thứ tự các ký tự.
* Cú pháp: **CONTAINSSTRINGEXACT(***<within\_text>, <find\_text>***)**
* Ví dụ: Tạo table với một số test cases

Formula: CONTAINSSTRINGEXACT\_ex = ROW(

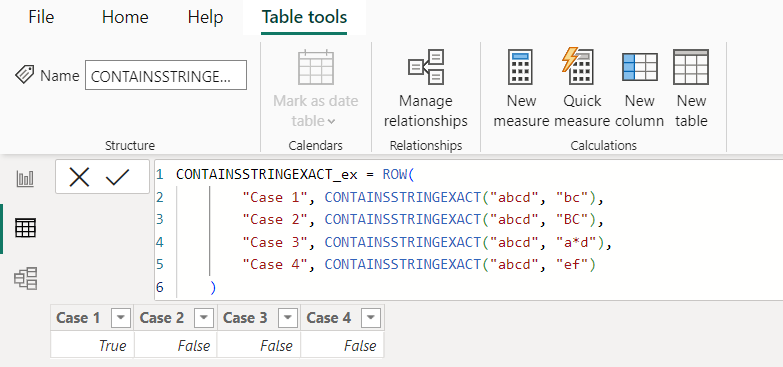
"Case 1", CONTAINSSTRINGEXACT("abcd", "bc"),

"Case 2", CONTAINSSTRINGEXACT("abcd", "BC"),

"Case 3", CONTAINSSTRINGEXACT("abcd", "a\*d"),

"Case 4", CONTAINSSTRINGEXACT("abcd", "ef")

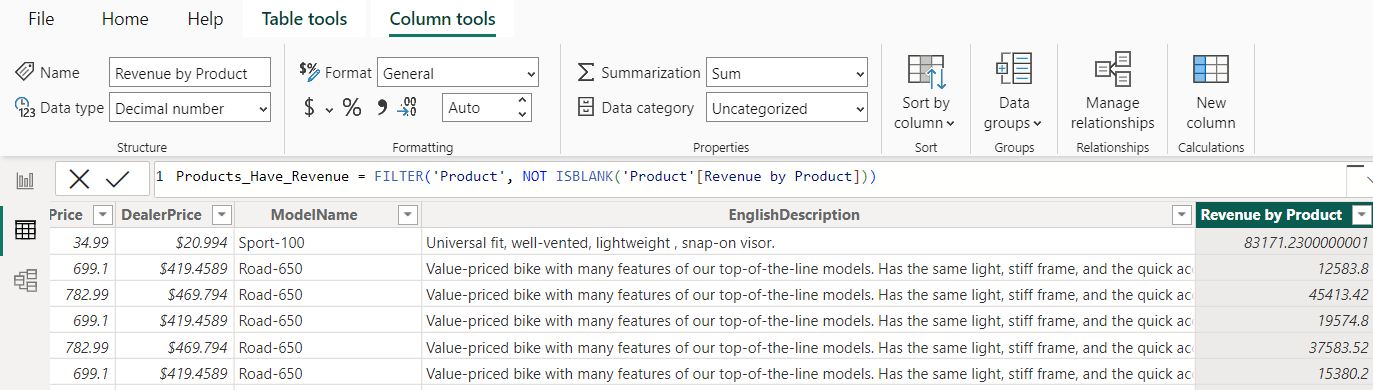
)



#### **IsBlank**

* Hàm này kiểm tra xem một giá trị có trống hay không và trả về true hoặc false.
* Cú pháp: **ISBLANK(***<value>***)**
* Ví dụ: Tạo một bảng Product mới bằng các lọc ra các sản phẩm tạo ra doanh thu (cột Revenue by Product không trống)

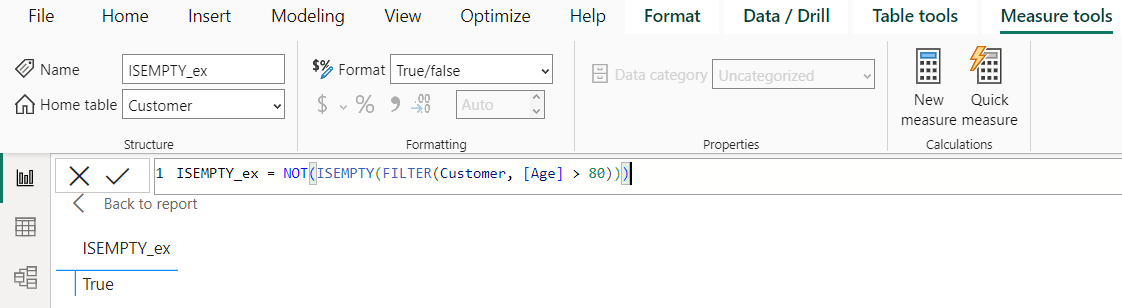
Formula: Products\_Have\_Revenue = FILTER('Product', NOT ISBLANK('Product'[Revenue by Product]))



#### **IsEmpty**

* Hàm này xác minh xem một bảng có trống không.
* Cú pháp: **ISEMPTY(***<table\_expression>***)**
* Ví dụ: Kiểm tra xem có khách hàng nào lớn hơn 80 tuổi hay không (Nếu có thì kết quả lọc khách hàng lớn hơn 80 tuổi không phải bảng trống)

Formula: ISEMPTY\_ex = NOT(ISEMPTY(FILTER(Customer, [Age] > 80)))



#### **IsNumber**

* Hàm này xác định xem một giá trị có phải là số hay không trước khi trả về giá trị true hoặc false.
* Cú pháp: **ISNUMBER(***<value>***)**
* Ví dụ: Tạo table với một số test cases

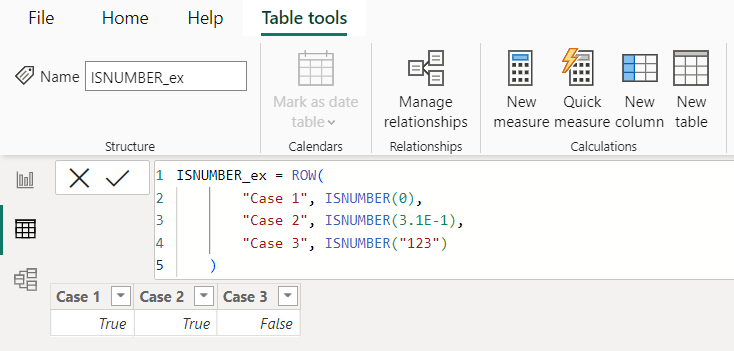
Formula: ISNUMBER\_ex = ROW(

"Case 1", ISNUMBER(0),

"Case 2", ISNUMBER(3.1E-1),

"Case 3", ISNUMBER("123")

)



### Table Manipulation Functions:

Các hàm quản lý bảng (table manipulation functions) được sử dụng để thực hiện các phép biến đổi và quản lý dữ liệu trong các bảng. Các hàm này trả về 1 bảng hoặc thao tác với bảng hiện có. Dưới đây là một số Table Manipulation Functions quan trọng:

#### **SUMMARIZE**

* Hàm SUMMARIZE được sử dụng để tạo một bảng mới bằng cách tóm tắt dữ liệu từ một hoặc nhiều bảng theo các cột được chỉ định.
* Cú pháp: **SUMMARIZE(***<table>, <groupBy\_columnName>[, <groupBy\_columnName>]…[, <name>, <expression>]…***)**

Các tham số được mô tả như sau:

|  |  |
| --- | --- |
| **Tham số** | **Ý nghĩa** |
| table | Bất kỳ biểu thức DAX nào trả về một bảng dữ liệu. |
| groupBy\_ColumnName | (Optional) Tên các cột dùng để gom nhóm |
| name | Tên của các cột mới được tạo ra |
| expression | Biểu thức trả về giá trị cho các cột mới |

* Ví dụ: Tạo bản tóm tắt về số lượng đơn hàng, tổng doanh thu, tổng lợi nhuận theo danh mục sản phẩm và theo khu vực

Formula: SUMMARIZE\_ex = SUMMARIZE('Internet Sales',

'Geography (Sales)'[Country],

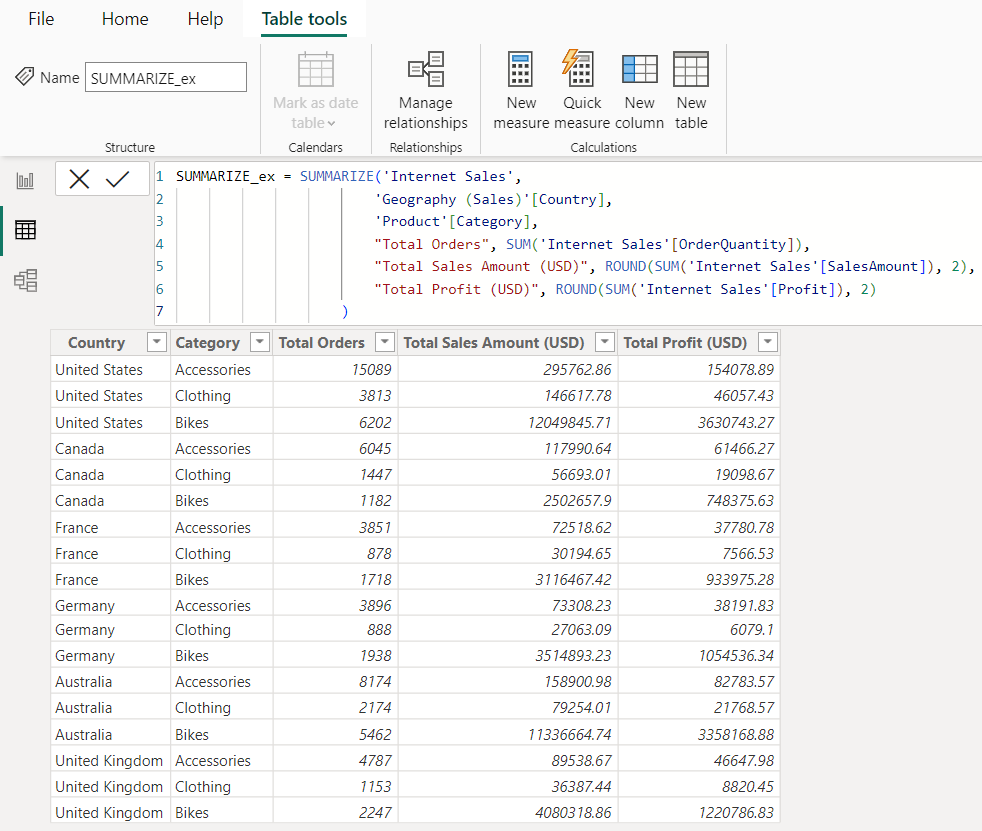
'Product'[Category],

"Total Orders", SUM('Internet Sales'[OrderQuantity]),

"Total Sales Amount (USD)", ROUND(SUM('Internet Sales'[SalesAmount]), 2),

"Total Profit (USD)", ROUND(SUM('Internet Sales'[Profit]), 2)

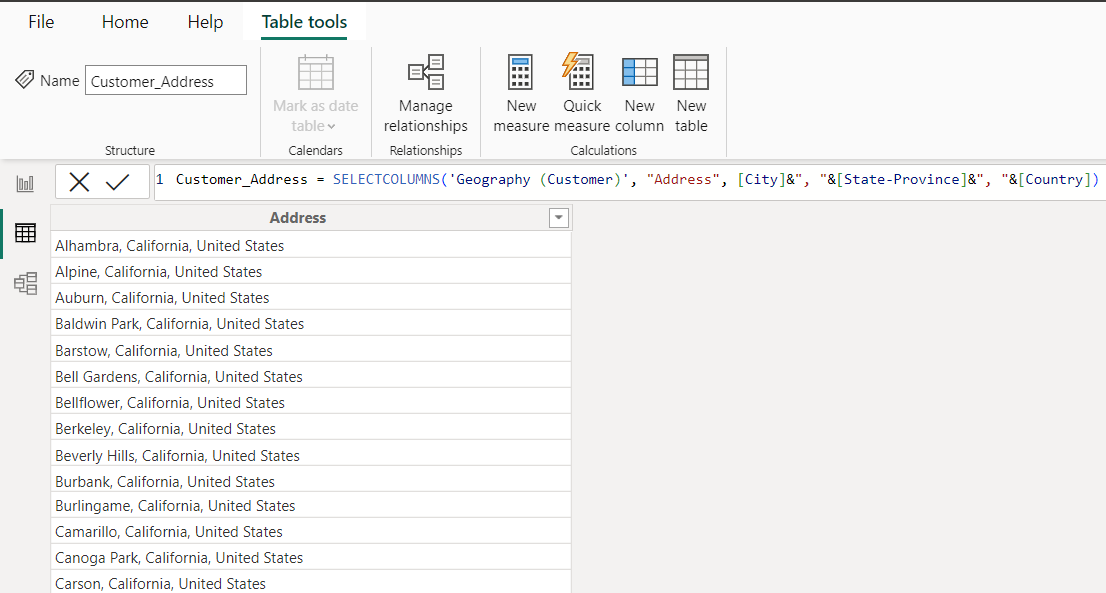
)



#### **SELECTCOLUMNS**

* Hàm SELECTCOLUMNS tạo một bảng mới bằng cách chọn một hoặc nhiều cột từ một bảng hiện tại và thêm các cột tính toán nếu cần.
* Cú pháp: **SELECTCOLUMNS(***<Table>, [<Name>], <Expression>, [<Name>], …***)**
* Ví dụ: Dùng hàm SELECTCOLUMNS để tạo 1 bảng mới chứa địa chỉ đầy đủ của khách hàng được nối từ các trường City, State-Province, Country trong bảng Geography (Customer)

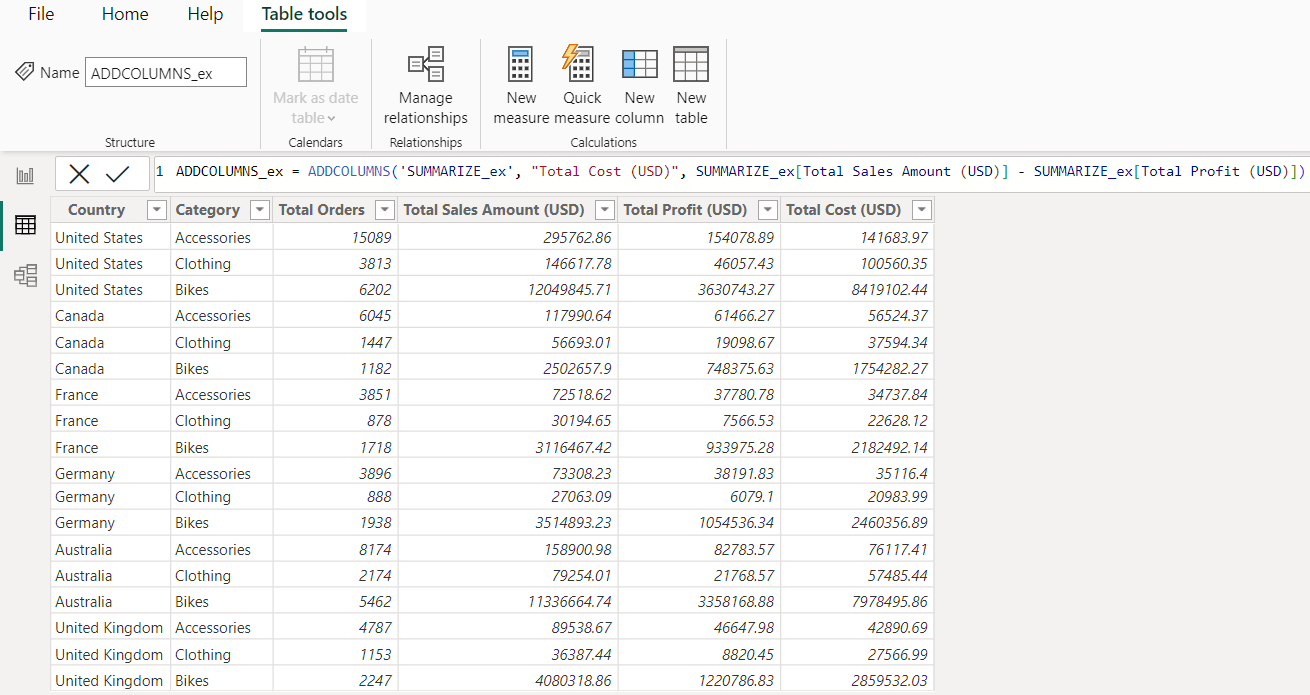
Formula: Customer\_Address = SELECTCOLUMNS('Geography (Customer)', "Address", [City]&", "&[State-Province]&", "&[Country])



#### **ADDCOLUMNS**

* Hàm ADDCOLUMNS thêm một hoặc nhiều cột mới vào bảng hiện tại dựa trên các biểu thức được chỉ định.
* Cú pháp: **ADDCOLUMNS(***<table>, <name>, <expression>[, <name>, <expression>]…***)**
* Ví dụ: Tạo 1 bảng mới bằng cách thêm 1 cột Total Cost (USD) chứa tổng chi phí (lấy Total Sales Amount – Total Profit) vào bảng SUMMARIZE\_ex ở trên

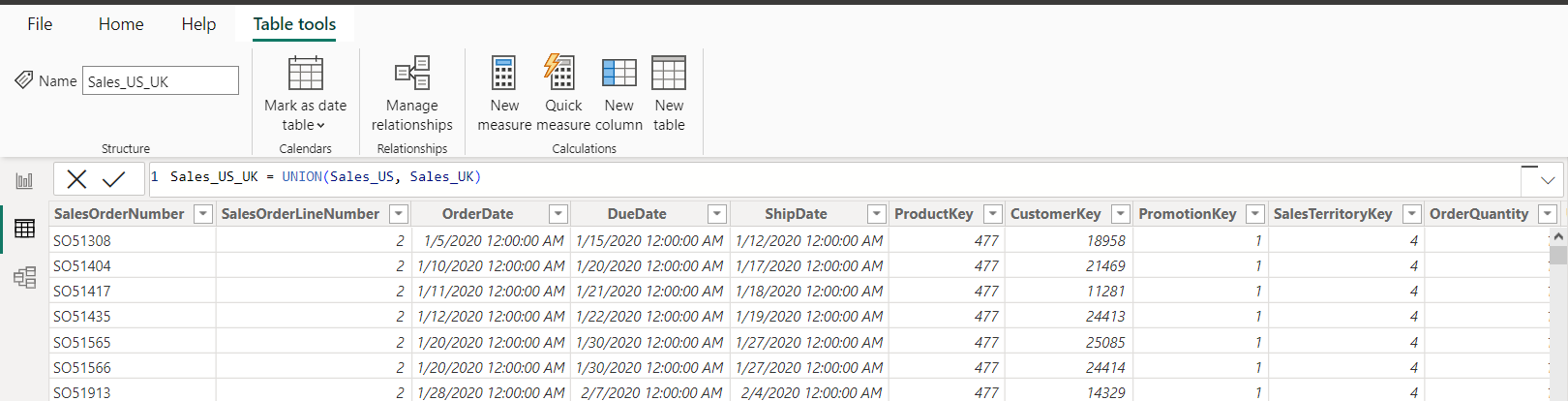
Formula: ADDCOLUMNS\_ex = ADDCOLUMNS('SUMMARIZE\_ex', "Total Cost (USD)", SUMMARIZE\_ex[Total Sales Amount (USD)] - SUMMARIZE\_ex[Total Profit (USD)])



#### **UNION**

* Hàm UNION kết hợp hai hoặc nhiều bảng thành một bảng duy nhất.
* Cú pháp: **UNION(***<table\_expression1>, <table\_expression2> [,<table\_expression>]…***)**
* Ví dụ: Tạo bảng Sales\_US\_UK bằng cách kết hợp 2 bảng Sales\_US và Sales\_UK

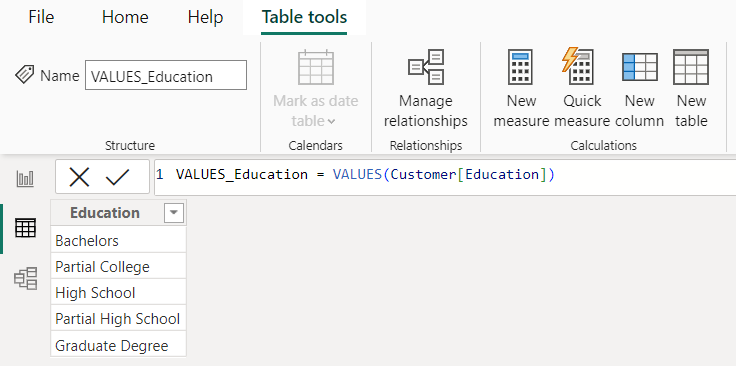
Formula: Sales\_US\_UK = UNION(Sales\_US, Sales\_UK)



#### **VALUES**

* Khi tham số đầu vào là tên cột, hàm trả về bảng một cột chứa các giá trị riêng biệt từ cột được chỉ định. Các giá trị trùng lặp sẽ bị xóa và chỉ trả về các giá trị duy nhất. Một giá trị trống có thể được thêm vào. Khi tham số đầu vào là tên bảng, trả về các hàng từ bảng đã chỉ định. Các hàng trùng lặp được giữ nguyên. Một hàng trống có thể được thêm vào.
* Cú pháp: **VALUES(***<TableNameOrColumnName>***)**
* Ví dụ: Tạo bảng mới chứa các giá trị trình độ học vấn của khách hàng

Formula: VALUES\_Education = VALUES(Customer[Education])



#### **TOPN**

* Hàm TOPN trả về một bảng chứa N dòng hàng đầu dựa trên giá trị được chỉ định trong một cột.
* Cú pháp: **TOPN(***<N\_Value>, <Table>, <OrderBy\_Expression>, [<Order>[, <OrderBy\_Expression>, [<Order>]]…]***)**
* Ví dụ: Tìm top 10 khách hàng mua hàng nhiều nhất

Formula: Top\_10\_Customers =

TOPN(

10,

SUMMARIZE(

'Internet Sales',

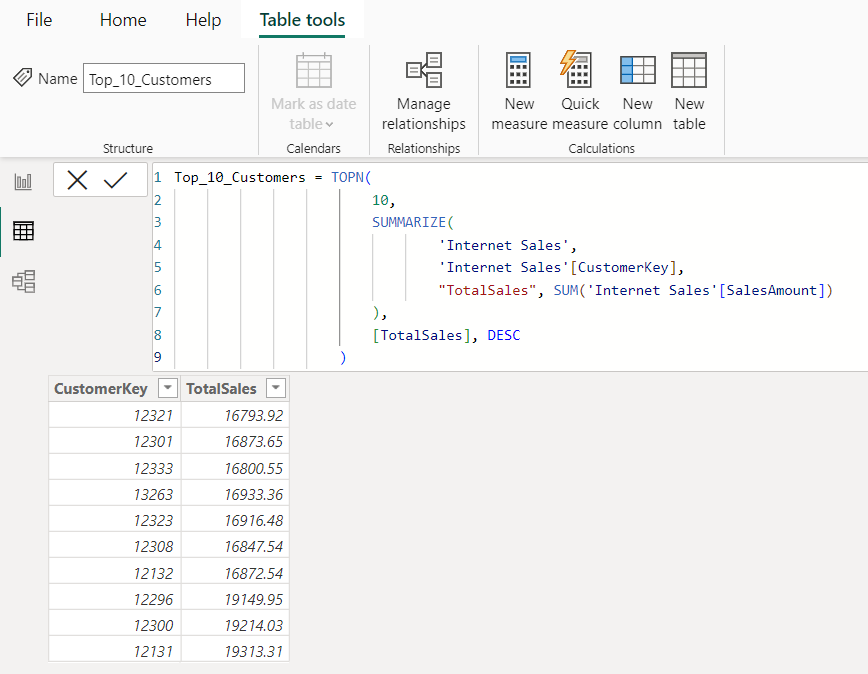
'Internet Sales'[CustomerKey],

"TotalSales", SUM('Internet Sales'[SalesAmount])

),

[TotalSales], DESC

)



### Logical functions:

Các hàm logic tác động lên một biểu thức để trả về thông tin về các giá trị hoặc tập hợp trong biểu thức. Ví dụ: bạn có thể sử dụng hàm IF để kiểm tra kết quả của biểu thức và tạo kết quả có điều kiện.

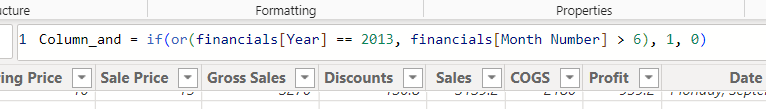
1. **AND**

* Kiểm tra hai đối số nếu cả hai đúng trả về TRUE, ngược lại trả về FALSE.
* Cú pháp: AND(<logical1>, <logical2>)
* Ví dụ:



1. **OR**

* Kiểm tra xem một trong các đối số có TRUE để trả về TRUE hay không. Hàm trả về FALSE nếu cả hai đối số đều FALSE.
* Cú pháp: OR(<logical1>,<logical2>)
* Ví dụ:



1. **IF**

* Kiểm tra một điều kiện và trả về một giá trị khi nó là TRUE, nếu không nó sẽ trả về giá trị thứ hai.
* Cú pháp: IF(<logical\_test>, <value\_if\_true>[, <value\_if\_false>])
* Ví dụ:



1. **BITAND**

* Hàm BITAND được sử dụng để thực hiện phép AND bit (AND logic) giữa hai số. Phép AND bit sẽ trả về kết quả là một số nguyên mới, trong đó mỗi bit tương ứng là kết quả của phép AND bit tương ứng từ hai số ban đầu.
* Cú pháp: BITAND(<number>, <number>)

1. **BITLSHIFT**

* Hàm trả về một số được dịch chuyển sang trái theo số bit đã chỉ định.
* Cú pháp: BITLSHIFT(<Number>, <Shift\_Amount>)

1. **BITOR**

* Hàm trả về một số là kết quả của phép OR bit từ hai số ban đầu.
* Cú pháp: BITOR(<number>, <number>)

1. **BITRSHIFT**

* Trả về một số được dịch sang phải theo số bit đã chỉ định.
* Cú pháp: BITRSHIFT(<Number>, <Shift\_Amount>)

1. **BITXOR**

* Hàm trả về một số là kết quả của phép XOR bit từ hai số ban đầu.
* Cú pháp: BITXOR(<number>, <number>)

1. **COALESCE**

* Trả về biểu thức đầu tiên không đánh giá thành BLANK. Nếu tất cả các biểu thức đánh giá là BLANK, BLANK sẽ được trả về.
* Cú pháp: COALESCE(<expression>, <expression>[, <expression>]…)

1. **FALSE**

* Là giá trị logic FALSE.
* Cú pháp: FALSE()

1. **NOT**

* Thay đổi FALSE thành TRUE hoặc TRUE thành FALSE
* Cú pháp: NOT(<logical>)

1. **TRUE**

* Là giá trị logic FALSE.
* Cú pháp: TRUE()

### Mathematical and trigonometric functions:

Các hàm toán học trong Biểu thức phân tích dữ liệu (DAX) gồm các hàm toán học và lượng giác. Phần này liệt kê các hàm toán học do DAX cung cấp.

1. **ABS**

* Trả về giá trị tuyệt đối của một số
* Cú pháp: ABS(<number>)

1. **ACOS**

* Trả về arccosine hoặc 1/cosin của một số. Góc trả về được tính bằng radian trong phạm vi từ 0 đến pi.
* Cú pháp: ACOS(number)

1. **ACOT**

* Trả về giá trị chính của arccotan hoặc cotan nghịch đảo của một số.
* Cú pháp: ACOT(number)

1. **ASIN**

* Trả về arcsine hoặc sin nghịch đảo của một số. Arcsine là góc có sin bằng số. Góc trả về được tính bằng radian trong phạm vi -pi/2 đến pi/2.
* Cú pháp: ASIN(number)

1. **ATAN**

* Trả về arctang hoặc tang nghịch đảo của một số. Arctang là góc có tiếp tuyến bằng số. Góc trả về được tính bằng radian trong phạm vi -pi/2 đến pi/2.
* Cú pháp: ATAN(number)

1. **CEILING**

* Làm tròn một số lên, đến số nguyên gần nhất hoặc bội số của significance có nghĩa gần nhất.
* Cú pháp: CEILING(<number>, <significance>)

1. **CONVERT**

* Chuyển đổi một biểu thức của kiểu dữ liệu này sang kiểu dữ liệu khác.
* Cú pháp: CONVERT(<Expression>, <Datatype>)

1. **COS**

* Trả về cosin của góc đã cho
* Cú pháp: COS(number)

1. **COT**

* Trả về cotang của một góc được chỉ định bằng radian.
* Cú pháp: COT (<number>), 0 != number < 2^27

1. **CURRENCY**

* Đánh giá đối số và trả về kết quả dưới dạng kiểu dữ liệu tiền tệ.
* Cú pháp: CURRENCY(<value>)

1. **DEGREES**

* Chuyển đổi radian thành độ.
* Cú pháp: DEGREES(angle)

1. **DIVIDE**

* Thực hiện phép chia và trả về kết quả thay thế hoặc BLANK() khi chia cho 0.
* Cú pháp: DIVIDE(<numerator>, <denominator> [,<alternateresult>])

1. **EVEN**

* Trả về số được làm tròn đến số nguyên chẵn gần nhất.
* Cú pháp: EVEN(number)

1. **EXP**

* Trả về e lũy thừa của một số đã cho. Hằng số e bằng 2,71828182845904, cơ số của logarit tự nhiên.
* Cú pháp: EXP(<number>)

1. **FACT**

* Trả về giai thừa của một số.
* Cú pháp: FACT(<number>)

1. **FLOOR**

* Làm tròn một số xuống(về 0), đến bội số có nghĩa gần nhất.
* Cú pháp: FLOOR(<number>, <significance>). Significance là bội số mà bạn muốn làm tròn. Number và significance phải cùng dương hoặc cả hai đều âm.

1. **GCD**

* Trả về ước chung lớn nhất của hai số nguyên trở lên.
* Cú pháp: GCD(number1, [number2], ...)

1. **INT**

* Làm tròn một số xuống số nguyên gần nhất.
* Cú pháp: INT(<number>)

1. **ISO.CEILING**

* Làm tròn một số lên, đến số nguyên gần nhất hoặc bội số có nghĩa gần nhất.
* Cú pháp: ISO.CEILING(<number>[, <significance>]), significance là bội số.

1. **LCM**

* Trả về bội số chung nhỏ nhất của số nguyên.
* Cú pháp: LCM(number1, [number2], ...)

1. **LN**

* Trả về logarit tự nhiên của một số. Logarit tự nhiên dựa trên hằng số e (2.71828182845904).
* Cú pháp: LN(<number>)

1. **LOG**

* Trả về logarit của một số về cơ số bạn chỉ định.
* Cú pháp: LOG(<number>,<base>), base là cơ số logarit.

1. **LOG10**

* Trả về logarit cơ số 10 của một số.
* Cú pháp: LOG10(<number>)

1. **MOD**

* Trả về số dư sau khi một số bị chia cho số chia (số chia phải khác 0). Kết quả luôn cùng dấu với số chia.
* Cú pháp: MOD(<number>, <divisor>)

1. **MROUND**

* Trả về một số được làm tròn thành bội số mong muốn.
* Cú pháp: MROUND(<number>, <multiple>), multiple là bội số muốn làm tròn.

1. **ODD**

* Trả về số được làm tròn lên (xa 0) số nguyên lẻ gần nhất.
* Cú pháp: ODD(number).

1. **PI**

* Trả về giá trị của Pi = 3.14159265358979, chính xác đến 15 chữ số.
* Cú pháp: PI()

1. **POWER**

* Trả về kết quả của một số được lũy thừa.
* Cú pháp: POWER(<number>, <power>)

1. **RADIANS**

* Chuyển đổi độ sang radian.
* Cú pháp: RADIANS(angle)

1. **RAND**

* Trả về một số ngẫu nhiên lớn hơn hoặc bằng 0 và nhỏ hơn 1, phân bố đều. Số được trả về sẽ thay đổi mỗi khi ô chứa hàm này được tính toán lại.
* Cú pháp: RAND()

1. **RANDBETWEEN**

* Trả về một số ngẫu nhiên trong khoảng giữa hai số bạn chỉ định.
* Cú pháp: RANDBETWEEN(<bottom>,<top>)

1. **ROUND**

* Làm tròn một số đến số chữ số được chỉ định.
* Cú pháp: ROUND(<number>, <num\_digits>)

1. **ROUNDDOWN**

* Làm tròn một số xuống.
* Cú pháp: ROUNDDOWN(<number>, <num\_digits>)

1. **ROUNDUP**

* Làm tròn một số lên.
* Cú pháp: ROUNDUP(<number>, <num\_digits>)

1. **SIGN**

* Xác định dấu của một số, kết quả của phép tính hoặc giá trị trong một cột. Hàm trả về 1 nếu số dương, 0 (không) nếu số đó bằng 0 hoặc -1 nếu số âm.
* Cú pháp: SIGN(<number>)

1. **SIN**

* Trả về sin của góc đã cho.
* Cú pháp: SIN(number)

1. **SQRT**

* Trả về căn bậc hai của một số.
* Cú pháp: SQRT(<number>)

1. **TAN**

* Trả về tang của góc đã cho.
* Cú pháp: TAN(number)

1. **TRUNC**

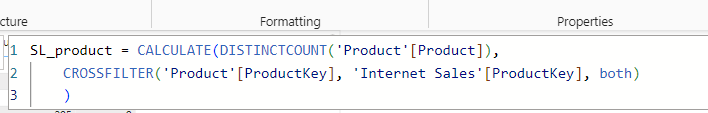
* Cắt bớt một số thành một số nguyên bằng cách loại bỏ phần thập phân hoặc phân số của số đó.
* Cú pháp: TRUNC(<number>,<num\_digits>)

### Relationship functions:

Các hàm trong danh mục này dùng để quản lý và sử dụng mối quan hệ giữa các bảng.

1. **CROSSFILTER**

* Hàm giúp chỉ định hướng lọc chéo sẽ được sử dụng trong phép tính cho mối quan hệ tồn tại giữa hai cột. CROSSFILTER sử dụng các mối quan hệ hiện có trong mô hình, xác định các mối quan hệ theo các cột điểm kết thúc của chúng.
* Cú pháp: CROSSFILTER(<columnName1>, <columnName2>, <direction>)
* Ví dụ: Ta có ba bảng Product, Customer, Internet Sales. Product và Customer có mối quan hệ một chiều hướng vào Internet Sales. Ta cần tính số lượng sản phẩm khác nhau mà mỗi khách hàng dùng dựa vào hai bảng Product và Customer. Khi này ta tạo mối quan hệ hai chiều giữa Product và Internet Sales để có thể tính đúng các giá trị.



A screenshot of a computer

Description automatically generated

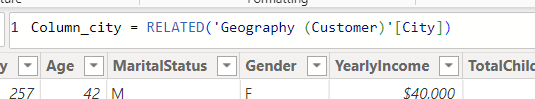
A screenshot of a table

Description automatically generated

* Ta dùng table để thể hiện cột vừa thêm.Nếu không có CROSSFILTER chỉ định mối quan hệ hai chiều thì việc tính số lượng sản phẩm cho mỗi khách hàng sẽ không được chính xác và có giá trị như cột count product bên cạnh.

1. **RELATED**

* Hàm RELATED giúp trả về một giá trị liên quan từ một bảng khác. Nó yêu cầu tồn tại mối quan hệ (nhiều - một) giữa bảng hiện tại và bảng có thông tin liên quan. Nếu một mối quan hệ không tồn tại thì cần phải tạo ra một mối quan hệ.
* Cú pháp: RELATED(<column>)
* Ví dụ: Ghi thêm cột city từ bảng Geography (Customer) vào bảng customer.



A screenshot of a computer

Description automatically generated A screenshot of a table

Description automatically generated

1. **RELATEDTABLE**

* Hàm RELATEDTABLE giúp trả về giá trị tương ứng từ một bảng có liên quan được lọc để nó chỉ bao gồm các hàng có liên quan. Không giống như hàm RELATED, RELATEDTABLE đánh giá các hàng dựa trên các bộ lọc được áp dụng.
* Cú pháp: RELATEDTABLE(<tableName>)
* Ví dụ: Tạo thêm cột Total\_product trong bẳng customer ứng với mỗi khách hàng sẽ có tổng số product người đó đã mua. Khi ấy ta dùng hàm relatedtable trong để lọc trong bảng Internet Sales và tính toán.

A screenshot of a computer

Description automatically generated

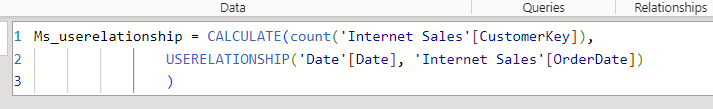
A screenshot of a computer

Description automatically generated

* Ứng với mỗi customerKey ta có tổng số lượng sản phẩm mà người đó đã mua.

1. **USERELATIONSHIP**

* Hàm chỉ định mối quan hệ sẽ được sử dụng trong một phép tính cụ thể như mối quan hệ tồn tại giữa cột Name1 và cột Name2 của hai bảng. Chỉ có thể sử dụng hàm USERELATIONSHIP đối với các bảng đã có mối quan hệ.
* Cú pháp: USERELATIONSHIP(<columnName1>,<columnName2>)
* Ví dụ: Ta muốn đếm số lượng customerkey trong bảng Internet Sales theo date trong bảng Date ta dùng USERELATIONSHIP xác định mối quan hệ giữ hai bảng và thực hiện tính toán.



A screenshot of a data

Description automatically generated

* SL khách hàng theo oderdate được chỉ định khác với SL khách hàng đếm theo cột thời gian bên cạnh

### Statistical functions:

Các hàm thống kê tính toán các giá trị liên quan đến phân bố thống kê và xác suất, chẳng hạn như độ lệch chuẩn và số lượng hoán vị.

1. **BETA.DIST**

* Trả về bản phân phối beta. Phân phối beta thường được sử dụng để nghiên cứu sự thay đổi về tỷ lệ phần trăm của một cái gì đó trên các mẫu.
* Cúp pháp: BETA.DIST(x, alpha, beta, cumulative, [A], [B]), với:
  + x: giá trị giữa A và B để đánh giá hàm.
  + alpha, beta: là các tham số của phân phối.
  + A, B: giá trị không bắt buộc, là giới hạn dưới và trên của x.

1. **CHISQ.DIST**

* Trả về phân bố chi bình phương.
* Cú pháp: CHISQ.DIST(<x>, <deg\_freedom>, <cumulative>), với:
  + x: Giá trị mà bạn muốn đánh giá phân phối.
  + deg\_freedom: Bậc tự do.
  + Cumulative: Một giá trị logic xác định dạng của hàm. Nếu là TRUE, trả về hàm phân phối tích lũy; nếu FALSE, nó trả về hàm mật độ xác suất.

1. **CHISQ.DIST.RT**

* Trả về xác suất bên phải của phân bố chi bình phương. Phân phối chi bình phương được liên kết với phép kiểm định chi bình phương.
* Cú pháp: CHISQ.DIST.RT(<x>, <deg\_freedom>)

1. **CHISQ.INV**

* Trả về nghịch đảo của xác suất bên trái của phân bố chi bình phương.
* Cú pháp: CHISQ.INV(probability,deg\_freedom)

1. **COMBIN**

* Trả về số kết hợp cho một số mục nhất định. Sử dụng COMBIN để xác định tổng số nhóm có thể có cho một số mục nhất định.
* Cú pháp: COMBIN(number, number\_chosen), với:
  + Number: số lượng items.
  + number\_chosen: số lượng items trong mỗi kết hợp.

1. **COMBINA**

* Trả về số lượng kết hợp (có số lần lặp lại) cho một số mục nhất định.
* Cú pháp: COMBINA(number,number\_chosen), với: number>=0, number\_chosen, number\_chosen >= 0

1. **CONFIDENCE.NORM**

* Khoảng tin cậy là một phạm vi giá trị.
* Cú pháp: CONFIDENCE.NORM(alpha,standard\_dev,size), với:
  + Alpha: Mức ý nghĩa được sử dụng để tính mức độ tin cậy.
  + standard\_dev: Độ lệch chuẩn tổng thể cho phạm vi dữ liệu.
  + Size: kích thước tập mẫu.

1. **EXPON.DIST**

* Trả về phân bố mũ.
* Cú pháp: EXPON.DIST(x, lambda, cumulative), với:
  + X: giá trị của hàm (bắt buộc).
  + Lambda: giá trị tham số (bắt buộc).
  + Cumulative: Một giá trị logic cho biết dạng hàm mũ nào sẽ được cung cấp. Nếu là TRUE, trả về hàm phân phối tích lũy; nếu FALSE, nó trả về hàm mật độ xác suất.

1. **GEOMEAN**

* Trả về trung bình hình học của các số trong một cột.
* Cú pháp: GEOMEAN(<column>)

1. **GEOMEEANX**

* Trả về giá trị trung bình hình học của một biểu thức được đánh giá cho mỗi hàng trong bảng.
* Cú pháp: GEOMEANX(<table>, <expression>), với:
  + Table: Bảng chứa các hàng mà biểu thức sẽ được đánh giá.
  + Expression: Biểu thức được đánh giá cho mỗi hàng của bảng.

1. **LINEST**

* Sử dụng phương pháp bình phương tối thiểu để tính đường thẳng phù hợp nhất với dữ liệu đã cho, sau đó trả về bảng mô tả đường thẳng đó. Phương trình của đường thẳng có dạng: y = Slope1\*x1 + Slope2\*x2 + ... + Intercept.
* Cú pháp: LINEST ( <columnY>, <columnX>[, …][, <const>] ), với:
  + Const: Nếu TRUE hoặc bị bỏ qua, thì giá trị Intercept sẽ được tính bình thường; Nếu FALSE, giá trị Intercept được đặt thành 0.

1. **MEDIAN**

* Trả về giá trị trung bình của các số trong một cột.
* Cú pháp: MEDIAN(<column>)

1. **MEDIANX**

* Trả về số trung vị của một biểu thức được đánh giá cho mỗi hàng trong bảng.
* Cú pháp: MEDIANX(<table>, <expression>), expression: cột cần tính trong bảng đã nêu.

1. **NORM.DIST**

* Trả về phân phối chuẩn cho giá trị trung bình và độ lệch chuẩn đã chỉ định.
* Cú pháp: NORM.DIST(X, Mean, Standard\_dev, Cumulative), với:
  + X: giá trị muốn tính phân phối.
  + Cumulative: giá trị logic xác định dạng hàm. Nếu là TRUE, trả về hàm phân phối tích lũy; nếu FALSE, nó trả về hàm mật độ xác suất.

1. **NORM.S.DIST**

* Trả về phân phối chuẩn chuẩn (có giá trị trung bình bằng 0 và độ lệch chuẩn là 1).
* Cú pháp: NORM.S.DIST(Z, Cumulative)

1. **POISSON.DIST**

* Trả về phân phối Poisson.
* Cú pháp: POISSON.DIST(x,mean,cumulative), với:
  + X : Số sự kiện.
  + Mean: giá trị số sự kiện dự kiến.

1. **RANK.EQ**

* Trả về thứ hạng của một số trong danh sách các số.
* Cú pháp: RANK.EQ(<value>, <columnName>[, <order>])

1. **STDEV.P**

* Trả về độ lệch chuẩn của toàn bộ tập hợp.
* Cú pháp: STDEV.P(<ColumnName>)

1. **STDEV.S**

* Trả về độ lệch chuẩn của một tập hợp mẫu.
* Cú pháp: STDEV.S(<ColumnName>)

1. **T.DIST**

* Trả về phân phối t bên trái của student.
* Cú pháp: T.DIST(X, Deg\_freedom, Cumulative)

1. **VAR.P**

* Trả về phương sai của toàn bộ tập hợp.
* Cú pháp: VAR.P(<columnName>)

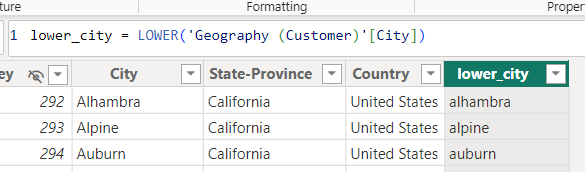
1. **VAR.S**

* Trả về phương sai của một tập hợp mẫu.
* Cú pháp: VARX.S(<table>, <expression>)

### Text functions:

1. **LOWER**

* Chuyển đổi tất cả các chữ cái trong chuỗi văn bản thành chữ thường.
* Cú pháp: LOWER(<text>)
* Ví dụ: tạo cột viết thường tên thành phố.



1. **CONCATENATE**

* Nối hai chuỗi văn bản thành một chuỗi văn bản
* Cú pháp: CONCATENATE(<text1>, <text2>)

1. **LEN**

* Trả về số ký tự trong một chuỗi văn bản
* Cú pháp: LEN(<text>)

1. **TRIM**

* Xóa tất cả khoảng trắng khỏi văn bản ngoại trừ khoảng trắng đơn giữa các từ.
* Cú pháp: TRIM(<text>)

1. **UPPER**

* Chuyển đổi một chuỗi văn bản thành tất cả các chữ in hoa.
* Cú pháp: UPPER (<text>)

1. **REPLACE**

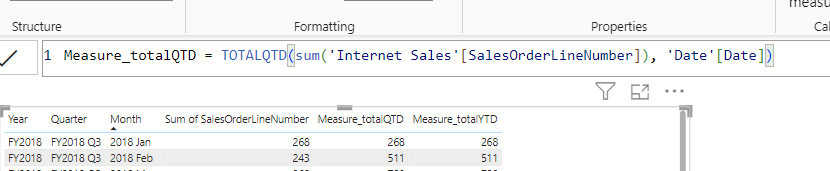
* REPLACE thay thế một phần của chuỗi văn bản, dựa trên số ký tự bạn chỉ định, bằng một chuỗi văn bản khác.
* Cú pháp: REPLACE(<old\_text>, <start\_num>, <num\_chars>, <new\_text>)

### Time intelligence functions:

Biểu thức phân tích dữ liệu (DAX) bao gồm các hàm thông minh về thời gian cho phép bạn thao tác dữ liệu bằng các khoảng thời gian, bao gồm ngày, tháng, quý và năm, sau đó xây dựng và so sánh các phép tính trong các khoảng thời gian đó.

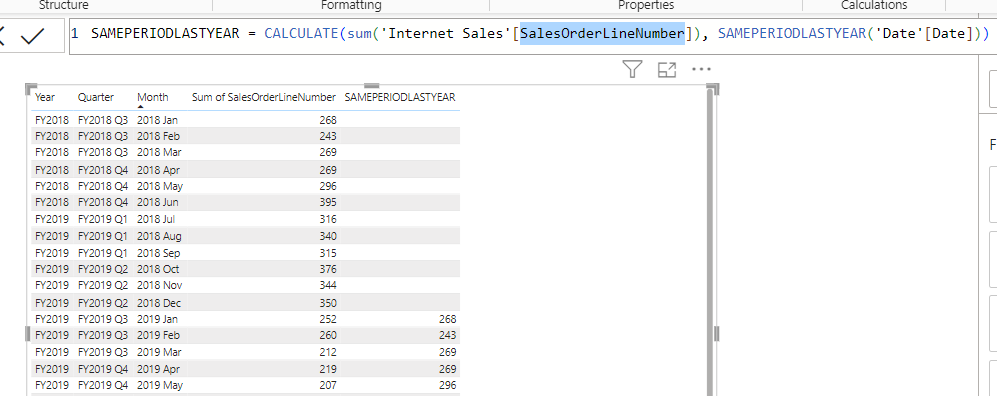
1. **TOTALQTD**

* Tính tổng các giá trị theo từng quý và thời gian được tính cho đến ngày hiện tại.
* Cú pháp: TOTALQTD(<expression>,<dates>[,<filter>])
* Ví dụ:



1. **SAMEPERIODLASTYEAR**

* Trả về một bảng chứa cột ngày được dịch chuyển ngược thời gian một năm so với những ngày trong cột ngày đã chỉ định, trong ngữ cảnh hiện tại (cùng kì năm ngoái).
* Cú pháp: SAMEPERIODLASTYEAR(<dates>)
* Ví dụ:



1. **DATESBETWEEN**

* Trả về một bảng chứa cột ngày được bắt đầu bằng ngày đã chỉ định và tiếp tục cho đến ngày kết thúc đã chỉ định. Hàm này phù hợp để chuyển làm bộ lọc cho hàm CALCULATE. Sử dụng nó để lọc một biểu thức theo phạm vi ngày tùy chỉnh.
* Cú pháp: DATESBETWEEN(<Dates>, <StartDate>, <EndDate>)

1. **FIRSTDATE**

* Trả về ngày đầu tiên trong ngữ cảnh hiện tại cho cột ngày đã chỉ định.
* Cú pháp: FIRSTDATE(<dates>)

1. **STARTOFMONTH**

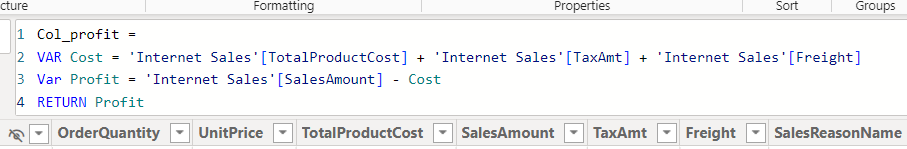
* Trả về ngày đầu tiên của tháng trong ngữ cảnh hiện tại cho cột ngày đã chỉ định.
* Cú pháp: STARTOFMONTH(<dates>)

## Variable

Biến (Variable) thường được biết đến qua cấu trúc lệnh VAR/RETURN, là một tính năng của PowerBI, nhằm mục đích giúp người dùng viết và quản lý công thức DAX một cách hiệu quả. Ta có thể hiểu biến giống như tên cho một biểu thức DAX. Thật vậy, thuật ngữ “variable” có thể gây ra hiểu lầm, bởi một biến trong DAX không thể thay đổi giá trị, mà nó là một hằng số. Nghĩa là ta tính toán một giá trị bằng biểu thức DAX, sau đó lưu giữ giá trị đó trong một tên biến.

### Cú pháp và cấu trúc của biến:

* Cú pháp: VAR <Tên biến> = <Biểu thức DAX> RETURN <Kết quả>
* Ví dụ: Ta thực hiện ví dụ tính profit:



A screenshot of a computer

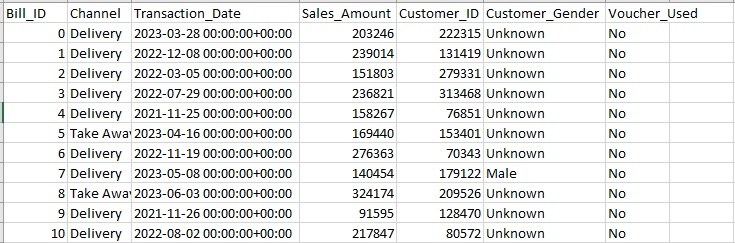
Description automatically generated

### Lợi ích:

* Dễ đọc: Bằng cách gán tên cho một biểu thức, ta đang thêm thông tin mô tả vào code của mình, giúp ta dễ dàng đọc và xác định các thành phần của DAX.
* Tách code thành các bước: Bộ não chúng ta dễ dàng nhận diện và xử lý các biểu thức code có chứa biến tốt hơn, bởi nó cho phép ta hình dung các bước thực thi để xây dựng lên biểu thức.
* Dễ debug: Nhìn vào các phần code đã được phân chia rõ ràng thành từng khối theo từng biến, ta có thể dễ dàng xác định được phần nào đang gặp lỗi để từ đó có cách xử lý nhanh chóng, hiệu quả. Nếu không sử dụng biến, mỗi khi xuất hiện bug, ta sẽ phải lần mò từng đoạn code để xác định tác dụng của nó và xem xét xem liệu nó đã hoạt động đúng cách hay chưa.
* Cải thiện hiệu năng code: Với các biến, ta đang cho DAX biết về những phần nào của toàn bộ biểu thức có thể được tính một lần và lưu lại để sử dụng sau này.

# Demo

## Bối cảnh

* Một cửa hàng pizza đang muốn phân loại khách hàng của họ thành các phân khúc khác nhau để thấy được hành vi đặc trưng của từng nhóm khách hàng nhằm tăng chất lượng phục vụ cũng như đưa ra nhiều chương trình khuyến mãi phù hợp cho từng phân khúc khách hàng.
* Về mặt data, họ đang có một tập dữ liệu về tình hình kinh doanh từ 30/09/2021 đến 30/06/2023. Với mỗi dòng dữ liệu chứa thông tin về một hoá đơn, ngày mua, kênh mua (Dine in, Take away, Delivery), tổng tiền, mã khách hàng mua, giới tính và đơn đó có áp dụng voucher gì hay không.

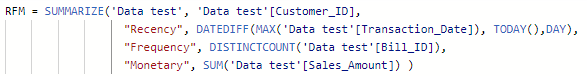
## Hướng giải quyết

* Áp dụng mô hình phân tích RFM (Recency, Frequency, Monetary): đây là mô hình phân tích và phân khúc khách hàng dựa vào đặc trưng hành vi trong dữ liệu giao dịch trong quá khứ.
* RFM sẽ tính điểm của mỗi khách hàng dựa trên 3 yếu tố:
* Recency: khoảng cách giữa ngày mua hàng gần nhất với hiện tại. Khách hàng có recency càng cao thì khả năng khách hàng đó rời bỏ cửa hàng càng cao.
* Frequency: Tổng số lần mua hàng của một khách hàng. Frequency càng cao thì khả năng khách hàng đó sẽ trở thành khách hàng trung thành của cửa hàng càng cao.
* Monetary: Tổng tiền khách hàng đã chi cho việc mua hàng tại cửa hàng. Monetary càng cao thì khả năng tiếp cận up-sell hay cross-sell càng cao.
* Áp dụng phương pháp phân tích RFM bằng cách sử dụng DAX trong Power BI và tạo một dashboard để thể hiện kết quá.

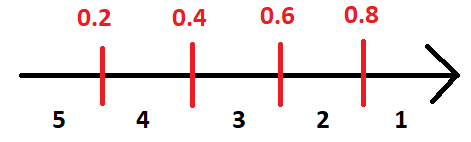
## Thực hiện

### Tính điểm RFM: sử dụng DAX, chia dữ liệu thành 5 phần bằng cách dùng hàm Percentile tương ứng từ 1  5 điểm.

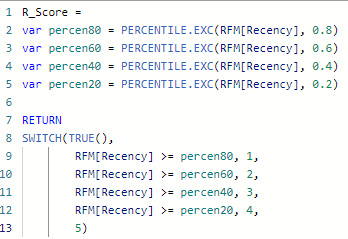
* Tạo bảng RFM bằng DAX.



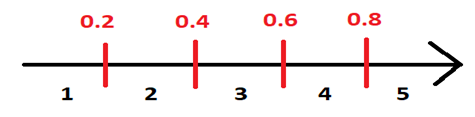
* Recency: R càng thấp thì điểm càng cao.



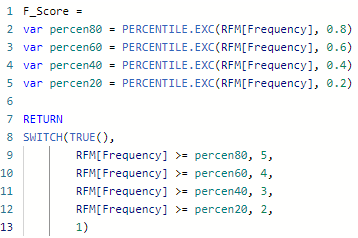
* Code DAX:



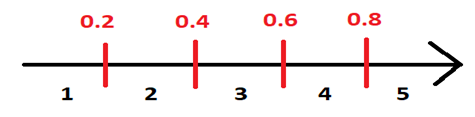
* Frequency: F càng cao thì điểm càng cao.



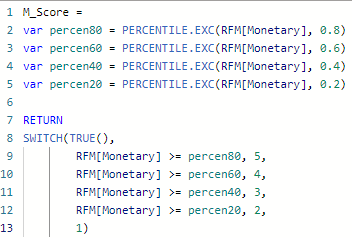
* Code DAX:



* Monetary: M càng cao thì điểm càng cao.



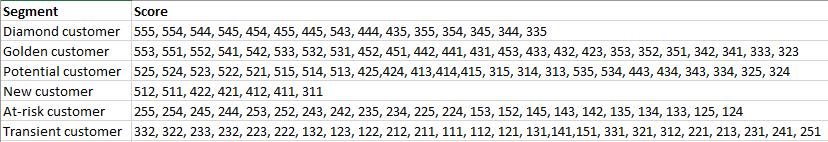
* Code DAX:



### Map điểm RFM vừa tính với phân khúc khách hàng tương ứng

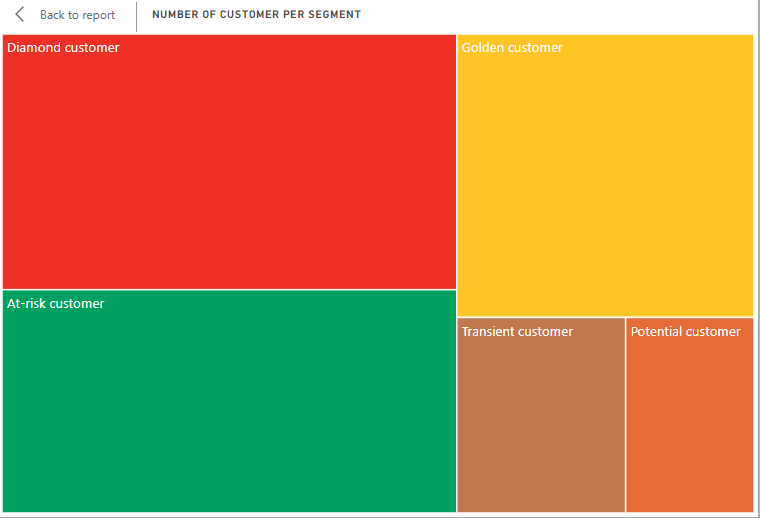
* Kết hợp 3 điểm R, F, M ta được chuỗi RFM.

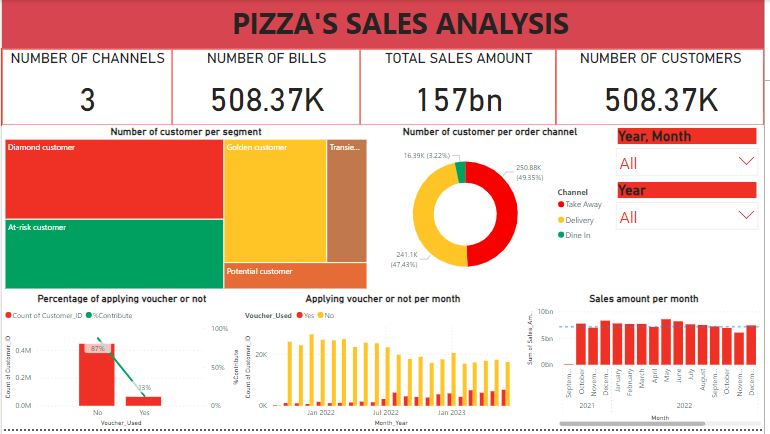


* 555: có mua hàng gần đây, đã mua hàng nhiều lần, mua đơn giá trị cao  phân khúc cao (diamond customer).
* 253: lần mua hàng gần nhất cách đây khá lâu, mua hàng nhiều lần, giá trị đơn khá lớn  tại sao gần đây họ không mua hàng nữa? (at-risk customer).
* 512: mua hàng gần đây, số lần mua hàng và giá trị đơn chưa cao  Khách hàng mới (new customer).
* Map điểm tính được với phân khúc khách hàng tương ứng.
* Code DAX:



## Kết quả

* Dùng Treemap trong Power BI để thể hiện kết quả
* Tạo dashboard để phân tích hành vi của từng nhóm khách hàng.



# Tài liệu tham khảo

* [**https://blog.tomorrowmarketers.org/phan-tich-rfm-la-gi/**](https://blog.tomorrowmarketers.org/phan-tich-rfm-la-gi/)
* [**https://learn.microsoft.com/en-us/dax/**](https://learn.microsoft.com/en-us/dax/)
* [**https://www.geeksforgeeks.org/dax-aggregate-functions-in-power-bi/**](https://www.geeksforgeeks.org/dax-aggregate-functions-in-power-bi/)
* [**https://www.geeksforgeeks.org/power-bi-dax-date-functions/**](https://www.geeksforgeeks.org/power-bi-dax-date-functions/)
* [**https://www.geeksforgeeks.org/power-bi-dax-filter-functions/**](https://www.geeksforgeeks.org/power-bi-dax-filter-functions/)
* [**https://www.geeksforgeeks.org/power-bi-dax-information-functions/**](https://www.geeksforgeeks.org/power-bi-dax-information-functions/)