Geschwindigkeit in Power Query

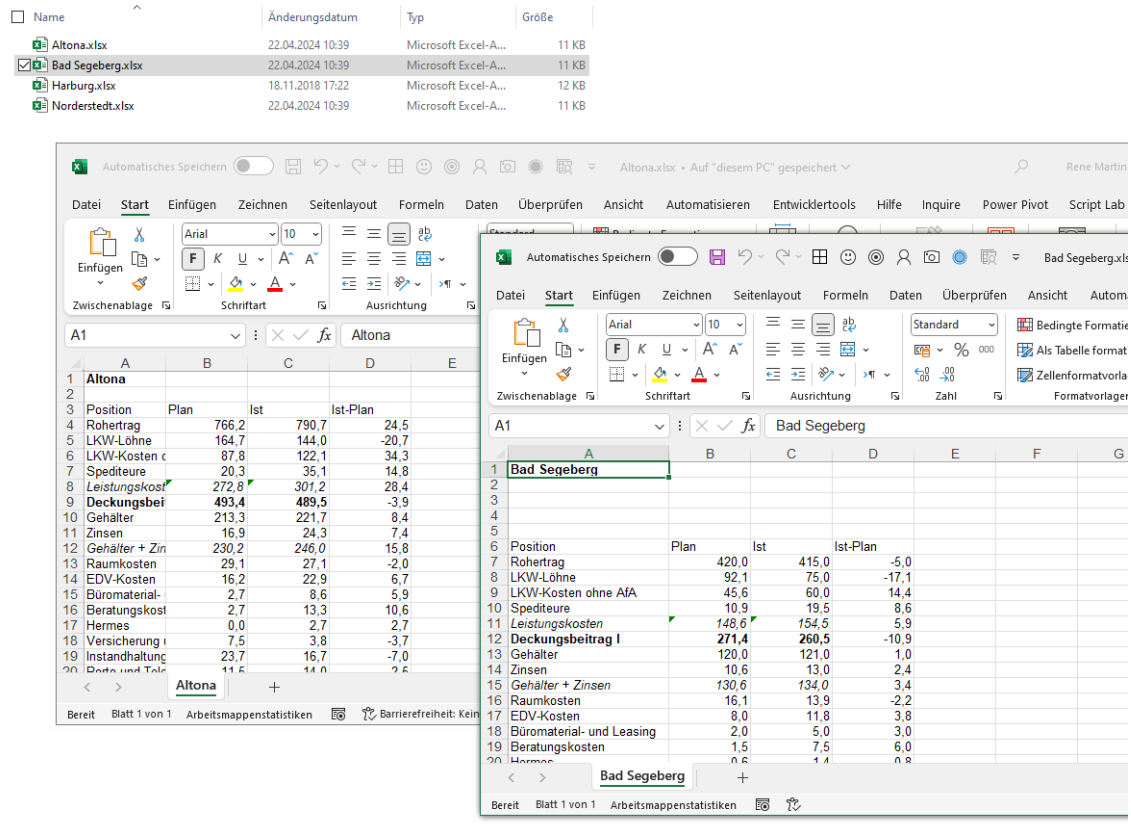
Excelstammtisch vom 22.04.2024

1. Chandeep Chhabra

Auf dem Blog von Chandeep finden sich interessante Beispiele zu Power Query (aber auch zu Excel und DAX). Man findet seinen Blog unter

https://goodly.co.in/

Beispielsweise folgendes Problem:



In einem Ordner befinden sich mehrere Dateien, die jeweils einen unterschiedlichen Header haben. Das heißt: eine unterschiedliche Anzahl leere Zeilen vor den Daten, die ausgewertet werden sollen. Um die Daten aus allen Dateien zusammen zufassen wäre ich wie folgt vorgegangen:

1.1. Lösung I

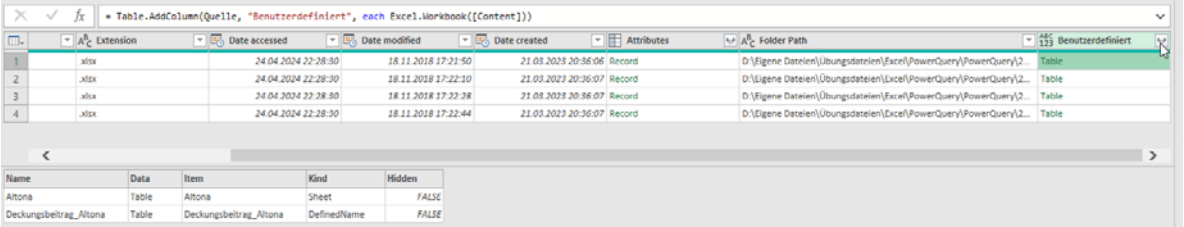
Im ersten Schritt wird auf den Ordner zugegriffen:

```
= Folder.Files("D:\Pfad\25_Plan_IstAusZweiDateien")
```

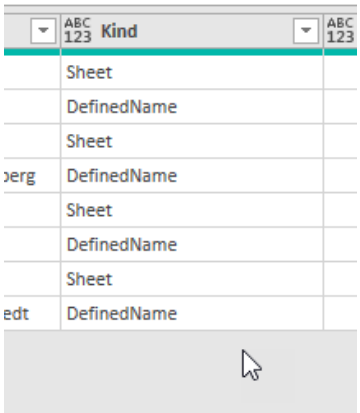
In einer neuen Spalte wird mit Excel.Workbook der (Datei-)Inhalt aus der Spalte Content herausgelöst:

```
= Table.AddColumn(Quelle, "Benutzerdefiniert",
    each Excel.Workbook([Content]))
```

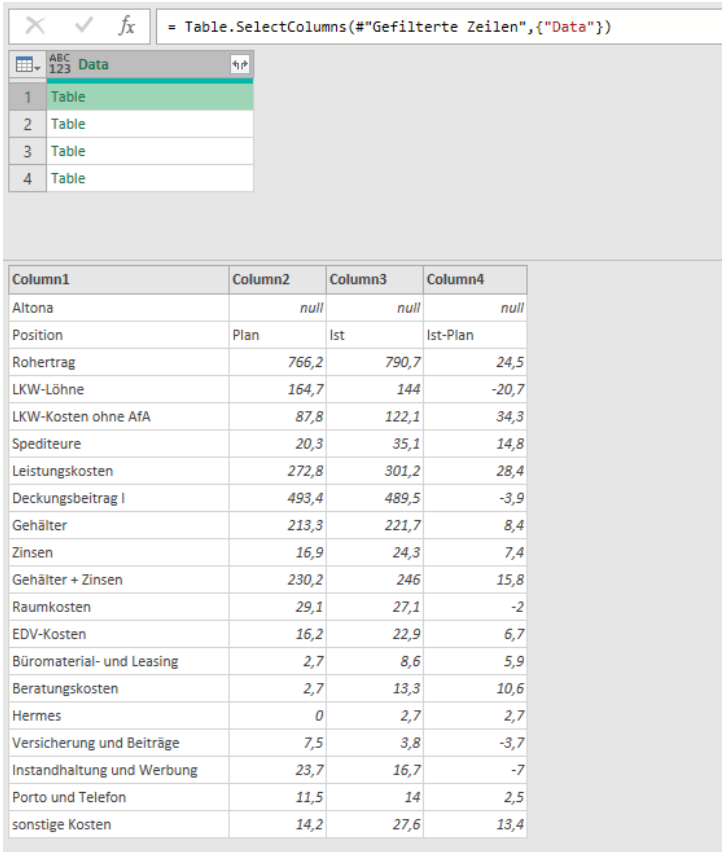
Die „Tabelle“ wird „geöffnet“:



Da sich in den Dateien auch Namen befinden wird aus der Spalte „Kind“ das „Sheet“ herausgefiltert:



Anschließend kann man die anderen Spalten – außer der Spalte „Data“ löschen:



Und diese Spalte „öffnen“:

```
= Table.ExpandTableColumn(#"Andere entfernte Spalten", "Data", {"Column1", "Column2", "Column3", "Column4"}, {"Column1", "Column2", "Column3", "Column4"})
```

	Column1	Column2	Column3	Column4
1	Altona	null	null	null
2	Position	Plan	Ist	Ist-Plan
3	Rohertrag	766,2	790,7	24,5
4	LKW-Löhne	164,7	144	-20,7
5	LKW-Kosten ohne AfA	87,8	122,1	34,3
6	Spediteure	20,3	35,1	14,8
7	Leistungskosten	272,8	301,2	28,4
8	Deckungsbeitrag I	493,4	489,5	-3,9
9	Gehälter	213,3	221,7	8,4
10	Zinsen	16,9	24,3	7,4
11	Gehälter + Zinsen	230,2	246	15,8
12	Raumkosten	29,1	27,1	-2
13	EDV-Kosten	16,2	22,9	6,7
14	Büromaterial- und Leasing	2,7	8,6	5,9
15	Beratungskosten	2,7	13,3	10,6
16	Hermes	0	2,7	2,7
17	Versicherung und Beiträge	7,5	3,8	-3,7
18	Instandhaltung und Werbung	23,7	16,7	-7
19	Porto und Telefon	11,5	14	2,5
20	sonstige Kosten	14,2	27,6	13,4
21	Bereitschaftskosten I	107,6	136,7	29,1
22	Deckungsbeitrag II	155,6	106,8	-48,8
23	AfA	108	127,7	19,7
24	LKW-Leasing	50,7	48,3	-2,4
25	Gewerbesteuer	0,7	0,6	-0,1
26	Bereitschaftskosten II	159,4	176,6	17,2
27	Deckungsbeitrag III	-3,8	-69,8	-66
28	Bad Segeberg	null	null	null
29	Position	Plan	Ist	Ist-Plan
30	Rohertrag	420	415	-5
31	LKW-Löhne	92,1	75	-17,1
32	LKW-Kosten ohne AfA	45,6	60	14,4
33	Spediteure	10,9	19,5	8,6
34	Leistungskosten	148,6	154,5	5,9
35	Deckungsbeitrag I	271,4	260,5	-10,9
36	Gehälter	120	121	1
37	Zinsen	10,6	13	2,4
38	Gehälter + Zinsen	130,6	134	3,4
39	Raumkosten	16,1	13,9	-2,2
40	EDV-Kosten	8	11,8	3,8
41	Büromaterial- und Leasing	2	5	3
42	Beratungskosten	1,5	7,5	6
43	Hermes	0,6	1,4	0,8

Mit einem geschickten Filter können die überflüssigen Zeilen gelöscht werden:

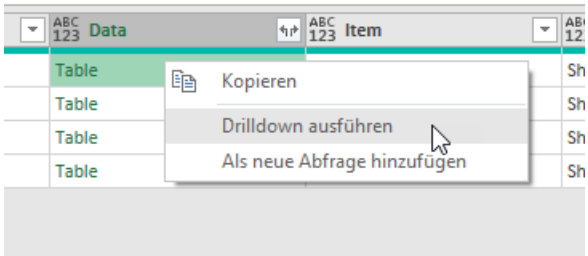
```
= Table.SelectRows(#"Erweiterte Data", each ([Column2] <> null) and ([Column1] <> "Position"))
```

	Column1	Column2	Column3	Column4
1	Rohertrag	766,2	790,7	24,5
2	LKW-Löhne	164,7	144	-20,7
3	LKW-Kosten ohne AfA	87,8	122,1	34,3
4	Spediteure	20,3	35,1	14,8
5	Leistungskosten	272,8	301,2	28,4
6	Deckungsbeitrag I	493,4	489,5	-3,9
7	Gehälter	213,3	221,7	8,4
8	Zinsen	16,9	24,3	7,4
9	Gehälter + Zinsen	230,2	246	15,8
10	Raumkosten	29,1	27,1	-2

1.2. Lösung II

Chandeep geht einen anderen Weg.

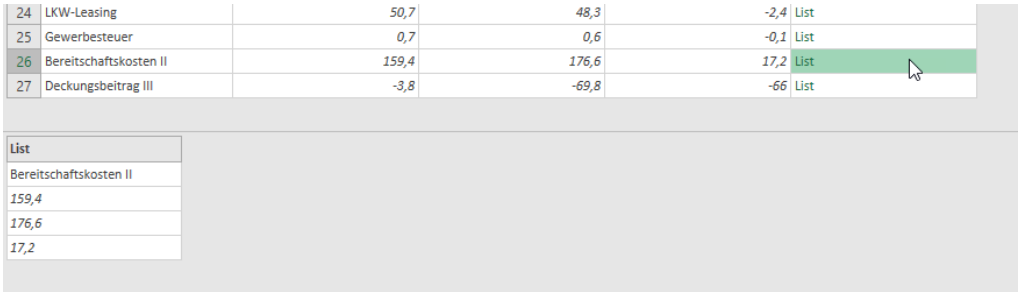
Man kann auf eine Tabelle einen Drilldown ausführen und sich so eine Tabelle genauer „ansehen“:



Fügt man eine neue Spalte hinzu mit dem Befehl

```
Record.ToList(_)
```

So erhält man eine Liste, welche sämtliche Zellinformationen der aktuellen Zeile enthält:



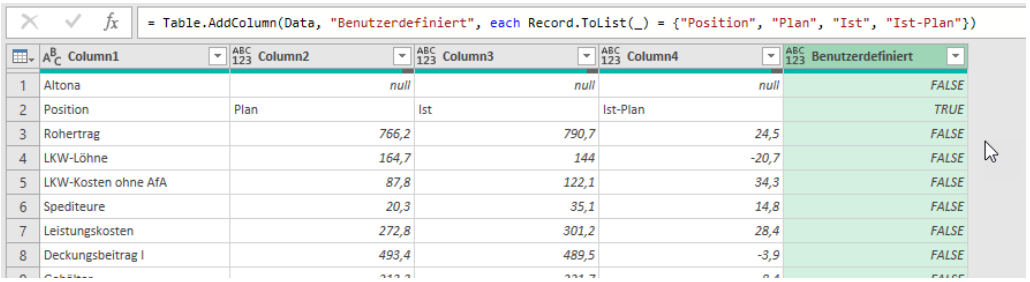
24	LKW-Leasing	50,7	48,3	-2,4	List
25	Gewerbesteuer	0,7	0,6	-0,1	List
26	Bereitschaftskosten II	159,4	176,6	17,2	List
27	Deckungsbeitrag III	-3,8	-69,8	-66	List

List
Bereitschaftskosten II
159,4
176,6
17,2

Erweitert man den Befehl auf:

```
Record.ToList(_) = {"Position", "Plan", "Ist", "Ist-Plan"}
```

Erhält man True oder False – je nachdem ob diese vier Texte in der Zeile auftauchen:



```
= Table.AddColumn(Data, "Benutzerdefiniert", each Record.ToList(_) = {"Position", "Plan", "Ist", "Ist-Plan"})
```

	Column1	Column2	Column3	Column4	Benutzerdefiniert
1	Altona	null	null	null	FALSE
2	Position	Plan	Ist	Ist-Plan	TRUE
3	Rohertrag	766,2	790,7	24,5	FALSE
4	LKW-Löhne	164,7	144	-20,7	FALSE
5	LKW-Kosten ohne AfA	87,8	122,1	34,3	FALSE
6	Spediteure	20,3	35,1	14,8	FALSE
7	Leistungskosten	272,8	301,2	28,4	FALSE
8	Deckungsbeitrag I	493,4	489,5	-3,9	FALSE

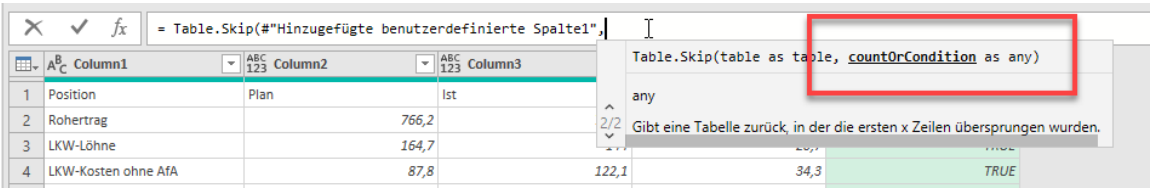
Da alle leeren Zeilen (bis zur ersten False-Zelle) gelöscht werden sollen, muss die Bedingung umgekehrt werden:

```
= Table.AddColumn(Data, "Benutzerdefiniert",
    each Record.ToList(_) <>
    {"Position", "Plan", "Ist", "Ist-Plan"}
)
```

Der Befehl „erste Zeile entfernen“ lautet:

```
= Table.Skip(#"Hinzugefügte benutzerdefinierte Spalte1",1)
```

Der letzte Parameter kann eine Zahl oder einen Wahrheitswert darstellen:



```
= Table.Skip(#"Hinzugefügte benutzerdefinierte Spalte1",
```

Table.Skip(table as table, countOrCondition as any)

any

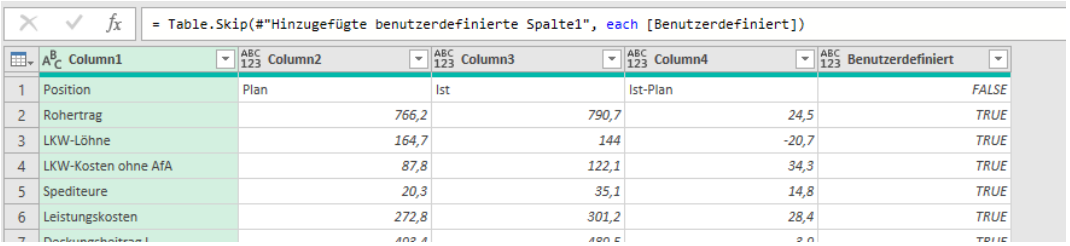
2/2 Gibt eine Tabelle zurück, in der die ersten x Zeilen übersprungen wurden.

	Column1	Column2	Column3
1	Position	Plan	Ist
2	Rohertrag	766,2	
3	LKW-Löhne	164,7	
4	LKW-Kosten ohne AfA	87,8	122,1

Verwendet man dort die Benutzerdefinierte Spalte:

```
= Table.Skip(#"Hinzugefügte benutzerdefinierte Spalte1",
    each [Benutzerdefiniert])
```

Werden alle Zeilen bis zur Überschriftszeile gelöscht:

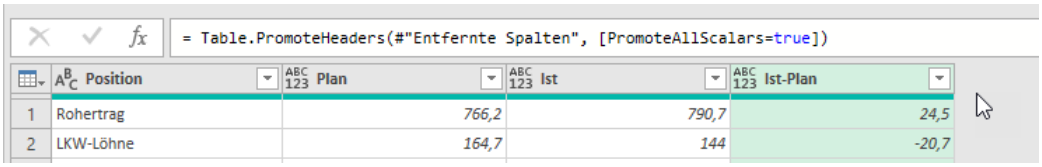


= Table.Skip(#"Hinzugefügte benutzerdefinierte Spalte1", each [Benutzerdefiniert])

	Column1	Column2	Column3	Column4	Benutzerdefiniert
1	Position	Plan	Ist	Ist-Plan	FALSE
2	Rohertrag	766,2	790,7	24,5	TRUE
3	LKW-Löhne	164,7	144	-20,7	TRUE
4	LKW-Kosten ohne AfA	87,8	122,1	34,3	TRUE
5	Spediteure	20,3	35,1	14,8	TRUE
6	Leistungskosten	272,8	301,2	28,4	TRUE

Die Hilfsspalte kann gelöscht werden; die erste Zeile wird zur Überschrift der Tabelle:

```
= Table.PromoteHeaders(#"Entfernte Spalten",
[PromoteAllScalars=true])
```



= Table.PromoteHeaders(#"Entfernte Spalten", [PromoteAllScalars=true])

	Position	Plan	Ist	Ist-Plan
1	Rohertrag	766,2	790,7	24,5
2	LKW-Löhne	164,7	144	-20,7

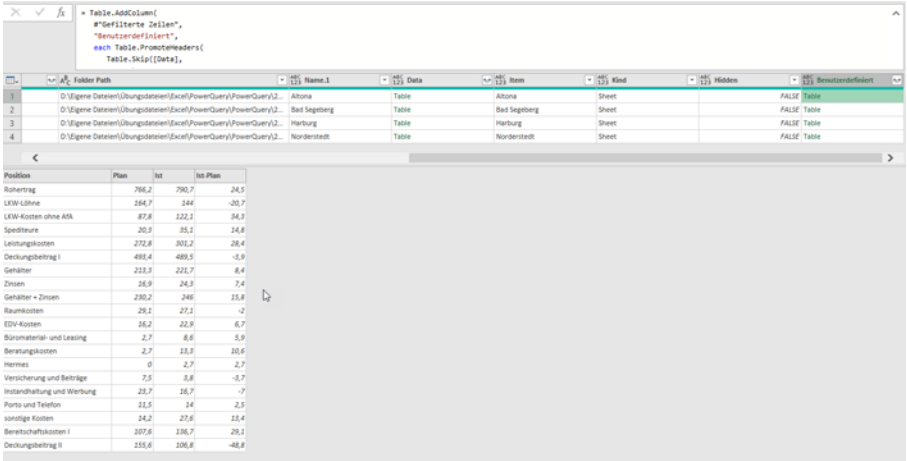
Diese Befehle

- `each Record.ToList(_) <>`
  `{"Position", "Plan", "Ist", "Ist-Plan"}`
- `Table.Skip`
- `Table.PromoteHeaders`

Können zu einem Befehl zusammengefügt werden und auf die ganze Spalte angewendet werden:

```
= Table.AddColumn(
   #"Gefilterte Zeilen",
   "Benutzerdefiniert",
   each Table.PromoteHeaders(
      Table.Skip([Data],
         each
         (Record.ToList(_) <>
         {"Position", "Plan", "Ist", "Ist-Plan"}
         )
      )
   )
)
```

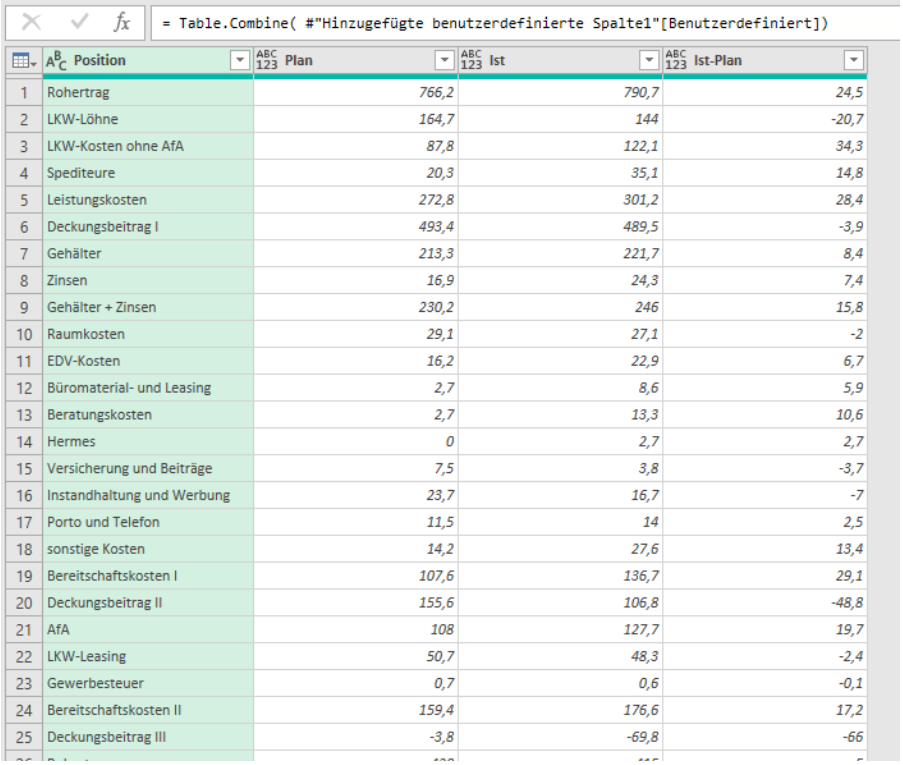
So haben alle Tabellen die korrekte Form:



Und zum Schluss können diese Tabellen mit

`Table.Combine`

Zusammengefasst werden.



= Table.Combine(#"Hinzugefügte benutzerdefinierte Spalte1"[Benutzerdefiniert])

	Position	Plan	Ist	Ist-Plan
1	Rohertrag	766,2	790,7	24,5
2	LKW-Löhne	164,7	144	-20,7
3	LKW-Kosten ohne AfA	87,8	122,1	34,3
4	Spediteure	20,3	35,1	14,8
5	Leistungskosten	272,8	301,2	28,4
6	Deckungsbeitrag I	493,4	489,5	-3,9
7	Gehälter	213,3	221,7	8,4
8	Zinsen	16,9	24,3	7,4
9	Gehälter + Zinsen	230,2	246	15,8
10	Raumkosten	29,1	27,1	-2
11	EDV-Kosten	16,2	22,9	6,7
12	Büromaterial- und Leasing	2,7	8,6	5,9
13	Beratungskosten	2,7	13,3	10,6
14	Hermes	0	2,7	2,7
15	Versicherung und Beiträge	7,5	3,8	-3,7
16	Instandhaltung und Werbung	23,7	16,7	-7
17	Porto und Telefon	11,5	14	2,5
18	sonstige Kosten	14,2	27,6	13,4
19	Bereitschaftskosten I	107,6	136,7	29,1
20	Deckungsbeitrag II	155,6	106,8	-48,8
21	AfA	108	127,7	19,7
22	LKW-Leasing	50,7	48,3	-2,4
23	Gewerbesteuer	0,7	0,6	-0,1
24	Bereitschaftskosten II	159,4	176,6	17,2
25	Deckungsbeitrag III	-3,8	-69,8	-66

1.3. Zusammenfassung

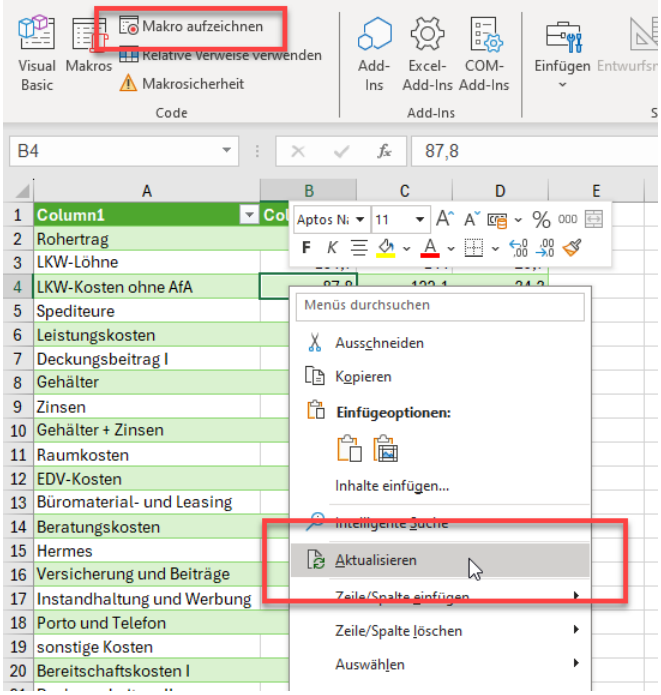
Chandeep bietet sehr clevere Lösungen an, die anders „denken“ und arbeiten als die schrittweise durchgeführten Abfragen, wie wir sie normalerweise verwenden. Zugegeben: der Code ist etwas schwer zu lesen, aber seine Lösung ist brillant und sehr kompakt.

Allerdings verwendet Chandeep auch Befehle wie List.Accumulate. Da stellt sich die Frage, was passiert, wenn man so eine kompakte, clevere Lösung nicht auf vier Dateien mit 20 Zeilen anwendet, sondern – sagen wir mal – auf Dateien mit 300.000 Datensätzen.

1.4. Ein Messwerkzeug für die Geschwindigkeit?

Zeichnet man mit dem Makrorekorder den Befehl des Aktualisierens auf, erhält man folgenden VBA-Befehl:

```
Selection.ListObject.QueryTable.Refresh BackgroundQuery:=False
```



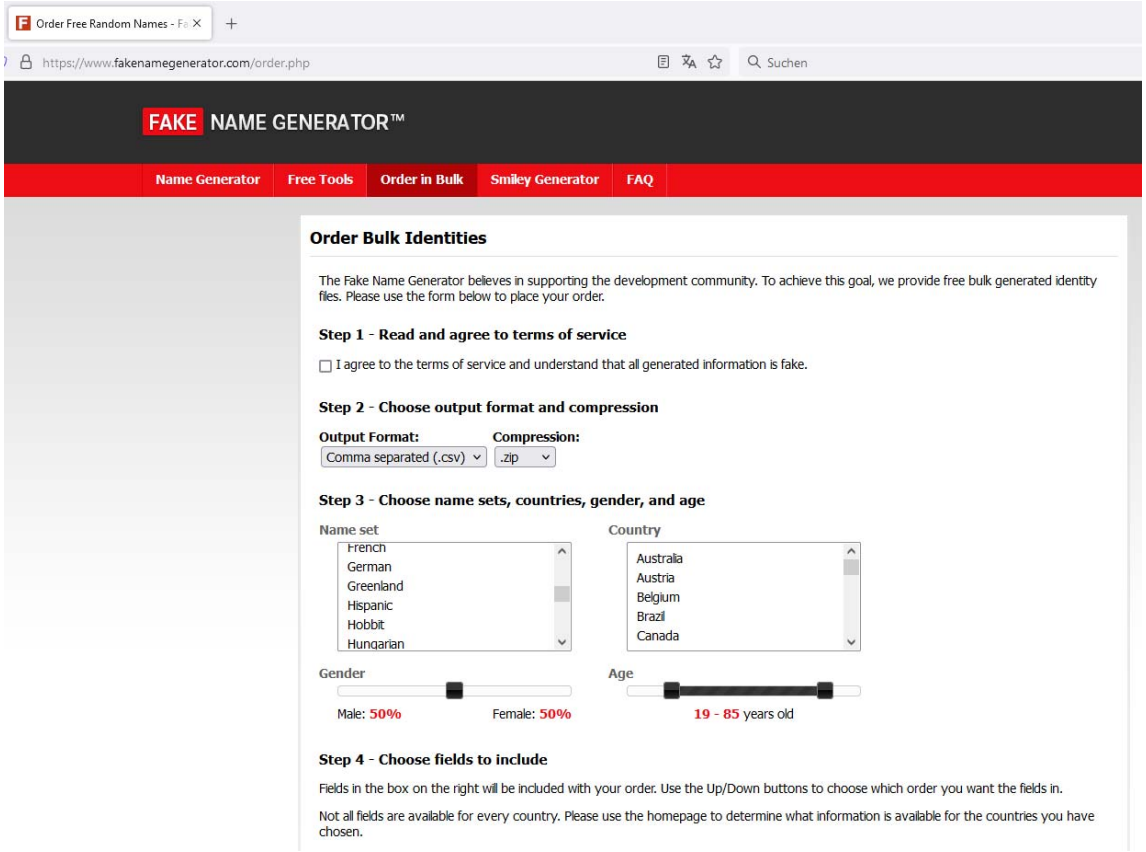
Und damit kann leicht ein kleiner Timer gebaut werden:

```
Sub TabelleAktualisieren()

    Dim dblBeginn As Double
    Dim dblEnde As Double
    Dim i As Integer
    Dim strAusgabe As String

    For i = 1 To ThisWorkbook.Worksheets.Count

        dblBeginn = Now

        ThisWorkbook.Worksheets(i).Range("A4").ListObject.
           QueryTable.Refresh BackgroundQuery:=False

        dblEnde = Now
```

```
        strAusgabe = strAusgabe & vbCr & _
           ThisWorkbook.Worksheets(i).Name & ": " & _
           Format(dblEnde - dblBeginn, "hh:mm:ss.000")
    Next i

    MsgBox strAusgabe

End Sub
```

1.5. Große Datenmengen

Über die Seite https://www.fakenamegenerator.com/ kann man Namen generieren lassen. Sie werden per Mail zugeschickt:



Damit lassen sich (schnell) große Datenmengen erzeugen. Wie beispielsweise hier die Datei, die 300.000 Datensätze (und 41 Spalten) beinhaltet.

1.6. Kein Switch?

Chandeep bemängelt, dass es in Power Query keinen Switch (oder Select Case oder WENNS) Befehl gibt. Also selbst schreiben.

Statt verschachtelter Wenn-Funktionen, die beispielsweise so aussehen könnten:

```
if Date.Year([Geburtsdatum]) > 2020 then "Kategorie A"
else if Date.Year([Geburtsdatum]) > 2000 then "Kategorie B"
else if Date.Year([Geburtsdatum]) > 1980 then "Kategorie C"
else if Date.Year([Geburtsdatum]) > 1960 then "Kategorie D"
else if Date.Year([Geburtsdatum]) > 1940 then "Kategorie E"
else if Date.Year([Geburtsdatum]) > 1920 then "Kategorie F"
else "Kategorie X"
```

verwendet Chandeep eine an Switch angelehnte selbst geschriebene Funktion, die auch ausgelagert werden kann:

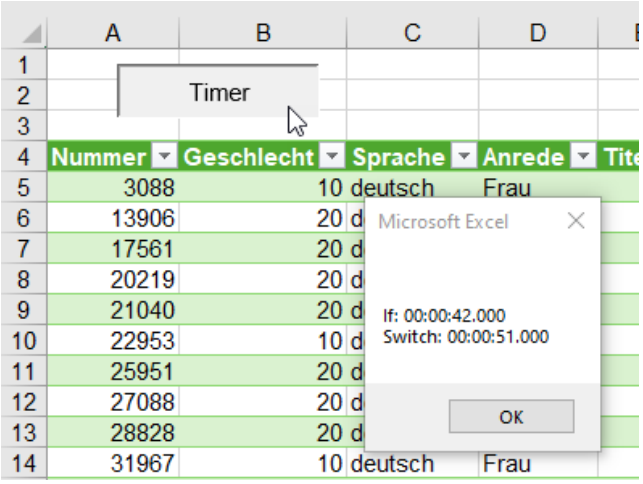
```
let

Bedingungen = {
    Date.Year([Geburtsdatum]) > 2020,
    Date.Year([Geburtsdatum]) > 2000 and
       Date.Year([Geburtsdatum]) <= 2020,
    Date.Year([Geburtsdatum]) > 1980 and
       Date.Year([Geburtsdatum]) <= 2000,
    Date.Year([Geburtsdatum]) > 1960 and
       Date.Year([Geburtsdatum]) <= 1980,
    Date.Year([Geburtsdatum]) > 1940 and
       Date.Year([Geburtsdatum]) <= 1960,
    Date.Year([Geburtsdatum]) > 1920 and
       Date.Year([Geburtsdatum]) <= 1940,
    Date.Year([Geburtsdatum]) <= 1920

},

Ergebnisse = {"Kategorie A", "Kategorie B", "Kategorie C",
"Kategorie D", "Kategorie E", "Kategorie F", "Kategorie X"}

in

Ergebnisse{List.PositionOf(Bedingungen, true)}
```

Das Ergebnis sieht folgendermaßen aus:

Jahrgang
Kategorie D
Kategorie F
Kategorie F
Kategorie F
Kategorie F
Kategorie F
Kategorie C
Kategorie C
Kategorie E
Kategorie E
Kategorie C
Kategorie C
Kategorie C
Kategorie C
Kategorie E
Kategorie D
Kategorie C
Kategorie C
Kategorie D
Kategorie F
Kategorie F
Kategorie F
Kategorie D
Kategorie C

1.7. Wer ist schneller?

Und nun gilt es ein Wettrennen zu starten: Wer ist schneller – die selbstgebaute Funktion oder die verschachtelten If-Anweisungen?



Die Antwort: Mehrere IFs sind schneller als eine Switch-Funktion!