

Power BI desktop

Pavel Lasák - Excel, Power BI, G-tabulky



Power BI Desktop

Načtení



Soubory (xls, csv, pdf)
Složky
Web
Databáze

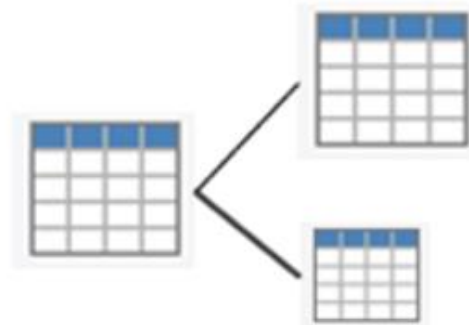
Úprava



Práce s řádky
Úpravy se sloupci
Řazení filtrování
Transformace

M - jazyk

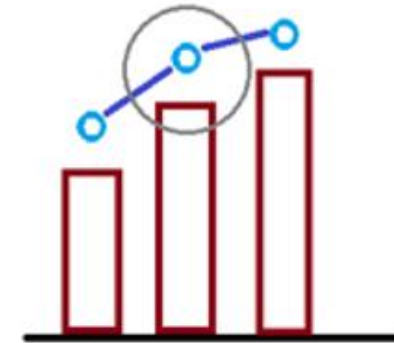
Propojení



Propojení, slučování
Hierarchie
Vazby

DAX

Vizualizace



Grafy
Ovládací prvky
Texty
Obrázky

Sdílení



Pdf
Csv
Cloud
....



Power Query – M-ko

- M-kód
- O co jde
- Jaká je syntaxe
- Jak zprovoznit
- Objekty v M-language (List, Záznam, Tabulka)
- Konstrukce Let in
- Funkce
- Konstrukce If then, Each, _

O jazyku M

A

- Označování

- M-kód
- M-jazyk
- M-code
- M-language
- Query jazyk
- ...

O jazyku M

- Ústřední konstrukcí jazyka M je *výraz*.
- Výraz je možné vyhodnocovat (počítat), čímž se získá *hodnota*.
- Jde o jazyk pro vytváření dotazu na práci s tabulkami.
 - Viz dále co je tabulka z čeho se skládá
- V základu si jej píše Power BI (Excel) v Power Query sám
- Efektivní využití schopností M-kódu vám ušetří
 - spoustu práce
 - času

O jazyku M

- Case-sensitive !
 - Rozlišuje velká malá písmena
- Čistý
 - Přehledný
- „Částečně“ líný
 - Nedělá co nemusí
- Datově typový
 - Sloupce musí mít typ
 - Nesečtu „1“ a 1
- Inspirace jazykem F#

Proč upravovat M - language

- Méně kroků > podstatně rychlejší
- Přehlednější
 - Pro opravy
 - Vylepšování doplňování > Poznámky
 - Hledání problémů
 - Univerzálnost
- Spoustu úprav „**nenakliknete**“ v GUI ☹
 - Hlavně pokročilé věci (?, další parametry funkcí, atd.)

Kde můžete (část) M-kód vidět

- Řádek vzorců

The screenshot shows the Power Query Editor interface. The formula bar at the top displays the formula: `= Table.RemoveColumns(#"Changed Type",{"Kategorie"})`. Below the formula bar is a table with the following data:

ID	Název výrobku	Hmotnost	Cena
1	Lehátko	15	1000
2	Židle	12	520
3	Stůl	35	1500
4	Pohovka	75	5000
5	Skříň	45	2500
6	Police	2	100
7	Knihovna	10	800

On the right side, the 'Applied Steps' pane shows the following steps:

- Source
- Navigation
- Changed Type
- Removed Columns** (highlighted with a blue arrow pointing to the formula bar)

The screenshot shows the Power Query Editor ribbon with the 'Zobrazení' (Display) tab selected. The 'Řádek vzorců' (Formula Bar) option is checked, indicated by a blue arrow. Other options visible include 'Neproporcionální', 'Distribuce sloupce', 'Zobrazit prázdné znaky', 'Profil sloupce', 'Kvalita sloupce', and 'Přejít do sloupce'.

Spuštění M-ka – pokročilý editor

- Home > Advanced Editor
- Domů > Rozšířený editor

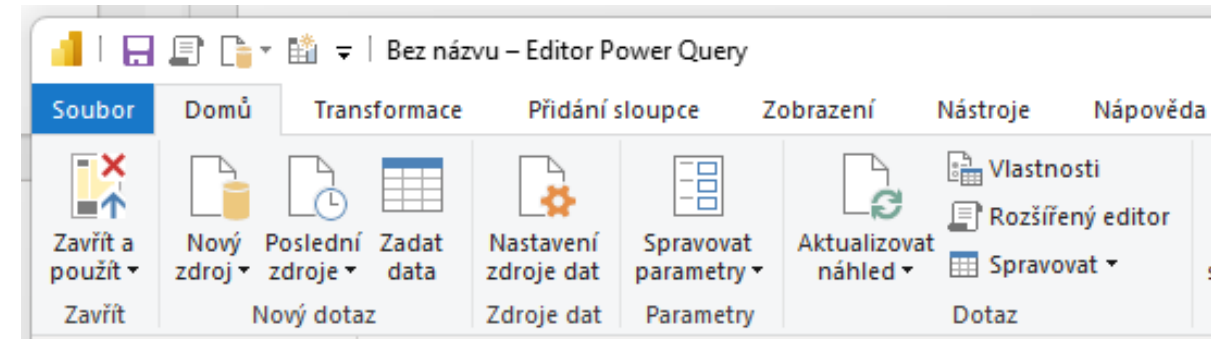
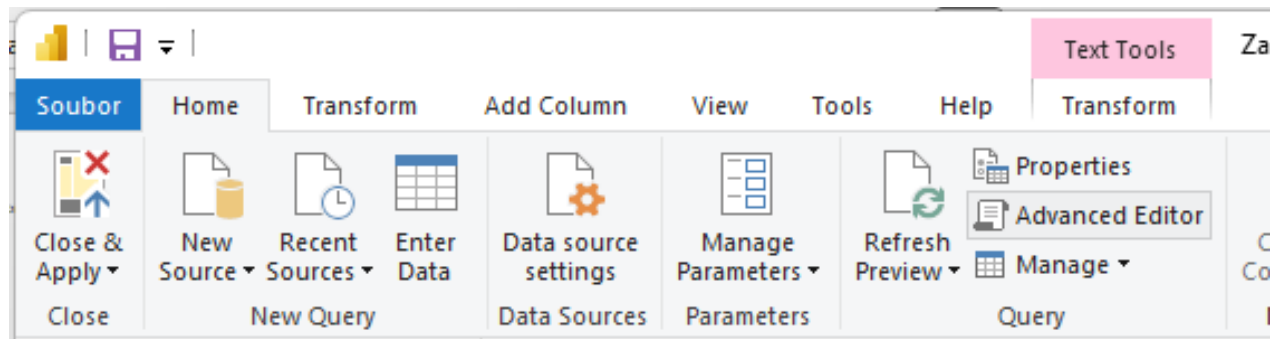
EN

CZ

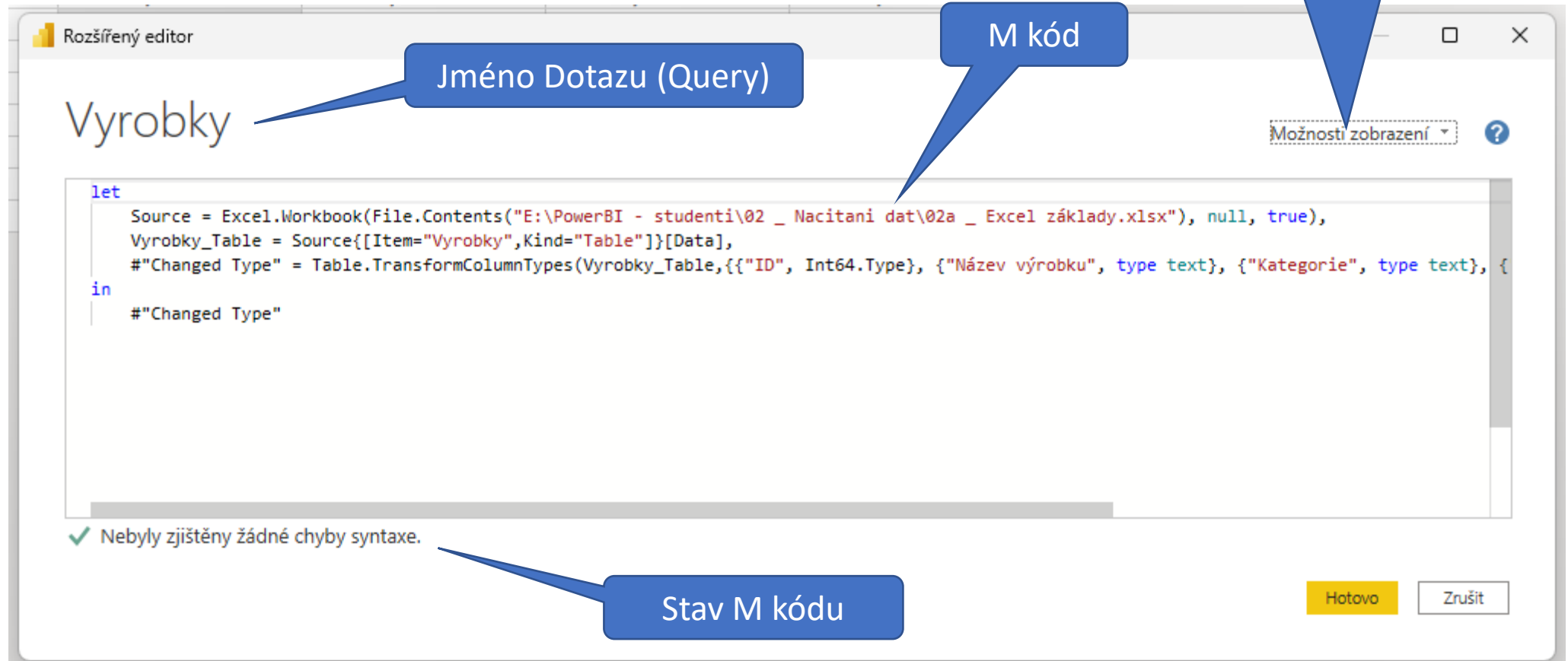
- Tip přemístit na rychlý pás karet

EN

CZ

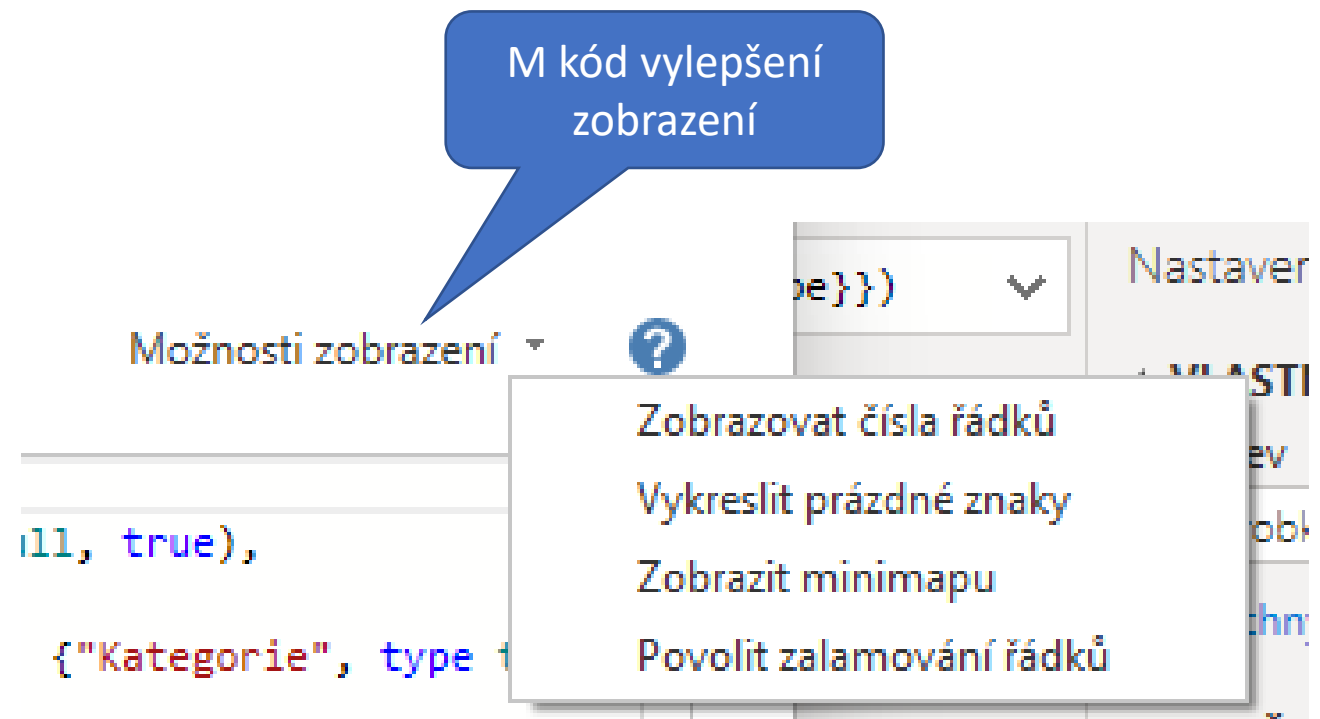


Rozšířený editor



Nastavení rozšířeného editoru

- Zobrazovat čísla řádku
- Prázdné znaky
- „Minimapa“
 - Pro velké a dlouhé kódy



Zpřehlednění M-kódu

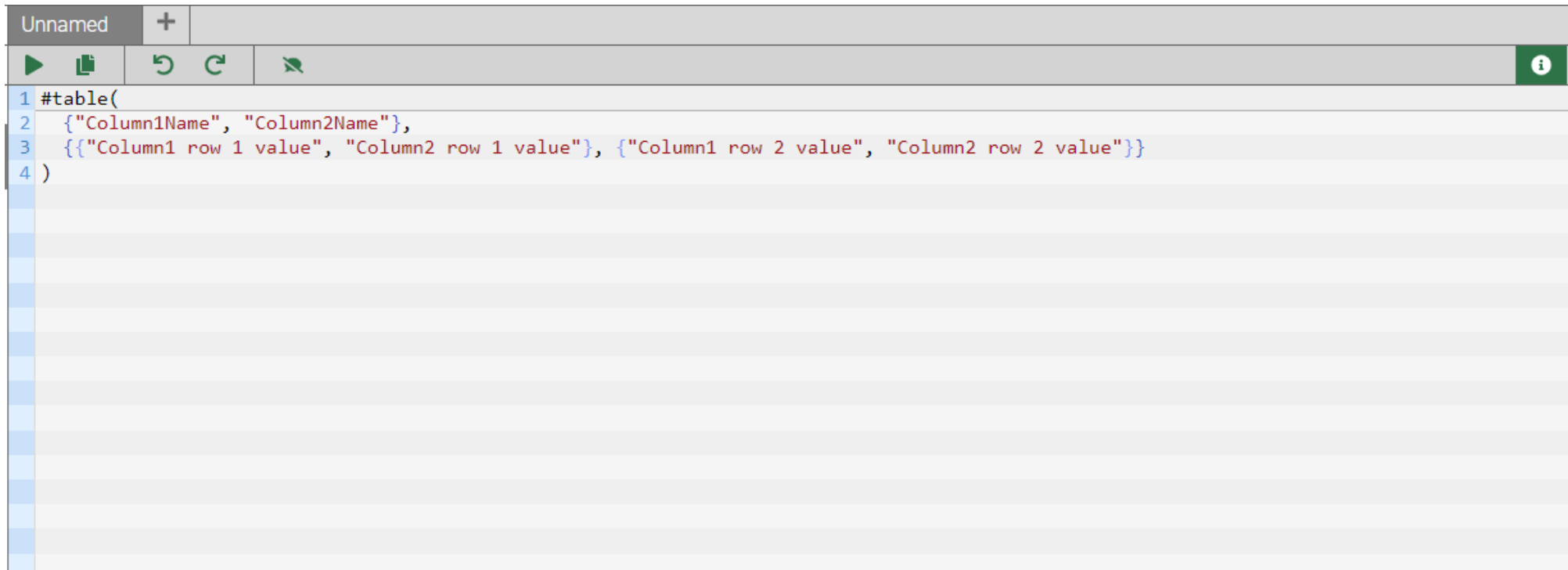
- Pro zpřehlednění kód formátovat
- Podobně jako DAX

```
let
    Source = Excel.Workbook(File.Contents("E:\PowerBI - studenti\02 _ Nacitani dat\02a _ Excel základy.xlsx"), null, true),
    Vyrobykky_Table = Source{[Item="Vyrobykky",Kind="Table"]}[Data],
    #"Changed Type" = Table.TransformColumnTypes(Vyrobykky_Table,{{"ID", Int64.Type}, {"Název výrobku", type text}, {"Kategorie", type text}},
in
    #"Changed Type"
```

- Tip: později podrobněji

Formátování

- <https://www.powerqueryformatter.com/formatter>



The screenshot shows the Power Query Formatter web application. At the top, there is a tab labeled 'Unnamed' with a plus sign to its right. Below the tab is a toolbar with icons for running (a green play button), copying (a green document icon), undo (a green curved arrow), redo (a green curved arrow), and deleting (a green trash can icon). To the right of the toolbar is a green information icon (an 'i' in a circle). The main area is a code editor with a light gray background and alternating light blue and white horizontal lines. It contains the following M code:

```
1 #table(  
2   {"Column1Name", "Column2Name"},  
3   {"Column1 row 1 value", "Column2 row 1 value"}, {"Column1 row 2 value", "Column2 row 2 value"}  
4 )
```

Formátovaný kód

```
1 let
2     Source = Excel.Workbook(
3         File.Contents("E:\PowerBI - studenti\02 _ Nacitani dat\02a _ Excel základy.xlsx"),
4         null,
5         true
6     ),
7     Vyrobky_Table = Source{[Item = "Vyrobky", Kind = "Table"]}[Data],
8     #"Changed Type" = Table.TransformColumnTypes(
9         Vyrobky_Table,
10        {
11            {"ID", Int64.Type},
12            {"Název výrobku", type text},
13            {"Kategorie", type text},
14            {"Hmotnost", Int64.Type},
15            {"Cena", Int64.Type}
16        }
17    )
18 in
19     #"Changed Type"
```

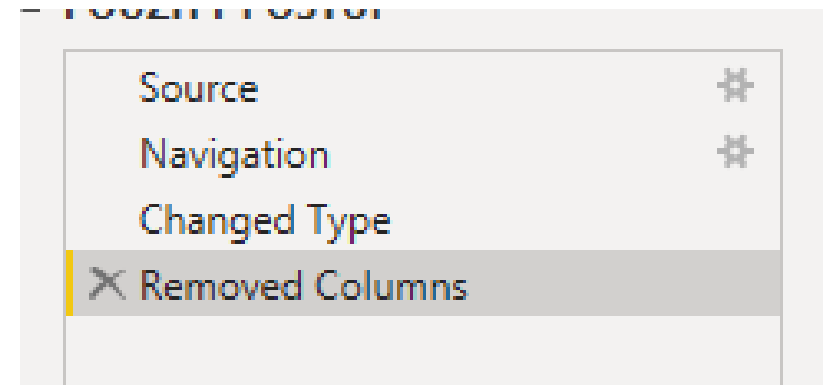
Další vylepšení M-ka při tvorbě v GUI

- Názvy kroků (bez mezer)

- Řádek nový
- **NovyRadek**
- New Row
- **NewRow**

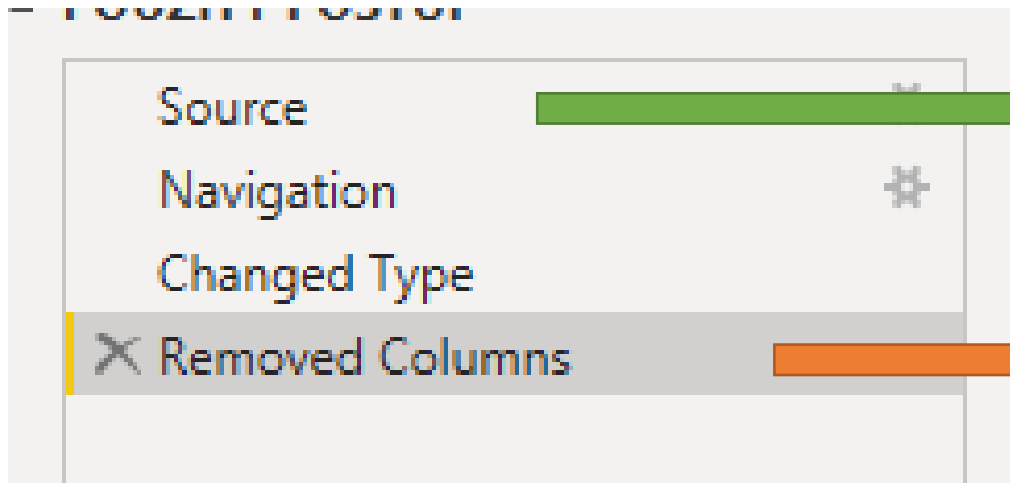
- Tip

- Bez háčku a čárek
- Bez mezer
- Raději Anglicky



Další vylepšení Mka při tvorbě v GUI

- Názvy > Přejmenovat
 - Už umíme



Source

Navigation

Changed Type

✕ Removed Columns

```
Source = Excel.Workbook(File.Conte  
Vyroby_Table = Source{[Item="Vyrol  
#"Changed Type" = Table.TransformC  
#"Removed Columns" = Table.RemoveC
```

Poznámky

A

Základní znaky (operátory) v syntaxi M

- Matematické operátory
- Porovnávací operátory
- Logické operátory
- Speciální znaky

M-ko a znaky - Matematické operátory

- +
- -
- *
- /
- ^

- Tip – (mínus) lze použít také jako unární operátor

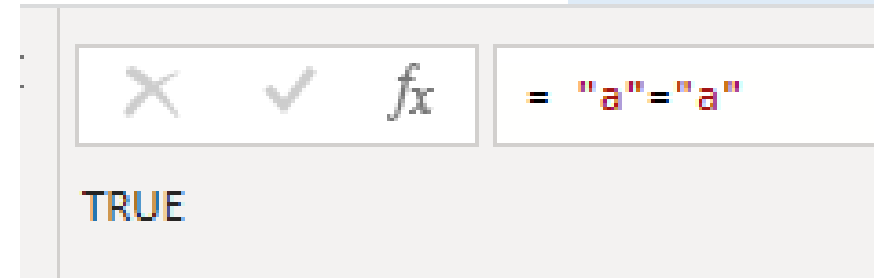
M-ko a znaky - Porovnávací operátory

- >
- <
- >=
- <=
- <>
- =

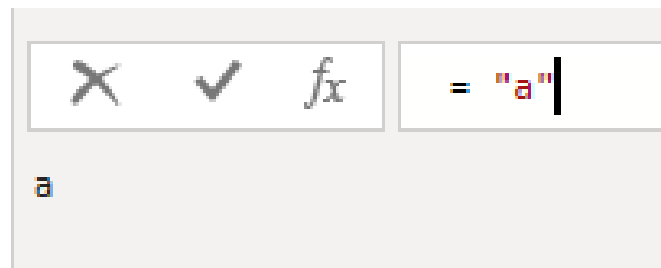
M-ko a znaky - Porovnávací operátory

Výsledek TRUE , FALSE

- $x = y$ se vyhodnotí jako true, pokud x se rovná y
- $x > y$ se vyhodnotí jako true, pokud x je větší než y
- $x < y$ se vyhodnotí jako true, pokud x je menší než y
- $x \leq y$ se vyhodnotí jako true, pokud x je menší nebo rovno y
- $x \geq y$ se vyhodnotí jako true, pokud x je větší nebo rovno y
- $x \neq y$ se vyhodnotí jako true, pokud x není rovno y



- = může jít i o přiřazení



Logické operátory

- Operátory:

- **not** - negace / opak
- **and** - a
- **or** - nebo

- Ukázky

- `[Number] > 8 and [Number] < 25`
- `[Number] < 15 or [Number] > 25`
- `not ([Number] < 25)`

M-ko a znaky

- Speciální operátory

- _ pro každý prvek (viz each)
- .. Rozsah od do (viz List - seznamy)
- # pro názvy spolu s uvozovkami #“A b c“
- ? Pokud nebude nalezeno *null*
- [] Record nebo název sloupce záznamu (viz Record – Záznam)
- { } List nebo číslo řádku, pořadí prvku (viz List - seznamy)
- " správné horní uvozovky – (palec „inch“)
- & spojení

- Bude vysvětleno dále

Další operátory

- **is** Výraz x je y vrátí hodnotu true, pokud je typ x kompatibilní s y, a vrátí hodnotu false, pokud typ x není kompatibilní s y.
- **as** Výraz x jako y určuje, že hodnota x je kompatibilní s y podle operátoru.
- Viz_dále

is, as - příklady

- `1 as number // 1 viz dále`
 - `"A" as number // error`
 - `null as nullable number // null`
-
- `= null is null // true`
 - `= null is number // false`

Poznámky

B

Sekce co může M-kód obsahovat

```
1  let
2      Konstanta = 10, // poznámka
3      List = {1,"Pavel", true, Tabulka, {1,2,3}, (x)=>x*x},
4      Record = [Jméno = "Pavel", Funkce = "MVP", Město="Brno"],
5      tabulka = #table({"Pismena", "Čísla"}, {{ "A", 1}, {"B", 2}, {"C", 3}}),
6      /*
7      Víceřadková poznámka
8      */
9      Source = "Demo ukázka sekce"
10 in
11     Source
```

„Sekce“ kroky v M-ku

- Hodnoty (Values)
 - Konstanty „jednoduché“
 - Strukturované (list, record, tabulka)
- Dotazy / výrazy (Expressions)
 - Funkce - někdy samostatná sekce
- Poznámky

Sekce co může M-kód obsahovat

let

```
Konstanta = 10, // poznámka
Tabulka = #table({"Pism", "Čís"}, {{ "A", 1}, {"B", 2}}),
List = {1, "Pavel", true, Tabulka, {1,2,3}, (x)=>x*x},
Record = [Jméno = "Pavel", Funkce = "MVP", Město="Brno"],
/*
Víceřádková poznámka
*/
Source = "Jmeno ukázka sekce"
```

Konstanta

Tabulka

Dotaz

List (Seznam)

Záznam (Record)

Poznámka

in

Source

Hodnoty (Values)

- **Hodnota** je kus datové informace nebo-li:
- Druhy hodnot:
 - Hodnoty jednoduché:
 - Hodnoty strukturované:

Hodnoty (Values)

- **Hodnoty jednoduché:**

- čísla
- text
- logická hodnota
- binární
- datu
- čas
- ...

Hodnoty (Values)

- **Hodnoty strukturované:**

- **List – seznam** samostatná sekce
- **Record – záznam** samostatná sekce
- **Tabulka – Table** samostatná sekce

- seznamy seznamů atd.
- seznamy tabulek
- ...

Dotazy / Výrazy (Expressions)

- **Dotaz** je něco, co se dá vyhodnotit.
- Dotazem se rozumí **datový podklad se všemi úpravami**, které provádíme nad datovým zdrojem pomocí dotazů v jazyce **M**
- Pokud vyhodnocuji mohu vrátit zpět nějakou "hodnotu".
 - Z dotazu $1 + 1$ tento výraz se vyhodnotí a získáte hodnotu 2.

Dotazy / Výrazy (Expressions)

```
Soucet = 1 + 1
```

```
Tabulka = #table({"Písm", "Čís"}, {{ "A", 1},  
{"B", 2})
```

- Each...
- If .. then
- ..

Funkce

- Někdo řadí samostatně
- Viz samostatná sekce
- Tip Google

=#shared

- Nový dotaz

• Tip
Google

  	= #shared
Vyrobky	Table
Dotaz1	Record
Value.ResourceExpression	Function
Resource.Access	Function
appFigures.Tables	Function
appFigures.Content	Function
MicrosoftAzureConsumptionInsights.Conten	Function
MicrosoftAzureConsumptionInsights.Test	Function
MicrosoftAzureConsumptionInsights.Tables	Function
AzureDevOpsServer.AccountContents	Function
AzureDevOpsServer.Feed	Function

Poznámky

- Pořádek v kódech
 - Nejsou nezbytné pro funkci kódu,
 - Vhodné pro lepší přehled a orientaci

// komentář jednořádkový

**/ pro
víceřádkový
komentář
/

Prostor pro poznámky

C

Hodnoty – Konstanty

- Prvkem pro všech ostatních hodnot
 - Záznam, seznam se skládá ze samostatných hodnot

Konstanta

Konstanta = 10, // poznámka

Tabulka

Tabulka = #table({"Písm", "Čís"}, {{ "A", 1}, {"B", 2}},

Samostatné hodnoty

- Typy
 - **125.44** - číslo
 - **"JakNaExcel"** - text
 - **true** - logická hodnota
 - (pozor na velikost písmen)
 - **null** - skutečně prázdná hodnota (*null*)
- Prázdné (pozor *null* a nula – zero a prázdná „“)
- Mezera
- atd.

Samostatné hodnoty vytvořené funkcí

- Pomocí funkcí můžete sestavit časové a datumové údaje:
 - **#time (hodiny, minuty, sekundy)**
 - - vytvoření času - *#time (10, 42, 20)*
 - **#date (roky, měsíce, dny)**
 - - vytvoření datum - *#date (2018, 12, 31)*
 - **#datetime (roky, měsíce, dny, hodiny, minuty, sekundy)** - - ...
 - **#datetimezone (roky, měsíce, dny, hodiny, minuty, sekundy, offset-hodiny, offset-minuty)** - - ...
 - **#duration (dny, hodiny, minuty, sekundy)** - - ...

Speciální konstanty

- ∞ kladné nekonečno = **1/0**
- $-\infty$ záporné nekonečno = **-1/0**
- **NaN** není číslo = **0/0**
 - Not a number



- V tabulce Power Query
 - -Infinity
 - NaN
 - Infinity

	ABC 123 Column1	ABC 123 Custom
1	-1	-Infinity
2	0	NaN
3	1	Infinity

Poznámky

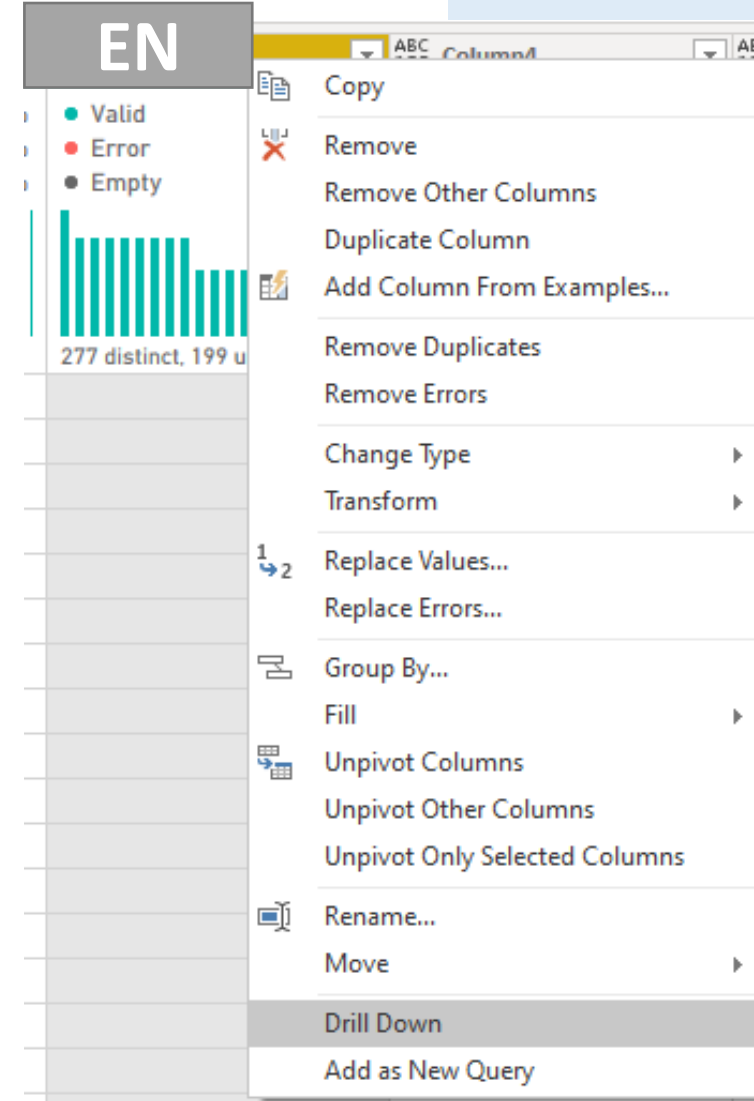
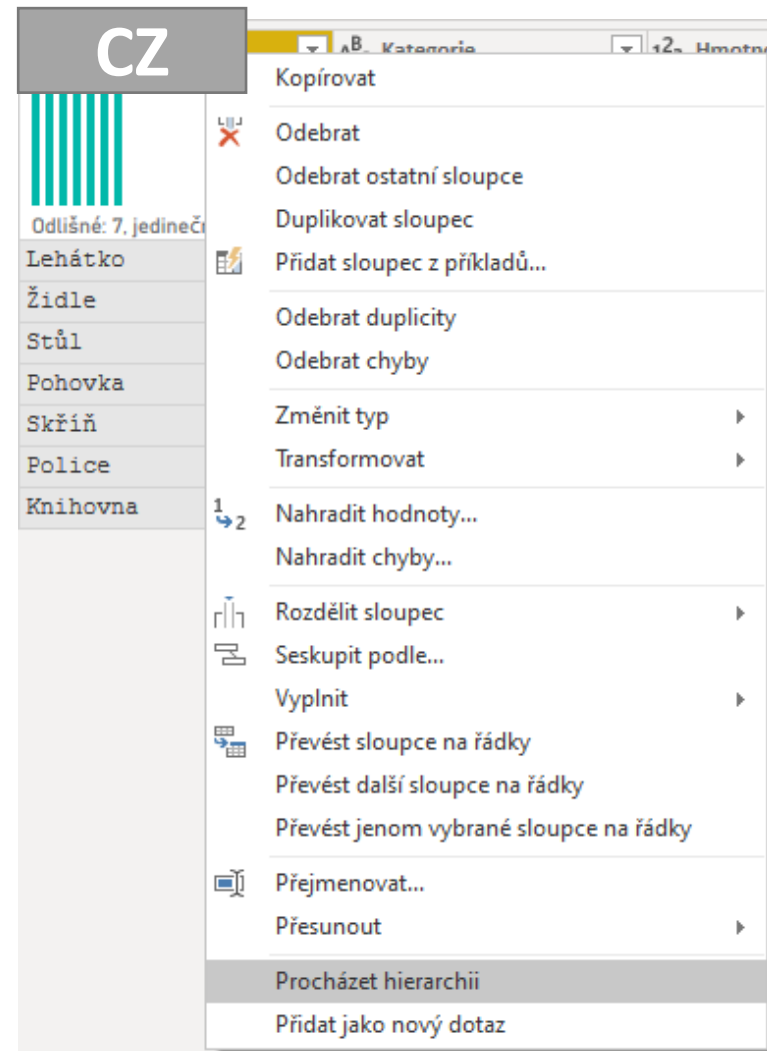
D

List - Seznam - Vektor

- Řada hodnot
 - Primitivní
 - Strukturované
- Není limitován počtem hodnot
 - Limit je u funkcí (pokud budou list zpracovávat)
 - Limit HW
- Bez vektoru se neobejdete
- Vektor může obsahovat vektor, tabulku atd...
- Tip: Sloupec jako seznam

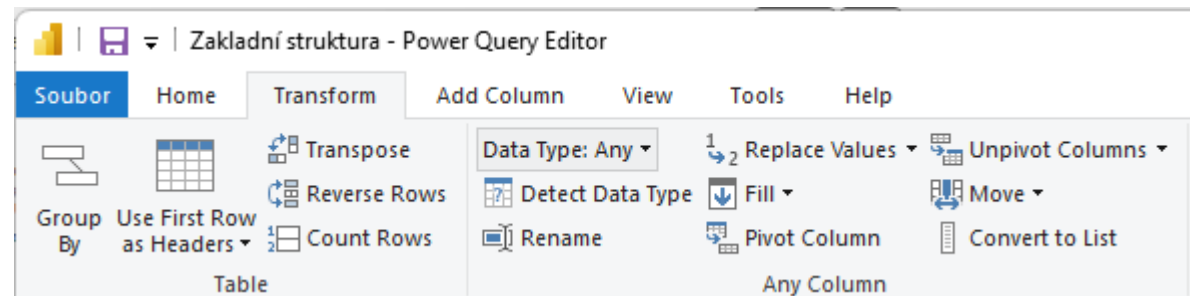
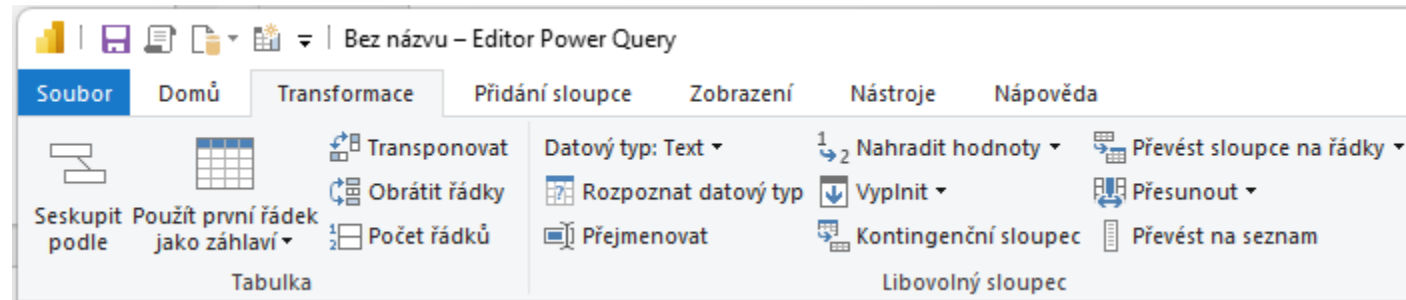
List - Seznam - Vytvoření

- 1) V PQ
 - Drill Down
 - Procházet hierarchií v CZ



List - Seznam - Vytvoření

- (1) V PQ druhá možnost
 - Převést na seznam
 - Convert to List



List - Seznam - Vytvoření

- 2) List vytvořit pomocí funkce
 - List.Dates(),
 - List.Numbers()
 - List.Random()
 - Text.ToList()
 - TableToList()
 - Table.ColumnName()
 - ..

List - Seznam – Vytvoření

(3) List lze vytvořit „Ručně“

$=\{1, 2\}$

$=\{1..2\}$

$=\{\{1, 2\}, \{3, 4, 5\}\}$

$=\{"a".."f"\}$

...

List - Seznam – Vytvoření

„Ručně“

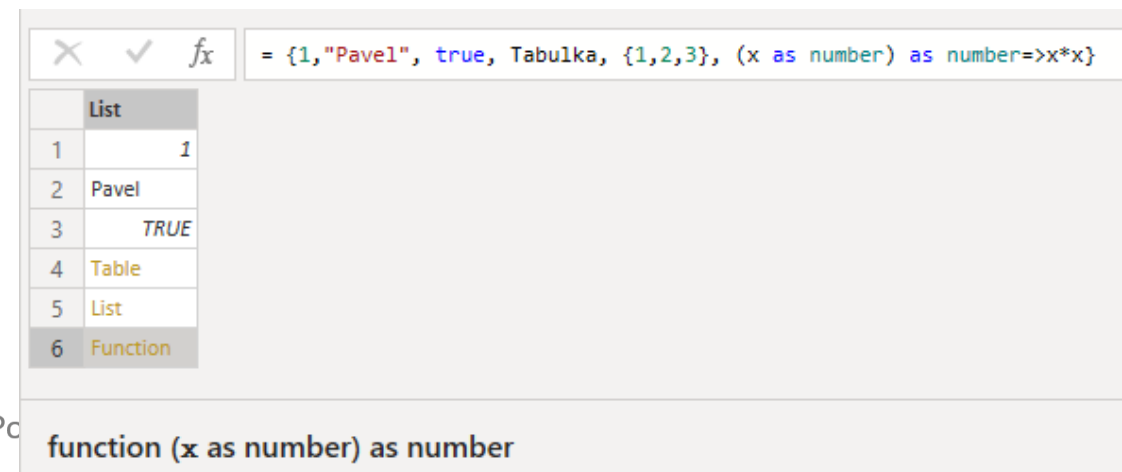
= {1, "Pavel", true, **Tabulka**, {1, 2, 3}, (x) => x*x}

- Může obsahovat funkci, odkaz na jiný dotaz Tabulka atd

- Nebo lépe

• = {1, "Pavel", true, Tabulka, {1, 2, 3}, (x as number) as number=>x*x}

- Definováno ve funkci (viz typ)



Kde lze využít?

$= \{ 1 \dots 12 \}$

- Měsíce v roce?

$= \{ 1 \dots 7 \}$

$= \{ 1 \dots 5 \}$

- Dny v týdnu (Po - Pá)

$= \{ "po", \dots \}$

Klasický seznam výběr prvku

- Vybere první prvek
 - Nula první
 - Jedna druhý
 - Programátorský počet
 - Viz indexace od 0

- První prvek

= {1, 2, 3}{0}

Klasický seznam výběr prvku

- V Módu před odkaz na dotaz

let

Seznam = {1, 2, 3},

Source = **Seznam**{1}

in

Source

Klasický seznam - odkazování

= List_vycet_prvku{1}

- Získám druhý prvek ze seznamu **List_vycet_prvku**

= List_vycet_prvku{7}

- Chci osmý prvek, ale není v seznamu **List_vycet_prvku**

- **Chyba!**

= List_vycet_prvku{7} ?

- Chci osmý prvek, není vrátí null protože není > neskončí chybou

= List_vycet_prvku{-7} ?

- Chyba -7 prvek nemůže existovat, 7 může

Otazník je
skvělá věc !!!

Klasický seznam – výběr prvku

- První prvek ze seznamu

= **List.First**({1, 2, 3})

- Poslední prvek ze seznamu

= **List.Last**({1, 2, 3})

- První dva prvky ze seznamu

= **List.FirstN**({1, 2, 3},2)

Poslední využitím funkce

- Poslední dva prvky ze seznamu

```
= List.LastN({1, 2, 3}, 2)
```

- Poslední větší rovno 2 než

```
= List.LastN({1, 2, 3, 1, 2, 3}, each _ >= 2)  
// 2, 3
```

List - Seznam – Hrátky...

```
MyText = "5;6;9",  
MyList = Text.Split(MyText, ";"),  
Myitem=MyList{2}
```

List - Seznam - Poznámky

- Odkaz_web

Záznam Rekord

- Strukturovaný datový typ
- Skupina pojmenovaných hodnot
 - Název hodnoty, Hodnota
- Záznam je v každém PQ
 - Let .. in... je vlastně záznam – viz dále

Vytvoření Recort - záznam

- (1) V Power Query Editoru
Jen zaobalí do Let In

Viz dále

```
1  let
2      Source = #table({"Pismena", "Čísla"}, {"A", 1}, {"B", 2}, {"C", 3})
3  in
4      Source
```

Vytvoření Recort - záznam

- (2) Využiji funkce

= `DateTime.LocalNow()`

– vrátí aktuální čas

- Převedení času na zaznam

= `DateTime.ToRecord (DateTime.LocalNow())`

Záznam (Record) ze Seznamu (List)

(2) Využijí funkce

```
=Record.FromList({1, "Pavel", "MVP"}, type  
[ID = number, Jmeno = text, Úspěch =  
text])
```

```
1 let  
2   Source = Record.FromList({1, "Pavel", "MVP"}, type [ID = number, Jmeno = text, Úspěch = text])  
3 in  
4   Source
```

Viz dále ruční vytvoření

(3) Ruční zadání - Záznam Rekord

- Definuje pomocí hranatých závorek []
 - Jméno které toto pole identifikuje
 - Znaménko rovná se.
 - Hodnota tohoto pole.
- = [Jméno = "Pavel"]

Identifikátor

Hodnota

The screenshot shows a user interface for entering a record. At the top, there is a formula bar with a close button (X), a checkmark, and a function button (fx). The formula bar contains the text `= [Jméno = "Pavel"]`. Below the formula bar is a table with two columns. The first column is labeled **Jméno** and the second column contains the value `Pavel`.

Jméno	
Pavel	

(3) Record Ručně

```
= [Jméno = "Pavel", Funkce = "MVP",  
Město="Brno"]
```

✕	✓	<i>fx</i>	= [Jméno = "Pavel", Funkce = "MVP", Město="Brno"]
Jméno	Pavel		
Funkce	MVP		
Město	Brno		

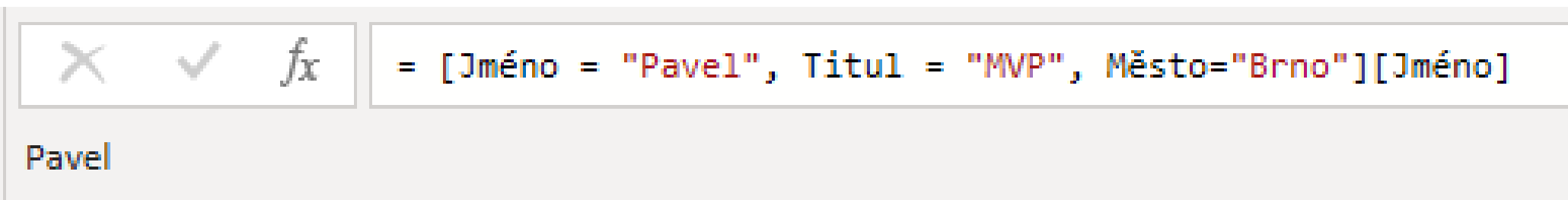
```
= [Osoba = [Jméno = "Jan", Věk = 38], Povolani =  
"Důchod"]
```

✕	✓	<i>fx</i>	= [Osoba = [Jméno = "Jan", Věk = 38], Povolani = "Důchod"]
Osoba	Record		
Povolani	Důchod		

Práce se záznamy-vyběr záznamu

- Výběr prvku
 - Má pojmenování
 - Potřebuji jméno

= [Jméno = "Pavel", Titul = "MVP", Město="Brno"] [Jméno]

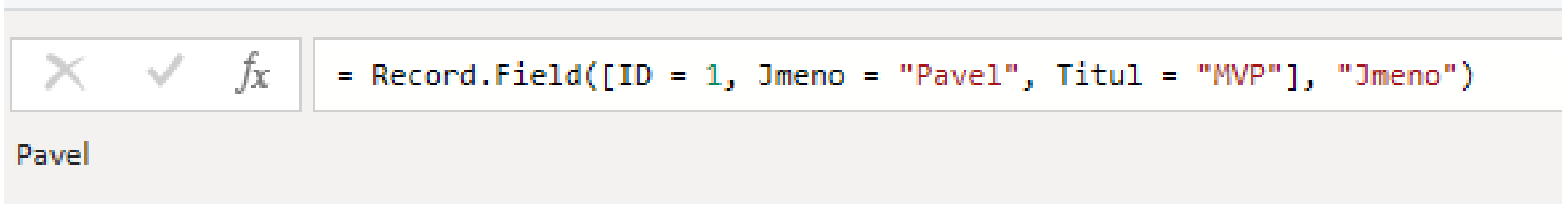


The screenshot shows the Excel formula bar. On the left, there are three icons: a red 'X' for cancel, a green checkmark for confirm, and a function icon 'fx'. The formula bar contains the text: `= [Jméno = "Pavel", Titul = "MVP", Město="Brno"] [Jméno]`. Below the formula bar, the word "Pavel" is displayed, indicating the result of the filter operation.

Poznámky – práce se záznamy

- Využitím funkce

```
= Record.Field([ID = 1, Jmeno = "Pavel", Titul = "MVP"], "Jmeno")
```



Není lepší?

```
= [Jméno = "Pavel", Titul = "MVP",  
Město="Brno"] [Jméno]
```

Z Record (Záznam) List (Seznam)

```
= Record.FieldValues([Jméno = "Pavel",  
Titul = "MVP", Město="Brno"])
```

✕	✓	<i>fx</i>	= Record.FieldValues([Jméno = "Pavel", Titul = "MVP", Město="Brno"])
	List		
1	Pavel		
2	MVP		
3	Brno		

- opačně Record.FromList

Nepřipomíná vám něco?

let

```
Source = [Jméno = "Pavel", Funkce = "MVP",  
Město="Brno"]
```

in

Source

Co záměna **let in** znaky **[]**?

Nepřipomíná vám něco?

let

```
    Source = [Jméno = "Pavel", Funkce = "MVP",  
Město="Brno"]
```

in

```
    Source
```

[

```
    Source = [Jméno = "Pavel", Funkce = "MVP",  
Město="Brno"]
```

]

```
    [Source]
```

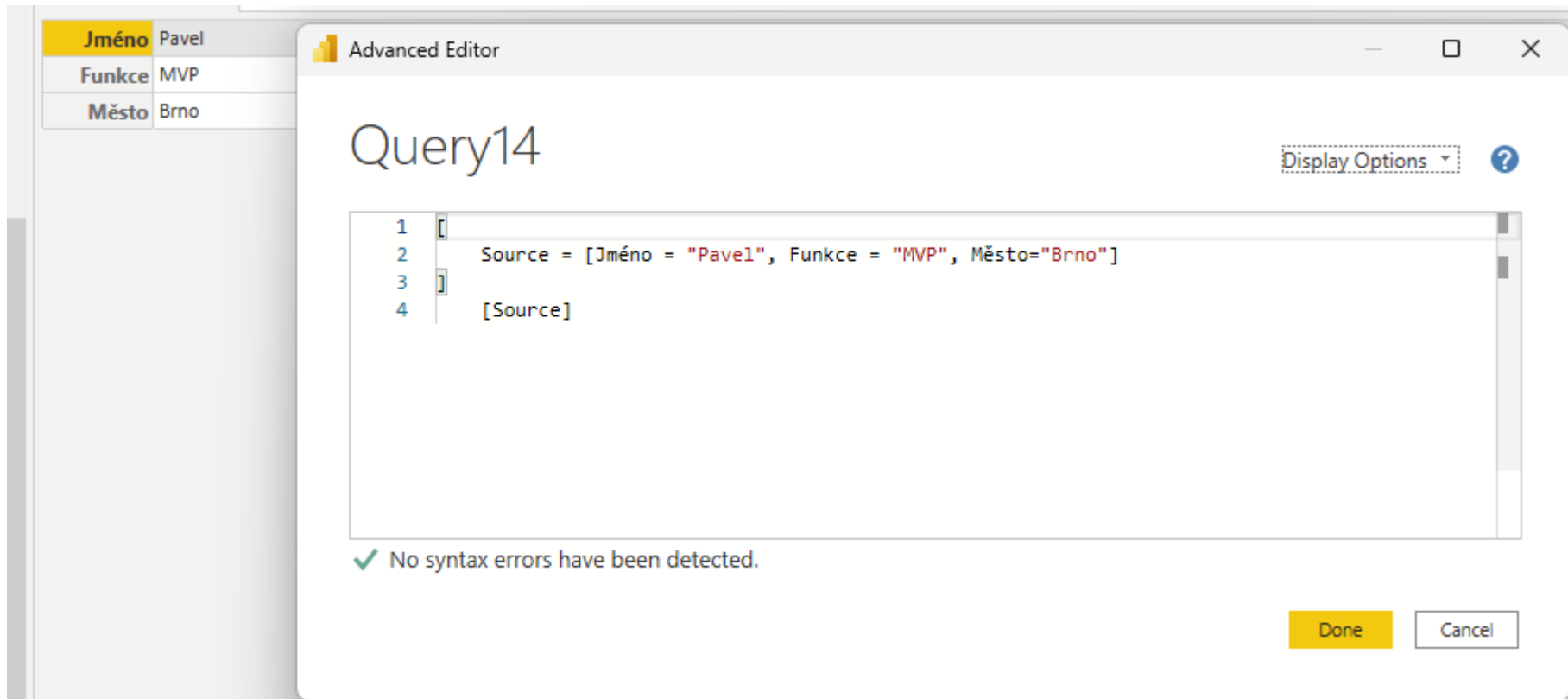
Jméno	Pavel
Funkce	MVP
Město	Brno

```
1  [  
2    Source = [Jméno = "Pavel", Funkce = "MVP", Město="Brno"]  
3  ]  
4  [Source]
```

Nepřipomíná vám něco?

Nebo lépe

```
[Jméno = "Pavel", Funkce = "MVP", Město="Brno"]
```



Prostor pro poznámky

Tabulky (Table)

- Uspořádaná posloupnost řádků, kde každý řádek je seznam
 - (viz popis seznamu výše).
- Uspořádání hodnot do řádků a sloupců
 - Má definované datové typy
 - Platí nad celým sloupcem
 - PQ pracuje a celým sloupcem
 - Nezáleží na pořadí sloupců
- Tabulky jsou nutnost!

Vytvoření tabulky

- (1) V PQE
- (2) Funkci
 - Table.FromRows
 - Table.FromList()
- (3) Ručně
 - #table

(1) V PQE

- Využitím PQ

Table.TransformColumnTypes(Tabulka2_Table,{{"Organizace", type text}},

	A ^B _C Organizace	A ^B _C Typ_organizace	1 ² ₃ Bilancni_suma	1 ² ₃ Stala_aktiva_br
1	Základní škola a mateřská ško...	ZŠ_MŠ	3146	
2	Mateřská škola Bačetín	ZŠ_MŠ	1415	
3	Mateřská škola Bartošovice v...	ZŠ_MŠ	1403	
4	Základní škola a mateřská ško...	ZŠ_MŠ	2770	
5	Základní škola a mateřská ško...	ZŠ_MŠ	1374	
6	Mateřská škola Borohrádek	ZŠ_MŠ	2525	
7	Základní škola T. G. Masaryka...	ZŠ_MŠ	11748	
8	Základní škola a mateřská ško...	ZŠ_MŠ	14487	
9	Masarykova základní škola a...	ZŠ_MŠ	11045	
10	Základní škola a mateřská ško...	ZŠ_MŠ	3495	

Query Settings

PROPERTIES

Name
Tabulka2

[All Properties](#)

APPLIED STEPS

- Source
- Navigation
- ✕ Changed Type

(2) Funkci vytvoření z List

```
= Table.FromList ({"a", "b", "c",  
"d"}, null, {"Písmena"})
```

(2) Funkci vytvoření z Record

```
= Table.FromRecords ({ [ID = 1, Jmeno =  
"Pavel", Tit = "MVP"], [ID = 2, Jmeno  
= "Eva", Tit = "Ing"], [ID = 3, Jmeno  
= "Iva", Tit = null] })
```

(2) Funkci načtení tabulky z Excel

- Source =
Excel.Workbook(File.Contents("D:\dotaznik.xlsx"), null, true){0}[Data]

(2) Funkci další možnost

- `Table.FromRows(Json.Document(Binary.Decompress(Binary.FromText("i45WCkgsS81R0lHySSw+vDBbKVYnWsm1LBEo4JuYc3ghmO8J5oe15mSDBGIB", BinaryEncoding.Base64), Compression.Deflate)), let _t = ((type nullable text) meta [Serialized.Text = true]) in type table [Jméno = _t, Příjmení = _t]),`

```
#table
```

Jméno funkce

```
(
```

Definice datové typy

```
type table
```

```
[
```

Názvy sloupců

```
    Jméno = text,
```

datové typy sloupců

```
    Výška = number
```

```
],
```

```
{
```

Data pro první řádek

```
    {"Pavel", 183},
```

Data pro druhý řádek

```
    {"Eva", 182}
```

```
}
```

```
)
```

(3) Ručně

- Tabulka se sestaví pomocí funkce `#table`
- první seznam názvy sloupců
`{"NázevSloupec", „S12“}`
- jednotlivé řádky se zapisují opět jako seznam
`{"A", 1}`
- s skupina záznamu je opět ve složených závorkách.

(3) Ručně – názvy sloupců

```
= #table({"Písmena", "Čísla"}, {"A", 1}, {"B", 2}, {"C", 3}))
```

	Písmena	Čísla
1	A	1
2	B	2
3	C	3

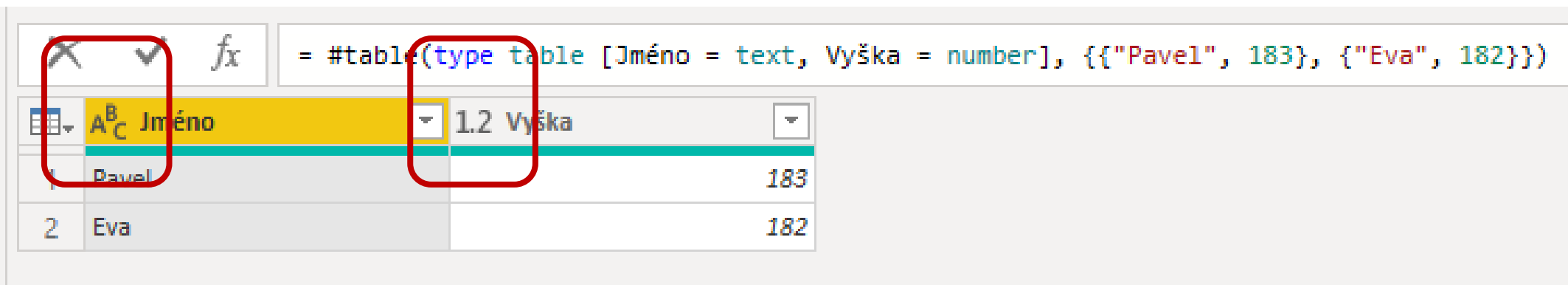
(3) ručně-bez názvů sloupců

```
= #table( 2, {{"A", 1}, {"B", 2}, {"C", 3}})
```

	Column1	Column2
1	A	1
2	B	2
3	C	3

(3) Ručně – názvy sloupců a typ hodnoty

```
= #table(type table [Jméno = text,  
Vyška = number], {"Pavel", 183},  
{"Eva", 182}))
```



= #table(type table [Jméno = text, Vyška = number], {"Pavel", 183}, {"Eva", 182}))	
Jméno	Vyška
Pavel	183
Eva	182

Práce s tabulkou

- Co potřebujete
 - Celý sloupec
 - Celý řádek
 - Jednu „buňku“
 - – průsečík řádek sloupec
- Asi už tušíte [], {}

Práce s tabulkou - sloupec

```
= #table(type table [Jméno = text, Vyška =  
number], {"Pavel", 183}, {"Eva", 182}) [Jméno]
```

✕	✓	<i>fx</i>	= #table(type table [Jméno = text, Vyška = number], {"Pavel", 183}, {"Eva", 182}) [Jméno] ▼
		List	
1		Pavel	
2		Eva	

Práce s tabulkou - sloupec

- Případně přehledněji

```
let
```

```
    Source = #table(type table [Jméno = text, Vyška =  
number], {{ "Pavel", 183}, {"Eva", 182}}),
```

```
    Sloupec = Source[Jméno]
```

```
in
```

```
    Sloupec
```

Práce s tabulkou - řádek

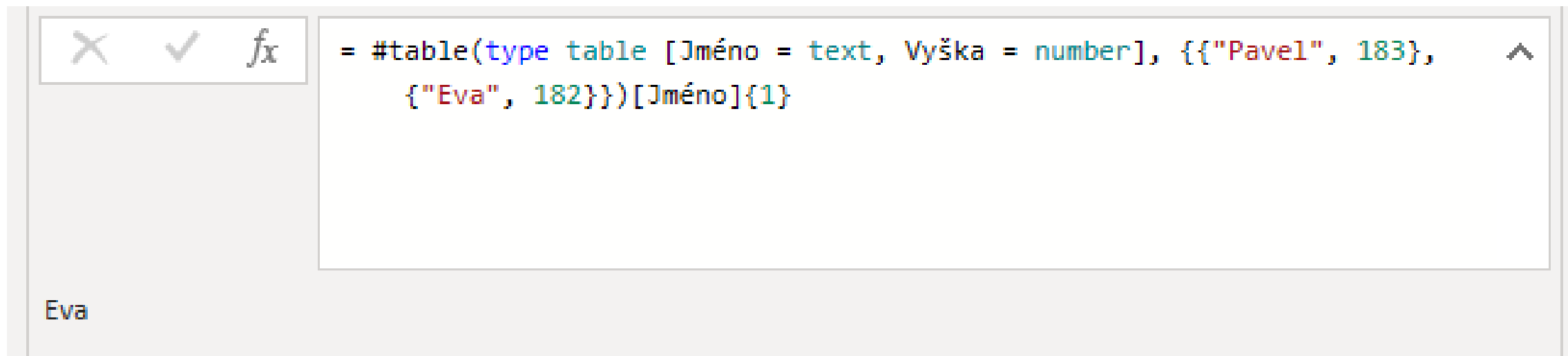
- Pokud chci druhý řádek (první je nula)

```
= #table(type table [Jméno = text, Vyška =  
number], {"Pavel", 183}, {"Eva", 182})){1}
```

✕	✓	<i>fx</i>	= #table(type table [Jméno = text, Vyška = number], {"Pavel", 183}, {"Eva", 182})){1}	▼
Jméno	Eva			
Vyška	182			

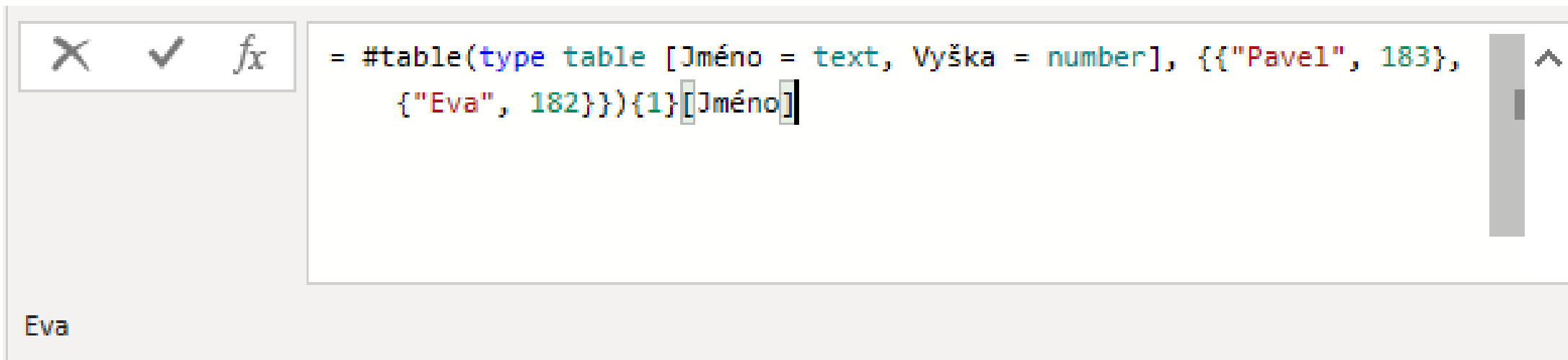
Práce s tabulkou - buňka

```
= #table(type table [Jméno = text, Vyška =  
number], {"Pavel", 183}, {"Eva",  
182})) [Jméno]{1}
```



Práce s tabulkou - buňka

```
= #table(type table [Jméno = text, Vyška =  
number], {{"Pavel", 183}, {"Eva", 182}}){1}  
[Jméno]
```



Práce s tabulkou - buňka

- Nezáleží zda vyberete:
 - Řádek anásleď sloupce
 - Sloupec anásleď řádek
- Což je logické

Práce s tabulkou – buňka-otazník?

- Nezapomenout na otazník
 - Co když neexistuje název sloupce
 - Co když neexistuje název řádku

MojeTabulka [Sloupec] ? { 0 } ?

Tabulky - poznámky

Expressions

J

- Dotaz (výrazy, vzorec, expressions, krok)
 - Cokoli, co lze vyhodnotit a vrátit „hodnotu“
 - List, Record, Tabulku
 - z A získat B
 - Mnohdy využijete X kroků
 - Vysledek = Operace(Neco)
 - Ono i hodnota je vlastně dotaz ;)
 - DPH = 21
 - Funkce, operace, výpočty

Matematické

- Součet

$$= 1 + 1$$

$$//2$$

- Nebo v M-ku

$$C=1$$

$$B=2$$

$$A=B+C$$

Porovnávací

- Porovnání

= 3 > 2 //true

- Použití v logických funkcích

Spojovací

```
= "Hello " & "World"
```

```
AhojSvete = "Hello " & "World"
```

Funkce

```
= Text.Upper("Hello World")
```

- Viz samostatná kapitola

```
VelkaPismena = Text.Upper("Hello World")  
// "HELLO WORLD"
```

- Viz dále

Poznámky

Let In - konstrukce

- Víme sekce co M-kód obsahuje
- Zobrazení konstrukce Let In
 - Tušíme co znamená

```
let
    x = 1,
    Datum = #date(2022,5,22),
    Funkce = Date.Year(Datum),
    // Text poznámky
    #"Proměna mezera" = {x,Datum, Funkce} ,
    /* víceřadková
       poznámka */
    vysledek = #"Proměna mezera"
in
    vysledek
```

Let in

- Let in
- Je vlastně záznam ;)
- jen [] se použije konstruce **let in**
- Aby kód nevypadal tak strašidelně ;)

Let in - náhrada Let in >> []

let

```
Source = [Jméno = "Pavel", Funkce = "MVP", Město="Brno"]
```

in

Source

- náhrada

```
[Source = [Jméno = "Pavel", Funkce = "MVP", Město="Brno"]]
```

- náhrada

```
[Jméno = "Pavel", Funkce = "MVP", Město="Brno"]
```

Let In - konstrukce

let	Sekce let
x = 1,	Hodnota do proměnné
Datum = #date(2022, 5, 22),	, oddělovač konců řádku
Funkce = Date.Year(Datum),	Použití funkce
// Text poznámky	Poznámka jednořádková
#"Promena mezera" = {x, Datum, Funkce},	Název kroku s mezerou proto # a název v " a tvorba listu
/* víceřádková poznámka */	Poznámka víceřádková
vysledek = #"Promena mezera"	Poslední řádek bez oddělovače řádku
in	Sekce in
vysledek	Výstup

Datum	=	#date(2022, 5, 22)
Název kroku	Přiřazení	Definice přiřazení

```
1 [Source = [Jméno = "Pavel", Funkce = "MVP", Město="Brno"]]
```

Blank Query – Prázdný dotaz

```
let  
    Source = ""  
in  
    Source
```

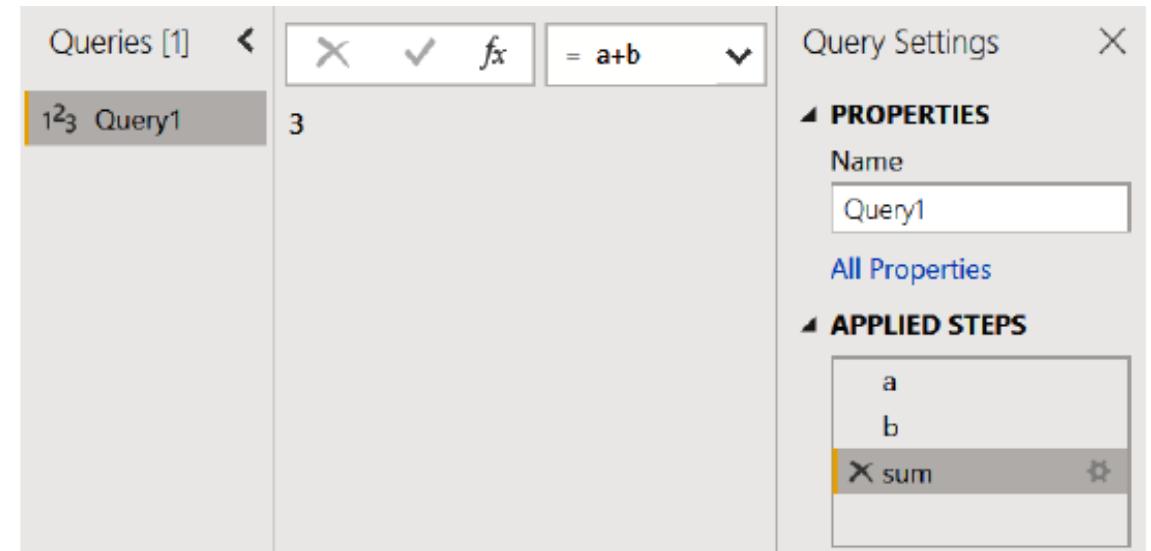
M-ko vs Editor Power Query

- // poznámky

Query1 Display Options ?

```
1 let
2   → a=1,
3   → b=2,
4   → sum=a+b
5 in
6   sum
```

✓ No syntax errors have been detected.



Pořadí kroků

let

A = 1,

B = 2,

C = A + B

in

C

let

C = A + B,

A = 1,

B = 2

in

C

Názvy kroků

let

#"A a" = 1,

#„B b" = 2,

#„C c" = #"A a" + #„B b"

in

#„C c"

Nepotřebné kroky

```
let
```

```
    A = 1,
```

```
    B = 2,
```

```
    X = 1111 + 5555,
```

```
    C = A + B
```

```
in
```

```
    C
```

Poznámky

```
let
    A = 1,
    B = 2,
    // Toto je poznámka
    C = A + B
in
    C
```

Poznámka k let-in

Each – pro každý



- Slouží ke snadnému vytváření funkcí
- Proved' pro každý záznam („řádek“)
- Each "každý..." je „syntaktický cukr“
- nahrazuje znak (_)
 - "(_) => ... "
- Each je užitečné v kombinaci s operátorem pro vyhledávání, který se standardně používá s _.
 - Each [CustomerID]
 - _[CustomerID],
 - (_) => _[CustomerID]

Each – vynásob dvěma

Ukázka 1

let

```
Zdroj = #table({"Numbers"}, {{1}, {2}, {3}, {4}, {5}}),  
NovySloupec = Table.AddColumn(Zdroj, "Double", each 2*[Numbers])
```

in

```
NovySloupec
```

Ukázka 2

let

```
Source = #table({"Numbers"}, {{1}, {2}, {3}, {4}, {5}}),  
NovySloupec = Table.AddColumn(Source, "Double", (_) => 2*__[Numbers])
```

in

```
NovySloupec
```

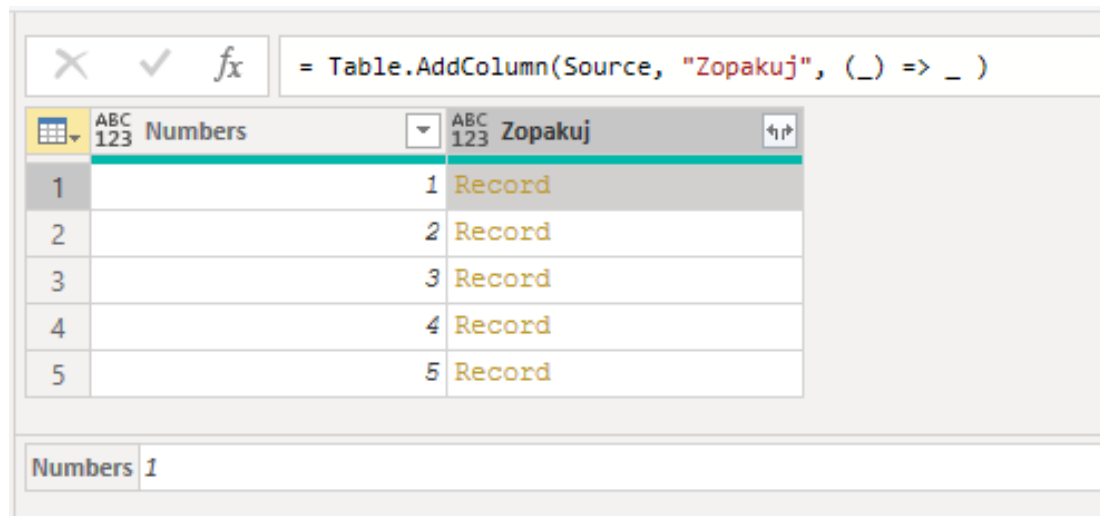
Each - zopakuj

let

```
Source = #table({"Numbers"}, {{1}, {2}, {3}, {4}, {5}}),  
NovySloupec = Table.AddColumn(Source, "Zopakuj", (_) => _ )
```

in

NovySloupec



ABC 123	Numbers	ABC 123	Zopakuj
1		1	Record
2		2	Record
3		3	Record
4		4	Record
5		5	Record

Numbers 1

Each - zopakuj

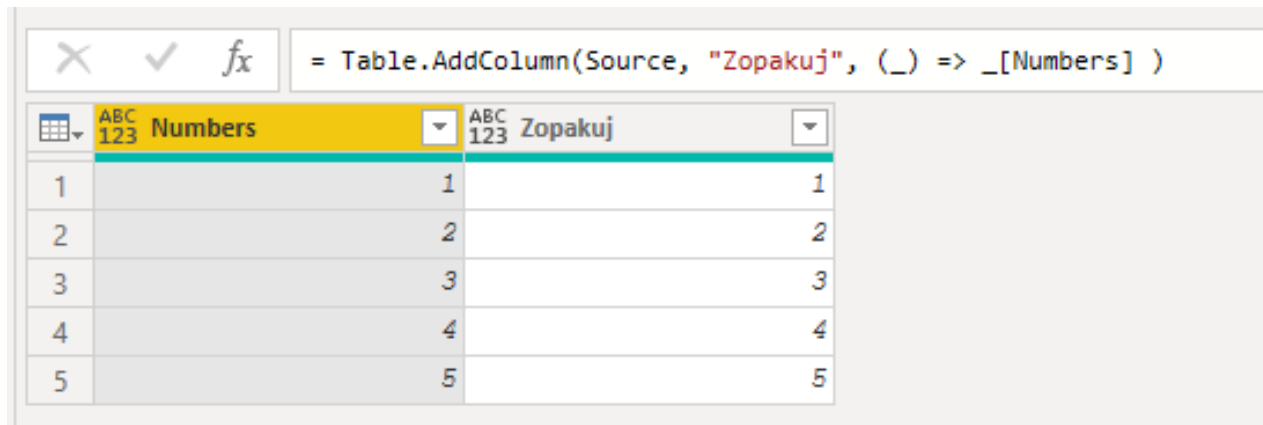
let

```
Source = #table({"Numbers"}, {{1}, {2}, {3}, {4}, {5}}),
```

```
NovySloupec = Table.AddColumn(Source, "Zopakuj", (_) => _[Numbers] )
```

in

NovySloupec



	ABC 123 Numbers	ABC 123 Zopakuj
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5

Each - zopakuj

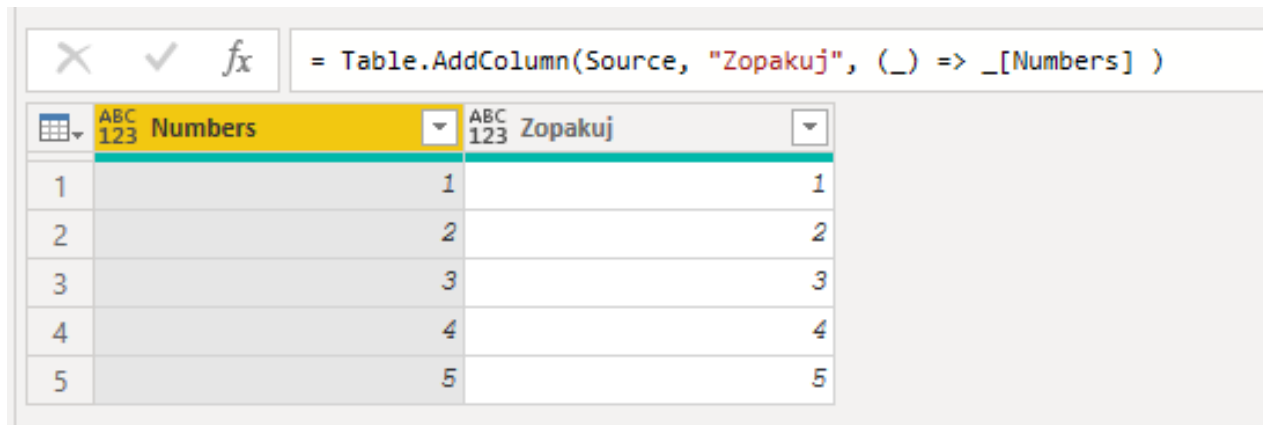
let

```
Source = #table({"Numbers"}, {{1}, {2}, {3}, {4}, {5}}),
```

```
NovySloupec = Table.AddColumn(Source, "Zopakuj", each [Numbers] )
```

in

NovySloupec



The screenshot shows an Excel interface with a table named 'Numbers' and a new column 'Zopakuj'. The formula bar displays the formula: `= Table.AddColumn(Source, "Zopakuj", (_) => _[Numbers] )`. The table has two columns: 'Numbers' and 'Zopakuj'. The 'Numbers' column contains values 1, 2, 3, 4, 5. The 'Zopakuj' column contains the same values 1, 2, 3, 4, 5, demonstrating the 'each' function's behavior.

	Numbers	Zopakuj
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5

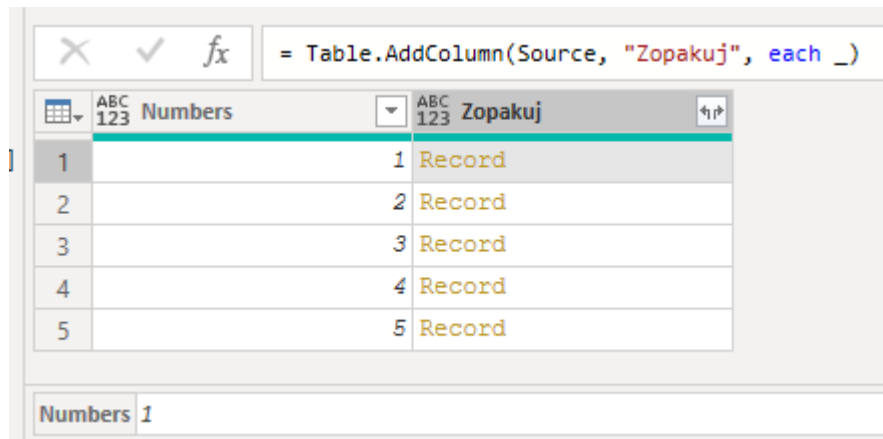
Each - zopakuj

let

```
Source = #table({"Numbers"}, {{1}, {2}, {3}, {4}, {5}}),  
NovySloupec = Table.AddColumn(Source, "Zopakuj", each _)
```

in

NovySloupec



Numbers	Zopakuj
1	Record
2	Record
3	Record
4	Record
5	Record

Each - filtry

```
let
    sample_table = Table.FromColumns({
        {"Pavel", "Eva", "Iva", "Jan"},
        {1, 2, 3, 4},
        {5, 2, 5, 1}},
        {"Jméno", "Znamka_1", "Znamka_2"}),

    filter_znamka_5 = Table.SelectRows(sample_table,
        each ([Znamka_2] = 5))
in
    filter_znamka_5
```

Each - filtry

```
let
    sample_table = Table.FromColumns({
        {"Pavel", "Eva", "Iva", "Jan"},
        {1, 2, 3, 4},
        {5, 2, 5, 1}},
        {"Jméno", "Znamka_1", "Znamka_2"}),

    filter_znamka_5 = Table.SelectRows(sample_table,
        (_) => _[Znamka_2] = 5)

in
    filter_znamka_5
```

Poznámky each

A large, bold, black letter 'L' is positioned in the top right corner of the slide, set against a light blue square background.

Podmínky - if then else

- V Excel KDYŽ (IF)
- Pokud splněno proved' A jinak B

Podmínky - If then else

*if [logický výraz k testování]
then [když je výraz pravda tak se provede]
else [jinak se provede toto]*

```
if [logical expression to test]  
then [do this when true]  
else [do this when false]
```

*if [logický výraz k testování]
 then [když je výraz pravda tak se provede]
else if [pokud předchozí není pravda, logický výraz k testování]
 then [když je výraz pravda tak se provede]
else [jinak se provede toto]*

Logické operátory

- Operátory:

- **not** - negace / opak
- **and** - a
- **or** - nebo

- Ukázky

- `[Number] > 8 and [Number] < 25`
- `[Number] < 15 or [Number] > 25`
- `not ([Number] < 25)`

Data

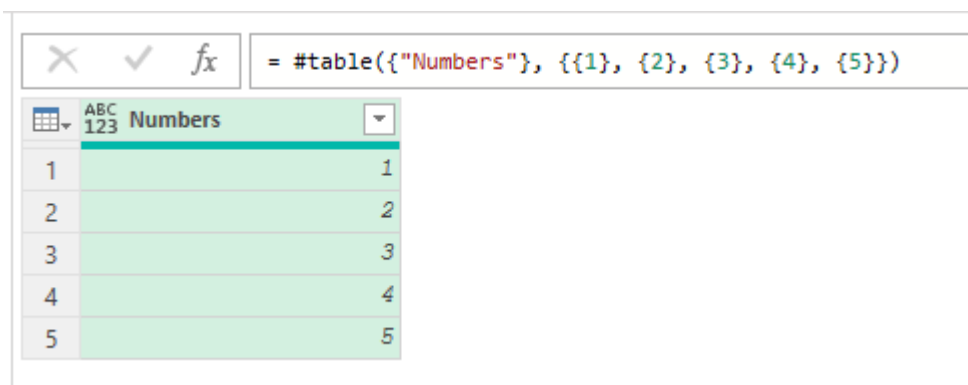
- Mějme tabulku

let

```
Source = #table({"Numbers"}, {{1}}, {2}, {3}, {4}, {5}}),
```

in

Source

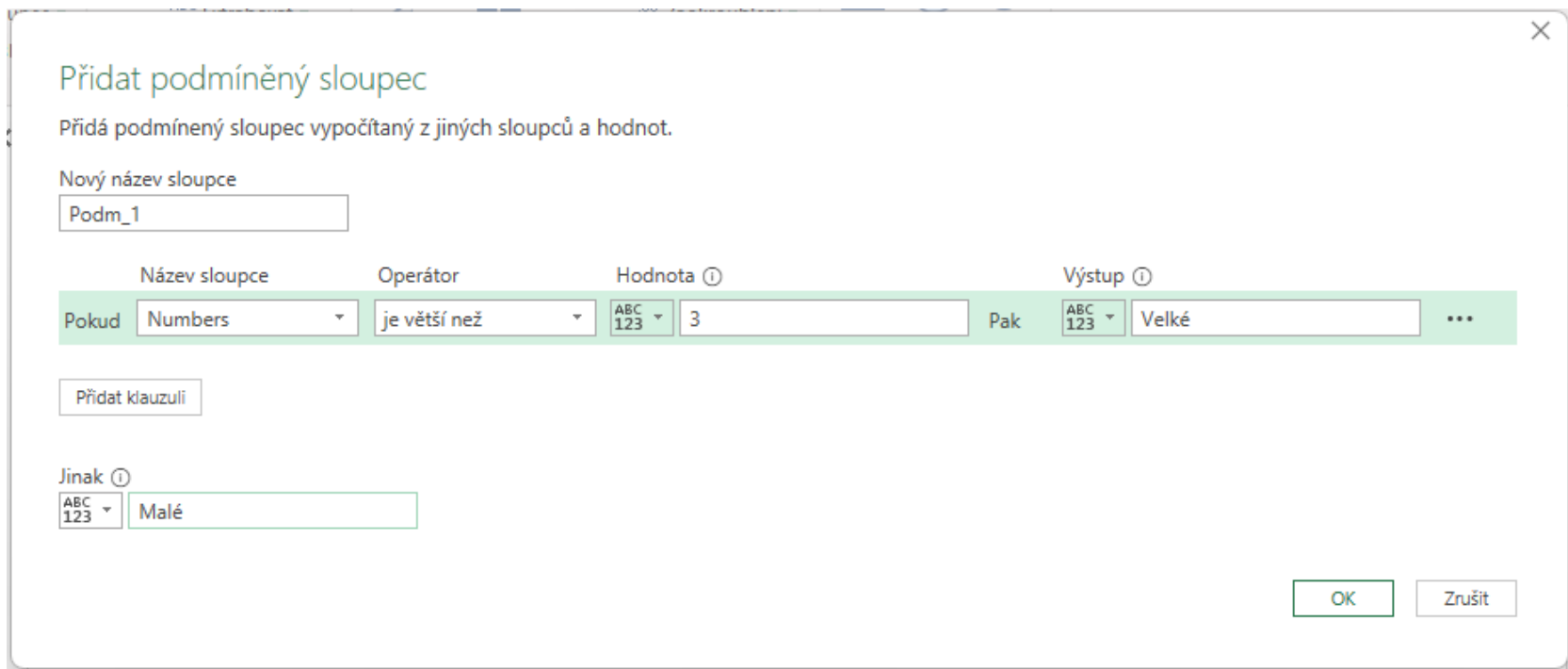


The screenshot shows an Excel interface. At the top, the formula bar contains the DAX formula: `= #table({"Numbers"}, {{1}}, {2}, {3}, {4}, {5}})`. Below the formula bar, a table named "Numbers" is displayed. The table has two columns: the first column contains the numbers 1 through 5, and the second column also contains the numbers 1 through 5. The table is styled with a light green background for the data cells.

	Numbers
1	1
2	2
3	3
4	4
5	5

Podmínky - if then else –ručně-opakování

- V PQE nakliknutí - Přidání sloupce – opakování
 - > 3 „Velké“, jinak „Malé“



Přidat podmíněný sloupec

Přidá podmíněný sloupec vypočítaný z jiných sloupců a hodnot.

Nový název sloupce

Podm_1

	Název sloupce	Operátor	Hodnota ①		Výstup ①
Pokud	Numbers	je větší než	ABC 123 3	Pak	ABC 123 Velké

Přidat klauzuli

Jinak ①

ABC 123 Malé

OK Zrušit

Výsledek

let

```
Source = #table({"Numbers"}, {{1}, {2}, {3}, {4}, {5}}),
```

```
    "Podmíněný sloupec je přidáný" = Table.AddColumn(Source,  
"Podm_1", each if [Numbers] > 3 then "Velké" else "Malé")
```

in

```
"Podmíněný sloupec je přidáný,,
```

Nezměnit název **"Podmíněný sloupec je přidáný"**

Podmínky - If then else – základ

```
if [Číslo] > 3 then "Velké" else "Malé"
```

Podmínka = Table.AddColumn("#Renamed Columns", "Vlastní", each if [Číslo] > 3 then "Velké" else "Malé")

Podmínky - If then else

- Vlastní sloupec

Vlastní sloupec

Přidat sloupec počítaný z ostatních sloupců

Nový název sloupce

Vlastní.1

Vlastní vzorec sloupce ⓘ

```
= if [Čísla] >3 then "Velké" else "Malé"
```

Dostupné sloupce

- Čísla
- Vlastní

<< Vložit

[Další informace o vzorcích produktu Power Query](#)

✓ Nebyly zjištěny žádné chyby syntaxe.

OK Zrušit

Podmínky - If then - více podmínek vnoření

```
if [Čísla] > 4 then ">4"  
    else if [Čísla] > 2 then ">2 <=4"  
    else "<=2"
```

Podmínky - If then else- or

```
let
    Source = #table({"Numbers"}, {{1}, {2}, {3}, {4}, {5}}),
    Sloupec = Table.AddColumn(Source, "Podm_1", each
        if [Numbers] = 3 or [Numbers] = 2 then "3 nebo 2"
        else "jiné")
in
    Sloupec
```

Speciální podmínky or and *null*

- if [A] <> *null* or [B] = „OK" then „Hotovo" else *null*

Neexistující sloupec

```
let
  Source = #table({"Numbers"}, {{1}, {2}, {3}, {4}, {5}}),
  // Sloupec neexistuje?
  Sloupec = Table.AddColumn(Source, "Podm_1", each
    if [C] = 3 or [C] = 2 then "3 nebo 2" else "jiné")
in
  Sloupec
```

- Ošetřit chyby?

Ošetření chyb - try otherwise

- Ošetření chyb v M-ku
- Konstrukce

`try ... otherwise`

- V případě chyby zobrazí požadovanou hodnotu

Příklad

- Sčítejme dva sloupce
 - Kdy se ve sloupci mohou nacházet písmena
- Klasické sečtení
 - Číslo + Číslo > číslo
 - Číslo + Psímeno > Chyba
- Potřebujeme ošetřit

Potřebuji pronásobit dva sloupce

let

```
Zdroj = #table({"Číslo",  
"Číslo_text"},{{1,2},{1,"a"}}),
```

```
Sloupec = Table.AddColumn(Zdroj, "Vypocet", each  
[Číslo]*[Číslo_text])
```

in

Sloupec

	ABC 123 Číslo	ABC 123 Číslo_text	ABC 123 Vypocet
1	1	2	2
2	1	a	Error



Potřebuji pronásobit dva sloupce

let

```
Zdroj = #table({"Číslo",  
"Číslo_text"},{{1,2},{1,"a"}}),
```

```
Sloupec = Table.AddColumn(Zdroj, "Vypocet", each try  
[Číslo]*[Číslo_text] otherwise 0)
```

in

Sloupec

✕ ✓ *f_x*

= Table.AddColumn(Zdroj, "Vypocet", each try [Číslo]*[Číslo_text] otherwise 0)

	ABC 123 Číslo	ABC 123 Číslo_text	ABC 123 Vypocet
1	1	2	2
2	1	a	0



Ošetření chyb - Try otherwise

- Dělení nulou
- try otherwise

Custom Column

Add a column that is computed from the other columns.

New column name

Result

Custom column formula ⓘ

= try [Start] / [End] otherwise 0 |

Available columns

Date

Start

End

<< Insert

[Learn about Power Query formulas](#)

✓ No syntax errors have been detected.

OK Cancel

Načítání dvou různých souborů

Load From Workbook

```
let
    Source =
        try Excel.Workbook(File.Contents("X:\Data\try_otherwise_data.xlsx"), null, true)
        otherwise Excel.Workbook(File.Contents("D:\Data\try_otherwise_data.xlsx"), null, true),
```

close the editor and now my backup file is loaded.

	Date	1 ² ₃ Start	1 ² ₃ End
1	1/01/2021	1	2
2	2/01/2021	2	4
3	3/01/2021	4	2
4	4/01/2021	5	3

Další možností ošetření

- Otazník
 - {}?
 - []?

Poznámky

O

Funkce v Mku - kategorie

- Funkce Lines
- Logické funkce
- Rozdělovací funkce
- Tabulkové funkce
- Textové funkce
- Časové funkce
- Funkce pro typy
- Funkce identifikátorů
URI
- Funkce pro přístup k
datům
- Binární funkce
- Kombinační funkce
- Porovnávací funkce
- Datové funkce
- Funkce DateTime
- Funkce DateTimeZone
- Funkce pro dobu trvání
- Funkce výrazů
- Hodnoty funkcí
- Funkce seznamů
- Funkce čísel
- Funkce záznamu
- Nahrazovací funkce
- Funkce pro hodnoty
- Zpracování chyb

Jak na funkce - syntaxe

- „Nakliknutí“ vs ruční zápis
- Ručně výhody
 - Lze využít skryté možnosti funkce

Jak na funkce - syntaxe prakticky

- Chci z tabulky slova obsahující „*Sto*“
- Všechna!
 - Stokoruna
 - Stovka
 - Tisíc
 - Milión
- Tip Velikost písmen

Vytvořím tabulku slov

```
// Vytvořit tabulku  
let  
    Source = #table({"Slova"}, {{ "Stokoruna"}, {"Sto"},  
    {"Stovka"}, {"Tisíc"}}),  
in  
    Source
```



	ABC 123 Slova
1	Stokoruna
2	sto
3	Stovka
4	Tisíc

Potřebuji slova se „sto“

- **Budu filtrovat**

- (1) Vyberu co chci
 - = Table.SelectRows(Source, each ([Slova] = "Sto" or [Slova] = "Stokoruna"))
- (2) Co když co nechci bude málo
 - = Table.SelectRows(Source, each ([Slova] <> "Tisíc"))
- (3) Má na začátku
 - = Table.SelectRows(Source, each Text.StartsWith([Slova], "Sto"))
- (4) Obsahuje
 - = Table.SelectRows(Source, each Text.Contains([Slova], "Sto"))

Jaký je rozdíl?

- ...
- Co další data...
- Velká malá písmena?

Co další data ve sloupci?

- Co malá / velká písmena?
- Předem
 - Duplikuji sloupec
 - Vše upravím na velká „malá“
 - I ve vyhledávaném dotazu...
 - Následně vyhodnotím?
 - Odstráním sloupec
- Nebo...
- Text.Contains(**text** as nullable text, **substring** as text, optional **comparer** as nullable function) as nullable logical

Text.Contains

Article • 08/04/2022 • 2 minutes to read • [6 contributors](#)

[Feedback](#)



Syntax

```
Text.Contains(text as nullable text, substring as text, optional comparer as nu
```

About

Detects whether `text` contains the value `substring`. Returns true if the value is found. This function doesn't support wildcards or regular expressions.

The optional argument `comparer` can be used to specify case-insensitive or culture and locale-aware comparisons. The following built-in comparers are available in the formula language:

- [Comparer.Ordinal](#): Used to perform a case-sensitive ordinal comparison
- [Comparer.OrdinalIgnoreCase](#): Used to perform a case-insensitive ordinal comparison
- [Comparer.FromCulture](#): Used to perform a culture-aware comparison

Rozlišuj malé velká

- = Table.SelectRows(Source, each Text.Contains([Slova], "Sto", **Comparer.OrdinalIgnoreCase**))

Vlastní funkce

Q

- Bez Google ani ránu

Vlastní funkce

```
// přičti jedničku  
let  
    PrictiJednicku = (x) => x + 1,  
    // Volání funkce  
    Výsledek = PrictiJednicku (10)  
in  
    Výsledek
```

Vlastní funkce

```
// a * b + 100  
let  
    Produkt = (x, y) => x * y + 100,  
    Výsledek = Produkt (2,3)  
in  
    Výsledek
```

Odkazování se na proměnné

- Viz List, Record, Tabulka

Poznámky

Q

Tip Google a nápověda...

R

- Název listu vs pořadové číslo listu v Excel

Načítání Excel

- Co se nám nelíbilo / způsobilo problém?
 - Název listů
 - Co pořadí listu
- Zjednodušení?
 - Počet řádků
 - Zrychlení...

(1) Název vs pořadové číslo listu v Excel

```
// UpravitŘádky_Sheet =  
Source{ [Item="UpravitŘádky", Kind="Sheet"] } [Data],  
// Source{0} [Data]  
UpravitŘádky_Sheet = Source{0} [Data],
```

Listy ...

- Další řešení?
- Google
-

Další informace

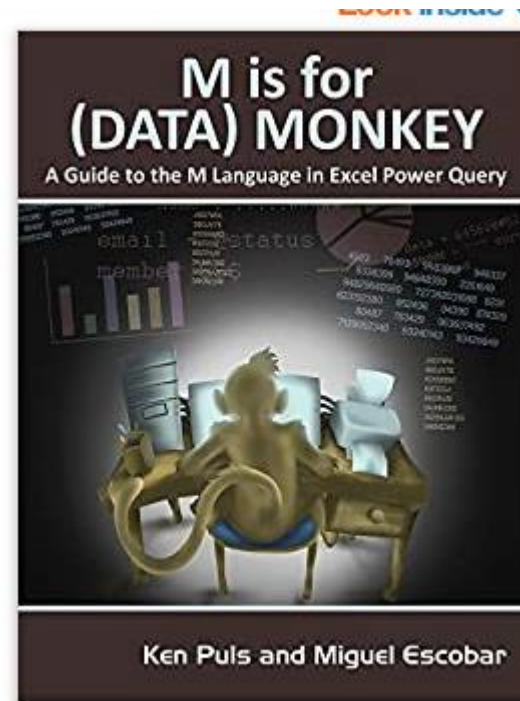
- <https://office.lasakovi.com/excel/Query-editor/struktura-M-kodu-Editor-Power-Query/>

Kam dále o M

- Video
- Knihy

Knihy

- <https://www.amazon.com/Data-Monkey-Guide-Language-Excel/dp/1615470344>



Weby o Mku

- **Imke Feldmann**

- <https://www.linkedin.com/in/imkefeldmann?originalSubdomain=de>
- <https://www.thebiccountant.com/>

- **Chris Webb**

- <https://blog.crossjoin.co.uk/>

Weby o Mku

Guy in a Cube https://www.youtube.com/results?search_query=guy+in+cube

Ken Puls https://www.youtube.com/watch?v=1Pym6Z36S0U&ab_channel=PowerBI%C4%B0stanbul

Curbal <https://www.youtube.com/c/CurbalEN>

<https://curbal.com/contact>

Matt Allington <https://exceleratorbi.com.au/>

https://www.youtube.com/watch?v=1KKalqmpcDg&ab_channel=DAXTIPS

Marco Russo <https://www.sqlbi.com/>

Alberto Ferrari <https://www.linkedin.com/in/albertoferrarisqlbi/>

Weby o Mku

- **Reza Rad**

- <https://radacad.com/reza>
- <https://radacad.com/blog>
- <https://www.youtube.com/channel/UCsOfIwAXj1fT6LDqEDEAb4g>

- **Guy in a Cube**

- https://www.youtube.com/results?search_query=guy+in+cube

Poznámky

Z