**M 365 Excel Class Video 12: Power Query M Code Diagram

Description automatically generated with medium confidence**

**Table of Contents**

[What does Power Query do and how does it work? (from Only App Matter Book) 2](#_Toc129780075)

[Excel and Power BI Power Query Ribbon Tabs 3](#_Toc129780076)

[Power Query Editor Window in Excel 4](#_Toc129780077)

[Power Query Editor Window in Power BI Desktop 4](#_Toc129780078)

[Excel Power Query Load To 5](#_Toc129780079)

[Power BI Desktop Close & Apply 5](#_Toc129780080)

[The story of how M Code works in Power Query: 6](#_Toc129780081)

[Microsoft Power Query M Code Specifications Guide 6](#_Toc129780082)

[Define Power Query 6](#_Toc129780083)

[Values possible in Power Query 6](#_Toc129780084)

[Power Query Data Types 7](#_Toc129780085)

[Data Types vs. Values 7](#_Toc129780086)

[Where you can view M Code 8](#_Toc129780087)

[Expressions 8](#_Toc129780088)

[Identifiers 8](#_Toc129780089)

[Keywords 9](#_Toc129780090)

[Operators and Punctuators 9](#_Toc129780091)

[Standard Library 9](#_Toc129780092)

[M Code Tips: #shared, Tab, Linefeed, Return, Lists of Numbers & Letters and Joining Lists, Records & Tables 9](#_Toc129780093)

[Let expression 10](#_Toc129780094)

[M Code Lookup 12](#_Toc129780095)

[Primary Keys 14](#_Toc129780096)

[Custom Functions 15](#_Toc129780097)

[each and underscore 18](#_Toc129780098)

# What does Power Query do and how does it work? (from Only App Matter Book)

Text

Description automatically generated

# Excel and Power BI Power Query Ribbon Tabs

**Excel (Figure 18.64):**

Graphical user interface, application, table, Excel

Description automatically generated

**Power BI (Figure 18.65):**

Graphical user interface

Description automatically generated with low confidence

# Power Query Editor Window in Excel

Graphical user interface, application, Word

Description automatically generated

# Power Query Editor Window in Power BI Desktop

Graphical user interface, application

Description automatically generated with medium confidence

# Excel Power Query Load To

Graphical user interface, text, application, email

Description automatically generated

# Power BI Desktop Close & Apply

Table

Description automatically generated

# The story of how M Code works in Power Query:

## Microsoft Power Query M Code Specifications Guide

<https://learn.microsoft.com/en-us/powerquery-m/power-query-m-language-specification>

## Define Power Query

* Power Query is a tool to import, clean, transform and load data, or as Microsoft calls it “Data Mashup”.
* The amazing tool Power Query is in Excel and Power BI Desktop (Power BI Online has a similar tool called Data Flows).
* Power Query uses a case sensitive, function-based M Code language to create **Values**.

## Values possible in Power Query

Table

Description automatically generated

Table

Description automatically generated

## Power Query Data Types

Table

Description automatically generated

## Data Types vs. Values

1. **Data Types** on Fields aid consistent data in field and help create accurate calculations from the data.
2. Power Query **Values** are the possible outputs that M Code can deliver.

## Where you can view M Code

1. Applied Steps = List of query step names, called identifiers (As shown below).
2. Formula Bar = Shows M Code formula for selected step in the Applied Step list (As shown below).
3. Advanced Editor = Shows all the M Code for a given query, which consists of the let expression with all its query steps that work in succession to deliver the final value of the query (As shown below).

Picture illustrates the three places to view and edit M Code:

Graphical user interface, application

Description automatically generated

## Expressions

* An **expression** is any M Code that results in a value.
* Some examples of expressions:
  + A function, like:
    - = Table.Group(Source, {"TypeDay"}, {{"TotalSales", each List.Sum([Sales])}}).
  + A list, like: = {1,2,3}.
  + A number and math operator, like: = 43 + 7.
  + A query step within a let expression, like:
    - DayTypeSales = Table.Group(Source, {"TypeDay"}, {{"TotalSales", each List.Sum([Sales])}}),
  + A full let expression, as shown in the above picture.

## Identifiers

* An **identifier** is used to refer to an expression (value).
  + If the identifier has a space, you must distinguish it from text by recording it with a leading # sign and place identifier in quotes, such as: #"Identifier Name".
  + If there is no space, you record the identifier as: IdentifierName.
* **Generalized Identifier** are identifiers that allow spaces without the # sign and quotes.
  + Generalized Identifiers are allowed in either:
    1. The name of a Field in a Record Literal like: [Boom Product = "Quad", Sales = 43]  
       or
    2. The name of a Field in a Field Access Operator like: [Boom Product].

## Keywords

* Reserved words that have an assigned meaning and not be used as identifiers for expressions.
* Here are some of the Keywords:



## Operators and Punctuators

* There are several kinds of operators and punctuators.
* Operators are used in expressions to describe operations involving one or more operands. For example, the expression a + b uses the + operator to add the two operands a and b.
* Punctuators are for grouping and separating. For Example { } for grouping items in a list: {1, 2, 3}.
* Examples:

Online article about operators; <https://learn.microsoft.com/en-us/powerquery-m/operators>

## Standard Library

* Standard Library is a list of built-in Functions and constants that deliver values.
* The Power Query Excel Standard Library has 837 items and the Power BI Desktop Power Query Standard Library has 1032 items (many more data connectors)
* Four examples of Standard Library:
  + Excel.CurrentWorkbook() function.
  + Number.Round() function.
  + Number.E or Number.PI (delivers the number e or pi).

## M Code Tips: #shared, Tab, Linefeed, Return, Lists of Numbers & Letters and Joining Lists, Records & Tables

* If you type **#shared** into power Query Editor Formula Bar, you get a full list of all elements in Standard Library. If you convert to a Table, you can search the list.
* Carriage-return, linefeed, or tab character in a text literal, the #(cr), #(lf), and #(tab).
* If you type the list: {1.. 43}, you get a list of numbers 1 to 43.
* If you type {"a".."z"} you get a lower case alphabet.
* You can join two lists with Join Operator, like: {1,3}&{7,43} = {1,3,7,43}
* You can join two records with Join Operator, like: [A=3]&[B=43] = [A=3, B=43]
* You can append tables using the Join Operator, like: Table01&Table02, as shown here:

Graphical user interface, text, application

Description automatically generated

## Let expression

* let expression allows you to define variables (steps) and use them throughout the let expression to define a final value. let expressions can be used in any M Code, but is most often used to define a new query.
* The rules for a let expression are:
  1. Start the let expression with lower case let.
  2. Each variable, or step, starts with an identifier, then an equal sign, then an expression.
  3. Variables are usually used in subsequent steps, can be used anywhere throughout the let expression, but cannot be used outside the let expression.
  4. Each variable is followed by a comma.
  5. The last query step does not end with a comma.
  6. End the let expression with lower case in followed by the identifier for the final value to be delivered by the new query. The result of the query is almost always the identifier of the last query step.
* Visual summary of let rules:

Text

Description automatically generated

* A **new query** created with the Power Query tool uses a let expression:
  1. In a query, a let expression is designed to combine all query steps to deliver a final value.
  2. The query name is the identifier for the let expression. This identifier can be used throughout the workbook.
  3. Example of a new query that uses let to deliver a Sales By Day Type Report:

Graphical user interface, text, application, email

Description automatically generated

* You **can use the let expression anywhere**, here is an example of a let expression that is being used inside the Table.AddColumn function:

Graphical user interface, text, application, email

Description automatically generated

* You can add **notes** to M Code with the following syntax:
  + Start the note with two forward slashes, like //, followed by written note (before a hard return).
  + If you have multiple lines, start at the first line with forward slash and an asterisk, /\*, followed by as many lines (hard returns) as you want, and then end with asterisk and forward slash, \*/.
  + Example is shown on next page:

Graphical user interface, text, application, email

Description automatically generated

## M Code Lookup

* Note: M Code is "base zero", which means: Row 1 = 0, Row 2 = 1, Row 3 = 2, and so on…
* M Code Lookup is similar to lookup use the Excel Worksheet function INDEX because INDEX can lookup value from a table based on a row index number and column index number, Like:

**INDEX( Array, Index Number, Column Number)**

* Two Types of M Code Lookup:
  1. **Row Index Lookup**
     + M Code Syntax:

**Table { Row Index Number }[ Field Name ]**

* + - This method uses a hard coded row number to determine the row position of the lookup. The row position does not change when the sort or content of the column changes.
    - This method uses a hard coded row index number in curly brackets (called **positional index operator**) and a field name in square brackets (called a **field access operator**) to get a value from a table at the intersection of the row number and designated column.
    - Examples:
      * **Source{0}[Content]** to get the first item from the Content field in the Source table.
      * **Source{0}** to get the first record from the Source table.
      * **Source[Content]** to get the Content field from the Source table as a list.
      * **Source[[Content]]** to get the Content field from the Source table as a field.
      * **{43,86}{0} = 43** (get the first item from the list).
      * **{43,86}{2} = error** (error because there is not a third item).
      * **{43,86}{2}? = null** (? is Optional Operator which avoids error and delivers a null when a match is not made)
  1. **Key Match Lookup:** 
     + M Code Syntax:

**Table{[Field Name = LookupValue]}[Field Name]**

* + - This method uses an exact match lookup (called **key match lookup**) to dynamically determine the row position based on a lookup value in a specified column. The row position changes when the sort or content of the column changes. You use this type of lookup when you have a field that contains a unique list, or the field is the primary key in the table.
    - This method uses a key match logical test inside square brackets (called **lookup operator**) inside the positional index operator to dynamically determine the row position of the lookup. The field name in the field access operator determines the column position. Then the intersecting value is retrieved from the table.
    - Examples:
      * **Source{[Product=”Sunshine”]}[Content]** to get the item in the Content field that corresponds to the row position of the Sunshine item in the Product field.
      * **Source{[Product=”Sunshine”]}** to get the record in the Source table where the item in the Product field is equal to Sunshine. There are no duplicates in the Product field and so it works.
      * **Source{[Product=”Quad”]}** yields the error: “The key matched more than one row in the table” because there was more than one “Quad” in the Product field.
      * **Source{[Product=”Aspen”]}** yields the error: “The key didn't match any rows in the table” because there were no Aspen items in Product field.
* Pictorial Summaries of M Code Lookup on next page:

Diagram

Description automatically generated

Text

Description automatically generated with medium confidence

Diagram

Description automatically generated

## Primary Keys

* Primary Keys for Power Query Tables are nearly invisible in Power Query. And yet they play a significant role in doing Lookup in Power Query and for some calculations such as Group By Aggregate.
* There is no way to see the Primary Keys in the User Interface.
* You can determine if a Table has a Primary key by using the **Table.Keys function**, as shown in below example:
* Power Query will define a Primary Key in a Table in these situations:
  + If you use the Remove Duplicates feature, i.e., the Table.Distinct function.
  + If you are connected to a database like an SQL database and a Primary Key has been defined in a table that you import.
  + If you use **Table.AddKey function** to add a primary key to a table.
  + When you use Excel.CurrentWorkbook() function to import Excel Tables from the current workbook, the table of objects that the function creates, adds a primary key to the object name field names “Name”.
* The main area in Power Query where a primary key comes into play is with the Drill Down feature.
  + If you Drill Down on a Primary Key, Key Match Lookup is preformed, like:

**= Excel.CurrentWorkbook(){[Name=""fTransactions""]}[Content]"**

* + If you Drill Down on a non-Primary Key, Row Index Lookup is preformed, like:

**= Excel.CurrentWorkbook(){0}[Content]"**

* Primary Key Example:

Graphical user interface, text, application

Description automatically generated

## Custom Functions

* An M Code Custom Function is a user-created function that defines the variables and the mapping for those variables to deliver a value.
* Custom functions can be created in any bit of M Code, but are most often created in three places:
  1. In a new query as a re-usable function (in a let expression that delivers a custom function value that can be used throughout the workbook).
  2. In function arguments that require functions
  3. As a step in a let expression
* Syntactical rules for defining a Custom Function are:
  + - * Type variables (function inputs) separated by commas in parentheses.
      * Type the go to operator: => (Equal, Greater Than).
      * Type a formula that uses zero or more of the variables.
      * Two Examples:
        + To add the inputs x and y:

**(x, y) => x + y**

* + - * + Calculate the effective rate based on the inputs APR and Periods:

**(APR, Periods) => Number.Power(1+APR/Periods,Periods)-1**

* Example of a Custom Function in a Custom Column that helps to perform approximate match lookup:

Table

Description automatically generated

* Example of a let expression in a Custom Column that helps to perform approximate match lookup:

Graphical user interface, application, table

Description automatically generated

* Example of a Re-usable Custom Function created in the Advanced Editor that calculates the effective rate:

Graphical user interface, text, application, email

Description automatically generated

* You can define data types for your Custom Function like this:

Timeline

Description automatically generated

* You can add notes to your Custom Function:
  1. Use two forward slashes, like // , to add a single line with notes (one hard return)
  2. When you have multiple lines (multiple hard returns) use /\* to start and then \*/ to close.
  3. Examples below:

Graphical user interface, text, application, email

Description automatically generated

* Four Examples of an Effective Rate Custom Function:

Graphical user interface, text, application

Description automatically generated

## each and underscore

* each = syntactical shorthand for defining an unnamed function taking a single untyped variable named underscore \_ .
* You can use each shorthand anywhere a function can be declared.
* each is often used to pass a function to an argument in another function, like Table.AddColumn.
* each can be thought of as "allowing you to make a calculation in each row of a table or list.
* The Underscore ( \_ ) can be thought of as extracting everything from the row the function is working in.
* Examples of equivalent Custom Functions with different syntax:  
  1. = Table.SelectRows(Source, each [TypeDay] = "Workday")
  2. = Table.SelectRows(Source, (\_) => \_[TypeDay] = "Workday")
  3. = Table.SelectRows(Source, (x) => x[TypeDay] = "Workday")
* Examples of equivalent Custom Functions with different syntax to extract a record from a table in a Custom Column:  
  1. = Table.AddColumn(CustomFunction, "Custom", each \_)
  2. = Table.AddColumn(CustomFunction, "Custom", (\_) => \_)
  3. = Table.AddColumn(CustomFunction, "Custom", (Record) => Record)

Here is picture of advanced editor with above examples of the different syntax available for Custom Functions:

Graphical user interface, text, application, email

Description automatically generated