

1	Alles Oberfläche	15
1.1	Der Slash	16
1.2	Die 1904-Zeitrechnung	17
1.2.1	Der Trick mit der Negativzeit	18
1.3	Der Z1S1-Bezug	19
1.3.1	Wechsel auf A1	20
1.4	Rechnet Excel falsch?	20
1.4.1	Das Problem mit Gleitkommazahlen	21
1.5	Bei der Billiarde ist Schluss	21
1.6	Was ist denn eine Makrovorlage?	21
1.7	Die verlorene Datenbank	23
1.7.1	Richtig markieren	23
1.7.2	Der Trick mit der Datenbank	23
1.7.3	Tabelle geht immer	24
1.7.4	... und eine Maske gibt es auch	24
1.8	Der Name der Rose	25
1.8.1	Alternative gefällig?	27
1.9	Seltsame Namen	27
1.9.1	Bereiche	27
1.9.2	Tabellenblätter	28
1.10	Startbildschirm abschalten	29
1.11	Von Dateien und Dateiformaten	29
1.11.1	Dateien richtig öffnen	29
1.11.2	Die Startvorlage	30
1.11.3	XLSX oder XLSB	30
1.11.4	XSLB – besser als XLSM?	32
1.11.5	Blattschutz in XML-Datei entfernen	32
2	Schneller, höher, weiter!	33
2.1	GUI überladen	34
2.2	Kontextspezifische Kontextmenüs	34
2.3	Weg mit der Maus	35
2.4	Zeitkiller ade	35
2.5	Top-11-Shortcuts	36
2.6	Funktionstasten? Aber sicher!	37

3	Von der Liste zur Tabelle	43
3.1	Listen in Tabellen umwandeln	45
3.2	Von der Wichtigkeit der Tabelle	45
3.3	Fünf Gründe, warum Tabellen besser sind als Listen	46
3.3.1	Tabellen sind dynamisch	46
3.3.2	Diagramme auf Tabellenbasis sind auch dynamisch	47
3.3.3	Externe Daten sind immer Tabellen	47
3.4	Tabellen und absolute Bezüge	48
3.5	Strukturierte Verweise mit variablen Bezeichnern	50
3.6	Die Tabelleninfo im Schnellzugriff	52
3.6.1	Datenschnitte für die Tabelle	53
4	Bereichsnamen und dynamische Bereiche	55
4.1	Wozu Bereichsnamen?	56
4.1.1	Namensregeln	56
4.1.2	Namen erstellen	57
4.1.3	Name und Präfix	60
4.1.4	Vorsicht! Bezüge immer absolut setzen!	61
4.2	Namensbezüge	61
4.3	Der Druckbereich	62
4.3.1	Ein mehrdimensionaler Druckbereich	63
4.3.2	Dynamische Druckbereiche	64
4.4	Die Bereichsnamenliste	65
4.5	Beispiel Break-even: Gewinnschwellenanalyse	65
4.6	Dynamische Bereichsnamen	68
4.6.1	Die dynamische Liste	69
4.6.2	Dynamische Bereiche in Tabellen	71
4.6.3	Bereichsnamen aus Tabellenspalten	72
4.6.4	Dynamische Bereiche in Formularelementen	73
4.6.5	Dynamische Bereichsnamen in Diagrammen	75
4.6.6	Dynamische Planung mit Monats- und Abschnittsauswahl	78
4.7	Matrix spiegeln mit Bereichsnamen	81
4.7.1	Beispiel: Sport und Umsatz	82
4.8	VBA: Bereichsnamen ausblenden	84

5	Das Kreuz mit den Formaten	85
5.1	Goldene Regeln für die Tabellenformatierung	86
5.1.1	Ein Unternehmensdesign	88
5.1.2	Der Formatpinsel	88
5.2	Minuszeichen rechts von der Zahl	89
5.3	Benutzerdefinierte Zahlenformate	90
5.3.1	Äpfel und Birnen	91
5.3.2	Nullunterdrückung	92
5.3.3	Zahlenformate für Datumswerte	92
5.3.4	Dezimalzahlen am Komma ausrichten	93
5.4	Bedingte Formatierung	94
5.4.1	Fehlerzellen	95
5.4.2	Jede zweite oder n-te Zeile	95
5.4.3	Suchen mit Wildcards	95
5.4.4	Urlaubsplan	96
6	Mit Funktionen zaubern	105
6.1	Funktionen-Allerlei	106
6.1.1	Datentypbeschreibung und -umwandlung	106
6.1.2	Rechnen mit Bedingungen	107
6.1.3	Anzahl unterschiedlicher Einträge ermitteln	110
6.1.4	Matrizen spiegeln mit WAHL()	111
6.1.5	WAHL() kombinieren mit Steuerelementen	112
6.1.6	Zeile und Spalte	115
6.1.7	Listenabgleich	117
6.1.8	Unter keinen Bedingungen – doch!	118
6.1.9	Unterschiedliche Einträge	119
6.1.10	Letzter Zelleintrag	121
6.1.11	Wo bin ich? – Tabellennavigation ohne Kompass	123
6.1.12	Die letzte Zelle	123
6.1.13	Benutzter Bereich	125
6.1.14	Suchkriterium in der Tabelle aufspüren	128
6.2	Textfunktionen	130
6.2.1	Anzahl Wörter	133
6.2.2	Text und Ziffern trennen	133
6.2.3	Anorganische Einzeller	134

6.2.4	Textanalyse: Adressen trennen	136
6.2.5	Textkette mit dynamischen Arrays	138
6.2.6	Texte richtig teilen	138
6.2.7	Plan-Ist-Vergleiche mit Textdiagrammen.	140
6.2.8	Anzahl der Zeichen ermitteln	145
6.3	Formatieren und Runden	146
6.3.1	Praxis: Quartal berechnen mit AUFRUNDEN	147
6.3.2	Gerundete Werte summieren.	148
6.4	Verweisfunktionen.	148
6.4.1	Der Spaltenverweis SVERWEIS()	149
6.4.2	Flexibler als SVERWEIS(): XVERWEIS()	151
6.4.3	Power Query statt SVERWEIS()	154
6.5	Bereichsrückgabefunktionen	158
6.5.1	BEREICH.VERSCHIEBEN()	160
6.5.2	INDIREKT()	162

7 Datum und Zeit **165**

7.1	Kalenderdaten eingeben	166
7.1.1	Das zweistellige Kalenderdatum	167
7.2	Datumsfunktionen.	168
7.2.1	Superfunktion DATEDIF	170
7.2.2	Alter berechnen	171
7.2.3	Datumsdifferenzen berechnen – Tage pro Monat	172
7.2.4	Das Zwillingsdatum	175
7.2.5	Die nächsten Geburtstage	175
7.2.6	Sommerzeit	176
7.2.7	Wochenbeginn oder -ende.	177
7.2.8	Folgender Wochentag	177
7.2.9	Feiertage berechnen	178
7.2.10	Kalender.	184
7.2.11	Einen bestimmten Arbeitstag ermitteln	192
7.2.12	Kegeln am dritten Samstag des Monats.	194
7.2.13	Kalenderwoche	194
7.2.14	Quartal.	195
7.2.15	Schaltjahr	196
7.2.16	Sternkreiszeichen	196
7.2.17	Muttertag	197
7.2.18	Weihnachten.	197

7.3	Zeitfunktionen.....	198
7.3.1	Industriezeit.....	199
7.3.2	Zeitformat über 24 Stunden.....	199

8 Matrix oder Array – die dritte Dimension 201

8.1	Was jetzt – Matrix oder Array?.....	202
8.1.1	Mit Arraykonstanten arbeiten.....	202
8.1.2	Matrix anzeigen und berechnen.....	204
8.2	Die CSE-Formel.....	204
8.3	Dynamische Arrays.....	206
8.3.1	Nur mit Tabellen.....	206
8.3.2	Das Prinzip »Überlauf«.....	206
8.3.3	Beispiel: Monatsreihe auswerten.....	208
8.3.4	Überlauffehler.....	208
8.3.5	Einzelwert und implizite Schnittmenge.....	210
8.3.6	Der #-Operator für dynamische Arrays.....	211
8.3.7	Logische Operatoren in Arrays.....	212
8.4	Arrayfunktionen.....	213
8.4.1	Dynamisch sortieren.....	215
8.4.2	Filtern, aber kalkuliert.....	216
8.4.3	Von zweideutig zu eindeutig.....	217
8.4.4	Zufälle gibt's.....	219
8.4.5	Bevorzugt sequentiell.....	219
8.4.6	Bitte wählen.....	221
8.4.7	Diagramme und dynamische Arrays.....	223
8.4.8	Gute Zeiten für Hochstapler.....	227
8.4.9	Nimm es oder lass es.....	228
8.4.10	Anders pivotieren.....	231

9 Die Funktionsfabrik: LAMBDA() 235

9.1	Das Prinzip.....	236
9.2	Eine neue Funktion entsteht.....	237
9.3	LAMBDA() in Tabellen.....	239
9.4	LAMBDA() für Diagramme.....	241
9.5	LAMBDA-Funktionen.....	243
9.5.1	MAP().....	244
9.5.2	St Andrews.....	245

9.5.3	MATRIXERSTELLEN()	246
9.5.4	NACHSPALTE()	246
9.5.5	NACHZEILE()	247
9.5.6	REDUCE()	248
9.5.7	SCAN()	249
10	Power Query	251
10.1	Nützliche Zaubertricks mit Power Query	252
10.1.1	Der Turbostart	253
10.1.2	Schnellzugriffsleiste	253
10.1.3	Automatische Typumwandlung verhindern	253
10.1.4	Wo ist der Zoom?	255
10.1.5	Spaltentitel sortieren	256
10.1.6	Alle Tabellen im Navigator importieren	256
10.1.7	Abfragen kopieren	257
10.1.8	Spalten mit Leerzeilen addieren	258
10.2	Mit Parametertabellen in Power Query arbeiten	258
10.2.1	Das Beispiel: Lagerbestand	259
10.2.2	Parametertabelle anlegen	260
10.2.3	Parametertabelle in Power Query einbinden	262
10.2.4	Abfragevariablen in M-Code einbinden	263
10.2.5	Daten aktualisieren per Makro	264
10.3	Fuzzy-Übereinstimmung (ungefähre Übereinstimmung)	264
11	Datenüberprüfung	267
11.1	Richtig zuweisen	268
11.2	Kriterien und Meldungen	269
11.3	Auf die Länge kommt es an	270
11.4	Nur Text, nur Zahl	270
11.5	Nix Doppeltes, bitte	270
11.6	Kriterien kombiniert	271
11.7	Zellschutz allgemein	272
11.7.1	Formelzellen sperren	273
11.7.2	Ein einfacher Passwortschutz	274
11.8	Zellen nach Bedarf sperren	275

11.9	Datenüberprüfungslisten	276
11.9.1	Schnell aufgeklappt	276
11.9.2	Basis: eindeutige Werte	277
11.9.3	Eindeutige Werte aus den markierten Daten	278
11.9.4	Dynamische Listen mit gegenseitigem Bezug	278
12	Gut in Form(ularen)	283
12.1	Damit zeichnet es sich leichter	286
12.2	Option und Gruppe	287
12.3	Jeder nur ein Kreuz	289
12.4	Drop-downs	289
12.4.1	Drop-down mit dynamischer Wertausgabe	291
12.5	Ein Fragebogen mit Formularelementen	292
13	Hyperlinks: »Beam uns rauf, Scotty!«	297
13.1	Der »normale« Hyperlink	298
13.2	Menüsteuerung mit Blattnamen	299
13.3	Dateilinks mit HYPERLINK()	301
13.4	Nicht rot werden, Hyperlink!	302
13.5	Dateien an einer bestimmten Stelle öffnen	303
13.6	Zugriff auf viele geschlossene Dateien	304
13.7	Der Such-Link – blitzschnelle Tabellennavigation	305
13.8	Der multiple Power-Link	307
14	Die dritte Dimension	311
14.1	Das Punkt (XYZ)-Diagramm	312
14.1.1	Darstellung eines dreidimensionalen Körpers	313
14.1.2	Darstellung einer dreidimensionalen Funktion	317
14.2	Das dreidimensionale Blasendiagramm	318
15	Finanzmathematische Schmankerl	323
15.1	ZEILE() – Multitalent als Finanzjongleur	324
15.1.1	Folgen und Reihen	324
15.1.2	Renten- und Tilgungsrechnung	325

15.1.3	Interner Zinsfuß	328
15.1.4	Interner Zinsfuß bei aperiodischen Zahlungen.	329
15.2	Die vorschüssige Macke – ein kleiner finanz- mathematischer Exkurs.	330

16 Alles ist Zahl – Anwendungen aus der Zahlentheorie 337

16.1	Teiler und Primzahlen	338
16.1.1	Teiler	338
16.1.2	Primzahlen.	340
16.2	Sehr große Primzahlen	341
16.2.1	Noch größere Primzahlen aufspüren.	343
16.3	Die nächste Primzahl.	344
16.4	Die größten Teiler und das kleinste Vielfache.	348
16.4.1	KGV	349
16.5	Exkurs MMULT: Welche Spalte ist sortiert?	351
16.6	Quersummen	352
16.7	Besonders schräge Quersummen	356

17 IKV – eine Funktion macht Karriere 361

17.1	Lösung von Gleichungen: Galois und Co.	362
17.2	Nominalzins versus Effektivzins.	367
17.3	Interner Zinsfuß.	369
17.3.1	Charmante Annäherungsversuche.	371
17.3.2	Dynamische und schwankende Zahlungen.	374
17.4	IKV legt los.	377
17.4.1	Polynomnullstellen finden	378
17.4.2	Ableitungen und Extrempunkte.	382
17.4.3	Negative Nullstellen.	384
17.5	Lineare und polynomische Regression.	387
17.6	Das Bestimmtheitsmaß.	390
17.7	Multiple Regression.	394
17.8	Multiple Regression versus polynomische Regression	396
17.9	Lösung linearer Gleichungssysteme.	398
17.9.1	Alternative Lösung mit dem Solver.	403

17.10	Von Wachstum und stetiger Verzinsung.....	405
17.11	Sonstige Trendsetter.....	409
17.11.1	Potenzielle Regression.....	409
17.11.2	Logarithmische Regression.....	410
	Stichwortverzeichnis.....	411

KAPITEL 9

Die Funktionsfabrik: LAMBDA()



Die Funktion LAMBDA() als nächste Stufe der Excel-Evolution zu bezeichnen, ist keine Übertreibung. Sie kann nämlich etwas, was keine andere Funktion kann. Funktionen kalkulieren in der Regel Ergebnisse oder produzieren Arrays. LAMBDA() produziert Funktionen. Das war bisher nur mit der Makrosprache VBA möglich und setzte entsprechende Kenntnisse in der Excel-Programmiersprache voraus.

Jetzt hören wir schon die Nörgler im Hintergrund: Über 600 Funktionen an Bord, die meisten kennen maximal 20 davon, reicht das denn nicht langsam? Was bitte kann Excel mit seiner Funktionspalette nicht abdecken?

Zunächst mal: Die Welt wird sich auch ohne LAMBDA() weiterdrehen. Die Funktion ist nur für die Power-User und die Zauberer unter uns, zu denen die Leser dieses Buches ja gehören. LAMBDA() hat zudem einen anderen Problemlösungsansatz als andere Funktionen.

Wer komplexe Funktionskonstrukte entwirft, Funktionen schachtelt und Bedingungen kombiniert, erhöht nicht nur seinen eigenen Arbeits- und Zeitbedarf, sondern auch den seiner Mitmenschen, falls diese mit seinen Kunstwerken arbeiten müssen. Außerdem steigt das Risiko für Kalkulationsfehler proportional zur Länge der Formeln.

Mit LAMBDA() werden Funktionsmonster in verständliche Begriffe verpackt. Der Benutzer muss nicht den ganzen Weg zum Ergebnis verstehen, er bekommt eine Anleitung, welche Parameter er angeben muss, den Rest macht die LAMBDA()-Funktion.

9.1 Das Prinzip

Zunächst die Syntax. In der Funktionsklammer wird eine Reihe von maximal 255 Argumenten angegeben, die entweder Parameter für die Weiterberechnung oder die Berechnung selbst darstellen. Das letzte Argument ist immer die Kalkulation. Sie kann, muss aber nicht, alle zuvor bestimmten Parameter enthalten.

=LAMBDA(Parameter oder Kalkulation; ... Kalkulation)

Hier ein Beispiel: Für das Produkt aus zwei Zahlen genügt die Formel

=zahl1*zahl2

Stehen die beiden Parameter in A1 und A2, sieht das Ganze so aus:

A1: 100

A2: 3

A3: =A1*A2

Die LAMBDA()-Funktion bekommt zwei (willkürlich benannte) Parameter und anschließend die Kalkulation:

```
=LAMBDA(zahl1;zahl2;zahl1*zahl2)
```

Damit die Funktion richtig rechnet, werden ihr in einem weiteren Klammernpaar die Zahlen übermittelt:

```
=LAMBDA(zahl1;zahl2;zahl1*zahl2)(A1;A2)
```

	A	B	C	D	E	F
1	100					
2	3		300			
3						

Abbildung 9.1: LAMBDA() berechnet einen Ausdruck mit zwei Variablen

Legen wir einen Bereichsnamen MULTI an und weisen diesem die LAMBDA()-Formel – ohne die zweite Klammer – als Bezug zu, lässt sich die Kalkulation anschließend so ausführen:

```
=MULTI(A1;A2)
```

9.2 Eine neue Funktion entsteht

In der Praxis sind die Formeln, die LAMBDA() übernimmt, natürlich komplexer. Entwickeln Sie eine weitere Funktion, die Excel so nicht hat: Sie berechnet, ob ein Suchkriterium in einem Textstring enthalten ist. Wir nennen die Funktion ENTHÄLT().

Beispieldaten: Die Liste in A1:B6 enthält Namen von Mitarbeitern und ihre Hobbys in einem Textstring mit dem Semikolon als Trennzeichen.

	A	B
1	Mitarbeiter	Hobbys
2	Marco	Golf;Tennis;Skifahren
3	Hennes	Tennis;Skateboarden;Kitesurfen
4	Stefan	Golf;Squash
5	Emma	Radfahren;Schwimmen
6	Kerstin	Tischtennis;Snowboarden;Tennis

Abbildung 9.2: Die Ausgangsbasis unseres Beispiels

Die Formel: Die erste Funktion, die zum Einsatz kommt, ist FINDEN(), sie sucht den Text (hier in \$C\$1) in der ersten Mitarbeiterzeile (B2) und liefert die Position des Textes oder #WERT.

Das Ergebnis wird mit ISTZAHL() in WAHR oder FALSCH umgewandelt und mit SUMME() aufsummiert:

=SUMME(--ISTZAHL(FINDEN(\$C\$1;B2))>0)

Die doppelten Negativzeichen stellen sicher, dass die Summe WAHR und FALSCH in 1 und 0 umwandelt.

Das Ergebnis für die Liste sieht so aus:

=SUMME(--ISTZAHL(FINDEN(\$C\$1;B2))>0)		
C	D	E
Tennis		
1		
1		
0		
0		
1		

Abbildung 9.3: Ein Zwischenergebnis

LAMBDA()-Funktion anlegen: Die Formel wird in eine LAMBDA()-Funktion verpackt. Die startet mit zwei Variablen (text und suchtext) und verwendet diese in der Suchformel:

=LAMBDA(suchtext;text;SUMME(--ISTZAHL(FINDEN(suchtext;text))>0))

LAMBDA()-Funktion anwenden: Ohne zusätzliche Parameter würde die Funktion einen Fehlerwert ausgeben. Deshalb wird in einer weiteren Klammer Suchtext und Text angegeben:

=LAMBDA(text;suchtext;
SUMME(--ISTZAHL(FINDEN(suchtext;text))>0))(B2;"Tennis")

Funktion ENTHÄLT() anlegen: Mit *Formel/Definierte Namen/Namen definieren* wird ein neuer Bereichsname ENTHÄLT angelegt. Als Bereich wird die Arbeitsmappe definiert, damit die Funktion für die gesamte Mappe gilt. Im Kommentar sind die Parameter erklärt, und die Bezugszeile enthält die LAMBDA()-Funktion ohne die Parameter in der zweiten Klammer.

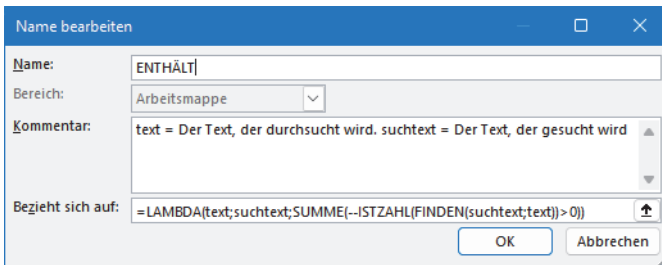


Abbildung 9.4: Ein neuer Bereichsname wird angelegt

Funktion ENTHÄLT() anwenden: Anstelle der LAMBDA()-Formel kann jetzt die neue Funktion verwendet werden, um die Liste nach einzelnen Hobbys zu durchsuchen.

	A	B	C	D
1	Mitarbeiter	Hobbys	Tennis	Golf
2	Marco	Golf;Tennis;Skifahren	1	1
3	Hennes	Tennis;Skateboarden;Kitesurfen	1	0
4	Stefan	Golf;Squash	0	1
5	Emma	Radfahren;Schwimmen	0	0
6	Kerstin	Tischtennis;Snowboarden;Tennis	1	0

Abbildung 9.5: Auf der Suche nach Hobbys

9.3 LAMBDA() in Tabellen

Für die Arbeit mit Tabellen bietet LAMBDA() den Vorteil, dass keine strukturierten Verweise in der Rechenformel nötig sind. Die »blasen« die Tabellenformeln in der Praxis nämlich ziemlich auf. Hier im Beispiel wird die Abfindung für die scheidenden Mitarbeiter berechnet. Die Parameter können wahlweise als Verweis oder als Bezug eingegeben werden.

Funktion Abfindung:

=LAMBDA(Austrittsjahr;Eintrittsjahr;Gehalt;Abfindung;(Austrittsjahr-Eintrittsjahr)*Gehalt*(100%-Abfindung))

In der Tabelle:

=Abfindung([@Austrittsjahr];[@Eintrittsjahr];[@Gehalt];[@[Abfindung %]])

Oder:

=Abfindung(B2;C2;D2;E2)

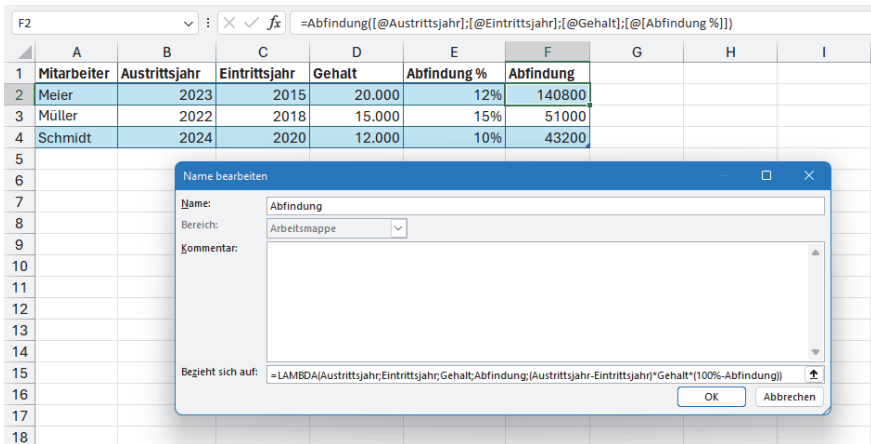


Abbildung 9.6: LAMBDA()-Funktion für die Tabelle

Strukturierte Verweise lassen sich als Parameter in Textform in der LAMBDA()-Funktion verwenden. Werden sie über Bezüge oder Bereichsnamen geliefert, müssen sie mit INDIREKT() vorbehandelt werden.

Hier ein Beispiel:

Die Tabelle enthält Umsätze über Produkte, in J1 steht der Name der Tabelle, in J2 die auszuwertende Spalte. Die sortierte Liste mit eindeutigen Werten aus der bezeichneten Spalte liefert diese Funktionsschachtel:

=SORTIEREN(EINDEUTIG(INDIREKT(J1&"["&J2&"]")))

Die LAMBDA()-Funktion führt Tabellenname und Spalte als Parameter.

```
=LAMBDA(tbl;spalte;SORTIEREN(EINDEUTIG(INDIREKT(tbl&spalte)))(J1;"["&J2&"]"))
Bereichsname: OBST
=OBST(J1;J2)
```

	A	B	C	D
1	Land	Produkt	Monat	Umsatz
2	Deutschland	Tomaten	Januar	1.200
3	Frankreich	Äpfel	Januar	1.500
4	Italien	Oliven	Januar	5.200
5	Spanien	Gurken	Januar	4.100
6	Deutschland	Tomaten	Februar	6.300
7	Frankreich	Äpfel	Februar	7.800
8	Italien	Oliven	Februar	3.200
9	Spanien	Gurken	Februar	1.900
10	Deutschland	Tomaten	März	1.800

I	J	K
Kontinent:	Europa	
Auswertung:	Produkt	


```
=SORTIEREN(EINDEUTIG(INDIREKT(J1&"["&J2&"]")))
```

Äpfel	Äpfel
Gurken	Gurken
Oliven	Oliven
Orangen	Orangen
Tomaten	Tomaten


```
=LAMBDA(tbl;spalte;SORTIEREN(EINDEUTIG(INDIREKT(tbl&spalte)))(J1;"["&J2&"]"))
```

Abbildung 9.7: Tabelle und Tabellenspalte als LAMBDA()-Parameter

9.4 LAMBDA() für Diagramme

Die Entscheidung, das eigene Reporting oder Berichtswesen von PivotCharts auf Standard-Diagramme auf Basis von dynamischen Arrays umzustellen, ist nicht die falsche. Die Vorteile liegen auf der Hand: PivotCharts müssen händisch oder per Intervall aktualisiert werden und brauchen PivotTables als Basis.

LAMBDA() kann hier den Arbeitsaufwand drastisch reduzieren. Aufwendige Formeln werden in LAMBDA()-Funktionen geschrieben und für die Diagrammdaten abgerufen.

Hier ein Beispiel mit Formularelement-Unterstützung:

Das Tabellenblatt *Obstimport* enthält zwei Tabellen, *Europa* und *Südamerika*, mit Importdaten und Umsätzen für Obst. Für die Auswahl des Kontinents wird über *Entwicklertools/Steuerelemente einfügen* ein Optionsfeldrahmen mit zwei Optionen gezeichnet, die gemeinsame Verknüpfung ist die Zelle \$H\$3.

Die zweite Optionsfeldgruppe bietet drei Optionen für die Spaltenwahl an, Ausgabeverknüpfung ist \$H\$8.

Das gewählte Ergebnis liefert je eine WAHL-Funktion in \$J\$1 und \$J\$2:

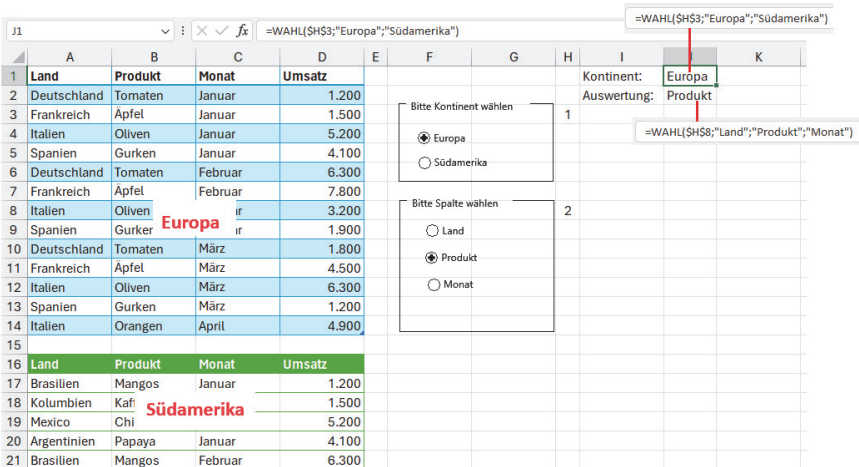


Abbildung 9.8: Auswahl der Tabelle und Spalte über Optionsfelder

Für Diagramme müssen alle Daten, also Rubrikenachse und Datenreihen, in Form von Bereichsnamen vorliegen. Schreiben Sie die komplexen Formeln als LAMBDA()-Funktionen und verwenden Sie diese für die beiden Bereiche *Obst_Rubrik* und *Obst_Umsatz*. Die Bereichsnamen sollten lokal sein, d. h. sich auf das Tabellenblatt beziehen.

Name	Formel
OBST	=LAMBDA(tbl;spalte; SORTIEREN(EINDEUTIG(INDIREKT(tbl&"["&spalte&"]"))))
Obst_Rubrik	=OBST(\$J\$1;\$J\$2)
Obst_Daten	=LAMBDA(produkt;SUMMEWENN(INDIREKT(Obstimport!\$J\$1["&Obstimport!\$J\$2&"]"); produkt;INDIREKT(Obstimport!\$J\$1["Umsatz"])))
Obst_Umsatz	=Obst_Daten(Obstimport!\$H\$11#)

Zeichnen Sie ein leeres Säulendiagramm in das Tabellenblatt und aktivieren Sie *Diagrammentwurf/Daten/Daten auswählen*. Der Rubrikenbereich wird aus *Obst_Rubrik* gebildet, die erste und einzige Datenreihe bekommt den Namen *Umsatz* und den Bezug auf *Obst_Umsatz*.

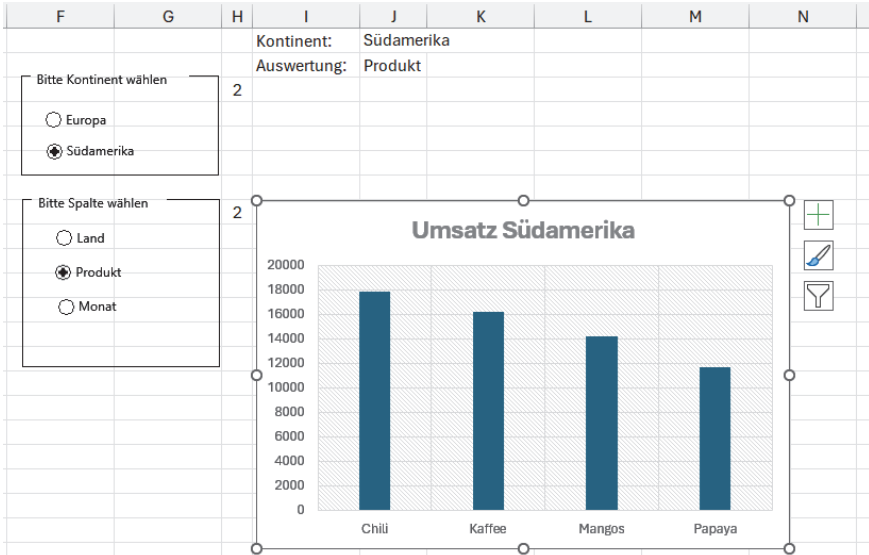


Abbildung 9.9: Das Diagramm mit Rubrik und Daten aus den Bereichsnamen

9.5 LAMBDA-Funktionen

Kurze Zeit nach der Einführung von LAMBDA() als Funktion tauchten im Microsoft-365-Update die ersten LAMBDA-Funktionen auf. Das sind Funktionen, die Aufgaben unter Verwendung einer LAMBDA-Funktion lösen. Hier eine Übersicht:

Funktion	Bedeutung
MAP()	Erstellt ein Array, in dem jedes Element über eine LAMBDA()-Funktion neu berechnet ist.
MATRIXERSTELLEN()	Generiert ein Array unter Verwendung einer LAMBDA()-Funktion.

Funktion	Bedeutung
NACHSPALTE()	Erstellt ein Spaltenarray unter Verwendung einer LAMBDA()-Funktion.
NACHZEILE()	Erstellt ein Zeilenarray unter Verwendung einer LAMBDA()-Funktion.
REDUCE()	Reduziert ein Array auf eine berechnete Größe unter Verwendung einer LAMBDA()-Funktion.
SCAN()	Scannt ein Array und berechnet jedes einzelne Element davon über eine LAMBDA()-Funktion

9.5.1 MAP()

Mit dieser Funktion wird ein Array Wert für Wert analysiert. Die LAMBDA()-Formel im zweiten Argument berechnet den Wert und gibt das Ergebnis in einem dynamischen Array aus. So werden zum Beispiel Wertebereiche mit logischen Bedingungen ermittelt: In E2 und E3 stehen die Grenzwerte für die Zahlen von A2 bis A10.

`=MAP(A2:A10;LAMBDA(b;UND(b>E2;b<E3)))`

Wie viele Werte ermittelt wurden, liefert die SUMME()-Funktion. Da diese aber WAHR und FALSCH nicht zählt, wird der Bezug noch mit Doppelpolus berechnet.

`=SUMME(--B2#)`

	A	B	C	D	E
1	Betrag			Bedingung	
2	120	WAHR		Größer als	50
3	30	FALSCH		Kleiner als	500
4	150	WAHR			
5	400	WAHR		Treffer:	
6	600	FALSCH		3	
7	900	FALSCH			
8	20	FALSCH			
9	800	FALSCH			
10	10	FALSCH			

Abbildung 9.10: Das Diagramm mit Rubrik und Daten aus den Bereichsnamen