Power BI desktop

Pavel Lasák - Excel, Power BI, G-tabulky

Power BI Desktop

Načtení

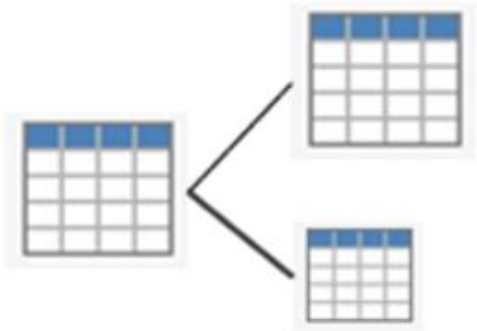
Soubory (xls, csv, pdf)
Složky
Web
Databáze

Úprava

Práce s řádky
Úpravy se sloupci
Řazení filtrování
Transformace

M - jazyk

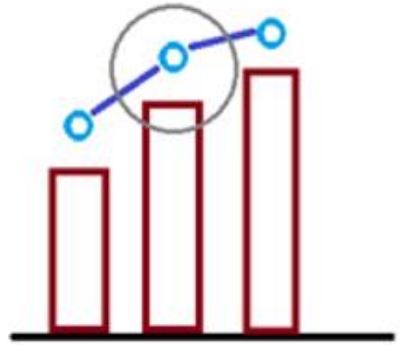
Propojení



Propojení,slučování
Hierarchie
Vazby

DAX

Vizualizace



Grafy
Ovládací prvky
Texty
Obrázky

Sdílení

Pdf
Csv
Cloud
….

DNES

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

- Označován
  - 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

A

- 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

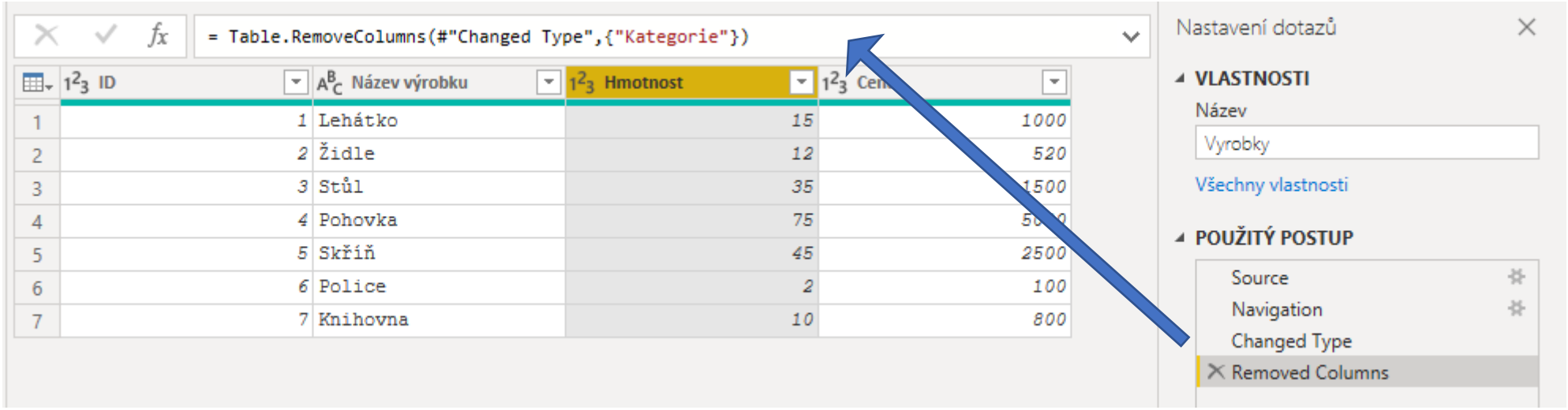
A

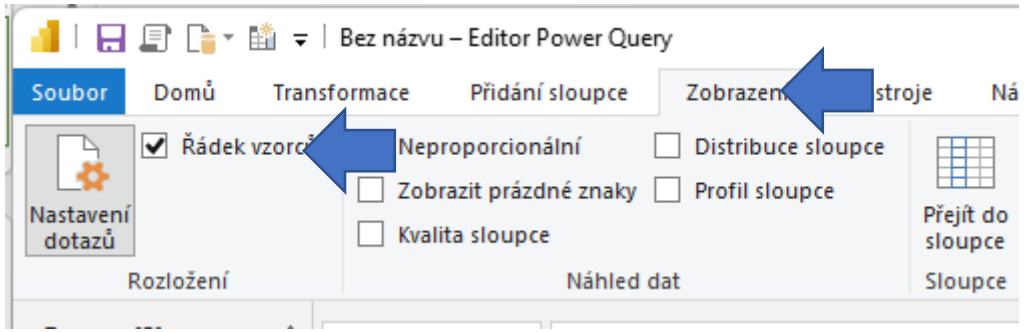
- Méně kroků > podstatně rychlejší
- Přehlednější
  - Pro opravy
  - Vylepšování doplňování > Poznámky
  - Hledání problémů
  - Univerzálnost
- Spoustu úprav „**ne**nakliknete“ v GUI ☹
  - Hlavně pokročilé věci (**?**, další parametry funkcí, atd.)

Kde můžete (část) **M-kód** vidět

A

- Řádek vzorců





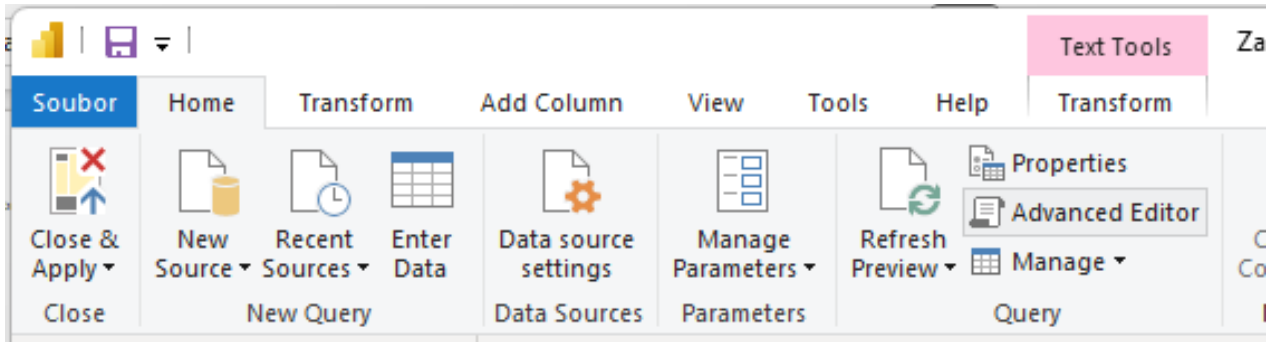
Spuštění M-ka – pokročilý editor

A

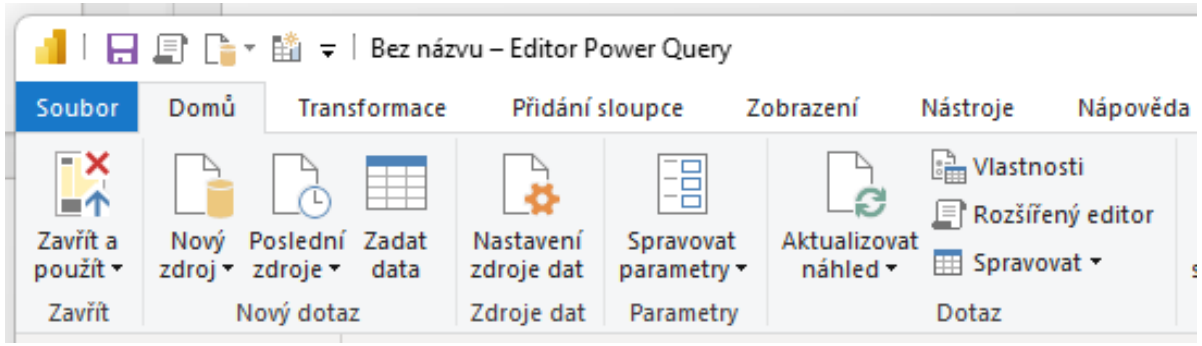
- Home > Advanced Editor EN
- Domů > Rozšířený editor CZ

- Tip přemístit na rychlý pás karet

EN

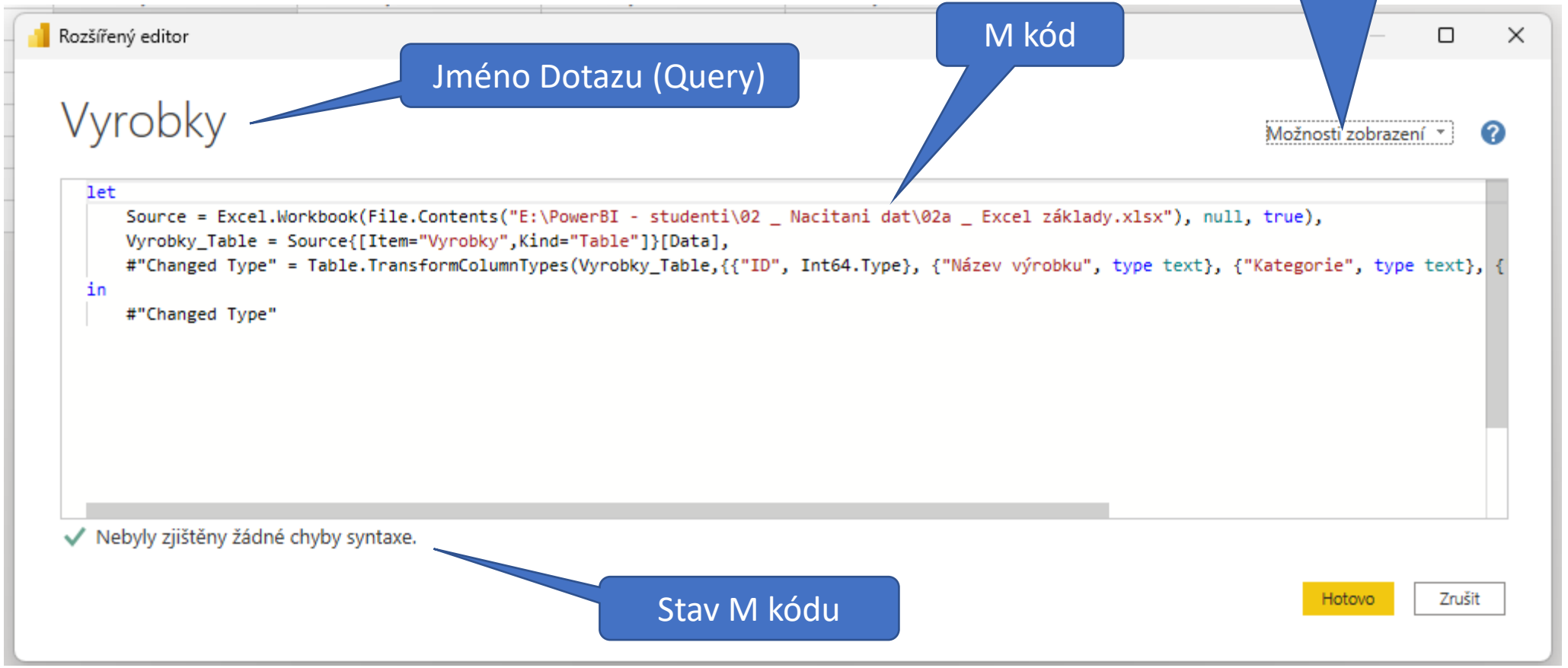


CZ



Rozšířený editor

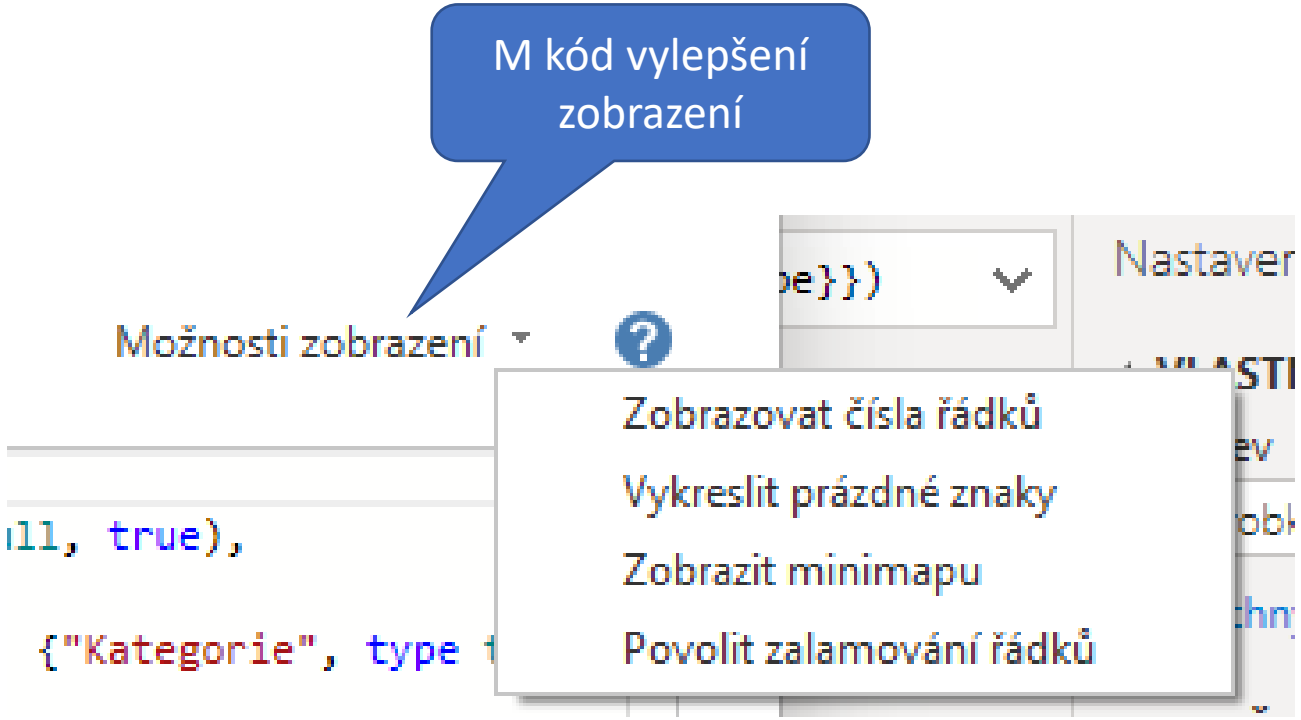
A



Nastavení rozšířeného editoru

A

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



Zpřehlednění M-kódu

A

- Pro zpřehlednění kód formátovat
- Podobně jako DAX

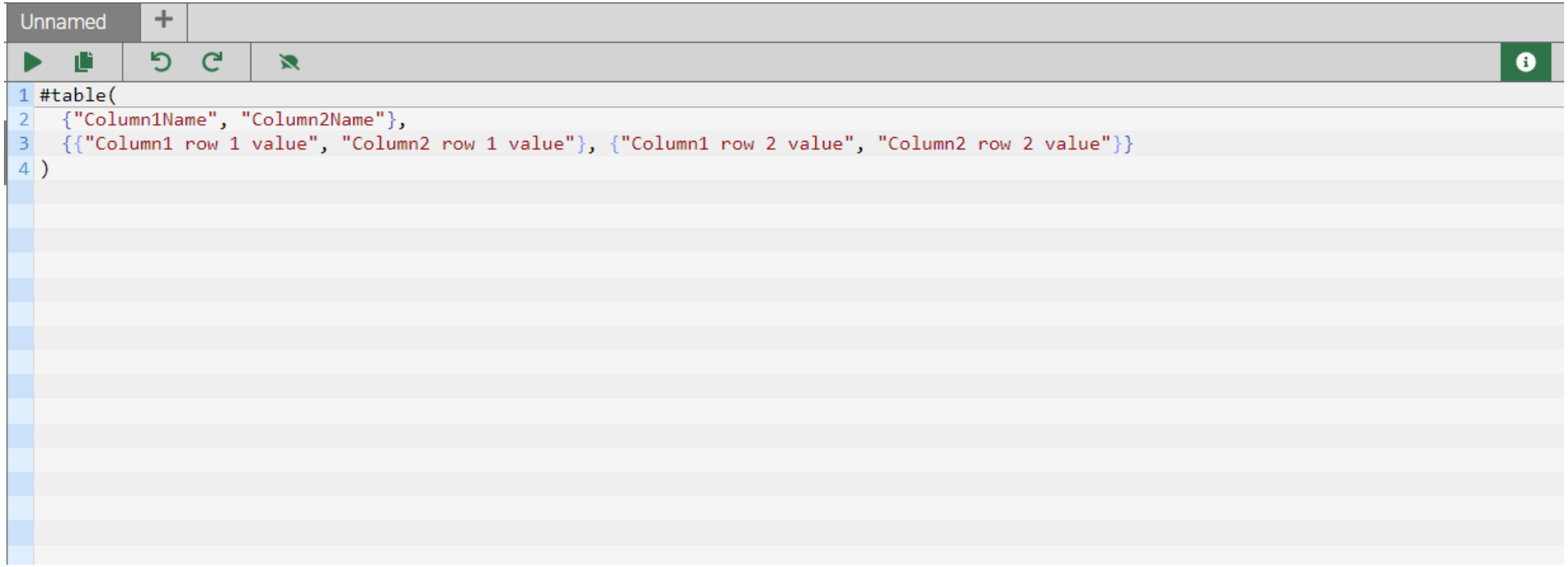
```
let
    Source = Excel.Workbook(File.Contents("E:\PowerBI - studenti\02 _ Nacitani dat\02a _ Excel základy.xlsx"), null, true),
    Vyrobky_Table = Source{[Item="Vyrobky",Kind="Table"]}[Data],
    #"Changed Type" = Table.TransformColumnTypes(Vyrobky_Table,{{"ID", Int64.Type}, {"Název výrobku", type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

- Tip: později podrobněji

Formátování

A

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



Formátovaný kód

A

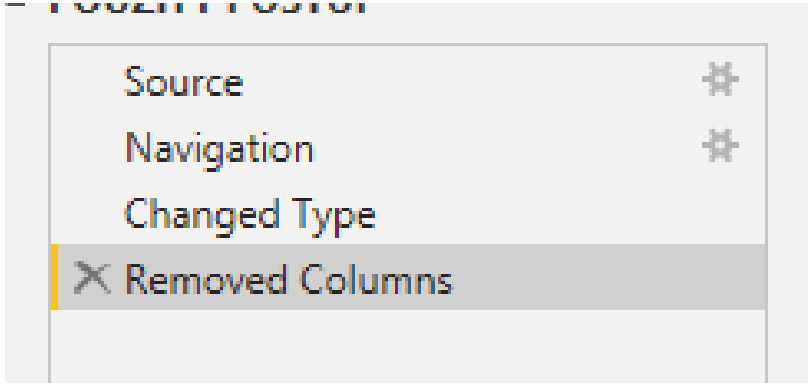
```
let
  Source = Excel.Workbook(
    File.Contents("E:\PowerBI - studenti\02 _ Nacitani dat\02a _ Excel základy.xlsx"),
    null,
    true
  ),
  Vyrobky_Table = Source{[Item = "Vyrobky", Kind = "Table"]}[Data],
  #"Changed Type" = Table.TransformColumnTypes(
    Vyrobky_Table,
    {
      {"ID", Int64.Type},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

Další vylepšení M-ka při tvorbě v GUI

A

- Názvy kroků (bez mezer)
  - Řádek nový
  - **NovyRadek**
  - New Row
  - **NewRow**

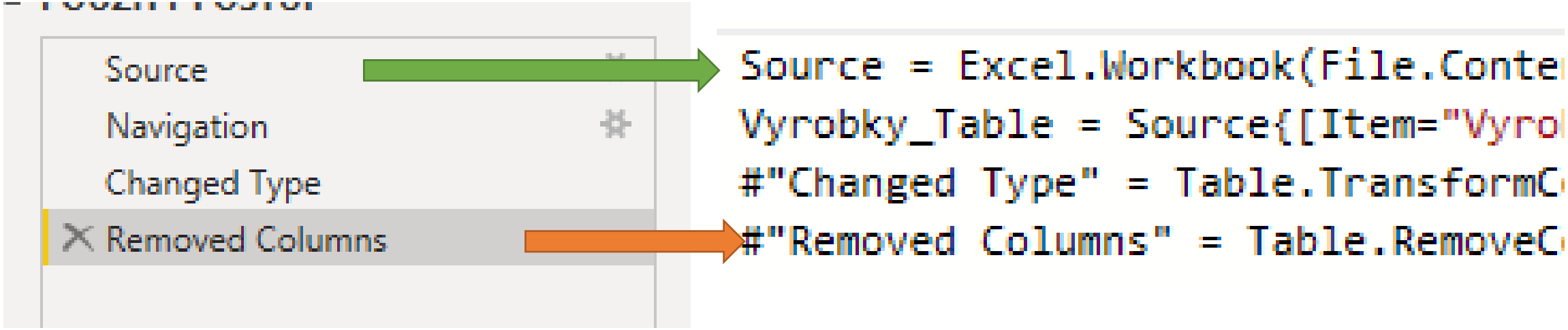
- Tip
  - Bez háčku a čárek
  - Bez mezer
  - Raději Anglicky



Další vylepšení Mka při tvorbě v GUI

A

- Názvy > Přejmenovat
  - Už umíme



Poznámky

A

Základní znaky (operátory) v syntaxi M

B

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

M-ko a znaky - Matematické operátory

B

- +
- -
- *
- /
- ^

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

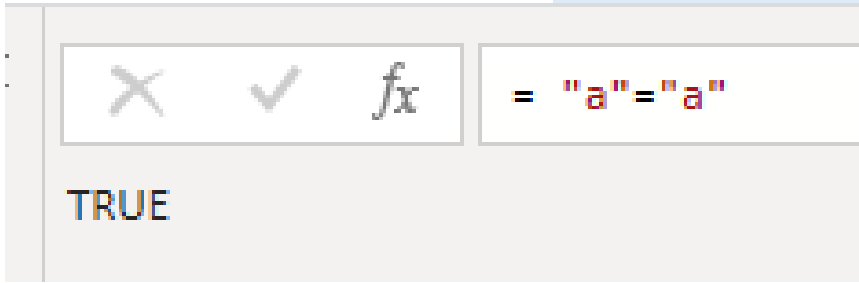
M-ko a znaky - Porovnávací operátory

B

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

M-ko a znaky - Porovnávací operátory

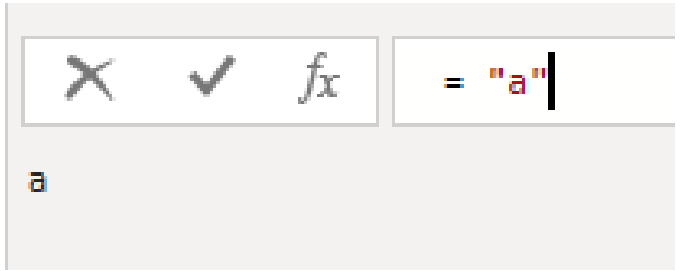
B



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 <= y se vyhodnotí jako true, pokud x je menší nebo rovno y
- x >= y se vyhodnotí jako true, pokud x je větší nebo rovno y
- x <> y se vyhodnotí jako true, pokud x není rovno y

- = může jít i o přiřazení



MUNI

Logické operátory

B

- 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

B

- 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

B

- **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

B

```
• 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

C

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

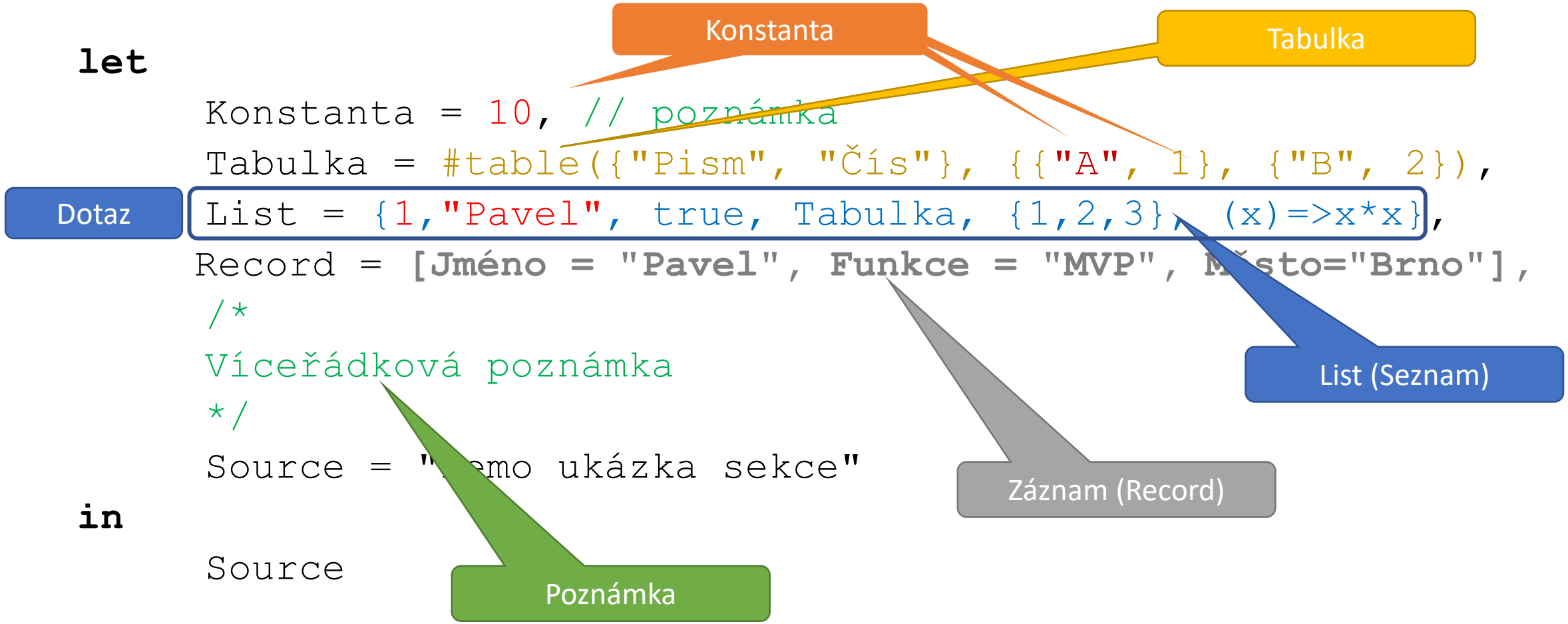
„Sekce“ kroky v M-ku

C

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

Sekce co může M-kód obsahovat

C




```
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 = "Demo ukázka sekce"
in
    Source
```

Hodnoty (Values)

C

- **Hodnota** je kus datové informace nebo-li:

- Druhy hodnot:
  - Hodnoty jednoduché:
  - Hodnoty strukturované:

Hodnoty (Values)

C

- **Hodnoty jednoduché:**
  - čísla
  - text
  - logická hodnota
  - binární
  - datu
  - čas
  - …

Hodnoty (Values)

C

- **Hodnoty strukturované:**
  - **List – seznam** samostatná sekce
  - **Record – záznam** samostatná sekce
  - **Tabulka – Table** samostatná sekce

  - seznamy seznamů atd.
  - seznamy tabulek
  - …

Dotazy / Výrazy (Expressions)

C

- **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)

C

```
Soucet = 1 + 1
```

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

- Each…
- If .. then
- ..

Funkce

C

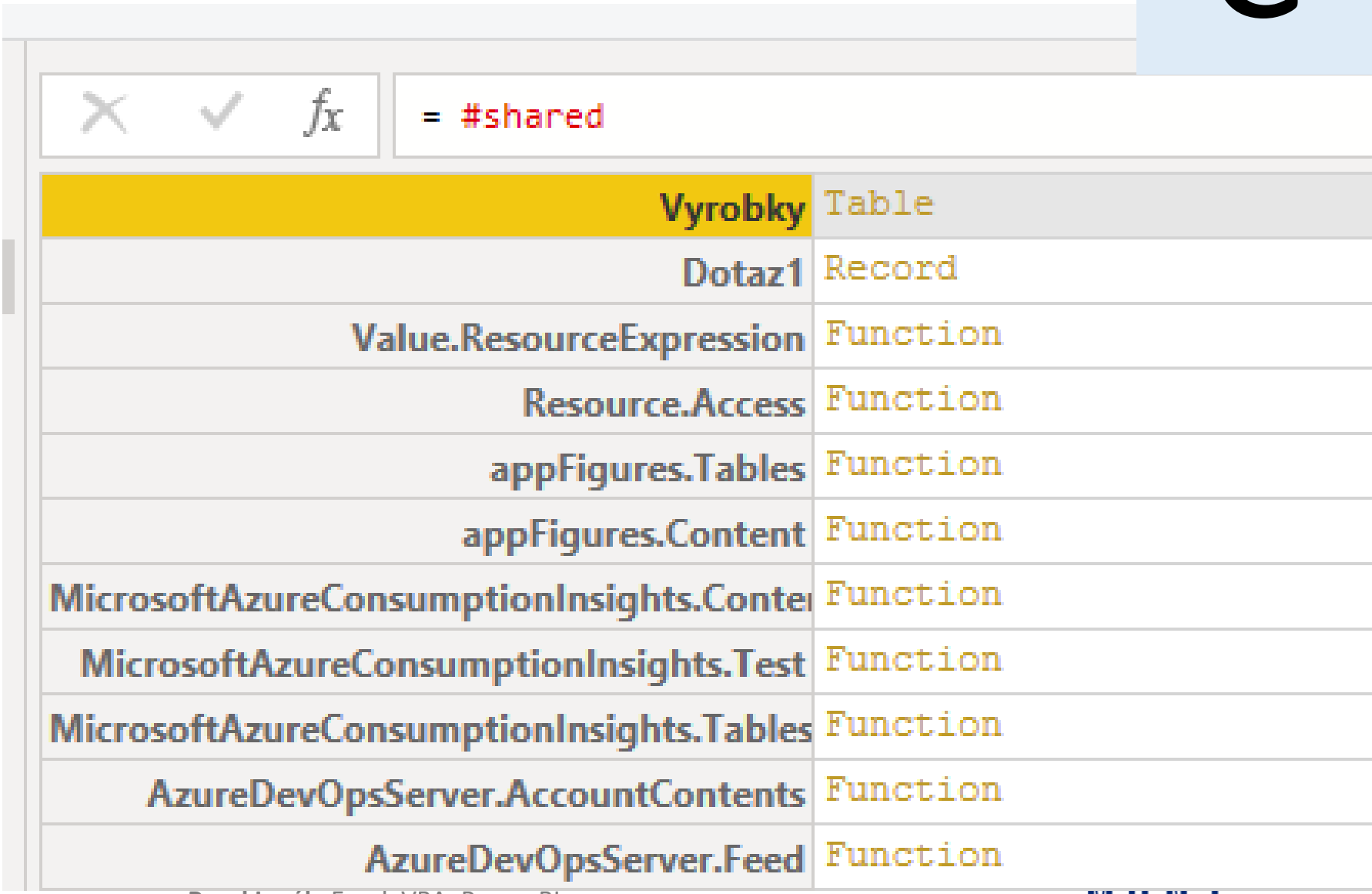
- Někdo řadí samostatně
- Viz samostatná sekce

- Tip Google

=#shared

- Nový dotaz
- Tip

Google



Poznámky

C

- 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

D

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

Samostatné hodnoty

D

- 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

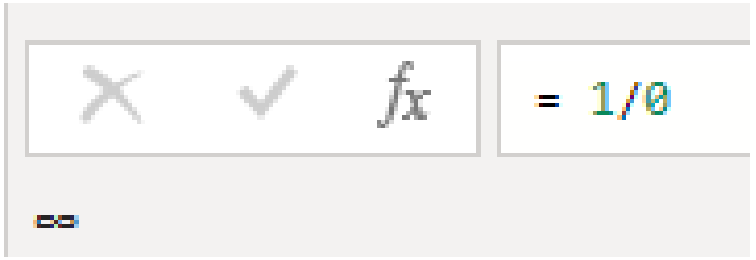
D

- 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

D

- ∞ kladné nekonečno **= 1/0**
- **- ∞** 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

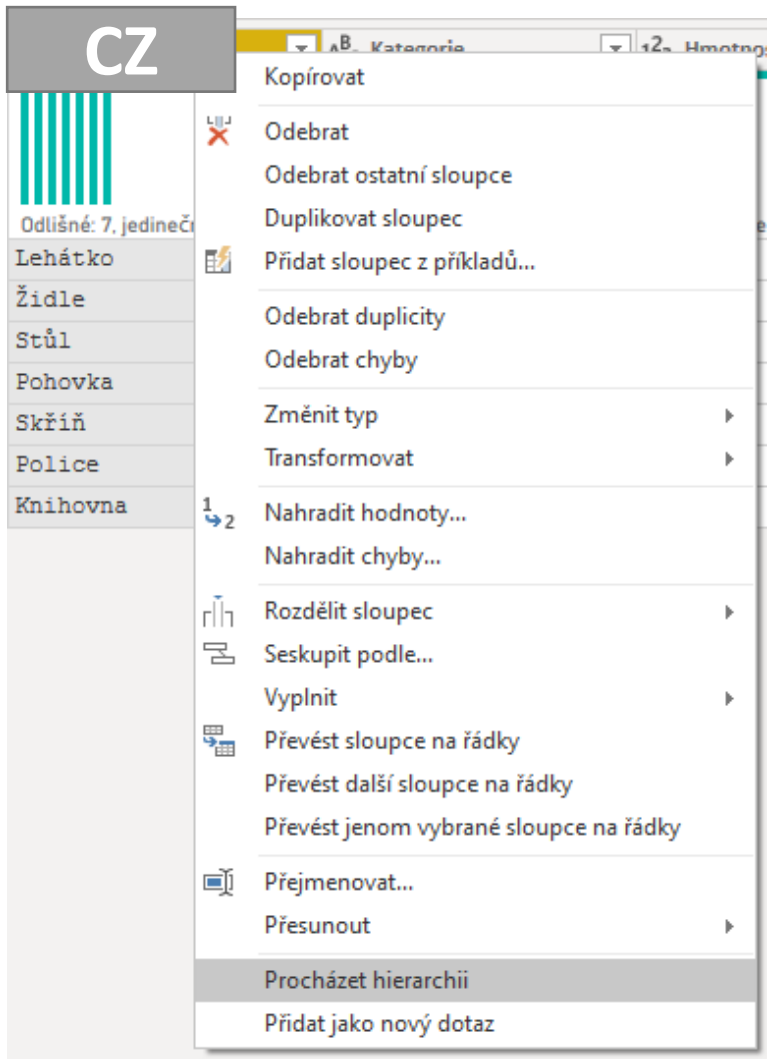
E

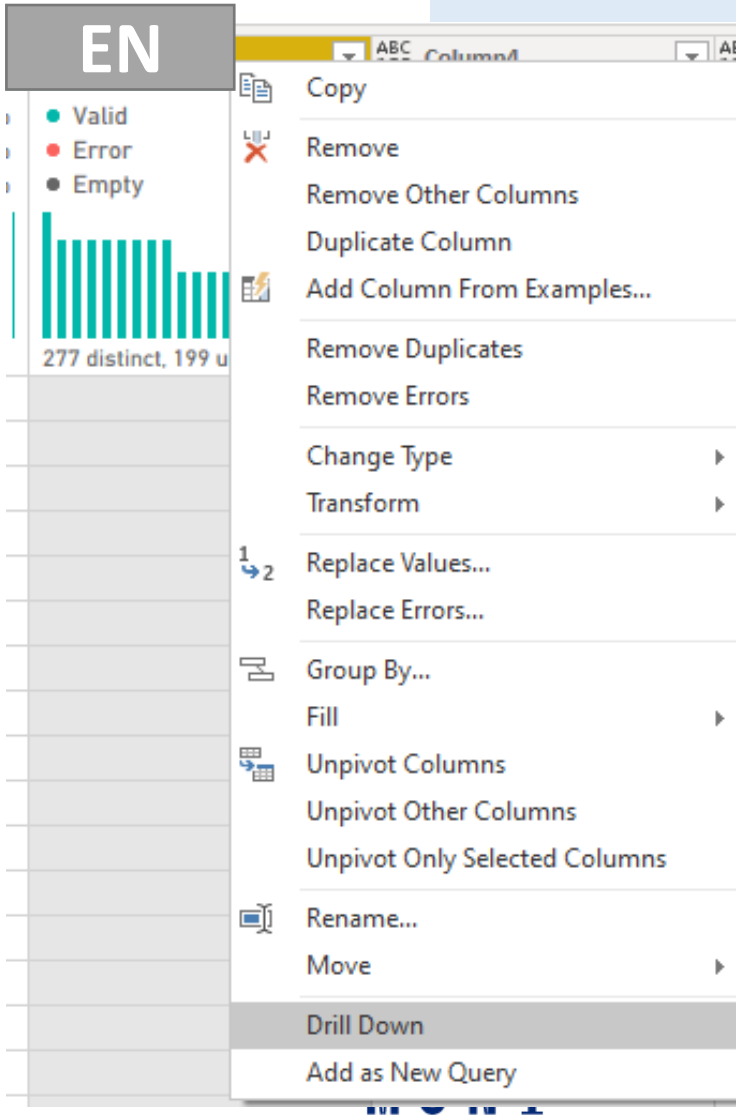
- Řada hodnot
  - Primitivní
  - Strukturované
- Není limitován počtem hodnot
  - Limit je u funkcí (pokud budou list zpracovávat)
  - Limit HW
- Bez vektoru se neobejdete
- Vektor může obsahovat vektor, tabulku atd…
- Tip: Sloupec jako seznam

List - Seznam - Vytvoření

E

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

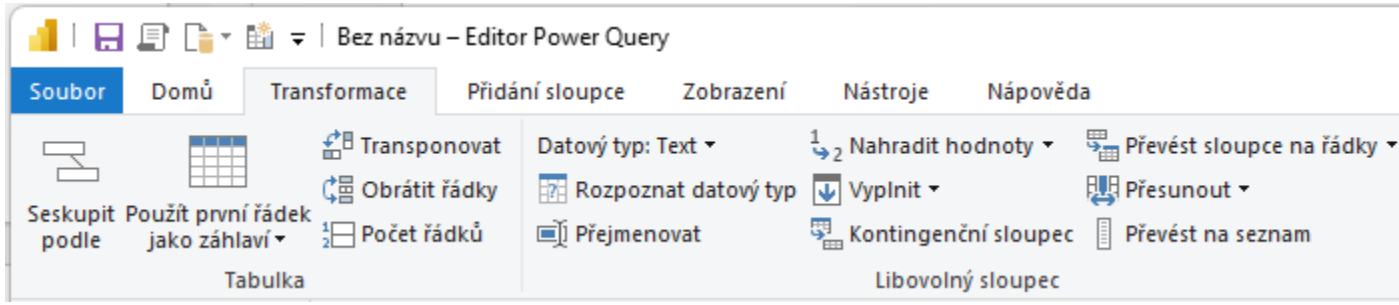


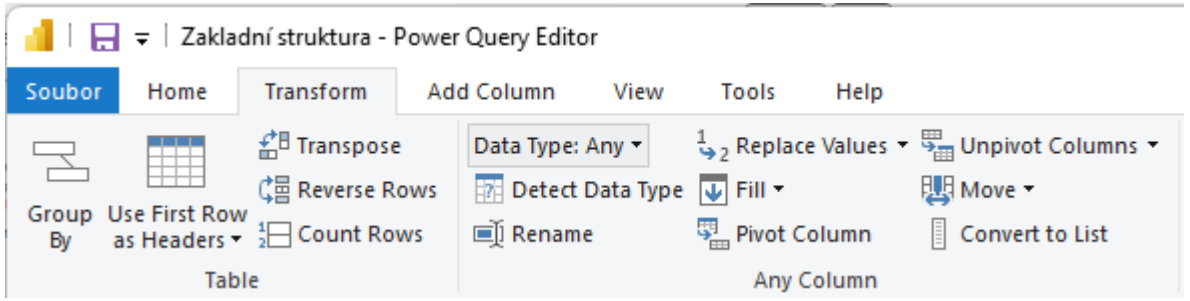


List - Seznam - Vytvoření

E

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





List - Seznam - Vytvoření

E

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

List - Seznam – Vytvoření

E

(3) List lze vytvořit „Ručně“

```
={1,2}
={1..2}
={{1, 2}, {3, 4, 5}}
= {"a".."f"}
```

…

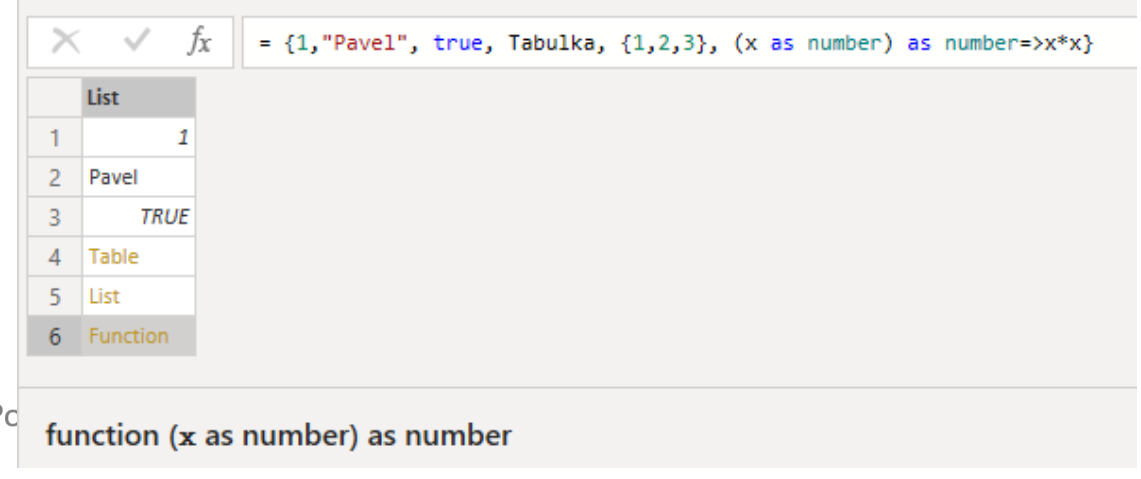
List - Seznam – Vytvoření

E

„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?

E

```
={1..12}
```

- Měsíce v roce?

```
={1..7}          ={1..5}
```

- Dny v týdnu (Po - Pá)

```
={"po", ….}
```

Klasický seznam výběr prvku

E

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

- První prvek

```
= {1, 2, 3}{0}
```

Klasický seznam výběr prvku

E

- V Módu před odkaz na dotaz

```
let
    Seznam = {1,2,3},
    Source = Seznam{1}
in
    Source
```

Klasický seznam - odkazování

E

```
= 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}?
```

Otazník je skvělá věc !!!

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

```
= List_vycet_prvku{-7}?
```

- Chyba -7 prvek nemůže existovat, 7 může

Klasický seznam – výběr prvku

E

- 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

E

- 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…

E

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

List - Seznam - Poznámky

E

- Odkaz_web

Záznam Rekord

F

- Strukturovaný datový typ
- Skupina pojmenovaných hodnot
  - Název hodnoty, Hodnota

- Záznam je v každém PQ
  - Let .. in… je vlastně záznam – viz dále

Vytvoření Recort - záznam

F

- (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

F

- (2) Využiji funkce

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

– vrátí aktuální čas

- Převedení času na zaznam

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

Záznam (Record) ze Seznamu (List)

F

(2) Využiji funkce

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

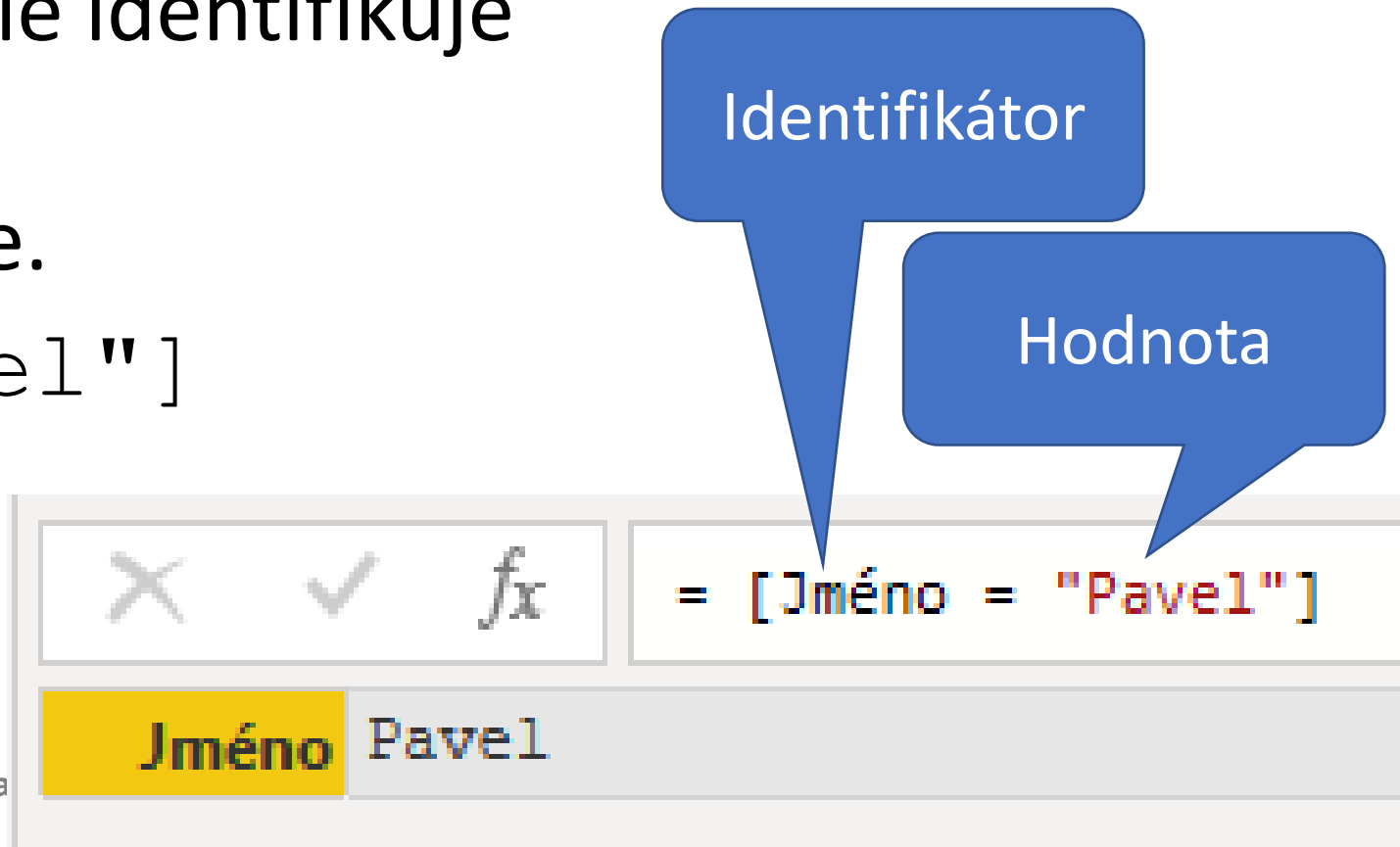
```
let
    Source = Record.FromList({1, "Pavel", "MVP"}, type [ID = number, Jmeno = text, Úspěch = text])
in
    Source
```

Viz dále ruční vytvoření

(3) Ruční zadání - Záznam Rekord

F

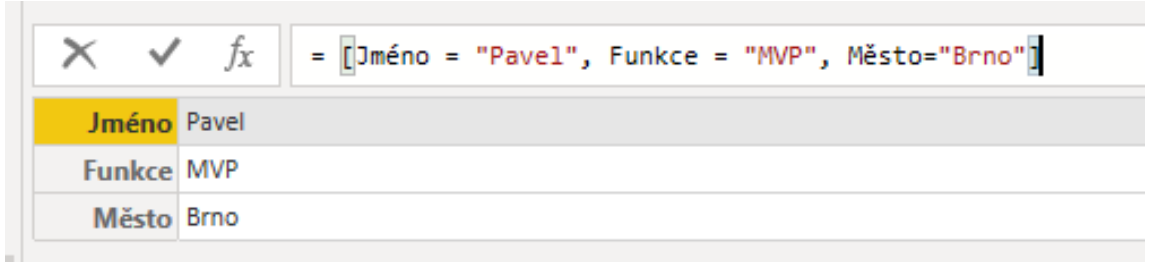
- Definuje pomocí hranatých závorek []
  - Jméno které toto pole identifikuje
  - Znaménko rovná se.
  - Hodnota tohoto pole.
- `= [Jméno = "Pavel"]`



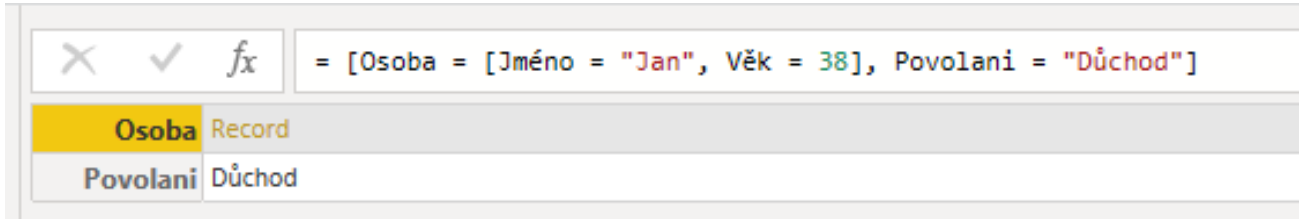
(3) Record Ručně

F

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




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

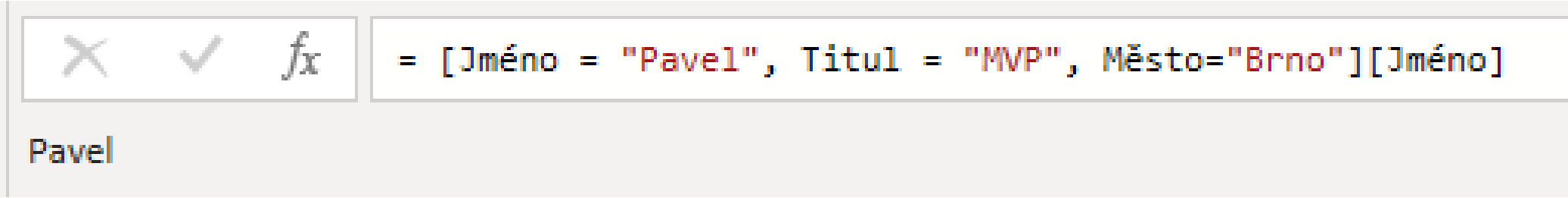


Práce se záznamy-výběr záznamu

F

- Výběr prvku
  - Má pojmenování
  - Potřebuji jméno

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

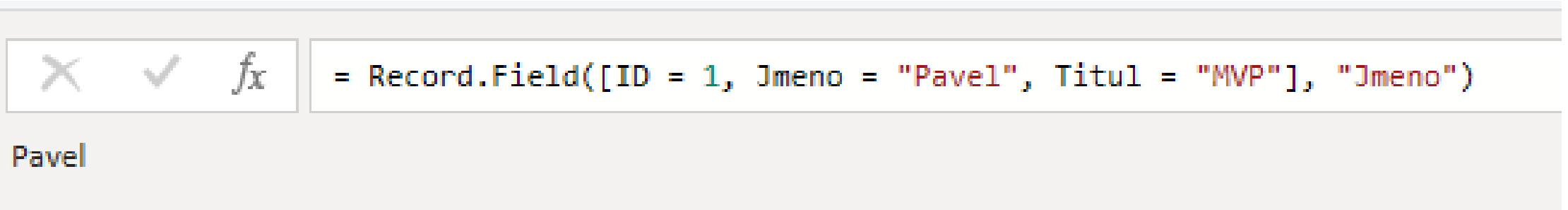


Poznámky – práce se záznamy

F

- Využitím funkce

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



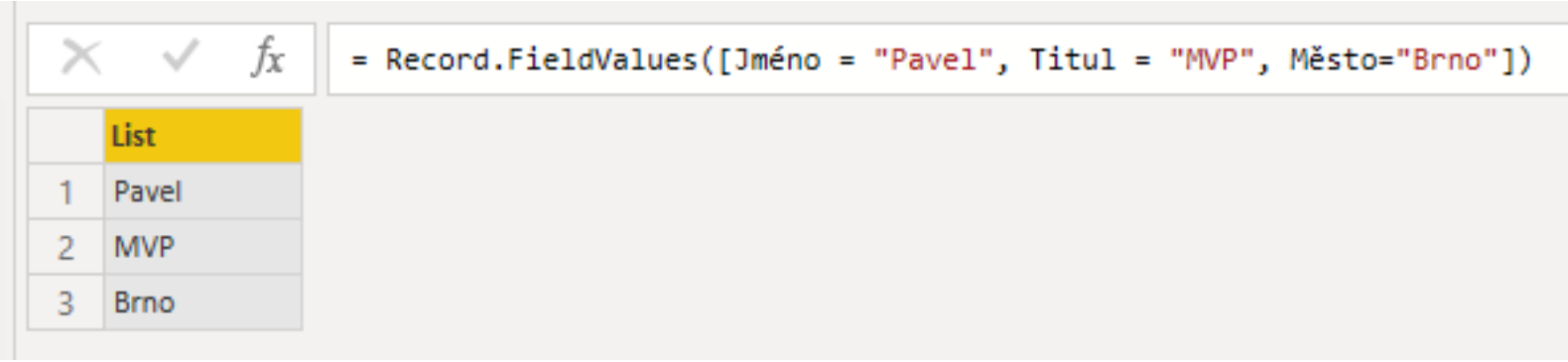
Není lepší?

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

Z Record (Záznam) List (Seznam)

F

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



- opačně `Record.FromList`

Nepřipomíná vám něco?

F

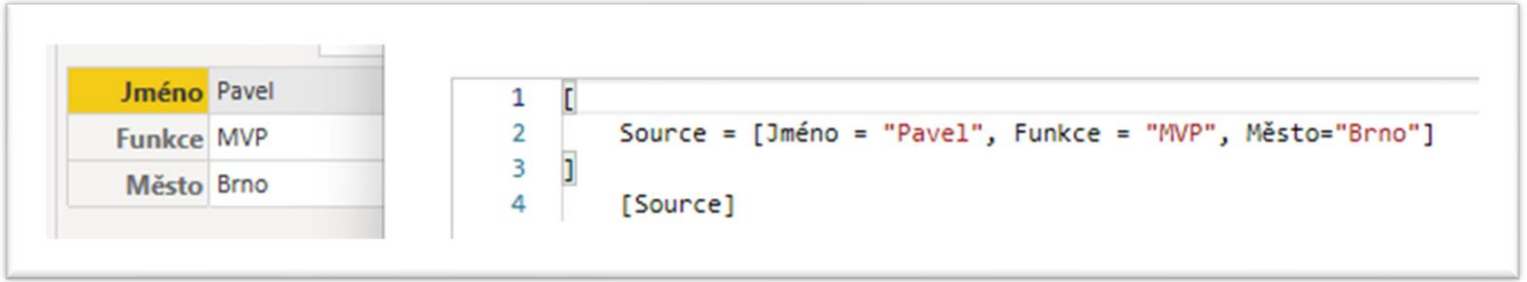
```
let
    Source = [Jméno = "Pavel", Funkce = "MVP",
Město="Brno"]
in
    Source
```

Co záměna **let in** znaky **[]**?

Nepřipomíná vám něco?

F

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

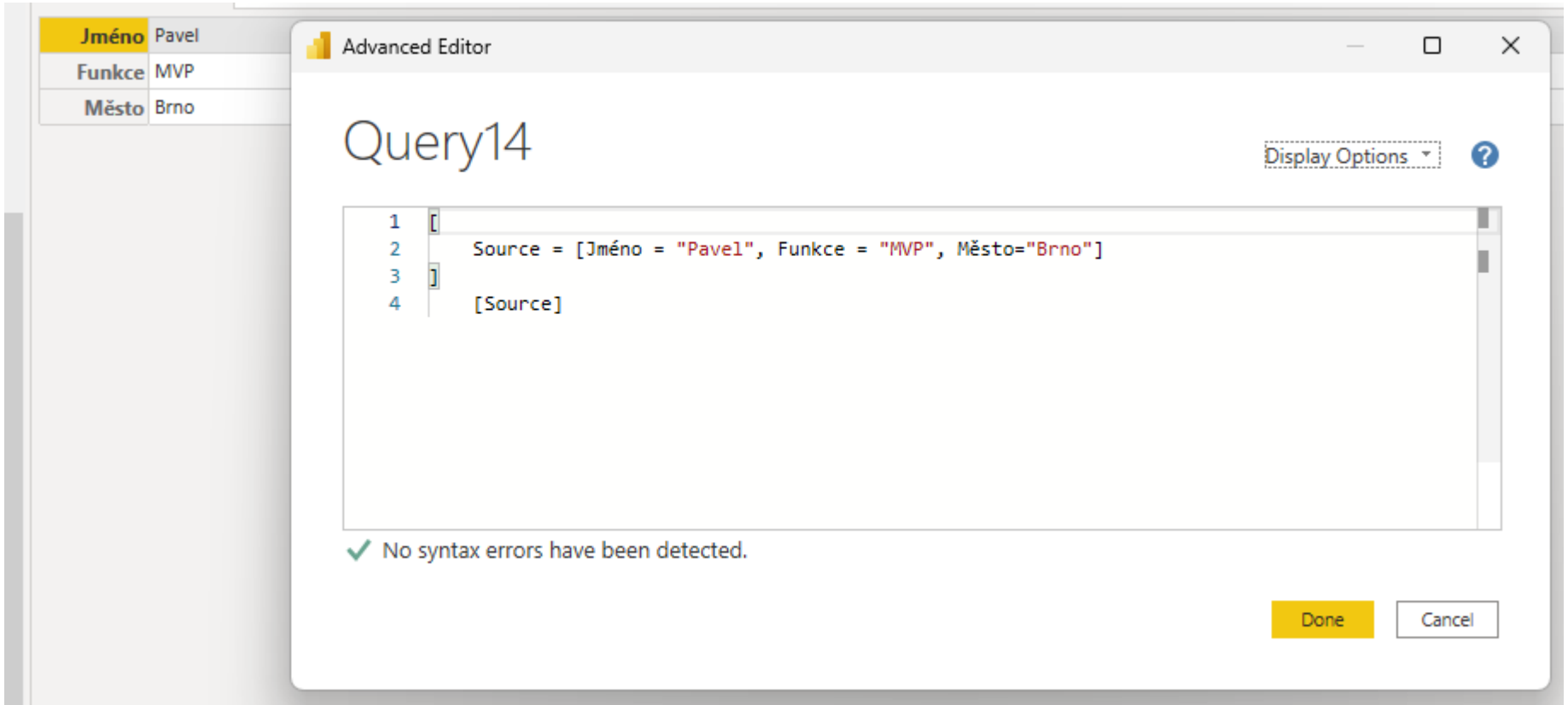


Nepřipomíná vám něco?

F

Nebo lépe

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



MUNI

Prostor pro poznámky

F

Tabulky (Table)

G

- 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

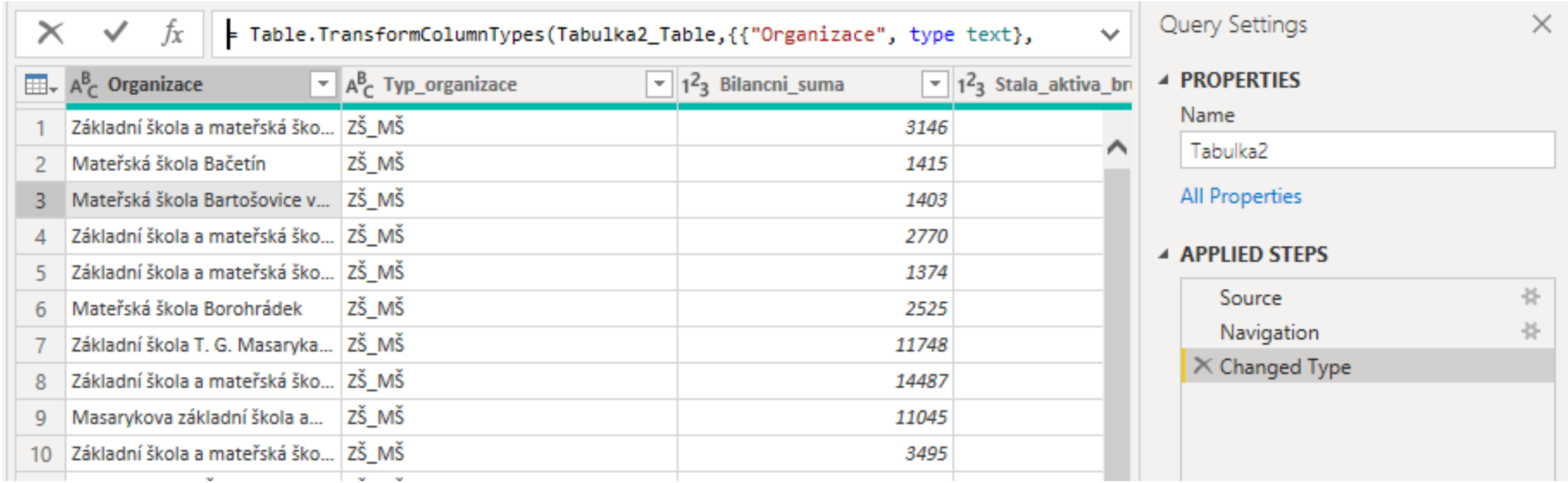
G

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

(1) V PQE

G

- Využitím PQ



	Organizace	Typ_organizace	Bilancni_suma	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	

(2) Funkci vytvoření z List

G

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

(2)Funkci vytvoření z Record

G

```
= 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

G

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

(2) Funkci další možnost

G

- ```
  Table.FromRows(Json.Document(Binary.Decompress(
  Binary.FromText("i45WCkgsS81R0lHySSw+vDBbKVYnWs
  m1LBEo4JuYc3ghmO8J5oel5mSDBGIB",
  BinaryEncoding.Base64), Compression.Deflate)),
  let _t = ((type nullable text) meta
  [Serialized.Text = true]) in type table [Jméno
  = _t, Příjmení = _t]),
  ```
  ```

G

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

Jméno funkce

Definice datové typy

Názvy sloupců

datové typy sloupců

Data pro první řádek

Data pro druhý řádek

(3) Ručně

G

- Tabulka se sestaví pomocí funkce

```
#table
```

- první seznam názvy sloupců

```
{"NázevSloupec", „Sl2"}
```

- jednotlivé řádky se zapisují opět jako seznam

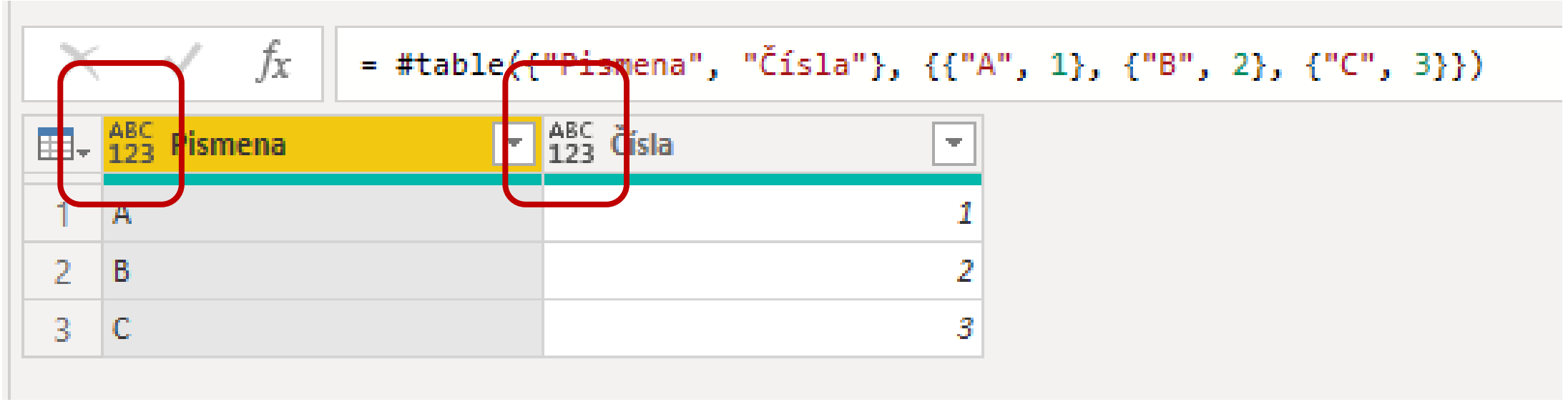
```
{"A", 1}
```

- s skupina záznamu je opět ve složených závorkách.

(3) Ručně – názvy sloupců

G

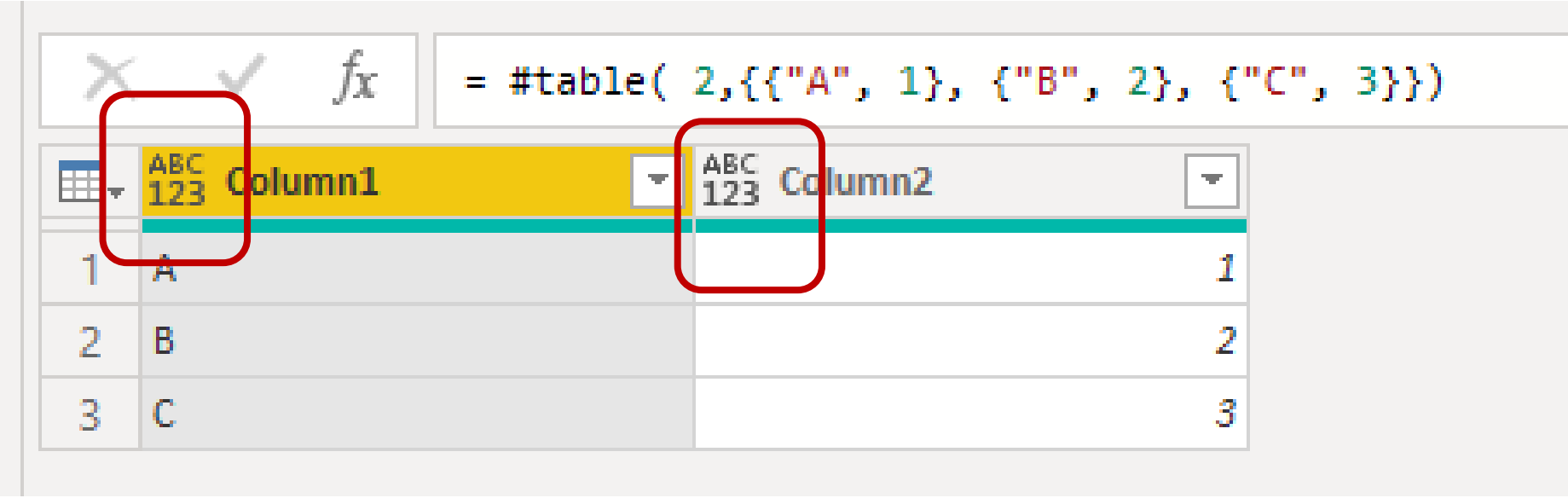
```
= #table({"Pismena", "Čísla"}, {{"A", 1}, {"B", 2}, {"C", 3}})
```



(3) ručně-bez názvů sloupců

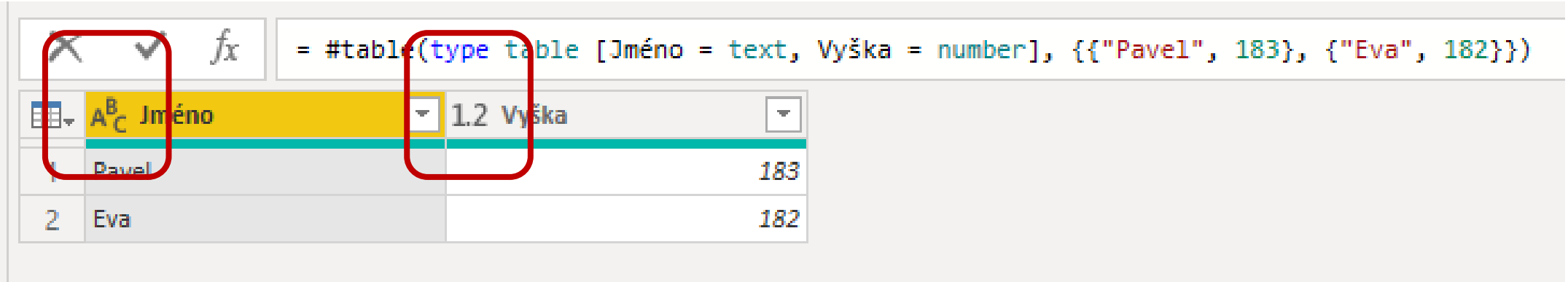
G

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



(3) Ručně – názvy sloupců a typ hodnoty

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



Práce s tabulkou

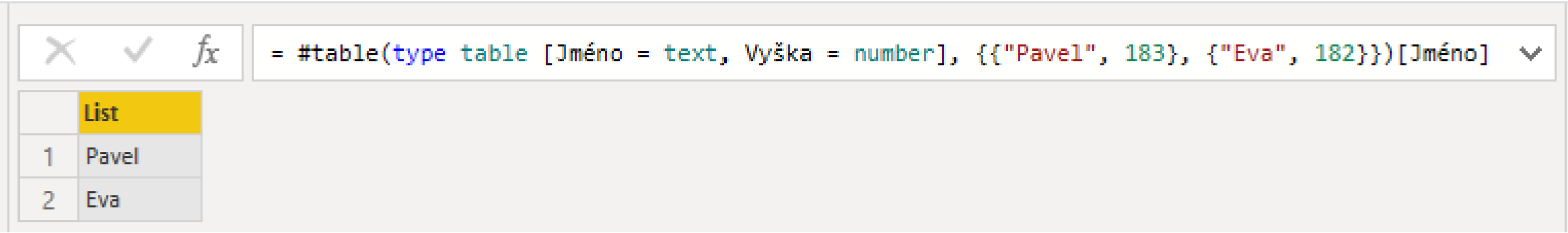
G

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

Práce s tabulkou - sloupec

G

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



Práce s tabulkou - sloupec

G

- Případně přehledněji

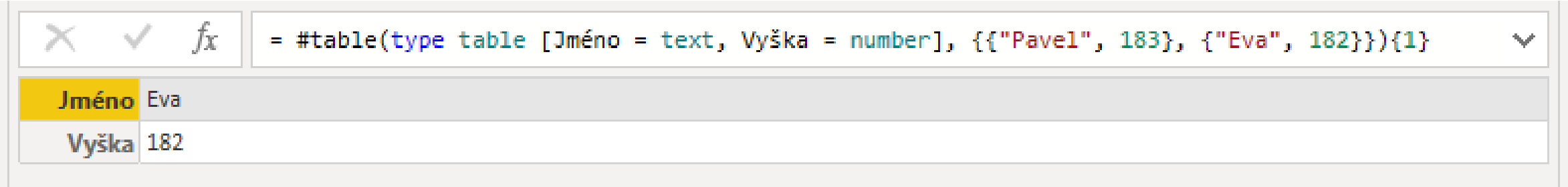
```
let
    Source = #table(type table [Jméno = text, Vyška = number], {{"Pavel", 183}, {"Eva", 182}}),
    Sloupec = Source[Jméno]
in
    Sloupec
```

Práce s tabulkou - řádek

G

- Pokud chci druhý řádek (první je nula)

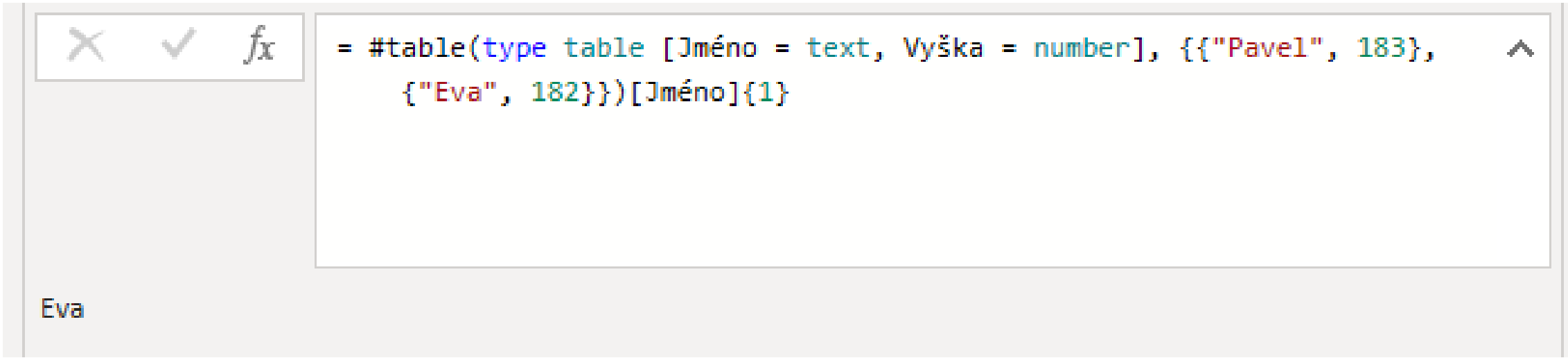
```
= #table(type table [Jméno = text, Vyška =
number], {{"Pavel", 183}, {"Eva", 182}}){1}
```



Práce s tabulkou - buňka

G

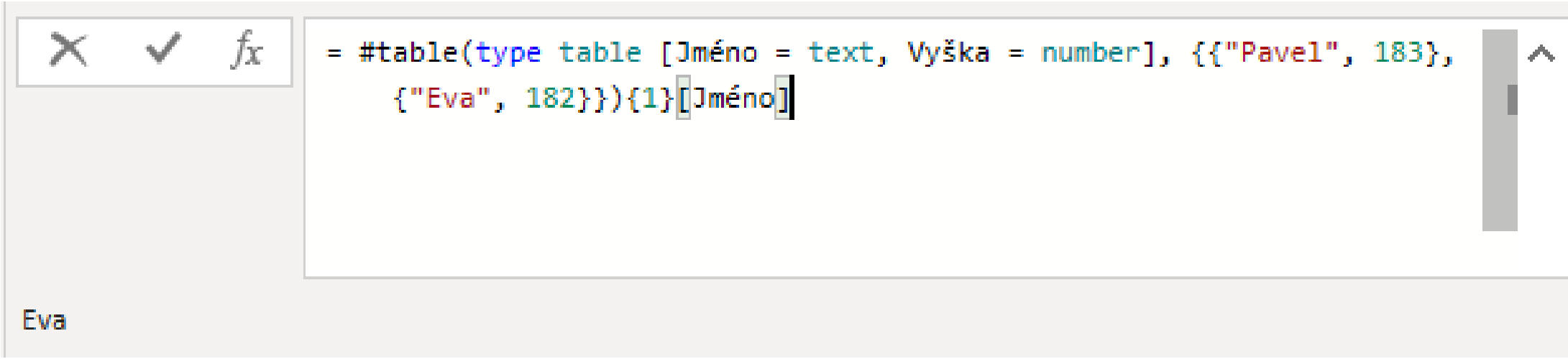
```
= #table(type table [Jméno = text, Vyška = number], {{"Pavel", 183}, {"Eva", 182}})[Jméno]{1}
```



Práce s tabulkou - buňka

G

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



Práce s tabulkou - buňka

G

- Nezáleží zda vyberete:
  - Řádekanásleněsloupce
  - Sloupecanásledněřádek

- Což je logické

Práce s tabulkou – buňka-otazník?

G

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

```
MojeTabulka[Sloupec]?{0}?
```

Tabulky - poznámky

G

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

J

- Soucet

= 1 + 1 //2

- Nebo v M-ku

```
C=1
B=2
A=B+C
```

Porovnávací

J

- **Porovnani**

```
= 3 > 2                    //true
```

- Použití v logických funkcích

Spojovací

J

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

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

Funkce

J

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

- Viz samostatná kapitola

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

- Viz dále

Poznámky

Let In - konstrukce

K

- 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
    #"Promena mezera" = {x,Datum, Funkce} ,
    /* víceřadková
        poznámka */
    vysledek = #"Promena mezera"
in
    vysledek
```

Let in

K

- Let in
- Je vlastně záznam ;)
- jen [] se použije konsturuce **let in**

- Aby kód nevypadal tak strašidelně ;)

Let in - náhrada Let in >> []

K

```
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
`/* víceřadková 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

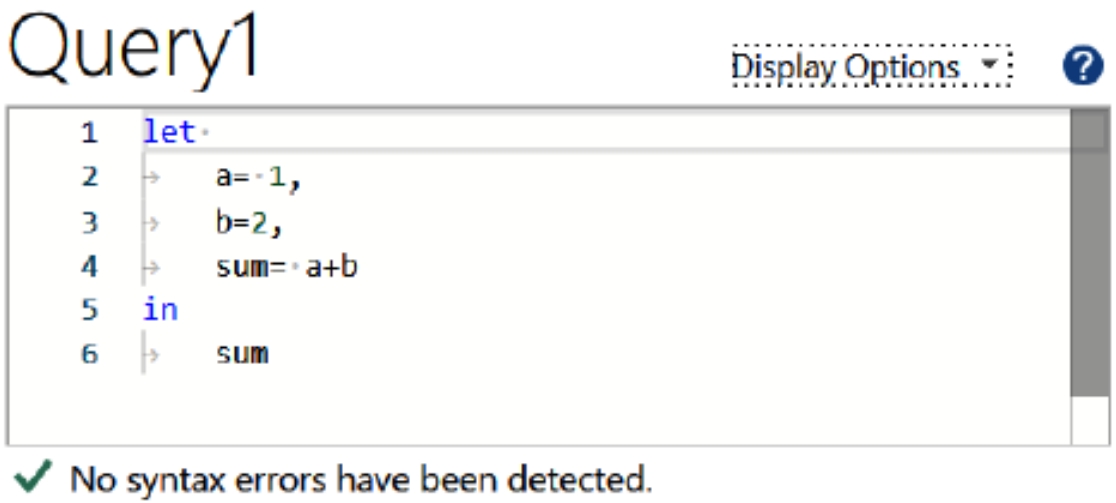
K

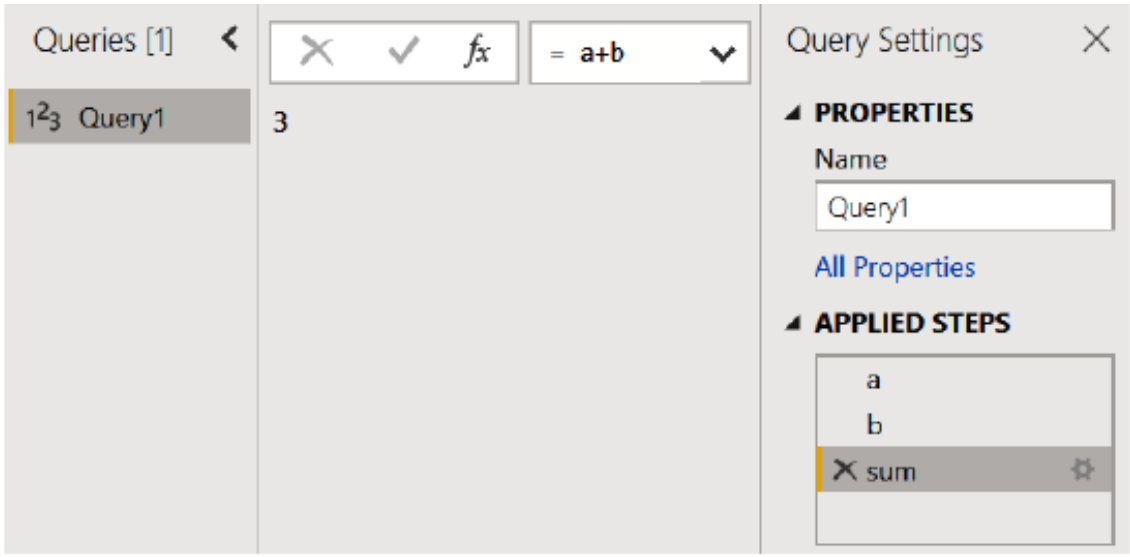
```
let
    Source = ""
in
    Source
```

M-ko vs Editor Power Query

K

- // poznámky





Pořadí kroků

K

```
let
    A = 1,
    B = 2,
    C = A + B
in
    C
```

```
let
    C = A + B,
    A = 1,
    B = 2
in
    C
```

Názvy kroků

K

```
let
   #"A a" = 1,
   #„B b" = 2,
   #„C c" = #"A a" + #„B b"
in
  #„C c"
```

Nepotřebné kroky

K

```
let
    A = 1,
    B = 2,
    X = 1111 + 5555,
    C = A + B
in
    C
```

Poznámky

K

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

Poznámka k let-in

Each – pro každý

L

- Slouží ke snadnému vytváření funkcí
- Proveď 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

L

Ukázka 1

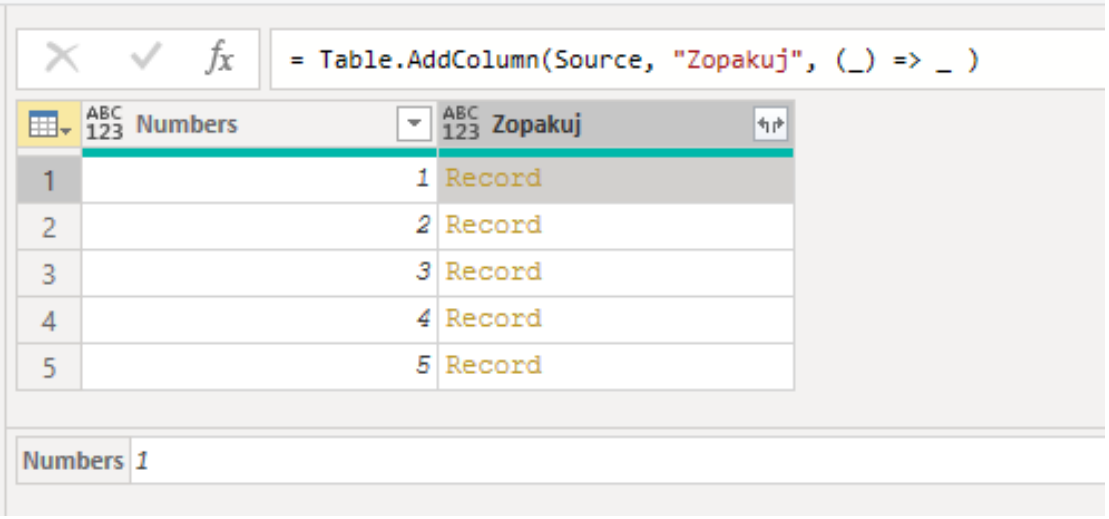
```
let
 Zdroj = #table({"Numbers"}, {{1}, {2}, {3}, {4}, {5}}),
 NovySloupec = Table.AddColumn(Zdroj, "Double", each 2*[Numbers])
in
 NovySloupec
```

Ukázka 2

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

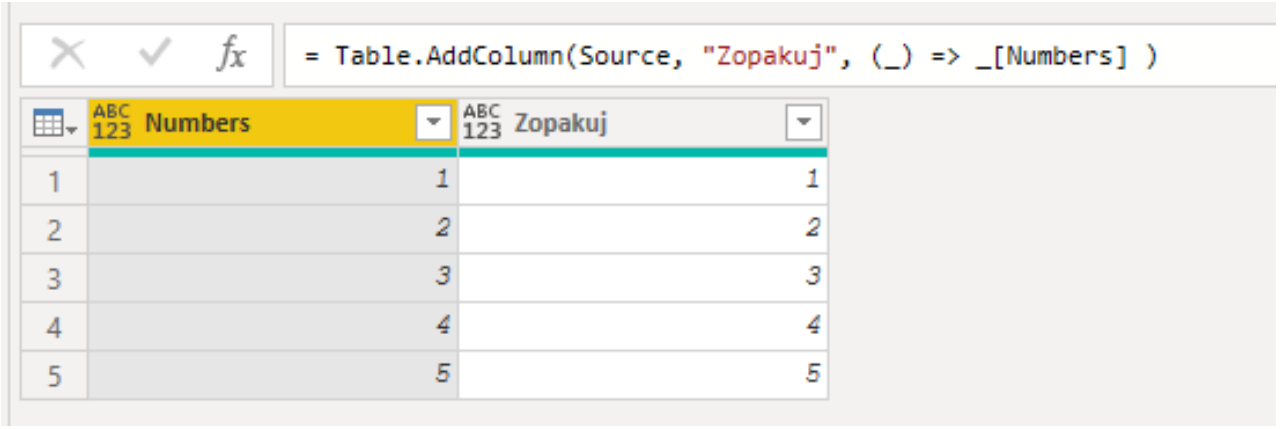
Each - zopakuj

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



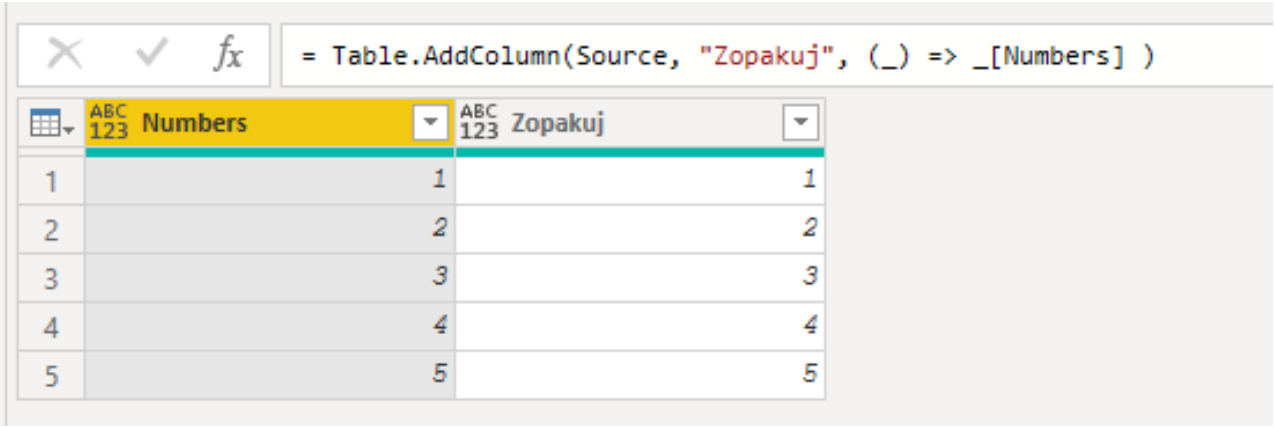
Each - zopakuj

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



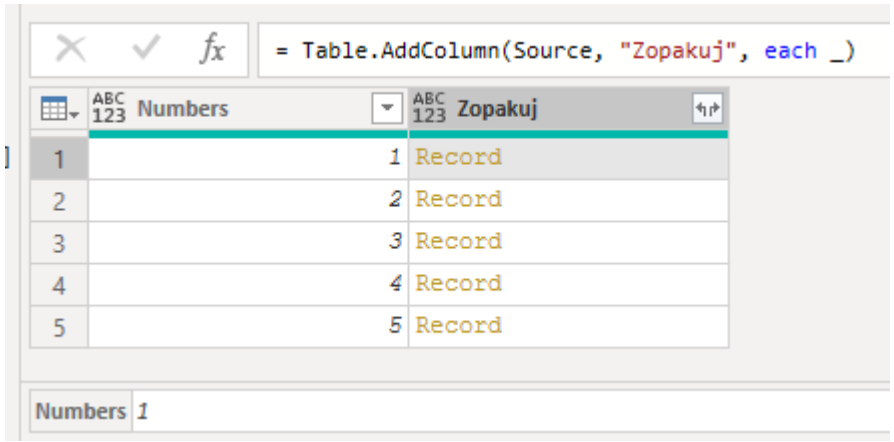
Each - zopakuj

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



Each - zopakuj

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



Each - filtry

L

```
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

L

Podmínky - if then else

M

- V Excel KDYŽ (IF)

- Pokud splněno proveď A jinak B

Podmínky - If then else

M

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

M

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

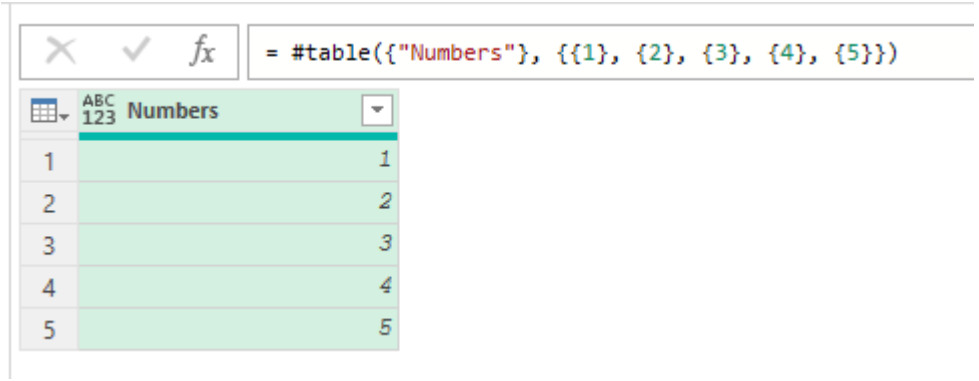
- Ukázky
  - `[Number] > 8 and [Number] < 25`
  - `[Number]< 15 or [Number] > 25`
  - `not ([Number] < 25)`

Data

M

- Mějme tabulku

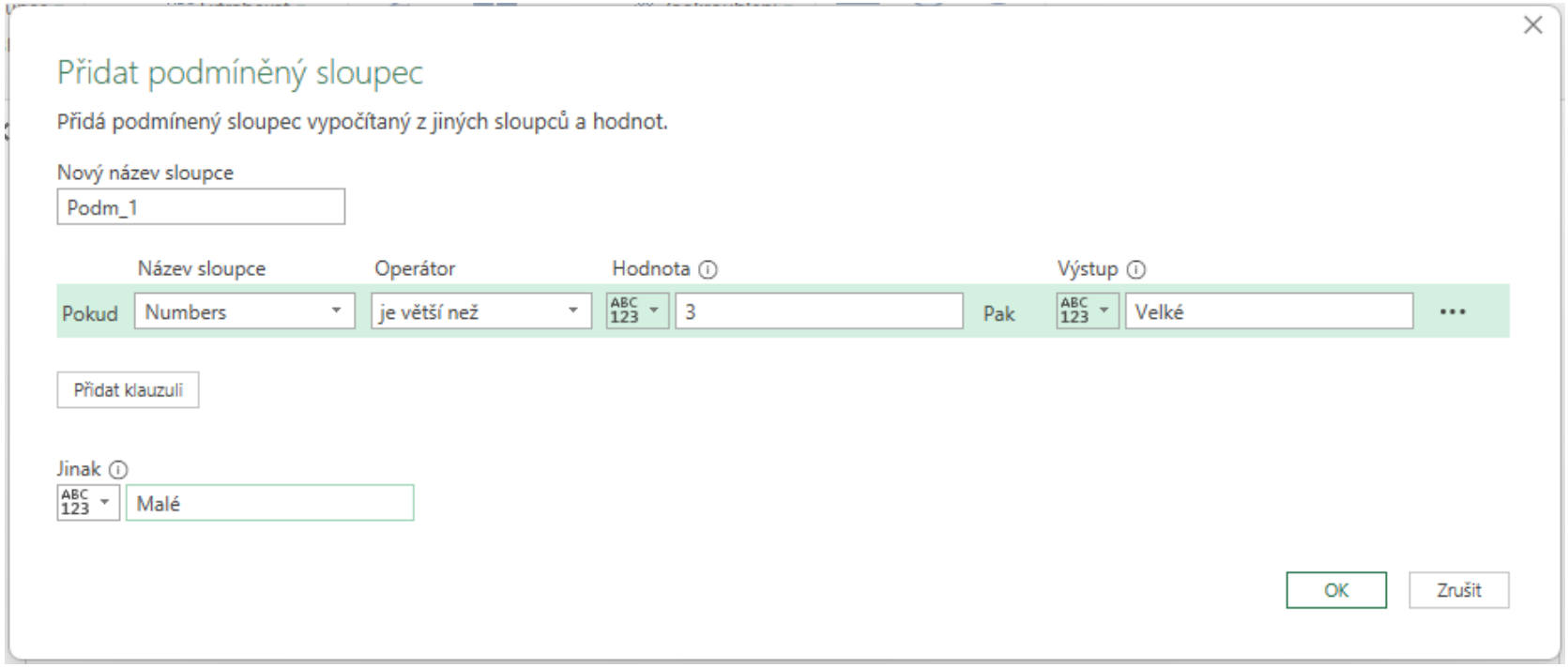
```
let
 Source = #table({"Numbers"}, {{1}, {2}, {3}, {4}, {5}}),
in
 Source
```



Podmínky - if then else –ručně-opakování

M

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



Výsledek

M

```
let
 Source = #table({"Numbers"}, {{1}, {2}, {3}, {4}, {5}}),
    #"Podmíněný sloupec je přidaný" = Table.AddColumn(Source,
"Podm_1", each if [Numbers] > 3 then "Velké" else "Malé")
in
 #"Podmíněný sloupec je přidaný„
```

Nezměnit název #"Podmíněný sloupec je přidaný"

Podmínky - If then else – základ

M

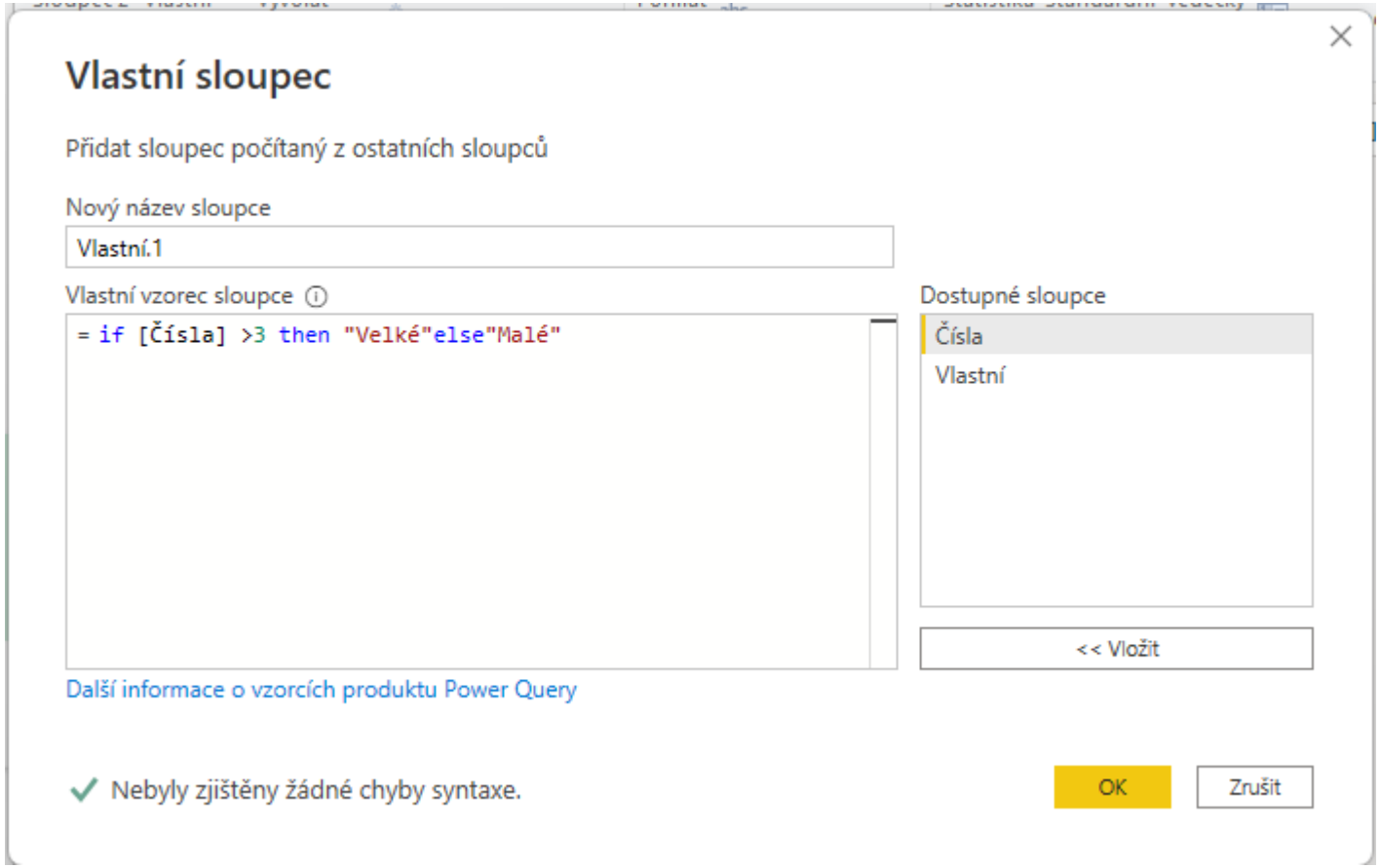
```
if [Čísla] > 3 then "Velké" else "Malé"
```

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

Podmínky - If then else

M

- Vlastní sloupec



Podmínky - If then - více podmínekvnoření

M

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

Podmínky - If then else- or

M

```
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*

M

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

Neexistující sloupec

M

```
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**

O

- Ošetření chyb v M-ku
- Konstrukce

```
try … otherwise
```

- V případě chyby zobrazí požadovanou hodnotu

Příklad

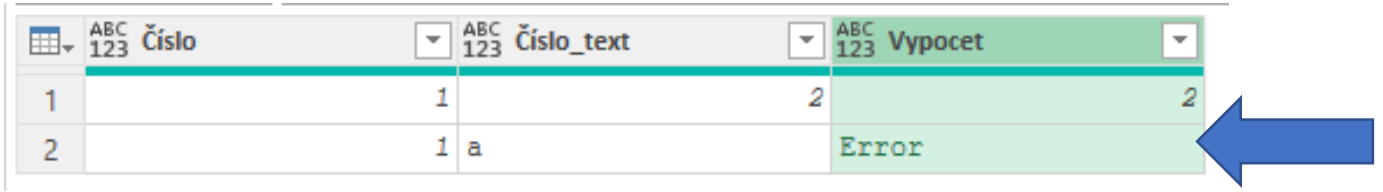
O

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

Potřebuji pronásobit dva sloupce

O

```
let
    Zdroj = #table({"Číslo", "Číslo_text"},{{1,2},{1,"a"}}),
    Sloupec = Table.AddColumn(Zdroj, "Vypocet", each [Číslo]*[Číslo_text])
in
    Sloupec
```

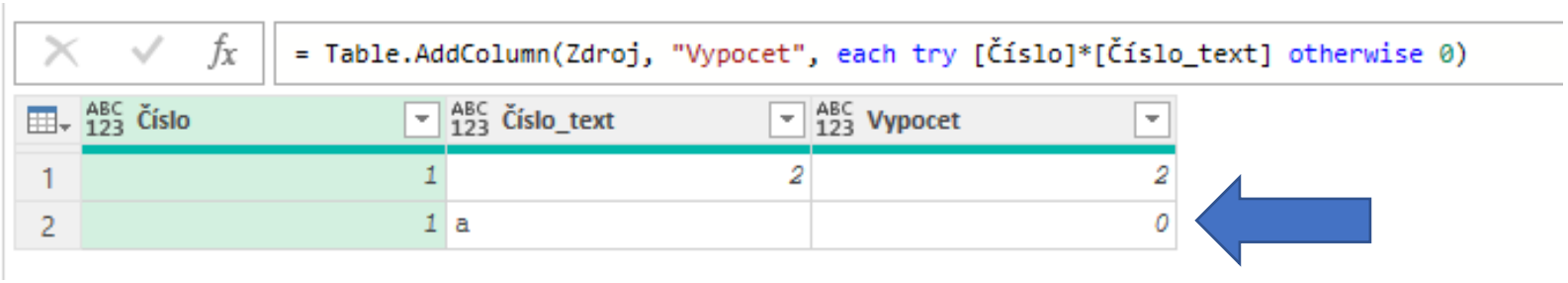


	Číslo	Číslo_text	Vypocet
1	1	2	2
2	1	a	Error

Potřebuji pronásobit dva sloupce

O

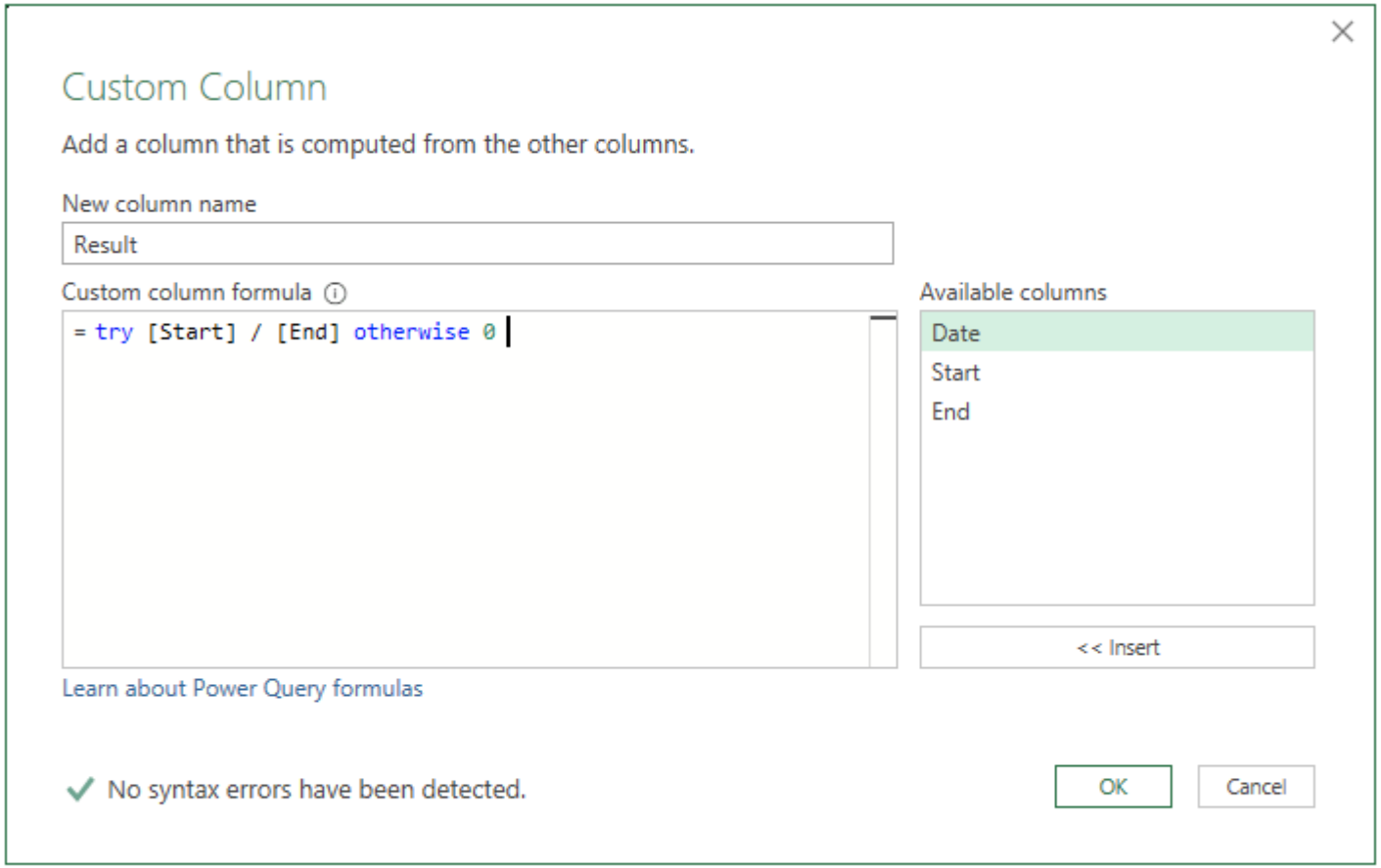
```
let
    Zdroj = #table({"Číslo", "Číslo_text"},{{1,2},{1,"a"}}),
    Sloupec = Table.AddColumn(Zdroj, "Vypocet", each try [Číslo]*[Číslo_text] otherwise 0)
in
    Sloupec
```



Ošetření chyb - Try otherwise

O

- Dělení nulou
- try otherwise



Načítaní 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	Start	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

O

- Otazník
  - {}?
  - []?

Poznámky

O

Funkce v Mku - kategorie

Q

- 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

Q

- „Nakliknutí“ vs ruční zápis

- Ručně výhody
  - Lze využít skryté možností funkce

Jak na funkce - syntaxe prakticky

Q

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

Vytvořím tabulku slov

Q

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

	Slova
1	Stokoruna
2	sto
3	Stovka
4	Tisíc

Potřebuji slova se „sto“

Q

- 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?

Q

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

Co další data ve sloupci?

Q

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

Google

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

Q

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

Vlastní funkce

Q

- Bez Google ani ránu

Vlastní funkce

Q

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

Vlastní funkce

Q

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

Odkazování se na proměnné

Q

- Viz List, Record, Tabulka ……

Poznámky

Q

Tip Google a nápověda…

R

- Název listu vs pořadové číslo listu v Excel

Načítání Excel

R

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

(1) Název vs pořadové číslo listu v Excel

R

```
    // 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

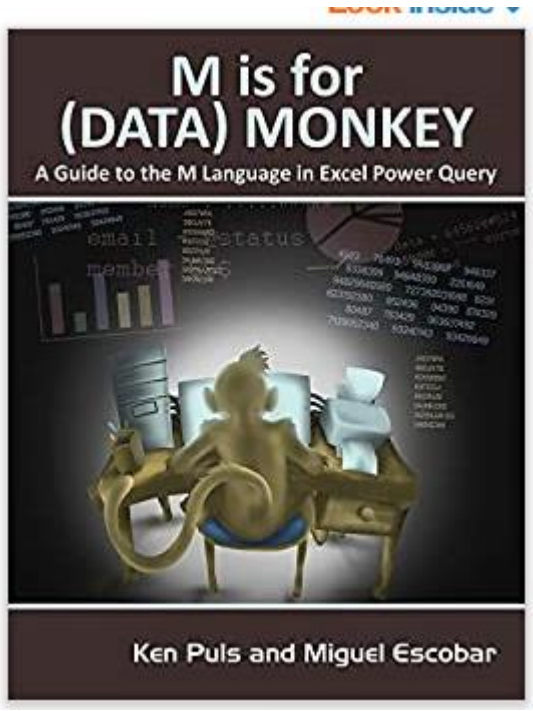
Z

- Videa
- Knihy

Knihy

Z

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



Weby o Mku

Z

- **Imke Feldmann**
  - https://www.linkedin.com/in/imkefeldmann?originalSubdomain=de
  - https://www.thebiccountant.com/
- **Chris Webb**
  - https://blog.crossjoin.co.uk/

Weby o Mku

Z

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

Z

- **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