



Brilliant  
& Simple

DSci

# Field of Data Science

**Paul Julitz**

Notizen

03.01.2020

# Inhaltsverzeichnis

<b>I Power BI</b>	<b>10</b>
1 Working with M (Power Query) in Power BI . . . . .	11
1.1 Variable . . . . .	11
1.1.1 Variable name . . . . .	11
1.1.2 Zuweisen / Syntax . . . . .	12
1.1.3 number, text, logical . . . . .	13
1.1.4 null . . . . .	13
1.1.5 Type . . . . .	13
1.1.6 Intrinsische Werte . . . . .	14
1.2 Strukturierte Variablen . . . . .	14
1.2.1 List . . . . .	14
1.2.2 Record . . . . .	14
1.2.3 Table . . . . .	14
1.2.4 Beispiele . . . . .	15
1.3 Function . . . . .	15
1.3.1 Implicit function . . . . .	15
1.3.2 Explicit function . . . . .	15
1.3.3 Bewertungen (Conditions) . . . . .	16
1.4 Spezielle . . . . .	18
1.4.1 Empty Variable . . . . .	18
1.4.2 Each . . . . .	19
1.4.3 try otherwise . . . . .	20
2 Business Intelligence: Part I - Power Query . . . . .	20
2.1 Introduction to Excel Tools . . . . .	20
2.2 Query Editor - Tabellenblätter . . . . .	21
2.2.1 Home . . . . .	22
2.2.2 Transform . . . . .	22
3 Business Intelligence: Part II - Data Modeling 101 . . . . .	25
3.1 Data Normalization . . . . .	25
3.2 Data Model . . . . .	25
3.2.1 Primary and Foreign Key . . . . .	25
3.2.2 Beziehungen . . . . .	26
3.2.3 Mehrere Daten Tabellen . . . . .	28
3.2.4 Filter . . . . .	28
3.2.5 Hide fields from client tools . . . . .	30
3.2.6 Hierarchy . . . . .	31
4 Business Intelligence: Power Pivot and DAX . . . . .	32
4.1 Power Pivot 101 . . . . .	32
4.1.1 Oberfläche von Power Pivot . . . . .	32
4.1.2 Berechnungen . . . . .	32
4.1.3 Funktion mit normalen Tabellen . . . . .	32
4.1.4 Calculated Columns . . . . .	34
4.1.5 Implicit Funktionen . . . . .	35
4.1.6 Explicit Funktionen - Measures . . . . .	35
4.1.7 Filterfunktion . . . . .	37

4.2	Basic DAX Function . . . . .	37
4.2.1	Syntax und Operatoren . . . . .	37
4.2.2	Function . . . . .	38
4.3	Advanced DAX Function . . . . .	40
4.3.1	Calculate . . . . .	40
4.3.2	Filter . . . . .	41
4.3.3	All . . . . .	42
4.3.4	Iterator (X) - SumX() . . . . .	43
4.3.5	Iterator (X)- RankX() . . . . .	44
4.3.6	Time-Intelligence Function . . . . .	45
5	Power BI Dataflow Essentials . . . . .	46
5.1	Gateways and Gatway Clusters . . . . .	46
5.2	Dataflows . . . . .	52
6	Advanced Microsoft Power BI . . . . .	55
6.1	Filter . . . . .	55
6.1.1	Filter für Measure und DAX Funktionen . . . . .	55
6.1.2	All-Funktion . . . . .	55
6.1.3	Filter-Funktion . . . . .	56
6.2	DAX Measures . . . . .	57
6.2.1	Quick Measure . . . . .	57
6.2.2	Divide and Exponent Function . . . . .	58
6.2.3	VAR and Return . . . . .	58
6.2.4	If-Statement . . . . .	58
6.2.5	SUMX() . . . . .	58
6.2.6	COUNTX() . . . . .	58
6.2.7	DateDiff . . . . .	58
6.2.8	DATESBETWEEN . . . . .	58
6.3	Rank . . . . .	58
6.3.1	Funktionsweise . . . . .	58
6.3.2	Beispiel . . . . .	59
6.3.3	Gruppierungen . . . . .	60
6.3.4	RANK.EQ . . . . .	61
6.4	ALL Functions . . . . .	61
6.4.1	ALL . . . . .	61
6.4.2	ALLSELECTED . . . . .	61
6.4.3	ALLEXCEPT . . . . .	62
6.4.4	SELECTEDVALUES . . . . .	62
6.4.5	Value . . . . .	63
7	Power BI Mistakes to be avoided . . . . .	63
7.1	Reduce Data . . . . .	63
7.2	Display additional information . . . . .	64
7.3	Query Folding . . . . .	65
7.4	Table View - Running Query Twice . . . . .	67

<b>II</b>	<b>Power Platform</b>	<b>68</b>
1	Governance and Administration . . . . .	69
1.1	Architecture . . . . .	69
1.1.1	Roles . . . . .	69
1.1.2	Environment(Umgebung) . . . . .	71
1.2	Lizenzstruktur . . . . .	76
2	Application Lifecycle Management (ALM) . . . . .	76
2.1	Allgemein . . . . .	76
2.2	Solution . . . . .	77
2.2.1	Un- and Managed Solution . . . . .	77
2.2.2	Update Solution through Layering . . . . .	79
2.3	ALM Strategy . . . . .	83

3	Building . . . . .	83
3.1	Open Topics . . . . .	83
3.2	Environment Variablen . . . . .	84
<b>III</b>	<b>VBA</b>	<b>88</b>
1	Das VBA-Tutorial . . . . .	89
1.1	Grundlagen . . . . .	89
1.1.1	Debug.Print . . . . .	89
1.1.2	Array . . . . .	89
1.2	Objekte . . . . .	89
1.2.1	Klassen . . . . .	89
1.2.2	Methoden . . . . .	91
1.2.3	Eigenschaft . . . . .	91
1.2.4	Auflistung . . . . .	94
1.2.5	Ereignisse . . . . .	95
1.2.6	Objektaweisung . . . . .	96
1.2.7	Me-Object . . . . .	97
1.2.8	Objektmodell . . . . .	98
1.3	Fehlerbehandlung . . . . .	99
1.4	Kurzer Einblick Formular . . . . .	100
1.5	Microsoft Access . . . . .	100
1.5.1	Marcos, Module . . . . .	100
1.5.2	Formulare, Reports . . . . .	100
1.5.3	Tabellen . . . . .	101
2	VBA: Access . . . . .	102
2.1	Sammelsurium . . . . .	102
2.2	Introduction VBA Basic . . . . .	102
2.2.1	VBA Syntax . . . . .	102
2.2.2	Object Modell for Access . . . . .	103
2.2.3	VBA Editor . . . . .	104
2.2.4	Processor: Subroutine and function . . . . .	104
2.2.5	Processor: Function . . . . .	105
2.2.6	Set Statement . . . . .	105
2.2.7	First Code-Marco . . . . .	105
2.3	Variables, Constants, Calculation . . . . .	107
2.4	Add Logic to your VBA Code . . . . .	107
2.4.1	Iterator-Prozeduren . . . . .	107
2.5	Debug Code . . . . .	108
2.6	Manipulate Database Object using DoCmd Object . . . . .	108
2.6.1	Open Objects . . . . .	108
2.6.2	Close Objects . . . . .	109
2.6.3	RunCommand Export . . . . .	109
2.7	Read and manipulate Table Data . . . . .	109
2.7.1	Add New . . . . .	109
2.7.2	Edit . . . . .	109
2.7.3	Data Perservation . . . . .	109
2.7.4	TableDef . . . . .	110
2.8	Manipulate a Database using the Application Object . . . . .	110
2.8.1	Function to Summarize . . . . .	110
2.8.2	DLookup() . . . . .	110
2.9	Control Forms and Reports . . . . .	110
2.9.1	Sinnvolle Einschränkungen . . . . .	110
2.9.2	Verfeinerung . . . . .	111

<b>IV</b>	<b>SQL</b>	<b>112</b>
1	SQL - Conceptual . . . . .	113
1.1	Keys . . . . .	113
1.1.1	Primäry Key . . . . .	113
1.1.2	Composite Key . . . . .	113
1.1.3	Forgein Key . . . . .	113
1.1.4	Composite Key . . . . .	114
1.1.5	Compound Key . . . . .	114
1.2	Normilization . . . . .	114
1.2.1	1NF - Atomicity . . . . .	114
1.2.2	2NF - Partial Dependency . . . . .	115
1.2.3	3NF - Transitive Dependency . . . . .	116
1.3	Junction . . . . .	116
1.3.1	One-to-One Relationship . . . . .	116
1.3.2	One-to-Many Relationship . . . . .	117
1.3.3	Many-to-Many Relationship . . . . .	117
1.4	Data Integrity . . . . .	117
2	SQL - Introduction . . . . .	117
2.1	SELECT Statement . . . . .	118
2.1.1	FROM - Input table . . . . .	119
2.1.2	WHERE - Where row is evaluated . . . . .	119
2.1.3	GROUP BY - zusammenfassen von Spalten . . . . .	120
2.1.4	ORDER BY - sort table . . . . .	120
2.1.5	LIMIT - Who many Rows . . . . .	122
2.1.6	OFFSET - Ausschluss der ersten Zeilen . . . . .	122
2.2	Concepts . . . . .	122
2.2.1	Function Types . . . . .	122
2.2.2	DISTINCT Clause . . . . .	123
2.2.3	CREATE TABLE ... () . . . . .	124
2.2.4	DROP TABLE . . . . .	125
2.2.5	INSERT INTO ... ()() . . . . .	125
2.2.6	UPDATE ... SET ... WHERE . . . . .	125
2.2.7	DELETE FROM ... WHERE . . . . .	126
2.2.8	NULL Value . . . . .	126
2.2.9	DEFAULT . . . . .	126
2.2.10	UNIQUE . . . . .	127
2.2.11	ALTER TABLE ... ADD . . . . .	127
2.2.12	CASE - Conditional Expression . . . . .	127
2.2.13	Ansteuern von Spalten . . . . .	128
2.3	Joints . . . . .	128
2.3.1	Join on or more tables . . . . .	128
2.3.2	Kinds of Joins . . . . .	129
2.4	Selection of Function . . . . .	129
2.4.1	Find the lenght of a string . . . . .	130
2.4.2	Substring of Field of Date . . . . .	130
2.4.3	Count()- Group by (aggregate function) . . . . .	130
2.4.4	Mix . . . . .	131
2.5	Transaction . . . . .	131
2.6	Triggers . . . . .	132
2.7	Subselects . . . . .	133
2.7.1	IN Operator . . . . .	133
2.7.2	View - temporay table . . . . .	134

<b>V Version Control</b>	<b>135</b>
1 GitHub Introduction . . . . .	136
1.1 Befehle . . . . .	136
1.2 Unterschied zwischen git bash, cmd und gui . . . . .	136
1.2.1 Staging Area / Index / Cache . . . . .	137
1.2.2 Commiting . . . . .	139
1.3 Example: github-slideshow . . . . .	140
1.3.1 Clone Respository and Create a Branch . . . . .	140
1.3.2 Check all the branches of one repository . . . . .	140
1.3.3 Chance branch name . . . . .	140
1.3.4 Change aktiv branch . . . . .	141
1.3.5 Push new local branch . . . . .	141
1.3.6 Check the status of a branch . . . . .	141
1.3.7 Create a new file . . . . .	141
1.3.8 Update Branch . . . . .	141
1.3.9 Pull Request . . . . .	142
1.3.10 Delete local branch . . . . .	142
1.3.11 Delete remote branch . . . . .	143
1.3.12 Prune deletet remote branches . . . . .	144
1.4 Designing your Github Page . . . . .	144
2 GitLab Introduction . . . . .	144
2.1 Blame . . . . .	144
2.2 Groups and Subgroups . . . . .	145
2.3 Role Permissions . . . . .	146
2.4 Milestones . . . . .	147
3 *Vitural Enviroment Anaconda . . . . .	147
4 Version Kontrolle Stategie . . . . .	148
4.1 Verschiedene Branches . . . . .	148
4.1.1 Master and Develop . . . . .	148
4.1.2 Feature and Hotfix . . . . .	148
4.2 Merging Optionen . . . . .	148
4.2.1 Merge (fast forward) . . . . .	149
4.2.2 Merge Recursive . . . . .	149
4.2.3 Merge Squash . . . . .	153
4.2.4 Merge --no -ff . . . . .	153
4.2.5 Rebase . . . . .	154
<b>VI Qlik Sense</b>	<b>155</b>
1 Sammelsurium - Qlik Sense . . . . .	156
1.1 Load . . . . .	156
1.2 Incremental Loading . . . . .	156
1.3 Data connection types . . . . .	156
2 3 - Udemy - Certificate in Qlik Sense Analytics Development . . . . .	157
2.1 Budgeting and KPI (goal) setting . . . . .	157
2.1.1 DevHub . . . . .	157
2.1.2 Von Hub bis Sheet . . . . .	158
2.1.3 Selection . . . . .	161
2.2 Qlik Key . . . . .	161
2.2.1 Composite Key . . . . .	161
2.2.2 Synthetic Key . . . . .	164
2.2.3 Speicherung . . . . .	165
3 Python Connection . . . . .	165

<b>VII Python Programming</b>	<b>166</b>
1 Object-Oriented Design . . . . .	167
1.1 Objekt Orientierte Grundlagen . . . . .	167
1.1.1 Beispiel . . . . .	167
1.1.2 Objekte . . . . .	167
1.1.3 Klassen . . . . .	168
1.1.4 Abstraction, Encapsulation, Inheritance, Polymorphismen . . . . .	169
1.2 Class Diagramm . . . . .	171
1.2.1 Creating Class Diagramm . . . . .	171
1.2.2 Konstruktor . . . . .	171
1.2.3 Destruktor . . . . .	172
1.2.4 Static / Instanz Variable/ Methode . . . . .	173
1.3 Inheritance and Composition . . . . .	174
1.3.1 Abstract class/ Methode . . . . .	174
1.3.2 Interfaces . . . . .	175
1.3.3 Aggregation / Composition . . . . .	176
1.3.4 Zusammenfassung Relationen . . . . .	178
1.4 Software Development . . . . .	178
2 Introduction in Python . . . . .	178
2.1 Running Python from VS Code . . . . .	178
2.1.1 Befehle . . . . .	178
2.1.2 Launch Json File . . . . .	179
2.1.3 Internal Console instead of Terminal . . . . .	179
3 Algorithme . . . . .	179
<b>VIII Machine Learning and Data Analysis</b>	<b>180</b>
1 Examples from common libraries . . . . .	181
1.1 Working with Excel . . . . .	181
1.1.1 Basic Pandas . . . . .	181
1.1.2 Dataframe Functions . . . . .	183
1.1.3 openpyxl . . . . .	191
1.2 Exploratory Data Analysis . . . . .	194
1.2.1 Overview . . . . .	194
1.2.2 Relationship Analysis . . . . .	201
1.3 Data Visualization - Matplotlib . . . . .	208
1.3.1 Preparation . . . . .	208
1.3.2 Simple Plot . . . . .	214
1.3.3 MATLAB-Style vs Object-Oriented Style . . . . .	217
2 Model Metric . . . . .	221
2.1 Performance . . . . .	221
2.1.1 Regression Analysis . . . . .	221
3 Theory of Regression, Clustering and Algorithms . . . . .	222
3.1 *Clustering: K-Means . . . . .	222
3.1.1 Introduction . . . . .	222
3.1.2 Silhouette . . . . .	222
3.1.3 Test Data Sets . . . . .	223
3.2 *Working with Predictive Analytics . . . . .	225
3.2.1 Data Preprocessing: Categorical Values . . . . .	225
<b>IX Machine Learning Theory</b>	<b>228</b>
1 Logistische Regression . . . . .	229
1.1 Overview . . . . .	229
1.2 General-Lineare-Model . . . . .	229
1.2.1 Model . . . . .	229
1.2.2 Unterschied Lineares Model und General-Lineare-Model . . . . .	230

1.2.3	Spezialfall: Lineares Model im General-Linearen-Model . . . . .	230
1.2.4	Spezialfall: Binäre Verteilung im General-Linearen-Model . . . . .	230
1.3	Hypothesenfunktion . . . . .	231
1.3.1	Regressionsmodell . . . . .	231
1.3.2	Lineare Regression . . . . .	231
1.3.3	(Exkurs) Nichtlineare Regression . . . . .	232
1.4	Binäre Klassifikation . . . . .	232
1.4.1	Modell . . . . .	232
1.4.2	Interpretation Lineares logistisches Modell . . . . .	233
1.4.3	Exkurs: Example Malignat . . . . .	233
1.4.4	Exkurs: Dummy-Variable . . . . .	233
1.5	Decision Boundary . . . . .	234
1.5.1	Simple Lineare Model . . . . .	234
1.5.2	Komplexe Polynomisches Model . . . . .	234
2	Bewertungsmetriken . . . . .	236
2.1	Regression Analyse . . . . .	236
2.1.1	Bestimmtheitsmaß $R^2$ . . . . .	236
2.1.2	Mittlere Quadratische Abweichung . . . . .	236
2.2	Klassifikation Analyse . . . . .	236
3	Klassen eines statistischen Prozesses . . . . .	236
<b>X Artificial Intelligence</b>		<b>238</b>
4	Overview . . . . .	239
<b>XI Stochastik</b>		<b>240</b>
1	Überblick . . . . .	241
2	Wahrscheinlichkeitstheorie . . . . .	243
2.1	Wahrscheinlichkeitsraum . . . . .	243
2.1.1	Ergebnismenge . . . . .	243
2.1.2	Ereignismenge . . . . .	244
2.1.3	Wahrscheinlichkeitsmaß . . . . .	244
2.2	Zufallsvariable . . . . .	245
2.2.1	Herleitung . . . . .	245
2.2.2	Eigenschaften und Besonderheiten . . . . .	249
2.2.3	Beispiele . . . . .	249
2.3	Momente einer Zufallsvariable . . . . .	250
3	Inferenz Statistik . . . . .	251
3.1	Statisches Model . . . . .	251
3.2	Schätzen . . . . .	254
3.3	Testen . . . . .	254
4	Modellierung statistischer Zusammenhänge . . . . .	254
5	Regression . . . . .	254
5.0.1	Beziehungen . . . . .	254
5.1	Deskriptive Statistik . . . . .	254
<b>XII Neues Familienmitglied-Zeitstrahl</b>		<b>256</b>
1	Physische Entwicklungslimitationen . . . . .	257
2	Dynamische Beziehungsstrukturen . . . . .	259
3	Zeitlose Komponenten . . . . .	259
4	Mathematik in der Welt der Töne . . . . .	259
4.1	Einführung . . . . .	259

<b>XIII Anhang</b>	<b>260</b>
<b>A Abkürzungsverzeichnis</b>	<b>261</b>
<b>B Glossar</b>	<b>265</b>
<b>C Literatur</b>	<b>269</b>
<b>D Definitions</b>	<b>270</b>

**Teil I**

**Power BI**

# 1 Working with M (Power Query) in Power BI

**Funktion** Eine **Funktion** ist ein mathematische Objekte, welches eine spezielle Relation, angewandt auf eine definierte Menge, ist. Dabei wir jedem Input genau ein Output zugewiesen. Um genau zusein, ist die Funktion die Abbildungsregel, das Objekte lautet Abbildung. Man nennt Abbildungen auch Funktionen, wenn sie ein ein Tupel aus Reelen Zahlen abbildet (Check). Beispiel:  $f : \mathbb{R} \rightarrow \mathbb{R}$

**Expression** Eine **Expression** ist ein **algebraische** oder **syntakische Phrase**. In Power Query spricht man von Expression und meint damit ganze Zeilen Ausdrücke.

**Value** Es gibt verschiedene Datenstrukturen. Dabei können diese auch aus mehreren zusammengebaut sein.

Die Datentypen in Power Query sind ähnlich zu anderen Programmierungssprachen. Es gibt jedoch Ausnahmen. Ist kein Wert für eine Zelle vorhanden, so wird Type **Null**. Weiter Datentypen werden im Folgenden beschreiben. Hierbei werden spezielle Funktionen verwandt, um diesen Wert zu erzeugen. Dabei werden bestimmt Syntaxen verlangt.

## 1.1 Variable

Variablen werden bei erzeugen in *M* nicht deklariert. Eine automatische Zuweisung der Typen funktioniert über die Syntax. Für Funktionen können jedoch Typen für Inputvariablen und die Funktion selbst definiert werden.

Kind	Literal
Null	null
Logical	true false
Number	0 1 -1 1.5 2.3e-5
Time	#time(09,15,00)
Date	#date(2013,02,26)
DateTime	#datetime(2013,02,26, 09,15,00)
DateTimeZone	#datetimezone(2013,02,26, 09,15,00, 09,00)
Duration	#duration(0,1,30,0)
Text	"hello"
Binary	#binary("AQID")
List	{1, 2, 3}
Record	[ A = 1, B = 2 ]
Table	#table({ "X", "Y"}, {{0,1}, {1,0}})
Function	(x) => x + 1
Type	type { number } type table [ A = any, B = text ]

Abbildung 1:

Link

### 1.1.1 Variable name

Variablen werden in Power Query, wie auch in anderen Sprachen, mit Hilfe einer Zeichenkette definiert. Es gibt jedoch die Besonderheit, das Variablennamen auch mit einem Leerzeichen definiert werden dürfen. Dafür muss der Name als String gesehen werden und mit einem # am Anfang versehen werden.

---

```
1 let
2   #"Das ist wirklich ein Variablenname!" = 13,
3   y = #"Das ist wirklich ein Variablenname!"
4 in
5   y // Returns 13
```

---

## 1.1.2 Zuweisen / Syntax

Für das Zuweisen eines Types zu einer Variablen wird der **as** Ausdruck verwandt. Wird eine Variable erzeugt ohne eine Funktion zu verwenden, erfolgt eine automatische Type-Zuweisung.

```
1 let
2   y = 1.3234 // type number
3   #"Variable 2" = "Text" // type text
4   #"Variable 3" = #date(1,1,1908) // Zuweisung erfolgt durch die intrinsische Funktion date
```

Der **as** Ausdruck wird verwendet, um Input und Output von Funktionen zu definieren.

```
1 let
2   Funktion1 = (Input as number) as number => 2 + 3
3 in
4   Funktion1
```

Wollen wir einen Typen einer Spalte einer Tabelle ändern, so nutzen wir die

Table.TransformColumnTypes  
8/1/2019 • 2 minutes to read

### Syntax

```
Table.TransformColumnTypes(table as table, typeTransformations as list, optional culture as nullable text) as table
```

### About

Returns a table from the input `table` by applying the transform operation to the columns specified in the parameter `typeTransformations` (where format is {column name, type name}), using the specified culture in the parameter `culture`. If the column doesn't exist, an exception is thrown.

### Example 1

Transform the number values in column [a] to text values from the table `(([a = 1, b = 2], [a = 3, b = 4]))`.

```
Table.TransformColumnTypes(Table.FromRecords(([{"a": 1, "b": 2}, {"a": 3, "b": 4}]), {"a", type text}, "en-US")
```

Abbildung 2:

Die Funktion lässt eine Tabelle als Input ein. Für die Transformation wir eine *Liste* benötigt. Diese benötigt jedoch für jeden Eintrag selber eine Liste, mit dem ersten Eintrag den Spaltennamen und den zweiten den zu bezeichneten Typ. Beispiel:

```
let
    Quelle1 = {1,2,3,5, "abc"}, 
    Quelle2 = #table({("a","B"),{(1,2),(2,3),(3,2)}}, 
    function = (table) => table[A], 
    table2 = function(Quelle2)
in
    table2
```

Abbildung 3: Eine Spalte

Abbildung 4: Mehrere Spalten

**Achtung** Es können verschiedenen Syntaxen verwendet werden um eine Typ zu definieren.

Data Type	Syntax 1	Syntax 2
Whole Number	Int64.Type	-
Decimal Number	Number.Type	type number
Dates	DateType	type date
Text	Text.Type	type text
Binary		type binary
Date/Time		type datetime
Date/Time/Timezone		type datetimezone
Duration		type duration
Function		type function
List		type list
True/False		type logical
Record		type record
Any		type any
Any Non-Null		type anynonnull
None		type none
Null		type null
Type		type type

Abbildung 5: Mehrere Spalten

Dies sieht man auch im Beispiel mit mehreren Spalten. Die Integer Typumwandlung erfolgt mit dem "Int64.Type" dieser wird jedoch nicht "type Int64" geschrieben. Für andere Typen kann diese Logik aber angewandt werden.

### 1.1.3 number, text, logical

Variablen müssen

---

```

1 let
2   x = 5,
3   y = x + 5,
4   w = "Test Text",
5   z = true,
6 in
7   x // Returns

```

---

### 1.1.4 null

Variablen besitzen eine Ausprägung für einen Wert. Das Konzept von "any" und "null" wird hier nicht berücksichtigt. Diese Typen können ebenfalls abgerufen werden. Dabei kann eine Funktion eine Input auf den type testen.

---

```

1 let
2   y = null // y ist vom Typ null
3   w = "null" // w ist vom Typ text
4   z = 0 // z ist vom Typ number
5 in
6   w // Return null

```

---

### 1.1.5 Type

Der Type **type** wird wie intrinsische Variablen mit einer Prefix deklariert. Der Prefix lautet "type" und der Variablentyp wird in { }. Für Tabellen können die Typen für die Spalten gleich mit im Tabellenkopf vermerkt werden.

---

```

1 let
2   x = type {number}, // Die Variable x ist vom type type.
3   y = #table(
4     {Column1, Column2}, // Kopf der Tabelle ohne Typen
5     {
6       {2, "text"},      // Type ist "text"
7       {4, "hello"}    // Type ist "text"

```

---

```

8         }
9     ),
10
11    w = #table(
12        type table [Column1 = type number, Column2 = type text],
13        \\ Kopf der Tabelle mit Typen
14        {
15            {2, "text"},
16            {4, "hello"}
17        }
18    )
19
20 in
21 y

```

---

### 1.1.6 Intrinsiche Werte

Um Werte wie *date*, *duration*, *binary* etc. zu schaffen, werden spezielle **single intrinsic function** benötigt. Man erkennt diese daran, dass vor ihnen ein # steht.

```

1 let
2   x = #date(2019,2,2)
3 in
4   x // Returns 2019-2-2

```

---

## 1.2 Strukturierte Variablen

Es gibt drei verschiedene Datentypen, welche aus mehreren Komponenten bestehen  
erden. [Link](#), [Link](#)

```

1 let
2   x = {a,2,abc} // List
3   y = [ // Record
4       variable1 = 1,
5       Item = "abc",
6       ID = 4
7   ],
8   w = #table(
9       {"Column1", "Column 2" },
10      {
11          {2, 4},
12          {"abs", 5},
13          {123j,#date(2019,2,2)}
14      }
15 in
16   x{3} // Returns abc
17   // y[Item] returns abs
18   // w{2}[Column1] returns 123j; 2 ist die zweite Zeile (Start 0)

```

---

### 1.2.1 List

Eine Liste ist ein einfaches Array, welches alle verschiedene Typen von Objekten beinhalten kann.

### 1.2.2 Record

Ein Record ist ein Array, welches zu jedem Eintrag eine Indexnamen ausweisen kann. Es können auch Records von Records gemacht werden.

### 1.2.3 Table

Eine Tabelle teilt sich in den Tabellenkopf und die Zeilen auf. Um eine Table zu erstellen, muss die intrinsische Funktion `#table()` angewandt werden.

Bei der Erstellung der Tabellen, können gleich **typen** von Datenformaten gewählt werden, siehe Zuweisen/Syntax.

#### 1.2.4 Beispiele

---

```

1 let
2   Source =
3 {
4   1,
5   "Bob",
6   DateTime.ToString(DateTime.Now(), "yyyy-MM-dd"),
7   [OrderID = 1, CustomerID = 1, Item = "Fishing rod", Price = 100.0]
8 }
9 in
10 Source

```

---

Das Result ist:

A List containing a Record	
1	
"Bob"	
2015-05-22	
OrderID	1
CustomerID	1
Item	"Fishing rod"
Price	100.0

Abbildung 6:

### 1.3 Function

Eigenen Funktionen können ebenfalls geschrieben werden. Es ist jedoch die sequenzielle Natur des M-Codes zu berücksichtigen. Eine Besonderheit zur Sprache C++ ist, dass die Typen für Input und Output Variablen nicht dekliniert werden müssen. Es wird dann von **impliziten Funktionen** gesprochen. Der Tye *any* wird dann als *Default* genutzt:

#### 1.3.1 Implicite function

Name der Funktion = ( Inputvariablen ) => Expression, (1.1)

---

```

1 let
2   Add = (x, y) => x + y,
3   AddResults =
4   [
5     OnePlusTwo = Add(1,2) // Ergebnis ist 3
6     OnePlusOne = Add(1,1) // Ergebnis ist 2
7   ]
8 in
9   AddResults
10  /*
11  OnePlusTwo 3
12  OnePlusOne 2
13 */

```

---

#### 1.3.2 Explicite function

Bei dieser Art von Funktionen müsse die Typen für die Input- wie Outputvariablen angegeben werden. Dies erfolgt im zweiten Block.

Name der Funktion = ( Inputvariablen as "type" ) as "type" => Expression, (1.2)

Mit etwas geschickt, können auch komplexere Funktionen gebaut werden.

```
1 let
2   GreaterThanFive = (list) =>
3     let
4       OrderedList = List.Select(list, each _ >= 5)
5       /*
6       List.Select(list as list, selection as function) as list
7       */
8       Result = List.First(OrderedList)
9     in
10    Record = [
11      Ergebnis1 = GreaterThanFive({1,2,6}) // Returns 6
12      Ergebnis2 = GreaterThanFive({12,12,45,23}) // Returns 12
13    ]
14  in
15  Record
```

### 1.3.3 Bewertungen (Conditions)

Es ist essentielle für das Verstehen der meisten Funktionen, wie Tabllen, Listen, Records abgerufen und Teilmengen der Informationen selektiert werden können.



Abbildung 7: Tabelle - Zeile und Spalte (type cell-type)

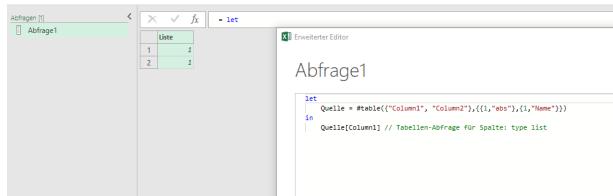


Abbildung 8: Tabelle - Spalte (type list)

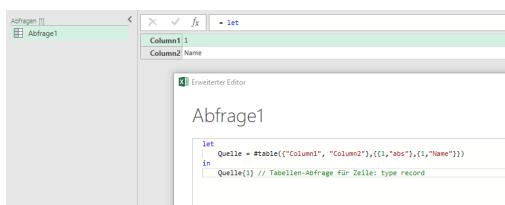


Abbildung 9: Tabelle - Zeile (type record)

## Table

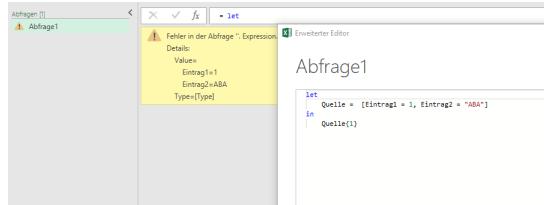


Abbildung 10: Record - Zeile : Geht nicht!)

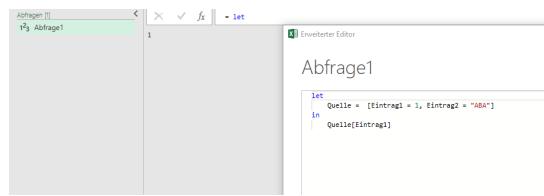


Abbildung 11: Record - Eintrag [] (type cell-type  
; Wichtig: Table.SelectRows()

## Record

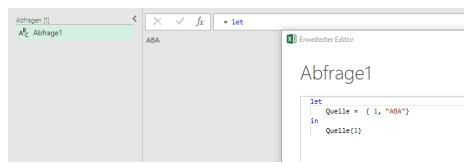


Abbildung 12: Liste - Zeile (type cell-type)

## List

Eine Funktion kann als Outputwert vom *type logic* sein, sie kann aber auch jeden anderen Wert annehmen. Wichtig ist jedoch, wird eine Logische Abfrage im Funktionskörper veranlasst und diese nicht verarbeitet, so nimmt die Funktion den Typen logic an.

Eine Funktion kann, wenn ihr eine Tabelle übergeben wird, eine Liste wieder geben, wenn aus der Tabelle eine Spalte mit [] ausgelesen wird, siehe 3. Wird die ausgelesene Liste verglichen mit einem einzelnen Wert, so wird die logische Abfrage *false* wiedergeben, da die beiden typen nicht gleich sind.

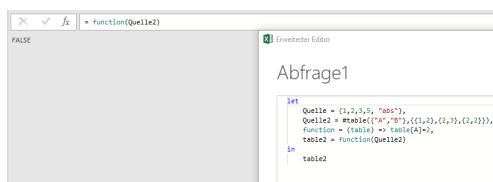


Abbildung 13:

Selbst wenn jeder Eintrag in der List den gleichen zuüberprüfenden Eintrag hat, gibt die Funktion *false* wieder

```

let
    Quelle = {1,2,3,4,5, "abs"},
    Quelle2 = #table({{"A","B"},{{2,2),(2,3),(2,2)}},
    function = [table] => table[A]=(2,2),
    table2 = function(Quelle2)
in
    table2

```

Abbildung 14:

Wird ein Abgleich mit einer List gemacht, welche die gleichen Einträge beinhaltet, so gibt die Funktion *true* wieder.

```

let
    Quelle = {1,2,3,4,5, "abs"},
    Quelle2 = #table({{"A","B"},{{2,2),(2,3),(2,2)}},
    function = [table] => table[A]=(2,2),
    table2 = function(Quelle2)
in
    table2

```

Abbildung 15:

Es sei vermerkt, es werden nicht die Typen abgefragt, sondern jeder Eintrag in der Liste wird verglichen.

```

let
    Quelle = {1,2,3,5, "abs"},
    Quelle2 = #table({{"A","B"},{{2,2),(2,3),(2,2)}},
    function = [table] => table[A]=(2,2),
    table2 = function(Quelle2)
in
    table2

```

Abbildung 16:

Im Abschnitt **each** wir über die eine spezielle Funktion gesprochen.

## 1.4 Spezielle

### 1.4.1 Empty Variable

Um den Ausdruck **each** verstehen zu können, sollte die Leere-Variable besprochen werden. In Python als auch M wir \_ für Puffervariablen genutzt, welche später nicht mehr gebracht werden.

---

```

1 let
2     _ = Table.FromColumns({
3         {"Fred", "Mary", "Phil", "Sue"},
4         {1, 2, 3, 4},
5         {5, 2, 5, 1}},
6         {"Person", "Index", "Score"} // Tabellenkopf
7     )
8 in
9     _ // Die temporäre Variable heißt _ und ist vom Type table.

```

---

Um verschiedenen Daten von der Tabelle abzurufen, nutzen wir die Indexschreibweise:

$$\text{Tabellen-Variablen-Namen} \{ \text{Zeile} \} [\text{Spaltennamen}] \quad (1.3)$$

Für den Variablennamen \_ gilt dies natürlich auch:

---

```

1 let
2     _ = Table.FromColumns({

```

```

3     {"Fred", "Mary", "Phil", "Sue"},  

4     {1, 2, 3, 4},  

5     {5, 2, 5, 1}},  

6     {"Person", "Index", "Score"} // Tabellenkopf  

7   )  

8 in  

9   _[Person] // Return Gesamte Spalte Person vom type list  

10  // _{2}[Person] // Return Zeile 2 der Spalte Person

```

---

Den temporären Variablennamen können wir auch weglassen, wenn Tabellen angesprochen werden. Dieser Prozess sollte für Listen nicht verwendet werden, da Power Query einen Zeilenverweis für Listen als neue Liste interpretiert:

```

1 let  

2   _ = {"ab", null, 6}  

3 in  

4   _{2} // Return null  

5   // _{2} // Erstellen einer weiteren Liste

```

---

Für Tabellen kann jedoch der `_` weggelassen werden.

```

1 let  

2   _ = Table.FromColumns({  

3     {"Fred", "Mary", "Phil", "Sue"},  

4     {1, 2, 3, 4},  

5     {5, 2, 5, 1}},  

6     {"Person", "Index", "Score"} // Tabellenkopf  

7   )  

8 in  

9   [Person] // Return Gesamte Spalte Person

```

---

#### 1.4.2 Each

Die Funktion `each` ist eine einfache Funktion, welche die Syntax der Leeren Variable nutzt. Es gilt:

$$\text{each } x := (\_) \Rightarrow \_x \quad (1.4)$$

Der Begriff `each` ist vielleicht etwas irreführend, da nicht wirklich ein *loop*-Durchlauf gestartet wird. Vielmehr wird die `each`-Funktion genutzt, wenn die ummantelnde Funktion den loop-Durchlauf startet und `each` eine Überprüfung für jede der übergebenen Element machen soll.

Es ist sehr wichtig zwischen Tabellen und Records zu unterscheiden. Wir der `each`-Funktion eine Tabelle übergeben und sie soll bewerten, ob eine Wert in jeder Zeile steht, funktioniert dies nicht mit einem einzelnen Wert

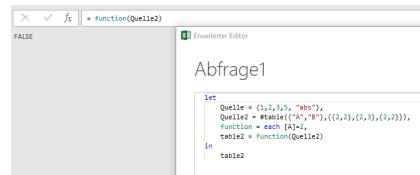


Abbildung 17:

Auch hier gilt, es muss mit einer Liste abgeglichen werden.

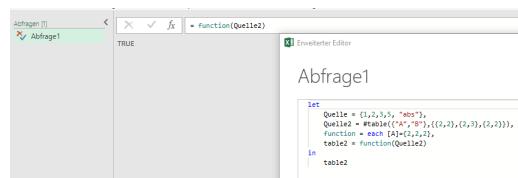


Abbildung 18:

Damit each true zurückgeben kann, muss die geprüfte List ebenfalls die richtigen Wert enthalten.

```

let
    Quelle = {1,2,3,4, "abc"},
    Quelle1 = #table({{"A","B"}},{{1,2},{3,4},{2,3}}),
    function = each [A]=(2,3),
    table2 = function(Quelle1),
    in
        table2

```

Abbildung 19:

Wird die Funktion **Table.SelectRows()** eingesetzt, so wird nicht eine Tabelle, sondern ein Record für die zu überprüfenden Zeile übergeben.

A	B
1	2
2	2
3	2

```

let
    Quelle = {1,2,3,5, "abc"},
    Quelle2 = #table({{"A","B"}},{{1,2},{3,4},{2,3}}),
    function = each [A]=(2,3),
    table2 = Table.SelectRows(Quelle2, each [A]=2)
in
    table2

```

Abbildung 20:

Der **each** Operator nimmt das übergegebenen Record-Element, welches aus der Tabelle extrahiert wurde und überprüft mit der Eintragssyntax `[]`, die auch die Spaltensyntax für Tabellen ist, den Wert der für die Zeile in der entsprechenden Spalte steht. Eine einfache logische Abfrage entscheidet dann, ob der Eintrag zu "jeder" Zeile in der entsprechenden Spalten richtig ist. Wenn ja, dann wird die Zeile ausgewählt, wenn nein, dann wird die Zeile gelöscht.

In der *Table.SelectRows()* Funktion wird jede Zeile als Record übergeben. Wie oben beschrieben, wird die Auswahl pro Zelleninhalt ausgewertet.

A	B
2	2
3	2

```

let
    Quelle = {1,2,3,5, "abc"},
    Quelle2 = #table({{"A","B"}},{{1,2},{3,4},{2,3}}),
    Quelle3 = {1,2,3,4, "abc"}, // Fehler
    Quelle4 = [A = 2],
    function = each [A]=2,
    table2 = Table.SelectRows(Quelle2, each [A]=2), // Erstelle Records, nicht Tabellen oder Listen!!!
    table3 = Function(Quelle4)
in
    table3

```

Abbildung 21:

#### 1.4.3 try otherwise

Das *try ... otherwise* Statement ist für besonders geeignet, um Fehler abzufangen. Wie das *if*-Statement werden zwei Szenarien festgelegt, welche ausgeführt werden, nach der Überprüfung einer Bedingung.

Das *try-otherwise* Statement fängt jedoch *Fehler* ab. Die Bedingung ist somit, wenn die Bedingung (Funktion) einen Fehler produziert, dann wird *otherwise* ausgeführt.

## 2 Business Intelligence: Part I - Power Query

### 2.1 Introduction to Excel Tools

Im ersten Teil des Kurses wird besprochen welche Methoden es in der Microsoft Power BI Umgebung gibt. In der Business Version sind einzelne Komponenten integriert, welche vollintegriert in der BI-Version sind. In der Übersicht wird gezeigt, wie die einzelnen Komponenten aufgebaut sind.

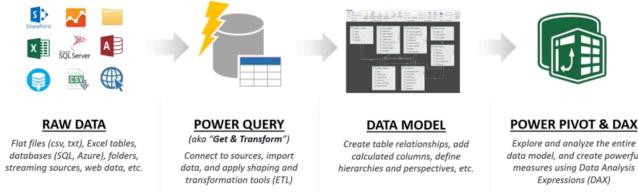


Abbildung 22:

- Datenbank**
- Access wird mehr als Datenbank gesehen. Welche Fragen noch offen sind, im Hinblick auf Power Query, ist: "Wie unterscheiden sich Access Queries von Power Query-Abfragen?".
  - Weitere Datenspeicherungen können in comma-separated values (.csv) oder Microsoft Excel Open XML Spreadsheet (.xlsx) Dateien liegen.
  - Es können aber auch Server wie SQL Server oder SharePoint angebunden werden.

**Query** Mit Hilfe der Abfragen können spezifische Fragen an die eingelesenen Daten gestellt werden. Diese werden in Excel geladen. Entweder werden die Daten direkt in Excel eingelesen und oder über Pivot Tabellen / Charts dargestellt oder sie können in das Datenmodell eingelesen werden. Per Query wird mit Prinzip Extract, Transform, Load (ETL) gearbeitet.

**Datenmodell** In dem Datenmodell werden Beziehungen zwischen verschiedenen Datenquellen oder spezifischer Tabellen bezogen. Die Daten werden mit einem speziellen Komprimierungsverfahren gespeichert. Gespeichert werden die Daten im Arbeitsspeicher, für die weitere Verarbeitung. Ergänzend zur Queries durch Power Query können auch weitere Spalten angefügt werden. Es ist wichtig zu vermerken, dass nur die Daten im Datenmodell gespeichert werden, die entweder selektiert durch die Query eingelesen wird oder direkt eingespeichert wird.

**PowerPivot** Dieses Option umfasst nicht nur die dynamischen Pivottabellen sondern auch die **Measures**. Power Pivot verwaltet nicht nur eine einfache Tabelle, sondern greift auf das komplette Datenmodell zu. Mit Measures können BI-Funktionen geschaffen werden. Diese werden mit Data Analysis Expressions (DAX) erstellt.

## 2.2 Query Editor - Tabellenblätter

Power Query bietet verschiedene Möglichkeiten an, Daten in Excel zu laden.

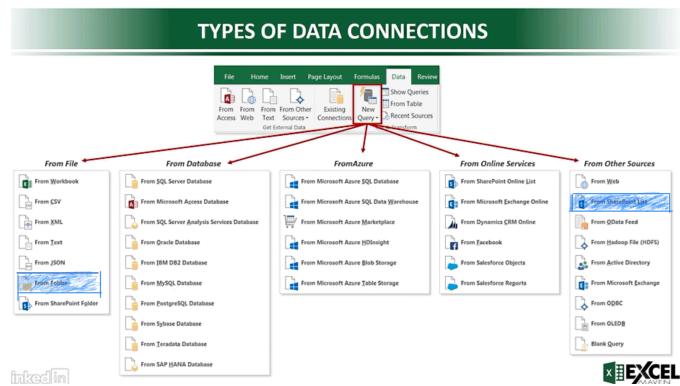


Abbildung 23:

Im ersten markierten Bereich wird drauf verwiesen, einen Pointer auf einen Ordner zeigen zu lassen. Ebenso ist es möglich, eine SharePoint-Seite anzusprechen.

Der Editor ist in drei Bereiche eingeteilt:

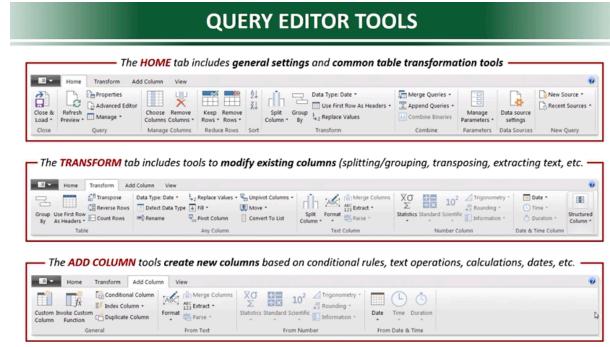


Abbildung 24:

- Home: Die Daten sind geladen und werden transformiert.
- Transform: Die Daten sind geladen und werden transformiert & verändert.
- Add Column: Die Daten sind geladen und werden transformiert & neue Daten werden angefügt.

### 2.2.1 Home

**Remove Duplicates** Diese Funktion ist nützlich, wenn man dynamische Indexfunktionen erstellen will. Zum Beispiel, kann für die

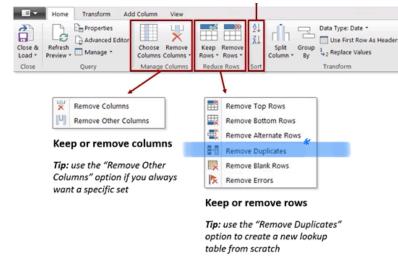


Abbildung 25:

**Proper Name** Es wird empfohlen die Queries auch als solche zu bezeichnen. Zum Beispiel "ETLName". Dies wird sich (vermutlich) bei Power Pivot erschließen.



Abbildung 26:

### 2.2.2 Transform

**Text Transformation** In dem Bereich unter dem Tab können Text-Einträge verändert werden. Die Funktionalität richtet sich vorrangig an Spalten vom String Typ.

- Trim - Leerzeichen vor dem Eintrag können gelöscht werden.
- Prefix - Textteile können angefügt werden.

- Groß- und Kleinschreibung kann angepasst werden
- Beispiel: Die Namen in der Liste können angepasst werden, falls die Vor- und Nachnamen nicht richtig großgeschrieben sind.

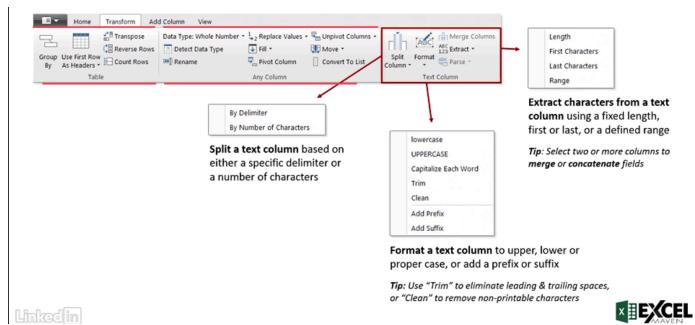


Abbildung 27:

### Zahlen Transformation

- Statistiken geben Werte zurück. Eine oder mehrere Spalten werden reduziert auf einen Wert. Die Statistik-Funktionen werden nicht in dem *Add Column* Tabellenblatt angezeigt. **Idee:** Statistik können bestimmt werden und per Verbindung gespeichert werden. Daraufhin werden diese in Power Pivot verwendet, um mit ihnen weiter zu rechnen. Den Wert kann in eine *Liste* oder eine *Tabelle* konvertiert werden.

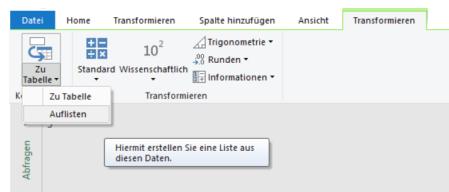


Abbildung 28:

- Standard, Wissenschaftliche oder andere Funktion für numerische Spalten lassen sich auf Spalten anwenden

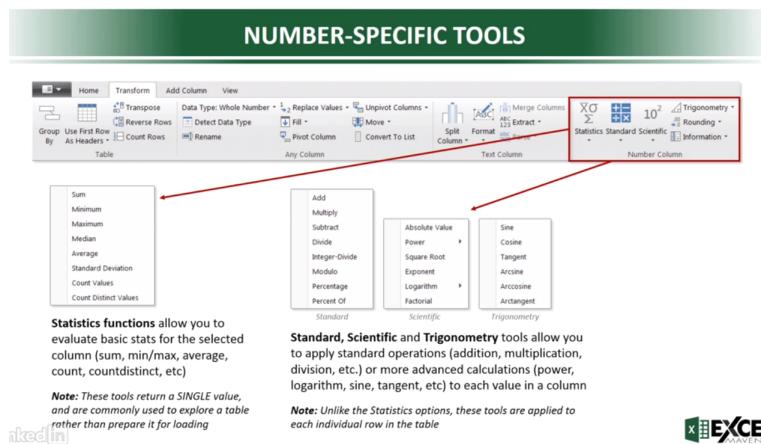


Abbildung 29:

- List Tool** Die Werte der Tabelle können nicht direkt verändert werden. Es kann nicht einfach in eine Zelle gegangen werden und ein Wert verändert werden. Mit List können aber ganz neue Spalten, von einer Satt aus, erstellt werden.

- Power M - List - Funktionen (siehe Dokument)

The screenshot shows the title 'List functions' and the date '8/6/2019 • 8 minutes to read'. Below the title, there is a brief introduction about the Power Query Formula Language (M) being a powerful 'mashup query language' optimized for building queries that mashup data. It is described as a functional, case sensitive language similar to F#, which can be used with Power Query in Excel and Power BI Desktop. A link to 'Power Query Formula Language (informally known as "M")' is provided.

The Power Query Formula Language (informally known as "M") is a powerful **mashup query language** optimized for building queries that mashup data. It is a functional, case sensitive language similar to F#, which can be used with [Power Query](#) in Excel and [Power BI Desktop](#). To learn more, see the [Power Query Formula Language \(informally known as "M"\)](#).

#### Information

FUNCTION	DESCRIPTION
<a href="#">List.Count</a>	Returns the number of items in a <a href="#">list</a> .
<a href="#">List.NonNullCount</a>	Returns the number of items in a list excluding null values
<a href="#">List.IsEmpty</a>	Returns whether a list is empty.

#### Selection

FUNCTION	DESCRIPTION
<a href="#">ListAlternate</a>	Returns a list with the items alternated from the original list based on a count, optional repeatInterval, and an optional

Abbildung 30:

- Auch hier kann man keine einzelnen Zellen verändern.

Wie kann man eine neue Tabelle in Power Query anlegen?

- Offen *Blank Power Query*
- Mit der Funktion = #date(2017, 1, 1) wird der erste Eintrag in die Liste gemacht.

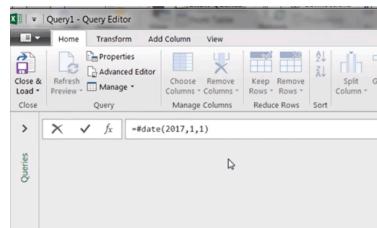


Abbildung 31:

Variablen werden in Power Query mit einem # begonnen.

The screenshot displays Power Query M code for 'IMDB Rating'. Annotations highlight several parts of the code:

- let block**: Points to the 'let' keyword at the beginning of the code.
- variable definitions**: Points to the definition of 'Data0' as a source of data.
- end of lines**: Points to the closing brace '}' at the end of the code.
- in block**: Points to the 'in' keyword followed by a variable name.
- function call**: Points to a function call within the code.

```

let
    Data0 = Web.Page(Web.Contents("http://www.imdb.com/chart/top"))
    #<removed>
    #>
    #>Changed Type = Table.TransformColumnTypes(Data0,{("","type text"), {"Rank & Title", type text}, {"IMDb Rating", type text}})
    #>Removed Other Columns = Table.SelectColumns(#>Changed Type, {"Rank & Title", "IMDb Rating"})
    #>Split Column by Delimiter = Table.SplitColumn(#>Removed Other Columns, "Rank & Title", Splitter.SplitTextByEachDelimiter({"\r\n"}, 1, 1))
    #>Renamed Columns = Table.RenameColumns(#>Split Column by Delimiter, {{"Rank & Title.1", Int64.Type}, {"Rank & Title.2", Int64.Type}})
    #>Renamed Columns1 = Table.RenameColumns(#>Renamed Columns, {"Rank & Title.1", "Rank & Title.2", "Year"})
    #>Renamed Columns2 = Table.RenameColumns(#>Renamed Columns1, {"Rank & Title.2", "Title", "Year"})
in
    #"Trimmed Text"

```

Abbildung 32:

Einfache Rechenoperationen sind auch möglich.

# 3 Business Intelligence: Part II - Data Modeling 101

## 3.1 Data Normalization

Um Datenmodell aufzubauen, ist es sinnvoll, dem Prinzip der **Separation** zu folgen. Daten die auf Sachverhalte mit verschiedenen Entstehungsgründen zurückzuführen sind, sollten auch in separate Tabellen stehen.

This Data Table contains "quantity" values, and connects to the Product table via product\_id.

This Calendar Lookup table provides additional attributes about each date (month, year, weekday, quarter, etc.)

This Product Lookup table provides additional attributes about each product (brand, product name, sku, price, etc.)

**MEET**

Abbildung 33:

In dem Beispiel sieht man, das die ergänzenden Daten zu dem Produkt in einer separaten Tabelle gespeichert werden sollte.

Wir können ebenfalls unterscheiden, zwischen **Daten Tabellen** und **LookUp table**. Was als Vermerk angebracht wird, ist sich Kalender Tabellen zu generieren. Dies ist deshalb sinnvoll, weil Power Pivot kein so leichte Oberfläche hat, wie Power Query um Daten zu extrahieren (Bis jetzt).

## 3.2 Data Model

### 3.2.1 Primary and Foreign Key

The primary key serves as a unique identifier. The foreign key allows to connect to the primary key.

These columns are **foreign keys**; they contain multiple instances of each value, and are used to match the **primary keys** in related lookup tables

These columns are **primary keys**; they uniquely identify each row of a table, and match the **foreign keys** in related data tables

Abbildung 34:

Im Datendiagramm werden Beziehungen zwischen foreign und primary key gelegt. Dabei zieht man vom foreign key feld auf das primary key feld.

Welche andere Variante funktioniert, ist unter *Design* zu sehen. Dabei erstellt man Beziehungen zwischen den Tabellen. Dies ist in Anlehnung an Power Query Zusammenführung.

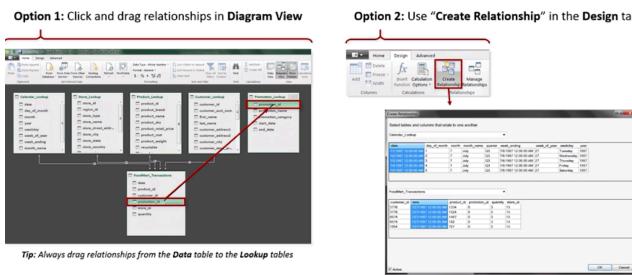


Abbildung 35:

### 3.2.2 Beziehungen

Beziehungen können auf zwei Arten erstellt werden. Die Beziehungen kann über die *Diagramm Ansicht*, durch ziehen der Einträge, erstellt werden. Zum anderen kann eine Beziehungen auch über den Entwurf-Tab erstellt werden

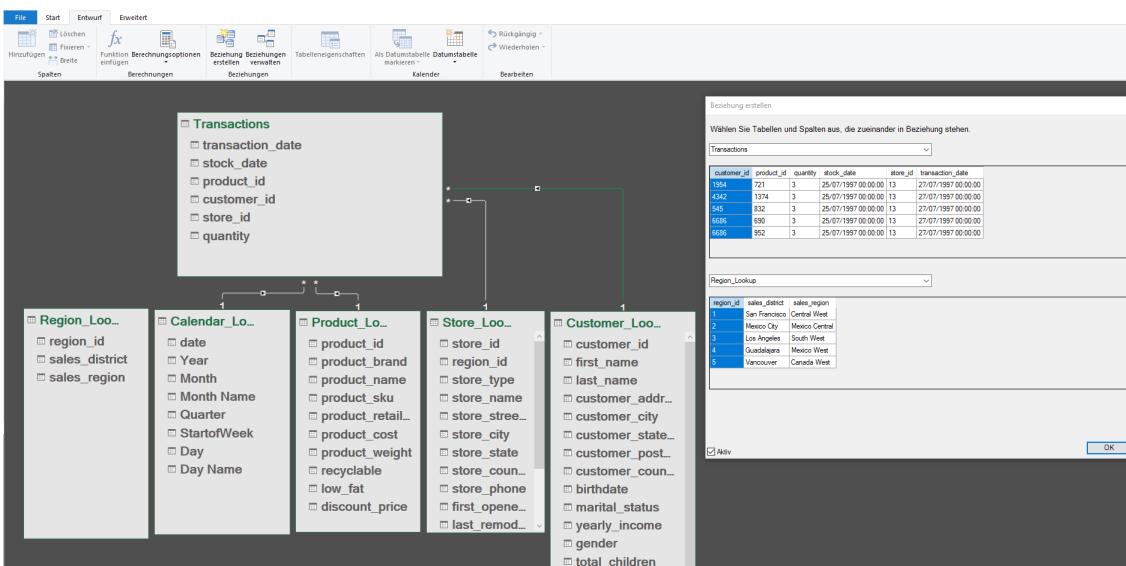


Abbildung 36:

In Power Query können verschiedene Schnittmengen gebildet werden. Es wird von *Merge* gesprochen. Es können jedoch nur  $n : 1$  Beziehungen erstellt werden. Eine  $m : n$  Beziehung von Power Pivot aktuelle nicht zur Verfügung gestellt. In Power Pivot kann nur eine Beziehungen zwischen zwei Tabellen erstellt werden. Eine weitere Beziehung wird dann als *inaktiv* verwaltet. Im Datendigramm wird dies mit einer gestrichelten Linie angezeigt.

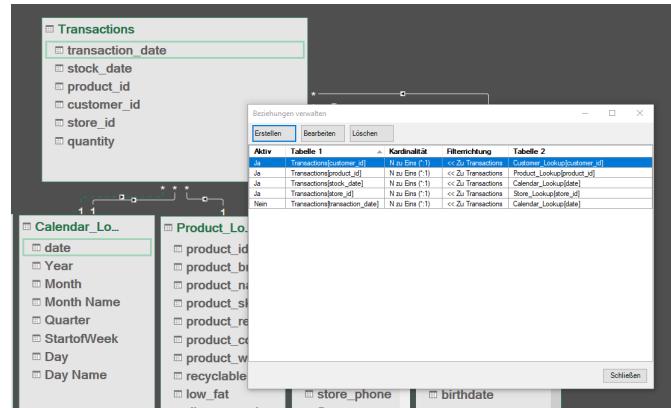


Abbildung 37:

**Star Model** In einem *star model* werden Beziehungen nur zwischen *primary key* Tabellen und *foreign key* Tabellen erstellt. Es bildet sich einen Sternenmuster aus Beziehungen heraus.

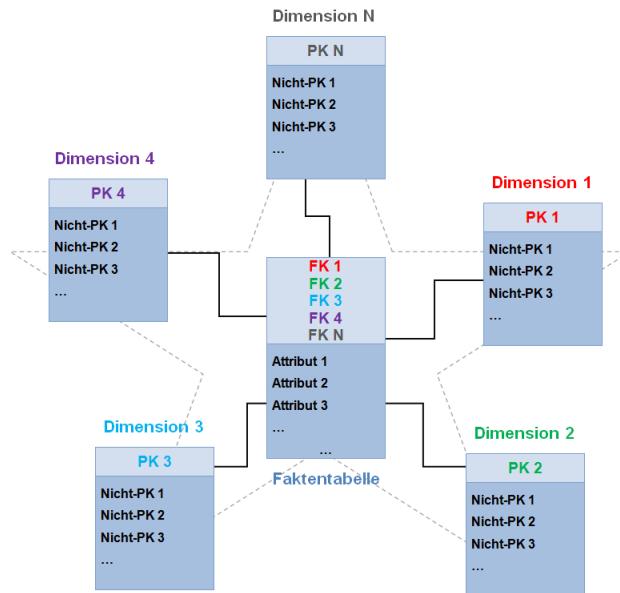


Abbildung 38:

**Snowflake Model** Ein *snowflake model* erlaubt Beziehungen zwischen *lookup tables*. Es entsteht ein immer weiter ausbreitendes Muster.

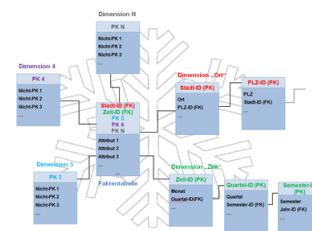


Abbildung 39:

In dem gewählten Beispiel wird **region\_id** in der primären Tabelle mit **region\_id** in der sekundären Tabelle verbunden. Eine verkettete Beziehung besteht jetzt zwischen der *Transaktionsliste* und der *Tabelle für Regionen*.

Dies wird über **store\_id** geregelt.

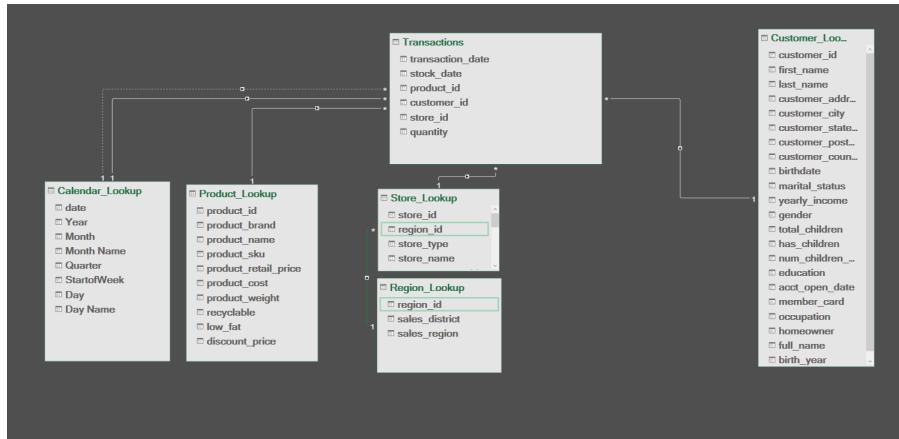


Abbildung 40:

### 3.2.3 Mehrere Daten Tabellen

Es wird stark empfohlen, zwischen *LookUp* und *Daten Tabellen* zu unterscheiden. Daten Tabellen werden aber nicht miteinander verbunden, sondern durch die  $n : 1$  Verbindung zu den jeweiligen LookUp Tabellen verbunden. Dabei wird die Pivot-Filterfunktion besser nutzbar gemacht, wie im späteren Verlauf genauer gezeigt.

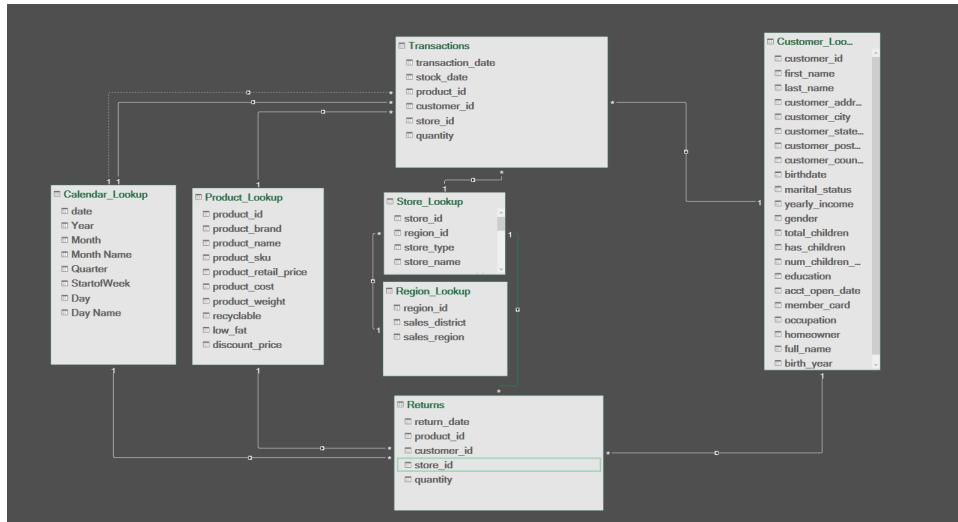


Abbildung 41:

### 3.2.4 Filter

Die Filterfunktion in den *PivotTable Fields* richtet sich nach den  $n : 1$  Richtung. Es kann nur sinnvoll gefiltert von dem LookUp table. **Filter und Rows** müssen aus den LookUP Tabelle kommen! Die linke Seite kommt somit aus den Lookup Tabellen und die rechte Seite von den Daten Tabellen.

## FILTER DIRECTION IS IMPORTANT (CONT.)

**Calendar\_Lookup filters flow "down" to both the Transactions and Returns tables, so we can filter or segment those metrics using any field from the Calendar table**

**Filtering by date in the Transactions table yields incorrect, unfiltered values from the Returns table, since filter context cannot flow "upstream" to the Calendar table**

Abbildung 42:

In dem Beispiel gibt es zwei Daten Tabellen: Transaction und Return. Die spezifischen Werte können in die  $\Sigma$  Values Spalte eingefügt werden. Die Felder für **Filter** können mit weiteren Spalten für die ID's in den Lookup Tabellen eingefügt werden. Dafür eignet sich auch ein *kontinuierlicher Kalender*. Führt man die Abfrage durch Power Query durch, so kann man die aktuellen und relevantesten Daten filtern. Diese werden dann extrahiert und ein eigener Kalender mit den extrahierten unter Bewertungen kann geschaffen werden.

Abbildung 43:

In dem Bereich für **Spalten** und **Zeile** gilt ebenfalls, soll eine Verfeinerung der Daten aufgegriffen werden & mehrere Werte werden aus unterschiedlichen Tabellen abgegriffen, so muss die Granulierung für jeden Daten Tabelle erfolgen. Dabei muss die Spalte aus der Lookup Tabelle kommen.

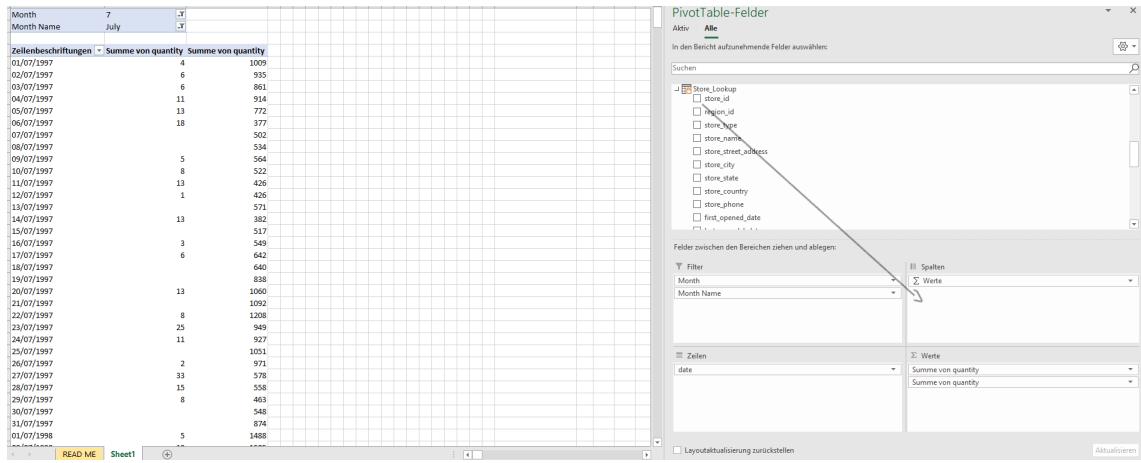


Abbildung 44:

Ein extrahieren aus einer der Daten Tabelle wird einen Fehler aufrufen, da die Filterung nicht über den Lookup Tabelle "durchfließt".

### 3.2.5 Hide fields from client tools

Im vorherigen Teil wurde der Fokus darauf gelegt, welche Bereich zum Filtern geeignet sind und welche nicht. Die Regel ist hier:

$$\text{Filter (Slice), Zeilen und Spalten enthalten } \textit{Foreign Key's}. \quad (3.1)$$

Ein fälschlicherweise Filtern eines primären Schlüssel (Spalte) wird verhindert, indem die primären Key's in der Daten Tabelle versteckt werden. Dies geschieht nicht im Modell selber, aber im Pivot Field Bereich.

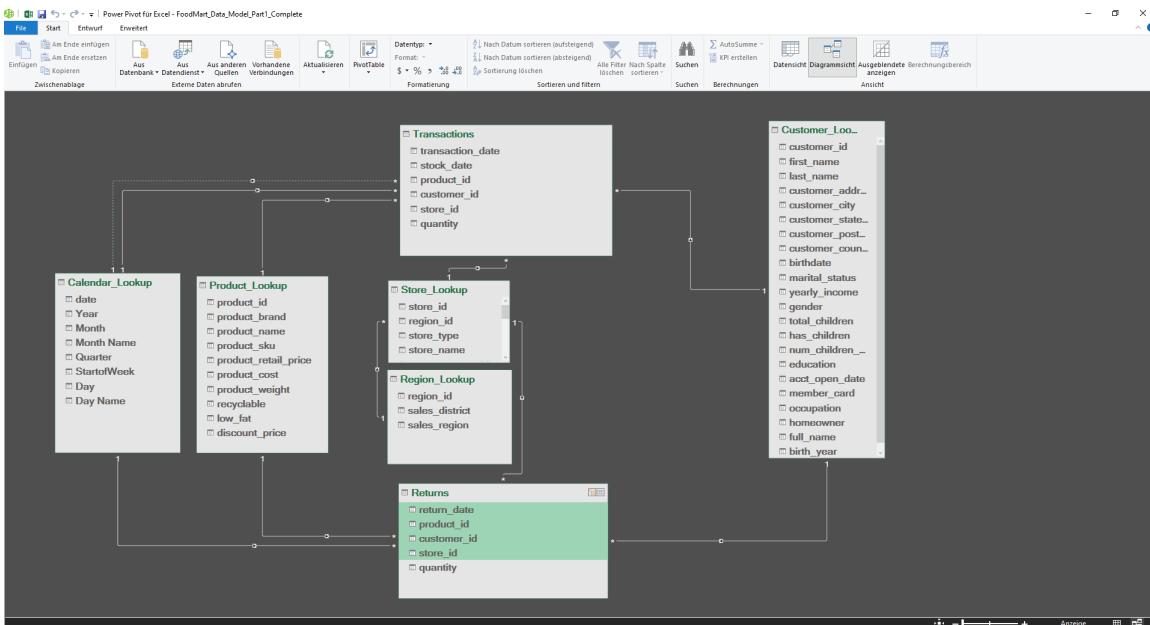


Abbildung 45:

Verwendet man die *auszublendenen Funktion*, werden die Key's nicht mehr angezeigt.

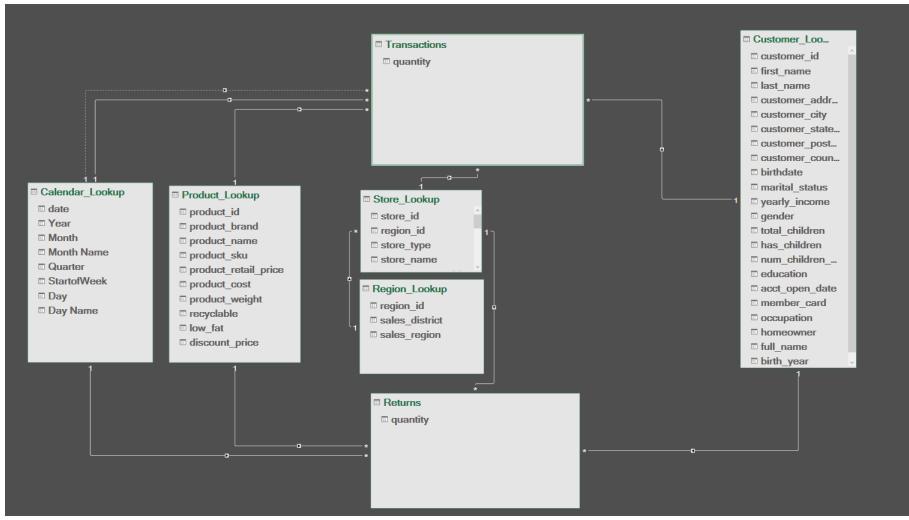


Abbildung 46:

Mit der Funktion an, wenden die Key's grau Schraffiert markiert.

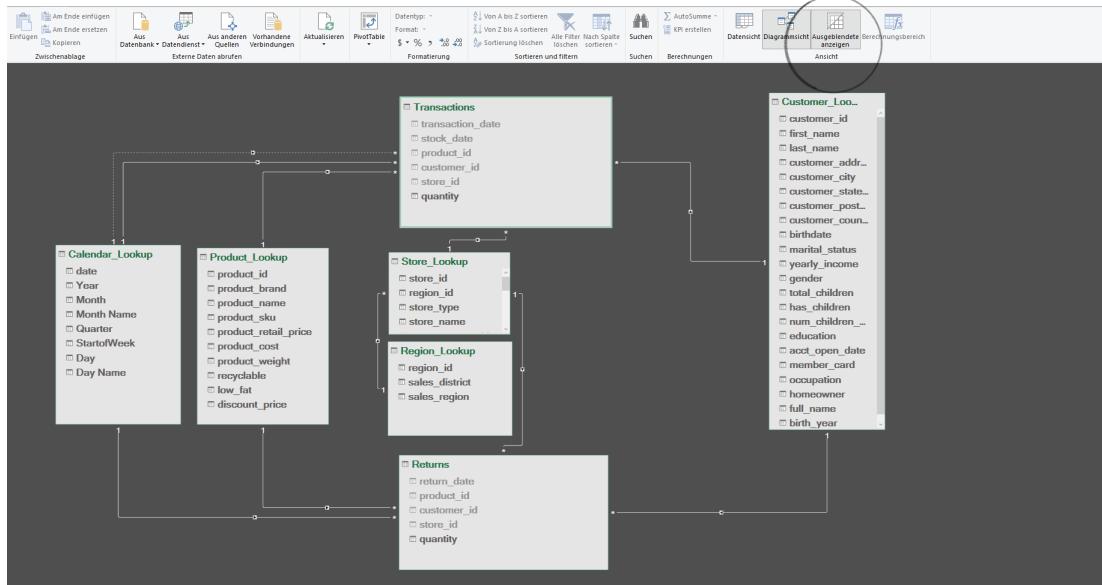


Abbildung 47:

### 3.2.6 Hierarchy

Mit Hilfe der Hierarchy können Gruppierungen von Foreign Key's erstellt werden, welche inhaltlich zusammenpassen. Dabei ist die Zeit eine naheliegende Tabelle.

**Hierarchies** are groups of nested columns that reflect multiple levels of granularity

- For example, a “**Geography**” hierarchy might include **Country**, **State**, and **City** columns
- Each hierarchy is treated as a **single item** in PivotTables and PivotCharts, allowing users to “drill up” and “drill down” through different levels of the hierarchy in a meaningful way



Abbildung 48:

Den Effekt wird im dritten Teil des Dreiteilers besprochen.

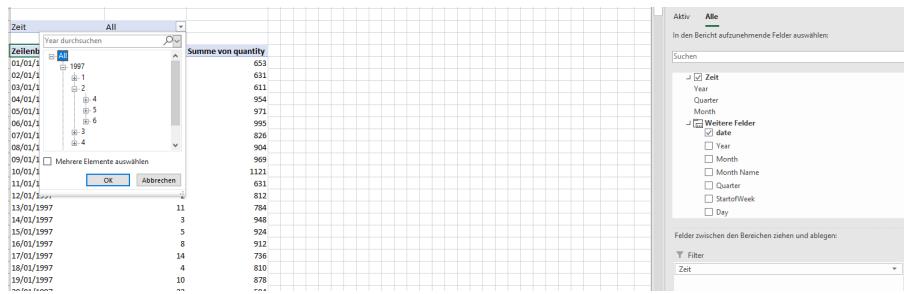


Abbildung 49:

## 4 Business Intelligence: Power Pivot and DAX

### 4.1 Power Pivot 101

#### 4.1.1 Oberfläche von Power Pivot

Einige Objekte der Oberfläche wurden schon in den vorherigen Kursen behandelt. Die Resultate können am Ende der Prozesses per *Pivot Tabelle* Schaltfläche in Excel eingeführt werden.

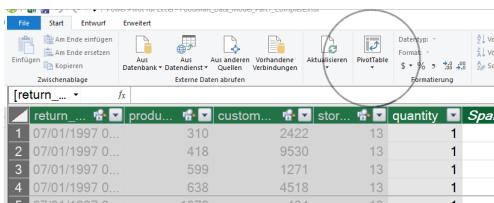


Abbildung 50:

Über diese Funktion können verschiedenen Kombinationen von Pivot Tabellen und Grafiken eingefügt werden.

#### 4.1.2 Berechnungen

#### 4.1.3 Funktion mit normalen Tabellen

Tabellen die nicht über das Datenmodell verarbeitet sind und direkt mit Power Pivot eingefügt werden können die die **Berechnete Feld** Funktion nicht nutzen. Diese sind ausgegraut.

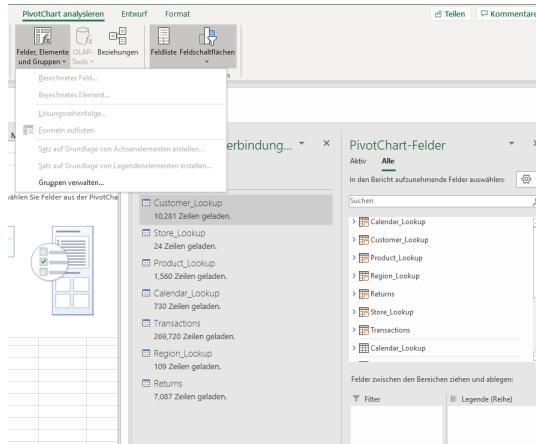


Abbildung 51:

Tabellen die nicht im Daten Modell liegen, können extra Felder zu der Tabelle hinzufügen ohne an den *Rohdaten* etwas zu verändern.

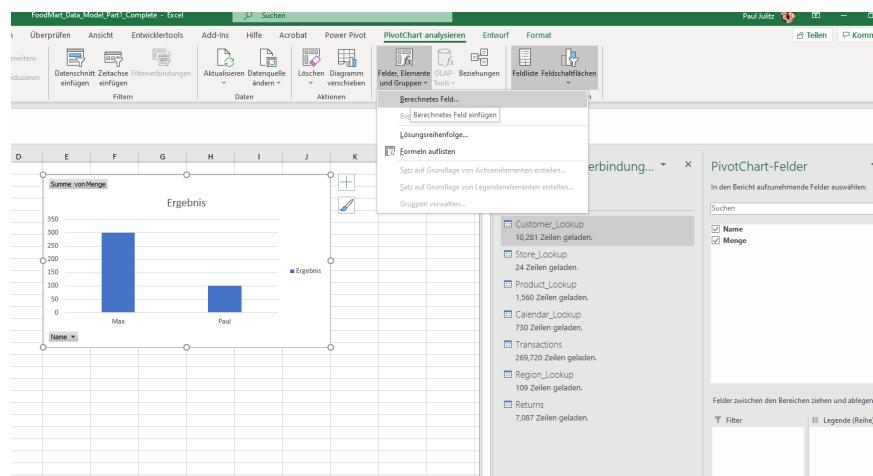


Abbildung 52: Öffnen

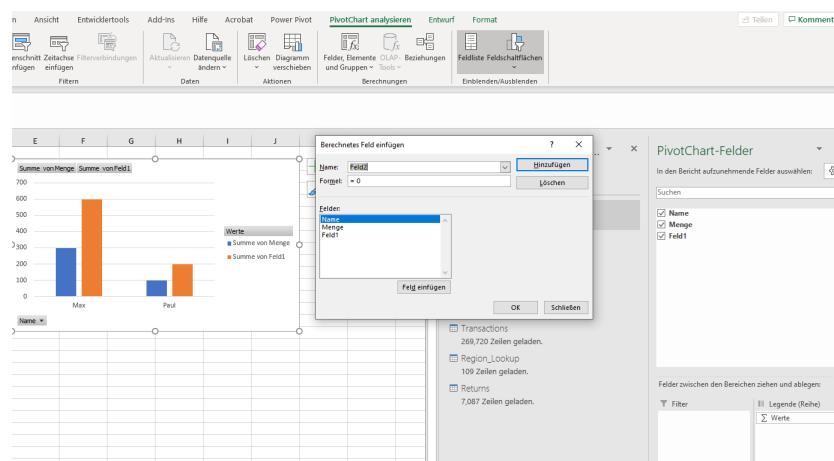


Abbildung 53:  $f(x) = 2 \text{ mal Menge}$

#### 4.1.4 Calculated Columns

Berechnende Spalten sind ähnlich aufgebaut, wie erweiternden Spalten in Tabellen. Es ist ratsam, diese gleich über Power Query anzulegen. Diese werden meist verwendet, um Slicer, Zeilen oder Spalten Granulierungen vorzunehmen.

Step 1: In the data model "Data View", choose a table and then select any cell in the "Add Column" section

Step 2: Enter a DAX function in the formula bar (we'll cover specific functions in the next section)

Step 3: Press "Enter", and all cells in the column will update

Abbildung 54:

Die benutzerdefinierten Spalten werden für einzelnen Tabellen verwendet. Die **Measures** werden für das gesamte Datenmodell verwendet. Alles was mit aggregierten Werten zu tun hat, wird darüber bestimmt.

produ...	custom...	stor...	quantity	Berechnete Spalte 1	Spalte hinzufügen
310	2422	13	1	1	2
418	9530	13	1	1	2
599	1271	13	1	1	2
638	4518	13	1	1	2
1078	484	13	1	1	2
75	8919	13	1	1	2
107	4261	13	1	1	2
189	3268	13	1	1	2
391	4507	13	1	1	2
475	1173	13	1	1	2
569	625	13	1	1	2
590	9929	13	1	1	2
804	545	13	1	1	2
959	8473	13	1	1	2
992	8164	13	1	1	2
1046	2548	13	1	1	2
1049	4261	13	1	1	2
1063	7416	13	1	1	2
1226	4342	13	1	1	2
1319	2548	13	1	1	2
1393	4507	13	1	1	2
1404	5782	13	1	1	2

Abbildung 55:

Ebenso ist zu berücksichtigen, dass es aggregierte Werte geben kann, diese werden dann aber für jeden Zeile verwendet.

In this case we've added a calculated column called `price_category`, which equals "High" if the retail price is >\$2, and "Low" otherwise (just like you would write in Excel!)

- Since calculated columns understand row context, a new value is calculated in every row based on that row's price
- This is a valid use of calculated columns. It creates a new row "property" that we can now use to filter or segment any related data within the model

Here we're using an aggregation function (SUM) to calculate a new column named `total_revenue`

- Since calculated columns do not understand filter context, the same grand total is returned in every single row of the table

`=SUM(Transactions[revenue])`

Abbildung 56:

Einige der DAX Funktion sind gleich von der Syntax und den Eingabewerten wie in Excel.

	total_chil...	num_children_at...	educa...	acct_type	DAY	er...	occup...	homeo...	has_children	full_name	birth_year	Spalte hinzufügen
	3	0	High Sch...	16/11/19	HOUR		Manual	N	Y	Bertha Ja...	1948	
	3	0	High Sch...	05/05/19	MINUTE		Manual	N	Y	Ole Weldon	1931	
	3	0	High Sch...	26/06/19	MONTH		Manual	N	Y	Paul Alcorn	1973	
	2	0	High Sch...	09/02/19	QUARTER		Manual	N	Y	Jared Bust...	1910	
	3	0	High Sch...	15/03/19	SECOND		Manual	N	Y	Margaret A...	1979	
	4	0	High Sch...	02/03/19	WEEK		Manual	N	Y	Vanessa T...	1930	
	2	0	High Sch...	04/06/19	YEAR		Manual	N	Y	Catherine ...	1966	
	1	0	High Sch...	05/03/1992	0...	Bronze	Manual	N	Y	Stacey Cer...	1943	
	4	0	High Sch...	17/05/1992	0...	Bronze	Manual	N	Y	Marin Coriell	1933	
	3	0	High Sch...	08/09/1992	0...	Bronze	Manual	N	Y	Deanna Sa...	1916	
	4	0	High Sch...	21/02/1991	0...	Bronze	Manual	N	Y	Joseph Th...	1968	
	4	0	High Sch...	12/06/1994	0...	Bronze	Manual	N	Y	Roberta St...	1919	
	1	0	High Sch...	24/04/1992	0...	Bronze	Manual	N	Y	Paula Toml...	1948	
	3	0	High Sch...	08/11/1991	0...	Bronze	Manual	N	Y	Troy Lipford	1953	

Abbildung 57:

## 4.1.5 Implicit Funktionen

Zieht man die Spalten von den Daten Tabellen in den *Werte Bereich* wir die Auswahl von verschiedenen Funktionen geben. Diese Funktionen nennt man **Implicit Function**.



Abbildung 58:

Die Funktionen sind fix vorgegeben. Warum oft darauf hingewiesen ist, *explicit function* zu nutzen, liegt daran, dass implizierte Funktionen fix und keine weiteren Änderungen an ihnen vorgenommen werden kann.

- Der Name der Funktion kann nicht verändert werden.
  - Zahlenformate zu den Funktionen kann nicht geändert werden. Hinweis: Soll ein gewisses Zahlenformat beibehalten werden, bietet Power Pivot explizite Funktionen an, das Format fix zu halten. Dabei wird bei jeder weiteren Verwendung der Funktion das Dateiformat angepasst.

#### 4.1.6 Explicit Funktionen - Measures

Einfache **explizite Funktionen** werden in die Tabellen in Funktionsbereich geschrieben.

### Abbildung 59:

Bildet man die gleiche implizierte Funktion nach, so funktioniert sie auch in gleicher Weise.

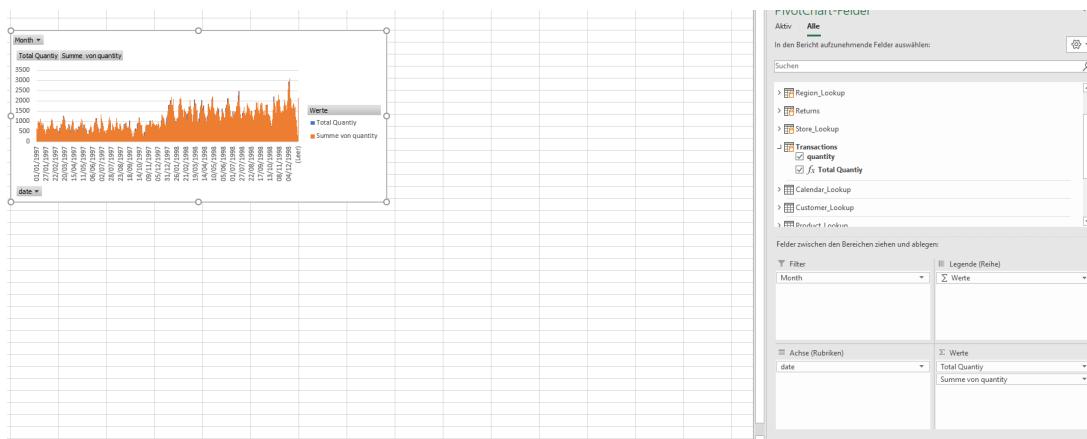


Abbildung 60:

Die Verwaltung der *expliziten Funktionen* kann auch über Excel unter *Power Pivot/ Measures* eingesehen werden.

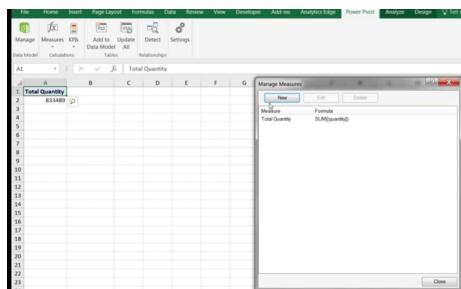


Abbildung 61:

Wir ein *Measures* erstellt, kann dies ebenfalls über das Eingabefeld erfolgen.

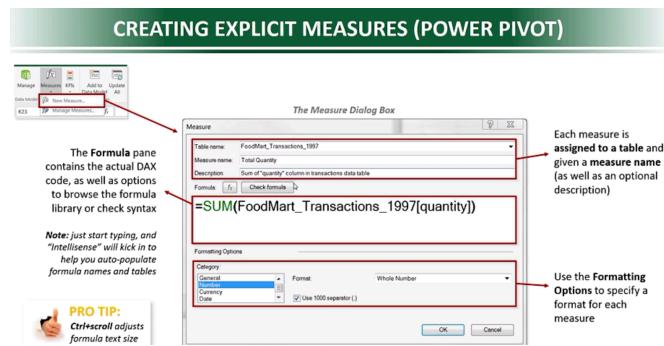


Abbildung 62:

Die Zuweisung hat nicht damit zu tun, dass nur Werte aus der Tabelle verwendet werden können, sondern damit zu welcher Tabelle es im Pivot Field angezeigt wird. Die *expliziten Funktionen* werden deshalb bevorzugt, weil

- eine genau Bezeichnung möglich ist.
- Measures aus verschiedenen Tabellen zusammengeführt werden können.
- Datentypen angepasst werden können.

A screenshot of the Microsoft Power BI interface. On the left is a PivotTable with columns A through M. The rows show data for cities like Acapulco, Bellingham, Bremerton, Camas, Querétaro, Guadalajara, Hidalgo, Los Angeles, Marida, Mexico City, Orizaba, Port Huron, Salem, San Diego, San Francisco, Seattle, Spokane, Tacoma, Vancouver, Victoria, Walla Walla, and Yakima. The last two rows are 'Gesamtergebnis' (Total Result) with values 833 489,00 and 833 489. The right side shows the 'PivotTable-Felder' pane with various lookup tables listed and the filter settings for the 'sales\_district' column.

Abbildung 63:

#### 4.1.7 Filterfunktion

Die Art und Weise wie Filter (Slicer), Zeilen und Spalten verwendet werden können, kann als Wasserfall-Diagramm verstanden werden. Die Filter werden in den Lookup Tabellen gesetzt. Über die Key's werden die Tabellen, welche in Beziehung zu der Lookup Tabelle stehen gefiltert. Soll ein Filter für mehrere Daten Tabellen angelegt werden, wo werden nämlich alle Tabellen gefiltert, welche in Beziehung stehen. Ein lokaler Slicer kann im Zweifelsfall über eine eigene Spalte in der Datentabelle erstellt werden. Die Filterfunktion erstreckt sich dabei über alle Spalten, die in der Lookup Tabelle zu finden sind, selbst wenn sie in der ursprünglichen Tabelle nicht angefügt wurden.

Erzeugte Slicer müssen mit anderen Verweisen verbunden werden, sodass die Daten upgedatet werden sollen. Sonst bleiben auch verknüpfte Koordinaten-Kategorien separat.

## 4.2 Basic DAX Function

### 4.2.1 Syntax und Operatoren

Die DAX Sprache ist eine Funktionssprache. Dies bedeutet, ein Measure muss eine Funktion enthalten, die eine Form der Aggregations zu lässt. Hingegen können Berechnende Spalte ohne Funktionen auskommen und einfache Operatoren verwenden.

Die gängigsten Operatoren für DAX sind:

DAX OPERATORS

Arithmetic Operator	Meaning	Example	Comparison Operator	Meaning	Example
+	Addition	$2 + 7$	=	Equal to	$[City] = "Boston"$
-	Subtraction	$5 - 3$	>	Greater than	$[Quantity] > 10$
*	Multiplication	$2 * 6$	<	Less than	$[Quantity] < 10$
/	Division	$4 / 2$	$\geq$	Greater than or equal to	$[Unit_Price] \geq 2.5$
$\wedge$	Exponent	$2 \wedge 5$	$\leq$	Less than or equal to	$[Unit_Price] \leq 2.5$
			$\neq$	Not equal to	$[Country] \neq "Mexico"$

Text/Logical Operator	Meaning	Example
&	Concatenates two values to produce one text string	$[City] & " " & [State]$
&&	Create an AND condition between two logical expressions	$(([State] = "MA") \&& ([Quantity] > 10))$
(double pipe)	Create an OR condition between two logical expressions	$(([State] = "MA")    ([State] = "CT"))$
IN	Creates a logical OR condition based on a given list (using curly brackets)	'Store Lookup'[State] IN {"MA", "CT", "NY"}



Abbildung 64:

Für *and* wird **&&** verwendet, und für *or* werden zwei vertikale Trennstriche verwendet. Für die Syntax der Funktionen kann Power Query verwandt werden. Die Referenz zu einer Spalte in einer Tabelle funktioniert gleich.

Variablen Namen werden aber anders gehandhabt. Namen mit Leerzeichen werden mit einfachen Anführungsstrichen umschlossen.

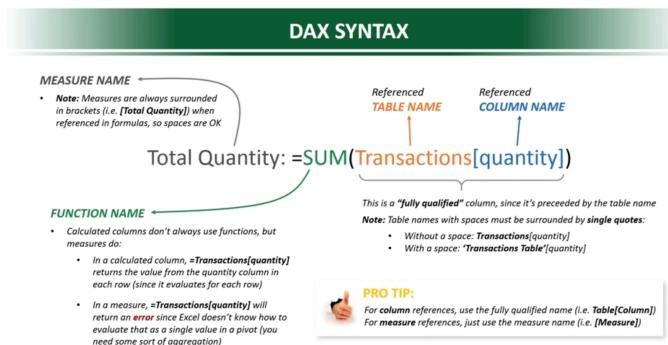


Abbildung 65:

Die Funktionen ähneln den die aus Excel und Power Query bekannt sind. Im weiteren Verlauf wird aber noch verstärkt auf einzelne Funktionen eingegangen.

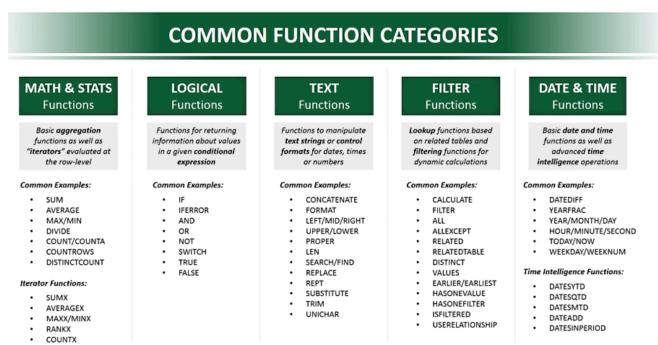


Abbildung 66:

#### 4.2.2 Function

Für numerische Werte können viele verschiedenen Spalten hinzugefügt werden.

Abbildung 67:

Werden mehrere Spalten hinzugefügt, so können Sortierungen über die Zeilenfilterung durchgeführt werden.

	Total Quantity	Avg Retail Price	Total Transactions	Unique Products	
Sort to Z	1.3 %	\$2.76	198	1	
Sort Z to A	0.9 %	\$3.75	350	2	
More Sort Options...	0.9 %	\$3.19	2,384	24	
Clear Filter From "product_brand"	1.3 %	\$2.88	376	2	
Label Filters	1.2 %	\$2.57	355	2	
Value Filters	1.0 %	\$1.95	3,345	8	
Search product_brand	1.1 %	\$3.89	5,727	29	
	0.8 %	\$2.36	714	6	
	0.8 %	\$2.31	6,000	36	
	1.0 %	\$1.75	4,073	22	
(Select All)	0.7 %	\$0.97	305	2	
ADT	1.0 %	\$2.18	5,770	31	
Afghan	0.8 %	\$1.90	3,224	19	
American	0.8 %	\$1.12	374	2	
Angelo	1.0 %	\$2.22	3,665	22	
Apple	0.9 %	\$2.41	800	5	
Atomic	0.9 %	\$1.99	3,854	20	
B&B Best	0.9 %	\$2.41	3,609	22	
	1.1 %	\$2.21	3,724	20	
Carrington	17,126	0.9 %	\$2.09	5,627	31
CDR	13,962	0.9 %	\$2.13	4,292	29
Colgate	4,132	0.9 %	\$1.34	3,313	8
Club	9,910	1.0 %	\$2.31	3,222	20
Colony	4,589	0.9 %	\$2.06	1,492	9
Colossal	2,692	1.2 %	\$2.64	850	5
Consolidated	9,806	0.9 %	\$1.98	3,178	19
Commodity	16,483	0.9 %	\$2.23	5,382	32
	**	**	**	**	1

Abbildung 68:



Abbildung 69:

Einfache Logische Funktionen verhalten sich wie schon oben beschrieben. Bei *And()* und *Or()* Funktionen können auch mit den dazugehörigen Operatoren beschrieben werden. Sollen mehrere Menge zusammengefasst werden, nimmt man die Operatoren.



Abbildung 70:

**Switch Function** Ein einfacher Austausch von Werten.

### SWITCH & SWITCH(TRUE)

**SWITCH()** Evaluates an expression against a list of values and returns one of multiple possible result expressions

=SWITCH(<expression>, <value1>, <result1>, <value2>, <result2>, ... <else>)

Any DAX expression that returns a single scalar value, evaluated multiple times (for each row/constant)

Examples:

- Calendar\_Lookup(month\_num)
- Product\_Lookup(product\_brand)

List of values produced by the expression, each paired with a result to return for rows/cases that match

Examples:

```
=SWITCH(Calendar_Lookup(month_num),
    1, "January",
    2, "February",
    etc...
```

```
=SWITCH(TRUE),
    [retail_price]<5, "Low Price",
    AND[retail_price]>=5, [retail_price]<20, "Med Price",
    AND[retail_price]>=20, [retail_price]<50, "High Price",
    "Premium Price")
```

Abbildung 71:

Switch erlaubt auch eine Variante, in der mehrere Überprüfungen vorgenommen werden können. In der ersten Übergabe wird *true()* übergeben.

TEXT FUNCTIONS		
LEN()	Returns the number of characters in a string	=LEN(<text>)
CONCATENATE()	Joins two text strings into one	=CONCATENATE(<text1>, <text2>)
LEFT/MID/ RIGHT()	Returns a number of characters from the start/middle/end of a text string	=LEFT/RIGHT(<text>, <num_chars>) =MID(<text>, <start_num>, <num_chars>)
UPPER/LOWER/ PROPER()	Converts letters in a string to upper/lower/proper case	=UPPER/LOWER/PROPER(<text>)
SUBSTITUTE()	Replaces an instance of existing text with new text in a string	=SUBSTITUTE(<text>, <old_text>, <new_text>, <instance>)
SEARCH()	Returns the position where a specified string or character is found, reading left to right	=SEARCH(<find_text>, <within_text>, <start_num>, <NotFoundValue>)

Abbildung 72:

## Text Funktionen

### 4.3 Advanced DAX Function

Die bisherigen Funktionen finden sich in einer oder anderen Form in Excel oder auch der M-Query Sprache wieder. Es gibt aber auch komplexerer Funktion in DAX. Die *Calculate* Funktion. Diese erlaubt komplizierte Berechnungen, basierend an verschiedenen Filtern. Dies ist nützlich, wenn Berechnungen nur für bestimmte Felder ausgeführt werden sollen.

CALCULATE	
CALCULATE()	Evaluates a given expression or formula under a set of defined filters
=CALCULATE(<expression>, <filter1>, <filter2>, ...)	
Name of an existing measure or a formula for a valid measure	List of simple Boolean (True/False) filter expressions (note: these require simple, fixed values; you cannot create filters based on measures)
Examples: • [Total Transactions] • SUM[Transactions[quantity]]	Examples: • Store_Lookup[store_country] = "USA" • Calendar[Year] = 1998 • Transactions[quantity] >= 5
 PRO TIP: CALCULATE works just like SUMIF or COUNTIF, except it can evaluate measures based on ANY sort of calculation (not just a sum, count, etc); it may help to think of it like "CALCULATEIF"	

Abbildung 73:

#### 4.3.1 Calculate

Dies Funktion nimmt in als erstes Argument eine Variable vom Typ table auf. Als zweites Argument wird ein Filter eingesetzte. Dabei wird die Tabelle, Stream up, gefiltert. Dieser Filter wird dann weiter gegeben und die Berechnungen im ersten übergebenen Tabelle oder Tabellen berechnet. Es ist eine verbesserte Variante der *Sumif*-Funktion. Die Funktionsweise für DAX Funktionen ist aber wichtig zu verstehen. Die Filter in der dazugehörigen Tabellen werden aktiviert und diese Filter werden für das Measure komplett im Datenmodell weitergegeben. Die Berechnungen mit *n*-Tabellenspalten im ersten Bereich werden dann durchgeführt, unter Berücksichtigung der Filter.

Wie man sieht werden vorherige, übergebene Filter überschrieben, weswegen in jeder Zeile der gleiche Wert steht.

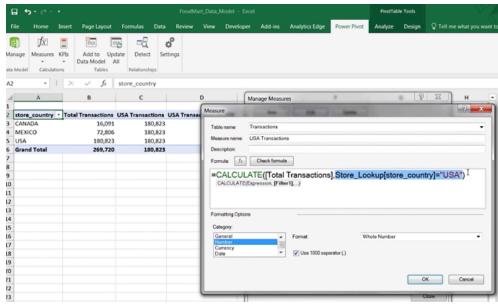


Abbildung 74:

Der Filter, als zweites Argument, kann auch eine Tabelle enthalten. In diesem Fall wird die Tabelle, die gefiltert wird, ersetzt und die Tabelle, welche durch die Filter-Funktion gefiltert wurde, ersetzt temporär die angegebene Tabelle.

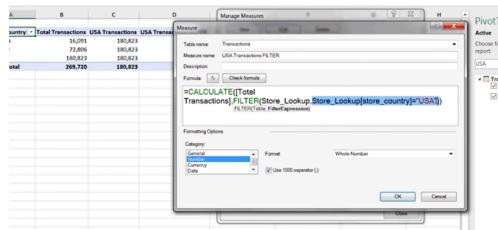


Abbildung 75:

Mit der Filter-Funktion existieren die gesuchten Filter nicht mehr und die Zeilen bleiben gleich.

store_country - Total Transactions, USA Transactions, USA Transactions FILTER			
1	Canada	56,078	180,823
2	Mexico	72,806	180,823
3	USA	180,823	180,823
4	Grand Total	209,720	180,823

Abbildung 76:

#### 4.3.2 Filter

Die Filterfunktion erhält als Input eine Tabelle. Die Filterfunktion wird mit Angaben von Spezifikation von Spalten erstellt. Der Funktion gibt eine Tabelle wieder.

**FILTER**

**FILTER()** Returns a table that represents a subset of another table or expression

=FILTER(<table>, <filter expression>)

Table to be filtered      A Boolean (True/false) filter expression to be evaluated for each row of the table

Examples:

- Store\_Lookup
- Product\_Lookup

Examples:

- Store\_Lookup[store\_country] = "USA"
- Calendar[Year] = 1998
- [retail\_price] > AVERAGE[retail\_price]

**HEY THIS IS IMPORTANT!**

FILTER is used to add filter context on top of what's already defined by the PivotTable layout.

Since FILTER returns a table (as opposed to a scalar), it's almost always used as an input to other functions, like enabling more complex filtering options within a CALCULATE function (or passing a filtered table to an iterator like SUMX).

Abbildung 77:

Wie kann man **Ergänzende-Slicer** in eine ein Datenmodell einfügen, ohne es mit den restlichen Tabellen zu verbinden → mit der Filterfunktion. Die Tabelle Der Filter in der Filterfunktion kann durch eine extra-eingefügte Tabelle gebunden werden.

1. Die MAX-Funktion gibt nur einen Wert wieder:

# MAX

04/19/2019 • 2 minutes to read • [Edit](#)

Returns the largest value in a column, or between two scalar expressions.

## Syntax

DAX	<code>MAX(&lt;column&gt;)</code>	<a href="#">Copy</a>
DAX	<code>MAX(&lt;expression1&gt;, &lt;expression2&gt;)</code>	<a href="#">Copy</a>

## Parameters

Term	Definition
column	The column in which you want to find the largest value.
expression	Any DAX expression which returns a single value.

Abbildung 78:

Die Einbindung läuft über zwei Measure.

**PRO TIP: FILTERING WITH DISCONNECTED SLICERS (PART 2)**

**STEP 4:** Place your new table on the pivot as a slicer:

**STEP 5:** Create a measure to capture the slicer selection, then reference it in a **FILTER** statement within **CALCULATE**:

The **Transactions Under Price Threshold** measure calculates Total Transactions when the product price is below the selected threshold

Abbildung 79:

Hinweis: Dies Funktion wäre auch sinnvoll, wenn es um die Prämienberechnung geht. In dem man die einzelnen **Threshold** angibt. Zusätzlich, es können auch Slicer direkt über die Ansicht der Pivot-Tabellen angezeigt werden.

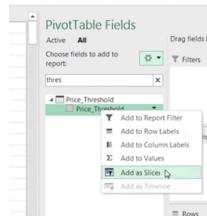


Abbildung 80:

### 4.3.3 All

Die All-Funktion ignoriert die bisher angewendeten Filter zu einer Tabelle. Dies erlaubt Berechnungen vorzunehmen, welche nicht ändern sollen, wenn interaktiv Filter angewandt werden.



Abbildung 81:

Angenommen, es ist verlangt, eine Tabelle mit prozentualen Verhältnissen zu zeigen.

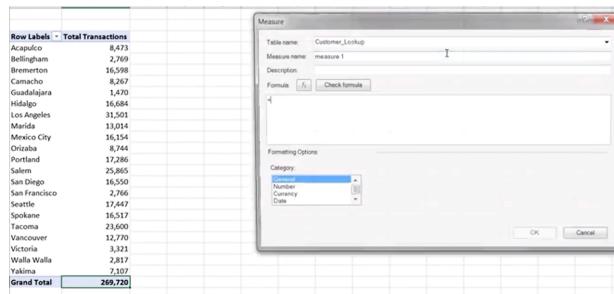


Abbildung 82:

Dies ist nur möglich, wenn die Summe am Ende gleich bleibt, wenn man möchte, dass durch Selektion die Prozente sich nicht verändern. Die All-Funktion erlaubt, die Verhältnisse bei zu behalten, selbst wenn die Zeilen gefiltert werden. Zum Beispiel könnte das Measures heißen:

$$\text{StatisProzent} = \frac{[Transaction]}{\text{Sum}(All([Transaction]))} \quad (4.1)$$

#### 4.3.4 Iterator (X) - SumX()

Das Grundprinzip von Measure ist, dass die Funktion einen Wert und keine Tabelle oder Spalte wieder gibt. Eine Aggregation von Text ist somit im einzelnen nicht möglich, außer, eine klvere Funktion wird geschrieben.

*Die Steuerung wird über die Filter gesteuert!*

Die Iteratorfunktionen sind nur erweiterte Funktionen ihrer zugrundeliegenden, einfachen Funktionen. Als Beispiel, die *Sum()* Funktion kann nur ein Aggregat einer Spalte sein.

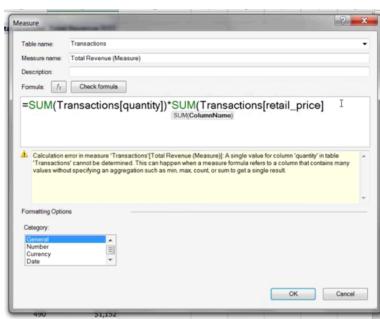


Abbildung 83:

Dabei wird erst die komplette erste Spalte und danach die komplette zweite Spalte aggregiert. Erst dann, werden die beiden Skalare miteinander multipliziert. Bei der Iterator Funktion wird, wird Zeile für Zeile summiert

und im Anschluss aggregiert. Dies macht auch die *Calculate()* Funktion. Dabei wird nur zusätzlich ein Filter benötigt.

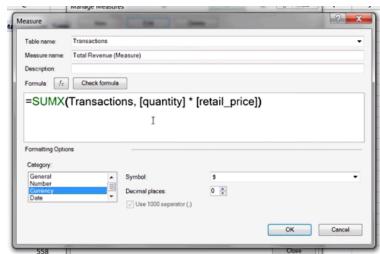


Abbildung 84:

Die *SumX()* Funktion verlangt eine Tabelle, von welcher die Berechnungen ausgehen. Es kann aber mit der *Related()* Funktion Daten aus anderen Tabellen gezogen werden.

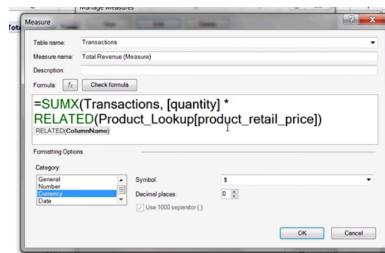


Abbildung 85:

#### 4.3.5 Iterator (X)- RankX()

Die *RankX()* Funktion ist eine iterative Funktion, weil sie erst die komplette Spalte durchlaufen muss, bevor jeder Wert in Reihe gebracht werden. Die Funktion gibt eine Liste von Zahlen wieder. Diese gibt an.

### RANKX

12/10/2018 • 2 minutes to read • ↗

Returns the ranking of a number in a list of numbers for each row in the *table* argument.

#### Syntax

DAX

```
RANKX(<table>, <expression>[, <value>[, <order>[, <ties>]]])
```

Abbildung 86:

Es ist dabei wichtig, zu berücksichtigen, dass die Filterfunktion einen Einfluss auf die Funktion hat. Die Funktion *All()* erlaubt, die Filter auszublocken und einen Ordnung der Werte zu ermöglichen, obwohl sie gefiltert sind.

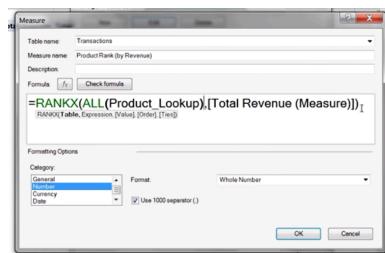


Abbildung 87:

Weitere Unterteilungen kann wie folgt aussehen.

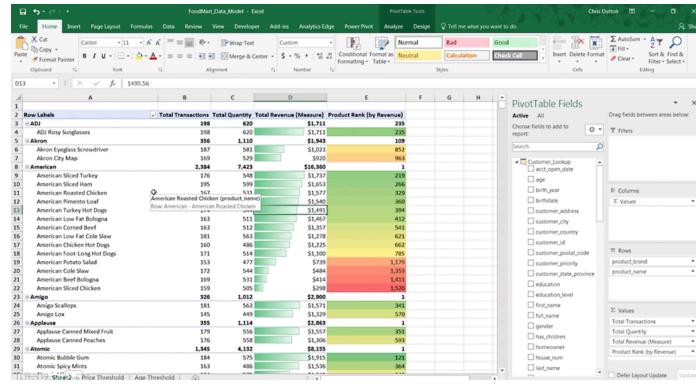


Abbildung 88:

#### 4.3.6 Time-Intelligence Function

Diese Funktionen verhalten sich ähnlich zu denen in  $M$ .

**TIME INTELLIGENCE FORMULAS**

Time Intelligence functions allow you to easily calculate common time comparisons:

<b>Performance To-Date</b>	=CALCULATE(<measure>, DATESYTD(Calendar[Date])) Use DATESQTD for Quarters or DATESMTD for Months
<b>Previous Period</b>	=CALCULATE(<measure>, DATEADD(Calendar[Date], -1, MONTH))
<b>Running Total</b>	=CALCULATE(<measure>, DATESINPERIOD(Calendar[Date], MAX(Calendar[Date]), -10, DAY))

**PRO TIP:** To calculate a moving average, use the running total calculation above and divide by the # of intervals!

Abbildung 89:

Die erste Funktion erlaubt in Abschnitten zu untergliedern. Dabei wird ein spezifischer Filter für das Datum verwandt. Nachdem die ersten Funktionen geschrieben sind, können die Funktionen einfach miteinander kombiniert werden. Die Funktion *DateAdd()* erlaubt:

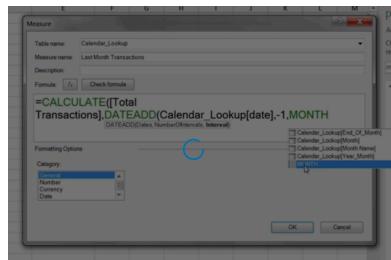


Abbildung 90:

Setzt man die Funktion ohne mit mit ins Verhältnis, so ergibt sich:

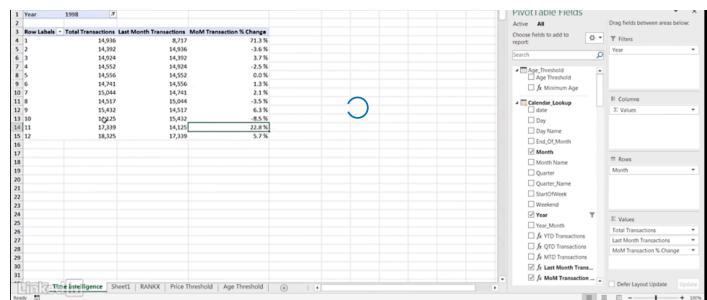


Abbildung 91:

Gleiches gilt auch für die Funktion für Perioden *Dateinperiode()*. Eine fixe Zeitspanne kann somit erstellt werden.

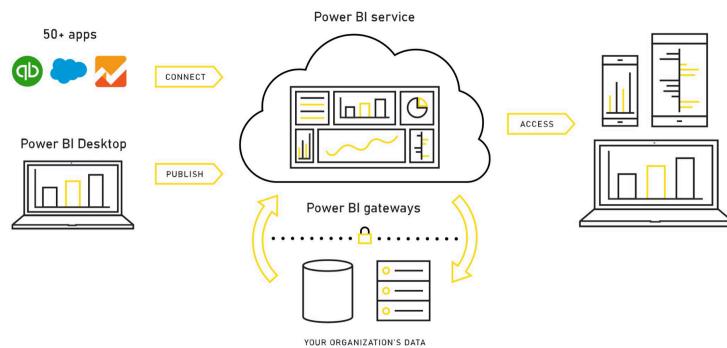
## 5 Power BI Dataflow Essentials

Dashboard werden über Power BI Desktop erstellt. In der hochgeladenen Datei (Power BI Desktop File) befinden sich zwei Komponenten

- Front End Dashboard
- Dataset

### 5.1 Gateways and Gateway Clusters

Um Daten einer Organisation mit der BI Cloud zu verbinden, benötigt es **On-Premise Gateways**. Diese werden auf den Server einer Organisation verbunden. Diese geben einen Zugang zum System. Die Daten müssen nicht auf der Recheneinheit liegen, auf welcher das Gateway installiert ist.



Im AdminCenter wird festgelegt, wer in der Organisation Gateways installieren kann. Die Einschränkung kann sogar aufgehoben werden, und alle User in einer Organisation können Gateways anlegen.

The screenshot shows the Microsoft Power Platform Admin Center interface. The main area displays a table of 'Data Gateways'. The columns include 'Gateway Cluster Name', 'Contact info', and 'Administrators'. Several entries are listed, such as 'AnnualUpgradeGW' (Contact: annalj@biatipdfmsitwus.onmicrosoft.com, Administrators: 'fresh office group Z'), 'Contoso gateway cluster' (Contact: benn@biatipdfmsitwus.onmicrosoft.com, Administrators: 'fresh office group'), and 'GatewayInstallTest' (Contact: annalj@biatipdfmsitwus.onmicrosoft.com, Administrators: Anna). On the right side, there's a section titled 'Manage Gateway Installers' with a note about managing who can install gateways. It includes a toggle switch for 'Restrict Users in your organization from installing gateways' (set to 'On'), a search bar for 'Users who can install gateways', and a list of current gateway installers (Anna Eason and Tim Mariani) with their email addresses.

Die Funktionalität Active Directory (AD) Group hinzuzufügen ist zum Stand 2019 noch nicht verfügbar. Wird das Recht entzogen, Gateways zu installieren, beeinflusst dies nicht die bereits administrierten Gateways. Stehen nicht die benötigten Rechte einem User zu, wird die Fehlermeldung

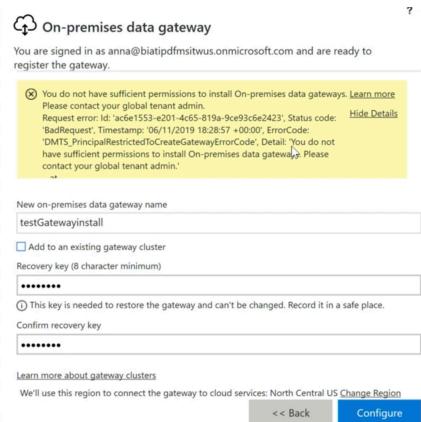


Abbildung 92: Contact Tenant Admin

Es besteht die Möglichkeit mehrer Gateways einem Cluster zu zuordnen. Diese Funktion ist, dass mehre Knotenpunkte geschaffen werden. Ist eine nicht mehr verfügbar, können auf andere Gateways in einer Cluster zugegriffen werden.

## High Availability

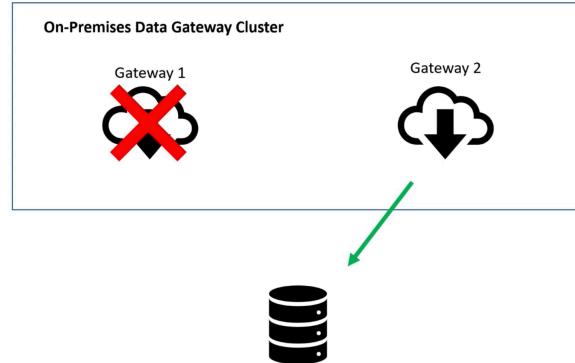


Abbildung 93: High Availabilty

In Admin Center können die verschiedenen Gateways Cluster gemanaget werden. In der letzten Spalte ist zu sehen, wieviele Gateways einem Cluster zugeordnet werden.

The screenshot shows the "Data gateways (preview)" section of the Power Platform admin center. It lists various gateway clusters and their details:

Gateway cluster name	Contact info	User	Gateways
Contoso free	pwine@contosofree.com	Item	1
Contoso Gateway	pwine@contosogateway.com	Adam, Patrick	2
HPCGIC	phcic@powerplatform.com	Patrick	1
Malco	halco@powerplatform.com	Patrick	1
Oliver	pholiver@powerplatform.com	Patrick	1
pwine@pwine.com	pwine@pwine.com	Patrick	1
SDGamerHQ	pholiver@powerplatform.com	Patrick	1
Surfacebook	pholiver@powerplatform.com	Patrick	1

Abbildung 94: admin.powerplattform.com

In dem Beispiel sind es zwei

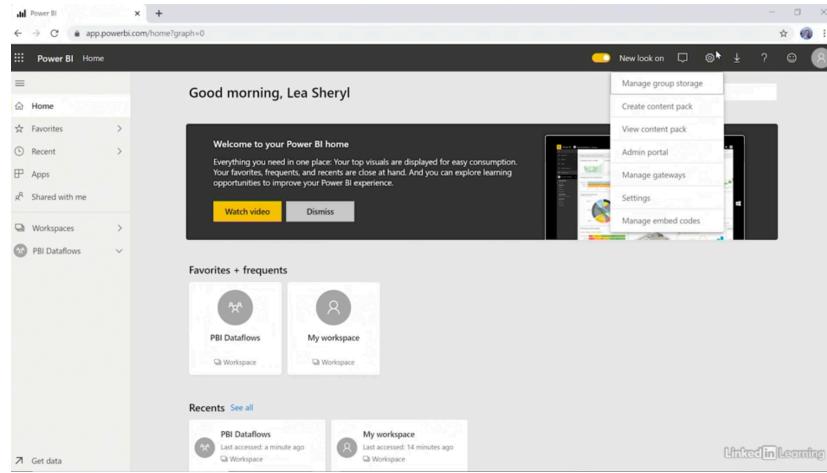
The screenshot shows the "Gateway cluster: Contoso Gateway" page. It displays the following table:

Name	Device	Version
Contoso Gateway (Pri...)	GUYINACUBESQL	3000.31.3
Contoso Gateway 2	GIACDESKTOP	3000.31.3

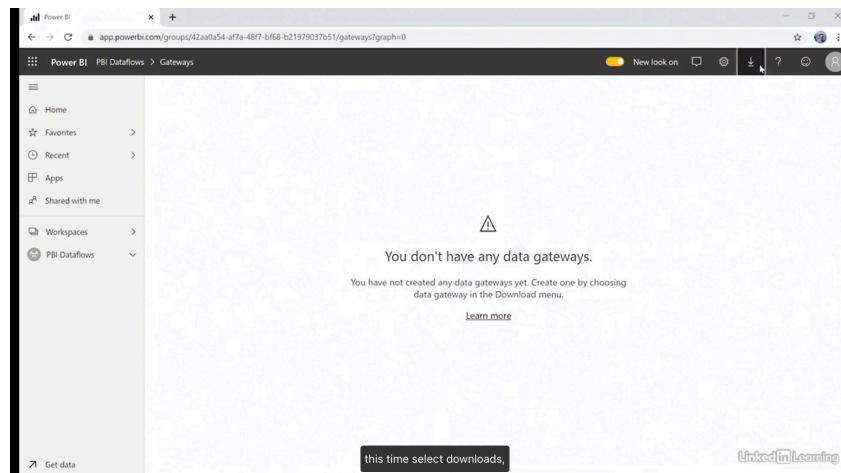
Dabei ist zu sehen, auf welchen Maschinen diese installiert sind.

Auf der Hauptseite befindet sich unter den Einstellungen die Auswahl **Manage Gateway**.

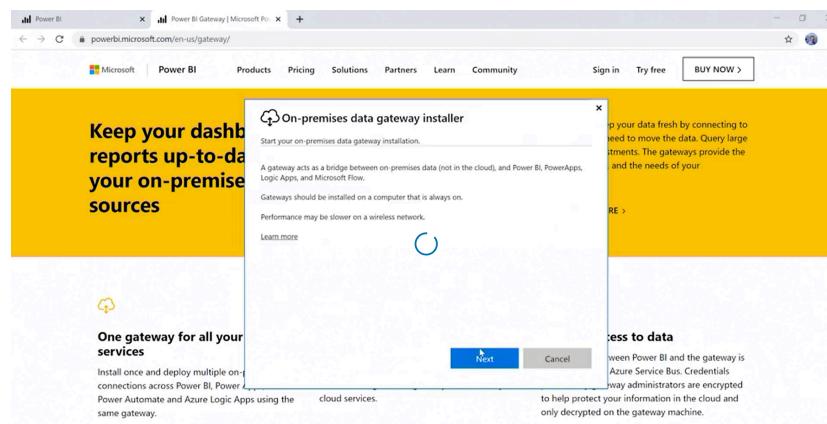
Unter diesem Punkt werden alle **Data Gateways** gemanagt.



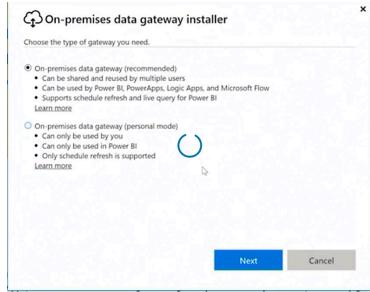
Wenn noch keins erstellt wurde, wird es als leer angezeigt.



The gateway is a on-premisse connection to local data. This briges local data and cloud connection to Power BI and co.

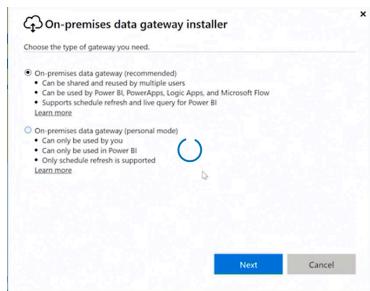


For optimization, the connection can be used by others and or for live connection



The new gateway can be added to an already existing cluster. Also a key is necessary. The email address used to set up the account is tied to admin account, see [Link](#)

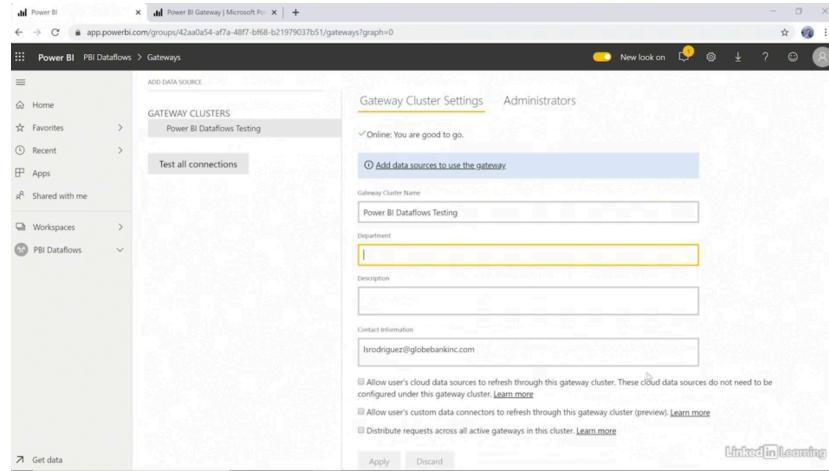
Wenn das Gateway erstellt ist, wird es auf der Landing Page angezeigt



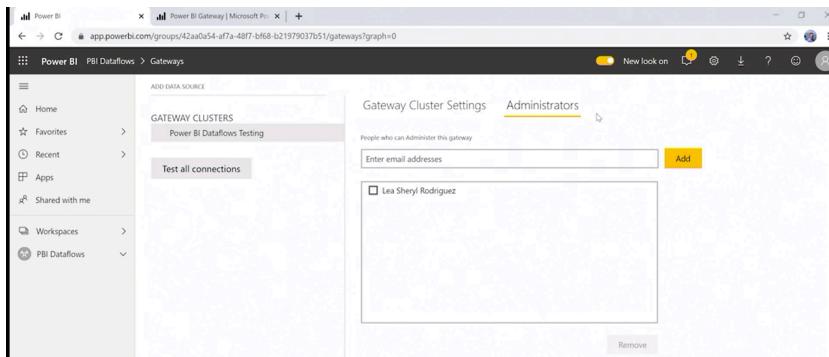
Connections können zum Gateway hinzugefügt werden.

Die Funktion hinter mehreren Gateways in einem Cluster ist, dass eine Load-Balance im Netzwerk hergestellt wird und eine robustere Übermittlung der Daten sicher gestellt wird. In der Verwaltung eines Clusters/ Mehrer Gateways bestehen mehrere Optionen:

- Daten Laden über das Data Gateway
- Durch die User neue Data Connections durch die User anlegen
- Verteile die Datenlast über alle Gateways



Im Administrativen Bereich können User hinzugefügt werden, welche das Gateway administrieren.



## 5.2 Dataflows

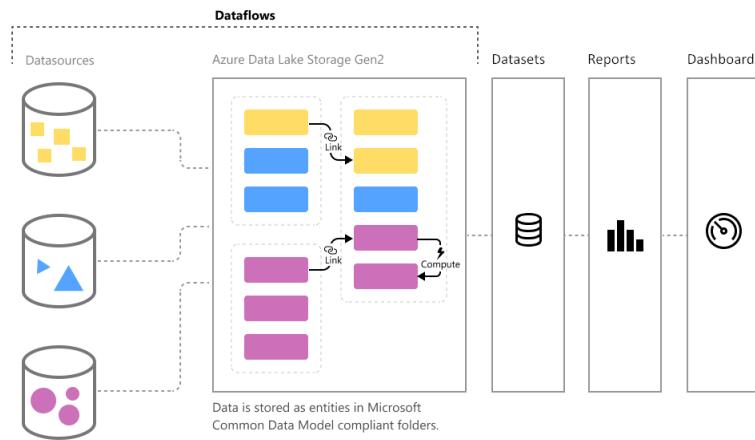


Abbildung 95: Quelle: <https://docs.microsoft.com/en-us/power-bi/transform-model/dataflows/introduction-self-service>

Das Konzept von Dataflow verlagert den ETL Prozess in die Cloud, und erlaubt eine Skalierung des selbigen. Mehrere Nutzer können damit die gleiche Datenvorbereitung zurückgreifen.

**Integration** Die Daten liegen im Azure Data Lake Storage Gen2 (ADLGen2), somit können andere Azure Services direkt darauf zugreifen. Ohne einen entsprechenden Dataflow müsste jeder Service direkt an die Quelle angebunden werden.

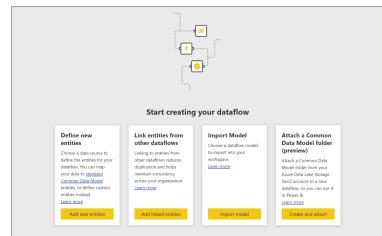
**Klarheit** Die Datenvorbereitung erlaubt einen einheitlichen Standard zu entwickeln, mit welchen die Daten präsentiert werden. Analysten können somit anhand der Dataflows ihre Reports bauen.

**Effizient** Die Daten werden im Azure Data Lake Storage Gen2 gespeichert, ein Laden erfolgt von den Quellen zu diesem. Diese reduziert die Ladungsmenge der darunterliegenden Datenquellen. Eine Skalierung erfolgt über ADLGen2 anstatt der vorgelagerten Systeme.

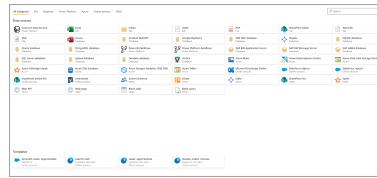
**Sicherheit** Direkter Zugang zu den Datenquellen kann verwehrt werden, und in gewünschter Menge/ Form über den Dataflow präsentiert werden.

**Create a Dataflow** Ein Dataflow besteht aus einem Bündel aus **Entities**. Ein Entity besteht aus einem Bündel von Feldern, und hat die gleiche Beziehung zum Dataflow wie eine Tabelle zur Datenbank.

Es gibt mehrere Möglichkeiten Dataflows zu erstellen.



Die erste Option umfasst, dass neue Datenquellen angebunden werden. Hinweis: Es besteht die Möglichkeit diese Felder standardisierten Entity zuzuordnen.



The entity is shown under the new create dataflow. You can add new entities to the dataflow by the *Add Entity*.

The add calculation or modify the entities the *enable Load*. Berechnungen werden an den geladenen Daten durchgeführt, nicht an der Datenquelle selbst. **Dass spart Zeit, weil Berechnungen den Upload Process verlangsamen.**

Column1	Column2
accountid	callDate
0011r00001mv6joAAA	5/29/2030
0011r00001mv6jyAAA	6/14/2030
0011r00001mv6jgAAA	1/23/2030
0011r00001mv6jvAAA	4/28/2030
0011r00001mv6jsAAA	1/2/2021
0011r00001mv6jyAAA	1/29/2030
0011r00001mv6juAAA	3/28/2030
0011r00001mv6jvAAA	2/6/2020
0011r00001mv6jwAAA	1/20/2030
11	0011r00001mv6jyAAA
12	0011r00001mv6jyAAA

## Use of a dataflow

- Linked
- Report
- Connection to other services

The screenshot shows the Power BI Dataflows interface. On the left, there's a navigation bar with Home, Favorites, Recent, Apps, Shared with me, Workspaces, and PBI Dataflows. The main area has a search bar and tabs for Dashboards, Reports, Workbooks, Datasets, and Dataflows. Below is a table with two rows:

NAME	ACTIONS	LAST REFRESH	NEXT REFRESH
Microsoft Stock Prices	...	1/31/2020, 11:42:27 AM	N/A
Testing Tables	...	N/A	N/A

At the bottom right, there are LinkedIn and Learning links.

This screenshot is similar to the first one but shows the 'Access' panel open on the right side. The panel title is 'PBI Dataflows' and it says 'Add admins, members, or contributors. Learn more'. It has a search bar for email addresses and a dropdown menu for selecting roles: Member, Admin, Contributor, and Viewer. Below is a table of users:

NAME	PERMISSION
Lea Sheryl Rodriguez	Admin

At the bottom right, there are LinkedIn and Learning links.

## Access

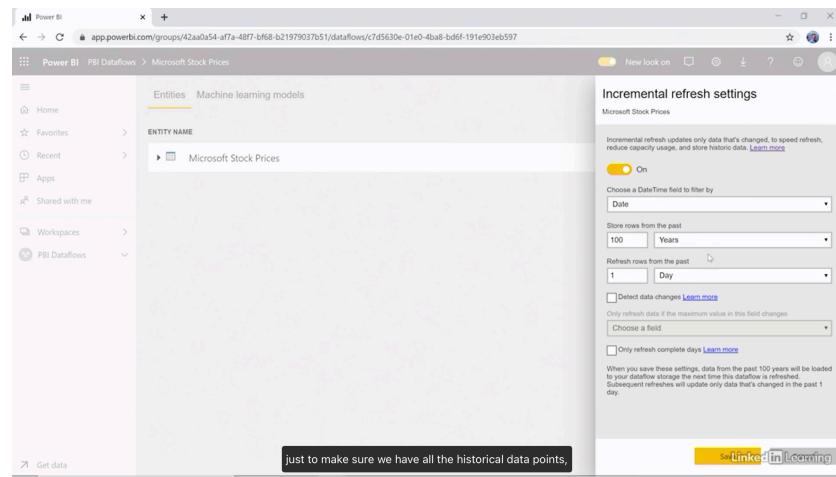
**Refresh Dataflow, and Incremental Load for entities** Dataflows können nur gesamthaft aktualisiert werden. Entities in einem Dataflow können inkremental geladen werden.  
Wählt man die Option Refresh aus, kommt man auf die folgende Übersicht:

This screenshot shows the 'Settings' page for a Dataflow named 'Microsoft Stock Prices'. The URL is <https://app.powerbi.com/groups/42aa054-a7a-48f7-bf68-b21979037b51/settings/dataflows/c7d5630e-01e0-4ba8-bd6f-191e903eb597>. The page header includes 'New look on' and a user icon.

The main content area shows 'Settings for Microsoft Stock Prices'. It says 'This dataflow has been configured by [lrodriog@globalbankinc.com](#)'. Below is a note: 'Last refresh succeeded: Fri Jan 31 2020 11:42:27 GMT-0800 (Pacific Standard Time) Refresh history'. There are sections for 'Gateway connection', 'Data source credentials', and 'Scheduled refresh'. Under 'Scheduled refresh', there are settings for 'Keep your data up to date' (On), 'Refresh frequency' (Daily), 'Time zone' (UTC-05:00 Eastern Time (US and Canada)), 'Time' (9:00 PM), and a checkbox for 'Send refresh failure notifications to the dataflow owner'. At the bottom, it says 'you can update them far more often than the Pro accounts.' and there are LinkedIn and Learning links.

Im Premium Konto kann man mehrere Ladungen erfolgen. Dies bedeutet, dass mehrere Zeiten hinterlegt werden können.

**Incremental Load** wirkt auf eine spezielle Spalte einer Entity angewandt. Die muss in einer Datums- oder Datumszeit Formatierung vorliegen. Es kann dabei in den zurückliegenden. Hinweis: Incremental Loading kann nur an Premium Konten geknüpft sein.



## 6 Advanced Microsoft Power BI

### 6.1 Filter

#### 6.1.1 Filter für Measure und DAX Funktionen

Es können zwei große Kategorien von Filter unterschieden werden:

- Pivot-Koordinaten-Filter (PKF)
  - Bei gleichen Spalten in der gleichen Tabelle, wird der Filter geblockt und so betrachtet, als wurde er nicht implementiert.
- DAX-Filter-Option (DFO)
  - Filter werden von außen nach innen weitergegeben.
  - Bei Spalten die in Beziehung stehen, muss aufgepasst werden, ob PKF von der Up-Stream oder Down-Stream Seite kommt. Entweder wird die der PKF geblockt oder führt dazu, dass die restliche Datenbasis leer wird.

#### 6.1.2 All-Funktion

- Die `All()` blockt alle Filter, die von anderen DAX Funktionen übergeben wurden und den angegebenen Bereich (spezifische Tabelle oder Spalte) betreffen.
- Die `All()` blockt alle Filter, die von anderen PKF übergeben wurden und den angegebenen Bereich (spezifische Tabelle oder Spalte) betreffen. **Achtung:** Der Fokus liegt auf die Filterung, nicht auf den Bereich der Berechnet wird. Wo

---

```

1  =Calculate(Sum([Anzahl_Fahrgäste]))
2  /Calculate(Sum([Anzahl_Fahrgäste]),All(ETL_FGZ_Prufer[Anzahl_Fahrgäste]))
```

---

hilft nicht, wenn die PKF aus der Datumstabelle kommen. Sollen die PKF für die Daten geblockt werden, so muss Folgendes angewandt werden.

---

```

1  =Calculate(Sum([Anzahl_Fahrgäste]))
2  /Calculate(Sum([Anzahl_Fahrgäste]),All('Calendar'[Date]))
```

---

- Der Rückgabewert ist eine Tabellen, aber `Calculate()` erkennt die Funktion an, und beitet nicht nur Boolean sondern auch Tabellen Input für DFO.

### 6.1.3 Filter-Funktion

Die Filterfunktion bietet den die Möglichkeit Filter in der DAX Funktion und über die PKF anzuwenden. Würde man die Filterfunktion nicht verwenden, wenden die PKF nicht berücksichtigt und ausgeblendet.

Date	Average Rate	AvgAg_Ext_Filter	AvgAg_Ext_Filter
Montag, 1. Januar 2018	3,20 %	3,30 %	3,30 %
Dienstag, 2. Januar 2018	3,30 %	3,30 %	3,30 %
Mittwoch, 3. Januar 2018	3,40 %	3,30 %	3,30 %
Donnerstag, 4. Januar 2018	3,00 %	3,30 %	3,30 %
Freitag, 5. Januar 2018	3,20 %	3,30 %	3,30 %
Samstag, 6. Januar 2018	3,40 %	3,30 %	3,30 %
Sonntag, 7. Januar 2018	3,10 %	3,30 %	3,30 %
Montag, 8. Januar 2018	3,30 %	3,30 %	3,30 %
Dienstag, 9. Januar 2018	3,10 %	3,30 %	3,30 %
Mittwoch, 10. Januar 2018	3,30 %	3,30 %	3,30 %
Donnerstag, 11. Januar 2018	3,10 %	3,30 %	3,30 %
Freitag, 12. Januar 2018	3,30 %	3,30 %	3,30 %
Samstag, 13. Januar 2018	3,30 %	3,30 %	3,30 %
Sonntag, 14. Januar 2018	3,40 %	3,30 %	3,30 %
Montag, 15. Januar 2018	3,20 %	3,30 %	3,30 %
Dienstag, 16. Januar 2018	3,30 %	3,30 %	3,30 %
Mittwoch, 17. Januar 2018	3,30 %	3,30 %	3,30 %
Donnerstag, 18. Januar 2018	3,30 %	3,30 %	3,30 %
Freitag, 19. Januar 2018	3,00 %	3,30 %	3,30 %
Samstag, 20. Januar 2018	3,20 %	3,30 %	3,30 %
<b>Gesamt</b>	<b>3,25 %</b>	<b>3,30 %</b>	<b>3,30 %</b>

**3,25%**  
Average Rate

- All-Funktion block alle spezifische Tabellen oder Spalten
- DAX-Filter blocken die PKF.
- Filter Funktion erlaubt eine Vor-Filterung und eine Einbeziehung der PKF. Ebenso ist eine dynamische Übergabe von Werten möglich. Als Beispiel kann ein Measure übergeben werden als zu filternder Wert.
- Die Funktion *Filter()* erlaubt kompliziertere Bedingungen zu formulieren. Im Vergleich zu DFO können folgenden Funktionen mit aufgegriffen werden:

#### – DAX

- \* Die Filterbedingung erlaubt nicht für

---

```

1   =Calculate(Sum([Anzahl_Fahrgäste]),
2   'Calendar'[Date]=Min(ETL_FGZ_Ptruefer[Datum]))

```

---

- \* Die *Filter()* erlaubt für

---

```

1   =Calculate(Sum([Anzahl_Fahrgäste]),
2   Filter('Calendar','Calendar'[Date]=Min(ETL_FGZ_Ptruefer[Datum])))

```

---

Dabei ist wichtig zu berücksichtigen, dass die PKF für den gleichen Bereich den Filter überschreiben.

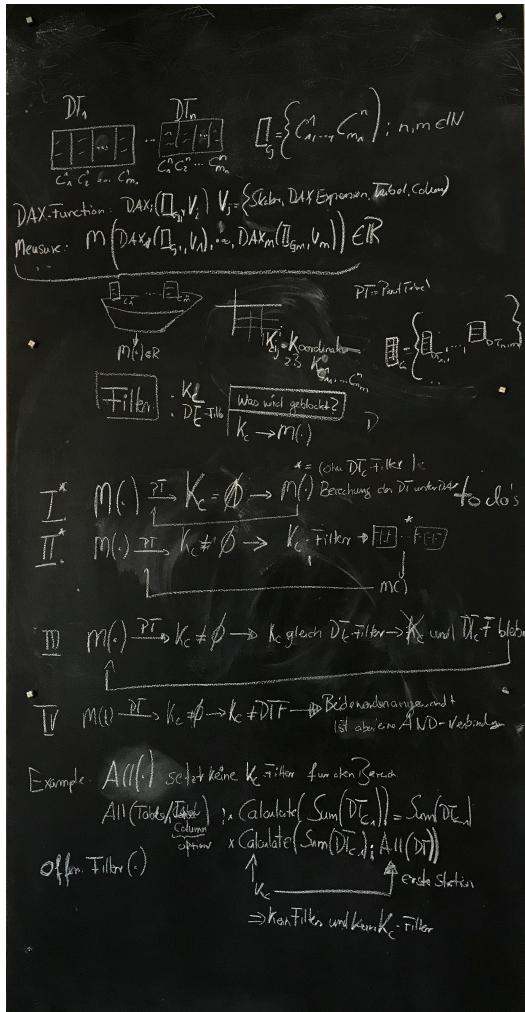


Abbildung 96:

Mit Hilfe einer Umschließung der *Calculate()* Funktion kann das PKF geblockt werden.

---

```

1   =Calculate(
2     Calculate(
3       Sum([Anzahl Fahrgäste]),
4       Filter('Calendar',
5         'Calendar'[Date]=Min(ETL_FGZ_Pruefer[Datum])
6       )
7     )
8     ,All('Calendar'[Date])
9   )

```

---

- Measure

- Vergleiche von Tabellenspalten

---

```

1   CALCULATE(Sum([Anzahl Fahrgäste]),Filter('Calendar','Calendar'[Date]='
2     Calendar'[Date]))

```

---

- Wie schon oben erwähnt, werden PKF nicht geblockt und können somit direkt angewandt werden.

## 6.2 DAX Measures

### 6.2.1 Quick Measure

Dies vorkonfigurierten Funktionen erlauben etablierte und gängige Measures zu verwenden.

### 6.2.2 Divide and Exponent Function

Die beiden Funktionen, Divide und Exponent erlauben eine saubere und schlanker Berechnung. Sie helfen Fehlermeldungen direkt in der Funktion mit einzubetten ohne ein Wenn-Dann Ausdruck herumzubauen.

### 6.2.3 VAR and Return

Ein Measure kann ebenso aus mehreren Measure bestehen.

Der Ausdruck `Var` erlaubt, dass Ausdrücke abgespeichert werden und intern weiter gegeben werden können. Mit dem Ausdruck `return` wird festgelegt, was das definiert Measure wieder geben soll.

Beide Ausdrücke werden am folgenden Beispiel verwendet, um eine Division für eine Prozentrechnung zu demonstrieren.

---

```
1 Prozент =
2     Var SingleMax = CALCULATE(Max('Inflation rates (all countries)'[Inflation]))
3
4     Var OverallMax = CALCULATE(Max('Inflation rates (all countries)'[Inflation]),All(
5         'Inflation rates (all countries)'))
6
6     return DIVIDE(SingleMax,OverallMax)
```

---

### 6.2.4 If-Statement

Mit dem `If` Statement können auch unterschiedliche Measure ausgewählt werden.

### 6.2.5 SUMX()

Die `SUMX()` addiert von Zeile zu Zeile den gesetzten Ausdruck. Wird als Ausdruck nur der Spaltenverweis verwendet, so ist `SUMX()` und `SUM()` gleich. Erst, wenn zeilenweise Multiplikation oder komplexere Berechnungen benötigen werden, die nicht linear, unterscheiden sich die Funktionen. Betrachten wir also nicht lineare Ausdrücke in der der `SUMX()`, so kommt es zu Unterschieden.

### 6.2.6 COUNTX()

Wie gerade beschrieben, werden Ausdrücke auf Zeilenebene bewertet.

### 6.2.7 DateDiff

Bestimmt das Zeitintervall zwischen zwei Datumsangaben. Die Intervalle in welche es zurückgegeben wird, können frei gewählt werden.

### 6.2.8 DATESBETWEEN

Diese Funktion wird verwendet, um einen Filter von Datumswerten genauer und tabellenscharf festzulegen.

## 6.3 Rank

### 6.3.1 Funktionsweise

Der folgende Abschnitt beginnt mit der `RANKX` Funktion und baut sich weiter auf. Dabei wird auf Gruppierung, Slicer und Mehrere Slicer eingegangen.

Die `RANKX` Funktion ist eine **Iterator** Funktion<sup>1</sup>

Der Aufbau gestaltet sich wie folgt:

`RANKX {< table >, < expression >, < value >, < ord >, < ties >}`

---

<sup>1</sup>Die `Filter` Funktion gehört auch zu dieser Kategorie.

Die Funktionsweise dieses Measure wird in der folgenden Übersicht dargestellt.

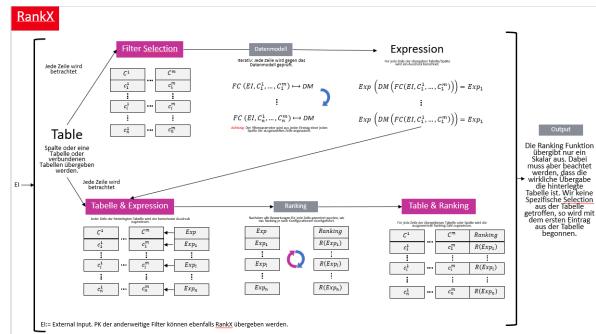


Abbildung 97: File: DAX RankX in folder chapter 1

### 6.3.2 Beispiel

Im mitgelieferten Beispiel wird das Measure *Simple MAX* erstellt, um das Maximum einer spezifischen Spalte zu bestimmen.

```
1 Simple Max =
2   CALCULATE(
3     MAX(
4       'Inflation rates (all countries)'[Inflation]
5     )
6   )
```

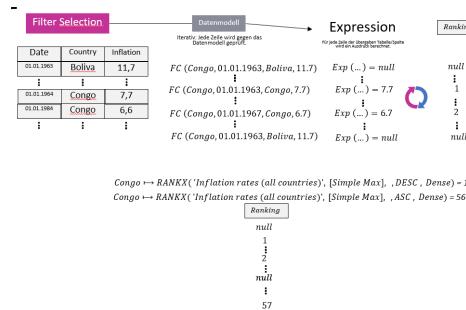
Wird dies in *RankX* eingebunden:

```
1 RankX Input: Country Table =
2   RANKX(
3     'Inflation rates (all countries)', // Input Table
4     [Simple Max], // Expression to evaluate against the Data Model
5     , // Parameter of RankX
6     ASC, // Parameter of RankX
7     , Dense // Parameter of RankX
8   )
```

Der Output:

Country	Simple Max	RankX Input: Country Table
Congo, Dem. Rep.	26.000,00	1
Georgia	15.000,00	1
Nicaragua	13.000,00	1
Bolivia	12.300,00	1
Peru	6.260,00	1
Angola	4.000,00	1
Armenia	4.110,00	1
Ukraine	3.330,00	1
Turkmenistan	3.090,00	1
Afghanistan	2.800,00	1
Brazil	2.700,00	1
Belarus	1.950,00	1
Kazakhstan	1.550,00	1
Russia	1.490,00	1
Azerbaijan	1.390,00	1
Macedonia, FYR	1.270,00	1
Gesamt	26.800,00	1

Zu jedem Land wird der Rang 1 angezeigt. Dies röhrt daher, dass die übergebene *Filtertabelle* von dem übergebenen PKF vorgefiltert wird.



Wird der Parameter der Funktion von *ASC* zu *DESC* umgestellt:

---

```

1 RankX Input: Country Table =
2   RANKX(
3     'Inflation rates (all countries)', // Input Table
4     [Simple Max], // Expression to evaluate against the Data Model
5     , // Parameter of RankX
6     , DESC // Parameter of RankX
7     ,Dense // Parameter of RankX
8   )

```

---

wird für jedes Land der letzte Rank angegeben, den gebildet wird aus allen Inflationsraten des jeweiligen Landes.

Country	Simple Max	RankX Input: Country Table
Congo, Dem. Rep.	26,800,00	57
Georgia	15,400,00	53
Nicaragua	13,600,00	56
Bolivia	12,300,00	54
Peru	6,260,00	55
Angola	4,800,00	58
Armenia	4,110,00	27
Ukraine	3,330,00	30
Turkmenistan	3,000,00	30
Argentina	3,060,00	54
Brazil	2,700,00	57
Belarus	1,950,00	26
Kazakhstan	1,930,00	26
Russia	1,490,00	27
Azerbaijan	1,390,00	27
Macedonia, FYR	1,270,00	27
Gesamt		2537
	26,800,00	

Für den Congo bedeutet dies, dass 57 Inflationsraten in der ausgewerteten Tabelle vorliegen.

### 6.3.3 Gruppierungen

Um ein Ranking über alle Inflationsraten zu erhalten, muss der jeweilige PKF geblockt werden. Wird diese erfüllt, so bewertet die RankX Funktion nur die Reihenfolge der Inflationsraten zum jeweiligen Krieg.

---

```

1 RankX Input: Country Table =
2   RANKX(
3     All('Inflation rates (all countries)'), // Input Table
4     [Simple Max], // Expression to evaluate against the Data Model
5     , // Parameter of RankX
6     , ASC // Parameter of RankX
7     ,Dense // Parameter of RankX
8   )

```

---

Das Resultat sieht wie folgt aus:

Country	Simple Max	RankX Input: Country Table
Congo, Dem. Rep.	26,800,00	1
Georgia	15,400,00	2
Nicaragua	13,600,00	3
Bolivia	12,300,00	4
Peru	6,260,00	5
Angola	4,800,00	6
Armenia	4,110,00	11
Ukraine	3,330,00	13
Turkmenistan	3,000,00	14
Argentina	3,060,00	15
Brazil	2,700,00	16
Belarus	1,950,00	24
Kazakhstan	1,930,00	28
Russia	1,490,00	29
Azerbaijan	1,390,00	30
Macedonia, FYR	1,270,00	34
Gesamt		1
	26,800,00	

Es wird für jeden Eintrag in der *Filtertabelle* ein Ranking erstellt unabhängig des übergebenen Country PKF. In der Darstellung sieht es so aus, als ob Ranking-Plätze nicht berücksichtigt werden. Diese liegt daran, dass für das jeweilige Land mehrere Ranking-Plätze vorliegen können. Weil die Auswahl *DESC* getroffen wurde, wird das höchste Ranking ausgewählt und dem jeweiligen Land zugeordnet.

Country	Simple Max	RankX Input: Country Table
Congo, Dem. Rep.	26,800,00	2537
Georgia	15,400,00	2536
Nicaragua	13,600,00	2535
Bolivia	12,300,00	2534
Peru	6,260,00	2533
Angola	4,800,00	2530
Armenia	4,110,00	2529
Ukraine	3,330,00	2528
Turkmenistan	3,000,00	2524
Argentina	3,060,00	2523
Brazil	2,700,00	2522
Belarus	1,950,00	2514
Kazakhstan	1,930,00	2510
Russia	1,490,00	2509
Azerbaijan	1,390,00	2508
Macedonia, FYR	1,270,00	2504
Gesamt		2537
	26,800,00	

Um ein internes Ranking der Länder zu schaffen, erlaubt die *All()* Funktion eine Spalte anzugeben:

---

```

1 RankX Input: Country Table =
2     RANKX(
3         All('Inflation rates (all countries)'[Country]), // Input Table
4         [Simple Max], // Expression to evaluate against the Data Model
5         , // Parameter of RankX
6         ,ASC // Parameter of RankX
7         ,Dense // Parameter of RankX
8     )

```

---

Dabei wird diese Spalte ausgelesen und nur die eindeutigen (*unique*) Werte zurückzugeben. Für jedes dieser Länder wird daraufhin ein Ranking der jeweils höchsten Inflationen aufgestellt.

Country	Simple Max	RankX Input: Country Table
Congo, Dem. Rep.	26,800.00	1
Georgia	24,600.00	2
Nicaragua	19,600.00	3
Bolivia	12,300.00	4
Peru	10,100.00	5
Angola	8,800.00	6
Armenia	4,110.00	7
Ukraine	3,830.00	8
Turkmenistan	3,600.00	9
Argentina	3,090.00	10
Brazil	2,700.00	11
Belarus	2,950.00	12
Azerbaijan	2,550.00	13
Russia	2,490.00	14
Azerbaijan	1,390.00	15
Macedonia, FYR	1,270.00	16
Gesamt		26,800.00
		1

### 6.3.4 RANK.EQ

Die Funktion *Rank.EQ* bietet sich für Berechnung von Rankings als Berechnende Spalte an.

RANK.EQ {< value >, < columnName >, < Order >}

Im Gegensatz zu *RankX* wertet diese Funktion keine Measure aus, sondert bewertet Integer einer verwiesenen Spalte.

## 6.4 ALL Functions

### 6.4.1 ALL

In dem Abschnitt **Rank** wurde die **All** Funktion verwendet, um die PKF aus dem Report zu blocken. Die Feinheit für diese Funktion besteht darin, dass eine spezifische Tabelle oder Spalte beblökt werden kann.

- Wird auf eine Spalte verwiesen, werden die eindeutigen Werte der Spalte als Spalte ausgegeben.
- Wird auf mehrere Spalten verwiesen, so werden diese in ihren möglichen Kombinationen zurückgegeben.
- Wird auf eine Tabelle verweisen, so wird diese Tabelle ungefiltert zurückgegeben - Alternativ ausgedrückt: Im Datenmodell wird diese nicht gefiltert und steht zur Berücksichtigung für das umschlossenen Measure zur Verfügung.

ALL {< TableOrColumnName >, < ColumnName >, < ColumnName >, >}

### 6.4.2 ALLSELECTED

Die **ALLSELECTED** Funktion verhält sich von den Inputvarialben und der Möglichkeit der Ausgabe gleich der **All** Funktion. Was hingegen möglich ist, dass externe Filter oder auch DFO übergeben werden, während die PKF geblockt werden.

Im Folgenden wir anhand des mitgelieferten Beispiel beschrieben, wie die **ALLSELECTED** von der **All** Funktion unterscheidet.

---

```

1 Simple Max =
2     CALCULATE(
3         MAX(
4             'Inflation rates (all countries)'[Inflation]
5         )
6     )

```

---

```

7 Max All =
8   CALCULATE(
9     MAX(
10      'Inflation rates (all countries)'[Inflation]
11    ),
12      ALL('Inflation rates (all countries)')
13  )
14
15 Max ALLSelected =
16   CALCULATE(
17     MAX(
18      'Inflation rates (all countries)'[Inflation]
19    ),
20      ALLSELECTED('Inflation rates (all countries)')
21  )

```

---

Date			
26.02.1991 25.11.1991			
O-O			
Country	Simple Max	Max All	Max ALLSelected
Afghanistan	26.800,00	13.600,00	
Albania	38,60	26.800,00	13.600,00
Algeria	38,60	26.800,00	13.600,00
Angola	12,90	26.800,00	13.600,00
Angola	108,00	26.800,00	13.600,00
Anguilla and Barbuda	12,40	26.800,00	13.600,00
Argentina	3.090,00	26.800,00	13.600,00
Argentina	9,940	26.800,00	13.600,00
Armenia	1,75	26.800,00	13.600,00
Australia	3,54	26.800,00	13.600,00
Austria	83,50	26.800,00	13.600,00
Azerbaijan	1,43	26.800,00	13.600,00
Bahrain	1,10	26.800,00	13.600,00
Bangladesh	11,73	26.800,00	13.600,00
Banladesh	13.600,00	26.800,00	13.600,00
Gesamt	13.600,00	26.800,00	13.600,00

#### 6.4.3 ALLEXCEPT

Die Funktion wird benötigt, wenn eine *Parent-Child* Hierarchy vorliegt. Die Funktionweise ist dabei umgedreht wie bei [ALL](#). Alle Spalten die extra angegeben werden, werden nicht berücksichtigt, während alle anderen geblockt werden.

Das folgende Beispiel hat eine *Parent-Child* Hierarchy hinterlegt.

Product Category	Total Orders	All Orders	Order Pct (ALL)	AllExcept Orders	Order Pct (ALLEXCEPT)
Furniture	44.345	214.798	20,64%	44.345	11,03%
Bookcases	4.891	214.798	2,28%	44.345	21,63%
Chairs & Chairmats	9.590	214.798	4,46%	44.345	46,61%
Office Furnishings	20.669	214.798	9,62%	44.345	20,74%
Tables	9.195	214.798	4,28%	44.345	
Office Supplies	118.278	214.798	55,06%	118.278	
Appliances	10.901	214.798	5,08%	118.278	9,22%
Binders and Binder Accessories	22.992	214.798	10,70%	118.278	19,44%
Envelopes	6.578	214.798	3,06%	118.278	5,56%
Labels	7.301	214.798	3,40%	118.278	6,17%
Paper	30.871	214.798	14,37%	118.278	26,10%
Pens & Art Supplies	16.980	214.798	7,91%	118.278	14,36%
Rubber Bands	4.920	214.798	2,29%	118.278	4,16%
Scissors, Rulers and Trimmers	3.931	214.798	1,83%	118.278	3,32%
Storage & Organization	13.804	214.798	6,43%	118.278	11,67%
Technology	52.175	214.798	24,29%	52.175	
Computer Peripherals	19.222	214.798	8,95%	52.175	36,84%
Copiers and Fax	2.128	214.798	0,99%	52.175	4,08%
Office Machines	7.856	214.798	3,66%	52.175	15,06%
Telephones and Communication	22.969	214.798	10,69%	52.175	44,02%
Total	214.798	214.798	100,00%	214.798	

#### 6.4.4 SELECTEDVALUES

Diese Funktion gibt immer Skalar aus einem Spaltenverweis zurückzugeben.

Die Grundlogik liegt nahe, dass ein User einen spezifischen Wert auswählen kann, dieser in einem Measure verwendet wird und dynamisch mit den Werten interagieren kann.

Was [SELECTEDVALUE](#) jedoch sicher stellt, es wird immer nur ein Skalar zurückgegeben. Wählt er User mehrere Werte<sup>2</sup> aus oder wird kein Wert ausgewählt, dann muss festgelegt werden, welcher Wert zurückgegeben werden soll. Die Rückgabe kann jedoch auch ein Text sein.

Das folgende Beispiel greift auf, wie die höchste Inflationsrate in anderen Größe dargestellt werden kann.

---

```
1 Max-Teiler =
```

<sup>2</sup>Außer es handelt sich um identische

```

2 // Teiler gibt den ausgewählten Teiler zurück. Wird keiner ausgewählt, so wird 1
  // zurückgegeben.
3 Var Nenner = SELECTEDVALUE(Teiler[Spalte "1"],1)
4
5 Var Zaehler = CALCULATE(
6   Max('Inflation rates (all countries)'[Inflation])
7   //All('Inflation rates (all countries)')
8 )
9
10 return DIVIDE(Zaehler,Nenner)

```

---

Country	MaxTeller
Afghanistan	22,40
Albania	2500,00
Algeria	51,00
Andorra	29,20
Angola	1000,00
Anguilla	14,80
Argentina	3160,00
Armenia	1,00
Australia	16,50
Gesamt	26.800,00

Abbildung 98: Keine Auswahl wurde getroffen

Country	MaxTeller
Afghanistan	0,02
Albania	0,25
Algeria	0,03
Andorra	0,03
Angola	4,10
Anguilla	0,01
Argentina	1,06
Armenia	0,11
Australia	0,02
Gesamt	26,80

Abbildung 99: Die Auswahl 1000 wurde getroffen.

#### 6.4.5 Value

Die Funktion `Value` verhält sich wie die `ALL` Funktion in dem speziellen Fall, dass der Verweis auf eine einzige Spalte zeigt. Der Rückgabe Wert ist eine Spalte mit den eindeutigen Werten der übergebenen Spalte. *Achtung: PKF können durchgereicht werden.*

`Value {< TableOrColumnName >}`

Am Beispiel der oben, angeführten Ranking Funktion wird als Filter-Tabelle zwar die Spalte mit den eindeutigen Werten übergeben. Der PKF der jeweiligen Zeile wird jedoch nicht geblockt, weshalb das Resultat **1** für jede Zeile ist.

```

1 RankX Input: Country Table =
2 RANKX(
3   Value('Inflation rates (all countries)'[Country]), // Input Table
4   [Simple Max], // Expression to evaluate Data Model
5   , // Parameter of RankX
6   DESC, // Parameter of RankX
7   Dense // Parameter of RankX
8 )

```

---

## 7 Power BI Mistakes to be avoided

### 7.1 Reduce Data

- Reduce the amount of data you are loading into Power Query
- Reduce the amount of data loaded into the data model

## 7.2 Display additional information

### Tooltip

- Format Tool Tip Page to tooltip page
- Select right visualisaiton in tool tip settings
- Aim: It used additional information (Measures) to show details

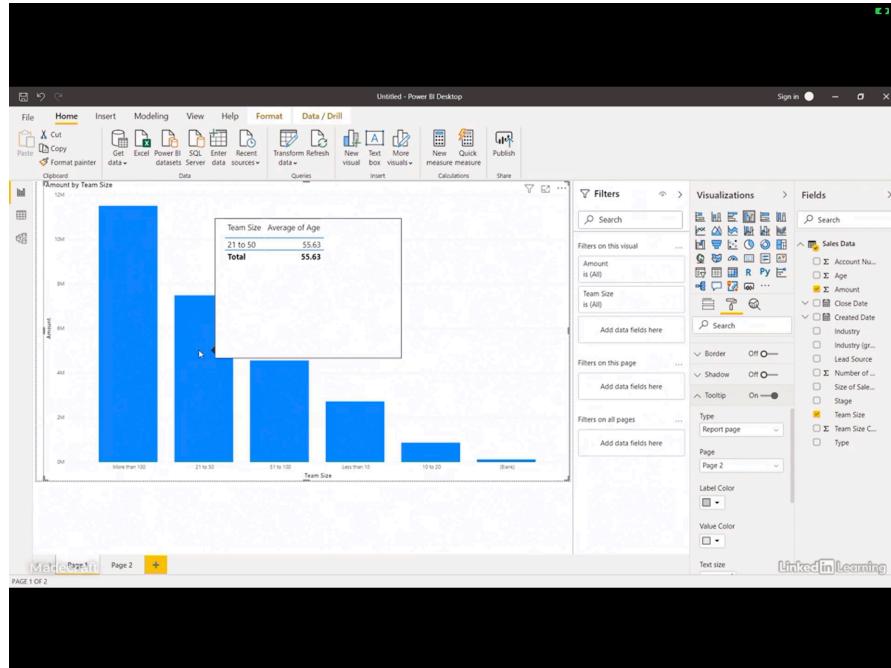


Abbildung 100: Toolkit

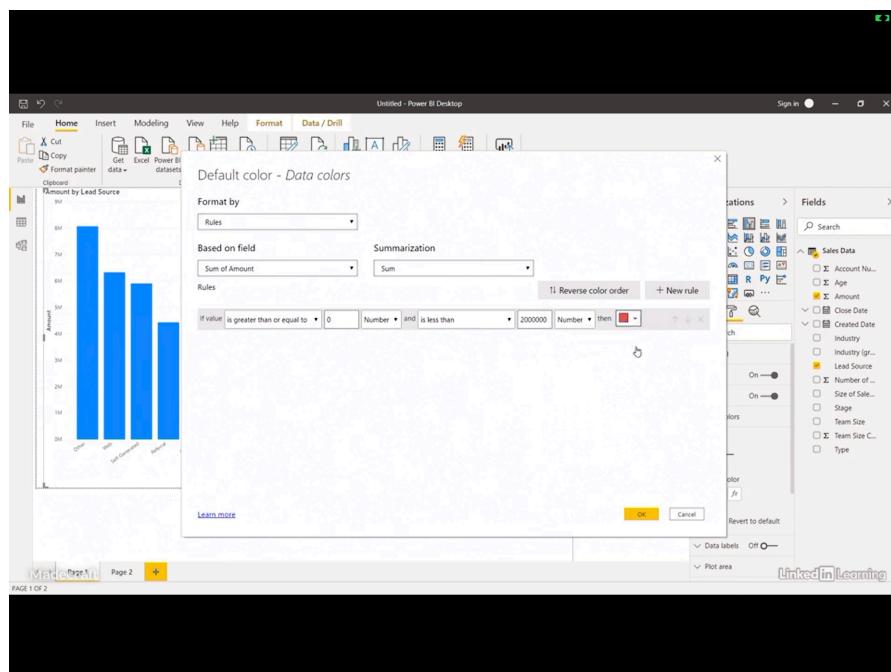


Abbildung 101: Added color rule

## Rule to color chart

### 7.3 Query Folding

**Full Folding** Transformation, die in der Query Sprache M Formula Language geschrieben werden, können direkt bei der Abfrage an die Datenquelle gestellt werden. Dabei werden Transformationen direkt auf Seiten der Datenquelle durchgeführt. Hierfür werden die Transformationsschritte oder Teile davon in

#### Native Query Language

überführt. Diese ist einsehbar, wenn auf den gewünschten, letzten Transformationsschritt mit der rechten Maus geklickt wird, und auf Native Query geklickt wird.

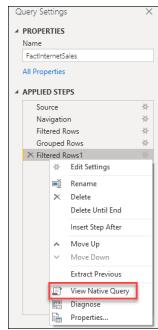


Abbildung 102: Option: Native Query Folding

```

Native Query

select [...] from
(
    select [row].[OrderDate], [...] from [dbo].[FactInternetSales]
    group by [...] having sum([row].[SalesAmount]) >= 15000
)
group by [...]
)

```

Abbildung 103: Native Query Language

Diese Anweisungen sind an die Datenquelle gesendet. **Achtung: Wenn diese oder die Transformationen dies zulassen.** An die Power Query Engine (PQE) werden die vorselektierten Daten übermittelt.

Der Vorteil ist, dass weniger Rechenkapazität lokal oder in de cloud PQE benötigt wird.

**Partial Folding** Wenn nicht alle Transformationsschritte übermittelt werden, kommt es zu

*partial folding.*

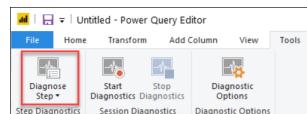
Der Transformationsschritt *Filter 2* zeigt die Auswahl *Native Query* nicht mehr an. Hingegen der Schritt *Filter 1* zeigt die *Native Query*

```

select [...] [OrderDate], ... [Day Name]
from ...
(
    select [rows] [OrderDate] as [OrderDate],
           sum([rows].[SalesAmount]) as [Total Sales Amount]
    from ...
    (
        select [...] [ProductKey],
               [...] [OrderDateKey],
               [...] [DueDateKey],
               [...] [ShipDateKey],
               [...] [CustomerKey],
               [...] [PromotionKey],
               [...] [CurrencyKey],
               [...] [SalesTerritoryKey],
               [...] [SalesOrderNumber],
               [...] [SalesOrderLineNumber],
               [...] [RevisionNumber],
               [...] [OrderQuantity],
               [...] [UnitPrice],
               [...] [ExtendedAmount],
               [...] [UnitPriceDiscountPct],
               [...] [DiscountAmount],
               [...] [ProductStandardCost],
               [...] [TotalProdCost],
               [...] [SalesAmount],
               [...] [TaxAmt],
               [...] [Freight],
               [...] [CarrierTrackingNumber],
               [...] [CustomerPONumber],
               [...] [OrderDate],
               [...] [dueDate],
               [...] [ShipDate]
        from [dbo].[FactInternetSales] as [...]
        where [...] [OrderDate] >= convert(datetime2, '2012-09-01 00:00:00') and [...] [OrderDate] <= convert(datetime2, '2012-09-30 00:00:00')
    ) as [rows]
    group by [OrderDate]
) as [...]
where [...] [Total Sales Amount] >= 15000

```

an. Alle Schritt welche nach dem letzten Möglichen Transformationsschritt erfolgen, werden in der PQE durchgeführt. Um herauszufinden, welche Transformationen zur Datenquelle übermittelt werden, kann dies über das Diagnose Tool eingesehen werden.



**Query Folding with Dataflow Connector** Es gibt einen neuen Connector **Dataflow** mit der M Formula Language Funktion

*PowerPlatform.Dataflow()*.

Dieser besitzt gegenüber **Power BI Dataflow** mit der Funktion

*PowerBI.Dataflow()*

den Vorteil, dass *Query Folding* möglich ist. Als Zusatz muss

*Enhanced Compute Engine*

aktiv sein.

▲ Enhanced compute engine settings  
Configure enhanced compute engine settings for this dataflow.

Disabled  
Turn off the enhanced compute engine for this dataflow.

Optimized  
We'll turn on the enhanced compute engine only when this dataflow is linked to another one, which will enhance performance.

On  
Turn on the enhanced compute engine for this dataflow.

## 7.4 Table View - Running Query Twice

In Power BI Desktop ist es möglich, dass Query zweimal durchgeführt werden. Dabei wird die doppelte Menge an Daten geladen. Dieses Hindernis ist bei der Cloud PQE nicht anfallend.

Grund für das doppelte Laden: Um Daten zu laden, müssen Informationen über die Konfiguration der Tabelle vorhanden sein. Kann die Datenquelle bei dem Query Aufruf dies nicht liefern. Wird die Datenquelle einmal mehr aufgerufen, um diese Informationen abzuholen.

[Run Twice Run Twice II](#)

## **Teil II**

# **Power Platform**

# 1 Governance and Administration

## 1.1 Architecture

### 1.1.1 Roles

**Default** Environment(Umgebung) haben zwei standardmäßig Verfügbare Rollen.

**Environment(Umgebung) Admin/ Environment(Umgebung) sadmin** Diese Rolle kann alle administrativen Aufgaben übernehmen, wie die folgenden

- Hinzufügen/ Entfernen der Rolle eines Environment(Umgebung) Admin oder Maker einem Nutzern oder einer Gruppen
- Erstellen einer Common Data Service/ Dataverse (CDS)<sup>3</sup>
- Einsicht und managen aller Ressourcen einer Environment(Umgebung)
- Erstellen von Data Loss Prevention Policies (DLP)

**Environment(Umgebung) Maker/ Umgebungshersteller** Mit dieser Rolle können Objekte/ Ressourcen in der Environment(Umgebung) wie Apps, Flows, Konnektoren und Gateways erstellt werden. Ebenso besitzt diese Rolle die Möglichkeit Apps mit individuellen User in einem Unternehmen zu teilen.

Wenn User eine Rolle zugewiesen bekommen hat, die damit nicht der Zugriff auf die eingebettet CDS sichergestellt, falls diese existiert. Dieser Zugriff muss separat erfolgen.

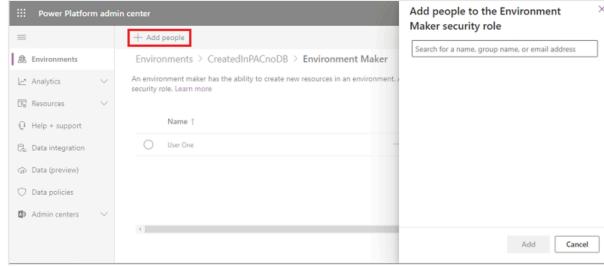
**Environment(Umgebung) ohne CDS** Sicherheitsrollen für Environment(Umgebung) ohne CDS sind nur *Environment(Umgebung) Admin* und *Maker*. Diese werden über die Environment(Umgebung) unter Access gemanagt.

1. Sign in to the [Power Platform admin center](#).
2. Select **Environments** > [select an environment].
3. In the **Access** tile, select **See all** for **Environment admin** or **Environment maker** to add or remove people for either role.

The screenshot shows the 'Power Platform admin center' interface. On the left, there's a sidebar with 'Environments' selected, showing options like Analytics, Resources, Help + support, Data integration, Data gateways (preview), Data policies, and Admin centers. The main area is titled 'Environments > CreatedInPACnoDB'. It has two main sections: 'Details' and 'Access'. The 'Details' section shows Type: Production, Region: United States, Refresh cadence: Moderate, and Purpose: Not specified. The 'Access' section contains two rows: 'Environment admin' with 'See all' and 'Environment maker' with 'See all', both of which are highlighted with a red box. Below these are 'Resources' sections for Power Apps and Flows.

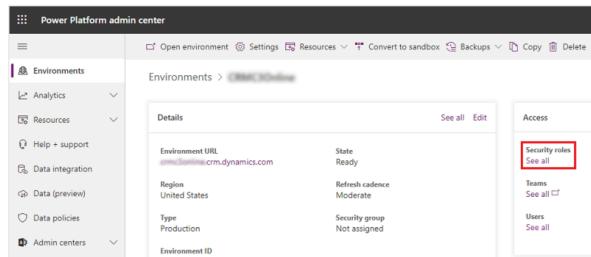
Unter der Auswahl können dann Namen oder Gruppen hinzugefügt werden.

<sup>3</sup>Nachdem eine CDS erstellt wurde, kann der Environment(Umgebung) Admin auf die System Administrator Role wechseln.

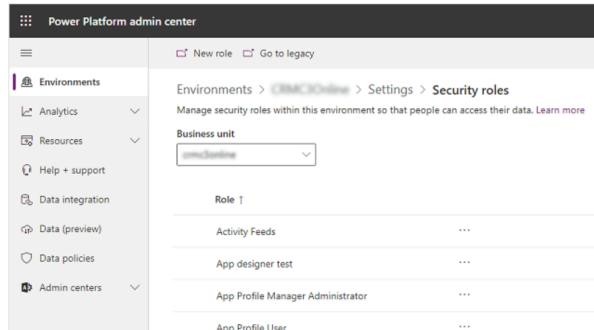


**Environment(Umgebung) mit CDS** Sicherheitsrollen für Environment(Umgebung) mit CDS habe eine Erweiterung. Dies werden über **Access/Security roles** gemanaged.

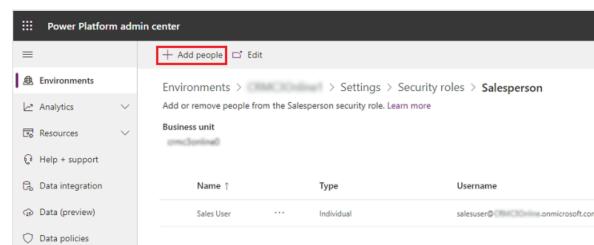
1. Sign in to the [Power Platform admin center](#).
2. Select **Environments** > [select an environment].
3. In the **Access** tile, select **See all** under Security roles.



Unter der Sicherheitsrolle gibt es die Auswahl **Business unit**. Wenn diese ausgewählt ist, steht die List von vor und eigen definierten Sicherheitsrollen zur Verfügung.



Unter Auswahl der Business Unit und der Rolle kann wie gewohnt, eine Person oder Gruppe hinzugefügt werden.



**Custom Security Role** Die Umgebung gibt eine URL an, welche unter Dynamics 365 das Advanced Admin Center erreicht.

The screenshot shows the 'Power Platform admin center' interface. On the left, there's a sidebar with options like 'Analytics', 'Resources', 'Help + support', 'Data integration', 'Data gateways (preview)', 'Data policies (preview)', and 'Admin centers'. The main area is titled 'Environments' and shows a list with one item: 'TestEnvironment'. A detailed view on the right shows the following information:

- Details**
- Environment URL:** <https://testenvironment.crm10.dynamics.com> (highlighted with a red box)
- State:** Ready
- Region:** Test in Production
- Refresh cadence:** Frequent
- Type:** Production
- Security group:** Not assigned

- If you see published apps and tiles, select the gear icon (⚙) in the upper-right corner, and then select **Advanced settings**.
- In the menu bar, select **Settings > Security**.

The screenshot shows the 'Dynamics 365' settings menu. The 'Settings' tab is selected. Under the 'System' section, the 'Security' option is highlighted with a red box. Other options include 'Administration', 'Processes', 'Microsoft Flows', 'Data Management', 'System jobs', 'Auditing', and 'Email Configuration'.

Über diesen Bereich können neue Rollen angelegt werden. Unter **Security Roles** wird genau beschrieben, welche Mindestanforderungen eine neue Sicherheitsrolle haben muss. Die Hauptprivilegen sind *lesen, schreiben und anfügen*.

**Different Types of Admin** Der Globale Admin ist der Admin des Tenants.

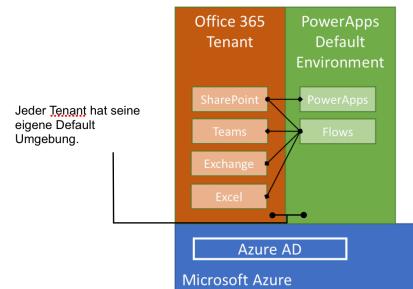
## Define organizational admin roles



Abbildung 104: Different admin roles in a organisation

### 1.1.2 Environment(Umgebung)

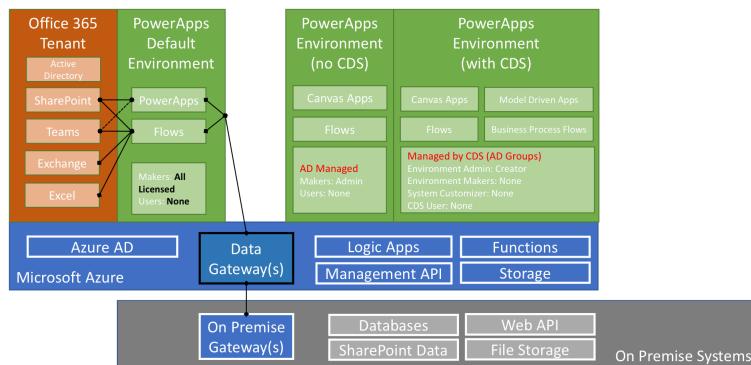
**Default Environment(Umgebung)** Die Power Platform (PP) ist auf Microsoft Azure aufgebaut. Die Verwaltung (Azure AD), der Speicher (Storage) und weitere Komponenten sind Teil des Fundaments.



Jeder Office 365 Tenant, meist für eine Organisation ist dies ein Tenant, erhält eine Default Environment(Umgebung) für alle Nutzer. In diese Umgebung hat die PP Default Umgebung über Power Apps Zugriff auf SharePoint und Flow kann sich zu den gängigen regional nächsten zur Azure AD des Tenants. Die Default kostenlose Kapazität beträgt 3 GB für Dataverse Database sowie Dateispeicher.

Für die Default Umgebung erhält jeder User die **Environment(Umgebung) Maker/ Umgebungs-Hersteller** Berechtigung. User sind, welche eine PP Lizenz erworben haben. Diese werden automatisch der Umgebung zu gewissen.

**Environment(Umgebung) with and without CDS** Neue Environment(Umgebung) können mit und ohne CDS erstellt werden.



In einer Umgebung ohne CDS können nur Canvas Apps erstellt werden. Mit einer CDS können Model-gestützte Apps erstellt werden.

**Typen von Environment(Umgebung)** Ein Umgebung kann verschiedenen Zwecke haben. Zur Auswahl stehen 6 Typen. Es gibt 3 bekannte Typen wie Produktion, Sandbox (Ursprünglich Develop) und Testing. Darüber hinaus gibt es noch diese haben besondere Funktionen und fall aus dem typischen Entwicklung-Testing-Deploy Prozess heraus.

**Production** Ist für die laufenden Arbeit in einer Organisation bestimmt. Es kann von einem Administrator oder einer Power Apps-Lizenz User erstellt werden. **Security:** Vollzugriff

**Sandbox** Die Umgebung ist für die Entwicklung und das Testen von Ressourcen gedacht. Um dies zu erleichtern, gibt es Funktionen wie Kopieren und Zurücksetzen. **Security:** Für das Test in Ressourcen ist nur ein Nutzer-Zugriff nötig. Um Ressourcen zu bearbeiten und erstellen ist für die Entwickler eine Environment(Umgebung) Maker Rolle notwendig. Die Erstellung von Sandbox kann unterdrückt werden. Die Umwandlung einer Sandbox in Produktion ist möglich.

**Trial/ Testing\*** Ist für kurzfristiges Test vorgesehen. Diese werden nach 30 Tage wieder entleert. Jeder Nutzer kann nur eine Umgebung erstellen. Zweck dieser ist, Proof-of-Concepts zu erstellen. **Security:** Die Erstellung kann durch den Admin blockiert werden. Diese können einstellen, dass nur Tenant Admin Trial Environment(Umgebung) erstellen können oder jeder im Tenant Bereich.

**Default/Standard\*** Dies wird für jeden Tenant/ Mandanten erstellt. Jeder lizenzierte User wird dieser Umgebung mit der Environment(Umgebung) Maker Rolle zugewiesen. **Security:** Limitierter Zugriff.

**Develop\*** Diese Umgebung kann nur von Nutzer erstellt werden, die einen Power Apps Develop Plan haben. Eine Erstellung dieser Umgebung kann nicht blockiert werden, wenn ein Nutzer die entsprechende Lizenz hat, außer durch ein Support Ticket. **Security:** Ersteller können weitere Nutzer als Environment(Umgebung) Maker hinzufügen.

**Dataverse Microsoft Teams\*** Ist für das Erstellen von Power Apps aus einem Team vorgesehen. Ein Dataverse wird automatisch erstellt, wenn eine App erstellt oder aus einem Katalog über Teams ausgewählt wird. **Security:** Eingeschränkt Möglichkeiten. Nutzer aus dem Team werden automatisch ihrer Rolle im Dataverse mit einer Sicherheitslevel ausgestattet. Änderungen sind nicht möglich.

Welcher Typ die Umgebung besitzt kann in der Auflistung eingesehen werden.

The screenshot shows the 'Environments' section in the Power Platform admin center. On the left, there's a sidebar with 'Analytics' selected under 'Capacity'. The main area displays a table with three environments listed:

Environment	Type	State
(CRM)Contoso	Production	Ready
(CRM)Contoso	Sandbox	Ready
(CRM)Contoso	Production	Ready

**Create Environment(Umgebung)** Wenn eine Umgebung mit einer Datenbank erstellt wird, muss wie oben erwähnt werden, der Type der Umgebung spezifiziert werden und unten die Option, ob diese Umgebung mit einer Datenbank erstellt wird, auf Ja gesetzt werden.

The screenshot shows the 'New environment' dialog box. It includes fields for 'Name' (Contoso), 'Type' (Production selected), 'Region' (United States - Default), 'Purpose' (empty), and a toggle for 'Create a database for this environment?' (Yes selected). At the bottom are 'Next' and 'Cancel' buttons.

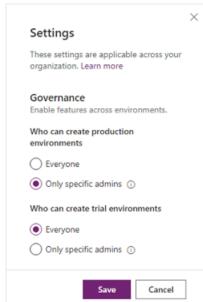
Für die Datenbank wird die Erweiterung benötigt,

- welche Sprache für die Umgebung gilt,
- wie die Firmen URL ist. Im Fall der Deutsche Bahn AG (DB) ist es *dbsw*,
- Erweiterung für Dynamics 365 und
- Sicherheitsgruppe. Wenn diese nicht angegeben wird, kann jeder unter dem Tenant die Umgebung sehen.

The screenshot shows the 'Add database' dialog box. It includes fields for 'Language' (English), 'URL' (Contoso, crm.dynamics.com), 'Currency' (USD (\$)), 'Enable Dynamics 365 apps' (Yes selected), 'Automatically deploy these apps' (Sales Enterprise), and 'Security group' (with a '+ Select' button). At the bottom are 'Save' and 'Cancel' buttons.

Ebenso ist es möglich eine Umgebung zu erstellen, welche ohne ein Dataverse erstellt wird.

Für Admins ist es möglich die Erstellung von Environment(Umgebung) zu unterdrücken. Diese können unter [admin.powerplatform.microsoft.com](https://admin.powerplatform.microsoft.com) auf Setting (nur für Admins einsehbar), die Einstellung



The following admins will be able to create new environments in the Power Platform admin center:

- Global admins
- Dynamics 365 admins
- Power Platform admins

auf *only Admins* setzen. Als Admins werden

- Global Admin,
- Dynamics 365 Admin und
- PPAdmin

verstanden. Diese können Environment(Umgebung) erstellen.<sup>4</sup>

Für Sandbox Environment(Umgebung) gilt, dass dies auch wieder reset werden können. Dies bedeutet, dass die Umgebung erhalten bleibt, aber alle Komponenten gelöscht werden.

**Convert Environment(Umgebung) Type** Neben dem Zyklus von Entwicklung-Test-Produktion können die Environment(Umgebung) Produktion und Sandbox auch ihren Type ändern. Dabei werden die Environment(Umgebung) konvertiert. Dies geht nur in der Administrator Rolle.

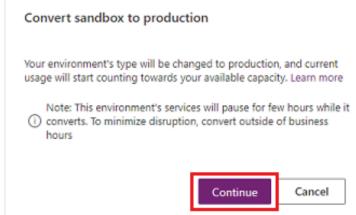
Select Convert to production or Convert to sandbox.

Es ist möglich *Sandbox* in *Produktion* umzuwandeln oder umgekehrt. Dieser Prozess kann einige Stunden dauern.

---

<sup>4</sup>Für die Kontrolle dieser Umgebung, kann *PowerShell cmdlets* heruntergeladen werden, sodass die Einstellung über die Comando Zeile erfolgt.

```
PowerShell
$settings = @{
    DisableEnvironmentCreationByNonAdminUsers =
    Set-TenantSettings $settings
```



**Adding a Database** Wurde eine Umgebung erstellt, in welcher keine Datenbank (Dataverse) erstellt wurde, kann dies auf zwei Arten erfolgen:

- Detailseite der Umgebung oder
- Daten Panal/ Enties

Auf der Hauptseite der Umgebung ist der Button *Add database* zu sehen. Diese ist nur für Admins mit der entsprechenden Lizenz zu nutzen. Wenn darauf geklickt wird, müssen die Details wie unter Literaturpar: Environment(Umgebung) with and without CDS eingegeben werden.

Im zweite Option besteht unter in der *Power Apps* Umgebung unter dem *Daten Panal* ebenfalls eine Datenbank hinzuzufügen.

1. On [make.powerapps.com](https://make.powerapps.com), expand the **Data** section and click or tap **Tables** in the left navigation pane.
2. Select **Create a database** to create the database.

**Copy Environment(Umgebung)** Dieser Zweck ist noch unklar.

**Administration Mode** Über die Einstellung einer Umgebung kann eine Umgebung (Sandbox, Trial, Produktion) in den Administrierten Modus gesetzt werden. Ist dies geschehen, können nur noch System Admins in die Umgebung sich einwählen. Hinweis. Wenn der Modus wieder aus dem Admin Modus genommen wird, kann es bis zu 24 Stunden dauern kann, bis alle Flows wieder aktiv sind.

## 1.2 Lizenzstruktur

**DB Lizenzen** Im Folgenden wird die Lizenzstruktur am Beispiel des DB Konzern aufgezeigt.  
Die angebotenen Lizenzen sind

- Power Automate & Apps Basic,
- Power Apps Full User,
- Power Automate Full User,
- Power Apps DualApp Packet und
- die Power Automate Flow Lizenz.

Die Betrachtung, welche Lizenzen benötigt werden, werden mehrere Kategorien betrachtet werden:

- Konnektoren,
- Application programming interface (API) Abfragen,
- Anwendungsfall: Automate, Apps oder beides
- Vier DB-Nutzergruppen
  - User (Benutzer)
  - Development-Verantwortliche (Anwendungsverantwortlicher)
  - Entwickler (Anwendungsentwickler)
  -

Themen Referenzen: [Lizenz Typen](#)

**Konnektoren** Bei der Betrachtung wird ebenfalls unterschieden, welcher Anwendungsfall berücksichtigt wird, Automate, Apps oder beides.

# 2 Application Lifecycle Management (ALM)

## 2.1 Allgemein

Der Application Lifecycle Management (ALM) Prozess unterstützt bei dem Zyklus der Low-Code Anwendungen der Power Platform. Die PP bittet die Entwicklung von Apps in einer einfacheren Gestalt. Die Konzepte wie Qualitätskontrollen, Governance und Continous Integration (CI) /Continous Development (CD) werden unter dem Begriff ALM zusammengefasst.

Die PP und andere Anwendungen der Microsoft Umgebung helfen den Lebenszyklus von PP Anwendungen in einer nachhaltigeren Weise umzusetzen.



Abbildung 105: Application Lifecycle

Für ALM steht im Kern

*No code development != No DevOps*

Die folgenden Unterkapitel greifen die Schwerpunkte des ALM in PP auf.  
Mehr zu dem Thema unter [Solution Lebenszyklus](#)

## 2.2 Solution

### 2.2.1 Un- and Managed Solution

Solutions sind Pakete, welche von einer oder mehreren Applikationen verwendet werden. Eine Solution dient als Transportmittel, welche alle relevanten Komponenten mindestens einer Applikation enthält. Eine Solution wird von einem *Publisher* veröffentlicht und befindet sich in einer Environment(Umgebung). Was eine Solution enthalten kann, ist vielfältig. Von einem Data Model, den User Interface (UI) Komponenten, Prozess Stritt und anderen relevanten Komponenten einer Applikation.



Abbildung 106: Inhalte einer Solution

Um Solutions zwischen Environments(Umgebungen) zu transportieren wird die Export Funktion in der Web Applikation verwendet. Später wird gezeigt, wie dies über Azure funktioniert. Wenn eine Solution in der Entwicklungsumgebung erstellt wird hat diese automatisch die Eigenschaft *unmanaged*. Bei Export kann zwischen *un-* und *managed* ausgewählt werden.

Eine *unmanaged Solution* ist für die Weiterbearbeitung freigeschalten. Eine *managed Solution* erlaubt eine Bearbeitung der Bestandteile in ihr nicht.

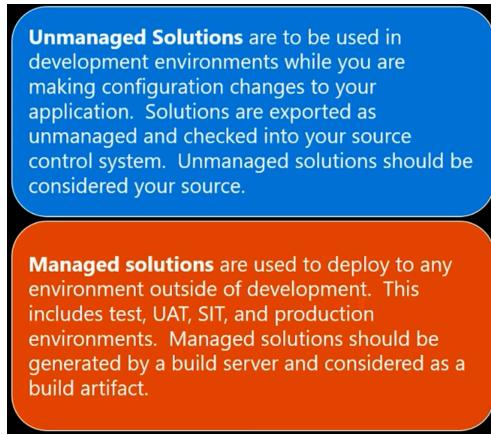


Abbildung 107: Managed and Unmanaged Solution

Im späteren Prozess wird gezeigt, wie ein unmanaged Solution in ein Git Repository geladen wird und dort als *Source of Truth* dient. Eine *managed Solution* wird als ein Build Artifacts verstanden und soll in nur in Test und Produktionsumgebungen überführt werden.

Wir eine Solution erstellt, wird die Versionnummer mit 1.0.0.0 festgelegt und ein Name sowie Publisher werden ausgewählt.

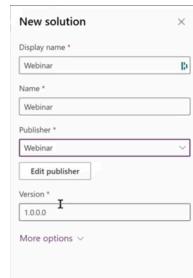


Abbildung 108: Neu Solution

Die Versionsnummer Logik ist <major>.<minor> ist für Kombinierungen von Patches oder größere Änderungen an der Solution. <build>.<revision> werden für Patches verwendet.

Die Eigenschaft, ob eine Solution managed oder unmanaged ist, wird mit dem Export einer Solution festgelegt.

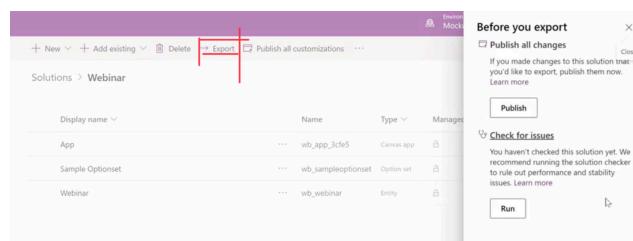


Abbildung 109: Step 1 of Export

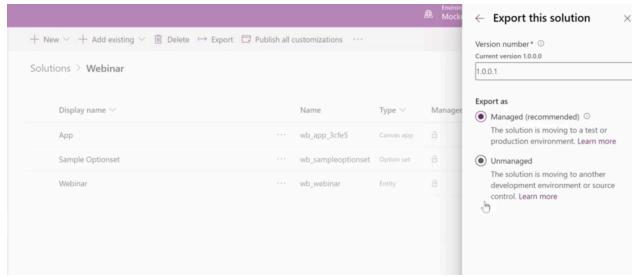
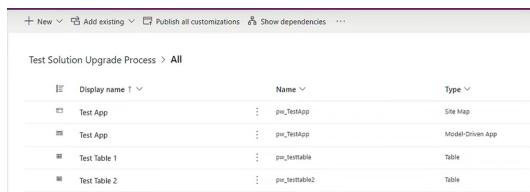


Abbildung 110: Step 2 of Export

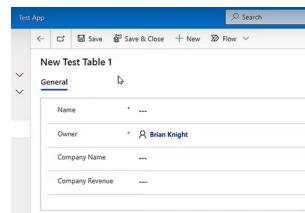
### 2.2.2 Update Solution through Layering

**Clone a patch** Die Patch Funktion bietet an, dass Komponenten ergänzt oder geändert werden. Löschen von Komponenten ist nicht möglich. Ebenso kann ein Patch nur angewandt werden, wenn es die gleiche <major>.<minor> Nummer hat.

In der Solution befindet sich eine *Site Map*, eine *Model-Driven App* und zwei *Table*.



In der Produktion zeigt die Test-App vier Einträge. Der vierte Eintrag ist *Company Revenue*. Dieser soll über ein Patch geändert werden.



In der Entwicklungsumgebung wird ein Solution von der Parent Solution *Test Solution Upgrade Process* erstellt. Warum die Änderungen nicht direkt in der Solution getätigkt wird, ist, dass noch weitere Änderungen vorgenommen wurden, welche noch nicht für die Produktion fertig sind.

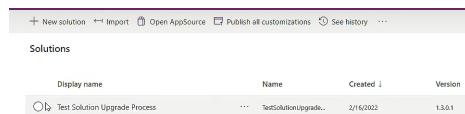


Abbildung 111: Parent Solution

Mit dem Clone für den Patch

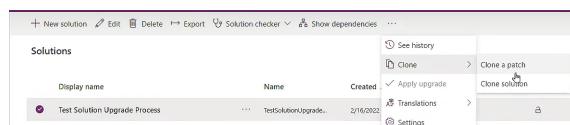


Abbildung 112: Clone a patch



kann eine neue Displayname erstellt werden und eine Inkrementierung für die Versionsnummer für <build>.<revision> erstellt werden.

Das Result ist eine neue Solution mit 1.3.1.1

Diese ist jedoch leer

Es enthält das Schema der Parent Solution. Somit kann *Test Table 1* aus dem Dataverse hinzugefügt werden. Dies gilt für alle anderen Komponenten einer Solution ebenfalls.

Dies ist jetzt in der Solution zu finden.

Das Ziel war, die Spalte *Company Revenue* zu ändern.

In der Form Ansicht ist der Wert der Spalte zu finden.

Dieser wird von *Company Revenue* zu *Company Money* geändert. Die App wird daraufhin gespeichert und die Solution exportiert mit dem Inkrement 1.3.1.2



In der Produktion oder Testumgebung wird dieser Patch importiert.

Die App in der Produktion oder Testumgebung zeigt jetzt die Änderungen mit dem Spaltennamen *Company Money*.

**Clone Solution - Merging** Wenn die komplette Entwicklung abgeschlossen ist in *Test Solution Upgrade Process* kann mit *Clone Solution* ein *Merge* zwischen der *Parent Solution* und der einen oder mehreren Patches angestoßen werden. Dafür wird unter der Parent Solution die Option *Clone Solution* ausgewählt.

Solutions				
Display name	Name	Created	Version	
Test Solution Upgrade Process Urgent Patch	... TestSolutionUpgrade...	3/2/2022	1.3.1.2	
Test Solution Upgrade Process	... TestSolutionUpgrade...	2/16/2022	1.3.0.1	
GWWR Child Solution 2	... GWWRChildSolution2	2/16/2022	1.0.0.2	
GWWR Parent Solution 1	... GWWRParentSolution1	2/16/2022	1.0.0.1	
Project Proposals	... ProjectProposals	1/4/2022	1.0.0.1	
Microsoft Flow Approvals	... MicrosoftFlowApprovals	12/8/2021	2.0.3.1	
Microsoft Flow Approvals Core Solution	... MicrosoftFlowApprovalsCoreSolution	12/8/2021	2.0.3.1	
Approvals Test	... ApprovalsTest	12/8/2021	2.0.3.1	
Dataverse Base Portal	... DataverseBasePortal	7/14/2021	9.2.2109.20	
Dynamics 365 Portals - Bot	... Dynamics365PortalsBot	7/14/2021	9.2.2109.20	

Abbildung 113: Vor dem Merge

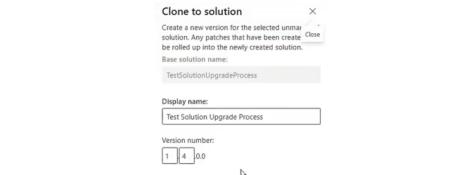


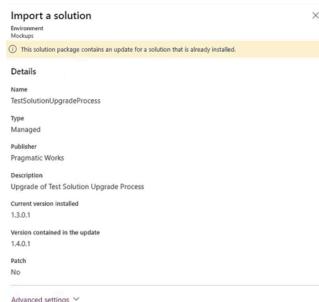
Abbildung 114: Für dies Version kann nur <major>.<minor> ausgewählt werden.



Solutions				
Display name	Name	Created	Version	
Test Solution Upgrade Process	... TestSolutionUpgrade...	2/16/2022	1.4.0.0	
GWWR Child Solution 2	... GWWRChildSolution2	2/16/2022	1.0.0.2	

Abbildung 115: Nach dem Merge

Nach dem Merge existiert nur noch eine Solution, welche mit der Versionsnummer 1.4.0.0 versehen ist. Angenommen in der Solution wird in der Entwicklungsumgebung noch weitere Änderungen vorgenommen, so werden diese ebenfalls mit integriert. Wird die gemergte Solution exportiert, wird diese mit 1.4.0.1 in die Produktions- oder Testumgebung importiert

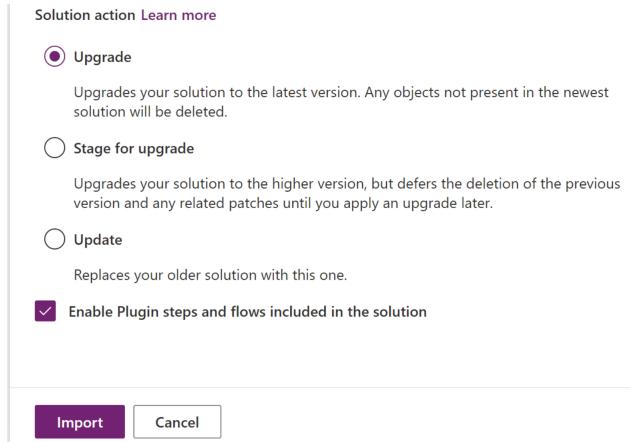


In der Zielumgebung verschwinden die Patch Solution ebenfalls, sodass nur noch die Solution mit der Versionsnummer 1.4.0.1 überbleibt.

**Layering** Die verschiedenen Solution werden in Dataverse in der Zielumgebung installiert, Updates, Upgrades oder Patches werden hier nach einander geschalten. Der Aufbau ist, dass ungemanagete Solution als oberste Schicht angelegt werden, danach kommen die gemanagten Lösungen, welche gemäß ihrer Versionsnummer gestappelt werden.

Mehr zu dem Thema und dem Verhalten verschiedenster Merge Probleme unter [Updates und Layering](#),

**Upgrade or Update** Der Unterschied ist nur, dass bei einem Upgrade, die Komponenten, welche nicht in der Zielsolutionumgebung vorhanden sind gelöscht werden. Bei einem Update bleiben diese erhalten.



## 2.3 ALM Strategy

Die hängt davon ab, wie groß und verknüpft die Applikationen mit einander sind.

Thema Referenzen: [Split Solution](#), [Solution Layer](#)

**Azure - Source of Truth - Repo** Die Allgemeinen Vorteile einer Zentralen Verwaltung des Quell-Codes sind:

- Deploy Hotfixes, während noch an der nächsten Version gearbeitet wird. Aus dem Repository werden die Solution gespeist.
- Mehrer Entwickler können an einer App bauen, welche über den ALM Prozess im Quell Verzeichnis zusammengefügt werden. Von dort aus, wird der Merge Prozess gestartet und die Solution gebildet.

Das Ziel ist dabei nicht eine Historie der Zip-Dateipacketen, sondern einer Verwaltung der einzelnen Dateien. Azure Pipeline bietet **Solution Packager**, welche die Zip-Files entpackt und in den Bestandteile abspeichert. **Solution Packager** kontrolliert Konfiguration und Performance Parameter. Zum Beispiel wird kontrolliert, dass die alle Bestandteile in einer Solution des gleichen Publisher stammen, somit dies bewegt werden können.

## 3 Building

### 3.1 Open Topics

**Connector** Diese ins API-Wrapper, welche die Zugangsdaten für die API regeln. Diese können in einer Solution separat hinterlegt werden, siehe [Create Connectionreference](#)

**Developer Terminology** Unter [Terminology](#) können die verschiedenen Bezeichnungen für Tabellen, Reihen, Spalten und weiteres in den verschiedensten Umgebungen zu sehen sein.

Power Apps UI	Dataverse SDK	Dataverse Web API
Table	Entity	EntityType
Column	Attribute	Property
Row	Record	Record
Choices	OptionSet/Picklist	OptionSet
Yes/No	Boolean	Boolean

Abbildung 116: Developer Terminology

### Fragestellungen

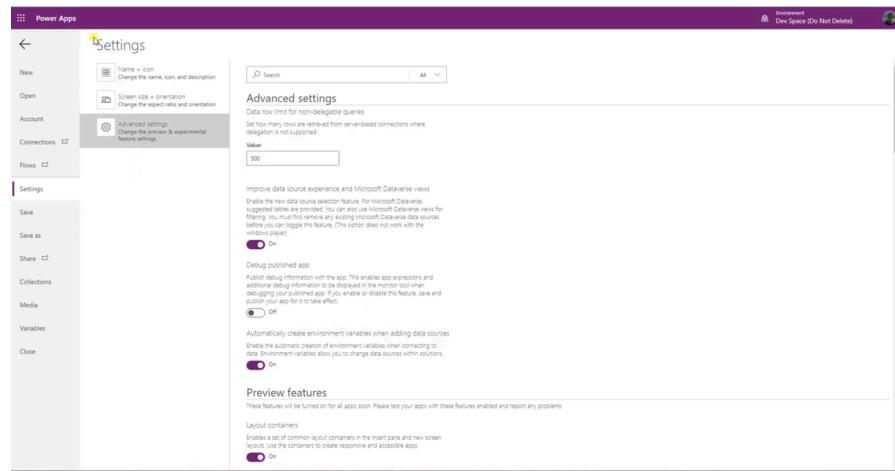
1. Die Splitting Solution ALM Strategy ist nicht klar, wie Komponenten aus einer Solution von einer App in einer andern Solution aufgegriffen werden können.

2. Test von Layering. Im direkten ALM Prozess der DB, die Upgrade Funktion testen.
3. Environment(Umgebung) Variablen: Was passiert, wenn man nicht alle Komponenten (Add required Component) mit in die Solution genommen hat?

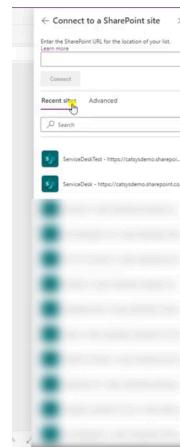
## 3.2 Environment Variables

**SharePoint Connector in Canvas App** Siehe [Environment Variables](#)

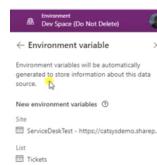
Um Connection Referenzen gleich mit einer Environment Variable zu verbinden, wird in der App unter Einstellung die Option: Erstelle automatische eine Environment Variable angeklickt.



Wenn sich mit dem SharePoint Connector verbunden wird, kann unter der Referenz eine spezifische Liste ausgewählt werden. Unter Advance kann, wenn vorhanden, eine Environment(Umgebung) Variable auswählen werden.



Wenn die Verbindung angelegt ist, wird angezeigt, dass eine Environment(Umgebung) Variable erstellt wurde.

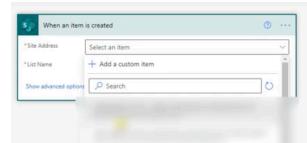


In der Solution sind zwei Variablen für den SharePoint Konektor angelegt worden.

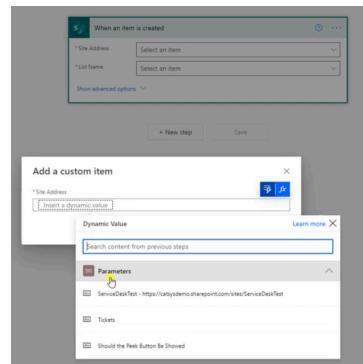
Display name	Name	Type
Service Desk App	new_servicedeskapp_c58a9	Canvas app
ServiceDeskTest - https://catysdemo.sharepoint.com/_s...	new_shared_sharepointonline_785a1950350a41035454b2b96a	Environment variable
Tickets	new_shared_sharepointonline_f6442f14ab7a4cd1b3fa1a5baa2	Environment variable

Die erste Environment(Umgebung) Variablen ist für den Ort, in welcher die Liste liegt. Die zweite für den Namen der Liste.

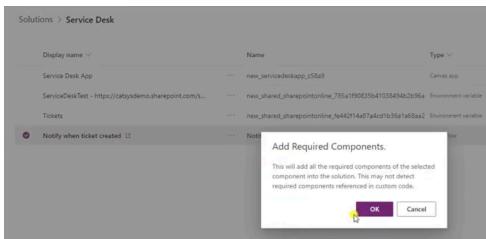
**Flow** Soll ein Nutzer informiert werden, wenn eine neues Element erstellt wird, wird ein Flow angelegt. Dieser verweist auf die SharePoint Seite. Anstatt die Adresse und Namen für die Liste zu hinterlegen,



wird auf *Add custom item* geklickt, kann unter Parameters, die hinterlegten Variablen ausgewählt werden.



**Solution Required Connectoren** Um sicherzustellen, dass alle Konnektoren in der Solution liegen, wird auf das Element, in dem Fall Canvas Apps und den Flow, unter *Add required Components*



geklickt. Für den Flow wird der Outlook und SharePoint Connecker hinzugefügt.

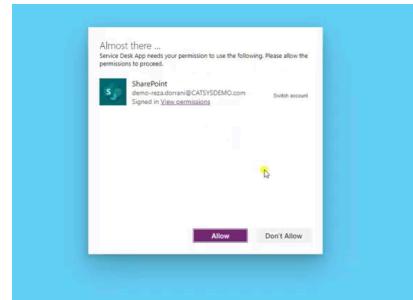
Solutions > Service Desk			
Display name	Name	Type	Manage
Service Desk App	new_servicedeskapp_c58a9	Canvas app	
ServiceDeskTest - https://catysdemo.sharepoint.com/_s...	new_shared_sharepointonline_785a1950350a41035454b2b96a	Environment variable	
Tickets	new_shared_sharepointonline_f6442f14ab7a4cd1b3fa1a5baa2	Environment variable	
Notify when ticket created	Notify when ticket created	Cloud flow	
Office 365 Outlook	Office 365 Outlook	Connection reference (previous)	
SharePoint	SharePoint	Connection reference (previous)	

**Delete Current Selection** Die aktuelle Auswahl der Environment(Umgebung) Variable wird vor dem Export gelöscht. Dabei besteht die Möglichkeit den hinterlegten Wert aus der Solution oder aus der Environment(Umgebung) zu löschen.

**Import Solution** Wenn die Solution in die Test oder Produktions Environment(Umgebung) geladen wird, wird nach den Connections References gefragt.

Im Folgenden wird nach den Environment(Umgebung) Variablen gefragt:

Zum Ende, wenn die Application gestartet wird, wird nach dem Zertifikat für gewissen Konnektoren gefragt.



## **Teil III**

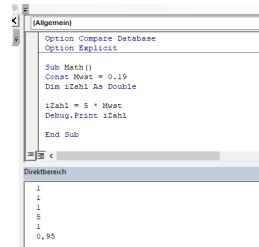
### **VBA**

# 1 Das VBA-Tutorial

## 1.1 Grundlagen

### 1.1.1 Debug.Print

Diese Funktion *Debug* in Visual Basic Application (VBA) ist ein Objekt, welches zwei Methoden besitzt. Diese sind *Print* und *Assert*. Die *Print* Methode erlaubt das einfache Wiedergeben der Werte von Variablen in VBA. Der Vorteil gegenüber *MsgBox* ist, dass eine Bestätigung der Ausgabe erfolgen muss. Ebenso bleibt die Darstellung im Direktfenster nach Beendigung der Prozedur erhalten.



```
(Allgemein)
Option Compare Database
Option Explicit

Sub Math()
    Const Mwst = 0.19
    Dim iZahl As Double
    iZahl = 5 * Mwst
    Debug.Print iZahl
End Sub

Direktbereich
l
l
l
S
l
0,95
```

Abbildung 117:

### 1.1.2 Array

Ein Array in VBA wird über () indiziert. Dabei kann ein Array jeglichen Datentypen enthalten.

---

```
1 Option Compare Database
2 Option Explicit
3
4 Sub Math()
5
6     Dim sAusgabe(2) As String
7
8     sAusgabe(0) = "Test im Feld 0"
9     Debug.Print sAusgabe(0) ' Test im Feld 0
10
11    sAusgabe(1) = "Test im Feld 1" 'Test im Feld 1
12    Debug.Print sAusgabe(1)
13
14 End Sub
```

---

Die Dimensionalisierung des Arrays kann auch über ein Variable erfolgt. Ebenso kann das Array auch neu definiert werden. Dafür wird *ReDim* verwendet. Dim iAnzahl As Integer

---

```
1 Sub Math()
2
3     iAnzahl = 3
4     ReDim sFeld(iAnzahl)
5     sFeld(0) = "Feld 0"
6     Debug.Print sFeld(0) ' Feld 0
7
8     iAnzahl = 4
9     ReDim sFeld(iAnzahl)
10    Debug.Print sFeld(0) ' Leer
11
12 End Sub
```

---

## 1.2 Objekte

### 1.2.1 Klassen

Wie auch in der Objekt-Orientierte-Programmierung (OOP) können über VBA Klassen erstellt werden. Dabei wird das *Klassen-Module* verwendet. Der Name des Klassenmoduls im *Klassenmodule* Ordner ist das Objekt.

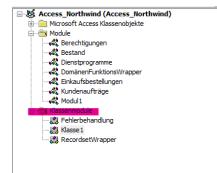


Abbildung 118:

Am Beispiel des *Autos* wird die Klasse **Auto** geschaffen. Die Klasse ist zwar leer, aber in VBA wird sie schon als Klasse erkannt. Dies kann nachvollzogen werden, wenn in einem *Standardmodul* eine Variable vom Typ der Klasse Auto erzeugt wird.

```
(Allgemein) (Deklarationen)
Option Compare Database
Option Explicit
Dim Opel As Auto
```

Abbildung 119:

Das Symbole

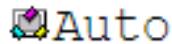


Abbildung 120:

zeigt dabei Auto als Objekt an. **Wichtig:** Die Dimensionierung einer Objektvariablen ist nur ein *Verweis*. Die Erzeugung des Objekte wird durch

$$\text{Set} \dots = \text{New} \text{ Auto} \quad (1.1)$$

instanziert. Sollen mehrere Auto erzeugt werden, so lässt sich dies durch weitere Objektverweise erzeugen.

---

```
1 Dim Opel As Auto
2 Dim VW As Auto
3 Dim Ferrari As Auto
```

---

Wie schon erwähnt, sind die Objekte noch nicht instanziert. Es folgt:

---

```
1 'Objektverweise
2 Dim Opel As Auto
3 Dim VW As Auto
4 Dim Ferrari As Auto
5
6 'Instanzisierung
7 Set Opel = New Auto
8 Set VW = New Auto
9 Set Ferrari = New Auto
```

---

Der Vorgang kann ebenso auch in einer Zeile erfolgen.

---

```
1 Dim Opel As New Auto
```

---

Man spricht auch hier von *Objektdeklaration*.

## 1.2.2 Methoden

Methoden werden in VBA mit dem Symbole

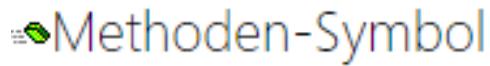


Abbildung 121:

dargestellt.

Für die Klasse `Auto` wird die Methode `hupen` definiert. Diese ist der Klasse zu gewissen und kann über die Klasse abgerufen werden. Eine Methode wird mit einer Prozedur definiert.

Two side-by-side screenshots of Microsoft Word VBA code editors. The left window shows a function named 'hupen' with an optional parameter 'Sekunden'. The right window shows a subroutine named 'Autofahren' which creates an object 'Opel' and calls its 'hupen' method.

Abbildung 122:

Die Funktion (Mehtode) `hupen()` verwendet eine Optionale Variable. Dabei wird mit `= 2` der voreingestellt Wert definiert, welcher verwendet wird, wenn die kein Wert übergeben wird. Oder kurzgesagt: *Default*. Die Syntax wie Variablen eingegeben werden, kann mit `()` oder ohne erfolgen.

## 1.2.3 Eigenschaft

**Property** Zusätzlich zu den Methoden einer Klasse, können auch Eigenschaften definiert werden. Im einfachsten Fall sind dies Intrinsic Variablen. Es können aber auch weitere Objekte als Eigenschaft definiert werden. Eigenschaften werden in mit Intellisense mit dem Symbole:

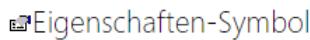


Abbildung 123:

dargestellt. Damit werden einzelnen Variablen, Unterobjekte und Property Prozeduren angesprochen. Wird das Objekt `Opel` der Klasse `Auto` über ein Modul initialisiert, so stehen alle Eigenschaften und Methoden.

A screenshot of a Microsoft Word VBA code editor showing a property procedure named 'AutoFahren'. It includes code for creating an object 'Opel', setting its color to green, and printing the color.

Abbildung 124:

**Property Prozedur** Die Eigenschaften zu einer Klasse können **public** oder **privat** sein. Wenn sie öffentlich hinterlegt sind, können diese über den Klassenverweis direkt geändert werden. Variablen die **private** sind, können nur über die Klasse selber geändert werden.

Die **Property Let/ Get** Prozedure hilft die Eingabe von Daten für die Eigenschaften zu verwalten. Dies bittet sich vorwiegend für Variablen vom Typ **private** an. Eine mögliche Variante die Verwaltung zu steuern, ist die privaten Variablen mit dem Prefix **pv** zu versehen. Die Prozedur hingegen, kann den Variablennamen tragen ohne Prefix. Gleichnamigkeit der Variable und der dazugehörigen Prozedur ist nicht vorgesehen. Es ist aber möglich, das **Get** und **Let** den gleichen Namen tragen. Wie schon erwähnt, werden die Prozeduren als Eigenschaften geführt und nicht als Methoden angezeigt.

- Verwendet man nur **Property Let**, so ist eine Konfiguration der Variable vorgesehen, aber ein Auslesen nicht.
- **Property Set** erlaubt eine Adressierung von Referenzen von Objekten.
- Verwendet man nur **Property Get**, so ist eine **private** Variable schreibgeschützt.

Am Beispiel des Objektes *Opel* werden die Eigenschaften über die **Property** Prozedur verwaltet. Bei **Get** wird auch der Rückgabewert definiert. Dabei verhält sich **Get** wie eine Funktion und **Let** wie ein **Sub**.

```

Access_Northwind - Auto (Code)
[Allgemein] Get_Geschwindigkeit [PropertyGet]
Public strFarbe As String
Private bytTempo As Byte
Private bytSperre As Boolean

Public Function hugen(Optional Sekunden As Integer = 2) As Boolean
    'Funktionsweise
    Debug.Print Sekunden
    'Miscargen
    hugen = True
End Function

Public Property Let Let_Geschwindigkeit(Tempo As Byte)
    If Tempo > 250 Then
        bytTempo = 250
        bytSperre = True
    Else
        bytTempo = Tempo
        bytSperre = False
    End If
End Property

Public Property Get Get_Geschwindigkeit() As Long
    Get_Geschwindigkeit = bytTempo
    ' The default value for a number is 0.
End Property

Access_Northwind - Modul1 (Code)
[Allgemein] Autofahren
Option Compare Database
Option Explicit

Public Sub Autofahren()
    'Objektverweis
    Dim Opel As Auto
    'Objektedeklaration
    Set Opel = New Auto
    'Auszuführen
    Opel.hugen (1)
    Opel.hugen
    Opel.strFarbe = "Schwarz"
    Debug.Print Opel.strFarbe
    Opel.Let_Geschwindigkeit = 170
    Debug.Print Opel.Get_Geschwindigkeit
End Sub

```

Abbildung 125:

Für die Deklaration von Eigenschaften wird meist die **Property** Prozedur verwendet. Eine Deklaration außerhalb einer Prozedur ist nicht in VBA vorgesehen.

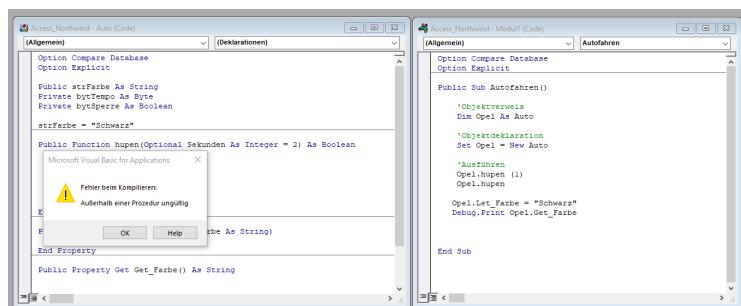


Abbildung 126: Keine Deklaration der Variablen außerhalb einer Prozedur

Innerhalb der Methoden kann dies jedoch vorgenommen werden.

```

'Property *****
Public Property Let Let_Farbe(eigFarbe As String)
    strFarbe = eigFarbe
End Property

Public Property Get Get_Farbe() As String
    Get_Farbe = strFarbe
End Property

```

Neukontext  
1  
2  
Schwarz

```

'Ausführen
Opel.hupen (1)
Opel.hupen

Opel.Let_Farbe = "Schwarz"
Debug.Print Opel.Get_Farbe

End Sub

```

Abbildung 127: Deklaration der Variablen mit Hilfe von [Property](#)

Erfolgt die Deklaration in dem *Modul*, so ist nichts weiter zu tun. Der große Vorteil bei der [Property](#) Prozedur ist, dass für [Let](#) und [Get](#) der gleiche Name verwendet werden kann. Es kann somit eine [private](#) Variable definiert werden, und diese über eine [Get](#) und [Let](#) abgerufen und verändert werden.

```

Option Compare Database
Option Explicit

Public strFarbe As String
Private byTempo As Byte
Private mySpuren As Boolean
Private myAirbags As New Collection

'Property *****
Public Property Let Tempo(getTempo As Byte)
    byTempo = getTempo
End Property

Public Property Get Tempo() As Byte
    Tempo = byTempo
End Property

```

Abbildung 128:

**Unterobjekte** Das Ziel soll sein, das ein Objekt ebenfalls aus mehreren Unterobjekten bestehen kann. Dafür werden Eigenschaften verwandt, welche als Objektevariablen definiert werden.

Die Klasse Radio wird mit einer Eigenschaft und einer Methode gefüllt.

```

Option Compare Database
Option Explicit

Public mName As String

Public Property Let Name(pName As String)
    mName = pName
End Property

Public Property Get Name() As String
    Name = mName
End Property

Public Function einschalten() As Boolean
    Debug.Print "Radio ist eingeschalten."
    einschalten = True
End Function

```

Abbildung 129:

In der Klasse Auto wird die Eigenschaft *ppRadio* vom Typ [Radio](#) deklariert. Damit zu jedem Objekt ein Radio initialisiert wird, wird die *Class\_Initialize* Prozedur aufgerufen. Diese wird immer angestoßen, wenn ein neues Objekt initialisiert wird.

```

Option Compare Database
Option Explicit

'Deklaration Variablen (Welche gibt es)
'.....
Public Name As String
Private ppRadio As Radio

'Get and Define
'.....
'Initialise Class
Private Sub Class_Initialize()
    Set ppRadio = New Radio
End Sub

'Gets Property Object Radio
Public Property Get Radio() As Radio
    Set Radio = ppRadio
End Property

```

Abbildung 130:

Mit dem Ausdruck `SetppRadio = New Radio` wird das Unterobjekt initialisiert. Die Get-Prozedur gibt den Verweis wieder. Dabei ist die Get-Prozedur vom Typ Radio. Mit `SetRadio = ppRadio` wird der Verweis an die Funktion übergeben. Hier müsste nochmal geprüft werden, ob Set benötigt wird.  
Am Ende kann das Radio mit seinen Eigenschaften und Methoden zu jedem initialisierten Auto abgerufen werden.

```

Option Compare Database
Option Explicit

Public Sub Autofahren()
    'Objektverweis
    Dim Opel As Auto
    Set Opel = New Auto

    'Test
    Opel.Name = "Objekt"
    Debug.Print Opel.Name
    Opel.Radio.Name = "Radioname"
    Debug.Print Opel.Radio.Name

End Sub

```

Abbildung 131:

#### 1.2.4 Auflistung

Eine **Auflistung** ist ein Objekt, welches vor definierte Methoden hat:

- Item - Diese erlaubt eine Zugriff auf die Elemente in *Collection* Objekt.
- Count - Zählt wie viele Elemente eine Auflistung besitzt.
- Add - Fügt ein Element der Auflistung hinzu.
- Remove - Entfernt ein Element aus der Auflistung.

```

Option Compare Database
Option Explicit

Public Sub Autofahren()
    'Objektverweis
    Dim Opel As Auto
    Set Opel = New Auto

    'Test
    Opel.Name = "Objekt"
    Debug.Print Opel.Name
    Opel.Radio.Name = "Radioname"
    Debug.Print Opel.Radio.Name
    Opel.Airbagseinstellen
    Debug.Print Opel.Airbags.Count
    Debug.Print Opel.Airbags
End Sub

```

Abbildung 132:

- 

An Hand des Auto-Beispiels muss Auflistung deklariert:

```

Public Property Get Airbags() As Collection
    Set Airbags = pVAirbags
End Property

```

Abbildung 133:

und initialisiert

```

Public Name As String
Private ppRadio As Radio
Private pVAirbags As Collection

'get and define*****
'*****initialize class
Private Sub Class_Initialize()
    Set ppRadio = New Radio
    Set pVAirbags = New Collection
End Sub

'sets Property Object Radio

```

Abbildung 134:

werden. In dem genannten Beispiel sollen zwei Airbags eingebaut werden. Dafür greift die Klasse Auto auf die Klasse Airbag zu und speichert diese im Objekt Airbags.

```

Public Sub Airbagseinbauen()
    Dim Airbag1 As Airbag
    Dim Airbag2 As Airbag
    Set Airbag1 = New Airbag
    Set Airbag2 = New Airbag
    Airbag1.Name = "Fahrer"
    Airbag2.Name = "Beifahrer"
    pvaIRbags.Add Airbag1, "Fahrer"
    pvaIRbags.Add Airbag2, "Beifahrer"
End Sub

```

Abbildung 135:

Über das Modul kann die Auflistung angesteuert werden, dafür gibt es verschiedene Syntaxen für eine Aufrufung der einzelne Elemente.

```

Opel.Airbagseinbauen
Debug.Print Opel.Airbags.Count
Debug.Print Opel.Airbags(1).Name
Debug.Print Opel.Airbags(1).Name
Debug.Print Opel.Airbags("Fahrer").Name

```

Abbildung 136:

### 1.2.5 Ereignisse

Ereignisse sind Prozeduren, welche nicht über das Objekt selbst abgerufen werden können, sondern durch Vorgänge mit dem Objekt hervorgerufen werden. Zwei Prozeduren haben wir schon kennengelernt. Für **Ereignisse** wird das Symbol



Abbildung 137:

verwendet. Im Objektkatalog wird diese neben Eigenschaften und Methoden angezeigt.



Abbildung 138:

Unter einem VBA-Application Objekt findet auf der linken Seite des Drop-Down

## 1.2.6 Objektanweisung

Zwei Anweisungen, **With... End With** und **For Each**, sind für das Referenzieren von Objekten wichtig. Es kann ein spezifisches Objekt angesteuert mit der With-Umgebung:

```
1 With Objekt  
2 .Methode  
3 .Eigenschaft  
4 .Objekt.Methode  
5 End With
```

Als Beispiel

```
1 Private Sub AddCustomer()  
2 Dim theCustomer As New Customer  
3  
4 With theCustomer  
5 .Name = "Coho Vineyard"  
6 .URL = "http://www.cohovineyard.com/"  
7 .City = "Redmond"  
8 End With  
9  
10 With theCustomer.Comments  
11 .Add("First comment.")  
12 .Add("Second comment.")  
13 End With  
14 End Sub  
15  
16 Public Class Customer  
17 Public Property Name As String  
18 Public Property City As String  
19 Public Property URL As String  
20  
21 Public Property Comments As New List(Of String)  
22 End Class
```

Mit dem **For Each** Loop können die verschiedenen Elemente in einer Auflistung angesteuert werden.

```
For Each Element-Objekt in Auflistung  
'Statement  
Next
```

Dies geschieht solange bis alle Objekt in der Auflistung durchlaufen sind. Anhand des Beispiels zum Airbagseinbau können muss ein Airbag Objekt zu erst separat deklariert werden.

```
1 Public Sub Autofahren()  
2  
3 'Objektverweis  
4 Dim Opel As Auto  
5 Set Opel = New Auto  
6 Dim LAirbag As Airbag  
7  
8  
9 'Test  
10 Opel.Name = "Objekt"  
11 Debug.Print Opel.Name  
12 Opel.Radio.Name = "Radioname"  
13 Debug.Print Opel.Radio.Name  
14 Opel.Airbagseinbauen  
15 Debug.Print Opel.Airbags.Count  
16 Debug.Print Opel.Airbags.Item(1).Name  
17 Debug.Print Opel.Airbags(1).Name  
18 Debug.Print Opel.Airbags("CFahrer").Name  
19 Debug.Print Opel.Airbags.Item(2).Name  
20 Debug.Print Opel.Airbags.Item(2).einbauen  
21 For Each LAirbag In Opel.Airbags  
22 Debug.Print LAirbag.Name 'Fahrer, Beifahrer  
23 Next  
24 End Sub
```

### 1.2.7 Me-Object

Wenn Makros geschrieben werden, kann dies in zwei Formen getan werden:

- Modules
- Class-Modules
  - Mit einem Interface
  - Ohne Interface

Es gibt also Klassen-Module, welche ein Interface-Design mit sich bringen. Das *ME*-Objekt hilft, die Adressierung in einem Module zu verallgemeinern. In Excel werden Tabellenblätter als *Excel-Klassenobjekte* in VBA. Am Beispiel von Northwind wird gezeigt:

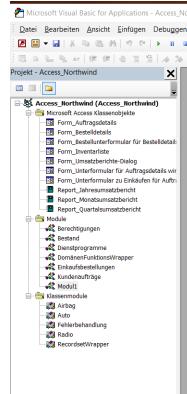


Abbildung 139:

Access weiß Form und Reports als Klassenobjekte aus. Die weiteren Module wurden ergänzt und die Klassenmodul sind die Objekte, welcher selber gewählt wurden. Wird ein Marco geschrieben, welches sich in einem Klassenmodule befindet, so kann das Objekt direkt mit *ME*. angesteuert werden.

Am Beispiel von Excel wird ein Wert "Hello Friends" der Zelle "A1" des Datenblattes "Dat Sheet" gespeichert.

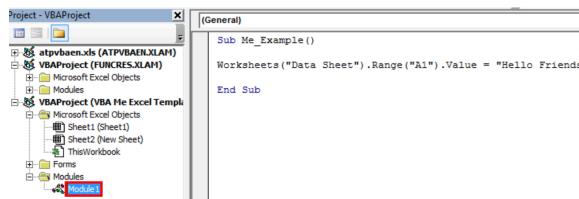


Abbildung 140:

Dieser Code könnte jedoch auch im Microsoft Excel Objekt *Sheet2 (Data Sheet)* (Achtung: Im Bild steht "New Sheet") eingetragen werden. Die Adressierung des Objekts mit

<sup>1</sup> `Worksheet ("Data Sheet") .`

könnte auch über

<sup>1</sup> `Me .`

erfolgen. Intellisense erkennt dies ebenfalls, und weiß die zugehörigen Methoden und Ereignisse aus.

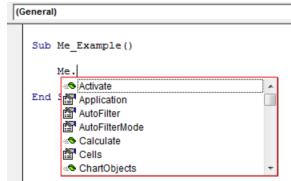


Abbildung 141:

Das Sub im Klassenobjekt selbst sieht dann wie folgt aus:

---

```

1 Sub Me()
2 Me.Range("A1").Value = "Hello Friends"
3 End Sub

```

---

Der Vorteil der sich daraus ergibt, ist, dass der Code leichter wiederverwendet werden kann. Gleichermaßen gilt für *Userforms*. Ein Userform ist ein instanziertes Objekt. Objekt-Eigenschaften können erzeugt werden und haben eine visuelle Erscheinungsbild. Die Ansteuerung kann in einem Userform-Module ebenfalls über *ME*. erfolgen.

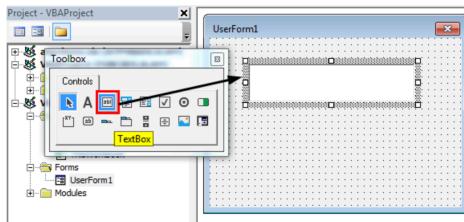


Abbildung 142:

---

```

1 Private Sub TextBox_Change()
2   UserForm1.TextBox1.Text = "Welcome to VBA!"
3 End Sub

```

---

oder

---

```

1 Private Sub TextBox_Change()
2   ME.TextBox1.Text = "Welcome to VBA!"
3 End Sub

```

---

mit dem Ergebnis

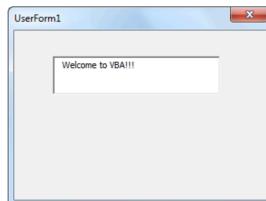


Abbildung 143:

### 1.2.8 Objektmodell

Bei der Programmierung eigener Objekte, ist es sinnvoller die Objektstruktur zu dokumentieren. Im Fall von Norhwind würde diese wie folgt aussehen:

Elemente des Auto-Objekts		
Ein Auto-Objekt verweist auf ein bestimmtes <a href="#">Auto</a> .		
Eigenschaften	Name	Beschreibung
<input checked="" type="checkbox"/> <a href="#">abgesegelt</a>	<a href="#">abgesegelt</a>	Zeigt an, ob die Geschwindigkeit abgeriegelt wurde. Boolean, schreibgeschützt.
<input checked="" type="checkbox"/> <a href="#">Airbags</a>	<a href="#">Airbags</a>	Gibt die Airbags-Auflistung des Objekts zurück.
<input checked="" type="checkbox"/> <a href="#">Farbe</a>	<a href="#">Farbe</a>	Farbe des Autos. String mit Lese-/Schreibzugriff.
<input checked="" type="checkbox"/> <a href="#">Geschwindigkeit</a>	<a href="#">Geschwindigkeit</a>	Ganzzahl, die die aktuelle Geschwindigkeit angibt. Lese-/Schreibzugriff.
<input checked="" type="checkbox"/> <a href="#">Radio</a>	<a href="#">Radio</a>	Gibt das Radio-Objekt zurück.
Methoden	Name	Beschreibung
<input checked="" type="checkbox"/> <a href="#">hupen</a>	<a href="#">hupen</a>	Verursacht ein Geräusch.
Ereignisse	Name	Beschreibung
<input checked="" type="checkbox"/> <a href="#">Initialisie</a>	<a href="#">Initialisie</a>	Wird beim Initialisieren des Autos ausgeführt.

Abbildung 144:

### 1.3 Fehlerbehandlung

**Error Resume Next** Fehler in der Zeile werden ignoriert. Die Prozedur wird daraufhin so ausgeführt, als wäre die Zeile nicht existent.

**On Error go to** Tritt ein Fehler auf, wir auf an einen definierten Punkt im Code gesprungen. Hier muss aufgepasst werden, dass die Übersichtlichkeit gewahrt wird, und die Funktion sollte nur zur Fehlerbehebung verwendet werden. Der Verweis kann ein Wort oder Zahl gefolgt von einem :.

---

```

1 Function GibFehler() As Double
2 Dim i As Double
3
4 On Error GoTo Eingabefehler
5
6 i = 1 / InputBox("Geben Sie eine Zahl ein")
7
8 GibFehler = i
9
10 Exit Function
11
12 Eingabefehler:
13 GibFehler = 0
14
15 End Function

```

---

**Resume** Mit Resume kann wieder zurück in den Code gesprungen werden. Zum Beispiel wird eine Schleife, basierend an dem Fehler weiter durchlaufen. Vermerk: **Err** Objekt fängt den Fehler einer Zeile ab. Mit **Err.Number** kann der Fehlerwert wiedergegeben werden.

---

```

1
2 Function GibFehler()
3 Dim i As Byte
4
5 On Error GoTo Eingabefehler
6
7 i = InputBox("Geben Sie eine Zahl ein")
8 i = (i ^ 2 + 1) / i
9
10 GibFehler = i
11
12 Ende:
13 Exit Function
14
15 Eingabefehler:
16 Select Case Err.Number
17     Case 6 'Überlauf (negativer Wert)
18         Resume
19     Case 11 'Division durch "0"
20         i = 0
21         Resume Next
22     Case 13 'Typen unverträglich (Text)
23         Resume Ende
24 End Select
25

```

---

## 1.4 Kurzer Einblick Formular

**Combbox** Zufügen der Werte für eine Combbox.

---

```

1 Privat Sub UserForm1_Initialize()
2 Dim Liste(2) as Variant
3
4 Liste(0) = "A"
5 Liste(1) = "B"
6 Liste(2) = "C"
7
8 Me.ComboBox1.List = Liste
9 Me.ComboBox1.AddItem ("D")
10
11 End Sub

```

---

## 1.5 Microsoft Access

### 1.5.1 Marcos, Module

Access bietet die Möglichkeit **Marcoaktionen** über ein Click-und-Drop-Modus zu generieren. Die Funktionen sind jedoch im Vergleich zu **Modulen** starr. Alle Makroaktionen sind in VBA über das **DoCmd** Objekt abrufbar.

### 1.5.2 Formulare, Reports

- Formulare werden über das das Auflistungen **Forms** angesteuert. Die Steuerelemente zugehörig zu einer Form werden über **Control** verwaltet (Collection).

---

```
1 Debug.Print Application.Forms(0).Control("txtName").Name
```

---

- Ereignisse** Die Steuerungselement eines Formulares können auf viele der Ereignisse reagieren.

Am Beispiel von Nordwind enthält das *Formular Inventarliste* viele Steuerelemente.

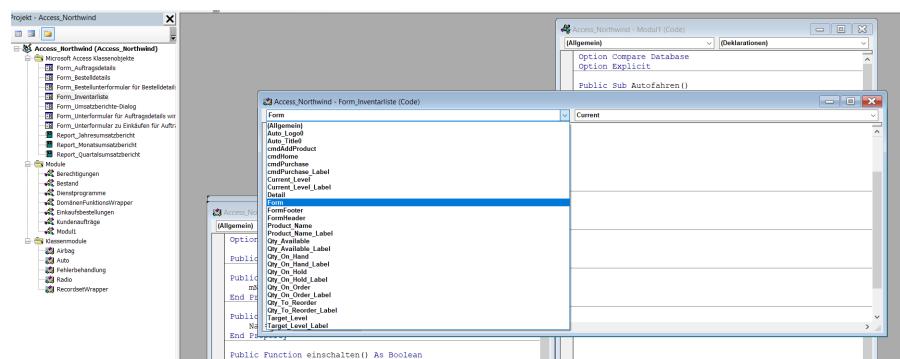


Abbildung 145:

Jedes dieser Element kann auf eine Vielzahl von **Ereignissen**.

Elemente des Auto-Objekts		
Ein Auto-Objekt verweist auf ein bestimmtes <a href="#">Auto</a> .		
Eigenschaften	Name	Beschreibung
<input checked="" type="checkbox"/> <a href="#">abgesegelt</a>	Zeigt an, ob die Geschwindigkeit abgeriegelt wurde. Boolean, schreibgeschützt.	
<input checked="" type="checkbox"/> <a href="#">Airbags</a>	Gibt die Airbags-Auflistung des Objekts zurück.	
<input checked="" type="checkbox"/> <a href="#">Farbe</a>	Farbe des Autos. String mit Lese-/Schreibzugriff.	
<input checked="" type="checkbox"/> <a href="#">Geschwindigkeit</a>	Ganzzahl, die die aktuelle Geschwindigkeit angibt. Lese-/Schreibzugriff.	
<input checked="" type="checkbox"/> <a href="#">Radio</a>	Gibt das Radio-Objekt zurück.	
Methoden	Name	Beschreibung
<input checked="" type="checkbox"/> <a href="#">hupen</a>	Verursacht ein Geräusch.	
Ereignisse	Name	Beschreibung
<input checked="" type="checkbox"/> <a href="#">Initialise</a>	Wird beim Initialisieren des Autos ausgeführt.	

Abbildung 146:

---

```

1 Option Compare Database
2 Option Explicit
3
4
5 Private Sub Form_Current()
6 Me.txtNachname.BackColor = RGB(255, 128, 128)
7 End Sub
8
9 Private Sub txtNachname_AfterUpdate()
10 Me.txtNachname.BackColor = RGB(128, 128, 255)
11 End Sub

```

---

### 1.5.3 Tabellen

- Der Zugriff auf Daten wird von Access nicht unterstützt. Dies geschieht alles über Datenbank Objekt Bibliotheken. Dabei ist die wichtigste für den einfachen Gebrauch **DAO** (Database Object).
- Für die Datenbasis in der eigenen Access-Datenbank wird gilt: DAO.Database = Currentdb
- Um ein spezifischen Datentabelle zuladen wird DAO.Recordset = Currentdb.OpenRecordset("Name")

## 2 VBA: Access

### 2.1 Sammelsurium

- Close Recordset am Ende einer Prozedur: Recordset.Close Set Recordset to nothing
- Close Database am Ende einer Prozedur, Set Variable to nothing
- Recordset.Move(Rows, [Optional]Bookmark): Rows ist ein Long-Parameter der notwendig ist.
- 
- Recordset.MoveNext() Bewegt sich
- Recordset.AddNew(): Fügt eine neue Zeile zum Datenset und macht die zugefügte Zeile zum *Current Record*.
- EOF: End of File gibt solange das Ende nicht erreicht ist den Wert `false` wieder.
- Idee: Ein Form anlegen, welcher über die verschiedensten Forms und Reports ansteuert. Ebenso, sollten die Buttons so angelegt werden, dass die man zurückspringen kann (Me-Objekt). Plus: Es sollten die anderen offenen Tabs geschlossen werden.
- DbEngine - DAO Objekt ist das übergeordnete Objekte. Es gibt auch keine Möglichkeit mehrere Instanzen davon zu erzeugen. Es existieren zwei große Auflistungen : Workspaces und Errors.
- Idee: Start-Seite: Hier werden alle wichtigen Informationen und Verlinkungen angezeigt.
- Über die Eigenschaften der Aktuellen Datenbank kann der Name, sowie weitere Eigenschaften angepasst werden.

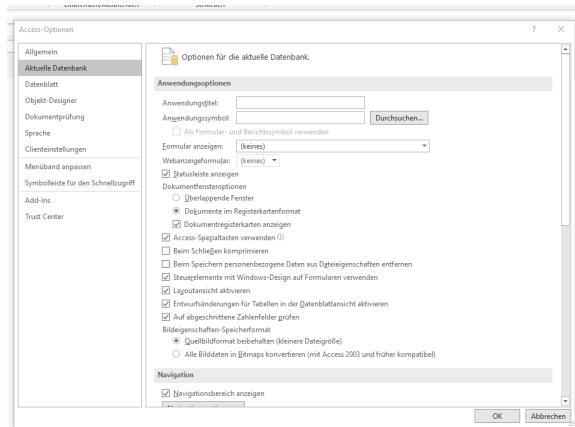


Abbildung 147:

### 2.2 Introduction VBA Basic

#### 2.2.1 VBA Syntax

**Method** siehe OOP. Diese ist eine spezifische Funktion. Diese ist nur anwendbar für ein vorgegebenes Object oder einer Familie von vererbten Objekten.

**Function** A function is a piece of code that is called by name. It can be passed data to operate on (i.e. the parameters) and can optionally return data (the return value). All data that is passed to a function is explicitly passed.

**Statement** Die ist ein einzeiliger Ausdruck. Dieser drückt meist eine Aktion oder eine Definition aus.

```
1  Dim obj As Object ' one statement
2  ...
3  If x = true then x = 1 else 5 ' one statement
```

Um mehrere Aktionen miteinander zu verbinden, wir : verwendet.

```
1   ' ...
2   rsDataSet.Close:           Set rsDataSet= Nothing
3   ' ...
```

Mit der *With* Funktion kann man mehrere statements zusammenfassen. Dabei ist es egal, das das statement aus mehreren Methoden abrufen besteht.

```
1   ' ...
2   With Currentdb.Sheet("Tabelle1").Range("B1")
3     .Color = blue
4     .Value = "This is a message."
5     .Foreground = Brushes.DarkSeaGreen
6     .Background = Brushes.LightYellow
7   End With
8   ' ...
```

Soll das statement unterbrochen werden, so verwendet man \_.

## 2.2.2 Object Modell for Access

Über die Seite zur Dokumentation für die Datenobjekte für VBA im Office Context bietet eine umfassende Sammlung aller Objekte die im Access Kontext wichtig sind.

The screenshot shows a Microsoft Docs page titled 'Access VBA reference'. The page is dated 10/10/2018 and has a reading time of 2 minutes. It features a sidebar with a navigation tree for 'Office VBA Reference' under 'Access', including sections like 'Overview', 'Concepts', 'Object model', 'Excel', etc. The main content area contains a 'Note' section with a purple background, which says: 'Interested in developing solutions that extend the Office experience across multiple platforms? Check out the new [Office Add-ins model](#)'. Below this, there's a note about using the table of contents on the left to view topics in sections like 'Concepts' and 'Object model reference'. There are also 'See also' and 'Support and feedback' links at the bottom.

Abbildung 148:

Es wird zwischen *Concepts* und *Objects* unterschieden. Im folgenden wird sich überwiegend auf Objekte fokussiert.

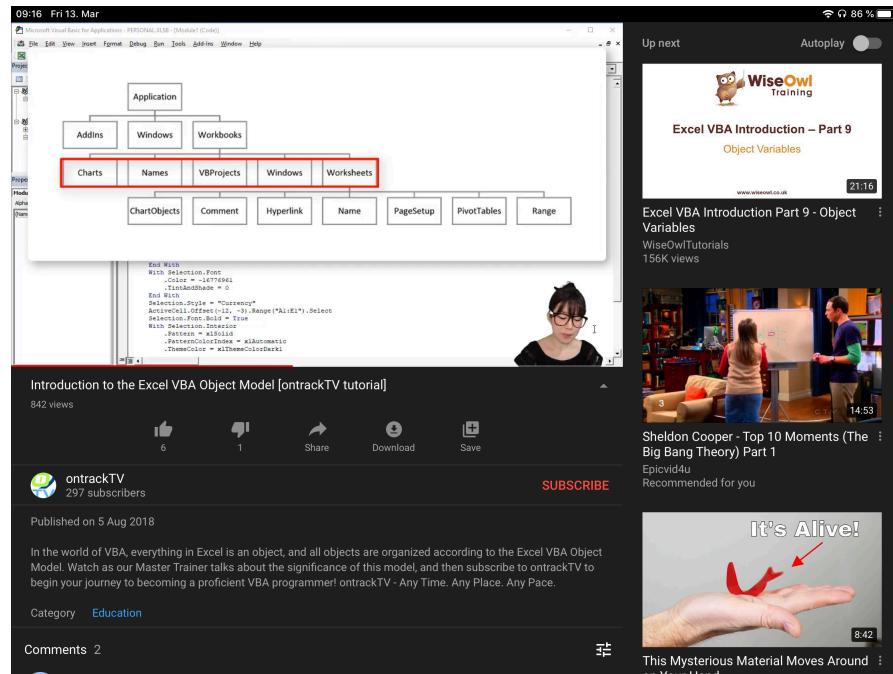


Abbildung 149:

Auf der object model wird vermutlich im späteren Verlauf nochmal eingegangen. Aktuelle wird im Kurse weiter vorangeschritten, um vielleicht das fehlende Verständnis über das Model aufzuarbeiten.

### 2.2.3 VBA Editor

Der Editor bietet über **F2** die Möglichkeit an, die Hilfefunktion abzurufen.



Abbildung 150:

Eine der einfachsten *Expressions* ist die Message Box. Die Funktion dazu lautet

---

```

1 Sub MessageBox ()
2   MsgBox "Hello Paul."
3 End Sub

```

---

### 2.2.4 Processor: Subroutine and function

In diesem Kontext ist es sinnvoll darüber zu sprechen, dass VBA zwischen Objekt-Module und Module unterscheidet. Subroutinen können Werte übertragen bekommen.

---

```

1 Sub DatabaseName ()
2   Dim sDbName As String ' Variablennamen können mit einem Kleinbuchstaben beginnen.
3   sDbName = "Paul"
4 End Sub

```

---

```
5
6 Sub ShowDatabaseName (sDbName As String)
7     MsgBox sDbName
8 End Sub
```

---

## 2.2.5 Processor: Function

Der Unterschied zur *Sub* ist, dass eine *function* einen Wert zurückgeben kann. Ein *Sub* nicht. The *return* ist called by the function name. Das bedeutet am Ende der Prozedur wird ein eine Ausgabe definiert. Diese muss mit den Namen der Funktion betitelt werden.

```
1 Function MultiplikationPy (dZahl As double)
2     MultiplikationPy = dZahl*3.14
3 End function
4
5 Sub Show ()
6     ' MsgBox MultiplikationPy(5); Dies wird nicht funktionieren, da die Funktion ein double
7     ' Wert benötigt.
8     Dim dZahl As double
9     dZahl = 5
10    MsgBox MulitplikationPy(dZahl)
11 End Sub
```

---

## 2.2.6 Set Statement

The *Set* statement is like a pointe in *C++*. VBA hat eine Objekt-Hierarchy aufgebaut. Dabei können innerhalb dieser Hierarchy Objekte angesteuert werden - es wird auf sie verweisen. Mit *Set* kann daraufhin innerhalb dieser Struktur Verweise geschaffen werden. Diese werden dann genutzt um Methoden abzurufen, auf weitere Objekte zu verweisen oder Werte auszulesen. Damit ist *Set* zu unterscheiden von dem einfachen Datentypen, wie *string*, *bool*. Die Verweise können zum einen mit einem einfachen *Object* Typ deklariert werden, zum anderen können aber auch spezifischer Typen angegeben werden.

```
1 Dim wbObjekt As Workbooks ' Variable wird initialisiert
2 Set Application.Workbooks("Name.xlsxm")
```

---

Wir eine neue Instant eines Objektes erzeugt, wird *New* verwendet. Bei der Deklarierung, *Set*, wird noch kein Objekt erzeugt, erst mit *Set*

```
1 Dim myChildForms(1 to 4) As Form1
2 Set myChildForms(1) = New Form1
3 Set myChildForms(2) = New Form1
4 Set myChildForms(3) = New Form1
5 Set myChildForms(4) = New Form1
```

---

Es können natürlich mehrere Referenzen zu einem Objekt geschaffen werden.

```
1 Dim wbObjekt As Workbooks
2 Set wb = Application.Workbooks("Name.xlsxm")
3 Set wb2 = Application.Workbooks("Name.xlsxm")
```

---

## 2.2.7 First Code-Marco

Wir ein Command Button erstellt, kann dieser über die Funktion *Event Builder* angesteuert werden.

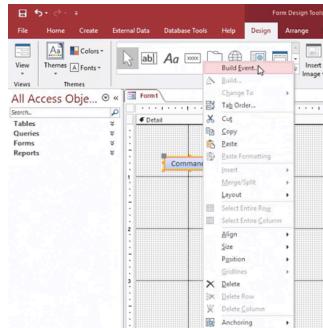


Abbildung 151:

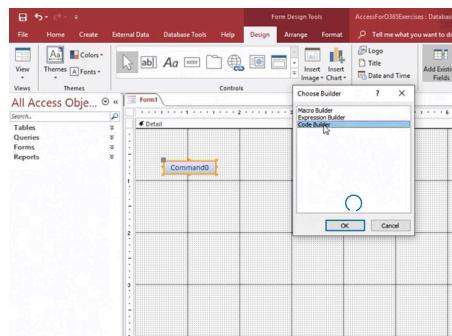


Abbildung 152:

Es wird ein Makro (Code) erstellt.

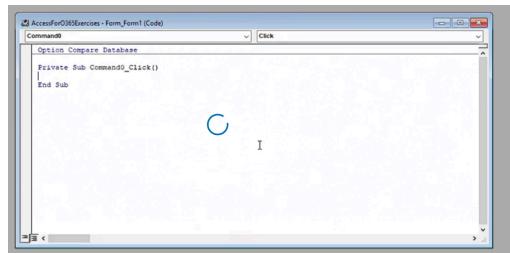


Abbildung 153:

Die Option *Option Compare Database* hat die Funktion, dass die folgenden Prozeduren auf Groß- und Klein-schreibung bei einem String-Vergleich nicht berücksichtigt wird. Ebenso wird das Modul unter dem Formblatt abgespeichert.

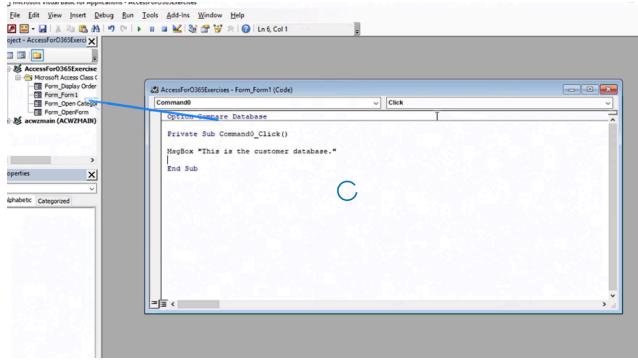


Abbildung 154:

So weit ich es verstanden habe, ist ein Form Objekt ein angesteuertes Object welches sich in Forms Objekt befindet. Methoden die für Form vorgesehen werden, können erst durch ein genaues ansteuerten eines einzelnen Form ausgeführt werden.

---

```

1 Sub AddressObject()
2     Dim myForm As Form
3     Set myForm = Forms("Product") ' Use of a different object
4     MsgBox myForm.Name
5     Set myForm = Nothing ' Releases the connection
6 End Sub

```

---

## 2.3 Variables, Constants, Calculation

Die Option *Option Explicit* zwingt, dass alle Variablen deklariert werden müssen. Variablen die außerhalb von Prozeduren stehen, werden als globale Variablen definiert und sind für das gesamte Modul verfügbar.

---

```

1 Dim iNumber As Integer = 20 ' Global
2 Const iNumber2 As Integer = 30 ' Global und behält den konstanten Wert
3 Static iNumber3 As Integer = 40 ' Global und behält den Wert über mehrere Ausführungen
4 Sub Main()
5
6 End Sub

```

---

## 2.4 Add Logic to your VBA Code

### 2.4.1 Iterator-Prozeduren

- For ... to ... Next: Integer Loop-Schleife
- For each ... in ... Next: Iteriert durch eine Array durch.

---

```

1 Sub ForEach()
2     Dim varItem As Variant
3     Dim varArray As Variant
4
5     varArray = Array("Eins", "Zwei", "Drei")
6
7     For each varItem in varArray
8         MsgBox (varItem)
9     Next
10
11 End Sub

```

---

Für das Form-Collection kann ebenfalls so angesteuert werden.

---

```

1 Sub ForEachForm()
2     Dim myF As Form
3

```

```

4  For Each myF In Forms
5    MsgBox (myF.Name)
6  Next
7
8 End Sub

```

---

- Do until ... Loop:

- Do while ... Loop:

Es gibt verschiedene Syntaxen/ Methoden um auf Spalten in einem Recordset zu verweisen.

- Itemmethode: myR!["Nachname"]
- Fields: myR.Fields("Nachname")

Der Verweis start immer am Anfang eines Tabelle. Der Loop muss jedoch separat gesteuert werden. In M wird die über die Funktion gesteuert und per each wird jede Zeile umgewandelt. Wir die *Recordset.AddNew()* Methode gestartet, zeigt der Zeiger am Ende der Tabelle.

```

1 Sub Ausgabe_Print()
2   Dim recTest As Recordset
3   Dim i As Long
4   Set recTest = Currentdb.OpenRecordset("Tabelle")
5
6   Do while i < recTest.RecordCount ' Oder recTest.EOF (Endoffile)
7     Debug.Print recTest.Fields("Nachname") ' Alle Daten in der Spalte Nachname werden
      ausgegeben.
8   i = i + 1
9   recTest.Move
10  End Sub

```

---

- If else; elseif
- Case Statement

## 2.5 Debug Code

- Siehe unter Fehlerbehandlung/VBA-Tutorial.
- Breakpoint. Klickt man am linken Rand im Editor wird ein roter Punkt gesetzt. Dieser stellt ein Breakpoint da.

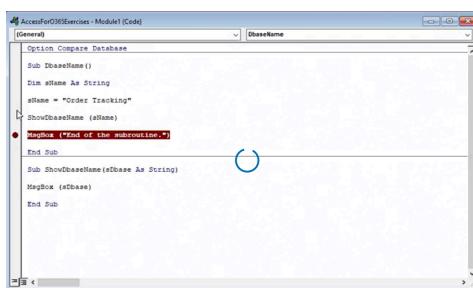


Abbildung 155:

- Add Watch

## 2.6 Manipulate Database Object using DoCmd Object

### 2.6.1 Open Objects

**DoCmd OpenForm** Diese Funktion bietet viele Option die Form in spezieller Weise zu öffnen: DoCmd.OpenForm

**DoCmd OpenRecord** Ein Record wird gleich an den Drucker gesendet. Um nur ein Report sich anzeigen zu lassen, muss die Funktion abgeändert werden.

**DoCmd OpenTable** Mit DoCmd.OpenTable "Name" wird die dazugehörige Tabelle geöffnet.

Die verschiedenen Ansichtsmodi helfen das Objekt im richtigen Modus zu öffnen.

### 2.6.2 Close Objects

Um ein Objekt zu schließen, kann

---

```
1 DoCmd.Close acForm "Name"
```

---

Unter **ac** wird das Objekt spezifiziert und über den Namen direkt angesteuert.

### 2.6.3 RunCommand Export

Unter DoCmd.RunCommand befinden sich die Menüoptionen. Diese können über diese Methode aufrufen. Am Beispiel Daten zu exportieren, wird der Befehl acCmdExportExcel ausgewählt.

---

```
1 DoCmd.RunCommand (acCmdExportExcel)
```

---

## 2.7 Read and manipulate Table Data

### 2.7.1 Add New

Es ist wichtig, dass nach dem die Methode AddNew() ausgewählt und die Eingaben getätigten wurden, die Update() Methode aufgerufen wird, um die Eingaben zu speichern.

### 2.7.2 Edit

Die Methode Edit() erlaubt einen spezifischen Datensatz zu bearbeiten. Es ist dabei wichtig vorher die entsprechende Zeile zu finden. Eine Methode, die spezifische Datensätze sucht, ist FindFirst(). Ebenfalls wird UpDate() benötigt.

### 2.7.3 Data Perservation

Um den Nutzer einzubinden, ob die geänderten Einträge auch wirklich verwendet werden sollen, wird gern die Transaktions-Methoden verwendet.

- BeginTrans()
- CommitTrans()
- RollbackTrans()

Diese Methoden können für verschiedenen Objekte angewandt werden. Es bietet sich an, .Workspace(0) zu verwenden. Ebenso, kann der Nutzer aufmerksam gemacht werden, ob er die Änderungen anwenden möchte.

---

```
1 Sub Preserv()
2     Dim worVar As Workspace
3     Dim recVar As Recordset
4     Set worVar = DAO.DBEngine.Workspace(0)
5     Set recVar = DAO.Currentdb.OpenRecordset("Name")
6
7     worVar.BeginTrans()
8
9     Do Until recVar.EOF
10
11     'Stuff
12
13     Loop
14
15     If MsgBox ("Do you want the Change?", vbQuestion + vbYesNo) = vbYes Then
```

```

16     worVar.CommitTrans()
17 Else
18     worVar.RollbackTrans()
19
20 End Preserv

```

---

## 2.7.4 TableDef

Diese Objekt ist unter Currentdb zufinden. Mit Hilfe dieses Objektes können unter anderem Eigenschaften eines Feldes abgerufen werden.

## 2.8 Manipulate a Database using the Application Object

### 2.8.1 Function to Summarize

- DAvg() - Nimmt den Average über eine Spalte hinweg: DAvg("Quanität")
- DMax(), DCount(), DMin(), etc.
- DFirst(), DLast() - Bei diesen beiden Funktionen wird die Tabelle als String verwendet: DFirst(SSpaltenname", "Tabellennamen")

### 2.8.2 DLookup()

Die DLookUp() Funktion gibt die Werte aus einer anderen Tabelle oder Querie wieder.

```

1 DLookUp(
2   "[Spalte_Wiedergabe_Sekundär]", //-- Der Wert der gesucht wird, aus einer meist anderen
   Tabelle/ Query.
3   "Tabelle_Sekundär", //-- Andere Tabelle/ Query
4   "[Spalte_Suche_Sekundär] =" & [Spalte_Suche_Primär]) //-- Vergleiche Spalten aus der zu
   suchenden Tabelle und aus der erstellten Tabelle.

```

---

## 2.9 Control Forms and Reports

### 2.9.1 Sinnvolle Einschränkungen

- Damit Nutzer nicht Daten einfach ändern, löschen oder hinzufügen, kann dies direkt über Eigenschaften der Forms gesteuert werden. Gesteuert werden die Eigenschaften am Besten, wenn das Objekt geladen wird.

```

1 Sub Form_Load()
2   Me.AllowAdditions = False
3   Me.AllowEdits = False
4   Me.AllowDeletions = False
5 End Sub

```

---

- Ebenso kann die Filterfunktion deaktiviert werden. Dies können auch in Formular angewandt werden.

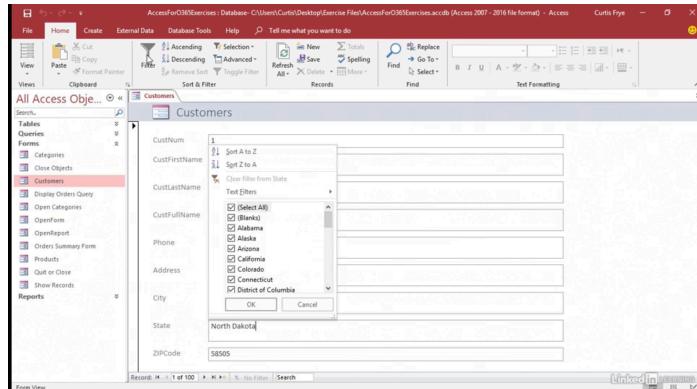


Abbildung 156:

Um diese Funktion nicht mehr zu erlauben.

---

```

1 Sub Form_Load()
2 Me.AllowFilter = False
3 End Sub

```

---

Wir das Formular neu geladen, so ist die Filter-Funktion ausgeblendet.

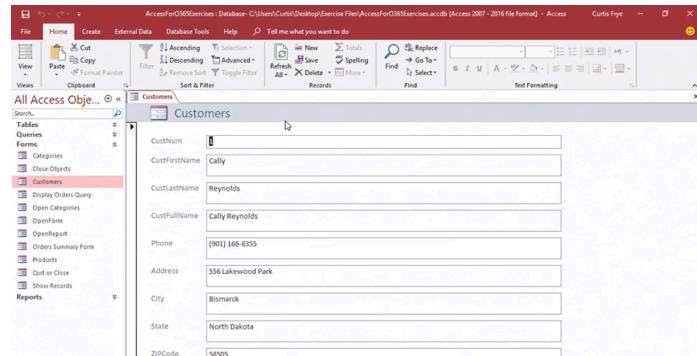


Abbildung 157:

## 2.9.2 Verfeinerung

**Dynamische Beschriftung** Der Hintergrund, sowie die Überschrift eines Formulars und Report können direkt über das `Me.` Objekt angesteuert werden.

**Aktualisierung des Formulars** Wird direkt ein Wert in der Tabelle hinterlegt, so ändert dies nicht automatisch die abgefragten Daten im dazugehörigen Formular. Mit `Me.Requery` wird unter `Form_Activate()` das Formular eine neue Abfrage starten, sobald es das aktive Fenster wird.

## **Teil IV**

### **SQL**

# 1 SQL - Conceptual

On: Youtube

## 1.1 Keys

### 1.1.1 Primäry Key

Ein Primärer Schlüssel ist eine Spalte, welche jede Zeile in einer Tabelle eindeutig definiert. Dieser Schlüssel unterliegt dabei zwei Eigenschaften

- Eindeutigkeit (Unique)
- Nicht Null (No Null)
- Nicht änderbar (No changing)

Es wird dafür meist eine Identitätsspalte eingefügt, welche eine Indexierung vornimmt. Diese muss aber nicht zwangsläufig der "Primäre Key" sein.

### 1.1.2 Composite Key

Eine Tabelle, in welcher keine Spalte für sich eine eindeutige Identifikationsmöglichkeit darstellt, benötigt ein Composite Key. Dieser beinhaltet mindestens eine zwei Spaltenreferenzen. Diese Kombination aus Spalten ergibt eine Eindeutigkeit einer jeden Zeile.

Composite Key	
ClassStudents	
ClassID	StudentID
7	3084
7	3072
203	3081

Abbildung 158:

In dem Beispiel ist erste jede Zeile identifizierbar, wenn ein Schlüssel aus beiden Spalten gebildet wird.

### 1.1.3 Forgein Key

Eine Tabelle, welche Werte aus einer anderen Tabelle abfragt, benötigte eine Referenz zu dieser benötigten Tabelle. Die Tabelle, aus der Werte benötigt werden, fällt unter der Kategorie **Eltern**. Die Tabelle, welche Werte aus der Eltern Tabelle benötigt, fällt unter der Kategorie **Kind**.

In dem Fall, dass eine Kind Tabelle eindeutige Zeile in der Eltern Tabelle referieren muss, wird ein **Forgein Key** benötigt. Dieser muss

- nicht null sein,
- eindeutig die Eltern Tabelle referieren.

Es ist jedoch möglich eine einen null Eintrag in der Forgein Key Spalte zu besitzen, welche nicht zur Spalte der Eltern Tabelle referiert. Daraus folgt, dass der Forgein Key nicht eindeutig sein muss und null sein kann. Diese ist ein Adopiv-Kind, welches nicht zu einer zu einer gewissen Zeile in der Primärschlüssel Spalte der Elterntabelle referiert.

#### 1.1.4 Composite Key

Wenn eine Tabelle zugrunde liegt, in welcher keine Spalte für sich als Primär Key fungiert, so wird eine Primär Key oder auch Composite Key aus mehreren Spalten gebildet.

In der Tabelle mit Studenten-ID und Class-ID, gibt es keinen Primären Schlüssel. Die Spalten müssen verbunden werden, sodass ein eindeutiger Schlüssel für die Tabelle geschaffen wird.

ClassStudents	
classID	studentID
7	3084
7	3072

Abbildung 159:

Es kann nämlich sein, dass in der Spalte mehrere gleiche Werte stehen.

Composite Key	
ClassStudents	
classID	studentID
7	3084
7	3072
202	3084

Abbildung 160:

Ein Composite Key ist somit ein Primäry Key, welcher sich über mehrer Spalten erstreckt.

#### 1.1.5 Compound Key

Ein Compound Key ist fast gleich zum Composite Key mit dem Unterschied, dass die verbundenen Spalten selber wieder Foreign Key sind, und somit zu anderen Tabellen verweisen.



Abbildung 161:

## 1.2 Normalization

### 1.2.1 1NF - Atomicity

In diesem Kontext wir von Entities und Attributes gesprochen. Dabei werden Entities in Tabellen umgesetzt und Attributes als Spalten definiert. Die Atomizierung beschreibt, dass die Zielsetzung einer Tabelle sind in der Wahl der Spalten niederschlagen soll. Das bedeutet, Werte in der Spalte mit dem Hauptbezug keine mehreren Werte besitzen soll. In anderen Spalten sind wiederkehrende Werte akzeptable. In anderen Hauptbezug-Spalten/Spalten ist dies nicht akzeptable. Der Primärkey ist damit jedoch nicht zwangsläufig gemeint.

Wenn einem Instanz einer Tabelle mehrere Werte zugeordnet werden müssen, dann sollte dies in einer ergänzenden Tabelle geschehen, welche den Hauptzweck besitzt, das Attribute abzudecken. Jedoch gilt auch hier, dass Eindeutigkeit benötigt wird.

Abbildung 162:

Am Beispiel zeigt sich, dass Interests verwendet wird, Interessen für verschiedene Tiere zu speichern. Dabei kann ein Tier mehrere Interessen haben, jedoch in jeder Zeile steht nur ein Tier mit einem eindeutigen Interesse.

Eine Tabelle befindet sich in 1NF,

- wenn ein oder mehrere Attribute kein Duplikat eines anderen Attributes ist.
- wenn Instanzen eines Attributes keine Gruppierung des zugehörigen Attributes ist.

Das bedeutet, dass mehrere Ausprägungen eines Attributes nicht in einer Tabelle stehen sollen. Ebenso sollten auch diese auch nicht in einer Zelle gruppiert stehen. Diese sollen eindeutige Einträge in eine weitere Tabelle übertragen werden.

Abbildung 163:

Die Lösung zu dem Problem kann sein, dass eine Junction Tabelle erstellt wird.

Abbildung 164:

Eine Junction Tabelle wird nicht benötigt, wenn die ausgedehnte Tabelle sich nicht vollständig auflisten lässt.

### 1.2.2 2NF - Partial Dependency

Die zweite Normalisierung setzt die erste voraus. Als weiteren Punkt müssen alle Non-Key Attributes sich auf den Primären Schlüssel beziehen. Bei einem Primären Key, welcher aus einer Spalte besteht, ist 2NF gleich 1NF.

In einer Tabelle, welche einen Compound Key besitzen, müssen sich alle Non-Attribute auf alle Bestandteile des Primären Key beziehen. Im folgenden Beispiel ist dies nicht gegeben.

TABLE_PURCHASE_DETAIL		
Customer ID	Store ID	Purchase Location
1	1	Los Angeles
1	3	San Francisco
2	1	Los Angeles
3	2	New York
4	3	San Francisco

Abbildung 165:

Mit Hilfe der 2NF wir entweder die Tabelle getrennt oder die Daten nicht extra mit einbezogen.

TABLE_PURCHASE		TABLE_STORE	
Customer ID	Store ID	Store ID	Purchase Location
1	1	1	Los Angeles
1	3	2	New York
2	1	3	San Francisco
3	2		
4	3		

Abbildung 166:

### 1.2.3 3NF - Transitive Dependency

Die dritte Normalisierung setzt sich aus den beiden vorherigen zusammen. Aufbauend auf das vorherige folgt, dass ein Non-Key Attribute sich nicht auf ein anderes Non-Key Attribute beziehen soll.

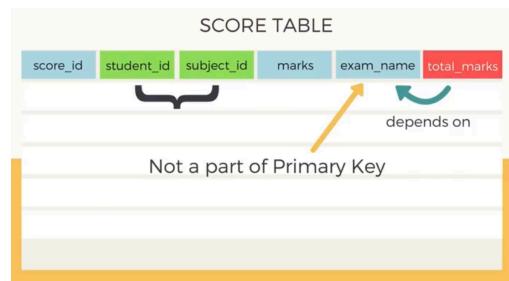


Abbildung 167:

Die gesamt Anzahl der Punkte ist Abhängig von der Art des Test. Diese Information sollte ausgeliedert werden.



Abbildung 168:

## 1.3 Junction

### 1.3.1 One-to-One Relationship

Trival.

### 1.3.2 One-to-Many Relationship

Trival.

### 1.3.3 Many-to-Many Relationship

Um eine Beziehung zwischen zwei Primärkeys herzustellen, welche in einer Many-to-Many Beziehung stehen, benötigt es eine intermediären Tabelle. In dieser besteht in keiner der Foreign Key Spalten eine Eindeutigkeit. Am Beispiel der Beziehung zwischen Autorinnen und Büchern lässt sich zeigen, dass es eines zwei One-to-Many Beziehungen benötigt, um eine Many-to-Many Beziehung abzubauen.

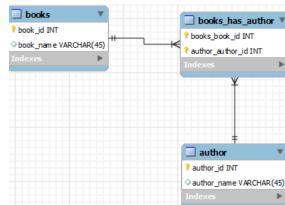


Abbildung 169:

Die Autorinnen können mehrere Bücher verfasst werden. Die Bücher können mehrere Autorinnen haben. In der Junction Tabelle wird die Permutation dargestellt, ohne dabei eine Tabelle zu erzeugen, in welcher mehrere Attribute von einem abgeleitet werden.

## 1.4 Data Integrity

Daten Integrität befasst sich mit Frage, wie können Datenbanken eine gewisse Stabilität und Verlässlichkeit gewähren. Dabei werden die vorherigen beschriebenen Themen verwendet. Daten Integrität lässt sich in drei Bereiche aufteilen.

- Entry Integrity
- Reference Integrity
- Domain Integrity

Die erste beiden nutzen als Grundlage die Primär und Foreign Keys.

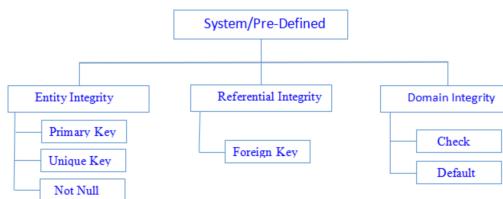


Abbildung 170:

Domain Integrity beschäftigt sich damit, das Einträge in die Tabellen die richtigen Formate und Rahmenbedingungen (Constraints) erfüllen.

## 2 SQL - Introduction

On: LinkedInLearning Eine Relational Database ist eine DB, in welcher Informationen in zweidimensionale Tabellen gespeichert werden.

## 2.1 SELECT Statement

Soll eine komplette Tabelle bezogen werden, so wird das Sternchen \* verwendet. In der Beispieldatenbank befindet sich die Tabelle Countries. Sollen alle Spalten und Wert der Tabelle bezogen werden, wird \* verwendet, um alle Spalten der Tabelle auszuwählen:

```
1 SELECT *
2 From Customers;
```

SQL Statement:

```
SELECT *
From Customers;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 91

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK

Abbildung 171:

Ein Teilmenge der Spalten wird über die genaue Adressierung angesteuert. Die Interne Ordnung legt dabei die Reihenfolge der Spalten in der übergebenen Tabelle fest.

```
1 SELECT CustomerID, Address
2 From Customers;
```

Result:

Number of Records: 91

CustomerID	Address
1	Obere Str. 57
2	Avda. de la Constitución 2222
3	Mataderos 2312
4	120 Hanover Sq.
5	Berguvvägen 8
6	Forsterstr. 57
7	24, place Kléber

Abbildung 172:

Wie unten beschrieben, können die Bezeichnungen der Spalten durch AS geändert werden.

```
1 SELECT CustomerID AS Index, Address
2 From Customers;
```

Result:

Number of Records: 91

ID	'Adresse'
1	Obere Str. 57
2	Avda. de la Constitución 2222
3	Mataderos 2312
4	120 Hanover Sq.
5	Berguvsvägen 8
6	Forsterstr. 57
7	24, place Kléber

Abbildung 173:

Für die Dauer der Abfrage kann der Spaltennamen und oder der Tabellennamen, der generierten Tabelle, mit **AS** geändert werden.

```
1 SELECT CustomerID AS ID, Address AS 'Adresse'  
2 From Customers;
```

Result:

Number of Records: 91

ID	'Adresse'
1	Obere Str. 57
2	Avda. de la Constitución 2222
3	Mataderos 2312
4	120 Hanover Sq.
5	Berguvsvägen 8
6	Forsterstr. 57
7	24, place Kléber

Abbildung 174:

Mit Apostrophen können auch Schlüsselwörter als Spaltennamen verwendet werden.

### 2.1.1 FROM - Input table

Mit **FROM** wir die Quelle der SQL Anfrage bestimmt. Dies kann eine einfache Tabelle sein oder auch Tabelle, welche **Joins** zu anderen Tabellen besitzt.

### 2.1.2 WHERE - Where row is evaluated

Die Clause **WHERE** ist besser im Englischen zu verstehen. Select the following rows **where ... are equal to**

....

```
1 SELECT *  
2 FROM Customers  
3 WHERE City = 'Berlin';
```

Number of Records: 1

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany

Abbildung 175:

Es kann auch nach einzelnen Begriffen gesucht werden. Mit **Like** kann nach einzelnen Begriffen gesucht werden.

- Mit % können Begriffe einzeln betrachtet werden.
- Mit '%...%' wird der Begriff gesucht, egal was davor oder danach steht.
- Mit '%...' wird der Begriff gesucht, egal was davor steht.
- Mit '...%' wird der Begriff gesucht, egal was danach steht.

Ebenso können einzelnen Buchen an gewissen Stellen gesucht werden. Dafür nutzt man \_.

---

```

1 SELECT Country
2 FROM Customers
3 WHERE Name Like '%Tim%' // Anfang oder Ende ist egal.

```

---

```

1 SELECT Country
2 FROM Customers
3 WHERE Name Like '_i%' // Zweites Zeichen ist i

```

---

Um mehrere Bedingungen festzulegen, wird der `IN()` Operator festgelegt.

---

```

1 SELECT Country
2 FROM Customers
3 WHERE Name IN('West', 'Ost')

```

---

### 2.1.3 GROUP BY - zusammenfassen von Spalten

Im einfachen Fall wird diese Clause genutzt um eindeutige Werte einer Spalte auszugeben.

---

```

1 SELECT Country
2 FROM Customers
3 GROUP BY Country

```

---

Result:	
Number of Records: 21	
Country	
Argentina	
Austria	
Belgium	
Brazil	
Canada	
Denmark	
Finland	
France	

Abbildung 176:

### 2.1.4 ORDER BY - sort table

Zum Ordnen einer Tabelle wird `Order by` verwendet. Danach wird der Spaltennamen eingetragen, nach dem absteigend geordnet werden soll.

---

```

1 SELECT *
2 From Country
3 Order by Name;

```

---

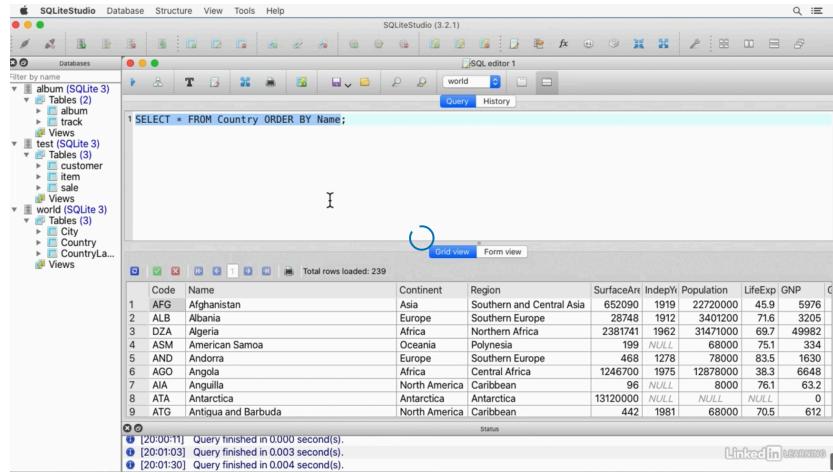


Abbildung 177:

Anstatt \* können auch einzelne Spaltennamen adressiert werden.

---

```

1 SELECT      Name , Code
2 From       Country
3 Order by   Name;
```

---

Es wird bei nicht wie bei M Formula Language [...] benötigt, um Spalten zu adressieren.

Wie schon erwähnt, soll die Spaltenbezeichnung geändert werden, kann dies für jede einzelne Spalte getan werden.

---

```

1 SELECT      Name AS "Name Neu" , Code AS "Country_Code"
2 From       Country
3 Order by   Name_Neu;
```

---

In diesem Fall verhält es sich ähnlich wie in M Formula Language. Für einen String als Spaltennamen werden Anführungszeichen verwendet.

---

```

1 SELECT      Name AS Name2 , Code
2 From       Country
3 Order by   Name2;
```

---

Mit Hilfe einer Aggregatfunktion können Werte für die gruppierte Spalte ausgegeben werden.

---

```

1 SELECT Count(Quantity) , ProductID
2 FROM OrderDetails
3 Group by ProductID;
```

---

Number of Records: 77	
ProductID	Expr1001
1	8
2	11
3	2
4	5
5	4
6	2
7	2
8	2

Abbildung 178:

Die Anzahl der Aggregationen ist nicht beschränkt.

---

```
1 SELECT SUM(Quantity), Count(ProductID)
2 FROM OrderDetails
3 GROUP BY ProductID;
```

---

Zu Vermerken ist, dass nur die Spalten generiert werden, welche auch durch `SELECT` ausgewählt werden.

---

```
1 SELECT ProductID, SUM(Quantity) AS SUM_Quantity, Count(ProductID) AS Count_Qunatity
2 FROM OrderDetails
3 GROUP BY ProductID;
```

---

Number of Records: 77		
ProductID	SUM_Quantity	Count_Qunatity
1	159	8
2	341	11
3	80	2
4	107	5
5	129	4
6	36	2
7	25	2
8	140	2

Abbildung 179:

### 2.1.5 LIMIT - Who many Rows

Mit dem Ausdruck `LIMIT` kann die Anzahl der Zeilen reglementiert werden. Dieser Ausdruck kommt nach `Order BY`.

---

```
1 SELECT Name, Contient, Region // column name
2 From Country // table name
3 WHERE Country = 'Germany'
4 ORDER BY Name
5 LIMIT 5
```

---

### 2.1.6 OFFSET - Ausschluss der ersten Zeilen

Die Auswahl der Zeilen kann auch noch der Indexierung der Tabelle erfolgen. Mit dem Begriff `OFFSET` wird eine Auswahl der der letzten  $m - n$  Zeilen getroffen. Wobei  $m$  die Anzahl der gesamten übergeben Tabellenanzahl ist und  $n$  der zu reduzierenden Zeilen ist.

---

```
1 SELECT Name, Contient, Region // column name
2 From Country // table name
3 WHERE Region = 'West'
4 ORDER BY Name
5 OFFSET 5 // Ersten 5 Zeilen werden nicht berücksichtigt.
```

---

## 2.2 Concepts

### 2.2.1 Function Types

Aggregatfunktionen haben die folgende Syntax

---

```
1 SELECT Funktiontype(Column) From Table
```

---

**Count()** Die Funktion `Count()` zählt die Zeilen der generierten Tabellen. Es können dabei zwischen der ganzen Tabelle und einzelnen Spalten unterschieden werden. Wählt man einzelnen Spalten aus, werden nicht leere Zeilen gezählt.

`Count()` angewandt auf die gesamt Tabelle

---

```
1 SELECT Count(*)
2 From Country // table name
```

---

generiert eine Tabelle mit dem Spaltentitel "Count(\*)", in welcher in der ersten Zeile der gesuchte Wert steht.

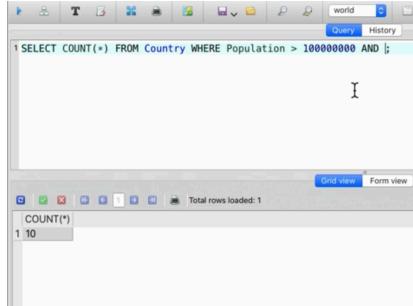


Abbildung 180:

Die Einschränkung auf eine gewisse Spalte, gibt die Anzahl der Zeilen, welche nicht null sind, wieder.

---

```
1 SELECT Count(Region)
2 From Country
```

---

Ebenso können weitere Einschränkungen und Überprüfungen an der Tabelle vorgenommen werden.

---

```
1 SELECT Count(Region), Contient
2 From Country
3 WHERE Contient = 'Europe'
```

---

Manche Anpassungen haben kein Effekt auf das Ergebnis der Funktion.

---

```
1 SELECT Count(*)
2 From Country // table name
3 ORDER BY Name
```

---

Sollen unterschiedliche Einträge verglichen werden, wird **DISTINCT** als Zusatz verwendet, siehe [DISTINCT Clause](#)

**Aggregatfunktionen** Zu den Aggregatfunktion gehört

- AVG()
- MAX(), MIN()
- SUM(),
- Count()

## 2.2.2 DISTINCT Clause

Innerhalb einer Aggregatfunktion werden nur alle ungleichen, nicht leere Zeilen ausgewählt.  
In der Funktion **Count()** angewandt, werden nur Zeilen gezählt, welche nicht gleich sind.

---

```
1 SELECT Count(DISTINCT Quantity)
2 FROM OrderDetails;
```

---

Die Clause kann auch vor einem Spaltennamen in der **SELECT** Clause angewandt werden.

---

```
1 SELECT DISTINCT Quantity
2 FROM OrderDetails;
```

---

Number of Records: 43	
Quantity	
1	
2	
3	
4	
5	

Abbildung 181:

Werden weitere Spalten hinzugefügt, bezieht sich die Eindeutigkeit auf die gesamte Zeilenbreite der Tabelle.

---

```
1 SELECT DISTINCT Quantity, ProductID
2 FROM OrderDetails;
```

---

Number of Records: 433	
Quantity	ProductID
1	19
1	37
1	69
2	13
2	17
2	26
2	40

Abbildung 182:

Im Vergleich:

---

```
1 SELECT Quantity, ProductID
2 FROM OrderDetails
3 ORDER BY Quantity;
```

---

sind 518 Zeilen generiert worden, hingegen nur 433 unter Berücksichtigung der Eindeutigkeit.

Number of Records: 518	
Quantity	ProductID
1	69
1	37
1	19
2	17
2	56

Abbildung 183:

### 2.2.3 CREATE TABLE ... ()

Um eine Tabelle zu erstellen, wird **CREATE Table** verwendet. Die Schema hinter diesem Ausdruck beschreibt die Spaltennamen und Typen. Dabei wird zuerst der Spaltenname genannt und danach der Typ.

---

```
1 CREATE TABLE Tabellenname (
2 Spaltenname_1 TEXT,
3 Spaltenname_2 INTEGER,
4 Spaltenname_3 BOOLEAN
5 );
```

---

Neue Werte können über **INSERT INTO** eingepflegt werden.

---

```
1 INSERT INTO Tabellenname
2 (Spaltenname_1, Spaltenname_2)
3 Value ('text', 4, True);
```

---

## 2.2.4 DROP TABLE ...

```
1 DROP TABLE Test;
```

Wenn die Tabelle nicht existiert, kommt es zu einem Fehler. Dieser kann umgangen werden, indem **IF EXISTS** vor dem Tabellennamen geschrieben wird.

```
1 DROP TABLE IF EXISTS Test;
```

## 2.2.5 INSERT INTO ... ()()

Daten können auch an Tabellen angefügt werden. Die **INSERT INTO** Clause baut sich in drei Teile auf

- Tabllenbezug
- Spaltenbezug
- Werte

```
1 INSERT INTO Customers
2 (Spalten_1, Spalte_2)
3 VALUE ('Test', 2);
```

Wir die Spalte nicht erwähnt, wir der Null-Wert eingesetzt.

```
1 INSERT INTO Customers
2 (Spalte_2)
3 VALUE (2);
```

Werte aus anderen Tabellen können über **SELECT** eingefügt werden. Anstatt **Value** wird das Select Statement verwendet.

```
1 Create table Test
2 (a integer, b text, c text);
3 INSERT INTO Test
4 Select ID, Name, Adress From Customer;
5 Select * From Test;
```

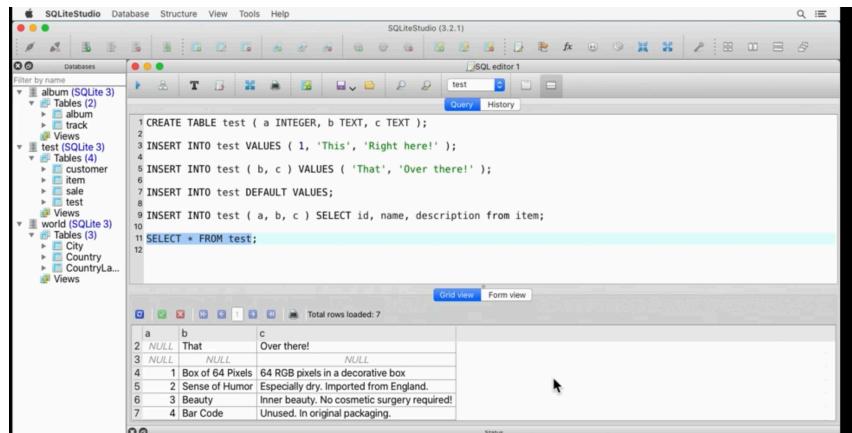


Abbildung 184:

## 2.2.6 UPDATE ... SET ... WHERE

Update wirkt auf die gesamte Tabelle. Daher ist es wichtig, die Bedingung oder Bedingungen festzulegen, welche Zeilen direkt angesprochen werden sollen. Der Aufbaue des Ausdruckes zieht wie folgt aus:

- Tabllenbezug

- **SET** Spaltenbezug mit Änderung
- **WHERE** Bedingung

---

```

1 UPDATE Customers
2 SET Spalte_1 = 'Test', Spalte_3 = 'Test'
3 WHERE Spalte_2 = 'Tesst';

```

---

Wenn keine Bedingung angegeben ist, werden alle Werte in einer spezifischen Spalte geändert.

---

```

1 UPDATE Customers
2 SET Spalte_1 = 'Test';

```

---

## 2.2.7 DELETE FROM ... WHERE

Unterscheidet sich zu **UPDATE** in der Kategorie, dass ganze Zeilen gelöscht werden, nicht nur einzelne Werte. Dies kann über **UPDATE** erfolgen.

---

```

1 DELETE
2 FROM Customers
3 WHERE id = 5;

```

---

In dem obigen Beispiel wird die Ziele mit Index 5 gelöscht.

## 2.2.8 NULL Value

Um Zellen auf den Eintrag **null** zu testen, kann in Structured Query Language (SQL) nicht nach null gesucht werden.

---

```

1 WHERE id = null // Gilt nie. Die Tabelle bleibt leer.

```

---

Damit nach **null** Werte gesucht werden kann, muss folgende Überprüfung erfolgen:

---

```

1 WHERE id IS NULL

```

---

Die Negation lautet **IS NOT NULL**.

Diese Beschreibung wird auch verwendet, wenn Datentypen für Spalten festgelegt werden.

---

```

1 CREATE TABLE test (
2 a Integer NOT NULL,
3 b Text NOT NULL,
4 c Text
5 );
6
7 INSERT INTO test Value
8 (1, 'abs', 'sde');
9
10 INSERT INTO test (b) Values ('asd'); // Ruft einen Fehler hervor, weil a NOT NULL fordert.

```

---

## 2.2.9 DEFAULT

Wie **NOT NULL** kann eine Default Bedingung erstellt werden.

---

```

1 DROP TABLE IF EXIST test;
2 CREATE TABLE test (
3 a Integer DEFAULT 0,
4 b Text NOT NULL,
5 c Text
6 );

```

---

## 2.2.10 UNIQUE

Siehe [DEFAULT](#)

---

```
1 DROP TABLE IF EXIST test;
2 CREATE TABLE test (
3   a Integer UNIQUE,
4   b Text NOT NULL,
5   c Text
6 );
```

---

## 2.2.11 ALTER TABLE ... ADD

Um Spalten zu einer Tabelle hinzuzufügen, wird die [ALTER TABLE ... ADD](#) Funktion verwendet.

---

```
1 CREATE TABLE test (
2   a Integer UNIQUE,
3   b Text NOT NULL,
4   c Text
5 );
6
7 ALTER TABLE test ADD d test;
```

---

Ebenso kann auch hier gleich ein [Default](#) Wert mit [ADD](#) d test DEFAULT 'panda' hinterlegt werden.

## 2.2.12 CASE - Conditional Expression

Um einen Werte in einer Spalte zu überprüfen, kann [CASE](#) verwendet werden. Im einfachen Fall wird eine Überprüfung der Werte einer Spalte durchgeführt. Mit [CASE](#) können mehrere Abfragen gestellt werden. Das Resultat kann in die gleiche Spalte oder eine andere geschrieben werden.

Die Werte, die gespeichert werden sollen, werden unter [End](#) abgespeichert. Der Aufbau der Funktion ist.

---

```
1 CASE (Spalte) WHEN (Bewertung) THEN (...)
```

---

```
2 ELSE (...)
```

---

```
3 END (Speicherort)
```

---

```
1 SELECT department, salary, name
2 CASE department WHEN 50 THEN 1,5*salary
3 WHEN 60 THEN 2,0*salary
4 END 'Finaly_Salary'
5 From Employee;
```

---

```

SELECT first_name, department_id, salary,
       CASE department_id WHEN 50 THEN 1.5*salary
                           WHEN 12 THEN 2.0*salary
                           ELSE salary
       END "REVISED SALARY"
FROM Employee;

```

FIRST_NAME	DEPARTMENT_ID	SALARY	REVISED SALARY
Vipul	50	30000	45000
Amit	12	27000	54000
Satish	12	3500	7000
Harshal	23	7300	7300
Archit	50	2950	4425

5 rows returned in 0.00 seconds [CSV Export](#)

Abbildung 185:

Die Funktion lässt zwei Möglichkeiten zu:

---

```

1 CASE (Spalte) WHEN (Bewertung) THEN (...)

2 ELSE (...)

3 END AS (Neuer Speicherort)

```

---

```

1 CASE WHEN (Spalten bezogene Bewertung) THEN (...)

2 ELSE (...)

3 END AS (Neuer Spaltenname)

```

---

### 2.2.13 Ansteuern von Spalten

In SQLite wird die Syntax [] für Spalten verwandt. Dabei können Spaltennamen auch ohne Eckige Klammern geschrieben werden, wenn der Tabellennamen angegeben wird.

---

```

1 SELECT Substr([a],1,2)
2 From t;
3 -- oder
4 SELECT Substr(a,1,2)
5 From t;

```

---

## 2.3 Joints

Die Relationsfunktion fußen auf den bestehenden Relation der Mengenlehre.

### 2.3.1 Join on or more tables

Die Joins sind wie folgt aufgebaut:

- Select aller Spalten die angezeigt werden sollen. Dabei können die Spalten aus den verbundenen Tabellen angesteuert werden. Ebenso können auch die Spalten der zugrundeliegenden Tabelle angezeigt werden.
  - Jede Tabelle kann einzeln eingesteuert werden *Tabellename*. Die Auswahl der Spalten erfolgt über den Operator ..

- Die Möglichkeit alle Spalten anzusprechen erfolgt ebenso den bekannten Operator \*.

---

```
1   Select customer.name AS "Customer-Name", item.name AS "Item-Name", sale.*
```

---

- Mit **From** wird die zugrundeliegende Tabelle angesteuert.

- Es können jetzt mehrere **Joins** angefügt werden. Der Aufbau eines **Joins** ist, dass an Ersterstelle die angefügten Tabelle steht. Danach wird festgelegt, über welchen Schnittpunkt die Tabellen verbunden werden.

---

```
1   Join customer ON sale.customer_id = customer.id
2   Join item ON sale.item_id = item.id;
```

---

- Weitere Funktionalitäten können angefügt werden

---

```
1   WHERE sale.id = 2;
```

---

Mit Joinsfunktion erlaubt so, dass verschiedene Datensätze verbunden werden.

---

```
1   --Ziel ist es die Namen aus der customer Tabelle zu laden.
2   Select customer.name AS "Customer-Name", item.name AS "Item-Name", sale.*
3   From sale
4   Join customer ON sale.customer_id = customer.id
5   Join item ON sale.item_id = item.id
6   ;
```

---

	Customer-Name	Item-Name	id	item_id	customer_id	date	quantity	price
1	Mary Smith	Box of 64 Pixels	1	1	2	2009-02-27	3	2995
2	Mary Smith	Sense of Humor	2	2	2	2009-02-27	1	1995
3	Bill Smith	Box of 64 Pixels	3	1	1	2009-02-28	1	2995
4	Bob Smith	Bar Code	4	4	3	2009-02-28	2	999
5	Mary Smith	Box of 64 Pixels	5	1	2	2009-02-28	1	2995

Abbildung 186:

### 2.3.2 Kinds of Joins

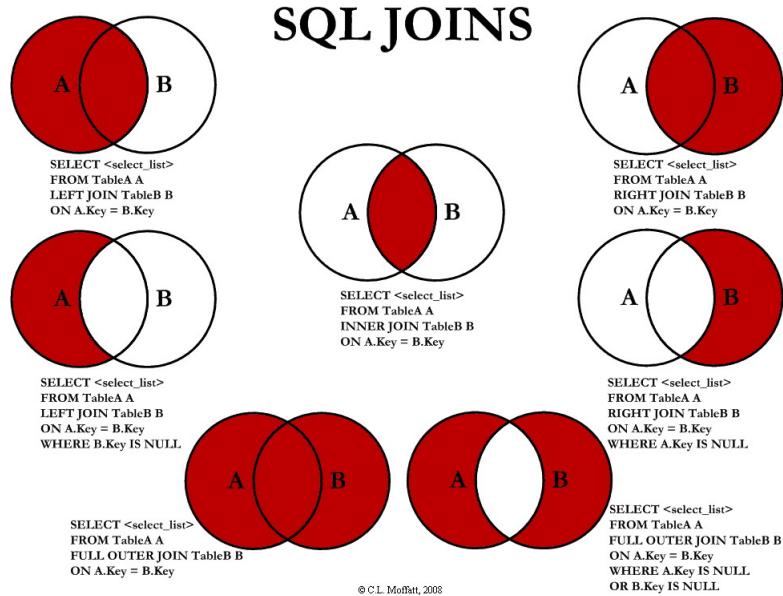


Abbildung 187:

## 2.4 Selection of Function

### 2.4.1 Find the lenght of a string

Die Funktion `LENGTH()` gibt die Länge eines String-Values wieder.

```
1 -- Bestimmen der Länge einer Spalte: Leerzeichen werden auch mitgezählt.
2 Select id, name, Length(name) AS 'Länge der Spalte Name'
3 From customer
4 ;
```

	id	name	Länge der Spalte Name
1	1	Bill Smith	10
2	2	Mary Smith	10
3	3	Bob Smith	9

Abbildung 188:

### 2.4.2 Substring of Field of Date

Datumseinträge können in Teil Einträge zerlegt werden.

```
1 Select date,
2   Substr(date,1,4) AS Year,
3   Substr(date,6,2) AS Month,
4   Substr(date,9,2) AS Day
5 From sale
6 ;
```

	date	Year	Month	Day
1	2009-02-27	2009	02	27
2	2009-02-27	2009	02	27
3	2009-02-28	2009	02	28
4	2009-02-28	2009	02	28
5	2009-02-28	2009	02	28

Abbildung 189:

### 2.4.3 Count()- Group by (aggregate function)

Die `Count()` Funktion auf eine Spalte oder eine Tabelle zählt die nicht leeren Einträge. Angewandt auf eine Tabelle, wir die Anzahl der Zeilen dargestellt. Erst mit der Kombination `Group by` wirkt die Funktion wie ein Funktion, welche ein Parameter übergeben bekommen wird.

**Count(), Group by - same coloum** Für jeden aggregierte Eintrag (grouped), wir die Anzahl ermittelt. Ohne dies zählt die `Count()` Funktion nur die Zeilen einer Tabelle.

```
1 Select state, Count(state)
2 From customer
3 Group by state
4 ;
```

**Count(), Group by - different coloum** Die `Count()` Funktion funktioniert, nicht anders, wenn sie auf eine spezifische Spalte angewandt wird. Die Funktion zählt nur die aggregierten Zeilen, für jede Tabelle.

```
1 Select state, Count(address) -- Wie viel unterschiedliche Adress gibt es in jedem Staat.
2 From customer
3 Group by state
4 ;
```

#### 2.4.4 Mix

**Trim** Die Trim-Funktionen werden weiter unterteilt. Der Kern ist, dass entweder Leerzeichen oder andere Zeichen aus dem Textfeld entfernt werden. Dabei unterscheidet **Trim()** in linksseitige **LTrim** und rechtseitige **RTrim** Funktionen.

**Case Function** Strings können in groß oder klein Darstellung der ASCII-Zeichen umgewandelt werden.

**Typeof()** gibt den Typ des Eintrages wieder.

**Datetime** Die **Datetime()** Funktion nimmt als Input Variablen *Strings*. Das aktuelle Datum wird mit '*now*' beschrieben.

---

```
1 Select Datetime('now','+1 day'),
```

---

## 2.5 Transaction

Zugriffe auf die Datenbank können in Packete geschnürt werden. Diese Transaktionen Starten und Enden mit einem Befehl. Dieser kann sich von SQL Datenbanksystem zu Datenbanksystem anders sein. Transaktionspackete sichern

- Daten Integrität
- Performance

In SQLite gilt

---

```
1 Begin Transaction;
2 ...
3 End Transaction;
```

---

Geht etwas schief, oder werden bestimmte Kriterien nicht erfüllt, so kann mit **Rollback** die bisherige Transaktion rückgängig gemacht werden.

---

```
1 Begin Transaction;
2 ...
3 Rollback;
4 ...
5 End Transaction;
```

---

Als Beispiel dient die folgende Übung:

---

```
1 -- 02 transactions
2 -- test.db
3
4 CREATE TABLE widgetInventory (
5   id INTEGER PRIMARY KEY,
6   description TEXT,
7   onhand INTEGER NOT NULL
8 );
9
10 CREATE TABLE widgetSales (
11   id INTEGER PRIMARY KEY,
12   inv_id INTEGER,
13   quan INTEGER,
14   price INTEGER
15 );
16
17 INSERT INTO widgetInventory ( description, onhand ) VALUES ( 'rock', 25 );
18 INSERT INTO widgetInventory ( description, onhand ) VALUES ( 'paper', 25 );
19 INSERT INTO widgetInventory ( description, onhand ) VALUES ( 'scissors', 25 );
20
21 SELECT * FROM widgetInventory;
22 SELECT * FROM widgetSales;
23
24 BEGIN TRANSACTION;
25 INSERT INTO widgetSales ( inv_id, quan, price ) VALUES ( 1, 5, 500 );
26 UPDATE widgetInventory SET onhand = ( onhand - 5 ) WHERE id = 1;
```

```

27 END TRANSACTION;
28
29 BEGIN TRANSACTION;
30 INSERT INTO widgetInventory ( description, onhand ) VALUES ( 'toy', 25 );
31 ROLLBACK;
32 SELECT * FROM widgetInventory;

```

---

## 2.6 Triggers

Triggers erlauben bestimmt Aktionen durchzuführen, wenn ein Ereignis an einer Tabelle durchgeführt wird. Die verschiedenen Trigger werden in SQLite an die Tabelle angefügt.

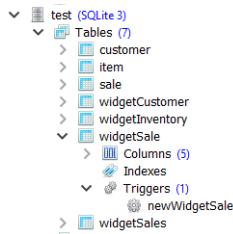


Abbildung 190:

Ein Trigger wird mit dem Namen, den Auslöser und den Bezug initialisiert.

```

1 Create TRIGGER [name] [Event] ON [Tabelle]
2 Begin
3 ...
4 End
5 ;

```

---

Im gewählten Beispiel ist der Auslöser das Einfügen einer neuen Zeile.

```

1 Create TRIGGER NewCustomerID After INSERT ON widgetSales
2 Begin
3     UPDATE widgetCustomer -- Die zu modifizierte Tabelle
4     SET last_order_id = NEW.id -- New. Ist das Objekt, welche generiert wird, wenn eine
        neue Zeile in widgetSales eingelegt wird.
5     WHERE widgetCustomer.id = NEW.customer_id --
6     ;
7 End
8 ;

```

---

Die Trigger Funktion kann verwendet werden, für das Ergänzen von Daten. Diese können Beispielsweise nach dem Einfügen einer Zeile ausgelöst werden.

```

1 -- 01 update triggers
2 -- test.db
3 Drop Table if exists widgetSale;
4 Drop Table if exists widgetCustomer;
5
6 CREATE TABLE widgetCustomer ( id INTEGER PRIMARY KEY, name TEXT, last_order_id INT, stamp
    TEXT);
7 CREATE TABLE widgetSale ( id INTEGER PRIMARY KEY, item_id INT, customer_id INT, quan INT,
    price INT, stamp TEXT);
8
9 INSERT INTO widgetCustomer (name) VALUES ('Bob');
10 INSERT INTO widgetCustomer (name) VALUES ('Sally');
11 INSERT INTO widgetCustomer (name) VALUES ('Fred');
12
13 SELECT * FROM widgetCustomer;
14 Select * From widgetSale;
15
16 CREATE TRIGGER stampSale After Insert ON widgetSale
17 BEGIN
18     UPDATE widgetSale SET stamp = Datetime('now') WHERE id = NEW.id;
19     UPDATE widgetCustomer Set last_order_id = New.id, stamp = DateTIme('now')

```

```

20 Where widgetCustomer.id = New.customer_id;
21 END
22 ;
23
24 INSERT INTO widgetSale (item_id, customer_id, quan, price) VALUES (1, 3, 5, 1995);
25 INSERT INTO widgetSale (item_id, customer_id, quan, price) VALUES (2, 2, 3, 1495);
26 INSERT INTO widgetSale (item_id, customer_id, quan, price) VALUES (3, 1, 1, 2995);
27 SELECT * FROM widgetSale;
28 SELECT * FROM widgetCustomer;

```

---

Was zu beachten ist, dass der **Trigger** selbst ausgeführt werden muss. Dabei wird dieser in unter

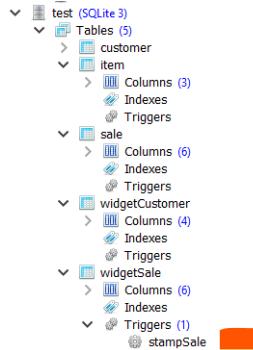


Abbildung 191:

abgespeichert. Das Objekt **NEW** steuert den Record der letzten Zeile der vorgegebenen Tabelle, in dem Fall *widgetSale*. Die Spalten werden dann als Attribute des Objektes **New** gesehen.

## 2.7 Subselects

Abfragen können ebenfalls als temporäre Tabelle betrachtet werden.

```

1 CREATE TABLE t ( a TEXT, b TEXT );
2 INSERT INTO t VALUES ( 'NY0123', 'US4567' );
3 INSERT INTO t VALUES ( 'AZ9437', 'GB1234' );
4 INSERT INTO t VALUES ( 'CA1279', 'FR5678' );
5 SELECT * FROM t;
6
7 -- Trennen der Strings
8 SELECT SUBSTR([a],1,2) AS State,
9     SUBSTR([b],1,2) AS Country,
10    SUBSTR([a],3) As State_Code,
11    SUBSTR([b],3) AS Country_Code
12   From t;
13
14 Select * From Country;
15
16 -- Suche Country-Name mit SELECT als Tabelle
17 SELECT Country.Name, st.Country_Code
18 From (SELECT SUBSTR([a],1,2) AS State,
19     SUBSTR([b],1,2) AS Country,
20     SUBSTR([a], 3) As State_Code,
21     SUBSTR([a], 3) AS Country_Code
22   From t) AS st
23 Join Country
24 On Country.Code2 = st.Country;
25
26 DROP TABLE t;

```

---

### 2.7.1 IN Operator

Mit dem In-Operator können mehrere Bedingungen geprüft werden. Das bedeutet

---

```

1 SELECT *
2 From Country
3 WHERE Name IN(Deutschland, Japan, Frankreich); -- Die Syntax ist nicht = In(...)
```

---

In der Spalte [Name] wird nach den Werten *Deutschland*, *Japan* oder *Frankreich* gesucht. Das gleiche Ergebnis ist auch erreicht, wenn zwei **OR** Operatoren verwendet werden würden. Mit Subselect kann eine Liste als Speicherort für multiple Bedingungen verwendet werden.

---

```

1 SELECT *
2 From Country
3 WHERE Code2 IN((SELECT SUBSTR([b],1,2) AS Country_Code, SUBSTR([b],1,1))
4 From t)
5 ;
```

---

Die Bezugstabelle muss als Liste vorliegen. Das bedeutet, eine Tabelle kann nicht vom IN() Operator verwendet werden.

## 2.7.2 View - temporary table

Subselect können in View gespeichert werden. Diese befinden sich als Unterpunkt unter **Tabellen**.



Abbildung 192:

Über *Drop View Beispielname* werden diese entfernt. Der Befehl zur Generierung sieht wie folgt aus:

---

```

1 CREATE VIEW view_name AS
2 SELECT column1, column2, ...
3 FROM table_name
4 WHERE condition;
```

---

Beispiel *track*:

	<b>id</b>	<b>album_id</b>	<b>title</b>	<b>track number</b>	<b>duration</b>
1	1	1	Bright Lights Big City	1	320
2	2	1	Night Life	2	344
3	3	1	Sweet Blues	3	296
4	4	1	Cardinalia	3	205
5	5	1	Stardust	4	308
6	6	1	Cards On My Mind	6	330
7	7	1	Rainy Day Blues	7	343
8	8	1	My Bucket's Got A Hole In It	8	296
9	9	1	It Ain't Necessarily Business	9	447
10	10	1	That's All	10	368
11	14	11	Johnny B. Goode	1	285
12	15	11	Blue Suede Shoes	2	185
13	16	11	Blue Suede x Shoes	3	296
17	17	11	Voodoo Chile	4	469
15	18	11	The Queen	5	160
16	19	11	Sgt. Pepper's Lonely Hearts Club Band	6	70

Abbildung 193:

Die Bewertung der Dauer wird in der View Tabelle *DurationView* gespeichert.

## **Teil V**

# **Version Controll**

# 1 GitHub Introduction

## 1.1 Befehle

**my-slide Verzeichnis**

**my-slide Verzeichnis**

**git ls-files** Zeigt alle Dateien in einem Verzeichnis an.

**copy con <file>** Dieser Befehl funktioniert unter Git bash. Die entsprechenden Datei wird in den aktiven Verzeichnis erstellt.

**git status** Gib alle Änderungen eines Verzeichnisses wieder.

**git clone <URL.git>** Das URL mit .git wird in dem aktiven Verzeichnis geklont.

**git branch <name>** Ein neuer Branch wird im lokalen Bereich erzeugt.

**git checkout <local branch>** Ändert den aktiven lokalen Branch.

**git push -set-upstream origin <Branch>** Der erzeugte Branch wird zum Remote Repository geladen.

**git pull origin <Branch>** Ein Update (pull) eines spezifischen Branch wird gezogen.

**git branch -all** Alle Branches lokal und remote werden angezeigt.

**git branch -m <Name>** Der Name des Branches, welcher gerade aktiv ist, kann mit dem Befehl geändert werden.

**git -add <file>** Bereitet Änderungen aus dem Working Bereich für den nächsten Commit vor.

**git commit -m "..."** Mit jedem Commit wird ein Snapshot der Dateien gemacht. Diese werden unter Version Control System (VCS) abgespeichert. Eine Nachricht ist notwendig.

**git push** Alle Commit werden für den ausgewählten Branch gepusht.

**git pull origin <Branch>** Aktualisiert den spezifischen Branch.

**git merge <Branch>** Der Master Branch wird mit dem ausgewählten Branch vereinigt.

**git branch -d <Branch>** Löscht den lokalen Branch, welcher schon mit master vereinigt ist.

**git branch -D <Branch>** Löscht den lokalen Branch.

**git push origin --delete <Branch>** Löscht den remote Branch, solang er auf Git Hub noch nicht "archiviert" wurde, sonst kann der Branch nicht gefunden werden.

**git remote prune origin** Löscht leere Branch Hüllen.

## 1.2 Unterschied zwischen git bash, cmd und gui

**Git bash** Es handelt sich hier um eine UNIX Shell. Dies bietet sich an, wenn Linux schon bekannt ist.

**Git CMD** Es handelt sich hier um eine UNIX Shell. Dies bietet sich an, wenn Windows bekannt ist.

**Git GUI** Bietet die Funktionalität von git in einer grafische Oberfläche an.

[stackoverflow o. D.](#)

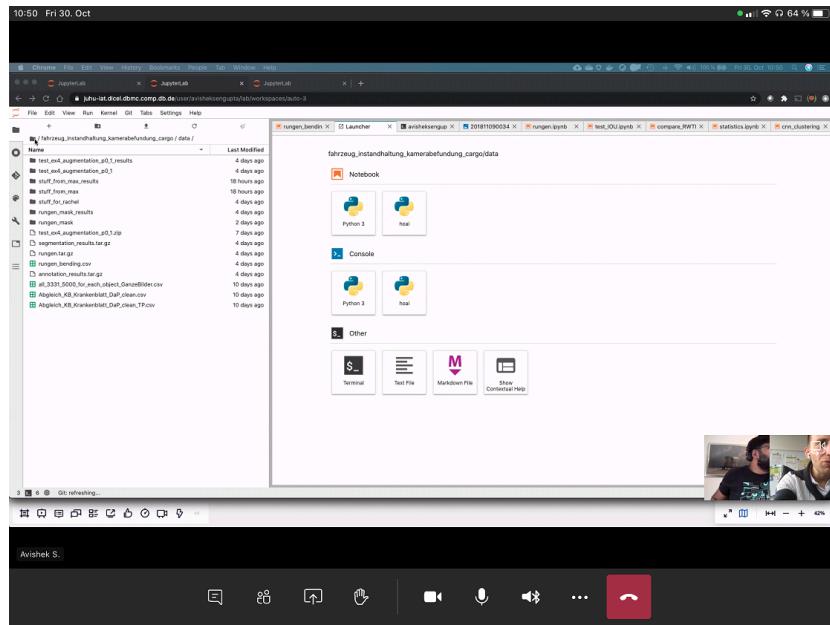
### 1.2.1 Staging Area / Index / Cache

Der Staging Bereich oder Index erlaubt inhaltlich verbundenen Dateien/ Änderungen an Dateien aus dem Working Bereich vorzuhalten und per **git add**.

```
1 C:\Users\Paul J\Documents\GitHub\github-slideshow> git status
2 On branch dell_Branch
3 Your branch is up to date with 'origin/dell_Branch'.
4
5 Untracked files:
6 (use "git add <file>..." to include in what will be committed)
7 - posts/test-dell.md - Copy.txt
8
9 nothing added to commit but untracked files present (use "git add" to track)
```

Dateien, welche diesem Index zugeordnet wurden, werden per **git commit** mit einer Nachricht zum lokalen Git Repository übertragen. Die gesammelten Commits können per **git push** zum Remote Repository übertragen werden.

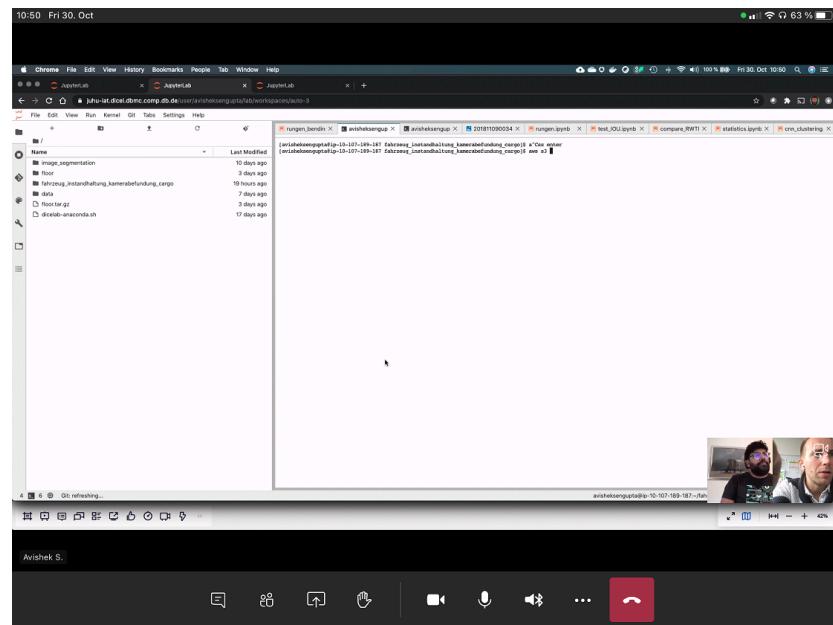
```
1 C:\Users\Paul J\Documents\GitHub\github-slideshow> git status
2 On branch dell_Branch
3 Your branch is ahead of 'origin/dell_Branch' by 1 commit.
4 (use "git push" to publish your local commits)
5
6 nothing to commit, working tree clean
```



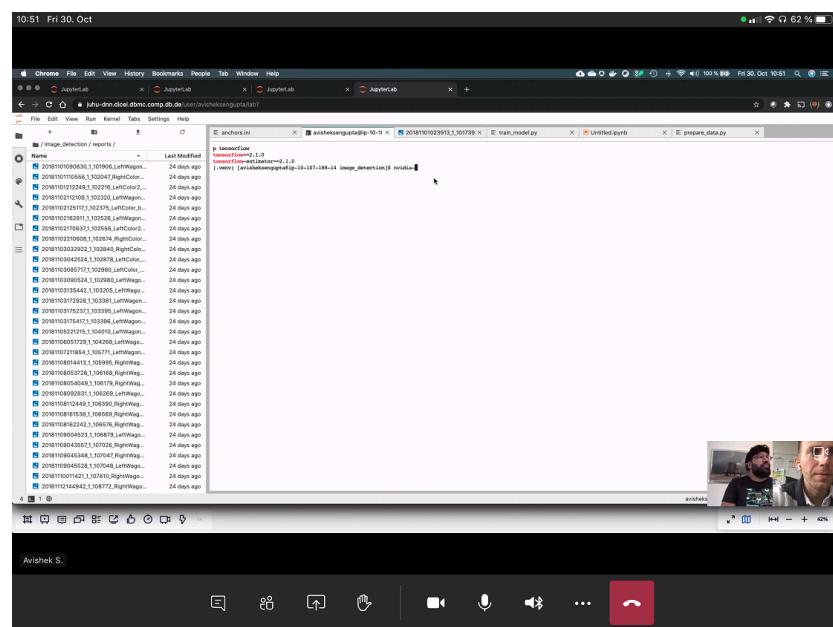
Eine Datei im Git Ordner kann in zwei Kategorien untergliedert werden:

- tracked und
- untracked.

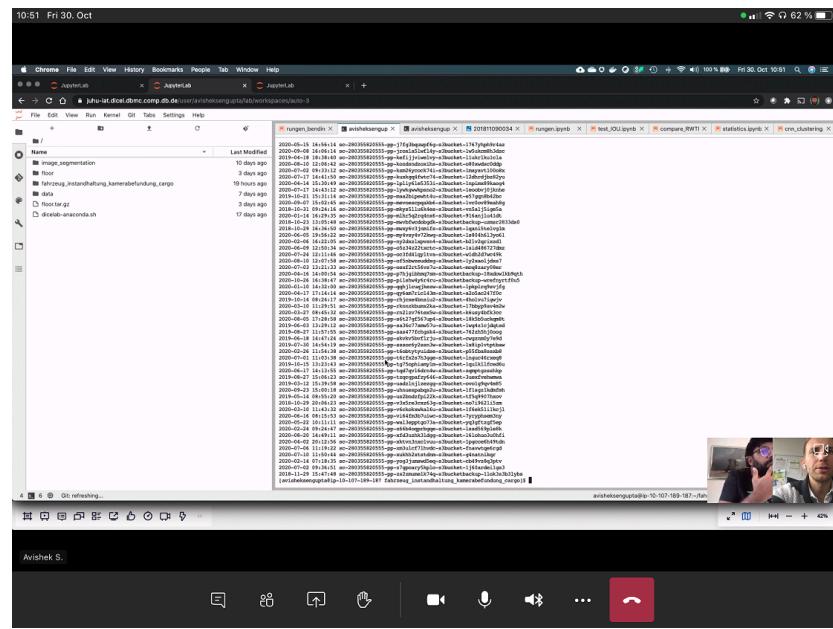
In früheren Git Versionen konnte eine ungetrackte Datei Git Repository zugeordnet werden ohne als *staged* markiert zu werden, sodass sie Teil der VCS wurde. Mit **git add <file>** wird diese Datei heute zum Staging Bereich zugeordnet und von Git getrackt. Beim nächsten **git push** wird diese Datei ans Remote Git Repository gesendet. Mit Git GUI ist eine Übersicht gegeben, welche Dateien noch nicht im Stage Modus sind und welche schon.



Die Commits werden pro Branch aufgeteilt.



Zu jedem Schritt existiert ein Befehl in Git Repository.



## 1.2.2 Committing

Commits sind Snapshots/ Milestones des Projektes/ Ordners oder Dateien. Dies werden werden auf dem lokalen Git Repository abgespeichert. Commits können auch wieder gelöscht werden. Wenn alle Commits bereit sind, geteilt zu werden, können sie per `git push origin <Branch>` auf das remote Git Repository geladen werden. Über Git Hub stehen die Veränderungen und die History bereit zur Bearbeitung.

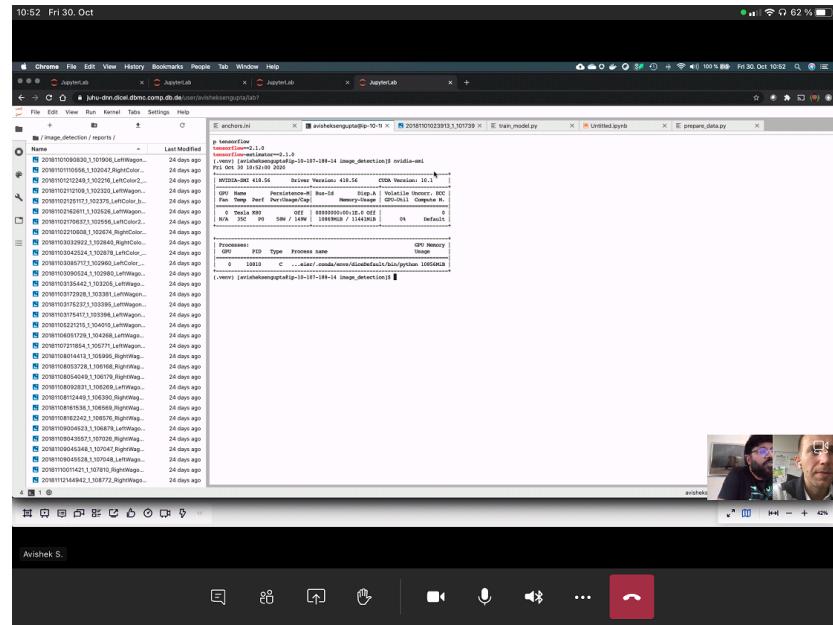


Abbildung 194: Richtig Auswahl des Branches & History anzeigen lassen.

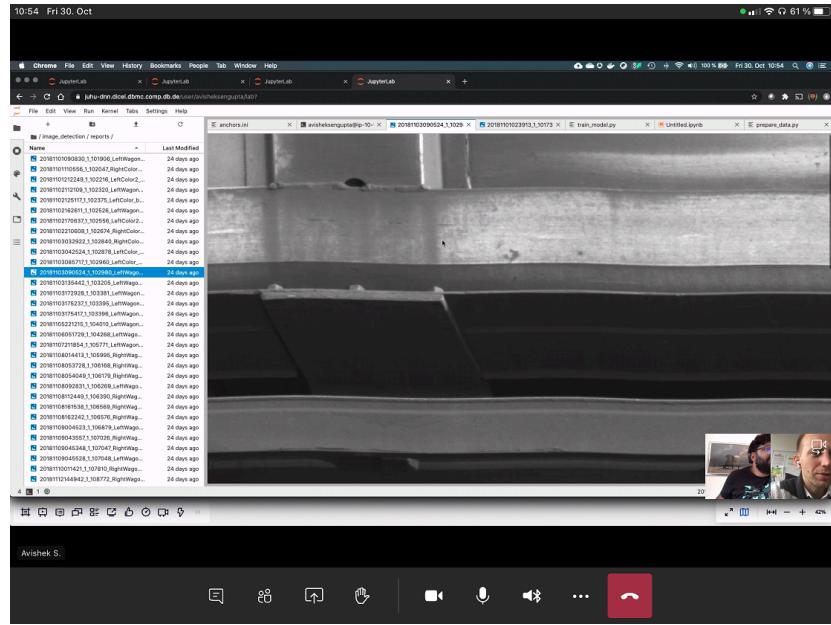


Abbildung 195: Zu jedem Milstone/ Linie besteht die Möglichkeit Kommentare anzubringen.

## 1.3 Example: github-slideshow

### 1.3.1 Clone Repository and Create a Branch

Ziel in diesem Abschnitt ist es, das Mit der Clone Anweisung wird das Git Repository auf den lokalen Server des Users gespielt. Die VCS wird in Git hinterlegt und speichert die Änderungen an den Daten. Damit die Änderungen sich einfach einbinden lassen und von verschiedenen User erstellt werden, wird ein **Branch** erstellt.

```

1  /// Change directory to where the repository should be saved
2  git cd C:\Users\PaulJulitz\Documents\GitHub
3  /// Clone repository to local server (.../Username/NameRepository)
4  git clone https://github.com/pauljulitz/github-slideshow.git /**
5  /// Navigate to the repository in your shell
6  git cd C:\Users\PaulJulitz\Documents\GitHub\github-slideshow
7  /// Create a new branch
8  git branch my-slide

```

### 1.3.2 Check all the branches of one repository

In einem Git Repository können mehrere Branches erstellt werden. Welche für das gewissen Git Repository zugewiesen sind, kann über **branch -all** in einen Git Repository dargestellt werden.

```

1  C:\Users\PaulJulitz>cd C:\Users\PaulJulitz\Documents\GitHub\github-slideshow
2
3  C:\Users\PaulJulitz\Documents\GitHub\github-slideshow> git branch --all
4  master
5  *my-slide
6  secoundBranch
7  remotes/origin/HEAD -> origin/master
8  remotes/origin/master
9  remotes/origin/my-slide
10  remotes/origin/secoundBranch

```

Der aktuelle Branch wird in **Grün** mit einer \* angezeigt.

### 1.3.3 Chance branch name

Mit dem Befehl **-m <new name>** kann der Name des lokalen, ausgewählten Branch geändert werden. Wenn jedoch der gleiche Name für einen Branch ausgewählt wurde, wird der Zusatz *-lokal* angefügt.

```
1 C:\Users\PaulJulitz\Documents\GitHub\github-slideshow> git branch -m newName
```

#### 1.3.4 Change aktiv branch

Ein Branch kann mit **git checkout <local branch>** gewechselt werden.

```
1 C:\Users\PaulJulitz\Documents\GitHub\github-slideshow> git branch --all
2 master
3 * my-slide
4 secoundBranch
5 remotes/origin/HEAD -> origin/master
6 remotes/origin/master
7 remotes/origin/my-slide
8 remotes/origin/secoundBranch
9
10 C:\Users\PaulJulitz\Documents\GitHub\github-slideshow> git checkout secoundBranch
11 Switched to branch 'secoundBranch'
12 Your branch is up to date with 'origin/secoundBranch'.
13
14 C:\Users\PaulJulitz\Documents\GitHub\github-slideshow> git branch --all
15 master
16 my-slide
17 * secoundBranch
18 remotes/origin/HEAD -> origin/master
19 remotes/origin/master
20 remotes/origin/my-slide
21 remotes/origin/secoundBranch
```

#### 1.3.5 Push new local branch

Mit **--set-upstream origin <branch>** wird der angesteuerte Branch hochgeladen.

```
1 git push --set-upstream origin my-slide
```

#### 1.3.6 Check the status of a branch

Der Befehl **git status** gibt an,

- welcher Branch aktuell aktiv ist,
- ob es Änderungen im Origin Projekt gibt und
- welche Änderungen gepusht werden müssen.

```
1 C:\Users\PaulJulitz\Documents\GitHub\github-slideshow> git status
2 On branch secoundBranch
3 Your branch is up to date with 'origin/secoundBranch'.
4
5 nothing to commit, working tree clear
```

#### 1.3.7 Create a new file

Unter Git CMD wird der Befehl **copy con** nicht ausgeführt. Der Befehl **ls** ebenso nicht. Mit

#### 1.3.8 Update Branch

Mit dem Befehl **git pull origin <branch>** wird der spezifische Branch aktualisiert.

```
1 C:\Users\PaulJulitz\Documents\GitHub\github-slideshow> git pull origin my-slide
2 remote: Enumerating objects: 10, done.
3 remote: Counting objects: 100% (10/10), done.
4 remote: Compressing objects: 100% (7/7), done.
5 remote: Total 8 (delta 3), reused 0 (delta 0), pack-reused 0
```

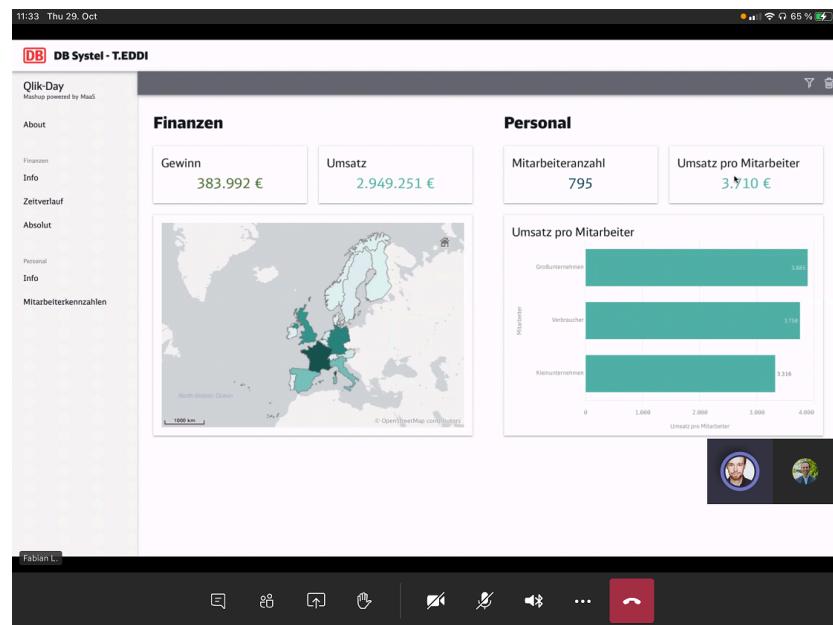
```

6 Unpacking objects: 100% (8/8), 1.46 KiB | 14.00 KiB/s, done.
7 From https://github.com/pauljulitz/github-slideshow
8 * branch      my-slide    -> FETCH_HEAD
9 d64d186..e0c65fb my-slide    -> origin/my-slide
10 Updating d64d186..e0c65fb
11 Fast-forward
12 _posts/0000-01-02-PaulJulitz.md | 6 ++++++
13 1 file changed, 6 insertions(+)
14 create mode 100644 _posts/0000-01-02-PaulJulitz.md
15
16 C:\Users\PaulJulitz\Documents\GitHub\github-slideshow> git status
17 On branch my-slide
18 Your branch is up to date with 'origin/my-slide'.
19
20 nothing to commit, working tree clean

```

### 1.3.9 Pull Request

Die kann über Github selbst erfolgen:



Oder über command-line interpreter (cmd) sind die folgenden Schritte notwendig. Der Master Branch wird aktiviert.

```
1 C:\Users\PaulJulitz\Documents\GitHub\github-slideshow> git checkout master
```

Der benötigte Branch wird über **git merge** vereinigt.

```
1 C:\Users\PaulJulitz\Documents\GitHub\github-slideshow> git merge my-slide
```

Dieser muss jetzt gepusht werden.

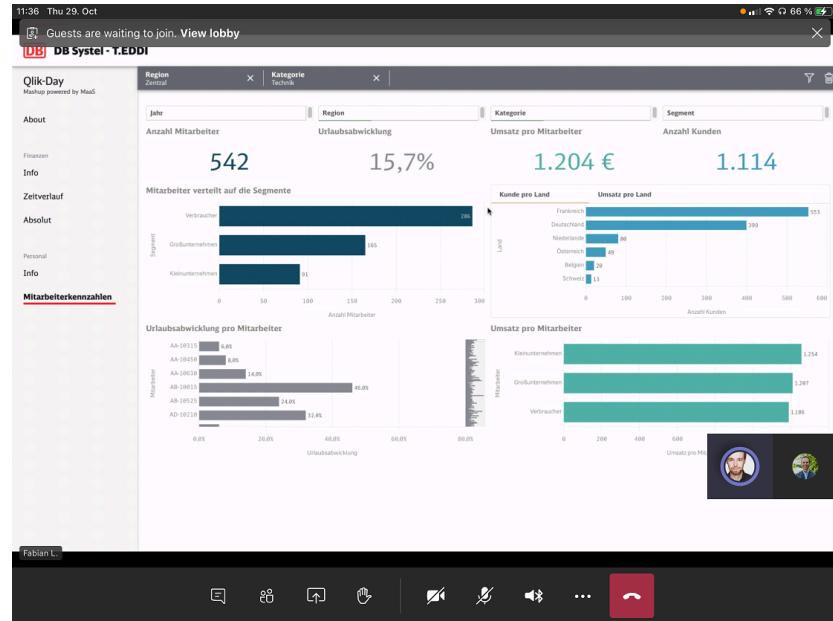
```
1 C:\Users\PaulJulitz\Documents\GitHub\github-slideshow> git push
```

Der alte Branch kann jetzt gelöscht werden.

```
1 C:\Users\PaulJulitz\Documents\GitHub\github-slideshow> git branch -d my-slide
```

### 1.3.10 Delete local branch

Wenn die Veränderungen überprüft wurden und der Branch vereinigt wurde, wird das Prinzip verfolgt, den Branch zu löschen. Wenn dies über die Git Hub Plattform passt, so bleibt eine "Ablage" weiter bestehen.



```

1 C:\Users\Paul J\Documents\GitHub\github-slideshow>git push origin --delete dell_Branch
2 error: unable to delete 'dell_Branch': remote ref does not exist
3 error: failed to push some refs to 'https://github.com/PaulJulitz/github-slideshow.git'

```

Wenn dies nicht getan wurde oder der Branch wieder zurückgesetzt wurde, wird er remote Branch gelöscht, durch:

```

1 C:\Users\Paul J\Documents\GitHub\github-slideshow>git branch --all
2 * dell_Branch
3 master
4 remotes/origin/HEAD -> origin/master
5 remotes/origin/dell_Branch
6 remotes/origin/dell_Branch_II
7 remotes/origin/master
8 remotes/origin/my-slide
9 remotes/origin/secoundBranch
10
11 C:\Users\Paul J\Documents\GitHub\github-slideshow>git pull
12 Already up to date.
13
14 C:\Users\Paul J\Documents\GitHub\github-slideshow>git push origin --delete dell_Branch_II
15 To https://github.com/PaulJulitz/github-slideshow.git
16 - [deleted]          dell_Branch_II
17
18 C:\Users\Paul J\Documents\GitHub\github-slideshow>git push origin --delete dell_Branch
19 To https://github.com/PaulJulitz/github-slideshow.git
20 - [deleted]          dell_Branch
21
22 C:\Users\Paul J\Documents\GitHub\github-slideshow>git branch --all
23 * dell_Branch
24 master
25 remotes/origin/HEAD -> origin/master
26 remotes/origin/master
27 remotes/origin/my-slide
28 remotes/origin/secoundBranch

```

### 1.3.11 Delete remote branch

Der Befehl ist **git push origin --delete origin/<branch>**. Dieser ist an ein Pull Request welcher den Branch im remote Git Repository löscht. Wenn dieser Branch nicht mehr existiert, kommt ein Fehler zurück.

### 1.3.12 Prune deletet remote branches

Der Befehl `git remote prune origin` löscht alle Hüllen eines Branches.

## 1.4 Designing your Github Page

Formatierung einer .md Datei. Das README.md wir als Startseite unter der Auflistung der Daten angezeigt. Dies zu Formatieren kann helfen, eine Einführung in das Git Repository zu geben.

- Ein Hashtag oder mehrere Hastagss geben die Größe der Überschrift an.
- Bilder aus dem Web werden per `![profile-image](https://apod.nasa.gov/...)` eingefügt.
- Wenn nur ein Bildlink eingefügt werden soll, muss das `!`  vor dem Platzhaltern `[...]`.
- Die Logik von Whatsapp ist auch hier Teilweise bei .md Formartierung anzuwenden (Kursiv, Fett, List).

## 2 GitLab Introduction

Dieser Kurs ist ein Einsteigerkurs. Im Weiteren werden nur die wichtigsten oder neuesten Erkenntnisse notiert.

### 2.1 Blame

Mit der Blame-Funktion kann der Versionsverlauf einer bestimmten Datei nachgeschlagen werden.

```
main-Hauptdokumentation.tex · Working_Paul · SBM_M12_....
```

Auskommentieren  
Paul Juilitz authored 4 days ago

```
1 \documentclass[10pt]{report} % Geeignet fuer langere Berichte, kurze Buecher oder Thesen
2
3 \input{alongside/usepackage}
4 \input{alongside/command}
5
6
7 \input{appendix/acronym}
8 \input{appendix/acronym_db}
9 \input{appendix/glossaries}
10 \input{appendix/glossaries_db}
11 \addbibresource{alongside/Bib_1.bib} % Einbinden der bib-Datei
12
13 %% Used Compiler XeLaTtex
14
15 \begin{document}
16
17 %% Titel Seite
18 \input{alongside/titlepage}
19
20 %% Inhaltsverzeichnis
21 \pagebreak
22
```

Dabei werden die Commits und die Veränderung direkt nebeneinander angezeigt.

```

\documentclass[10pt]{report} % Geeignet fuer langere Berichte, kurze Bue
\input{alongside/usepackage}
\input{alongside/command}
\input{appendix/acronym}
\input{appendix/acronym_db}
\input{appendix/glossaries}
\input{appendix/glossaries_db}
\addbibresource{alongside/Bib_1.bib} % Einbinden der bib-Datei
%%% Used Compiler XeLaTatex

```

Dies unterscheide die History Funtkion, welche nur die Commit History anzeigt.

## 2.2 Groups and Subgroups

Unter Gruppen können auch Subgruppen erstellt werden. Diese dienen einen klarern Überblick zu schaffen.

SBM\_M12\_DSci

**Subgroups and projects** Shared projects Archived projects

Search by name Name

P Pipelines Owner

A Algorithmus\_Trupp\_Finding ★ 0 1 month ago

D Documentation ★ 0 2 weeks ago

E ETL\_Fraud\_Analysis ★ 0 1 month ago

O OKR\_Prognose-XPD ★ 0 2 months ago

P Pipeline\_FAKS ★ 0 2 months ago

P Pipeline\_iPD ★ 0 2 months ago

P Pipeline\_Paigo ★ 0 2 months ago

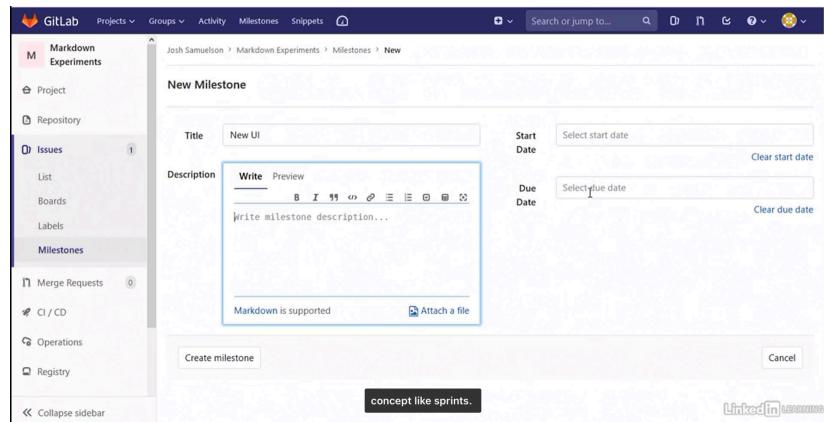
P Pipeline\_SharePoint ★ 0 1 month ago

## 2.3 Role Permissions

Action	Guest, Reporter	Developer	Maintainer	Admin
See commits and jobs	✓	✓	✓	✓
Retry or cancel job		✓	✓	✓
Erase job artifacts and job logs		✓ (1)	✓	✓
Delete project			✓	✓
Create project			✓	✓
Change project configuration			✓	✓
Add specific runners			✓	✓

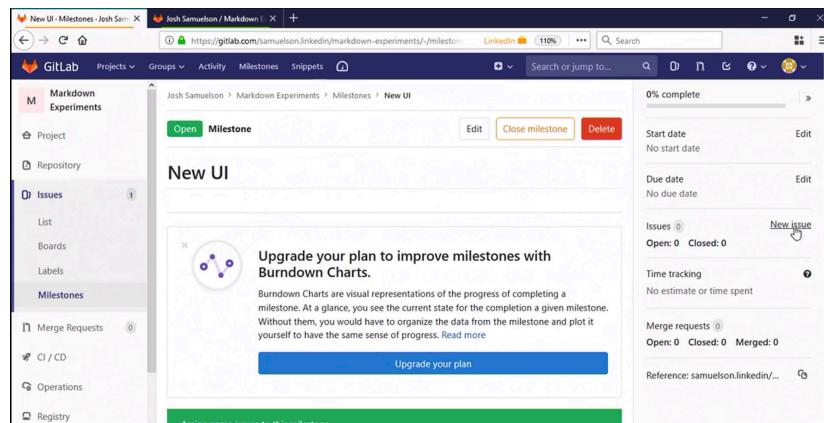
## 2.4 Milestones

Mehrere Issues können einem Milestone zugeordnet werden. Es gibt noch die Möglichkeit Epics zu erstellen, diese wird jedoch nicht in jedem GitLab Packet zur Verfügung gestellt.



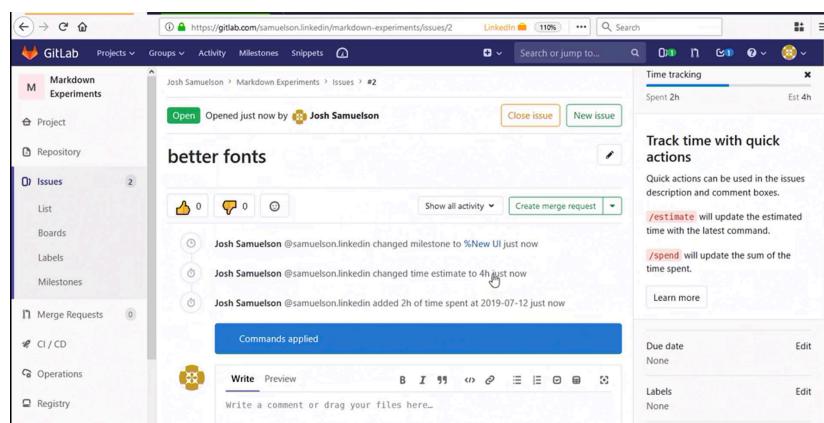
Bei einem Milestone wird ein Start- und Enddatum angefügt.

In der Darstellung eines Milestone wird auf der rechten Seite der Status über den Milestone angezeigt.



Die Funktionsweise ist, dass über jedes erstellte Issue auch eine *estimated* und *spend* Zeit angegeben werden kann.

Hinweis: Es können mehrere Teilaufgaben zu einem Issue erstellt werden - siehe Checkboxen.



## 3 \*Virtual Environment Anaconda

## 4 Version Kontrolle Stategie

Es gibt verschiedene Strategien wie in Programmierungsprojekt über Git verwaltet werden können. Im Folgenden wird Bezug auf die Feature-Hotfix-Stategie von Vincent Driessen auf [Successful git branching model](#) beschrieben wird.

### 4.1 Verschiedene Branches

- master (protected, stays)
- develop (protected, stays, merge into master, default)
- feature (branched from develop, merge into develop, delete after, syntax f\_)
- hotfix (branched from master, merge in develop and master, delete after both merges)
- release (optional)

#### 4.1.1 Master and Develop

Es existieren zwei dauerhafte Branches im Repository. Dieses sind **master** und **develop**. Diese sind geschützt (protected). Die Einstellungen zu den beiden Branches sind,

- dass nur der **Maintainer** Merger Pulls oder Push bewilligen kann.
- dass keine Commits gesendet werden können. Es kann nur ein Merge stattfinden.

Der **Default** Branch wird von master auf develop umgestellt. Dies hat den Hintergrund, dass Features auf Basis von Develop gebracht werden sollen. Ebenso sollen fertige Feature Branches in Develop gemerget werden. In diesem Stadium kommt es zur Überprüfung, ob das Programm dies tut, was es tun soll. Erst dann, wird es in master gemerget, welcher als Produktion Branch fungiert.

#### 4.1.2 Feature and Hotfix

- In Feature branch, kann auch aus einem Issue heraus erstellt werden.
- Ebenso kann ein Pull-Request aus einem Issue erstellt werden.
- Feature werden erst in Develop eingepflegt, wenn der Review Prozess abgeschlossen ist.<sup>5</sup>
- Um Komplikationen beim Merging zu verhindern, sollen features nicht auf die gleichen Dateien ändern.
- Es empfiehlt sich, –no-ff bei Merge von Feature Branches zu verwenden. Dieser Befehl erzeugt, ein neues Commit-Objekt am Ende. Diese verursacht mehr Komplexität, diese lässt jedoch zu, dass im Develop Branch nachzu vollziehen ist, welche Commits aus welchem Feature kommen.
- Hotfix ist ähnlich zu Feature. Dieser kann direkt von master gebracht werden. Die Reparaturen werden dann in Master und Develop eingepflegt. Danach wird der Hotfix Branch gelöscht.

### 4.2 Merging Optionen

Git bietet verschiedene Merging-Strategie. Einzelnen können dabei kombiniert oder gleich im der Integrated development environment (IDE) verwendet werden. Im Folgenden werden

- Merge (fast forward)
- Merge Recursive
- Merge Squash
- Merge –no –ff (no fast forward)
- Merge Rebase

behandelt.

---

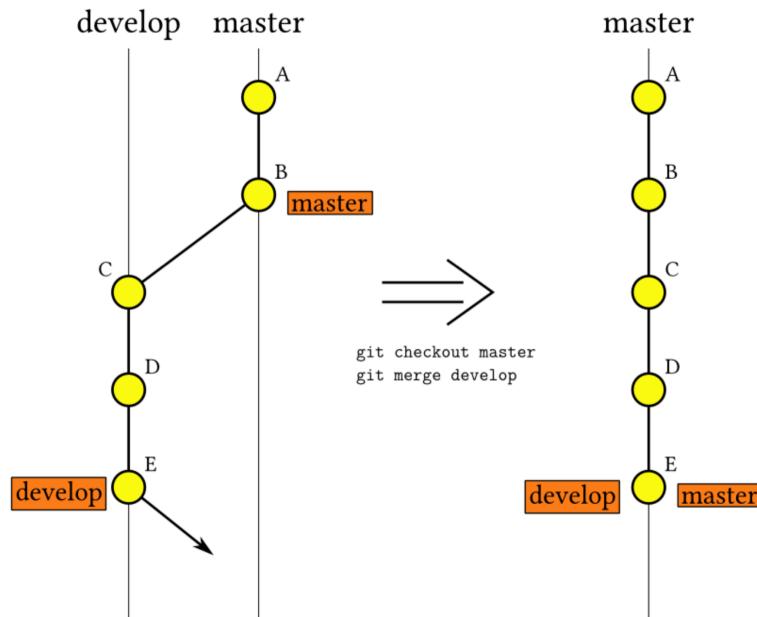
<sup>5</sup>Wird über die Kommentare sich ausgetauscht, welche Änderungen noch hinzugefügt werden müssen. Der Autor kann daraufhin weitere Commits tätigen. Diese Prozess wiederholt sich, bis alle Anforderungen akzeptiert sind.

#### 4.2.1 Merge (fast forward)

Sei Develop Branch von Master am Punkt *B* geteilt worden. Dies stellt die gemeinsame Basis für beider Branches dar. Wurde daraufhin in Develop drei commits getätigt, und keiner in Master, kann durch in Merge ein natloser Übergang geschaffen werden.

Git wählt die Strategie eigenständig aus, welche am besten zur Situation passt.

Mit dem Merge Befehl wird Develop in Master übernommen. Dabei werden die drei Commits in Master übernommen. Liegt der Pointer bei Master wird Master wie folgt nach dem Merge dargestellt.



Das warum hinter Merge *fast forward* steht, liegt daran, dass der Pointer von Master ohne ein weiteren Commit zum Ende von Develop geführt wird, ohne ein weiteren Commit auszulösen. Diese Option besteht jedoch nur, wenn in zu mergenden Branch keine Commits getätigt werden.

#### 4.2.2 Merge Recursive

Feature wurde von Master vom Commit *Add new file* geteilt. In Master wurden zwei Commits *m1* und *m2* getätigt. In Feature befinden sich zwei Commits *f1* und *f2*, welche nach dem Branching commetiert wurden.

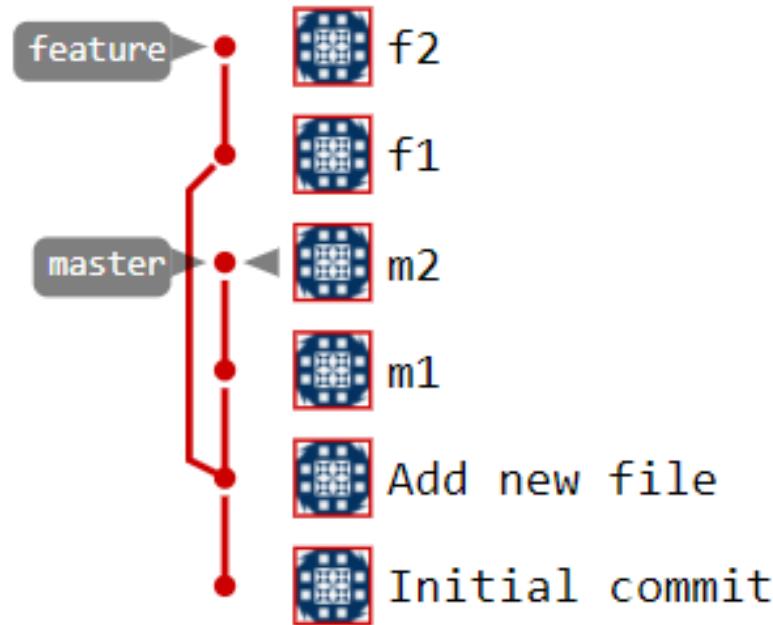


Abbildung 196: Pointer auf Master M2

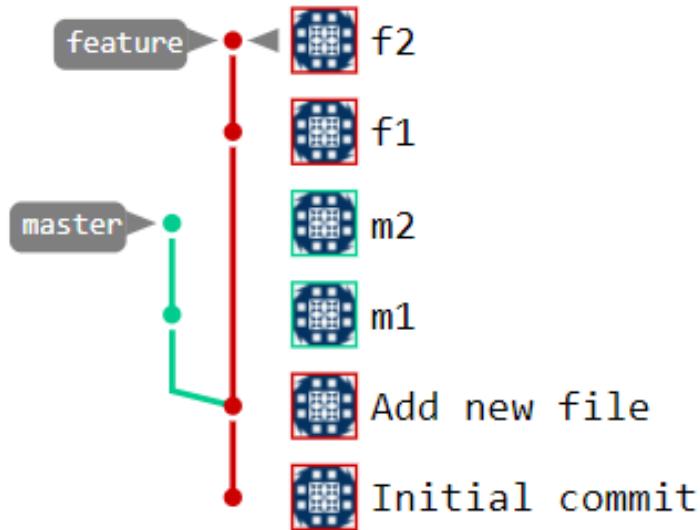
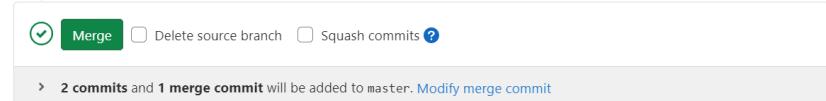


Abbildung 197: Pointer auf feature f2

Dies bedeutet, unter dem Merge-Request in Master befinden sich zwei Commits.

Request to merge feature into master  
The source branch is 2 commits behind the target branch

Wie angezeigt, wird ein zusätzlicher Commit zu Master generiert.



Die Option Feature zu löschen wird erst mal ausgelassen. Die Option Squasch commits findet sich unter **Merge Squash** wieder. Nachdem Feature unter master gemerget wurde, wird diese in dem Beispiel mit dem Commit **a33fd106** festgehalten.



In Graph wird auf Master der neue Commit angezeigt, und die f2 und f1 im Feature Branch angezeigt.

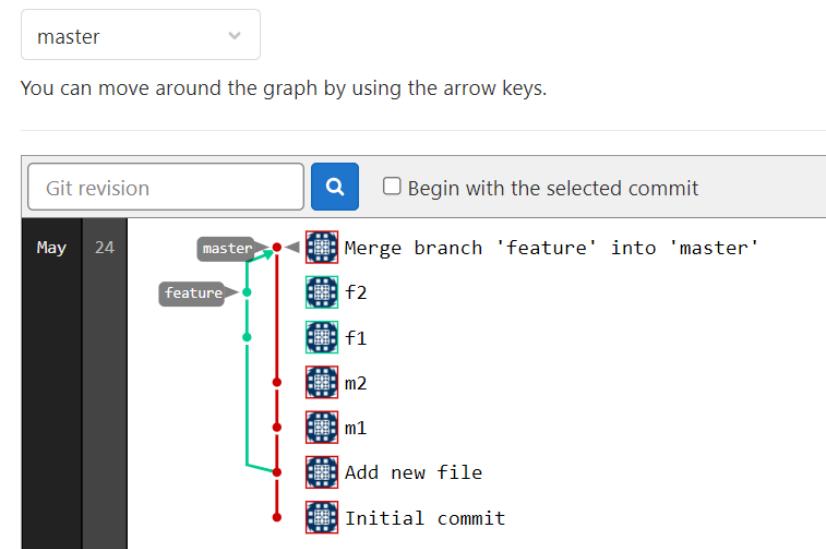


Abbildung 198: Pointer auf Master Merge Branch feature into master

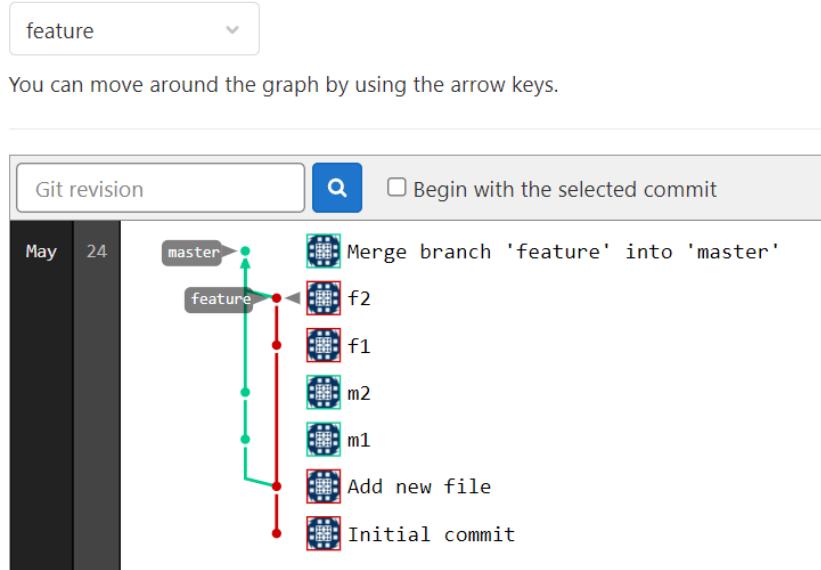


Abbildung 199: Pointer auf Feature f2

Die individuell, veränderten Dateien zweigen in Master die zugehörigen Commits an.

This screenshot shows a Git commit history for the 'master' branch. The top part displays a single merge commit: 'Merge branch 'feature' into 'master''. Below this, a table lists individual files and their corresponding commits:

Name	Last commit	Last update
README.md	m2	8 minutes ago
secound.py	f2	7 minutes ago

Die Historie zweigt alle Commits an. Mit Hinblick auf die anderen Strategien: Es wird nicht angezeigt, welcher Commit zu welchem Source Branch gehört.

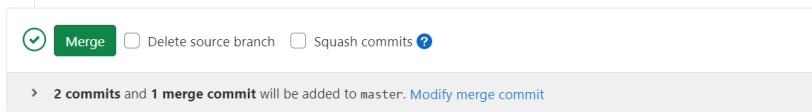
This screenshot shows a Git commit history for the 'master' branch. It lists all commits from both the 'feature' and 'master' branches. The commits are ordered by author and date. Each commit includes a small icon, the author's name, the commit message, and a unique commit hash.

Wird der Feature Branch nicht gelöscht, so bleibt er weiter bestehend. Die Commits aus Master werden jedoch nicht übernommen. Weitere Commits können in Feature getätigert werden, der Commit *m1* aus Master wird jedoch nicht mit übernommen. Für eine Änderung des letzten gemeinsamen Knoten wird **Rebase** verwendet.

Wie im Merge (fast forward) beschrieben, wählt Git selbst die beste Strategie aus. In diesem Fall wählt Git **Recursive Strategie** aus. Dabei wird ein Commit erstellt, welcher auf beide Branches verweist. Die Commits werden in der History unter dem Reisverschlussprinzip dargestellt. Die Strategie wird von Git angewandt wenn im Sourcebranch sowie im zu commettierenden Branch Commits vorliegen.

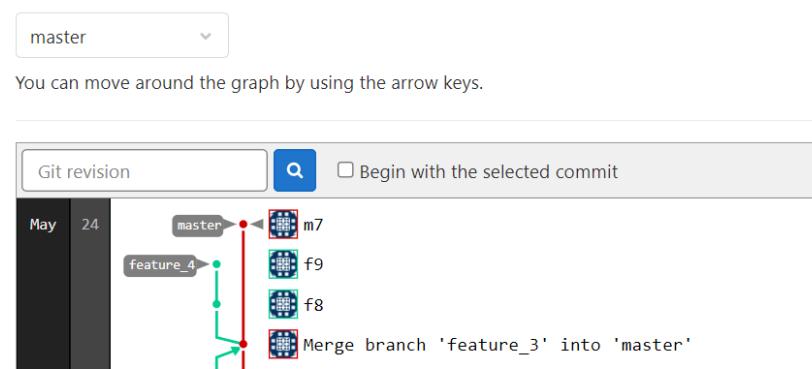
#### 4.2.3 Merge Squash

Die Option Squash

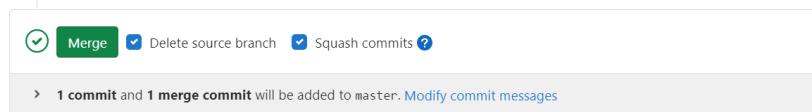


unter dem Merge von Feature, hätte alle Commits mit den jeweiligen Änderungen und aggregiert sie zusammengefasst. Alle Änderungen werden daher gebündelt in der Historie angezeigt.

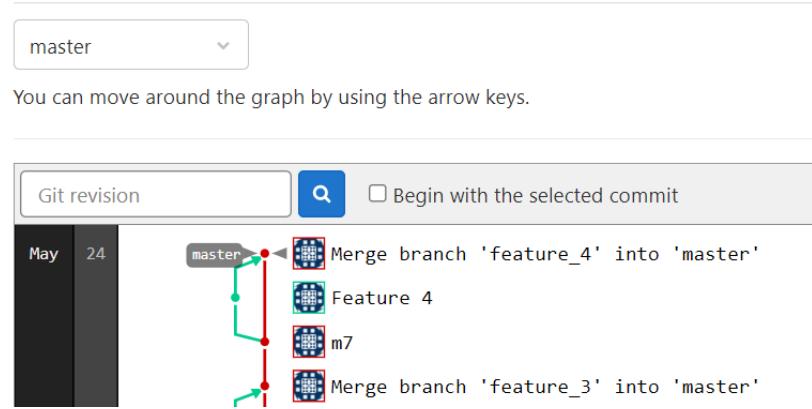
Unter Feature\_4 wurden Commits f8 und f9 getätigt. Ebenso wurde in Master m7 hinzugefügt.



Wir im Merge Squash ausgewählt, wird f8 und f9 zu einem Commit zusammengefasst.

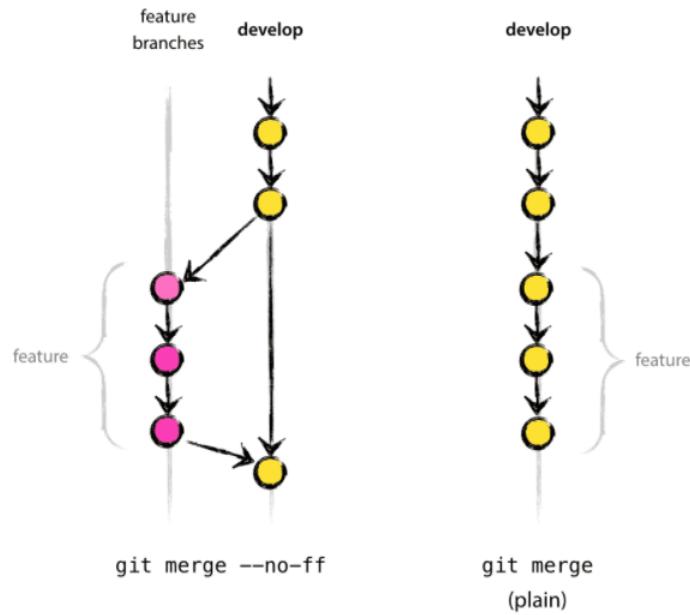


Weiter bleibt bestehend, dass durch die Recursive Strategy ein Commit für den Merge erstellt wird. Die Besonderheit ist, dass das Branching vom letzten Commit auf Master m7 angezeigt wird.



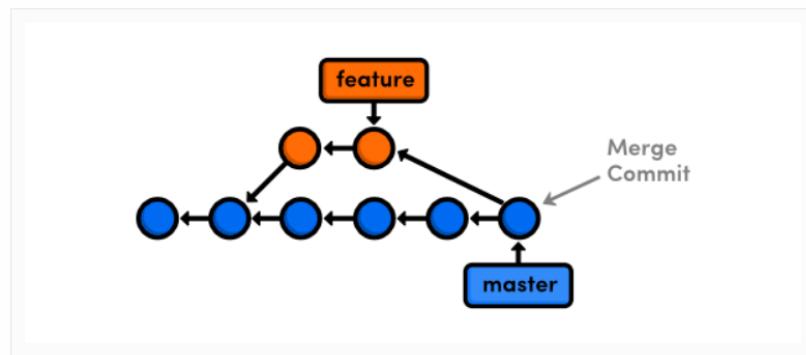
#### 4.2.4 Merge -no -ff

Mit der Option no fastward ist ähnlich des Squash Befehl ausgeführt. Die Ausnahme besteht daran, ein zusätzlicher Merge Commit getätigter wird.

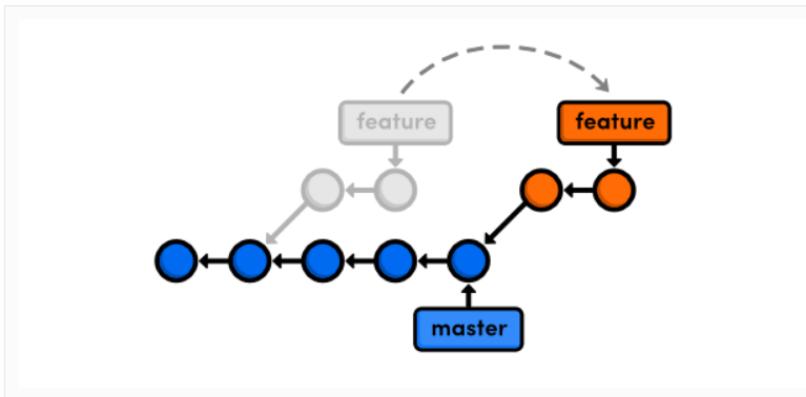


#### 4.2.5 Rebase

Mit der Funktion Rebase wird der gemeinsame, letzte Konten definiert.



Mit Rebase werden alle Comits in dem Feature Branch nach dem neuen Knotenpunkt neu commetiert.



## **Teil VI**

### **Qlik Sense**

# 1 Sammelsurium - Qlik Sense

## 1.1 Load

Load and Select Statement Was macht den Unterschied aus und wo liegen die Gemeinsamkeiten.

```
1 TabellenName :  
2 Load  
3 Spaltenname_1 ,  
4 Spaltenname_2 ,  
5 [  
6 ] ;  
7
```

## 1.2 Incremental Loading

Es ist möglich, nicht eine komplette Datenbank immer und immer wieder zu laden, um an die neuesten Datensätze zu kommen. Es ist möglich, durch [Incremental Loading](#). In dem Beispiel, [Qlik Snippet - Incremental Loading](#), wird über einen aufgesetzten SQL-Server gezeigt, wie neue Datensätze geladen werden können.

- Erst wird die Initialladung vorgenommen. Diese wird in einem QlikView Data (.qvd) gespeichert.
- Im zweiten Schritt wird die Incremental Loading Prozedur vor der Laden-Prozedur geschalten. Diese prüft jetzt, ob in der SQL Datenbank neuere Werte als in der dazugehörigen .qvd vorliegen. Diese werden dann geladen und in die alte .qvd gespeichert.

## 1.3 Data connection types

**Folder** Daten die lokal auf dem Server hinterlegt werden, können so abgerufen werden.

**Open Database Connectivity (ODBC)** Datenbank-Verbindungen, die am Server angelegt werden, können so mit Qlik Sense verbunden werden.

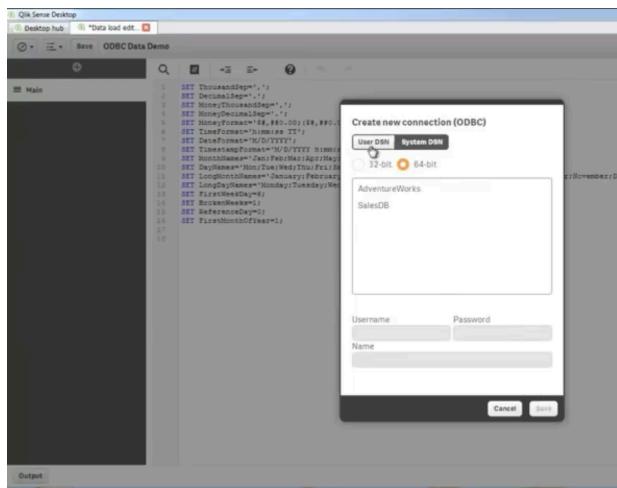


Abbildung 200:

Qlik bietet an, dass ODBC über Qlik Sense direkt angesteuert werden können oder über Microsoft.

**Load/ Select** Beide Ausdrücke generieren Tabellen. Dabei wird der Load Ausdruck verwendet, wenn es sich um lokale Dateien handelt. Der Select Ausdruck wird verwendet, wenn Datenbanken adressiert werden. Oft wird dabei auf SQL zurückgegriffen.

## 2 3 - Udemy - Certificate in Qlik Sense Analytics Development

### 2.1 Budgeting and KPI (goal) setting

#### 2.1.1 DevHub

Im Hauptmenü von Qlik Sense kann die DevHub Umgebung angesteuert werden.

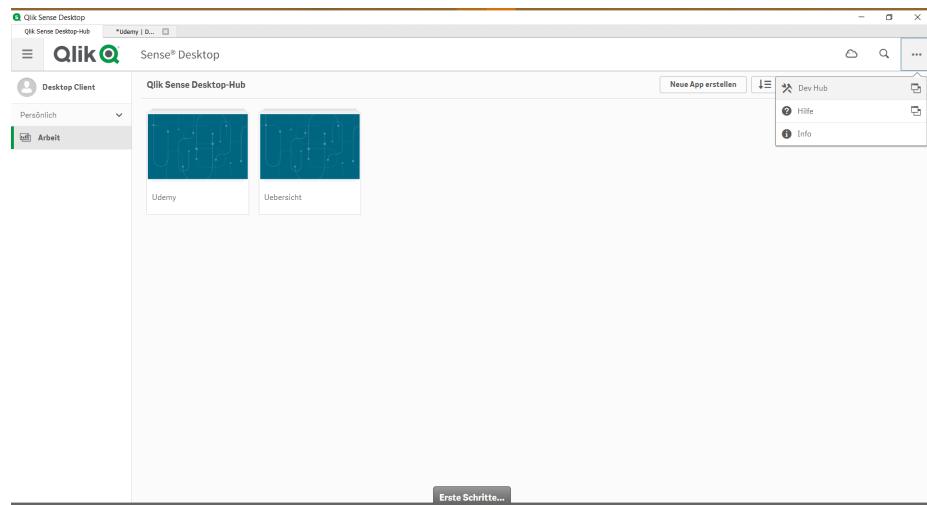


Abbildung 201:

Diese bietet an

- MashUps,
- Erweiterungen von Visualisierungen und
- Widget zur Verfügung zu stellen.

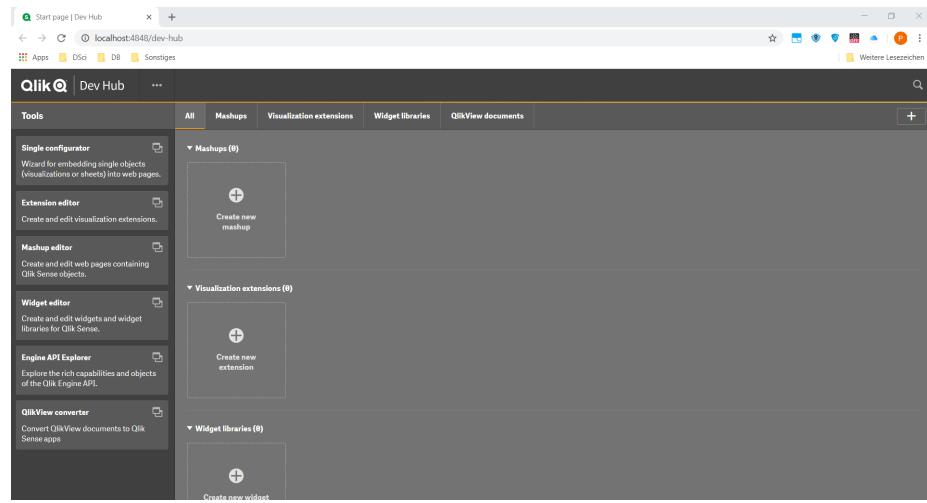


Abbildung 202:

Wichtige Punkte sind:

- Diese Features werden verwendet, um sie in andere Umgebungen einzubinden. Mashups werden so verwendet, um sie in SharePoint einzubinden.

- Die Plattform Qlik Branch gibt eine Übersicht über die verschiedenen Anbindungsmöglichkeiten zu Qlik Sense.
- Visualisierungserweiterungen werden über JavaScript (JS) programmiert.

### 2.1.2 Von Hub bis Sheet

Die Hub Oberfläche unterscheidet sich von dem Qlik Management Console (QMC), dass die Datenaufbereitung über diese Oberfläche zur Verfügung gestellt wird und QMC die Verwaltung übernimmt.

**Stream** In einem Stream können mehrere Apps veröffentlicht werden. Die Berechnungen wird über die Stream festgelegt.

**Apps** In einer App wird die Datengrundlage bestimmt. Das zugrundeliegende Datenmodell dient den Sheets.

**Sheets** Über Sheets können Visualisierungen erstellt werden.

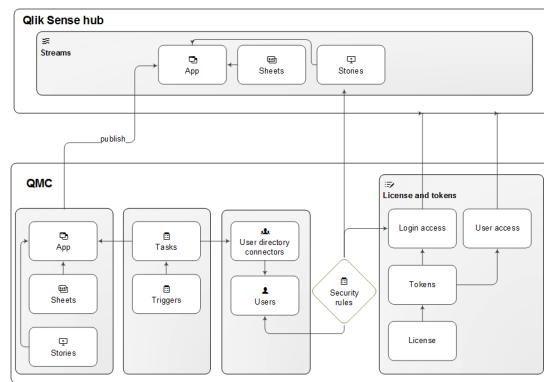


Abbildung 203:

Am Beispiel des Streams **Arbeit** wird die App **Udemy** erstellt.

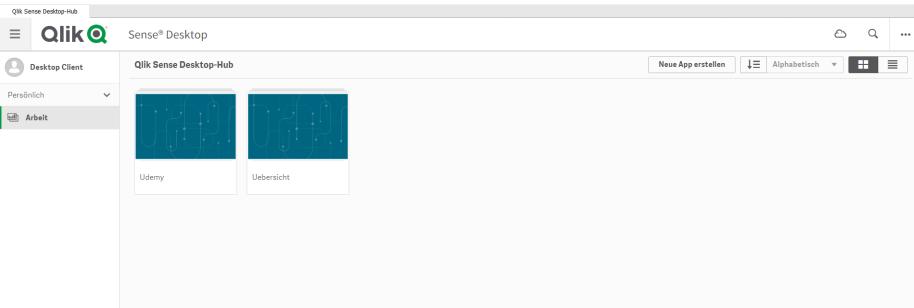


Abbildung 204:

Beim Öffnen der App erscheint die der Datenmanager.

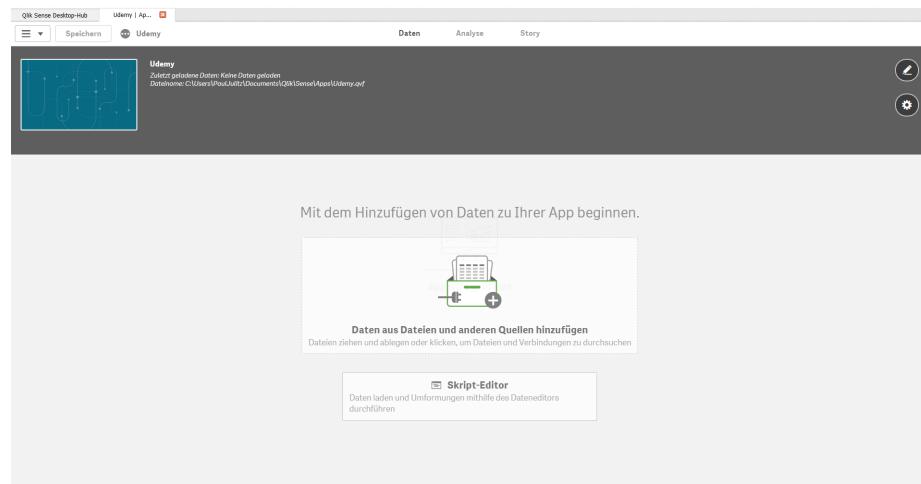


Abbildung 205:

Über diesen können verschiedenen Verbindung angesteuert werden.

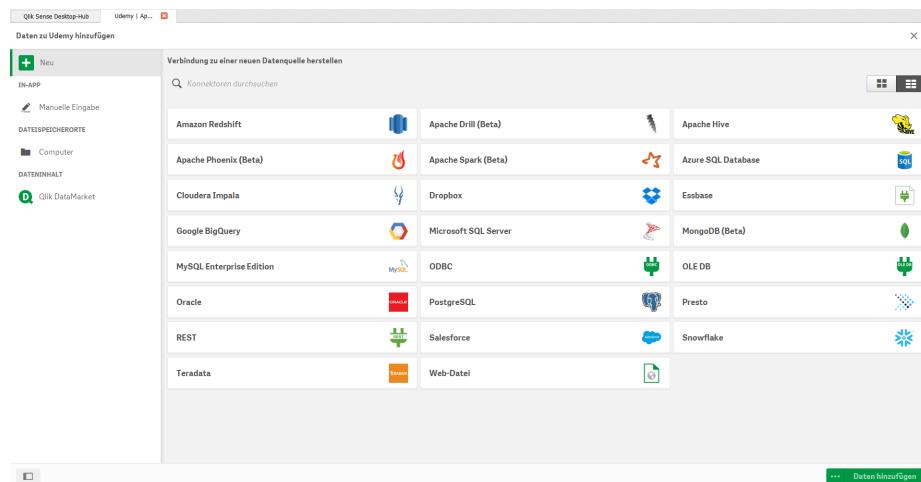


Abbildung 206:

Wird auf lokale Dateien zugegriffen, sieht das wie folgt aus:

Abbildung 207:

Nachdem die Daten geladen werden.

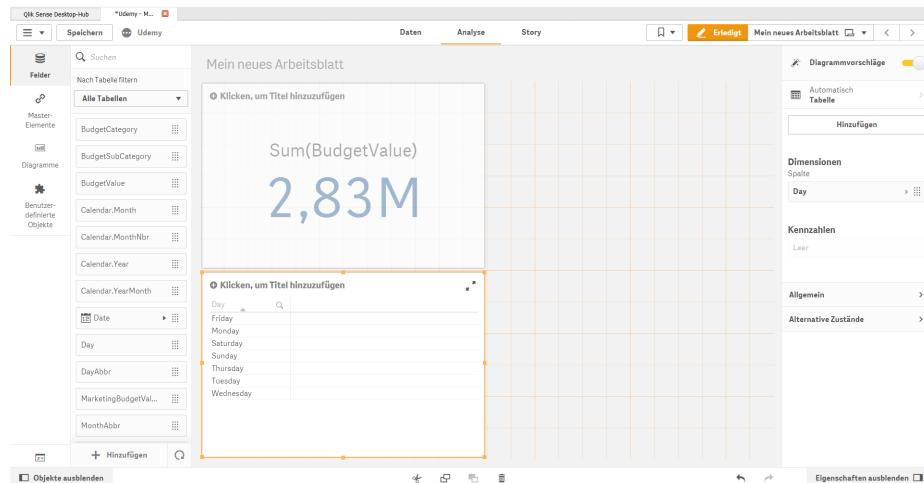


Abbildung 208:

kann ein neues Arbeitsblatt angelegt werden.

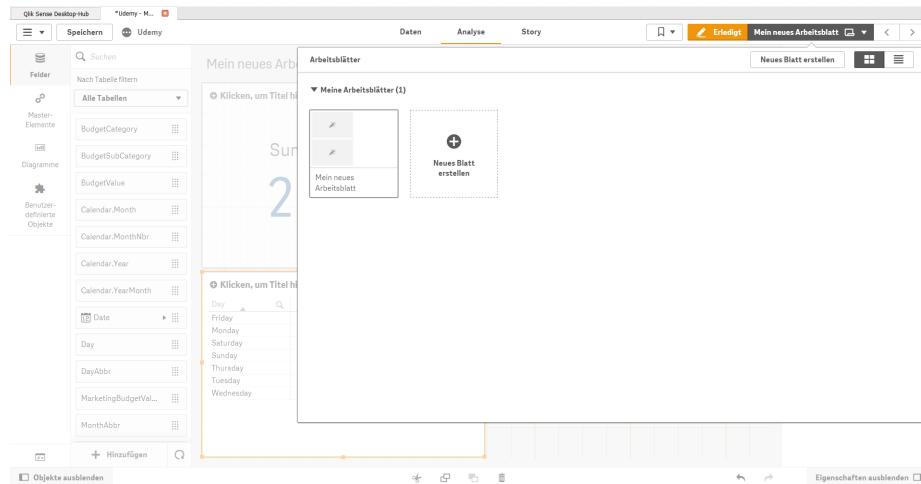


Abbildung 209:

Die Qlik Sense speichert die Daten für eine App mit allen Informationen zu den Arbeitsblättern in Qlik Sense Desktop App File (.qvf) Dateien. In Qlik Sense Desktop ist der vordefinierte Pfad:

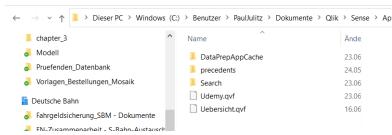


Abbildung 210:

### 2.1.3 Selection

Die QIX Engine agiert über Associationen. Tabellen werden über Schlüssel verbunden. Filter werden auf das gesamte Datenmodell angewandt. Es gibt verschiedene Farben, die die Auswahl markieren.

Alle möglichen Auswahlen erscheinen weiß (Possible). Wir ein oder mehrere Auswahlen getroffen, so erscheinen diese in grün (Selected). Alle anderen Möglichkeiten erscheinen in einem leichten grau (Alternative). Durch das Assoziativ Modell werden alle andern Filter angepasst. Auch hier erscheinen alle möglichen Auswahlen in weiß (Possible). Die Möglichkeiten die nicht ausgewählt werden können, erscheinen in einem dunklen grau (Excluded)

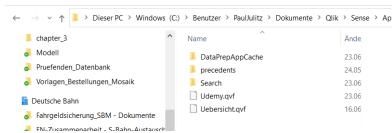


Abbildung 211:

## 2.2 Qlik Key

### 2.2.1 Composite Key

Die Assoziationen zwischen Tabellen kann auch manuell angestoßen werden. Die drei Begrifflichkeit sind Synonyme für den gleichen Sachverhalt.

Am folgenden Beispiel kann gezeigt werden, wie ein Composite Key verwendet wird, eine Kombination von Feldern aus einer Tabelle mit einem anderen Feld aus einer anderen Tabelle verbunden werden kann. 1) Zu erst werden die beiden Tabellen geladen.

```

LOAD
  id,
  first_name,
  last_name,
  email,
  gender,
  salary,
  Company
FROM [lib://Qlik-Daten/smart_compound.xlsx]
(Excel, embedded labels, table is Employee);

LOAD
  StudentName,
  CourseDescription,
  CourseDate,
  CourseGrade
FROM [lib://Qlik-Daten/smart_compound.xlsx]
(Excel, embedded labels, table is Course);

```

Abbildung 212:

2) Im Datenmanager kann keine Verbindung automatisch erzeugt werden, da keine Felder übereinstimmen.

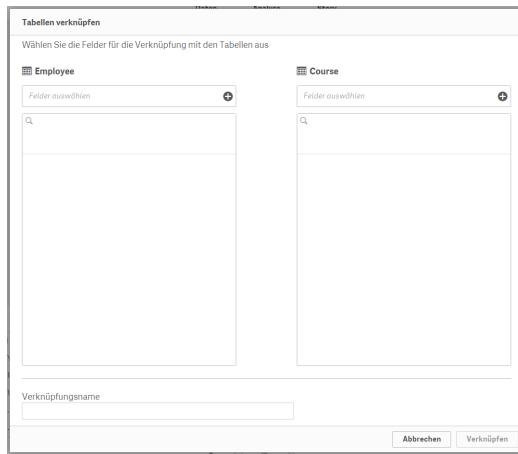


Abbildung 213:

3) Ein Composite Key kann jetzt direkt eingegeben werden. Dabei wird mit Hilfe eines Leerzeichen der [Vorname] und [Nachname] in der Employee Tabelle mit dem [StudentName] in der Course Tabelle verbunden.

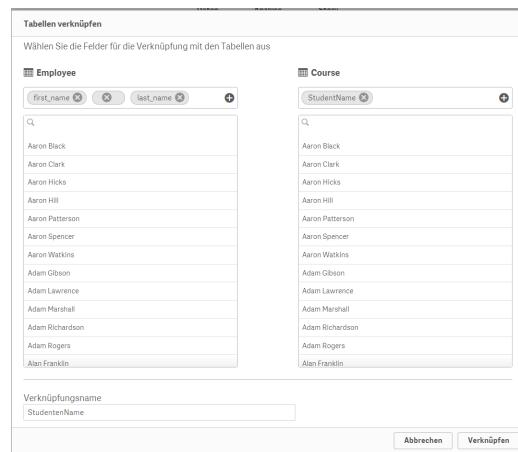


Abbildung 214:

4) Im Datenmanager wird die Verbindung jetzt rot angezeigt.

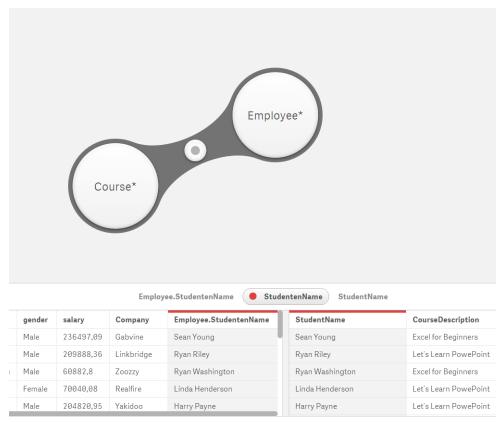


Abbildung 215:

### 5) Sheet Composite Key

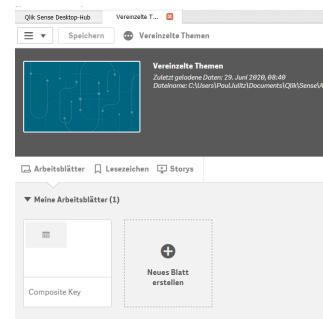


Abbildung 216:

Die App hat jetzt alle relevanten Daten geladen. Als nächstes kann ein Datenblatt angelegt werden. Dabei beinhaltet die App das Datenmodell und die Sheets geben Auskunft über die Visualisierungen.

6) Als Objekt der Wahl wird in dem Beispiel eine Tabelle eingefügt.

The screenshot shows the Qlik Sense desktop application. On the left, there is a sidebar with various visualization options. In the center, there is a 'Composite Key' editor with a search bar for 'StudentenName', 'gender', and 'CourseDescription'. To the right, there is a 'Eigenschaften des Arbeitsblatts' (Properties of the Worksheet) panel with settings for 'Titel' (Title), 'Titelformat' (Title format), 'Beschreibung' (Description), 'Gitternetzabstand' (Grid distance), 'Arbeitsblattgröße' (Worksheet size), and 'Layout für kleinen Bildschirm' (Layout for small screen). A preview window shows a small grid.

Abbildung 217:

7) Anhand der Verknüpfung [StudentenName] kann das Feld [CourseDescription] von der Tabelle [Course] mit [Gender] von der Tabelle [Compony] in Verbindung gesetzt werden.

Abbildung 218:

Ein Composite Key kann auch direkt über den Daten Editor erstellt werden. Dabei wird die Spalte direkt über das `Load` Statement abgeändert.

```
[Employee_temp_c3d2d039-a067-47b3-b5d4-1d66c025];
LOAD
[id],
[first_name],
[last_name],
[email],
[gender],
[city],
[company],
[first_name] + ' ' + [last_name] AS [StudentenName]
RESIDENT [Employee];
DROP TABLE [Employee];

[Course_temp_1ddfd4df-8f9b-7074-75cf-187c3488];
LOAD
[StudentName] AS [StudentenName],
[CourseDescription],
[CourseDate],
[CourseGrade]
RESIDENT [Course];
DROP TABLE [Course];
```

Abbildung 219:

Die Spalte [First Name] und [Last Name] wir verbunden und über `as` als neue Spalte definiert.

## 2.2.2 Synthetic Key

Ein synthetischer Schlüssel wird erstellt, wenn zwei oder mehrere Felder in einer oder mehreren Tabellen gleich sind.

Abbildung 220:

Im gewählten Beispiel sind die Felder `[Region]` und `[Region Code]` in zwei Tabellen gleich. Qlik Sense erstellt eine Synthetic Tabelle `$Syn 1 Tabele`. Diese enthält das Feld `$ Syn1`, und `[Region]` und `[Region Code]`.

Pointer	Value
000	aa
001	asdf

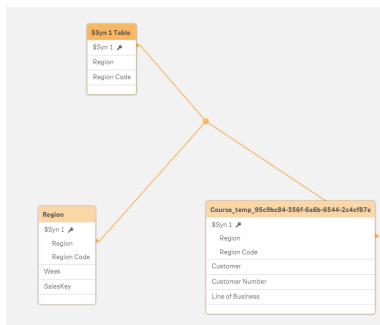


Abbildung 221:

Gleiches wird auch in den anderen Tabellen angelegt, obwohl diese nur über den synthetischen Schlüssel verbunden sind.

Sind Tabellen identisch, so werden diese verbunden und erscheinen im Modell als eine Tabelle.

### 2.2.3 Speicherung

Damit Qlik effizient und schnell Berechnungen durchführen kann, werden alle geladenen Tabellen in zwei Typen gespeichert. Die eigentlichen Datentabellen und die Symboltabellen. Jeder Tabellenspalte wird eine Symboltabelle zugeordnet. In diesen gibt es zwei Spalten, die Pointer und Daten Spalte. Die Daten Spalte greift alle unterschiedlichen Werte einer Spalte auf. Die Anzahl der Zeilen bestimmt die Größe des Pointers. Dieser wird binär dargestellt. Qlik geht hier platzsparend vor. Der Pointer ist nämlich nur so lang - verbraucht nur so viele Stelle, wie benötigt wird. Gibt es vier unterschiedliche Werte, so ist der Pointer drei Stellen lang - drei Bit

## 3 Python Connection

```

1  \item Maschine Learning with Qlik: https://youtu.be/7E944kz1l5s
2  \item https://developer.qlik.com/garden/5af5217ab2606a3c2c1f4d1d
3  \item

```

# **Teil VII**

# **Python Programming**

# 1 Object-Oriented Design

## 1.1 Objekt Orientierte Grundlagen

### 1.1.1 Beispiel

Die Grundlagen für OOP werden in dem Kurs besprochen. Es soll dabei darum gehen, die Logik hinter den Prinzipien von OOP zu verstehen.

Es wird versucht die Vorteile zu beschreiben und die Abwegung zwischen verschiedenen Konzepten wie Funktionales-Programmierung aufzuzeigen.

Ein kurzes Einführungsbeispiel ist das Kuchen-Backen. Wir können das Backen eines Kuchen als Abfolge verschiedener Prozesse beschreiben, welches zu dem Resultat, einen fertiger Kuchen, führt. Beschreiben wir jedoch den Backvorgang mit OOP, dann werden die verschiedenen Gerätschaften beschrieben, mit dem was man mit ihnen tun kann.

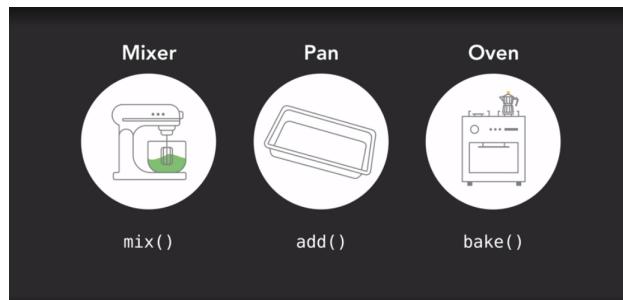


Abbildung 222:

### 1.1.2 Objekte

Es wird zuerst ein **Objekt** beschrieben. Diese kann verschiedenen **Attribute** besitzen. Oft wird für Attribute auch andere Begrifflichkeiten verwendet:

$$\text{Attribute} \cong \text{Eigenschaften, Status, Variable, Daten etc}^6 \quad (1.1)$$

Im Abstraktenraum sprechen wir von Objekten die Attribute besitzen und mit denen man **Methoden** vollziehen kann. Auch hier werden oft verschiedenen Begrifflichkeiten verwendet:

$$\text{Methoden} \cong \text{Operationen, Funktionen etc.} \quad (1.2)$$

Eine Methode ist dabei nur verwenbar von einem Objekt der zugehörigen Klasse, in welcher die Mehtode definiert wurde. Zwei Objekte vom Typ Konto können wir folgt aussehen:

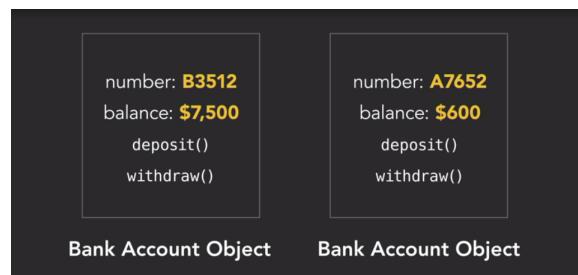


Abbildung 223:

Auf das Objekte Bankkonto mit der Nummer *B3512* können wir die Operation *deposit()* ausführen. Dies gilt für alle Bankkonten.

### 1.1.3 Klassen

Die Blaupausen für die Objekte sind die **Klassen**. Eine Klasse enthält alle Bestandteile für ein Objekte. Ein Objekt wird erst durch eine Klasse erstellt, somit kann eine Klasse mehrere Objekte besitzen. Die Klasse besitzt einen Namen:

$$\text{Typ} \cong \text{Name} \quad (1.3)$$

und Attribute (Properties) sowie Methoden (Operations)



Eine Klasse wird meist in kompakterer Schreibweise geschrieben:

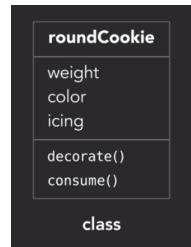


Abbildung 224:

Wird ein Objekt erzeugt, so nennt man den Vorgang **Instanzierung**. Ein Objekt wird auch manchmal **Instanz** genannt.



Abbildung 225:

Die verschiedenen Sprachen habe bereit verschiedene Classen definiert. So zum Beispiel ist ein String, Boolean oder andere Datentypen Classen in den verschiedenen Sprachen. Diese Klassen werden zu **Bibliotheken** zusammengefasst.

## Frameworks and Libraries

- Java Class Library
- .NET Framework BCL
- C++ Standard Library
- Ruby Standard Library
- Python Standard Library

Abbildung 226:

### 1.1.4 Abstraction, Encapsulation, Inheritance, Polymorphismen

**Abstraction** Zum Beschreiben einer Klasse werden nur die wichtigsten Informationen benötigt und die in der einfachsten Form.

**Encapsulation** Wir ein Objekt erstellt (instanziert), dann werden die Attribute definiert. Wollen wir aber auf die Attribute zugreifen, so wird der Zugang nur mit Hilfe von Methoden erlaubt. Das bedeutet, soll das Attribut geändert werden, so muss die dafür definierte Methode aufgerufen werden.

In dem Beispiel wollen wir ein Keks aus dem Objekt Keksdose entnehmen, dafür wird jedoch die Methoden `requestCookie()`:

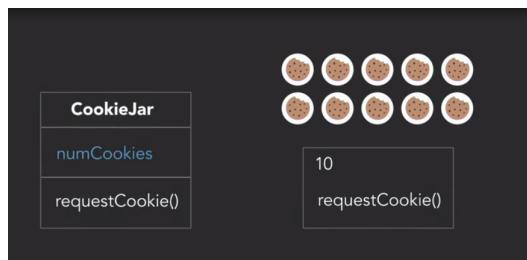


Abbildung 227:

Im Code wird dies durch den Ausdruck **private** hervorgerufen.

```
1   class Account {  
2       private int account_number;  
3       private int account_balance;  
4  
5       public void showData() {  
6           //code to show data  
7       }  
8  
9       public void deposit(int a) {  
10          if (a < 0) {  
11              //show error  
12          } else  
13              account_balance = account_balance + a;  
14      }  
15  }
```

Eine Variable kann somit nicht außerhalb der Klasse aufgerufen oder verändert werden. Die Variable ist nur in der Klasse zugänglich. Wollten wir zum Beispiel einen ein Konto aufrufen und einen negativen Betrag von 100 Euro eintrag, würde dies nicht gehen.

```
1   Account a = new Account();  
2   a.account_number = -100;
```

Diese Operation könnte somit nicht ausgeführt werden, weil die Variable nicht außerhalb der Klasse verändert werden kann, sondern nur folgendes geht:

---

```

1 Account a = new Account();
2 a.deposit(-100);

```

---

Der gesamte Code kann so als Kapsel verstanden werden.

**Inheritance** Die **Vererbung** von Klassen-Komponenten ist eine sinnvolle Funktion, um weitere Abstraktions-ebenen zu erlauben:

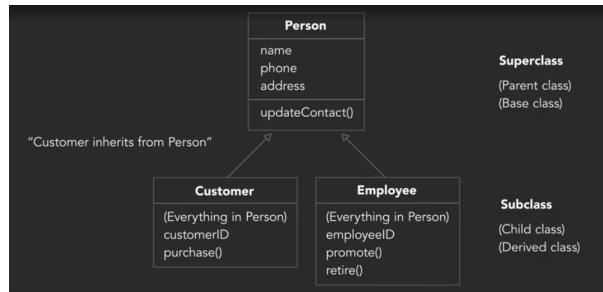


Abbildung 228:

Eine Klasse kann auch von mehreren **Superklassen** eine Vererbung erhalten:

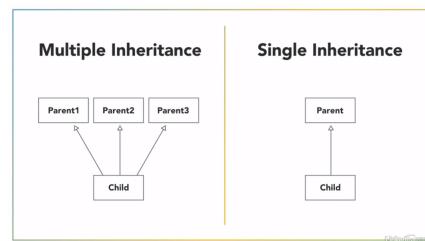


Abbildung 229:

---

```

1 Class A(...);      //Base class
2 Class B : public A(...); //B derived from A
3 Class C : public B(...); //C derived from B

```

---

**Polymorphism** Hier wird zwischen **Statischer** und **Dynamischer Polymorphism** unterschied. Wie diese direkt unterschieden werden, hängt von vom Complierprozess ab. Zu jeder der beiden Kategorien lässt sich aber besondere Methode definieren:

- Method Overload: Mit Hilfe dieses Prinzip können verschiedene Methoden mit der gleichen Signatur oder Namen definiert werden. Es muss aber möglich sein, Anhand der Inputanzahl oder Typen die Methoden zu differenzieren.

---

```

1 void sum (int a , int b);
2 void sum (int a , int b, int c);
3 void sum (float a, double b);

```

---

- Method Overriding: Soll eine Klasse abgeleitet von einer Superklasse werden, so kann es sein, das nicht alle Methoden genau so verwendet werden sollten. Damit aber nicht die ganze Klasse neu aufgesetzt werden muss, schreibt man nur die Methode um, die anzupassen ist.

---

```

1 class X{
2     public int sum(){
3         // some code

```

---

```

4
5
6
7     class Y extends X{
8         public int sum(){
9             //overridden method
10            //signature is same
11        }
12    }

```

---

## 1.2 Class Diagramm

### 1.2.1 Creating Class Diagramm

In dem UML Diagramm definieren wir die Variablen/ Attribute zuerst. Hinter dem Doppelpunkt definieren wir den Typ der Variablen und mit mit dem Gleichheitszeichen definieren wir den *Defaultvalue*.

ClassName
callSign: String ="Excellent"
shieldActive: Boolean
position: Coordinate
Methoden

Für die Methoden werden meistens Verben in ersten Teil der Bezeichnung für eine Methode verwendet.

Spaceship
callSign: String ="Excellent"
shieldActive: Boolean
position: Coordinate
getPosition(): Coordinate
setPosition(integer): Coordinate
move(): Coordinate
lowerShield(integer, string): Boolean

In der Klammer werden die Inputvariablen definiert, die die Methode benötigt. Weiter wird auch hier der Wiergabetyp, hinter dem Doppelpunkt, definiert.

Um **Private** Klassifizierung in der Klasse zu definieren, werden **Plus** und **Minus** genutzt. Ein Plus signalisiert, dass das Attribut oder Methode **public** ist und ein Minus, dass das Attribut oder die Methode **private** ist.

Spaceship
+ callSign: String ="Excellent"
- shieldActive: Boolean
- position: Coordinate
+ getPosition(): Coordinate
- setPosition(integer): Coordinate
+ move(): Coordinate
+ lowerShield(integer, string): Boolean

### 1.2.2 Konstruktor

Ein **Konstruktor** ist eine Methode, welche beim erzeugen der Klasse aufgerufen wird. Wollen wir ein Objekt erzeugen (instanzieren), verwenden wir

---

```

1   class X{
2       public int sum(){
3           // some code
4       }
5   }
6
7   X myX = new X();

```

---

Damit nicht die Defaultvalues für die Attribute verwendet werden, sondern wie für *callSign = "Excellent"*, wird der Konstruktor verwendet.

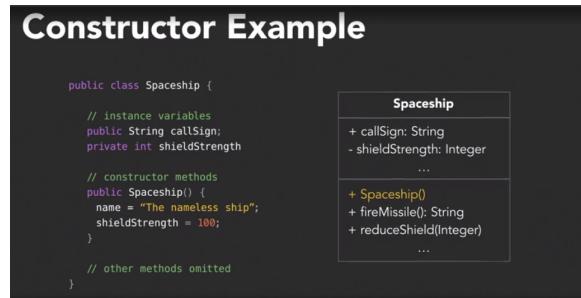


Abbildung 230:

Hinweis: Es soll

---

```

1   public Spaceship() {
2       callSign = "The nameless ship";
3       NOT name
4       shieldStrength = 100;
5   }

```

---

heißen. Es können durch die Fähigkeit von *Overloading* mehrere Konstruktoren erstellt werden. Diese müssen sich jedoch in ihren Inputvariablen unterscheiden lassen:

---

```

1   public class Spaceship {
2       // instance variables
3       public String callSign;
4       private int shieldStrength;

5       // konstruktor
6       public Spaceship() {
7           callSign = "The nameless ship";
8           shieldStrength = 100;
9       }

10      // overloading konstruktor
11      public Spaceship(String name) {
12          callSign = name
13          shieldStrength = 200;
14

15          // other methods omitted
16      }

```

---

Instanziert man das Objekt, so folgt, dass man verschiedene Konstruktoren aufrufen kann.

---

```

1 Spaceship yourShip = new Spaceship(); // Erzeugt das "Default" Objekt
2 Spaceship myShip = new Spaceship("Enterprise"); // Ruft den Overload Konstruktor auf.

```

---

### 1.2.3 Destruktor

Sollte ein Objekt gelöscht werden, so verwendet man den Destruktor.

#### 1.2.4 Static / Instanz Variable/ Methode

Es gibt zwei Arten von Variable, die **Instanz Variablen** und die **Statischen Variablen**, in einer Klasse.

**Instanz Variable** Dies Form der Variable wird für jedes Objekt in einer Klasse angelegt. Erzeugen wir 3 Objekte in einer Klasse, so erzeugen wir auch 3 *Instanzen Variablen*, wenn diese in der Klasse angelegt wurden.

```
1  public class Spaceship {  
2      // Instanz Variable  
3      public String callSign;  
4      private int shieldStrength;  
5      // [...]  
6  }  
7  
8  // Create two objects  
9  Spaceship ship1 = new Spaceship();  
10 Spaceship ship2 = new Spaceship();
```

Soll die zugängliche Variable *callSign* geändert werden, so wird das Objekt und dann die *Instanz Variable* aufgerufen:

```
1  ship1.callSign = "Excalibur";
```

**Static Variable** Die Form der Variablen ist dann nötig, wenn Variablen für mehrere Objekte definiert werden sollen, diese aber nicht für jedes einzelne Objekt geändert werden können. Dies sind mit Globalen Variablen vergleichbar, nur dass sie nicht außerhalb der Klasse existieren. Ändert man eine *statische Variable*, so ändert man den Wert für jedes Objekt der zugehörigen Klasse.

```
1  public class Spaceship {  
2  // Static variables  
3  public static float toughness;  
4  // [...]
```

Soll der Wert der zugänglichen Variable, *toughness* geändert werden, wir die Klassen NICHT ein bestimmtes Objekt aufgerufen.

```
1  Spaceship.toughness = 0.85; // In jeder Klasse wird der Wert angepasst.
```

**Static Method** Aufbauend auf dem Prinzip von *Statischen Variablen* können **Static Methods** nur von der Klasse aufgerufen werden. Das Grundprinzip ist, dass auch Statische Variablen nur über Statische Methoden aufgerufen und verändert werden. Ein Konstruktor kann so, zwar eine Initialisierung der Instanzvariablen erzeugen, aber für weitere Änderungen ist es ratsam, die Methoden zu nutzen, um Variablen zu ändern.

```
1  public class Spaceship {  
2  // Static variables  
3  public static float toughness;  
4  // [...]  
5  public static increaseDifficulty(float t){  
6      toughness += t  
7  }
```

Eine *Static Method* kann nur eine *Static Variable* verändern und vom *Objekt* aufgerufen werden.

```
1  Spaceship.increaseDifficulty(0.2); // In jeder Klasse wird der Wert um 0,2 erhöht,  
2  // wenn die Initialisierung von 0.8 erfolgreich war.
```

**Static Notierung** In dem UML Diagramm werden statische Objekte mit einem Unterschrich ausgewiesen.

Spaceship
<pre>+ callSign: String ="Excellent" - shieldActive: Boolean - position: Coordinate + toughness: float</pre>
<pre>+ getPosition(): Coordinate - setPosition(integer): Coordinate + move(): Coordinate + lowerShield(integer, string): Boolean + increaseDifficulty(float)</pre>

### 1.3 Inheritance and Composition

Es ist nicht ratsam, nach *Vererbung* zu suchen und verschiedenen Ebenen von Beziehungen zu erstellen. **Inheritance** "will reveal is self". **Overriding** ist ein essentieller Teil der Vererbung. Ebenso ist es möglich weitere Methoden einer Klasse zuzufügen.

In C++ wird der Doppelpunkt verwendet, um eine Klasse einer Superklasse zuzuweisen:

---

```
1 public class CargoShuttle : Spaceship {
2     \\\ CargoShuttle inherent all from Spaceship
3 }
```

---

Beim Beschreiben der hergeleiteten Klasse, wird von Sprache zu Sprache unterschied, welche Syntax zu verwenden ist. In Java wird der Prefix **super**, in C# **base**, verwendet. **Hinweis:** In einer Klasse, kann auch eine Methode beschrieben werden, die ein Objekt der Klasse intanziiert.

Im folgenden Beispiel

---

```
1 class left {
2     public:
3         void foo();
4 };
5
6 class right {
7     public:
8         void foo();
9 };
10
11 class bottom : public left, public right {
12     public:
13         void foo()
14     {
15         //base::foo(); // ambiguous
16         left::foo();
17         right::foo();
18
19         // and when foo() is not called for 'this':
20         bottom b;
21         b.left::foo(); // calls b.foo() from 'left'
22         b.right::foo(); // call b.foo() from 'right'
23     }
24 };
```

---

wird der Synatik :: benötigt, um auf die Superklasse zu referieren. In C++ kann auch auf zwei Superklassen referiert werden. Andere Klassen verweisen, in der Regel nur auf eine Superklasse.

#### 1.3.1 Abstract class/ Methode

Eine **Abstrakte Klasse** ist eine Form der Klasse, welche eine Initialisierung zu lässt. Der Zweck dieser Form ist, dass die Klasse als Super Klasse dient. Dabei kann eine Abstrakte Klasse auch selber eine Subklasse sein, jedoch wird von dieser Ebenen mindestens eine weitere Ebene erschlossen.

---

```

1 public abstract class GraphicObject {
2     // declare variables
3     // declare nonabstract methods
4     abstract void draw();
5 }
```

---

Eine **Abstrakte Methode** wird ohne Implementation geschrieben. D. h., hinter (...) wird nur ; gesetzt. In der Subklasse wird diese Methode dann implementiert: *Override*. Wird die Methode nicht implementiert, so muss die Sub Klasse auch abstrakt sein.

Die Methoden die zu überschreiben sind, werden dann, im Normalfall, als abstrakt dekliniert. Selbst, wenn diese nicht notwendig ist, hilft es, die Intention in dem programmierten Text besser zu vermitteln.

---

```

1 abstract class GraphicObject {
2     int x, y;
3     ...
4     void moveTo(int newX, int newY) {
5         ...
6     }
7     abstract void draw();
8     abstract void resize();
9 }
10
11 class Circle extends GraphicObject {
12     void draw() {
13         // Overriding
14     }
15     void resize() {
16         // Overriding
17     }
18 }
19 class Rectangle extends GraphicObject {
20     void draw() {
21         // Overriding
22     }
23     void resize() {
24         // Overriding
25     }
26 }
```

---

Schlussfolgerung: Eine **abstract class** ist nur zur Vererbung angelegt. Objekte können auch nicht über diese Klasse instantiiert werden.

### 1.3.2 Interfaces

Es gibt einige Unterschied zwischen *Abstract class* und **Interface**. Auf dem ersten Blick wirken beide jedoch sehr ähnlich. Das Interface wirkt auch wie extreme Variante der Abstrakten Klasse. Die Zielsetzung mit der man die Klassen erstellt, sind aber andere.

**Abstrakte Klasse** wird verwendet, wenn Klassen verschiedenen Beziehungen miteinander haben und eine Hierarchische Struktur besitzen. Bei C++ können Klassen von mehreren Superklassen die Eigenschaften erhalten, bei den meisten anderen Sprachen jedoch nicht - Hierarchische Struktur.

**Interface** wird verwendet, wenn Fähigkeiten übertragen werden sollen. Die Fähigkeit *move()* und *draw()* lässt sich besser jeweiligen Interface Klassen zuordnen, als eine hierarchische Abstrakte Klasse zu erzeugen.

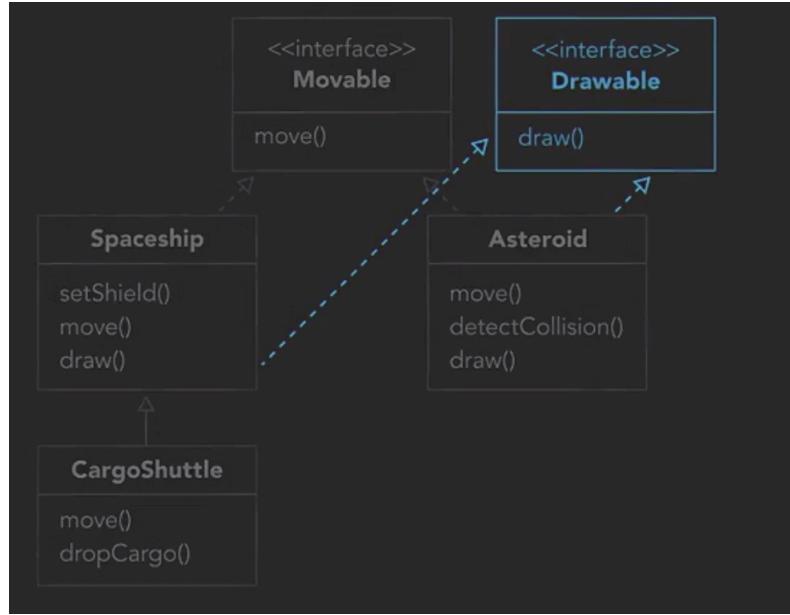


Abbildung 231:

In dem UML Diagramm werden Interface mit gepunkteten Linien gezeichnet.

#### Eigenschaften

- Alle Variablen und Methoden sind `public, static`. ≠ In einer Abstrakten Klasse kann eine Methode abstrakt sein.

---

```

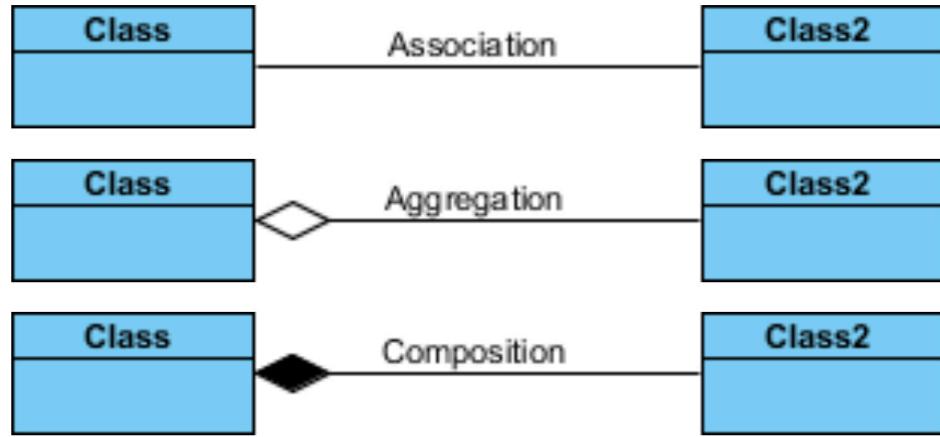
1     interface Moveable {
2         // methode signate
3         void move(int x, int y);
4
5         class X implements Moveable {
6             // Implementation for interface method
7             public void move(int x, int y){
8                 // Implementation code
9             }
10            // Additional methodes
11        }
  
```

---

- Alle Methoden besitzen keine Implementation.
- Keine Methode muss mit `public` oder `private` deklariert werden.

#### 1.3.3 Aggregation / Composition

Diese beiden Relationen beziehen sich nicht so sehr auf eine besondere Syntax, sondern auf die Gestaltung des Codes.



The figure below shows a generalization.

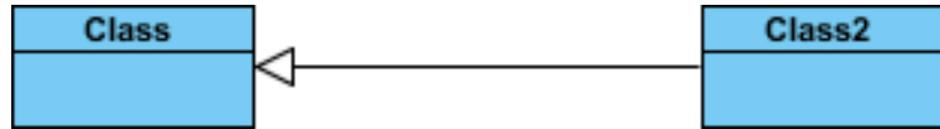


Abbildung 232:

- Die Relation **Aggregation** wird so verstanden, dass ein Subobjekt, in einer Gruppenabhängigkeit einbezogen wird. Löst sich jedoch auf, bleibt das erzeugt Objekt erhalten.
- Die Realation **Composition** wird so verstanden, dass ein Subobjekt, in einer Gruppenabhängigkeit einbezogen wird. Löst sich jedoch auf, so hört das erzeugt Objekt auf zu existieren.

Zum Beispiel: Ein Spaceship kann teil einer Field sein. Verschwindet die Field, so verschwindet das Spaceship nicht zwangsläufigerweise.

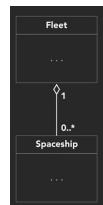


Abbildung 233:

Zum Beispiel: Ein Auto besteht aus einem Motor und Reifen. Wenn das Auto nicht mehr ist, wird der Motor nicht mehr benötigt.

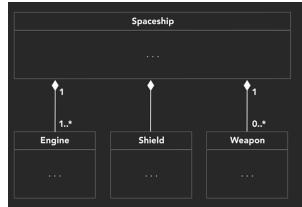


Abbildung 234:

**Hinweis** Mit Hilfe der UML Beziehungsanalyse soll die Programmiererin erkennen, welche Objekte mit einem *Destruktor* gelöscht werden müssen und welche nicht.

#### 1.3.4 Zusammenfassung Relationen



Abbildung 235:

#### 1.4 Software Development

Object-Oriented Languages						
Language	Inheritance	Call to Super	Typing	Interfaces	Abstract Classes	
Java	single	super	static	Yes	Yes	
C#	single	base	static	Yes	Yes	
Python	multiple	super	dynamic	Abstract Class	Yes	
Swift	single	super	static	Protocols	No	
C++	multiple	name of class::	static	Abstract Class	Yes	
Ruby	mixins	super	dynamic	n/a	n/a	
JavaScript	prototypes	n/a	dynamic	n/a	n/a	

Abbildung 236:

Die verschiedenen Sprachen können unterschiedlich mit den Prinzipien, die wir besprochen haben, umgehen.

- Inheritance: Kann eine Klasse vererbt werden. Bei *C ++* können auch mehrere Klassen einer Klasse vererbt werden.
- Call to super: Wie lautet die Syntax, um eine Vererbung anzuzeigen.
- Typing: ? Das Thema ist noch unklar. Stichphrase: Wie wird kompiliert? Es gibt hierzu Vor- und Nachteil. Zum Beispiel: Bei static languages müssen alle Variablen vom Typ definiert werden, da sie vor dem Ausführen vorliegen. Bei dynamischen Sprachen wird der Typ bei Ausführung erkannt. Dies kann jedoch zu Problemen führen, falls die Erkennung nicht richtig erfolgt. Das Schreiben des Codes ist aber einfacher.

## 2 Introduction in Python

### 2.1 Running Python from VS Code

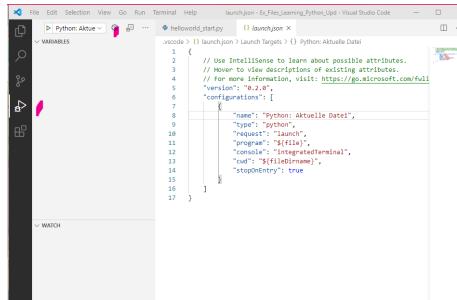
#### 2.1.1 Befehle

**Python –version** Zeigt die aktuelle Version an

**py** In PowerShell/ Terminal/ cmd wird Python über **py** gestartet.

### 2.1.2 Launch Json File

Unter Run-Debug/Settings werden die globalen Debug Lauch Konfigurationen eingelesen.



The screenshot shows the Visual Studio Code interface with the 'Launch.json' file open in the editor. The code content is as follows:

```
1 // Use IntelliSense to learn about possible attributes.
2 // Hover to view descriptions of existing attributes.
3 // For more information, visit: https://go.microsoft.com/fwlink/?LinkId=723263
4 "version": "0.2.0",
5 "configurations": [
6     {
7         "name": "Python: aktuelle Datei",
8         "type": "python",
9         "request": "launch",
10        "program": "${file}",
11        "console": "integratedTerminal",
12        "cwd": "${fileDirname}",
13        "stopOnEntry": true
14    }
15 ]
16
17 }
```

Abbildung 237:

- `cwd` gibt an wo CurrentWorkingDirectory ist.
- `stopOnEntry` Stop den Vorgang zum Anfang der Datei

### 2.1.3 Internal Console instead of Terminal

Eine verschlankte Anzeige ist, wenn die Console nicht über das Terminal gestartet wird. Eine Einschränkung gibt es, Python kann keine Eingaben erhalten.



The screenshot shows the Visual Studio Code interface with the 'Launch.json' file open in the editor. The code content is as follows:

```
1 // Use IntelliSense to learn about possible attributes.
2 // Hover to view descriptions of existing attributes.
3 // For more information, visit: https://go.microsoft.com/fwlink/?LinkId=723263
4 "version": "0.2.0",
5 "configurations": [
6     {
7         "name": "Python: aktuelle Datei",
8         "type": "python",
9         "request": "launch",
10        "program": "${file}",
11        "console": "internalConsole",
12        "cwd": "${fileDirname}",
13        "stopOnEntry": true
14    }
15 ]
16
17 }
```

- `console integratedTerminal` - Vollumfang
- `console internalConsole` - Verschlankte Darstellung in Debug Console, Kein Input

## 3 Algorithme

## **Teil VIII**

# **Machine Learning and Data Analysis**

# 1 Examples from common libraries

## 1.1 Working with Excel

### 1.1.1 Basic Pandas

Im folgenden werden zwei Pakete benötigt: *Pandas* und *openpyxl*.

```
1 import pandas as pd
2 from openpyxl.workbook import Workbook
```

Panda ist ein freie Daten Analyse Bibliothek. Bei *openpyxl* handelt es sich um eine Bibliothek, welche das Lesen und Schreiben von Excel Dateien ermöglicht.

Pandas nutzt die folgenden Konzepte um Daten zu speichern und zu manipulieren:

**Dataframe** Dies wird genutzt, um 2D Tabellen abzuspeichern.

**Series** Dieses wird genutzt, um 1D Arrays abzuspeichern.

**Panals** Diese wird genutzt, um 3D Daten abzuspeichern.

**pd.read()** Die Funktion *.read* importiert Daten in den Dataframe. Für verschiedenen Dateiformate bietet Pandas unterschiedliche *.read* Funktionen an:

```
The following methods are called when data or markup elements are encountered and they are meant to be
overridden in a subclass. The base class implementations do nothing (except for handle_startendtag()):
HTMLParser.handle_starttag(tag, attrs)
This method is called to handle the start of a tag (e.g. <div id="main">).

The tag argument is the name of the tag converted to lower case. The attrs argument is a list of (name,
value) pairs containing the attributes found inside the tag's <> brackets. The name will be translated to
lower case, and quotes in the value have been removed, and character and entity references have been
replaced.

For instance, for the tag <A href="https://www.cwi.nl/">, this method would be called as
handle_starttag('a', [('href', 'https://www.cwi.nl/')]).

All entity references from html.entities are replaced in the attribute values.

HTMLParser.handle_endtag(tag)
This method is called to handle the end tag of an element (e.g. </div>).

The tag argument is the name of the tag converted to lower case.

HTMLParser.handle_startendtag(tag, attrs)
Similar to handle_starttag(), but called when the parser encounters an XHTML-style empty tag <img
... />. This method may be overridden by subclasses which require this particular lexical information; the
default implementation simply calls handle_starttag() and handle_endtag().

HTMLParser.handle_data(data)
This method is called to process arbitrary data (e.g. text nodes and the content of <script>...</script>
and <style>...</style>).

HTMLParser.handle_entityref(name)
```

Das Paket *xld* wird benötigt, um die *xlsx* Formate einzulesen. Alternativ kann auch *openpyxl* verwendet werden, um die Dateien einzulesen.

```
1 # Import Data to Dataframe
2 df_excel = pd.read_excel("regions.xlsx", engine='openpyxl')
3 # The engine openpyxl was manuell added and installed. Also the Package
4 # xlrd was installed in iOS.
5 df_csv = pd.read_csv("Names.csv")
6 df_text = pd.read_csv("data.txt", delimiter="\t")
```

Beim Darstellen der Inhalte werden die ersten fünf und letzten fünf Zeilen ausgewiesen.

	ID	EGF_Baseline	EGF_Stimulus
1	#		
2	#0	FBgn0029994	-1.25
3	#1	FBgn0037191	-1.05
4	#2	FBgn0036810	2.08
5	#3	FBgn0033320	1.15
6	#4	FBgn0051156	-1.77
7	#...	...	...
8	#11761	FBgn0026136	0.57
9	#11762	FBgn0037356	1.14

```

10 #11763 FBgn0038214      -1.86      -0.67
11 #11764 FBgn0042110      1.49       0.43
12 #11765 FBgn0034202      -1.32      -0.22

```

In ETL Prozess einer csv Datei kann wie folgt aussehen.

```

1 # Extract, Transform and Load
2 df_csv = pd.read_csv("Names.csv", header=None)
3 df_csv.columns = ["Vorname", "Nachname", "Street", "City", "State", "
   Postcode", "Unkown"] # Has to be complete.
4 df_csv.to_excel("modified.xlsx") # same directory

```

**Lambda Function** Lambda ist eine Funktion, welche zu einer anonymen Funktion referiert.

Lambda (Argumgents) : expression (1.1)

Für weiteres siehe iloc Lambda Function.

**set\_index** Mit der Index Funktion kann eine oder mehrere Spalten als Index ausgewählt werden. Ebenso kann verifiziert werden, ob es sich um einen eindeutigen Schlüssel handelt oder nicht.

```

1 df_new = df_csv.set_index("State", verify_integrity=True)
2 ### Output
3 print(df_new)
4 # ValueError: Index has duplicate keys: Index(['NJ', 'SD'], dtype='
   object', name='State')

```

Ohne diese Einschränkung und mit einer anderen Indexspalte wird folgendes wiedergegeben.

```

1 df_new = df_csv.set_index("Postcode")
2 ### Output
3 print(df_new)
4 #
5 #           Vorname Nachname
6 #          Street      City State Income
7 #Postcode
8 #8074           John     Doe      120 jefferson
9 #             st.    Riverside NJ 45000
10 #9119            Jack McGinnis      220 hobo
11 #             Av.      Phila PA 18000
12 #8075            John "Da Man" Repici      120 Jefferson
13 #             St.    Riverside NJ 120000
14 #91234           Stephen Tyler 7452 Terrace "At the Plaza"
15 #             road   SomeTown SD 90000
16 #298            NaN   Blankman
17 #             NaN   SomeTown SD 30000
18 #123            Joan "Danger", Anne Jet      9th, at Terrace
19 #             plc   Desert City CO 68000

```

**str.split** Die Splitfunktion zerlegt eine Spalte in ihre Komponenten. Mit der expand Option wird jede benötigte Abtrennung dargestellt.

```

1 df_new = df_csv.set_index("Postcode")
2 df_new = df_new["Vorname"].str.split(expand=True)
3 #df_new = df_new.Vorname.str.split(expand=True) # Same Result
4 ### Output
5 print(df_new)
#           0         1         2

```

```

7 #Postcode
8 #8074      John      None  None
9 #9119      Jack      None  None
10 #8075     John      "Da  Man"
11 #91234    Stephen   None  None
12 #298       NaN      NaN  NaN
13 #123      Joan  "Danger", Anne

```

Die erste Spalte wird übertragen, wenn dieser direkt einer andern Spalte zugewiesen wird.

```

1 df_new = df_csv.set_index("Postcode")
2 df_new.Vorname = df_new["Vorname"].str.split(expand=True)
3 ### Output
4 print(df_new)
#           Vorname Nachname                      Street
#      City State Income
#Postcode
7 #8074      John      Doe      120 jefferson st.
#      Riverside NJ 45000
8 #9119      Jack  McGinnis      220 hobo Av.
#      Phila PA 18000
9 #8075      John      Repici      120 Jefferson St.
#      Riverside NJ 120000
10 #91234     Stephen Tyler 7452 Terrace "At the Plaza" road
#      SomeTown SD 90000
11 #298       NaN      Blankman      NaN
#      SomeTown SD 30000
12 #123      Joan      Jet      9th, at Terrace plc Desert
#      City CO 68000

```

**.replace** Mit der replace Funktion wird ein Wert durch einen anderen Wert ersetzt. Die Konstante np.nan (Convert a string or number to a floating point number, if possible.) gibt Wahr oder Falsch wieder, wenn ein leerer Eintrag existiert.

```

1 df_new = df_csv.set_index("Postcode")
2 df_new.Vorname = df_new["Vorname"].str.split(expand=True)
3 df_new = df_new.replace(np.nan, "NaN")

```

### 1.1.2 Dataframe Functions

Anhand des Beispiel *Names.csv*.

```

1 #           Vorname Nachname                      Street
#      City State Postcode Unkown
#  # 0      John      Doe      120 jefferson st.
#      Riverside NJ 8074 45000
#  # 1      Jack  McGinnis      220 hobo Av.
#      Phila PA 9119 18000
#  # 2      John "Da Man"      Repici      120 Jefferson St.
#      Riverside NJ 8075 120000
#  # 3      Stephen Tyler 7452 Terrace "At the Plaza" road
#      SomeTown SD 91234 90000
#  # 4      NaN      Blankman      NaN
#      SomeTown SD 298 30000
#  # 5      Joan "Danger", Anne      Jet      9th, at Terrace plc
#      Desert City CO 123 68000

```

## Basics

**.columns()** weißt den Spalten neue Bezeichnungen zu.

**df.[]** selektiert ausgewählte Spalte, z. B.: .["Vorname"] aus.. Mit einem Array an Spalten [[“Vorname”],[“Nachname”]] werden mehrere Spalten ausgelesen. Wenn keine Spalte vorliegt, wird diese erzeugt, wenn ihr ein Wert zugewiesen wird. (.[“Test”]=“Test”)

**df.[][]** selektiert im ersten Bereich die Spalte und im zweiten Bereich die Zeilen.

**df.index** Ließt die Index Vektor aus.

**df.drop()** Spalten können entfernt werden. Der Parameter *inplace* gleich *false* gibt, an ob eine Kopie erstellt werden soll. Wird daraufhin *.print()* angewandt, wird der Dataframe ohne die Spalten angezeigt. Wird der Parameter auf *true* gesetzt, so werden die Daten gelöscht.

```
1 # Drop
2 df_new = df_csv.drop(columns=["Postcode"])
3 df_new = df_new.groupby(["State"]).mean()
4 #Income
5 #State
6 # NJ      82500
7 # PA      18000
8 # SD      30000
9 #CO      68000
10 #SD     90000
11 #>>>
```

**df.head()** Shows a preview

**df.groupby()** Die Spalte, welche gruppiert werden soll, wird hier angegeben.

```
1 ### Group by
2 df_names_modifield = df_names_modifield.groupby(["State"])
3 # Object at 0x...
```

Ohne eine spezielle Operation bleibt es bei einem Objekt. Wird die Funktion *.mean()* hinzugefügt, wird diese auf jede Spalte mit Integer angewendet.

```
1 ### Group by
2 df_names_modifield = df_names_modifield.groupby(["State"]).mean()
3 #           Postcode  Income  Tax Percentage  Tax Amount
4 # State
5 # NJ      8074.5  82500.0      0.25      15000.0
6 # PA      9119.0  18000.0      0.25        0.0
7 # SD      298.0   30000.0      0.25        0.0
8 # CO      123.0   68000.0      0.25      17000.0
9 # SD     91234.0  90000.0      0.25      22500.0
```

**df.groupby().mean()** berechnet den Durchschnitt einer Spalte.

```
1 ### Group by
2 df_names_modifield = df_names_modifield.groupby(["State"]).mean().
3             sort_values("Tax Amount")
4 #           Postcode  Income  Tax Percentage  Tax Amount
5 # State
```

```

5 # PA      9119.0 18000.0      0.25      0.0
6 # SD      298.0 30000.0      0.25      0.0
7 # NJ     8074.5 82500.0      0.25  15000.0
8 # CO     123.0 68000.0      0.25  17000.0
9 # SD    91234.0 90000.0      0.25  22500.0

```

Wie oben gezeigt, kann die Spalte [Postcode] auch entfernt werden.

**df.apply** Apply() bietet an, dass Funktionen auf den Dataframe angewandt werden. Diese können weiter auf Bereiche (Achsen) spezifiziert werden. Mit der Lambda Funktion kann eine Funktion für jeden Eintrag definiert werden. Dabei ist eine genauer Definition nach den Achsen nicht relevant.

```

1 mydict = [{ 'a': 1, 'b': 2, 'c': 3, 'd': 4},
2 {'a': 100, 'b': 200, 'c': 300, 'd': 400},
3 {'a': 1000, 'b': 2000, 'c': 3000, 'd': 4000 }]
4 df_num = pd.DataFrame(mydict)
5
6 # Select
7 df_new = df_num.apply(lambda x: x+2)
#df_new = df_num.apply(lambda x: x + 2, axis=0) # Same Result
#df_new = df_num.apply(lambda x: x + 2, axis=1) # Same Result
10
11 # Output
12 print(df_new)
13 #      a      b      c      d
14 #0      3      4      5      6
15 #1    102    202    302    402
16 #2   1002   2002   3002   4002
17 #>>>

```

Mit *axis* 0 oder 1 wird die Funktion entlang der Spalten oder Zeilen durchgeführt. Dies Mit 0 wird der definierte Funktionsausdruck entlang aller Zeile für jede Spalte durchgeführt. Mit 1 wird für jede Zeile über alle Spalten gerechnet.

Für eine Auswahl an Funktion kann das Paket **numpy** geladen werden.

Aggregationen können entlang einer Achse durchgeführt werden. Zurückgegeben wird ein Array. Als default wird die Achse 0 ausgewählt.

```

1 # Select
2 df_new = df_num.apply(np.sum)
3 ### Output
4 print(df_new)
5 #a    1101
6 #b    2202
7 #c    3303
8 #d    4404

```

Soll entlang einer Zeile die Funktion angewandt werden, wird Achse 1 ausgewählt.

```

1 # Select
2 df_new = df_num.apply(np.sum, axis=1)
3 ### Output
4 print(df_new)
5 #0      10
6 #1    1000
7 #2   10000

```

Um Berechnungen an bestimmten Spalten durchzuführen, werden diese an an neue übergeben.

```

1 # Select

```

```

2     df_csv["Tax Percentage"] = df_csv["Income"].apply(lambda x: .15 if 0 <
3         x < 10000 else .30 if 10000 <= x < 30000 else .45)
4 # if-then-else in one line has change to value_when_true if condition
5 #     else value_when_false
6 df_new = df_csv
7 ##### Output
8 print(df_new)
#          Vorname Nachname                               Street
#          City State Postcode Income Tax Percentage
#0        John Doe           120 jefferson st.
#1      Riverside NJ       8074 45000      0.45
#2        Jack McGinnis           220 hobo Av.
#3        Phila PA       9119 18000      0.30
#4        John "Da Man" Repici           120 Jefferson St.
#5      Riverside NJ       8075 120000      0.45
#6        Stephen Tyler 7452 Terrace "At the Plaza" road
#7      SomeTown SD       91234 90000      0.45
#8      Joan "Danger", Anne Jet           9th, at Terrace plc
#9      Desert City CO       123   68000      0.45

```

Eine Berechnung von zwei Spalten wird durch direktes Aufrufen durchgeführt.

```

1 # Select
2 df_csv["Tax Percentage"] = df_csv["Income"].apply(lambda x: .15 if 0 <
3         x < 10000 else .30 if 10000 <= x < 30000 else .45)
4 df_csv["Tax Amount"] = df_csv["Tax Percentage"]*df_csv["Income"]
5 df_new = df_csv
6 ##### Output
7 print(df_new)
#          Vorname Nachname                               Street
#          City State Postcode Income Tax Percentage Tax Amount
#0        John Doe           120 jefferson st.
#1      Riverside NJ       8074 45000      0.45      20250.0
#2        Jack McGinnis           220 hobo Av.
#3        Phila PA       9119 18000      0.30      5400.0
#4        John "Da Man" Repici           120 Jefferson St.
#5      Riverside NJ       8075 120000      0.45      54000.0
#6        Stephen Tyler 7452 Terrace "At the Plaza" road
#7      SomeTown SD       91234 90000      0.45      40500.0
#8        NaN Blankman           NaN
#9      SomeTown SD       298   30000      0.45      13500.0
#10     Joan "Danger", Anne Jet           9th, at Terrace plc
#11    Desert City CO       123   68000      0.45      30600.0

```

**iloc - Index Number Locator** Die Methode `iloc[]` oder kurz `loc` filtert einen Datensatz nach Zeilen und Spalten. Dies Auswahl kann jedoch nur über den jeweiligen zu adressierten Index passieren. Die Berechnungszeit für `iloc` ist schneller als für `loc`, genau aus dem Grund, dass die Zeilen und Spalten schon geordnet sind, und somit leichter zu durchsuchen sind.

**Slicer Object** Im ersten Beispiel werden alle Zeilen (erste Eingabe vor dem Komma) ausgewählt. Der zweite Eintrag gibt die an, dass alle Spalten bis zur ersten ausgewählt werden sollen. Damit werden beide Achsen gefiltert.

```

1 df_new = df_csv.iloc[:, :1]
2 print(df_new)
3 ##### Output
4 #          Vorname

```

```

5      #0          John
6      #1          Jack
7      #2      John "Da Man"
8      #3          Stephen
9      #4          NaN
10     #5 Joan "Danger", Anne
11     #>>>

```

Mit der Änderung auf 3 werden alle Spalten bis einschließlich der dritten Spalte gefiltert.

```

df_new = df_csv.iloc[:, :3]
print(df_new)
### Output
#
#           Vorname ...
#0          John ...
#1          Jack ...
#2      John "Da Man" ...
#3          Stephen ...
#4          NaN ...
#5 Joan "Danger", Anne ...
#
#[6 rows x 3 columns]
#>>>

```

Das Slicer Object kann auch mit einem booleschen Array kombiniert werden. Die Länge des Arrays muss gleich der Länge der zweiten Achse sein.

```

# Select
df_new = df_csv.iloc[:, [True, False, False, False, False, False]]
# Nachname      Street
#0    Doe  120 jefferson st.
#1 McGinnis   220 hobo Av.

```

Ebenso kann auch eine leere Lambda Funktion oder ein Array für die zweite Achse eingelesen werden.

```

# Select
df_new = df_csv.iloc[:, [0, 2]]
# df_new = df_csv.iloc[:, lambda x: [0,2]] # Same result
print(df_new)
### Output
#
#           Vorname ...
#0          John ...
#1          Jack ...
#2      John "Da Man" ...
#3          Stephen ...
#4          NaN ...
#5 Joan "Danger", Anne ...
#
#[6 rows x 3 columns]

```

**Scalar** Wird Scaler als Eingabe übergeben, wird eine Series wiedergegeben und nach Zeilen gefiltert.

```

df_new = df_csv.iloc[1]
print(df_new)
### Output
#
#Vorname      Jack
#Nachname      McGinnis
#Street      220 hobo Av.
#City        Phila
#State         PA
#Postcode      9119

```

```

11 #Income           18000
12 #Name: 1, dtype: object
13 #>>>

```

Mit einem **zweiten Scaler** wird die Spalte (Achse) ausgewählt. Der Eintrag der Zelle wird wiedergegeben.

```

df_new = df_csv.loc[1,1] # Zeilen Index 1 (Zweite Zeile), Spalten Index 1 (Zweite Spalte)
print(df_new)
#McGinnis

```

**Booleen Array** Wird eine logische Operation ( $==$ ,  $<$ ,  $>$ ) auf eine bestimmt Spalte des Dataframes angewendet, so wird eine *Series* zurückgegeben. Die Series hat die gleiche Länge wie die Tabelle selbst.

```

1 # Select
2 df_new = df_csv["State"]=="SD"
3 print(df_new)
4 ### Output
5 0    False
6 1    False
7 2    False
8 3    True
9 4    False
10 5   False
11 Name: State, dtype: bool

```

Damit die Abfrage genutzt werden kann, muss aus der Serie das Array ausgelesen werden.

```

1 # Select
2 ser_new = df_csv["State"]=="SD"
3 #df_new = df_csv.loc[ser_new]
4 print(df_new)
5 ### Output
6 # Vorname Nachname                               Street      City State
7 # Postcode Income
8 #3 Stephen Tyler 7452 Terrace "At the Plaza" road SomeTown SD
9      91234 90000
10 #4      NaN Blankman                           NaN SomeTown SD
11      298 30000

```

Mit `iloc` kann nur ein Array von Boolschen Werten eingelesen werden. Dies Serie muss nur nach dem Array ausgewertet werden.

```

1 # Select
2 ser_new = df_csv["State"]=="SD"
3 #df_new = df_csv.iloc[ser_new.array] # Array gibt ein narray wieder.
4 print(df_new)
5 ### Output
6 # Vorname Nachname                               Street      City State
7 # Postcode Income
8 #3 Stephen Tyler 7452 Terrace "At the Plaza" road SomeTown SD
9      91234 90000
10 #4      NaN Blankman                           NaN SomeTown SD
11      298 30000

```

Alternativ muss ein Array der Länge des Dataframes eingegeben werde.

```

1 # Select
2 df_new = df_csv.loc[[False, False, False, True, False, False]]
3 #df_new = df_csv.iloc[[False, False, False, True, False, False]] # loc
4      bietet die gleiche Eingabemöglichkeit
5 print(df_new)

```

```

5     ### Output
6     #    Vorname Nachname           Street      City State
7     #    Postcode Income
8     #3  Stephen   Tyler   7452 Terrace "At the Plaza" road SomeTown SD
9         91234   90000
10    #4      NaN   Blankman          NaN SomeTown SD
11        298   30000

```

**List of Integers(Index)** Über iloc können auch die Indizes direkt eingegeben werden. Dafür werden sie als Array eingespielt.

```

1  # Select
2  df_new = df_csv.iloc[[0,1,2]]
3  print(df_new)
4  ### Output
5  #    Vorname Nachname           Street      City State
6  #    Postcode Income
7  #0      John   Doe   120 jefferson st. Riverside NJ
8      8074   45000
9  #1      Jack   McGinnis       220 hobo Av.      Phila PA
10     9119   18000
11  #2  John "Da Man"   Repici  120 Jefferson St. Riverside NJ
12     8075   120000

```

Mit dem zweiten Eintrag wird die zweite Achse gefiltert.

```

1  # Select
2  df_new = df_csv.iloc[[0,1,2],[1,2]]
3  print(df_new)
4  ### Output
5  #    Nachname           Street
6  #0      Doe   120 jefferson st.
7  #1  McGinnis       220 hobo Av.
8  #2      Repici  120 Jefferson St.

```

**iloc with Lambda Function** Lambda ist eine Funktion, welche zu einer anonymen Funktion referiert.

Lambda (Argumgents) : expression (1.2)

Eine Lambda Funktion kann ebenso mit iloc verwendet werden. Im folgenden Beispiel wird der Ausdruck Modular 2 auf die Indizes der Zeilen angewandt. Das Argument  $x$  ist dabei der Dataframe selbst. Mit .index werden die Indizes ausgelesen. Jeder gerade Zeilen Index wird ausgewählt.

```

1  # Select
2  df_new = df_csv.iloc[lambda x: x.index % 2 == 0]
3  print(df_new)
4  ### Output
5  #    Vorname Nachname           Street      City State
6  #    Postcode Income
7  #0      John   Doe   120 jefferson st. Riverside NJ
8      8074   45000
9  #2  John "Da Man"   Repici  120 Jefferson St. Riverside NJ
10     8075   120000
11  #4      NaN   Blankman          NaN SomeTown SD
12        298   30000

```

## loc - Index Name Locator

**Slicer Object, Scalar, List of Name Integers** Der Fokus der loc Funktion ist, dass die Indexes eine String Bezeichnungen besitzen. Im folgenden Beispiel wurden die Indizes für Zeilen und Spalten mit Strings näher definiert.

```

1 # Select
2 df_new = df_csv.iloc[:, [0, 2]]
3 df = pd.DataFrame([[1, 2], [4, 5], [7, 8]], 
4 index=['cobra', 'viper', 'sidewinder'],
5 columns=['max_speed', 'shield'])
6
7 df_new = df.loc["cobra"]
8 print(df_new)
9 ### Output
10 #max_speed    1
11 #shield      2
12 #Name: cobra, dtype: int64

1 # Select
2 df = pd.DataFrame([[1, 2], [4, 5], [7, 8]], 
3 index=['cobra', 'viper', 'sidewinder'],
4 columns=['max_speed', 'shield'])
5
6 df_new = df.loc["cobra", "max_speed"]
7 print(df_new)
8 ### Output
9 #1

```

Für den Slicer lässt sich die Logik auch übertragen.

```

1 # Select
2 df = pd.DataFrame([[1, 2], [4, 5], [7, 8]], 
3 index=['cobra', 'viper', 'sidewinder'],
4 columns=['max_speed', 'shield'])
5
6 df_new = df.loc["cobra": "sidewinder"]
7 print(df_new)
8 ### Output
9 #           max_speed   shield
10 #cobra          1          2
11 #viper          4          5
12 #sidewinder     7          8

```

**Booleen Array/ Series** Es gilt das gleich wie bei iloc, bis auf die Ergänzung, dass Abfragen mit einem Serie ebenfalls gefüllt werden kann.

```

1 # Select
2 df_new = df_csv.loc[df_csv["State"]=="SD"]
3 #df_new = df_csv.loc[[False, False, False, True, False, False]] # Same
4     Result
5 print(df_new)
6 #   Vorname Nachname                      Street       City State
7 #   Postcode Income
8 #3  Stephen   Tyler  7452 Terrace "At the Plaza" road SomeTown   SD
9         91234   90000

```

Ergänzend kann auch eine bedingte Abfrage mit einer Spaltenauswahl getroffen werden.

```

1 df = pd.DataFrame([[1, 2], [4, 5], [7, 8]], 
2 index=['cobra', 'viper', 'sidewinder'],

```

```

3     columns=['max_speed', 'shield'])
4
5     df_new = df.loc[df["max_speed"]>0, ["shield"]]
6     print(df_new)
7     ### Output
8     #           shield
9     #cobra          2
10    #viper          5
11    #sidewinder    8

```

**Change specific Values based on condition** Im Abschnitt df.apply wurden Steuersätze eingefügt. Diese Beispiel soll dienen, wie genau Daten geändert werden.

```

1   # Select
2   df_csv["Tax Percentage"] = df_csv["Income"].apply(lambda x: .15 if 0 <
3       x < 10000 else .30 if 10000 <= x < 30000 else .45)
4   df_csv["Tax Amount"] = df_csv["Tax Percentage"]*df_csv["Income"]
5
6   df_new = df_csv.drop(columns=["Street", "City", "State", "Postcode"])
7   df_new.loc[df_new["Tax Percentage"]<=.3, "Tax Amount"] = 0
8   ### Output
9   print(df_new)
#           Vorname Nachname Income Tax Percentage Tax Amount
#0          John    Doe    45000      0.45      20250.0
#1          Jack  McGinnis   18000      0.30        0.0
#2  John "Da Man"   Repici  120000      0.45      54000.0
#3         Stephen    Tyler   90000      0.45      40500.0
#4            NaN  Blankman   30000      0.45      13500.0
#5 Joan "Danger", Anne       Jet   68000      0.45      30600.0

```

### 1.1.3 openpyxl

**Load existing workbook** Um bestehenden Excel Dokumente zu laden, wird das Paket *load\_workbook* benötigt.

```

1   from openpyxl import load_workbook
2
3   wb_load = load_workbook("regions.xlsx")
4   output = wb_load.sheetnames
5
6   # Print
7   print(output)
# ['Sheet1']

```

**Workbook** Das Paket Workbook bietet die Möglichkeit Workbooks direkt im Arbeitsspeicher zu erstellen, ohne diese vorher gespeichert zu haben.

```

1   from openpyxl import load_workbook
2   from openpyxl import Workbook
3
4   wb_load = load_workbook("regions.xlsx")
5
6   # Create new Workbook
7   wb_new = Workbook()
8
9   # Create new Worksheets

```

```

10 ws1 = wb_new.create_sheet("New Sheet2",1)
11 ws0 = wb_new.create_sheet("Legende", 0)
12 ws2 = wb_new.create_sheet("Vorletze",-1)
13
14 output = wb_new.sheetnames
15
16 # Print
17 print(output)
18 # ['Legende', 'Sheet', 'Vorletze', 'New Sheet2']

```

Die Funktion `create_sheet` hat als zweiten optionalen Parameter die Reihenfolge definiert. Diese startet bei 0. Den Titel eines Sheets kann direkt mit `.title` abgefragt werden.

```

1 print(wb_new["Legende"].title)
2 print(ws0.title)
3 #Legende
4 #Legende

```

Ebenso kann über alle Worksheets iteriert werden.

```

1 for ws in wb_new:
2     print(ws.title)
3     #Legende
4     #Sheet
5     #Vorletze
6     #New Sheet2

```

Ein bestimmtes Worksheet zu bearbeiten, erfordert es das zugehörige Workbook zu aktivieren.

```

1 wb_new.active
2 ws1["A4"] = 4
3 output = ws1["A4"].value
4 # 4
5 wb_new.save("regions_new.xlsx")

```

Um das Workbook zu speichern, wird die `save()` Funktion genannt.

**Select** Zellen, Zeilen und Spalten können direkt über den ausgewählten Sheet angesteuert werden.

```

1 #from openpyxl import Workbook
2 #import openpyxl as op
3 from openpyxl import load_workbook
4 from openpyxl import Workbook
5
6 wb_load = load_workbook("regions.xlsx")
7
8 # Create new Workbook
9 wb_new = Workbook()
10
11 # Create new Worksheets
12 ws1 = wb_new.create_sheet("New Sheet2",1)
13 ws0 = wb_new.create_sheet("Legende", 0)
14 ws2 = wb_new.create_sheet("Vorletze",-1)
15
16 wb_load.active
17
18 for i in wb_load:
19     print(i.title)
20
21 range = wb_load["Sheet1"]["A1": "B7"]
22

```

```

23     for i in range:
24         for j in i:0
25             print(j.value)

```

Es gibt ebenso eine Iteration Funktion für Zeilen und Spalten.

```

1 #from openpyxl import Workbook
2 #import openpyxl as op
3 from openpyxl import load_workbook
4 from openpyxl import Workbook
5
6 wb_load = load_workbook("regions.xlsx")
7
8 # Create new Workbook
9 wb_new = Workbook()
10
11 # Create new Worksheets
12 ws1 = wb_new.create_sheet("New Sheet2",1)
13 ws0 = wb_new.create_sheet("Legende", 0)
14 ws2 = wb_new.create_sheet("Vorletze",-1)
15
16 ws = wb_load.active
17
18 for i in wb_load:
19     print(i.title)
20
21 range = ws.iter_rows(min_row = 1, max_row = 3, max_col = 3, values_only
22                     = True)
23
24 for i in range:
25     for j in i:
26         print(j)
27         ### Output
28         #Sheet1
29         #Region
30         #Units
31         #Sales
32         #South
33         #54
34         #332
35         #North
36         #20
37         #110
38         #>>>
39
40         ### Output without values_only = True
41         #Sheet1
42         #<Cell 'Sheet1'.A1>
43         #<Cell 'Sheet1'.B1>
44         #<Cell 'Sheet1'.C1>
45         #<Cell 'Sheet1'.A2>
46         #<Cell 'Sheet1'.B2>
47         #<Cell 'Sheet1'.C2>
48         #<Cell 'Sheet1'.A3>
49         #<Cell 'Sheet1'.B3>
50         #<Cell 'Sheet1'.C3>
51         #>>>

```

**Mutliple Sheets** Es können mehrere Dateien und mehrere Tabellenblätter ausgelesen werden.

```
1 import pandas as pd
2 import time
3 start_time = time.time()
4
5 # Load data
6 df_1 = pd.read_excel("shifts.xlsx", sheet_name= "Sheet1", skiprows=2,
7     engine='openpyxl', usecols=lambda x: 'Unnamed' not in x)
8 df_2 = pd.read_excel("shifts.xlsx", sheet_name= "Sheet1", skiprows=2,
9     engine='openpyxl', usecols=lambda x: 'Unnamed' not in x)
10 df_3 = pd.read_excel("shift_3.xlsx", skiprows=2, engine='openpyxl',
11     usecols=lambda x: 'Unnamed' not in x)
12
13 df_4 = pd.concat([df_1, df_2, df_3])
14
15 print(df_4)
16 print("---- %s seconds ----" % (time.time() - start_time))
17 ### Output
18 #      Shift ... Units Sold
19 #0      2 ...     157
20 #1      2 ...      83
21 #2      2 ...     180
22 #3      2 ...     186
23 #4      2 ...     117
24 #..     ... ... ...
25 #94     3 ...      42
26 #95     3 ...     168
27 #96     3 ...     132
28 #97     3 ...      111
29 #98     3 ...     151
30 #
31 #[299 rows x 6 columns]
32 #---- 0.4967920780181885 seconds ----
33 #>>>
```

**Store Data from Dataframe into Workbook** Mit der Funktion dataframe-to-rows wird ein Dataframe in Zeilen zerlegt. Die Werte aus diesen können einzeln adressiert werden und spezifischen Zellen in einem Worksheet zugewiesen werden.

## 1.2 Exploratory Data Analysis

Das Konzept Exploratory Data Analysis (EDA) wird unterschiedlich verwendet. Diese dient, die zu verwendenden Daten zu verstehen.

### 1.2.1 Overview

- df.head()
- for col in df.columns
- df.tail()
- df.shape
- df.describe()

- df[].unique()
- df.nunique()
- df.drop()

Als Datenquelle werden die COVID Zahlen des Robert-Koch-Institute (RKI) mit [Flat-File](#). Die Legende findet sich unter [Legende-RKI](#)

**head()** Im ersten Schritt werden die ersten Zeilen angezeigt.

```

1 import pandas as pd
2 import numpy as np
3 import seaborn as sb
4
5 df_rki = pd.read_csv("RKI_COVID19.csv")
6
7 print(df_rki.head())
8 ### Output
9 #   ObjectId ... Altersgruppe2
10 #0      1 ... Nicht übermittelt
11 #1      2 ... Nicht übermittelt
12 #2      3 ... Nicht übermittelt
13 #3      4 ... Nicht übermittelt
14 #4      5 ... Nicht übermittelt
15 #
16 #[5 rows x 18 columns]
17 #>>>

```

**columns** Die einzelnen Namen der Spalten werden über eine Iteration angezeigt.

```

1 for col in df_rki.columns:
2     print(col)
3 ### Output
4 ObjectId
5 IdBundesland
6 Bundesland
7 Landkreis
8 Altersgruppe
9 Geschlecht
10 AnzahlFall
11 AnzahlTodesfall
12 Meldedatum
13 IdLandkreis
14 Datenstand
15 NeuerFall
16 NeuerTodesfall
17 Refdatum
18 NeuGenesen
19 AnzahlGenesen
20 IstErkrankungsbeginn
21 Altersgruppe2
22 >>>
23
24 # tail() shows the last five entries
25 print(df.tail())
26 ### Output
27 #           ObjectId ... Altersgruppe2

```

```
28 #1134991    1134992    ...  Nicht übermittelt
29 #1134992    1134993    ...  Nicht übermittelt
30 #1134993    1134994    ...  Nicht übermittelt
31 #1134994    1134995    ...  Nicht übermittelt
32 #1134995    1134996    ...  Nicht übermittelt
33 #
34 #[5 rows x 18 columns]
35 #>>>
```

**describe()** Die Funktion `describe()` gibt einen Standardabfrage an den Dataframe wieder.

```
1 df = pd.read_csv("RKI_COVID19.csv")
2 df_a = df["Altersgruppe"]
3 x = df_a.describe()
4 print(x)
5 ### Output
6 #
7 #count      1134996
8 #unique      7
9 #top        A35-A59
10 #freq       386139
11 #Name: Altersgruppe, dtype: object
12 #>>>
```

Als nächstes wird ein Jupyter Auszug einfügt, in welcher weiter mit der `describe()` Funktion gearbeitet wird.

# EDA\_Begin

January 31, 2021

```
[3]: import pandas as pd
import numpy as np
import seaborn as sb

# URL: https://npgeo-corona-npgeo-de.hub.arcgis.com/datasets/
# dd4580c810204019a7b8eb3e0b329dd6_0
df = pd.read_csv("RKI_COVID19.csv")
df_a = df[["Altersgruppe"]]

x = df.describe()

print(x)
```

	ObjectId	IdBundesland	AnzahlFall	AnzahlTodesfall	\
count	1.134996e+06	1.134996e+06	1.134996e+06	1.134996e+06	
mean	5.674985e+05	7.925296e+00	1.931618e+00	4.908828e-02	
std	3.276453e+05	3.577952e+00	3.087737e+00	2.809964e-01	
min	1.000000e+00	1.000000e+00	-5.000000e+00	-1.000000e+00	
25%	2.837498e+05	5.000000e+00	1.000000e+00	0.000000e+00	
50%	5.674985e+05	8.000000e+00	1.000000e+00	0.000000e+00	
75%	8.512472e+05	9.000000e+00	2.000000e+00	0.000000e+00	
max	1.134996e+06	1.600000e+01	1.640000e+02	1.500000e+01	

	IdLandkreis	NeuerFall	NeuerTodesfall	NeuGenesen	\
count	1.134996e+06	1.134996e+06	1.134996e+06	1.134996e+06	
mean	8.253901e+03	6.401785e-03	-8.636367e+00	-1.136446e+00	
std	3.541520e+03	8.455908e-02	1.774146e+00	3.001681e+00	
min	1.001000e+03	-1.000000e+00	-9.000000e+00	-9.000000e+00	
25%	5.566000e+03	0.000000e+00	-9.000000e+00	0.000000e+00	
50%	8.216000e+03	0.000000e+00	-9.000000e+00	0.000000e+00	
75%	9.762000e+03	0.000000e+00	-9.000000e+00	0.000000e+00	
max	1.607700e+04	1.000000e+00	1.000000e+00	1.000000e+00	

	AnzahlGenesen	IstErkrankungsbeginn
count	1.134996e+06	1.134996e+06
mean	1.672409e+00	6.793460e-01
std	2.955976e+00	4.667282e-01
min	-3.000000e+00	0.000000e+00

1

**nunique() unique()** Mit dieser Funktion werden alle Spalten abgegangen und die verschiedenen Werte wieder gegeben.

```

1 df.nunique()
2 ObjectId          1134996
3 IdBundesland      16
4 Bundesland        16
5 Landkreis         412
6 Altersgruppe      7
7 Geschlecht        3
8 AnzahlFall        128
9 AnzahlTodesfall   16
10 Meldedatum        380
11 IdLandkreis      412
12 Datenstand        1
13 NeuerFall         3
14 NeuerTodesfall    4
15 Refdatum          394
16 NeuGenesen        4
17 AnzahlGenesen     127
18 IstErkrankungsbeginn 2
19 Altersgruppe2     1
20 dtype: int64

```

Sollen die Werte der jeweiligen Spalte ausgegeben werden, so bietet sich die Funktion unique() an.

```

1 df["Geschlecht"].unique()
2 array(['M', 'W', 'unbekannt'], dtype=object)
3 df["NeuerFall"].unique()
4 array([ 0, -1,  1], dtype=int64)

```

**Check for Nullvalues** In zwei Schritte kann eine Übersicht geschaffen werden, welche Ausschluss über die leere Werte im Dataframe gibt.

```

1 df.isnull().sum()
2 ObjectId          0
3 IdBundesland      0
4 Bundesland        0
5 Landkreis         0
6 Altersgruppe      0
7 Geschlecht        0
8 AnzahlFall        0
9 AnzahlTodesfall   0
10 Meldedatum        0
11 IdLandkreis      0
12 Datenstand        0
13 NeuerFall         0
14 NeuerTodesfall    0
15 Refdatum          0
16 NeuGenesen        0
17 AnzahlGenesen     0
18 IstErkrankungsbeginn 0
19 Altersgruppe2     0
20 dtype: int64

```

Mit isnull() wird eine logische Überprüfung durchgeführt werden. Mit der Summe über alle Spalten wird die Menge der leeren Werte summiert angezeigt.

**Convert Date as String to Date type** Die übergebenen Datumswerte sind als String übergeben. Um diese filtern und weiter bearbeiten zu können, werde diese mit dem Paket *datetime* umgewandelt.

```

1 #%%
2 import pandas as pd
3 import numpy as np
4 import seaborn as sb
5 import datetime as dt
6
7 # URL: https://npgeo-corona-npgeo-de.hub.arcgis.com/datasets/
8     dd4580c810204019a7b8eb3e0b329dd6_0
9 df = pd.read_csv("RKI_COVID19.csv")
10
11 df_SKA = df.loc[df["Landkreis"]=="SK Augsburg"]
12 df_SKA = df_SKA[["Melde datum", "Anzahl Fall"]]
13 df_SKA = df_SKA.iloc[0:3]
14 df_SKA["Date"] = df_SKA["Melde datum"].apply(lambda x: dt.datetime.
15     strptime(x, '%Y/%m/%d %H:%M:%S')) # apply
16 df_SKA["year"] = df_SKA["Date"].apply(lambda x: x.year)
17 df_SKA["Month"] = df_SKA["Date"].apply(lambda x: x.month)
18 df_SKA["Day"] = df_SKA["Date"].apply(lambda x: x.day)
19 df_SKA.drop(["Melde datum"], axis=1, inplace=True) # inplace modifize
20     the same variable
21 df_SKA.head()
22
23 # %%
24 #AnzahlFall Date year Month Day
25 #845500 1 2020-09-13 2020 9 13
26 #845503 1 2020-05-28 2020 5 28
27 #845504 1 2020-05-31 2020 5 31

```

### Histogramm Anteil der Infektionen:

```

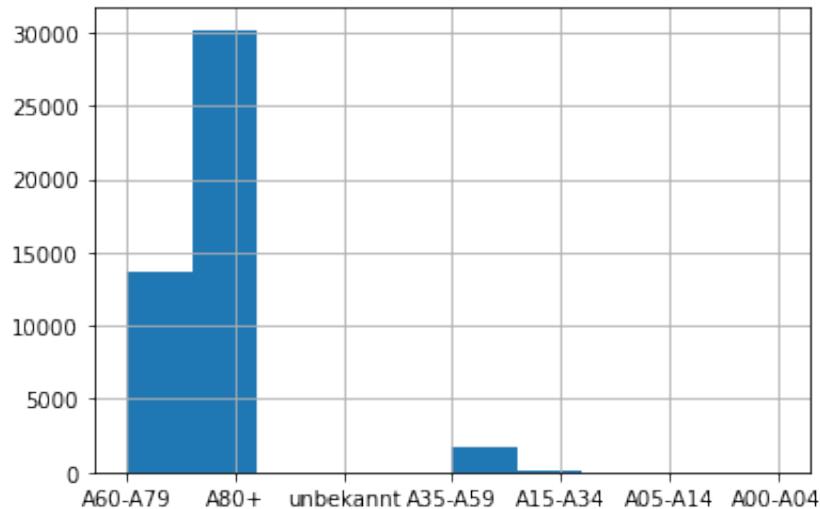
1 #%%
2 import pandas as pd
3 import numpy as np
4 import seaborn as sb
5 import datetime as dt
6
7 # Source
8 # URL: https://npgeo-corona-npgeo-de.hub.arcgis.com/datasets/
9     dd4580c810204019a7b8eb3e0b329dd6_0
10 df_SKA = pd.read_csv("RKI_COVID19.csv")
11
12 #Select Data for Augsburg
13 df_SKA = df_SKA.loc[df_SKA["Landkreis"]=="SK Augsburg"]
14
15 # Filter for Infected People
16 df_SKA = df_SKA.loc[lambda df: (df["Anzahl Fall"]>0) & (df["
17     Anzahl Todesfall"]<=0)]
18
19 # Drops other columns
20 Import_Col = [
21     "Melde datum",
22     "Anzahl Fall",
23     "Neuer Fall",
24     "Anzahl Todesfall",
25     "Altersgruppe"
26 ]
27 df_SKA = df_SKA[Import_Col]

```

```

26
27 # Fix Date
28 df_SKA["Date"] = df_SKA["Meldedatum"].apply(lambda x: dt.datetime.
29     strptime(x, '%Y/%m/%d %H:%M:%S'))
30 df_SKA["year"] = df_SKA["Date"].apply(lambda x: x.year)
31 df_SKA["Month"] = df_SKA["Date"].apply(lambda x: x.month)
32 df_SKA["Day"] = df_SKA["Date"].apply(lambda x: x.day)
33 df_SKA.drop(["Meldedatum"], axis=1, inplace=True)
34
35 # Sort Values
36 df_SKA.sort_values(by=["Date"], inplace=True)
37
38 # Show Histogram
39 df_SKA["Altersgruppe"].hist()

```

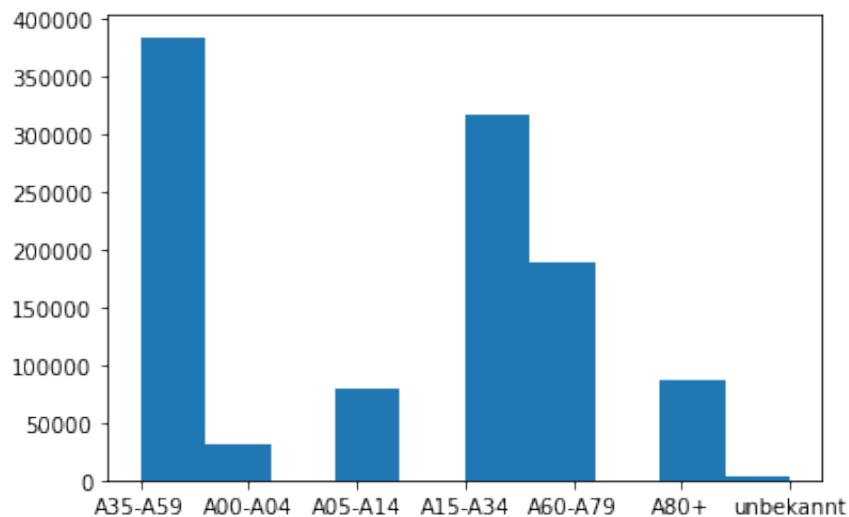


und der Todesfälle in Deutschland.

```

1 # Filter for Infected People
2 df_SKA = df_SKA.loc[df["AnzahlFall"]>0] & (df["
    AnzahlTodesfall"]>0]

```



## 1.2.2 Relationship Analysis

### Correlation, Headmap, Correlation

- seaborn df.corr()
- sns.heatmap(df.corr())
- sns.pairplot(df)

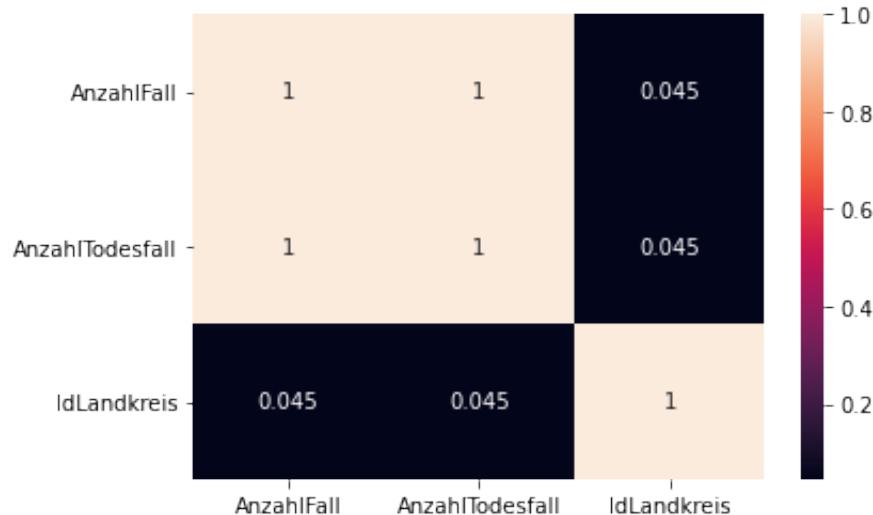
Beide Funktionen corr() und sns.heatmap() können nur auf metrische Variablen angewendet werden.

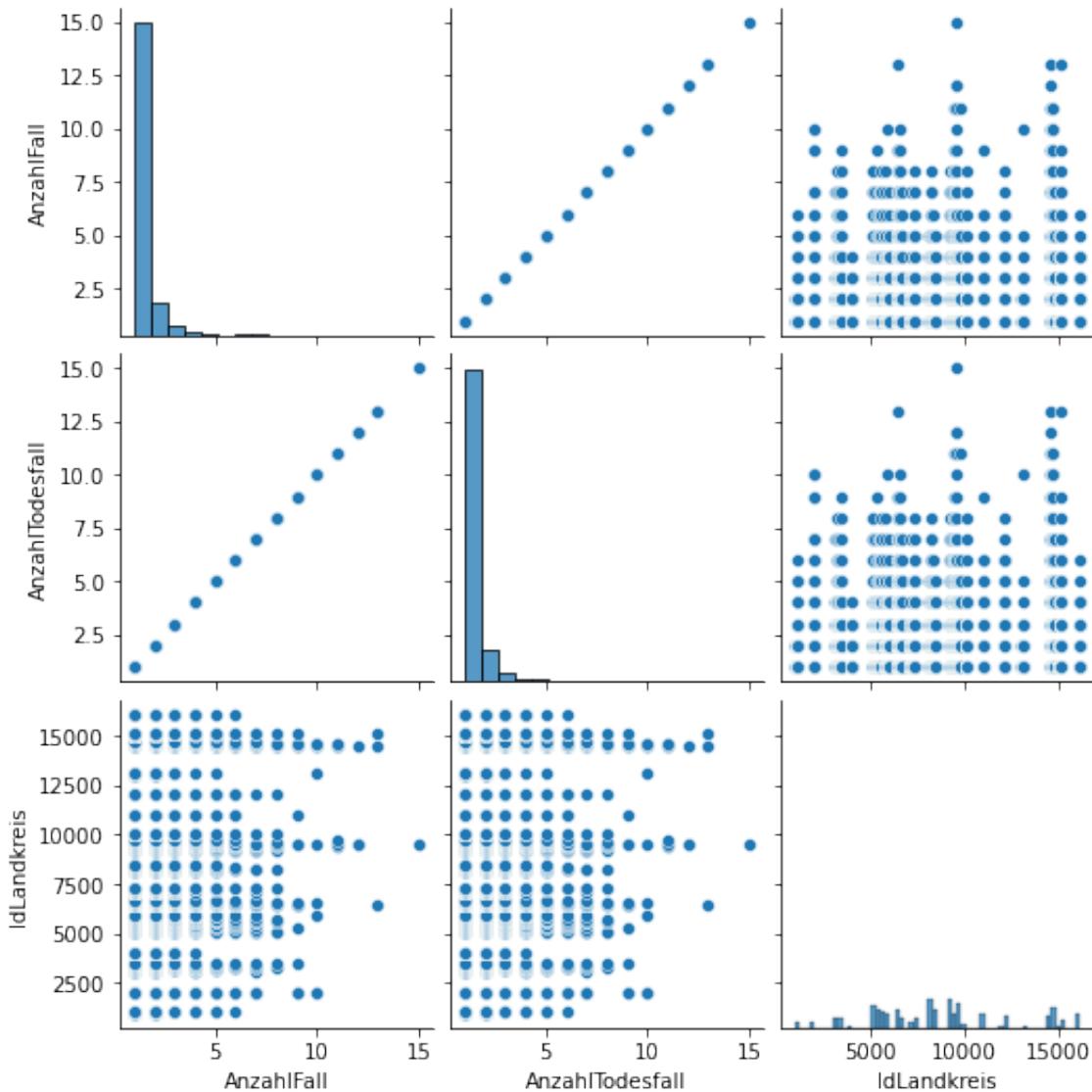
```
1 import pandas as pd
2 import numpy as np
3 import seaborn as sns
4 import datetime as dt
5
6 from seaborn.palettes import color_palette
7
8 # Source
9 # URL: https://npgeo-corona-npgeo-de.hub.arcgis.com/datasets/
10 # dd4580c810204019a7b8eb3e0b329dd6_0
11 df_SKA = pd.read_csv("RKI_COVID19.csv")
12
13 #Select Data for Augsburg
14 #df_SKA = df_SKA.loc[df_SKA["Landkreis"]=="SK Augsburg"]
15 # ObjectId,
16 # IdBundesland
17 # Bundesland
18 # Landkreis
19 # Altersgruppe
20 # Geschlecht
21 # AnzahlFall
22 # AnzahlTodesfall
23 # Meldedatum
24 # IdLandkreis
25 # Datenstand
26 # NeuerFall
27 # NeuerTodesfall
28 # Refdatum
29 # NeuGenesen
30 # AnzahlGenesen
31 # IstErkrankungsbeginn
32 # Altersgruppe2
33
34 # Filter for Infected People
35 df_SKA = df_SKA.loc[lambda df: (df["AnzahlFall"]>0) & (df["AnzahlTodesfall"]>0)]
36
37 # Drops other columns
38 Import_Col = [
39 "Meldedatum",
40 "AnzahlFall",
41 "AnzahlTodesfall",
42 "Altersgruppe",
43 "#IdBundesland",
44 "IdLandkreis",
45 "Geschlecht"
46 ]
```

```

46 df_SKA = df_SKA[Import_Col]
47
48 # Fix Date
49 df_SKA["Date"] = df_SKA["Meldedatum"].apply(lambda x: dt.datetime.
50     strptime(x, '%Y/%m/%d %H:%M:%S'))
51 #df_SKA["year"] = df_SKA["Date"].apply(lambda x: x.year)
52 #df_SKA["Month"] = df_SKA["Date"].apply(lambda x: x.month)
53 #df_SKA["Day"] = df_SKA["Date"].apply(lambda x: x.day)
54 df_SKA.drop(["Meldedatum"], axis=1, inplace=True)
55
56 # Sort Values
57 #df_SKA.sort_values(by=["Date"], inplace=True)
58
59 # Relationship
60 correlation = df_SKA.corr()
61 sns.heatmap(correlation,xticklabels=correlation.columns, yticklabels=
62     correlation.columns, annot=True)
63
64 sns.pairplot(df_SKA)

```





```

1 #%%
2 import pandas as pd
3 import numpy as np
4 import seaborn as sns
5 import datetime as dt
6
7 from seaborn.palettes import color_palette
8
9 # Source
10 # URL: https://npgeo-corona-npgeo-de.hub.arcgis.com/datasets/
11 # dd4580c810204019a7b8eb3e0b329dd6_0
12 df_SK = pd.read_csv("RKI_COVID19.csv")
13
14 #Select Data for Augsburg
15 #df_SK = df_SK.loc[df_SK["Landkreis"]=="SK Augsburg"]
16 # ObjectId,
17 # IdBundesland
18 # Bundesland
19 # Landkreis

```

```

19 # Altersgruppe
20 # Geschlecht
21 # AnzahlFall
22 # AnzahlTodesfall
23 # Meldedatum
24 # IdLandkreis
25 # Datenstand
26 # NeuerFall
27 # NeuerTodesfall
28 # Refdatum
29 # NeuGenesen
30 # AnzahlGenesen
31 # IstErkrankungsbeginn
32 # Altersgruppe2
33
34 # Filter for Infected People
35 df_SKA = df_SKA.loc[lambda df: (df["AnzahlFall"]>0) & (df["AnzahlTodesfall"]>0)]
36
37 # Drops other columns
38 Import_Col = [
39 "Meldedatum",
40 "#AnzahlFall",
41 "AnzahlTodesfall",
42 "Altersgruppe",
43 "#IdBundesland",
44 "#IdLandkreis",
45 "Geschlecht"
46 ]
47 df_SKA = df_SKA[Import_Col]
48
49 # Fix Date
50 df_SKA["Date"] = df_SKA["Meldedatum"].apply(lambda x: dt.datetime.
51     strptime(x, '%Y/%m/%d %H:%M:%S'))
52 #df_SKA["year"] = df_SKA["Date"].apply(lambda x: x.year)
53 #df_SKA["Month"] = df_SKA["Date"].apply(lambda x: x.month)
54 df_SKA["Day"] = df_SKA["Date"].apply(lambda x: x.day)
55 df_SKA.drop(["Meldedatum"], axis=1, inplace=True)
56
57 # Sort Values
58 #df_SKA.sort_values(by=["Date"], inplace=True)
59
60 # Relationship
61 sns.relplot(x= "Date",y="AnzahlTodesfall", hue="Geschlecht", data=
62     df_SKA)
63 # %%

```

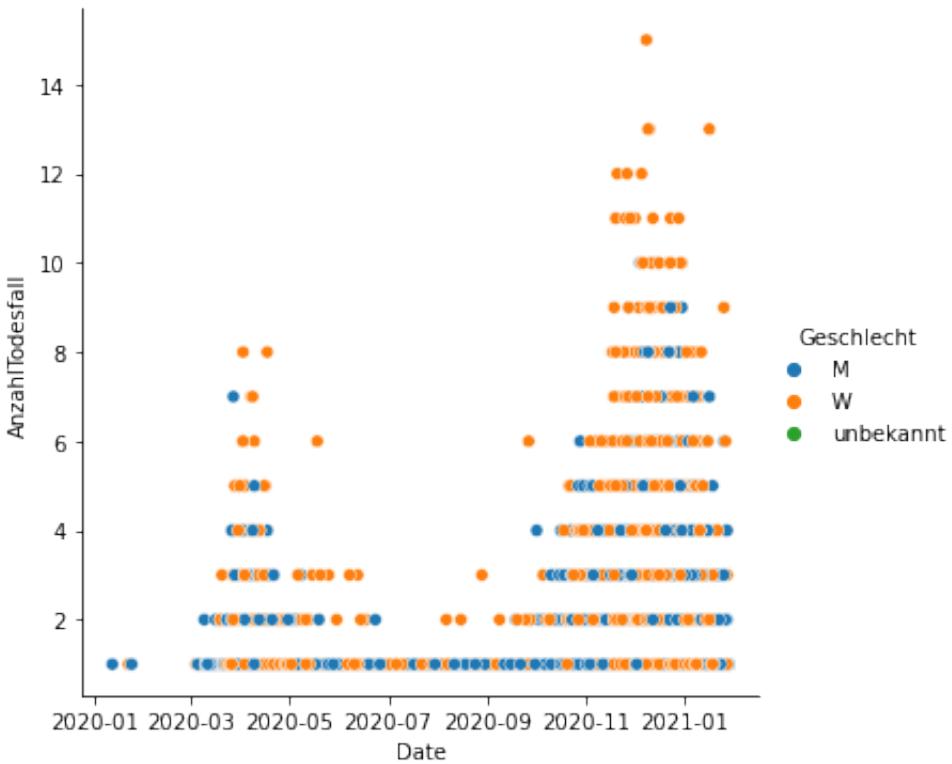


Abbildung 238: Todesfälle

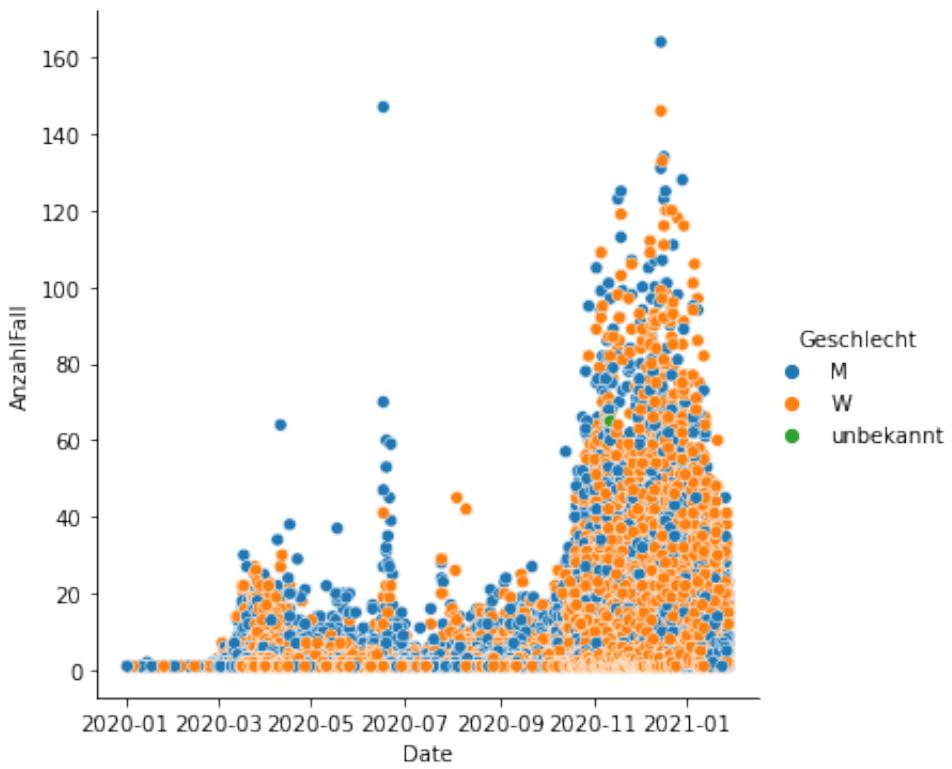


Abbildung 239: Infektionen

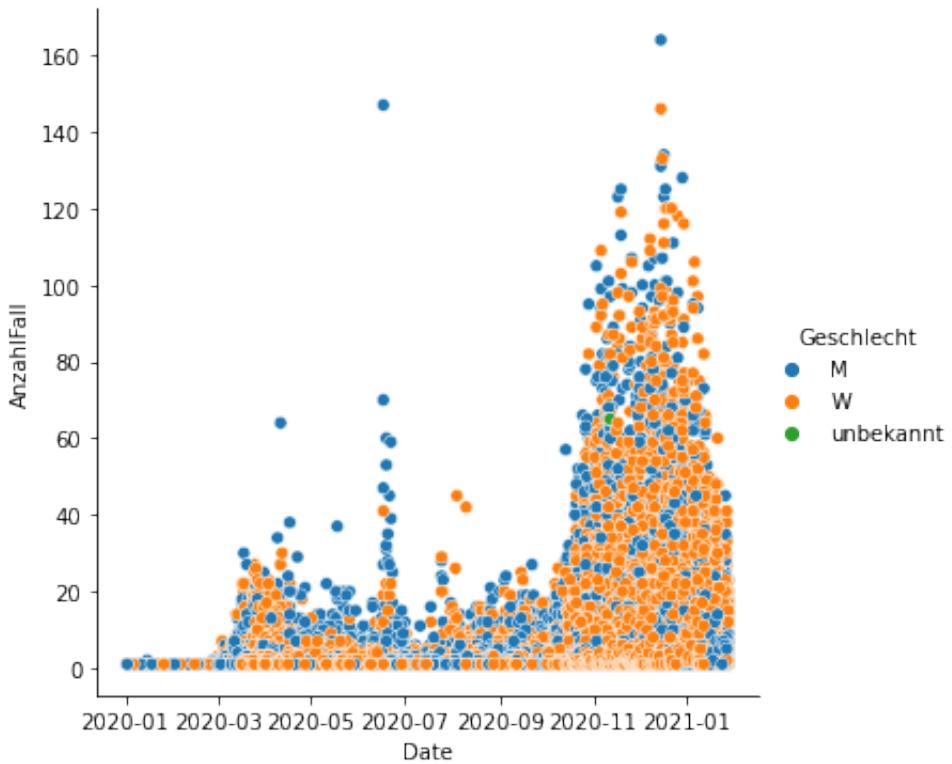


Abbildung 240: FallAnzahl nach Altersgruppen

```
1 sns.distplot(df_SKA["AnzahlFall"])
```

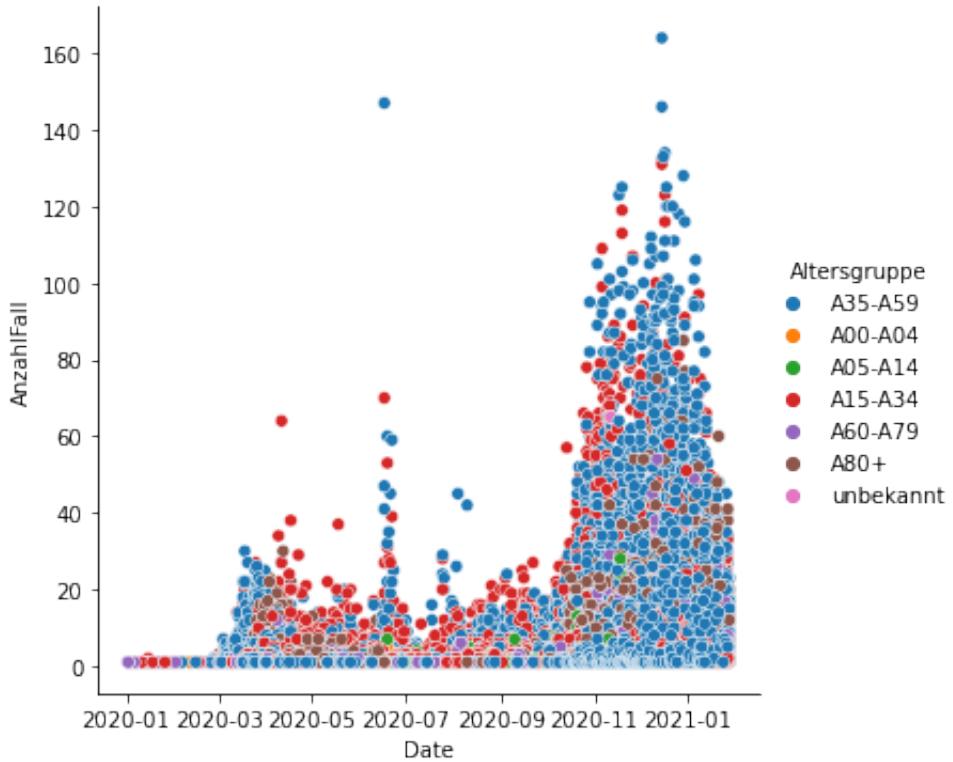
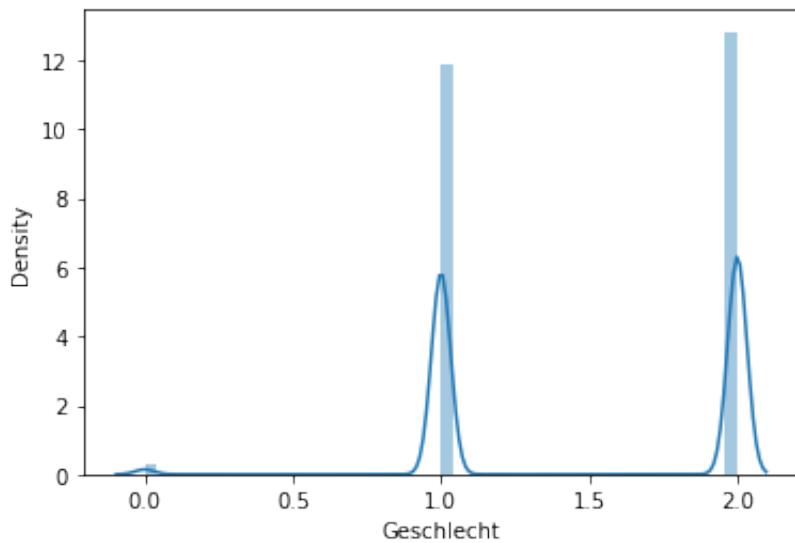


Abbildung 241: FallAnzahl nach Altersgruppen

Mit `.replace()` können die string Werte in numerische Werte überführt werden.



```
1     sns.catplot(x="AnzahlFall", kind="box", data=df_SKA)
```

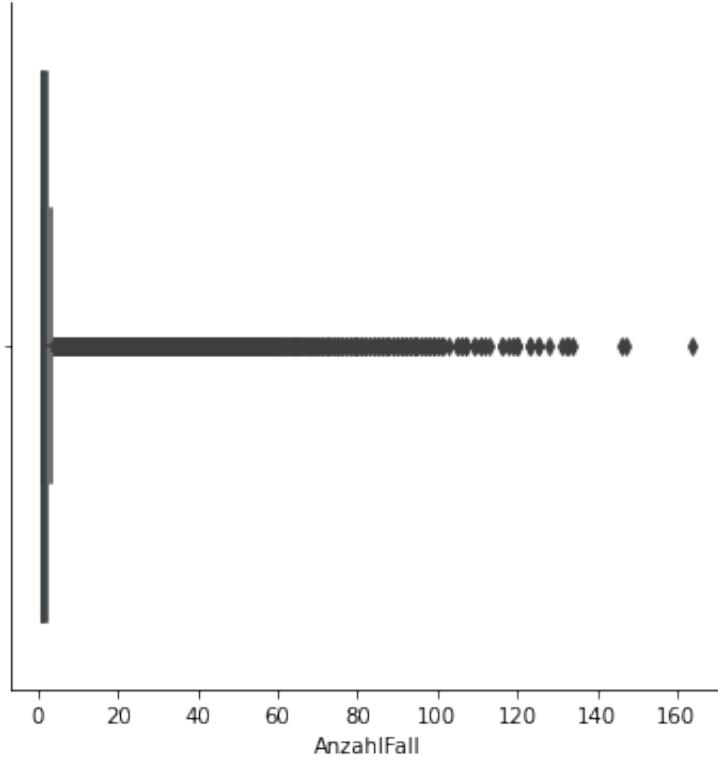


Abbildung 242: FallAnzahl nach Altersgruppen

## 1.3 Data Visualization - Matplotlib

### 1.3.1 Preparation

Die Daten für die Grafiken in *Matplot.pyplot* bereit zu stellen, müssen einige Kriterien erfüllt sein.

**Spalteauslesen** Es bietet sich an, die Spalten sich als Dataframe ausgeben zu lassen. In *VStudio* können diese übersichtlich angezeigt werden.

```
1 df_i_columns = Dataframe(df_i.columns)
```

Es dürfen keine NULL Werte beinhaltet sein. Über den *.info* werden die Kriterien des Dataframes ausgegeben.

```
1 df_i.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60 entries, 0 to 59
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   month            60 non-null    int64  
 1   starting_balance 60 non-null    float64 
 2   interest_paid    60 non-null    float64 
 3   principal_paid  60 non-null    float64 
 4   new_balance      60 non-null    float64 
 5   interest_rate   60 non-null    float64 
 6   car_type          60 non-null    object  
dtypes: float64(5), int64(1), object(1)
memory usage: 3.4+ KB
```

**Filter - Praxis** Es gibt mehrere Möglichkeiten Filter bezüglich des Dataframes anzuwenden. Eine Methode ist, Filter anzulegen und diese kombiniert über `.loc` anzuwenden.

```
1 #%%
2 # Configure Filters
3
4 df_i_filter_1 = df_i[df_i.columns.iloc[8,0]] == "VW Golf R"
5 df_i_filter_2 = df_i[df_i.columns.iloc[7,0]] == 0.029
6
7 #%%
8 # Apply one filters
9 # First Methode
10 df_i[df_i_filter_1]
11
12 #%%
13 # Apply one filters
14 # First Methode
15 df_filter = df_i.iloc[df_i_filter_1.array]
16 df_filter.head()
17
18 #%%
19 # Apply two filters
20 # First Methode
21 df_filter_two = df_i[df_i_filter_1 & df_i_filter_2]
22 df_filter_two.head()
23
24
25 # %%
26 # Apply two filters
27 # Secound Methode
28
29 df_filter_two = df_i.loc[df_i_filter_1 & df_i_filter_2]
30 df_filter_two.head()
```

**Transformation: Numpy - ndarray** Einzelne Spalten müssen wir die einzelne Darstellung in `ndarray` umgewandelt werden. Es gibt zwei Methoden in `pandas` welche dies tun. Ebenso gibt es Methoden, welche nur Arrays ausgeben, dies aber nicht in `ndarrays` transformieren.

- `df.values` oder
- `df.to_numpy()`

```
1 # %%
2 ### Preparation for mathplot
3
4 # ndarray (numpy) Month
5 varSelCol_3 = df_i.columns.iloc[0,0]
6 nda_Month = df_i[varSelCol_3].values
7
8 # ndarray (numpy) Interest Paid
9 varSelCol_3 = df_i.columns.iloc[[0,1,2],0]
10 nda_interest_paid = df_i[varSelCol_3].values
11
12 # ndarray (numpy) Prinzipal Paid
13 varSelCol_3 = df_i.columns.iloc[3,0]
14 nda_prinzipal_paid = df_i[varSelCol_3].to_numpy()
```

Variables				
	Name	Type	Size	Value
df_i	DataFrame	(60, 7)	month starting_	
df_i_columns	DataFrame	(7, 1)	0 0 month 1 sta	
nda_Month	ndarray	(60,)	[ 1 2 3 4 5 6 7	
nda_interest_f	ndarray	(60, 3)	[[1.000000e+00	
nda_prinzipal_	ndarray	(60,)	[484.3 487.13 4	
path_financia]	str	22	data/table_i702	
varSelcol_3	str	14	principal paid	

Die letzte Funktion wird von *pandas* vorgeschlagen, als die zu bevorzugenden Funktion.

Die Auslesung einer Spalte reicht nicht aus. Hierbei wird nur eine *Series* zurückgegeben.

)

**Historischer Abriss** Um fehlende Werte in Python zu verstehen, muss die historisch, gewachsenen Struktur der Pakete *pandas*, *NumPy*, sowie der Sprache R - Statistical Programming Language (R) betrachtet werden.

- Die Begriffe None und NULL (R) sind inhaltlich gleich. In Python wird nur None verwendet. Als Beispiel in C++ (CPP) und Power BI wird NULL (R) verwendet.
- In Python selbst gibt es viele Pakete (*NumPy*, *Math*) welche NaN ermöglichen: *np.nan*.
- Die Pakete *pandas* und *NumPy* sind in Anlehnung an R geschrieben worden.
- Die Begrifflichkeit Not available (Na) gibt in Python nicht, sondern wurde nur aus historischen Gründen übernommen.
- *Pandas* ist aufbauend auf *NumPy* geschrieben worden. Es unterscheidet jedoch nicht mehr zwischen NaN aus *NumPy* und None. Die Funktionen

*Dataframe.isnull()* (1.3)

*Dataframe.isna()* (1.4)

geben das gleich aus.

**Umgang in Pandas mit NaN** *Pandas* unterscheidet bei der Suche nach fehlenden Daten nicht zwischen den beiden Datentypen None und Not a Number (NaN). Die Funktionen

*pandas.isnull()* (1.5)

und

*pandas.isna()* (1.6)

geben die gleichen *Wahr/Falsch* Werte zurück. Dies gilt ebenso bei der Anwendung auf Dataframes

```

1 df = pd.DataFrame([2,np.nan, None])
2 df["2"] = df[0].isnull()
3 df["3"] = df[0].isna()
4 df

```

	0	2	3
0	2.0	False	False
1	NaN	True	True
2	NaN	True	True

Hinzu kommt, dass Pandas None in NaN in NaN umwandelt. Die Funktion `isnull()` und `isna()` ist auf das Fundament in R zurück zuführen.

Die Werte in Variablen abgespeichert, erzielen das gleiche Resultat, bei einer Gleichheitsabfrage:

```
1 x = np.nan
2 y = np.nan
3 x == y # False
```

Wir gefragt, ob es sich um das gleiche Objekt `is` handelt, wird bei den Variablen

```
1 x is y # True
```

dies bejaht.

Befindet sich `NaN` in einem Dataframe, wird `false` zurückgegeben:

```
1 df = pd.DataFrame([None,np.nan, 3], [4, None, None])
2 df["5"] = df[0] == np.nan
3 df["6"] = df[0] is np.nan
4 df
```

	0	5	6
4	NaN	False	False
NaN	NaN	False	False
NaN	3.0	False	False

Der Grund ist, dass im Dataframe NaN anders codiert ist. Für weiteres siehe das Glossar: NaN.

**Umgang mit None** Der Wert `None` ist immer der selbe, weshalb die Gleichheitsabfrage `true` zurückgibt. Ebenso ist die Identität eines jeden `None` Wertes auf ein Objekt zurückzuführen: `TypNone`

```
1 x = None
2 y = None
3 x == y # True
4 x is y # True
```

**Näheres zu `None` und `NULL`** In Python existiert das Konzept `None`, welches aus die Funktion von `NULL` (R) aus anderen Sprachen übernimmt. Die Funktion `print()` ist Funktion, welche `None` wieder gibt.

```
1 print("Hello")
2 >>> Hello
3 print(print("Hello"))
4 >>> Hello
5 >>> None
```

## Removing or filling in missing data

```
1 #### Loading Data
2 #path_financial = "data/car_financing.csv"
3 path_financial = "data/table_i702t60.csv"
4 df = pd.read_csv(path_financial)
```

<sup>7</sup>Unterschied in R zwischen `NaN` und `NULL`, Unterschied zwischen den Funktionen in Pandas und NumPy, r-na-vs-null, Null in Python

```

6   # %% Scanning Data
7   df.info()
8
9   # %% Adding a missing value
10  df.iloc[2,2] = np.nan
11
12  # %% Check up
13  df.info()
14
15  # %% Droping rows
16  ...
17
18  The function .dropna() allows to delete any rows, with np.nan values.
19  axis:[0,1], default 0
20  how : ['any','all'], default 'any'
21  - 'any': If any nan are present drop the row or column
22  - 'all': If all values are nan, drop row or column
23  etc.
24  ...
25
26  df.dropna(how='any', inplace = True)

```

**Fill-In: General** Im allgemeinen ist `.iloc()` nicht dafür ausgelegt, dass diese Funktion Werte im ursprünglichen Dataframe anpassen. Es kann verwendet werden, wie zum Beispiel:

```

1  #%% Replace Values
2  for i in range(3,6):
3      df.iloc[i,2]= 3

```

	0	4	7
4	NaN	False	False
NaN	NaN	False	False
NaN	3.0	False	False

Gleich ist es möglich mit der doppelte Indexierung die Werte zu ändern:

```

1  # %% For loop Changing Value
2  for i, j in enumerate(df_column):
3      df.loc[i,j] = np.nan

```

Achtung, wenn Zeilen fehlen, oder durch vorherige Schritte entfernt werden, sind die Indexes nicht mehr vorhanden, und können zu Problemen führen:

**Fill-In: `fillna()`** Der entsprechende Bereich kann ersetzt werden.

```

1  ...
2  DataFrame.fillna(value=None, method=None, axis=None, inplace=False,
3      limit=None, downcast=None)[source]
4  Fill NA/NaN values using the specified method.
5
6  Parameters
7  valuescalar, dict, Series, or DataFrame
8  Value to use to fill holes (e.g. 0), alternately a dict/Series/
9  DataFrame of values specifying which value to use for each index (
    for a Series) or column (for a DataFrame). Values not in the dict/
    Series/DataFrame will not be filled. This value cannot be a list.
8
9  method{'backfill', 'bfill', 'pad', 'ffill', None}, default None

```

```

10 Method to use for filling holes in reindexed Series pad / ffill:
11     propagate last valid observation forward to next valid backfill /
12     bfill: use next valid observation to fill gap.
13
14 axis{0 or 'index, 1 or 'columns}
15 Axis along which to fill missing values.
16
17 inplacebool, default False
18 If True, fill in-place. Note: this will modify any other views on this
19     object (e.g., a no-copy slice for a column in a DataFrame).
20
21 limitint, default None
22 If method is specified, this is the maximum number of consecutive NaN
23     values to forward/backward fill. In other words, if there is a gap
24     with more than this number of consecutive NaNs, it will only be
25     partially filled. If method is not specified, this is the maximum
26     number of entries along the entire axis where NaNs will be filled.
27     Must be greater than 0 if not None.
28
29 downcastdict, default is None
30 A dict of item->dtype of what to downcast if possible, or the string
31     ''infer' which will try to downcast to an appropriate equal type (e.g.
32     .float64 to int64 if possible).
33
34 Returns
35 DataFrame or None
36 Object with missing values filled or None if inplace=True.
37

```

Die Funktion `.fillna()` ermöglicht es NaN Wert direkt zu ersetzen.

df (60, 7)									
		index	month	starti...	interes...	princi...	new_ba...	interes...	car_ty...
		Y	Y	Y	Y	Y	Y	Y	Y
0	0	nan	34689.96	202.93	484.3	34205.66	0.0702	Toyota S...	
1	1	2	nan	200.1	487.13	33718.53	0.0702	Toyota S...	
2	3	4	33228.55	194.38	nan	32735.7	0.0702	Toyota S...	
3	4	5	32735.7	3	495.73	nan	0.0702	Toyota S...	
4	5	6	32239.97	3	498.63	31741.34	nan	Toyota S...	
5	6	7	31741.34	3	501.55	31239.79	0.0702	nan	
6	7	8	31239.79	182.75	504.48	30735.31	0.0702	Toyota S...	
7	8	9	30735.31	179.8	507.43	30227.88	0.0702	Toyota S...	
8	9	10	30227.88	176.83	510.4	29717.48	0.0702	Toyota S...	
9	10	11	29717.48	173.84	513.39	29204.09	0.0702	Toyota S...	
10	11	12	29204.09	170.84	516.39	28687.7	0.0702	Toyota S...	

```
1 df.info()
```

```
[97] df.info()
x <class 'pandas.core.frame.DataFrame'>
Int64Index: 60 entries, 0 to 2
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   month       58 non-null    float64
 1   starting_balance 58 non-null    float64
 2   interest_paid   59 non-null    float64
 3   principal_paid 58 non-null    float64
 4   new_balance    58 non-null    float64
 5   interest_rate  58 non-null    float64
 6   car_type      58 non-null    object  
dtypes: float64(6), object(1)
memory usage: 5.8+ KB
```

```
1 df.fillna("x", inplace = True)
```

```
1 df.info()
```

```
[102] df.info()
x <class 'pandas.core.frame.DataFrame'>
Int64Index: 60 entries, 0 to 2
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   month       60 non-null    object 
 1   starting_balance 60 non-null    object 
 2   interest_paid   60 non-null    object 
 3   principal_paid 60 non-null    object 
 4   new_balance    60 non-null    object 
 5   interest_rate  60 non-null    object 
 6   car_type      60 non-null    object  
dtypes: object(7)
memory usage: 5.8+ KB
```

```
1 varIsNull = df["interest_paid"].isna()
2 df["interest_paid"].iloc[varIsNull] = "y"
```

```
1 df.to_numpy() # NumPy Array
2 df.values # NumPy Array
3 df.to_dict() # Dictionaries
```

### 1.3.2 Simple Plot

Die Funktion `plt.plot()` benötigt als Inputs **NumPy Arrays**. Liegt Datensätze in Form eines pandas Dataframe vor, so müssen die einzelnen Spalten in solche konvertiert werden, siehe 1.3.1.

```
1 #%% Import Packages & Data
2
3 import pandas as pd
4 import numpy as np
5 import matplotlib.pyplot as plt
6 import seaborn as sns
7
8
9 '''Loading Data'''
10 path_financial = "data/car_financing.csv"
```

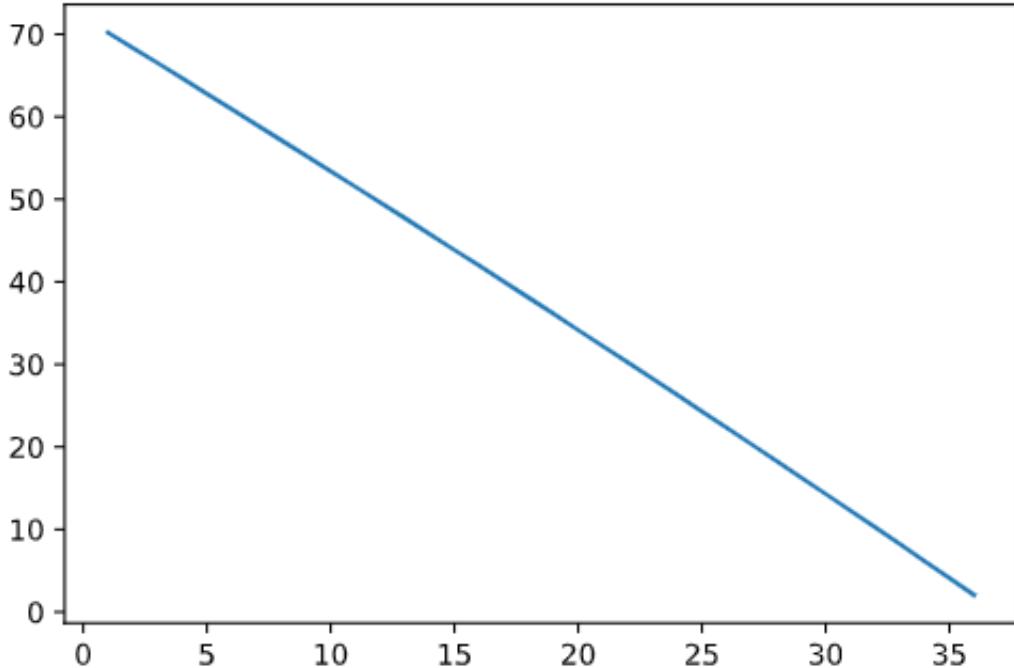
```

11 #path_financial = "data/table_i702t60.csv"
12 df_i = pd.read_csv(path_financial)
13
14 ''' Extract column names'''
15 df_i_columns = pd.DataFrame(df_i.columns)
16
17 #%% Check for NaN or Null Values
18 df_i["car_type"].unique()
19
20 # %% Filter with iloc
21 filter_cartype = df_i["car_type"] == "Toyota Corolla"
22 df_i = df_i.iloc[filter_cartype.array]
23
24
25 # %% Preparation for mathplot
26
27 '''Month'''
28 varSelCol_3 = df_i_columns.iloc[0,0]
29 nda_Month = df_i[varSelCol_3].to_numpy()
30
31 ''' Interest Paid'''
32 varSelCol_3 = df_i_columns.iloc[3,0]
33 nda_interest_paid = df_i[varSelCol_3].values
34
35 '''Prinzipal Paid'''
36 varSelCol_3 = df_i_columns.iloc[4,0]
37 nda_prinzipal_paid = df_i[varSelCol_3].to_numpy()

```

Ist dies geschehen, kann mit `plt.plot()` ein Graf erzeugt werden.

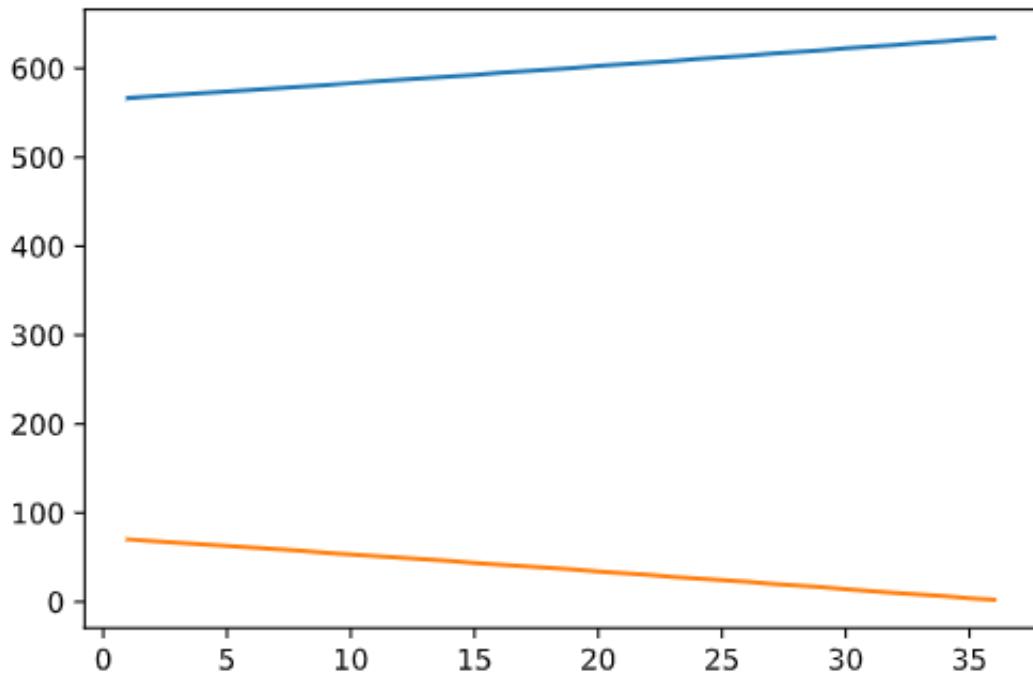
```
1 plt.plot(nda_Month,nda_interest_paid)
```



Wird zwei `plt.plot()` erstellt, werden diese in einer Grafik zusammengefügt

```
1 # %% Plot two graphs in one graphic
```

```
2 plt.plot(nds_Month, nds_prinzipal_paid)
3 plt.plot(nds_Month, nds_interest_paid)
```



**Styles** Mit `plt.style.available` werden die verschiedensten Styles in Matplotlib angezeigt. Unter Einbindung von seaborn stehen mehr Styles zur Verfügung.

```
1 # %% Choose Style
2 plt.style.available
```

```
['Solarize_Light2',
 '_classic_test_patch',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn',
 'seaborn-bright',
 'seaborn-colorblind',
 'seaborn-dark',
 'seaborn-dark-palette',
 'seaborn-darkgrid',
 'seaborn-deep',
 'seaborn-muted',
 'seaborn-notebook',
 'seaborn-paper',
```

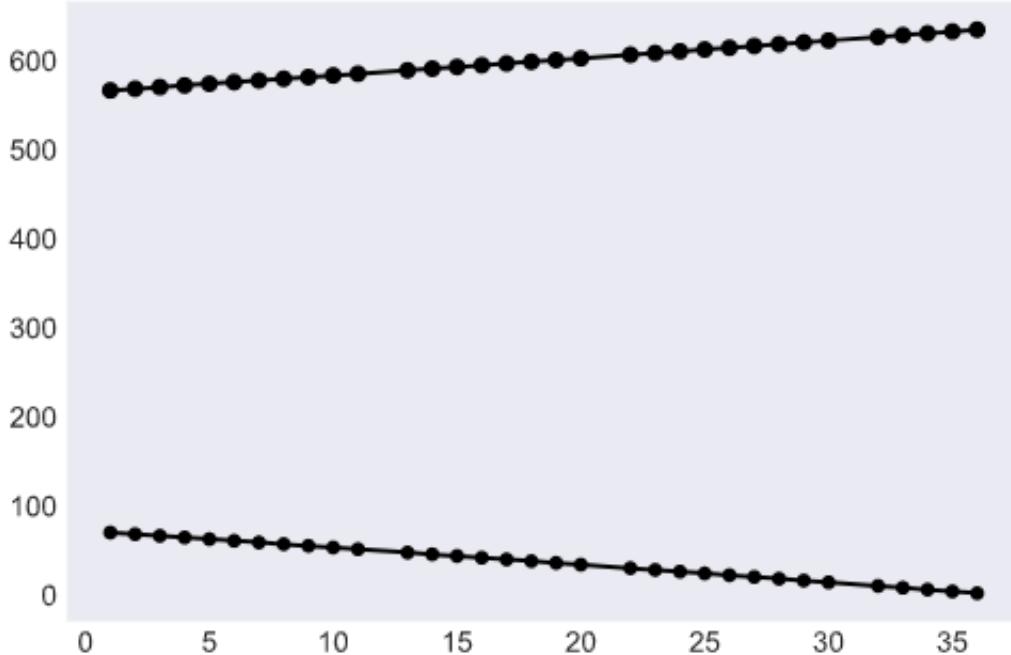
Den Style zu ändern, erfolgt über `plt.style.use()`.

```
1 # %% Choose Style
2 plt.style.use("seaborn-dark")
3 plt.plot(nds_Month, nds_prinzipal_paid)
```

## Marker Type and Colors

- marker - Einzelne Datenpunkte
- markersize - Größe von Marker
- c - Colour Template

```
1 # %% Marker and Colour
2 Grafik_1 = plt.plot(nds_Month, nds_prinzipal_paid, marker = ".",
3                     markersize = 10, c = "k")
4 Grafik_2 = plt.plot(nds_Month, nds_interest_paid, marker = ".",
5                     markersize = 8, c = "k")
```



### 1.3.3 MATLAB-Style vs Object-Oriented Style

**Subplots - Two simple Figures** Über Matplotlib können zwei verschiedene Syntaxen verwendet werden:

- **MATLAB** Style und oder
- **Objektorientierte** Syntax.<sup>8</sup>

Die Syntax wurde von MATLAB übernommen, weshalb es viele Gleichheiten mit der Python Alternative, Matplotlib, gibt.

**Matlab** Im ersten Beispiel wird dargestellt, wie in dem *MATLAB-Style* Grafiken mit Hilfe von *plt.subplot()* dargestellt werden.<sup>9</sup>

- Die Parameter *nrows*, *ncols* und *index* der Funktion *subplot()* geben an, wie viele Paneele erstellt werden.
- Dabei wird für jeden Subplot die gleich Konfiguration benötigt. Der *index* gibt dabei an, in welchem Panel der Plot angefügt werden soll.

<sup>8</sup>*Pyplot-Tutorial o. D.*, Book.Python-DSci

<sup>9</sup>Wird *plt.subplot()* ohne Argument erstellt, so wird eine Objekt mit einer Zeile und einer Spalte erstellt.

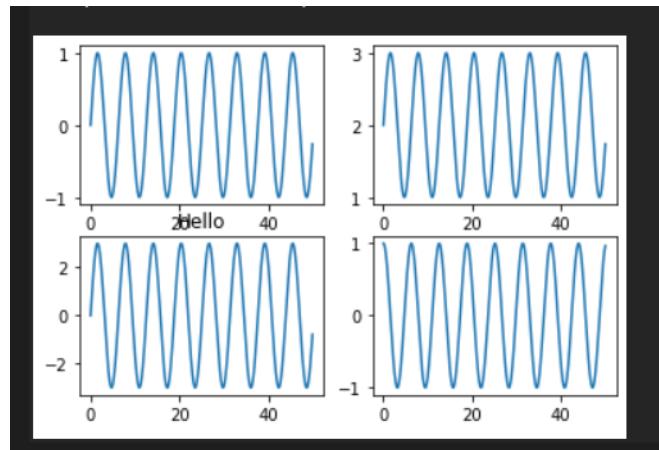
- Wird `plt.figure` nicht angegeben, wird bei dieses `plt.subplot` impliziert erstellt

```

1  %% Import Packages
2  import pandas as pd
3  import numpy as np
4  import matplotlib.pyplot as plt
5  import seaborn as sns
6
7  %% Creating Variables
8  x = np.linspace(0,50,1000)
9
10 %% Create a plot figure
11 plt.figure() # create a figure object
12
13 plt.subplot(2,2,1) # row, column, panel
14 plt.plot(x, np.sin(x))
15
16 plt.subplot(2,2,4)
17 plt.plot(x, np.cos(x))
18
19 plt.subplot(2,2,2)
20 plt.plot(x, np.sin(x)+2)
21
22 plt.subplot(2,2,3)
23 plt.plot(x, np.sin(x)*3)
24
25 plt.title("Hello") # Title for the figure
10

```

Der Nachteil dieses Styles ist es, dass es aufwendig ist, neue Panale hinzuzufügen. Es müssen alle Konfigurationen angepasst werden.



**OO** In dem objekt-orientierenden Style wird eine Figure Objekt erstellt und die Panale in dem Objekt `ax` gespeichert. Die Anzahl der Panale bestimmt die Anzahl der geplotteten Grafiken.

**ACHTUNG:** Die Funktion heißt `plt.subplots` nicht `plt.subplot`.

```

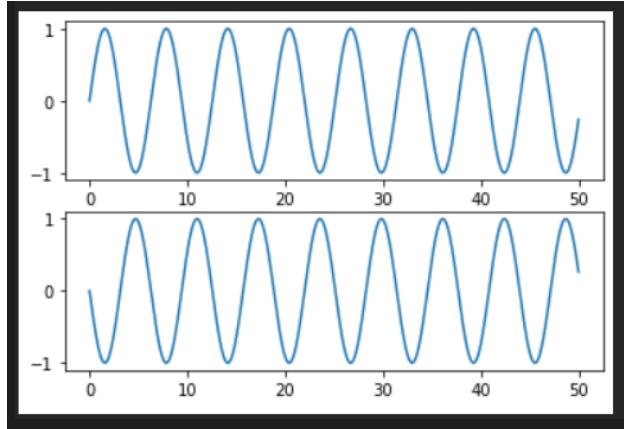
1  fig, ax = plt.subplots(2)
2
3  ax[0].plot(x, np.sin(x))

```

---

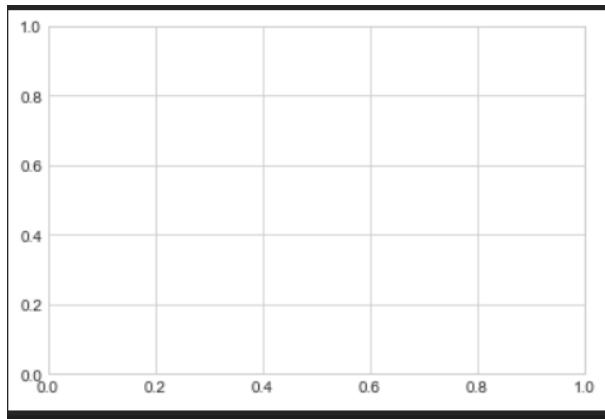
<sup>10</sup>Python Data Science Handbook o. D., S. 240

```
4     ax[1].plot(x, np.sin(-x))
```



**Simple Linie Plot** Die Instanzen der Objekte Figure und Axes sehen wie folgt leer aus.

```
1 # %% Seaboard Style
2 plt.style.use('seaborn-whitegrid')
3
4 figure_1 = plt.figure() # Create instance of the object figure
5 axis_1 = plt.axes() # Create instance of the object axes
```



Dabei produziert der Befehl ohne `plt.figure` eine Grafik, hingegen anders herum nicht.

Die Instanz `Figure` kann so verstanden werden, dass es alle Instanzen von `axis`, `label`, `title` und `graphics` umschließt.

Im Folgenden soll die Funktion  $f(x) = \sin(x)$  mit  $x \in [0, 10]$  in den zwei Formen abgebildet werden. Im objekt-orientierten Style wird dies über die Objekte `Axis` und `Figure` dargestellt:

```
1 # %% Seaboard Style
2 plt.style.use('seaborn-whitegrid')
3
4 x_sin = np.linspace(0,5,num=1000) # Generate Input
5
6 figure_1 = plt.figure() # Create instance of the object figure
7 axis_1 = plt.axes() # Create instance of the object axes
8
9 axis_1.plot(x, np.sin(x))
```

Ohne dies kann es über

```
1 plt.plot(x,np.sin(x))
```

angesteuert werden.

**Same Function in Both Styles** Der Prefix lautet *plt.* oder *ax.*

- `legend()`
- `plot()`
- etc.

**Different Function in Both Styles** Es gibt Funktionen, die den gleichen Inhalt vermitteln, aber unterschiedlich Syntax besitzen.

- Achsen Grenze bestimmen:

```
1 # M
2 plt.plot()
3 plt.xlim()
4 # OO
5 ax.plot()
6 ax.set_xlim()
```

- Achsen Titel bestimmen:

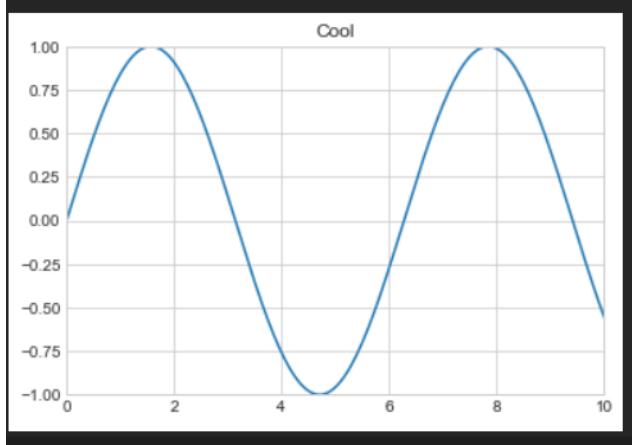
```
1 # M
2 plt.plot()
3 plt.xlabel()
4 # OO
5 ax.plot()
6 ax.set_xlabel()
```

- Grafik Titel bestimmen:

```
1 # M
2 plt.plot()
3 plt.title()
4 # OO
5 ax.plot()
6 ax.set_title()
```

Der Vorteil gegenüber de Matlab-Notierung ist, dass über die `.set()` Funktion die einzelnen Bestanteile gemeinsam aufgerufen werden können.

```
1 # %% Seaborn Style
2 plt.style.use('seaborn-whitegrid')
3
4 x_sin = np.linspace(0,5,num=1000) # Generate Input
5
6 figure_1 = plt.figure() # Create instance of the object figure
7 axis_1 = plt.axes() # Create instance of the object axes
8
9 axis_1.plot(x, np.sin(x))
10
11 axis_1.set(xlim=(0,10), ylim=(-1,1), title="Cool")
```



## 2 Model Metric

### 2.1 Performance

#### 2.1.1 Regression Analysis

Die gängigsten Beurteilungen uns Messgrößen für die Aussakraft von Regressionsmodellen im Bereich Machine Learning (ML) , sind:

**Empirisches Bestimmtheitsmaß ( $R^2$ ) (Engl.: Coefficient of Determination)** Gegeben eines Vektor  $\mathbf{Y}$  mit der Länge  $n$ , welcher als unabhängiger Teil eines Datensatzes eines Regressionsmodells fungiert, gibt es einen Vektor  $\hat{\mathbf{Y}}$ , welcher die Vorhersagewerte zu jedem Datenpunkt  $i$  beinhält. Ein *Residualwert* ist  $\epsilon_i = y_i - \hat{y}_i$ . Der *Durchschnitt mean* der beobachteten Werte wird definiert als  $\bar{y} = \frac{1}{n} \sum y_i$ .

#### Definition VIII.1: empirisches Bestimmtheitsmaß

Sei  $y_i$  beobachtet Werte  $i \in \{1, \dots, n\}$  der unabhängigen Variabel  $Y$  eines Regressionsmodells. Sei  $\hat{y}_i$  der geschätzte Werte für den gegebenen Datensatz  $i$ .

- Die erklärenden Quadratsumme (SQE)/ (Engl.: Residual Sum of Squares (SSR)) wird definiert durch

$$\sum_i \epsilon_i^2 = \sum_i (y_i - \hat{y}_i)^2 \quad (2.1)$$

- Die totale Quadratsumme (SQT)/ (Engl.: Total Sum of Squares (SST)) wird definiert durch

$$\sum_i (y_i - \bar{y})^2 \quad (2.2)$$

mit  $\bar{y} = \frac{1}{n} \sum y_i$ .

Für das *empirische Bestimmtheitsmaß  $R^2$*  gilt

$$1 - \frac{SQE}{SQT} \quad (2.3)$$

**Mittlere quadratischer Fehler (MQE) (Engl.: Root Mean Square Error (MSE))** Um die Verzerrung eines Punktschätzers kann mit Hilfe von MSE bestimmt werden.

#### Definition VIII.2: I

halt...

### Definition VIII.1: Parameterfunktion

sd

- Root Mean Square Error (RMSE)

$$RMSE = \sqrt{MSE(\hat{\theta})} = \sqrt{E((\hat{\theta} - \theta)^2)}$$

- Mean Absolute Error

$$MAE = \frac{\sum |y_i - x_i|}{n}$$

## 3 Theory of Regression, Clustering and Algorithmen

Anwendungsfälle: in Learning/Scikitlearn - 3 hours

### 3.1 \*Clustering: K-Means

#### 3.1.1 Introduction

- Die Grundidee ist, einen Datensatz in  $k$  Partitionen zu teilen, sodass die Summe der quadrierten Abweichungen von den Cluster-Schwerpunkten minimal ist.

$$J = \sum_{i=1}^k \sum_{x_j \in S_i} \|x_j - \mu_i\|^2 \quad (3.1)$$

mit den Datenpunkten  $x_j$  und den Schwerpunkten  $\mu_i$  der Cluster  $S_i$ .

- Man spricht von *Clustering durch Varianzminimierung*.
- Der Ausdruck  $\|x_i - \mu_i\|^2$  beschreibt die Euklidische Distanz zwischen dem Datenpunkt und dem Schwerpunkt  $\mu_i$  des jeweiligen Clusters  $S_i$ .
- Zum Anfang werden  $k$  Cluster ausgewählt.
- Das Ziel ist, dass jedes Objekt am Ende einem Clusterschwerpunkt zugewiesen wird.
- Schwachstellen des Algorithmus ist, dass die mehrere Lösungen gefunden werden können. Diese hängen von der Wahl der Startpunkt für  $\mu_i$  ab.

Es gibt verschiedene Algorithmen die die Möglichkeit anbieten, mehrere Iterationen durchzuführen, mit verschiedenen Startwerten.

- Eine weitere Schwachstelle ist, dass die Anzahl von  $k$  am Anfang festgelegt werden muss. Welches  $k$  das richtig ist, kann somit nach den jeweiligen Iterationen passieren.
- Für beide Schwachstellen hilft der *Silhouettenkoeffizient*.
- Der Algorithmus sucht stets nach konvexen Clustern. Andere Algorithmen können auch andere anders geformte, dichtebasierende Cluster finden, z. B. *DBSCAN*.
- Weil der Algorithmus jeden Punkt einem Cluster zuweist, können Außreiße das Ergebnis verfälschen. Eine *Noisereduction* ist vorher durchzuführen oder andere Algorithmen auszuwählen.

#### 3.1.2 Silhouette

Eine Silhouette gibt Auskunft, wie gut die Zuordnung einer Beobachtung  $o$  zu den nächstgelegenen Clustern  $A$  und  $B$ .

Darauf aufbauend ist der *Silhouettenkoeffizient*. Dies ist eine Maßzahl für die Qualität der Cluster unabhängig deren Anzahl.

**Silhouette** Gehört  $o$  zum Cluster  $A$ , so ist die *Silhouette* von  $o$  definiert als:

$$S(o) = \begin{cases} 0 & , \text{ wenn } o \text{ einziges Element von } A \text{ ist} \\ \frac{dist(B, o) - dist(A, o)}{\max \{dist(B, o), dist(A, o)\}} & , \text{ sonst} \end{cases} \quad (3.2)$$

Die Distanz zwischen dem Objekt  $o$  und dem Cluster  $A$  sowie  $B$  wird durch die maximale Distanz. Daraus folgt, dass  $S(o)$  zwischen  $-1$  und  $1$  liegt.

- $S(o) < 0$ , dann gehört das Objekt zum nächstgelegenen Cluster  $B$ . - Das Cluster kann verbessert werden oder Außreißer entfernt.
- $S(o) \approx 0$ , dann liegt das Objekt zwischen beiden Clustern.
- $S(o) \approx 1$ , dann liegt das Objekt im richtigen Cluster.

Die Berechnung der Distanz zwischen dem Objekt  $o$  und dem Cluster  $A$  erfolgt wie:

$$dist(A, o) = \frac{1}{n_A - 1} \sum_{a \in A, a \neq o} dist(a, o) \quad (3.3)$$

als der Mittelwert der Distanz zwischen *allen anderen Objekten* im Cluster  $A$  und dem Objekt  $o$ . Die Anzahl der Objekte im Cluster  $A$  wird mit  $n_A$  definiert.

Analog wird die Distanz zum nächstgelegenen Cluster  $B$  berechnet als die minimale durchschnittliche Distanz

$$dist(B, o) = \min_{C \neq A} \underbrace{\left( \frac{1}{n_c} \sum_{c \in C} dist(C, o) \right)}_{=dist(C, o)} \quad (3.4)$$

Dabei wird für alle Cluster, die das Objekt  $o$  nicht enthalten<sup>11</sup> die Distanz  $dist(C, o)$  berechnet. Der nächstgelegene Cluster ist der mit der kleinsten Distanz.

**Silhouettenkoeffizient** Der Silhouettenkoeffizient ist definiert als

$$s_C = \frac{1}{n_C} \sum_{o \in C} S(o) \quad (3.5)$$

arithmetisches Mittel aller  $n_C$  Silhouetten eines Clusters  $C$ . Diese Koeffizient kann für jeden Cluster oder den gesamten Datensatz berechnet werden.

### 3.1.3 Test Data Sets

Neue Algorithmen können problemlos arbeiten, diese zu testen, ob sie das tun, was sie tun sollen, ist nicht immer erkennbar.

Dafür hat *scikit* Datensatz Pakete. Diese ermöglichen spezifischen Datensätze zu generieren, die

- in ihren Hyperparameter angepasst werden
- und skalierbar sind.

Im folgenden sollen drei Problem von *Klassifikation* aufgegriffen. Dabei beschreibt *Klassifikation*, dass einem Teilmengen eines Datensatzes *Labels* zu gewissen werden.

---

<sup>11</sup>Weil  $o$  in  $A$  liegt.

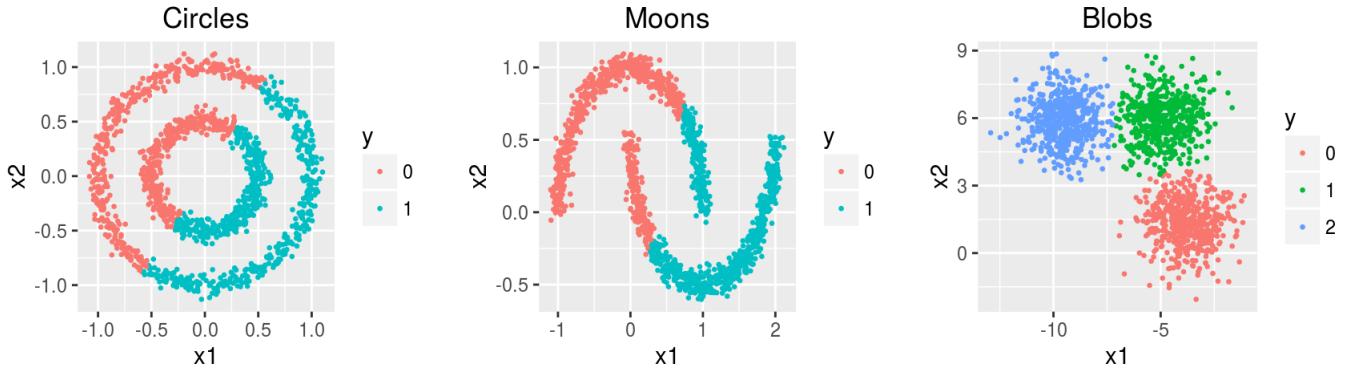


Abbildung 243: Blobs, Moons, Circles

**Blobs Classification Problem** Die Funktion `make_blobs()` generiert einen Datensatz, welcher  $n$  Cluster beinhaltet. Dabei werden die Cluster zufällig gesetzt und die Datenpunkte um die Zentren normalverteilt. Die unabhängigen Dimensionen, oder auch *Features* genannt, können ebenso bestimmt werden. Eine Visualisierung von zwei bis drei Features ist nur möglich. Die Labels werden ebenfalls als zweiter Output mit ausgegeben.

```

1 # Generate classification dataset
2 X, labels = make_blobs(
3     n_samples=100, #n-datapoints equalie devided amoung the centers
4     centers=2, #centers in the dataset
5     n_features=2 #columns return
6 )

```

Im Folgenden wird eine Größe der *Features* höher 2 nicht betrachtet, weil der Plot auf 2D ausgelegt ist.

	0	1
0	7.256785...	-8.48941...
1	5.966333...	-11.7388...
2	8.466035...	-7.44978...
3	-0.10425...	6.095778...
4	-1.06028...	5.199687...
5	-1.82802...	5.105729...
6	6.995060...	-9.34214...
7	7.677998...	-9.85031...
8	-1.89933...	4.054489...
9	6.469902...	-9.36599...
10	5.160248...	-8.67499...
11	7.513579...	-9.03352...
12	6.739191...	-7.27909...
13	7.277591...	-8.18505...

Abbildung 244: Die Variable  $X$  sieht mit  $n_{features} = 2$  wie folgt aus

	0	1	2	3
0	4.775104...	10.08805...	-7.25475...	7.939298...
1	2.852507...	9.663155...	-5.79071...	7.096490...
2	6.686783...	0.199314...	7.576286...	4.708144...
3	8.495404...	1.687908...	-3.29883...	4.706539...
4	3.186371...	10.04269...	-4.89292...	7.563860...
5	9.053116...	-7.95417...	7.789004...	1.412568...
6	5.858198...	7.828060...	-6.28722...	6.169989...
7	4.001935...	-1.52300...	8.080340...	5.311498...
8	8.831817...	3.395334...	-3.58928...	2.407640...
9	7.956215...	0.067946...	-1.21186...	2.902388...

Abbildung 245: Die Variable  $X$  sieht mit  $n_{features} = 4$  wie folgt aus

Der Testdatensatz ist somit erstellt. Für die Plotung wird diese im Folgenden vorbereitet.

```
1 # Collect all in one df
2 df = Dataframe(dict(x = X[:,0], y = X[:,1], label = y))
12 13 14
```

Für die unterschiedlichen Labels werden eine *Series* erstellt, welche verschiedenen Farben für die verschiedenen Clusters.

Für die grafische Bewertung wird eine Farbpalette zusammengestellt.

```
1 # scatter plot, dots colored by class value
2 colors = {0:'red', 1:'blue', 2:'green',3:'yellow'}
```

## 3.2 \*Working with Predictive Analytics

### 3.2.1 Data Preprocessing: Categorical Values

Prozesse und Algorithmen im ML Bereich benötigen als Input numerische Werte. Daten diese dies nicht bieten können, müssen in solche kodiert werden.

Es gibt dafür zwei bestehende Methoden im sklearn Paket:

```
• from sklearn import preprocessing
  le = preprocessing.LabelEncoder() # Create a class object
```

<sup>12</sup>class pandas.DataFrame(data=None, index=None, columns=None, dtype=None, copy=False)

- **data** ndarray (structured or homogeneous), Iterable, dict, or DataFrame  
Dict can contain Series, arrays, constants, dataclass or list-like objects. If data is a dict, column order follows insertion-order.
- **index** ...

<sup>13</sup> $X[:, 0]$  bedeutet, dass alle Zeile : und die erste Spalte mit Index 0 ausgewählt wird.

<sup>14</sup>Die Funktion *dict()* liefert drei Werte ein  $x$ ,  $y$  und *label*. Die Auslesung erfolgt über die Ausgabe von  $X$  und  $y$ . Dabei wird  $X[Zeile, Spalte]$  angewandt, um die entsprechenden Werte auszulesen. Bsp.:  $X[1; 1] >>> -8, 2313234234$

**Label Encoder** Der Preprocessor von scikit learn bietet eine Classe an, welche Kategorische Werte eines Arrays in numerische umwandelt. Dies ist notwendig, weil ML Modelle nur numerische Werte einlesen. Haben die Werte eine ordinale Struktur, kann ein ML Modell diese abbilden. Ist dies jedoch nicht vorhanden, so bietet One Hot Encoder eine bessere Struktur an.

```

1 ##### Python Libraries
2 import pandas as pd
3 import numpy as np
4 from sklearn.model_selection import train_test_split
5 from sklearn.impute import SimpleImputer
6 import matplotlib.pyplot as plt
7
8 # Label Encoder
9 from sklearn.preprocessing import OneHotEncoder
10 from sklearn.preprocessing import LabelEncoder
11 from sklearn.compose import ColumnTransformer
12
13 ##### Import Date
14 data = pd.read_csv("insurance.csv")
15
16 ##### EDA
17
18 #print(data.head(3))
19 #print(data.describe())
20 #print(data["bmi"].unique())
21 #data_nan = data.isnull().sum()
22 #print(data_nan)
23
24 ##### Data Preparation
25
26 data["bmi"].fillna(0, inplace = True) # replace None
27 #print(data.head(3))
28 data_mod = data
29
30 ##### Label Encoding
31 LE = LabelEncoder()
32 data_mod["sex"] = LE.fit_transform(data["sex"]) # input is an array
33 data_mod["region"] = LE.fit_transform(data["region"])
34 data_mod["smoker"] = LE.fit_transform(data["smoker"])
35 print(data_mod.iloc[:,1:].head())

```

Im Klassenobjekt werden die Kategorien wie weitere Funktion zum Input Array gespeichert. Wir das Klassen-objekt auf eine anderes Array angesetzt, so werden die vorherigen Daten überschrieben.

```

1 LE = LabelEncoder()
2 data_mod["sex"] = LE.fit_transform(data["sex"]) # input is an array
3 data_mod["region"] = LE.fit_transform(data["region"])
4 data_mod["smoker"] = LE.fit_transform(data["smoker"])
5 print(data_mod.iloc[:,1:].head())
6
7 # Inverse encoder
8 print(LE.classes_)
9 #data_mod["New"] = LE.inverse_transform(data_mod["region"]) #Fuehrt zum
# Fehler
10 data_mod["New"] = LE.inverse_transform(data_mod["smoker"])

```

**One Hot Encoder** Um eine Ordnung in den transformierten Daten zu vermeiden, wenn keine existiert, werden Dummy Variablen angelegt.

Im ersten Schritt wird der Konstruktor geschaffen, dann die jeweiligen Operationen durchgeführt.

```
1 O_OHE = OneHotEncoder(categorical_features=[0])  
2 O_OHE.fit_transform1
```

Das label-encodedet Array wird aufsplittet in die Anzahl Spalten wie das Array eindeutige Kategorien hat.

## Teil IX

# Machine Learning Theroy

# 1 Logistische Regression

## 1.1 Overview

Um Klassifikationsprobleme zu lösen, wobei die abhängige Variable  $Y$  diskret ist, ist eins der gängigsten Modell die Logistische Regression. Die *Logistische Regression* teilt sich in

**Binäre** Die kategorische Zielvariable hat nur zwei Ausprägungen

**Multinomial** Die kategorische Zielvariable hat mehr als zwei Ausprägungen

**Ordinal** Die Ausprägungen der Zielvariablen habe eine ordinale Skalierung

auf. Der Fokus in diesem Kapitel wird auf der *Binäre Klassifikation* sein.

Warum die Lineares Regressionsmodell (LRM) in unter diesen Bedingungen keine plausiblen Ergebnisse liefern kann, liegt an daran, dass Wahrscheinlichkeiten von über 100 und unter 0 Prozent erreicht werden können. Für für  $Y$  nur Ausprägungen von 0 oder 1 erreicht werden können. Könnte eine LRM mit  $Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$  Werte von größer 1 oder kleiner 0 erreichen.

Als Exkurs wird gezeigt, wie polynomische Terme in der Hypothesenfunktion helfen, Entscheidungsgrenze genauer um die Daten zu legen.

## 1.2 General-Lineare-Model

### 1.2.1 Model

Mit der Erweiterung des LRM, kann ein bereiteres Spektrum an Fragestellungen, mit den Methoden der Linearen Regression beantwortet werden.

#### Definition IX.1: Generalisiertes lineares Modell

Man spricht von einem General Linear Model (GML), wenn folgenden Annahmen gelten:

- Die Beobachtungen  $y_i$  mit  $i = 1, \dots, n$  von  $Y$  können als voneinander unabhängigen Realisationen der Zufallsvariablen  $Y_i$  mit unbekannten Erwartungswert  $\mu_i := E[Y_i]$  angesehen werden.
- Die Zufallsvariablen  $Y_i$  stammen alle aus der gleichen Verteilung, deren Parameter für jede Beobachtung  $i$  unterschiedlich sein können, und welche aus der Familie der Exponentialfamilie von Verteilungen stammt.
- Es gibt eine monotone, differenzierbare Funktion  $g$ , sodass für die erklärenden Variablen  $X_j$  gilt

$$g(\mu_i) = h(\theta_1, \dots, \theta_n; X_{i,1}, \dots, X_{i,m}) + \varepsilon_i \quad (1.1)$$

für  $i = 1, \dots, n$  und  $j = 1, \dots, m$  erfüllt ist.

Das GML teilt sich auf, in einen **linearen Schätzer** (engl. *linear Predictor*)

$$\eta_i = h(\theta_1, \dots, \theta_n; X_1, \dots, X_n) \quad (1.2)$$

welcher lineare in den Parametern  $\theta_i$  ist, und die erklärenden Variablen  $X_j$  mit  $j \in \mathbb{N}$  in der bekannten LRM Struktur vereint. Es ist dabei wichtig zu beachten, dass die Hypothesenfunktion  $h()$  nicht  $Y_i$  beschreibt, sondern  $\eta_i$ .

Die Verbindung zwischen  $Y_i$  und  $h()$  wird mit **Link-Funktion** gesichert. Je nach Verteilung sind andere Überlegungen für  $g()$  notwendig. Es bleibt jedoch bestehen

$$g(\mu_i) = \eta_i \quad (1.3)$$

$$\text{mit } E[Z_i] = \mu_i. \quad (1.4)$$

In den Spezialfällen wird ersichtlich, dass es verschiedenen Funktion für  $g$  geben kann.<sup>15</sup>

<sup>15</sup>Allgemein: Es gibt zwei Sichtweisen für eine statistische Fragestellung: **Erwartung des Zielvariablen**  $Y$  für die Ausprägung  $i$ , sodass

### 1.2.2 Unterschied Lineares Model und General-Lineare-Model

Das GML ist eine Erweiterung des LRM (Linearen Regressionsmodell).<sup>16</sup> Ein Schwerpunkt ist, die Zielvariable  $Z$ <sup>17</sup>, welche nicht normalverteilt sind. Ziel ist es für Daten, welche binäre oder kategorische Ausprägungen haben, die verfügbaren Werkzeuge des LRM einzubinden. Es handelt sich somit um eine Überkategorie des bekannten LRM. Dieser stellt ein Spezialfall des GML dar, wie im Folgenden zu sehen ist. Im Weiteren werden sich auf drei Erweiterungen fokussiert, welche den Vergleich mit dem Spezialfall LRM aufzeigen:

**Verteilung Zielvariable** In LRM ist  $Y$  normalverteilt. Dies wird unter GML erweitert, und  $Z$  ist exponential verteilt. Zu der Familie der Exponential Verteilungen gehört auch die Normalverteilung. Weitere Verteilungen sind Binomial, Gamma, etc.. Die Aussagekraft eines GML wird damit um diskrete Fragestellungen erweitert.

**Hypothesenfunktion** Anders als bei LRM wird die  $Y$  nicht direkt durch die Hypothesenfunktion beschrieben. Die lineare Hypothesenfunktion bezieht sich auf eine **Predictor (Schätzer)  $\eta$** :

$$\eta_i = h(\theta_1, \dots, \theta_n; X_1, \dots, X_n) \quad (1.7)$$

**Link Funktion** Um den *Predictor* mit der  $Y$  zu verbinden, wird eine beliebig gewählte Link-Funktion  $g$ <sup>18</sup> gewählt. Diese stellt die Beziehung zwischen dem Erwartungswert von  $Z$  und dem *Predictor* her:

$$E[Y_i] = g(\eta_i) \quad (1.8)$$

**Verteilung der Fehler** Für ein LRM wird angenommen, dass der Fehler  $\varepsilon_i$  bei Messungen von  $Y_i$  unabhängig und identisch verteilt ((engl.) independent and identically distributed) (i.i.d) bei jeder Ziehung  $i$  ( $i \in \mathbb{N}$ ) ist, sodass

$$E[\varepsilon] = 0 \quad (1.9)$$

$$\text{und } Var[\varepsilon] = 0. \quad (1.10)$$

Für die Verteilung wird angenommen, dass  $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ . In GML kann keine homoskedastisch angenommen werden, weil für die Fehler mit Zu- oder Abnahme der abhängigen Variablen sich ändern können.

### 1.2.3 Spezialfall: Lineares Model im General-Linearen-Model

Sei  $\varepsilon_i \sim N(0, \sigma^2)$  und  $g$  gleich der Identitätsabbildung, folgt

$$\eta_i = E[\mu_i] = \mu_i \quad (1.11)$$

### 1.2.4 Spezialfall: Binäre Verteilung im General-Linearen-Model

Sei  $Y_i \sim \text{Binomial}(n_i, p_i)$  und  $Z_i = Y_i/n_i$ , es folgt

$$E[Y_i/n_i] = p_i \text{ und } Var[Y_i/n_i] = \frac{1}{n_i} p_i (1 - p_i) \quad (1.12)$$

---

gilt

$$E[Y_i] = h(\theta_1, \dots, \theta_n; X_1, \dots, X_n) \quad (1.5)$$

Oder als **Konstruktion der Messung der systematischen und stochastischen Komponenten des Modells**

$$Y_i = h(\theta_1, \dots, \theta_n; X_1, \dots, X_n) + \varepsilon_i. \quad (1.6)$$

Zu der Interpretation des Erwartungswertes wird in der Schätzung als *Regression von  $Y$  gegeben  $X$*  neu formuliert.

<sup>16</sup>Quellen für diese Kapitel sind *log.1 o. D., log.2 o. D., log.3 o. D., log.4 o. D., log.5 o. D., log.6 o. D., log.7 o. D., log.8 o. D., log.9 o. D., log.10 o. D., log.11 o. D., log.12 o. D., log.13 o. D., log.14 o. D., log.15 o. D., log.16 o. D., log.17 o. D.*

<sup>17</sup>In der Literatur zu dem Thema wird die Zielvariable mit  $Y$  notiert, selbst wenn die Zielvariable eine Zusammensetzung mehrerer Zufallsvariablen ist, wir ein Teil meist mit  $Y$  abkürzt.

<sup>18</sup>In der Literatur zu dem Thema wird der Unterschied von einer Funktion zu einer Abbildung kaum getroffen. Weshalb im in dem Kapitel Funktion ebenfalls Abbildung bedeuteten kann.

Die *Link-Funktion* ist die *Logit-Funktion* mit  $\text{logit}(x) = \log\left(\frac{x}{1-x}\right)$ :

$$g : \begin{cases} (0, 1) & \rightarrow (-\infty; \infty), \\ \mu_i & \mapsto g(\mu_i) \end{cases} \quad \text{mit } g(\mu_i) = \text{logit}(\mu_i) \quad (1.13)$$

## 1.3 Hypothesenfunktion

### 1.3.1 Regressionsmodell

In der Regression Statistik wird der Zusammenhang zwischen der Zielgröße  $Y$  und den abhängigen Variablen  $X_1, \dots, X_m$  dargestellt.<sup>19</sup> Hierbei ist  $Y_i$  eine Abbildung für jede Ziehung eines Zufallsexperiments. Es handelt sich um eine Zufallsvariable, sie 2.2. In der Regressionsanalyse wird eine funktionalen Beziehung  $Y_i = h(X_{i,1}, \dots, X_{i,m}) + \varepsilon_i$  angenommen. Das Ziel ist, eine Funktion zu schätzen, welche die Daten am nächsten abbilden kann. Für  $\varepsilon_i$  gilt, dass es alle Störungen aufgreift, welche funktional nicht abgebildet werden können.<sup>20</sup>

In dem Prozess der Schätzung, wird der Funktionszusammenhang um weitere Variablen  $\Theta_1, \dots, \Theta_n$  erweitert  $\rightarrow h(X_{i,1}, \dots, X_{i,m}; \Theta_1, \dots, \Theta_n)$ . Für  $\Theta_j$  gilt, dass es sich um Variablen handelt, welche für jede Ziehung konstant bleiben: diese nennen wir *Parameter*. Deshalb kann die Hypothesenfunktion  $h$  aus der Perspektive mit und ohne Parameter betrachtet werden. Wird  $Y_i$  beschrieben durch  $h$  und  $\varepsilon_i$ , erhält  $Y_i$  Zufallsvariablen-Struktureigenschaften durch  $\varepsilon_i$ , weil es sich hierbei um eine Zufallsvariable handelt, für die gilt  $\sim \mathcal{N}(0, \sigma^2)$ . Dies Annahme über die Verteilung der Störung wir mit der Erweiterung *GML* abgeändert, jedoch bleibt  $\varepsilon_i$  eine Zufallsvariablen.

Für  $X$  kann angenommen werden, dass es sich hierbei um eine stochastische Variablen handelt, diese müsste dann die Bedingungen einer Zufallsvariablen erfüllen. Handelt es sich bei  $X_i$  um eine Zufallsvariable, so müssen die folgenden Annahmen für ein Regressionsmodell gelten:

$$E[\varepsilon_i | X_i] = 0, \quad (1.14)$$

$$(X_1, Y_1), \dots, (X_n, Y_n) \text{ sind i.i.d.,} \quad (1.15)$$

$$E[X_i^k] < \infty \text{ und } E[Y_i^k] < \infty \text{ für alle } k \in \mathbb{N} \text{ und } \text{Var}[X_i] > 0. \quad (1.16)$$

In diesem Fall spricht man von einem **bedingt heteroskedastischen Modell**. Wird die Annahme 1.14 durch  $E[\varepsilon_i | X_i] = 0$  und  $\text{Var}[\varepsilon_i | X_i] = \sigma_\varepsilon^2$  ersetzt, spricht man von einem bedingten **homoskedastisches Modell**. Wie im oberen Abschnitt erwähnt, gilt  $\varepsilon_i | X_i \sim \mathcal{N}(0, \sigma_\varepsilon^2)$  spricht man von einem **klassischen Modell**.

#### Definition IX.2: Regressionsmodel

Der Funktionszusammenhang für die abhängige und unabhängigen Variablen heißt *Regressionsmodel*<sup>a</sup> und für diesen gilt:

$$Y_i = h(X_{i,1}, \dots, X_{i,m}; \Theta_1, \dots, \Theta_n) + \varepsilon_i. \quad (1.17)$$

Die Funktion  $h$  bildet die Beziehung von  $X_i = [X_{i,1}, \dots, X_{i,m}]$  und  $\Theta = [\Theta_1, \dots, \Theta_n]$  ab und wird *Hypothesenfunktion der Regression* bezeichnet. Jede Messstörung  $i$  wird unter der Zufallsvariablen  $\varepsilon_i$  aufgegriffen.

<sup>a</sup>Der Begriff Regression bezieht sich auf stetige Erwartungsfunktion der Zielvariablen  $Y_i$ . Ist die Zielvariable diskret, so wird von *Klassifizierung* gesprochen.

### 1.3.2 Lineare Regression

Als Synonym für das Regressionsmodell wird auch nur *Regression* verwandt.<sup>21</sup>

<sup>19</sup>Die Begrifflichkeiten von Zielgrößen und Variablen werden in fachübergreifender Literatur nicht immer eindeutig benannt. Um es mit dem Konzept der Abbildung zu vereinigen, siehe ??

<sup>20</sup>Bei  $\varepsilon$  handelt es sich um eine Zufallsvariable. Konvention ist, dass Zufallsvariablen Großbuchstaben erhalten. Mit dieser Konvention wird hier gebrochen.

<sup>21</sup>Die Parameter werden üblicherweise mit  $\beta$  geschrieben.

### Definition IX.1: Lineares Regressionsmodell

Ist  $h$  lineare in den Parametern  $\Theta$ , so spricht man von einem *linearen Regressionsmodell*.

Die Bezeichnung der Linearität bezieht sich auf die Parameter, wie im Exkurs angezeigt, kann das Modell auch nicht linear sein. Selbst wenn der Zusammenhang zwischen  $Y_i$  und  $X_i$  per *multipler Regression* oder *polynomial Regression* beschrieben wird, so ist meist das Regressionsmodell linear in  $\Theta$ .

### Beispiele

- Mit  $h = \Theta_0 + \Theta_1 X_{i,1} + \Theta_2 X_{i,2}$  liegt ein *Multiple Regressions-Hypothesenfunktion* vor.
- Sei  $h = \Theta_0 + \Theta_1 \log(X_i)$  ist die Regressionsmodell linear, aber der Funktionszusammenhang zwischen  $Y_i$  und  $X_i$  nicht.
- Für *Polynomische Regression* gilt dies ebenfalls. Im einfachen Fall mit einer abhängigen Variablen  $X_i$ ,  $p \in \mathbb{N}$  und  $h = \Theta_0 + \Theta_1 X_i^1 + \dots + \Theta_p X_i^p$  ist der Zusammenhang zwischen  $X_i$  und  $Y_i$  nichtlinear, aber für  $h$  in  $\Theta$  schon.
- Ebenfalls für *Mehrdimensionale Polynome* im Regressionsmodell mit beispielsweise  $h = \Theta_0 + \Theta_1 X_{i,1} + \Theta_1 X_{i,1}^2 + \Theta_2 X_{i,1} X_{i,2} + \Theta_1 X_{i,1}^2 + \Theta_1 X_{i,2}^2$  ist  $h$  lineare in  $\Theta$ .

### 1.3.3 (Exkurs) Nichtlineare Regression

In der *nichtlineare Regression* werden Funktionen  $h$  untersucht, welche sich nicht als lineare Funktionen in den Parametern schreiben lassen. Die Modellierung der Hypothesenfunktion leitet sich oft direkt aus der Theorie ab. Zum Beispiel wird für den Zusammenhang der Subrat-Konzentration  $x$  (in ppm) die Funktion Michealis-Menten-Funktion herangezogen, um das Regressionsmodell mit

$$h(\mathbf{X}_i, \Theta) = \frac{\Theta_1 X_i}{\Theta_2 + X_i} \quad (1.18)$$

zu modellieren. Für den biochemischen Sauerstoffverbrauch in [mg/l] wird

$$h(\mathbf{X}_i, \Theta) = \Theta_1 (1 - e^{-\Theta_2 X_i}) \quad (1.19)$$

modelliert.

### Definition IX.3: linearisierbar

Ein nichtlineares Regressionsmodell, welches durch monotone Transformation der Zielvariablen und der Ausgangsvariablen in eine lineare Funktion in den Parametern überführt werden kann, wird *linearisierbar* genannt.

Für  $h = \Theta_1 X_i^{\Theta_2}$  wird die Transformation mit ln angewandt. Es folgt

$$\ln(\Theta_1) + \Theta_2 \ln(X_i) \quad (1.20)$$

$$\rightarrow \beta_0 + \beta_1 \ln(X_i) \text{ mit } \beta_0 = \ln(\Theta_1) \text{ und } \beta_1 = \Theta_2 \quad (1.21)$$

Hiermit wird eine neue Hypothesenfunktion aufgestellt.

## 1.4 Binäre Klassifikation

### 1.4.1 Modell

Im dem Kapitel wird der Schwerpunkt auf die kategoriale Verteilung mit zwei Ausprägungen von  $Y$  gelegt. Das Logistische Regressionsmodell hat das Ziel die diskrete Zufallsvariable  $Y_i$  mit zwei Ausprägungen

$$y \in \{0, 1\} \text{ mit } 0 : \text{Negative Ausprägung der Klasse} \quad (1.22)$$

$$1 : \text{Positive Ausprägung der Klasse} \quad (1.23)$$

zu modellieren.

#### Definition IX.4: Logistisches Modell

Unter den folgenden Annahmen wird von einem **Logistischen Modell** gesprochen:

- Für die Zufallsvariable  $Y_i$  wird eine dichotome Verteilung angenommen, sodass  $E[Y_i] = p_i$  gilt.
- Sei  $\eta_i = h(\mathbf{X}, \boldsymbol{\Theta})$  gilt:

$$\eta_i = g(\mu_i), 0 < p < 1 \text{ und } g(z) = \ln\left(\frac{z}{1-z}\right) \quad (1.24)$$

$$p = g^{-1}(\eta_i), -\infty < \eta_i < \infty \text{ und } g^{-1}(z) = \frac{1}{1 + e^{-z}} \quad (1.25)$$

Die Funktion  $g^{-1}$  wird *Sigmoid Funktion* und  $g$  die *Logit Funktion* genannt.

Würde es sich um eine lineare Regression von  $Y_i$  handeln, so müsste für  $h$  gelten  $0 \leq h(\mathbf{X}, \boldsymbol{\Theta}) \leq 1$ . Mit dem Rahmen des GML muss nicht mehr  $Y_i$  modelliert werden, sondern der Schätzer  $\eta_i$ . Die Beziehung zwischen dem Erwartungswert von  $Y_i$  wird mit der Logit-Funktion hergestellt.

#### 1.4.2 Interpretation Lineares logistisches Modell

Gegeben sei eine binomialverteilte Zufallsvariable  $Y_i$  mit Erwartungswert (Erfolgswahrscheinlichkeit)  $p_i = E[Y_i/n_i]$ . Diese Variable hängt von  $\mathbf{X}$  in folgender Form ab:

$$\text{logit}(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 X_{1,i} + \dots + \beta_n X_{n,i} \quad (1.26)$$

Die Parameter von  $h$  werden mit der Maximum Likelihood-Methode geschätzt. Jeder gegebene Wert für  $\beta_i$  gibt dabei nicht direkt Auskunft über die Eintrittswahrscheinlichkeit, sondern nur über den *log-Odds*. Wird aber nach  $p_i$  aufgelöst, erhält man die gegebene Eintrittswahrscheinlichkeit.

#### 1.4.3 Exkurs: Example Malignat

Für diskrete Zufallsvariable  $Y$  – Malignat Cancer wurden folgende Kombinationen von  $(X_i, Y_i)$  gefunden: Eine

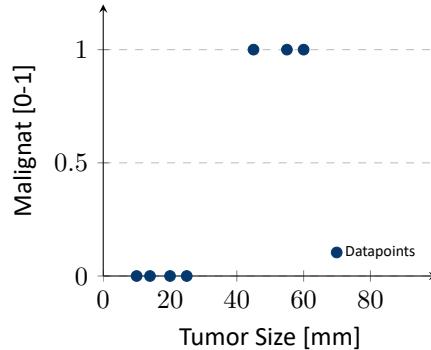


Abbildung 246: Realisation von  $(Y_i, X_i)$

Hypothesenfunktion  $h$  lineare in  $\boldsymbol{\Theta}_k$  und  $X_i$  kann nicht sicherstellen, dass geschätzte Werte  $\hat{y}$  nicht über 1 oder 0 erreicht werden. Für die Interpretation würde dies bedeuten, dass eine Wahrscheinlichkeit von über 100 % oder 0 % erreicht werden.

#### 1.4.4 Exkurs: Dummy-Variable

Seinen  $X_1$  und  $X_2$  zwei nicht-stochastische Merkmals-Variablen. Für das Regressionsmodell von  $Y_i$  wird angenommen  $Y_i = \beta_0 + \beta_1 X_{1,i} + \beta_2 X_{2,i} + \varepsilon_i$ . Für  $X_2$  gilt jetzt, dass es sich um kategorische Merkmals-Variablen

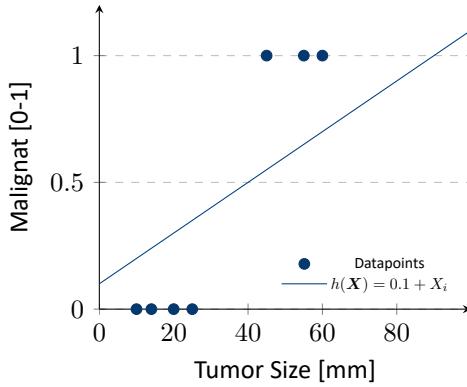


Abbildung 247: Lineare Hypothesenfunktion

handelt, eine sogenannte *Dummy-Variablen*. Beispiel:  $Y$  – Vitalkapazität,  $X_1$  – Alter und  $X_2$  – Exposition mit  $x_2 = \{0 : \text{nicht exponiert}; 1 : \text{exponiert}\}$  Für nicht exponierte Arbeiter ( $x_2 = 0$ ):

$$y_i = \beta_0 + \beta_1 x_{1,i} + \varepsilon_i.$$

Für exponierte Arbeiter ( $x_2 = 1$ ):

$$y_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \varepsilon_i.$$

Modelliert man  $X_2$  nicht als eigenen Dimension, so wird der Effekt über den Achsenabschnitt in der Regressionsgleichung erfasst. Ob  $X_2$  einen Effekt besitzt, lässt sich in der Verschiebung der Regressiongerade darstellen.

## 1.5 Decision Boundary

### 1.5.1 Simple Lineare Model

Sei  $h(\mathbf{X}, \boldsymbol{\Theta}) = \Theta_0 + \Theta_1 X_i$ . Ebenfalls wird angenommen, dass für die Vorhersage von  $y_i = 1$  muss

$$g^{-1}(h(\mathbf{X}, \boldsymbol{\Theta})) \geq 0,5 \rightarrow h(\mathbf{X}, \boldsymbol{\Theta}) > 0$$

und für  $y_i = 0$  muss

$$g^{-1}(h(\mathbf{X}, \boldsymbol{\Theta})) < 0,5 \rightarrow h(\mathbf{X}, \boldsymbol{\Theta}) < 0$$

gelten. Dies bedeutet, bei einer Wahrscheinlichkeit von größer gleich 0,5 die Vorhersage für  $y_i = 1$  getroffen wird.

Daraus lässt sich eine *Decision Boundary* Entscheidungslinie ziehen, ab welchen Punkt das Modell die Vorhersage für oder gegen das Eintreten bestimmt. Sei  $\boldsymbol{\Theta} = [-3, 1, 1]^T$  gilt für  $h(\mathbf{X}, \boldsymbol{\Theta}) = -3 + X_1 + X_2$ . Es kann jetzt die Entscheidungsgrenze bestimmt werden, bei welcher entschieden wird, wann  $y_i = 1$  geschätzt werden kann.

$$\begin{aligned} \text{Für } y = 1 \longrightarrow -3 + X_1 + X_2 &\leq 0 \\ X_2 &\leq 3 - X_1 \end{aligned}$$

Für beispielhafte Datenpunkte ergibt sich eine Entscheidungsgrenze, wobei die Punkte auf der einen oder anderen Seite farblich markiert werden, um zu signalisieren, ob gegeben der Datenpunkte, dass Modell sich für das Eintreten von  $y_i = 1$  entscheidet oder nicht.

### 1.5.2 Komplexe Polynomisches Model

Sei  $h(\mathbf{X}, \boldsymbol{\Theta}) = \Theta_0 + \Theta_1 X_i + \Theta_2 X_2 + \Theta_3 X_1^2 + \Theta_4 X_2^2$  und  $\boldsymbol{\Theta} = [-1, 0, 0, 1, 1]^T$  gilt für die Decision Boundary

$$\begin{aligned} \text{Für } y = 1 \longrightarrow -1 + X_1^2 + X_2^2 &\leq 0 \\ X_2 &\leq \sqrt{1 - X_1^2} \end{aligned}$$

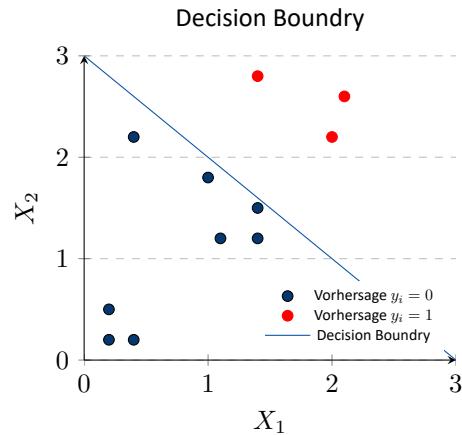


Abbildung 248: Lineare Hypothesenfunktion

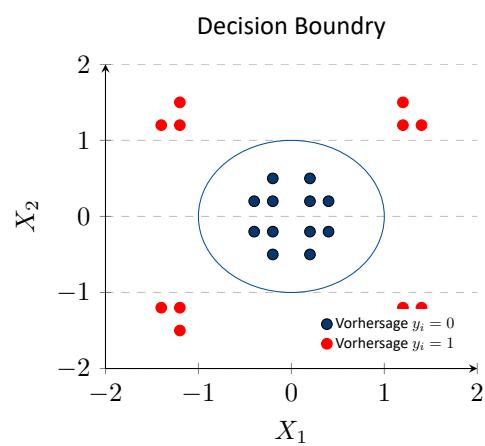


Abbildung 249: Polynom Hypothesenfunktion

## 2 Bewertungsmetriken

[https://ichi.pro/de/allgemeine-bewertungsmetriken-für-die-regressionsanalyse-82886198762157](https://ichi.pro/de/allgemeine-bewertungsmetriken-fur-die-regressionsanalyse-82886198762157)

### 2.1 Regression Analyse

#### 2.1.1 Bestimmtheitsmaß $R^2$

Gegeben eines Vektor  $\mathbf{Y}$  mit der Länge  $n$ , welcher als unabhängiger Teil eines Datensatzes eines Regressionsmodells fungiert, gibt es einen Vektor  $\hat{\mathbf{Y}}$ , welcher die Vorhersagewerte zu jedem Datenpunkt  $i$  beinhaltet. Ein *Residualwert* ist  $\epsilon_i = y_i - \hat{y}_i$ . Der *Durchschnitt mean* der beobachteten Werte wird definiert als  $\bar{y} = \frac{1}{n} \sum y_i$ .

##### Definition IX.5: Bestimmtheitsmaß

Sei  $y_i$  beobachtet Werte  $i \in \{1, \dots, n\}$  der unabhängigen Variablen  $Y$  eines Regressionsmodells. Sei  $\hat{y}_i$  der geschätzte Werte für den gegebenen Datensatz  $i$ .

- Die SQE wird definiert durch

$$\text{SQE} := \sum_i \epsilon_i = \sum_i (y_i - \hat{y}_i)^2 \quad (2.1)$$

- Die SQT wird definiert durch

$$\text{SQT} := \sum_i (y_i - \bar{y})^2 \quad (2.2)$$

mit  $\bar{y} = \frac{1}{n} \sum y_i$ .

Für das *empirische Bestimmtheitsmaß  $R^2$*  gilt

$$R^2 := 1 - \frac{\text{SQE}}{\text{SQT}} \quad (2.3)$$

#### 2.1.2 Mittlere Quadratische Abweichung

Diese Bewertungsmetrik ist nicht mit der Standardabweichung zu wechseln. Diese ist ein Maß für die Streuung für eine Zufallsvariable, und wird auch *Mittlere Quadratische Abweichung*. Um die Verzerrung eines Punktschätzers kann mit Hilfe von MSE bestimmt werden.

- Root Mean Square Error (RMSE)

$$RMSE = \sqrt{MSE(\hat{\theta})} = \sqrt{E((\hat{\theta} - \theta)^2)}$$

- Mean Absolute Error

$$MAE = \frac{\sum |y_i - x_i|}{n}$$

## 2.2 Klassifikation Analyse

## 3 Klassen eines statistischen Prozesses

Das Merkmal (Variable) eines statistischen Prozesses lässt nach verschiedenen Kriterien klassifizieren.

### Anzahl der Ausprägungen

- Stetig
- Diskret
  - abzählbar
  - unendlich Ausprägungen

## Art der verwendeten Messskala/ Skalenniveau/ Messniveau

- Nominalskala (Kategorial)
- Ordinalskala (Kategorial)
- Metrische Skala/ Kardianlskala
  - Verhältnisskala
  - Intervallskala
  - Absolutskala

Variablen, die zur Gruppe das Skalenniveau Nominal und Ordinal aufwesen, können auch unter dem Begriff der **Kategorial Variable** gefasst werden. Das Skalenniveau gibt zum Ausdruck, welche Methoden/ Operationen auf eine bestimmte Variable durchgeführt werden kann. Der Begriff Skala gibt dabei zum Ausdruck, dass mit zunehmender Steigerung, mehr Operationen mit der Variable durchgeführt werden können. D. h., Variablen die in einem höheren Skalenniveau liegen, bieten eine größere Möglichkeit an Operationen.

4. OG: <b>Absolutskala</b>	Rangordnung, Abstände, natürlicher Nullpunkt und natürliche Einheit sind definiert	Anzahl an Fachsemestern, Verlegenheitsgesten
3. OG: <b>Verhältnisskala</b>	Rangordnung, Abstände und natürlicher Nullpunkt sind definiert	Gehalt, Reaktionsgeschwindigkeit, Gewicht
2. OG: <b>Intervallskala</b>	Rangordnung und Abstände sind definiert	Temperatur in Celsius, IQ
1. OG: <b>Ordinalskala</b>	Rangordnung ist definiert	Schulnoten, Bildungsstand
EG: <b>Nominalskala</b>	Klassifikationen, Kategorien	Geschlecht, Automarke, Stadt, Studiengang

Abbildung 250: Beispiele für Variablen unterschiedlicher Messskalen

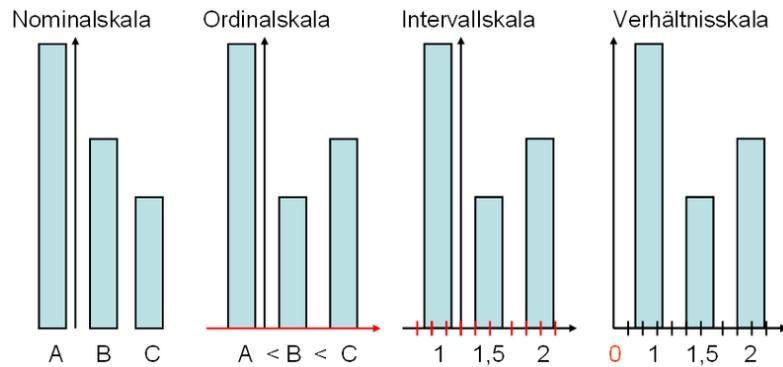


Abbildung 251: Anordnungsmöglichkeiten für Ausprägungen von Variablen verschiedener Messskalen

# **Teil X**

# **Artificial Intelligence**

## 4 Overview

Es kann erstmal eine Sammlung aufgemacht werden.

Bei Künstliche Intelligence (K.I.)/ engl. Artificial Intelligence (A.I.) ist erstmal zu klären, was man darunter versteht und wie am diese Intelligence misst.

Das Gebiet von K.I. beschäftig sich mit,

- Robotics
- ML
-

# **Teil XI**

# **Stochastik**

# 1 Überblick

Das Gebiet der Stochastik beschreibt zwei Gebiete

- Wahrscheinlichkeitstheorie: Diese betrachtet die theoretische Modellierung von Zufallsereignissen.
- Statistik: Dieses Gebiet versucht anhand von realen Daten Parameter und Verteilungen theoretisch modellierte Zufallsereignisse zu schätzen.

Die beiden Gebiete sind diametral zu einander.

In der Wahrscheinlichkeitstheorie wird angenommen, dass alle Parameter eines Zufallsexperimentes vorhanden sind. Gefragt wird, welche Ausgänge mit welcher Wahrscheinlichkeit beobachtet werden können.

7 Eine Münze, die mit der Wahrscheinlichkeit  $p = 0,50$  Kopf zeigt, wird  $n = 100$  geworfen. Mit welcher Wahrscheinlichkeit wird  $k = 60$  mal Kopf getroffen?

In der Statistik wird angenommen, dass das Zufallsexperiment schon durchgeführt wurde, jedoch dies nicht vollkommen beschrieben ist, und sein Ausgang (am Beispiel  $k$ ) schon bekannt ist. Gefragt wird, nach den Parametern die dieses Zufallsexperiments beschreiben, um zukünftige Ausgänge mit ihren jeweiligen Wahrscheinlichkeiten zu beschreiben.

Eine Münze wurde  $n = 100$  geworfen und hat dabei  $k = 60$  Kopf angezeigt. Bestimmen (schätzen) die Wahrscheinlichkeit  $p$ , mit der die Münze bei einem Wurf Kopf zeigt.

Die Statistik versucht nicht Zufälligkeit aus der Welt zu eliminieren, sondern nur die Unsicherheit zu reduzieren, in welcher Welt man sich befindet.

(1.1)

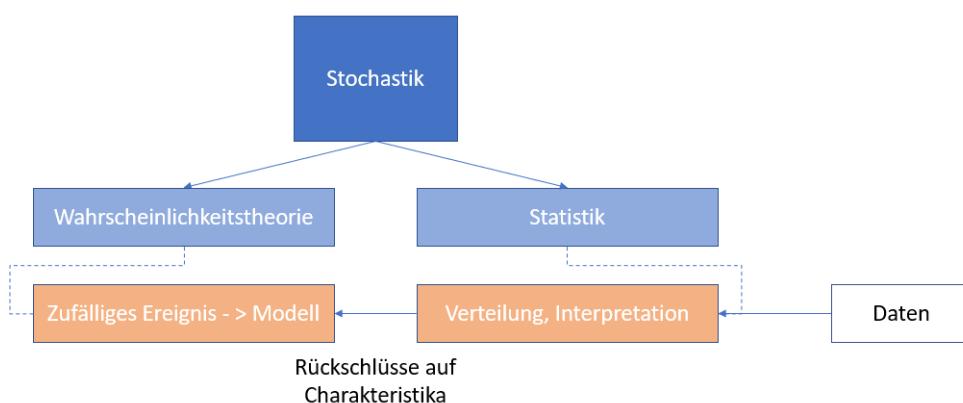


Abbildung 252: Feld der Stochastik

Um wichtige mathematische Objekte wieder aufzufrischen, wird im Folgenden kurz das Konzept einer Abbildung erklärt.

In der Mathematik versteht man unter einer Abbildung oder unter gewissen Bedingungen Funktion, eine Beziehung zwischen zwei Mengen, wobei Elementen der einen Menge (Definitionsmenge, *unabhängige Variablen*, *x-Werte*, *Funktionsargument*) Elementen der anderen Menge (Ergebnismenge, *abhängige Variable*, *y-Werte*, *Funktionswert*) zugeordnet werden. Man spricht von einer Funktion, wenn die Abbildung in einen reelle- oder komplexwertigen Körper abbildet. [Notation](#). [Abbildung o. D.](#)

**Definition XI.1: Abbildung**

Eine Abbildung ordnet jedem Element der Definitionsmenge  $D$  genau ein Element der Zielmenge  $Z$  zu:  
 $f : D \rightarrow Z, x \mapsto y := f(x)$

Die Begrifflichkeiten von Unabhängiger und Zielvariablen hat Ihren Ursprung in der Physik und Physik nahen Bereichen.

## 2 Wahrscheinlichkeitstheorie

### 2.1 Wahrscheinlichkeitsraum

Der Wahrscheinlichkeitsraum<sup>22</sup> ist das Kernstück der Wahrscheinlichkeitstheorie. Diese gibt die Werkzeuge, Zufallsexperimente aus der realen Welt zu modellieren.

Im Folgenden wird das Konzept von *Ergebnis* und *Ereignis* konzeptioniert. Mit der gewonnen Ergebnismenge wird das Wahrscheinlichkeit(s)(W.) Maß bestimmt.<sup>23</sup>

#### Definition XI.2: Wahrscheinlichkeitsraum

Sein  $\Omega$  eine nichtleere Menge, sei  $\mathcal{A}$  eine  $\sigma$ -Algebra auf  $\Omega$  und  $P$  das W-Maß auf  $\mathcal{A}$ , heißt das Tupel  $(\Omega, \mathcal{A}, P)$  **Wahrscheinlichkeitsraum**.

<sup>24</sup> Mit den drei Komponenten Das Konzept der Zufallsvariable (ZV) wird helfen, einen leichteren Umgang mit dem W-Raum zu erhalten. Arens, 2018, Henze, 2018

#### 2.1.1 Ergebnismenge

Die Menge der möglichen Ergebnisse eines stochastischen Vorgangs wird üblicherweise mit  $\Omega$  abgekürzt. Am Anfang einer stochastischen Modellierung ist diese der entscheidende Schritt, zu definieren, wie diese Menge aussieht.

#### Definition XI.3: Ergebnismenge/raum

Die Menge  $\Omega$  ist die *Ergebnismenge* oder der *Grundraum* eines Zufallsexperiments und enthält alle möglichen Ergebnisse  $\omega$ .

Dies Definition greift einen naiven Mengenbegriff auf, welcher nur erklärt, ob  $\omega \in \Omega$  oder  $\omega \notin \Omega$  ist. Diese Eigenschaft scheint in der Theorie trivial zu sein, in der Praxis ist es nicht immer klar, ob ein Ergebnis zu einer Grundgesamtheit oder nicht gehört. Ein Beispiel hierfür ist, das *Zufallsvorgang (Experiment): Schwere Unfälle in Berlin*. Ist das Zufallsexperiment ein *Münzwurf*, dann ist der Grundraum leichter zu bestimmen:

$$\begin{aligned}\Omega &= \{K, Z\} \text{ oder} \\ &= \{Kopf, Zahl\}.\end{aligned}$$

In diesem Fall ist die Mächtigkeit gleich, die genaue Ausprägung von  $\omega$  kann jedoch für das gleichgemeinte Zufallsexperiment ungleich sein. Im Falle eines Kartenspiels ist gut zu zeigen, wie unterschiedlich die Grundgesamtheit sogar in ihrer Mächtigkeit ausfallen kann:

$$\begin{aligned}\Omega &= \{Rot, Schwarz\} \text{ oder} \\ &= \{Rot, Schwarz\} \text{ oder} \\ &= \{RotBild, RotZahl, SchwarzBild, SchwarzZahl\} \text{ oder} \\ &= \{R2, R3, \dots, SAss\} \text{ oder} \\ &\text{etc.}\end{aligned}$$

<sup>22</sup>engl. Probability

<sup>23</sup>Wenn im Folgenden W- als Prefix verwendet wird, ist der Begriff Wahrscheinlichkeit gemeint.

<sup>24</sup>Begrifflichkeit aus der Maß Theorie:

#### Definition XI.1: Maßraum

Das Triplet  $(\Omega, \mathcal{A}, \mathbb{P})$  heißt *Maßraum*,

- wenn  $\Omega$  eine beliebige, nicht leere Menge ist. Diese wird *Grundraum* genannt.
- $\mathcal{A}$  eine  $\sigma$ -Algebra über den Grundraum  $\Omega$  ist.
- $\mathbb{P}$  ein Maß, dass auf  $\mathcal{A}$  definiert ist.

Werden  $n$  hintereinander durchgeführte Einzelexperimente durchgeführt, können dies als  $n$ -Tupel eines Gesamtexperiments gesehen werden. Angenommen es liegen zwei Zufallsexperimente vor: 1. Experiment ist ein Münzwurf, das 2. Experiment ist ein Würfelwurf, welche hintereinander durchgeführt werden. Ein adäquater Grundraum für das Gesamtexperiment ist

$$\Omega_1 \times \Omega_2 = \{(a_1, a_2) : a_1 \in \{K, Z\}, a_2 \in \{1, 2, \dots, 6\}\}$$

### 2.1.2 Ereignismenge

Bei stochastischen Vorgängen interessiert oft nur, ob ein Ergebnis  $\omega$  zu einer *gewissen Menge von Ergebnissen* gehört. Es wird also eine Menge gesucht, welche alle Fragestellungen an das Zufallsexperiment beantworten kann. Jede Teilmenge von  $\Omega$  heißt jetzt ein **Ereignis**. Diese Teilmengen werden mit Großbuchstaben abgekürzt.

Am Beispiel es *Würfel* wird jedes Ergebnis als Ereignis dargestellt:  $\{1\}, \{2\}, \dots, \{6\}$ . Die Grundmenge  $\Omega$  ist ein **sicheres Ereignis**. Das Ereignis  $A$ , dass eine gerade Zahl fällt, ist  $\{2, 4, 6\}$ . Das keine Zahl getroffen wird, wird **Unmögliches Ereignis** und entspricht  $\emptyset$ .

Um hinreichend komplexe Modelle zu entwickeln, müssen Anforderungen aus der Mengenlehre erfüllt werden. In der Sprache der Mengenlehre, muss die Gesamtheit der Ereignisse eine  $\mathcal{A}$  bilden, sie XI.2 - Sigma-Algebra (S.: 244).

#### Definition XI.4: Ereignismenge

Sei  $\Omega$  eine nichtleere Ergebnismenge, dann heißt das Mengensystem  $\mathcal{A}$  von Teilmengen auf  $\Omega$  eine **Ereignismenge**, wenn  $\mathcal{A}$  eine  $\sigma$ -Algebra ist (siehe XI.2 - Sigma-Algebra (S.: 244)).

Ist die  $\Omega$  eine endliche oder abzählbar unendliche Menge, so die größte  $\sigma$ -Algebra die aufgespannte Potenzmenge. Die kleinste ist  $\{\emptyset, \Omega\}$ .

Beim *Wurf* einer Münze ist  $\Omega = \{K, Z\}$ . Die erzeugte von  $\Omega$  erzeugenden  $\sigma$ -Algebra die Potenzmenge  $\{\emptyset, \{K\}, \{Z\}, \{K, Z\}\}$ .

### 2.1.3 Wahrscheinlichkeitsmaß

Der Begriff der Wahrscheinlichkeit ist inhaltlich leer. Dieser beruht auf das axiomatische Fundamente von *Kolmogorov*. Dies lässt einen großen Spielraum für die Interpretationen und Berechnungen offen. Solang die Axiome eingehalten werden, kann eine Wahrscheinlichkeit als relative Häufigkeit, Grenzwert von Häufigkeiten als Expertenmeinung oder etc. verstanden werden.

<sup>25</sup> Rechtschreibung: Am Ende eines Ganzsatzes setzt man nach Abkürzungen nur einen Punkt. Ausnahmen sind Fragesätze oder Anforderungen. (§ 103 des amtlichen Regelwerks zur deutschen Rechtschreibung)

<sup>26</sup> Begrifflichkeit aus der Maß Theorie:

#### Definition XI.2: Sigma-Algebra

Sei  $\Omega$  eine nichtleere Obermenge und sei  $\mathcal{A}$  eine Menge von Teilmengen der Obermenge, dann heißt  $\mathcal{A}$  eine  $\sigma$ -Algebra, wenn gilt:

- $\mathcal{A}$  enthält die Grundmenge:  $\Omega \in \mathcal{A}$ .
- $\mathcal{A}$  ist stabil bezüglich der Komplimentbildung: Ist  $A \in \mathcal{A}$ , dann ist auch  $A^C \in \mathcal{A}$ . Dabei ist  $A^C$  das Komplement von  $A$  bezüglich  $\Omega$ .
- $\mathcal{A}$  ist stabil bezüglich abzählbarer Vereinigungen: Sind  $A_i \in \mathcal{A}, i \in \mathbb{N}$ , so ist auch  $\bigcup_{i=1}^{\infty} A_i \in \mathcal{A}$ .

### Definition XI.5: Drei Axiome von Kolmogorov/ Wahrscheinlichkeitsmaß

Sei  $\Omega$  eine Obermenge, sei  $\mathcal{A}$  eine  $\sigma$ -Algebra von  $\Omega$ , heißt die Abbildung  $P$  von  $\mathcal{A}$  nach  $\mathbb{R}$  **Wahrscheinlichkeitsverteilung** oder **Wahrscheinlichkeitsmaß** von  $\Omega$  (genauer: auf den Teilmengen von  $\Omega$ ), wenn für  $P$  gilt:

- Axiom Nichtnegativität und Normiertheit:  $\forall A \in \mathcal{A} : 0 \leq P(A) \leq 1$
- Axiom Normiertheit:  $P(\Omega) = 1$
- Axiom  $\sigma$ -Additivität: Für jede abzählbare Folge von disjunkten Mengen  $A_i \in \mathcal{A}$  gilt

$$P\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} P(A_i).$$

Der Wert von  $P(A)$  heißt Wahrscheinlichkeit.

<sup>27</sup> Das W-Maß kann somit als Abbildung in  $[0, 1]$  verstanden werden.

Für den Wurf eines Münze würde für  $\Omega = \{K, Z\}$  das W-Maß wie folgt aussehen:

$$P : \mathcal{A} \rightarrow [0, 1], A \mapsto P(A) = \begin{cases} P(\{K\}) & = 0,5 \\ P(\{Z\}) & = 0,5 \\ P(\{Z, K\}) & = 1 \\ P(\emptyset) & = 0 \end{cases}$$

Nicht immer wird eine Vorschrift für jede Menge  $A \in \mathcal{A}$  definiert oder auch der das Ergebnis richtig definiert.

- Sei  $\Omega = 1, 2, \dots, 6$  die Ergebnismenge für einen Würfelwurf. Es kann beobachtet werden, dass  $P(i) = \frac{1}{6} \forall i \in \Omega$  angegeben wird. Das W-Maß nimmt jedoch nur Mengen auf, weshalb es nicht wäre  $P(\{i\})$  zu notieren.
- Selbst wenn  $P(\{i\})$  notiert wird, wird durch die  $\sigma$ -Additivität  $P(\{2, 4, 6\}) = \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = \frac{3}{6}$  hergeleitet.
- Wenn bei wiederholten Experimenten *Totale Unabhängigkeit* vorliegt, kann bei einem dreimaligen Münzwurf  $P(\{KZK\}) = P(\{K\})P(\{Z\})P(\{K\})$  mit  $(\frac{1}{2})^3$ .

## 2.2 Zufallsvariable

Das Konzept der ZV dient des leichteren Umgangs mit dem W-Raum. Rechnungen und weitere aufgesetzten Konzepte werden nicht mehr in einem abstrakten Raum  $\Omega$  bestimmt, sondern im  $\mathbb{R}^1$  oder auch bei mehrdimensionale ZV der  $\mathbb{R}^k$  mit  $k \in \mathbb{N}$ .

### 2.2.1 Herleitung

Schlussendlich erzeugt die Zufallsvariable einen neuen W-Raum, welcher auf einem Erzeugerraum  $(\Omega, \mathcal{A}, P)$  definiert wird. Dabei werden unter gewissen Bedingungen Eigenschaften dieses Erzeugerraums auf die Zufallsvariable übertragen, so auch die Bildung des neuen W-Maß  $P_X$ .

<sup>27</sup> Alternative Beschreibung aus der Maß Theorie:

### Definition XI.3: Maß

Es sein  $\mathcal{A}$  eine  $\sigma$ -Algebra einer nichtleeren Grundmenge  $\Omega$ . Eine Abbildung  $f : \mathcal{A} \rightarrow [0, \infty]$  heißt *Maß* auf  $\mathcal{A}$ , wenn die beiden Bedingungen erfüllt sind:

- $f(\emptyset) = 0$
- $\sigma$ -Algebra Additivität.

## Bemerkung

- ZV werden mit Großbuchstaben aus dem hinteren Bereich des Alphabets notiert:  $X, Z, W$ , etc.
- Es gibt auch anderen Begrifflichkeiten für ZV. Vom Prinzip handelt sich bei einer ZV nicht um eine Variable sondern um eine Abbildung. Weil ZV jedoch im Verbindung mit dem W-Raum betrachtet und weiterverarbeitet werden, kommt diese Bezeichnung zustande. Realisationen einer ZV werden nie einzeln betrachtet, sie werden immer in der ein oder anderen Weise in Abhängigkeit ihres zugehörigen W-Maß betrachtet. Aus diesem Zusammenhang leitet sich auch der Begriff des Zufalls ab, für die Interpretation einer Realisierung kommt immer in Frage, wie wahrscheinlich das eintreten dieses Event ( $\mathcal{A}$ ) ist. Der Zufall entscheidet über die Verteilung und gibt so eine variable Größe wieder. Daher kommt auch der andere Name *Zufallsgröße* zustande. Betrachtet man höherdimensionale ZV spricht man von *Zufallsvektoren*.

**Definition** In der einfachsten Form ist eine ZV eine Abbildung, welche aus  $\Omega$  eine W-Räume  $c$  auf  $\mathbb{R}$  abbildet. Diese Abbildung muss nicht bijektiv oder injektiv sein. Somit könnte gelten

### Definition XI.6: Zufallsvariable

Sei  $(\Omega, \mathcal{A}, P)$  ein W-Raum. Die Abbildung

$$X : \Omega \rightarrow \mathbb{R}$$

heißt **Zufallsvariable**.

Allgemeiner Vermerk: Eine mathematische Definition kann auf verschiedenen Weise beschreiben werden. Welche Konzepte wann angebracht werden, kann aus den verschiedensten Gründen erfolgen. Schlussendlich besteht das Ziel, ein Problem lösen zu können. Bei dem Konzept der Zufallsvariable gilt das gleiche. Es können Element der Maßtheorie in die Definition gezogen werden. Ebenso können Bestandteile des W-Maß  $P_X$  vorgenommen werden, und an die Definition des Zufallsvariablen angefügt werden. Bsp.:

### Definition XI.4: Alternative Definition I der Zufallsvariable

Sei  $(\Omega, \mathcal{A}, P)$  ein W-Raum. Die Abbildung

$$X : \Omega \rightarrow \mathbb{R}$$

heißt **Zufallsvariable**, wenn bei der das vollständige Urbild  $X^{-1}(B)$  jeder Borel-Menge  $B \in \mathcal{B}$  ein Element von  $\mathcal{A}$  ist.

### Definition XI.5: Alternative Definition II der Zufallsvariable

Sei  $(\Omega, \mathcal{A}, P)$  ein W-Raum und  $(\Omega_2, \mathcal{A}_2)$  ein Messraum, so heißt die Abbildung

$$X : \Omega \rightarrow \Omega_2$$

**Zufallsvariable**. Ist  $\Omega_2 = \mathbb{R}^k$ , so wird  $X$  als  $k$ -dimensionale reelle Zufallsvariable und im Fall  $k = 1$  als reelle Zufallsvariable bezeichnet.

Die ZV ist eine Vorschrift, welche jedem Ergebnis  $\omega$  eines stochastischen Vorgang eine Wert  $X(\omega)$  zu weißt. Dies Werte werden als *Realisationen* von  $X$  bezeichnet. Ebenso können sie als Messung einer stochastischen Experiment interpretiert werden.

**Borel Sigma Algebra** Um das W-Maß  $P_X$  zu definieren, wird die  $\sigma$ -Algebra auf  $\mathbb{R}$  benötigt. Für abzählbare Mengen  $\Omega$  war die größte  $\sigma$ -Algebra die Potenzmenge  $\mathcal{P}(\Omega)$ . Für  $\mathbb{R}$  ist es die Borel- $\sigma$ -Algebra.<sup>28</sup>

<sup>28</sup>Referenz: Wikipedia/MessbareFunktion

### Definition XI.7: Borelsche Sigma-Algebra

Sei  $\Omega$  eine topologischer Raum und  $\mathcal{O}$  das System der offenen Teilmengen von  $\Omega$ . Dann heißt

$$\otimes := \sigma(\mathcal{O}) \quad (2.1)$$

die *Borelsche  $\sigma$ -Algebra* über  $\Omega$ . Ihre Elemente  $B \in \mathcal{B}$  heißen *Borelsche-Mengen* oder *Borel-Mengen*. Im Fall  $\Omega = \mathbb{R}^k$  setzen wir  $\mathcal{B}^k := \mathbb{R}^{\parallel}$ , im Fall  $k = 1$  ist  $\mathcal{B} := \mathbb{R}^{\infty}$ .

#### Bemerkung

- Borel- $\sigma$ -Algebrawird über einen Erzeuger  $E$  definiert und wird nicht direkt von  $\Omega$ . Beispiel für  $\Omega = \mathbb{R}$  sind

$$E_1 = \{(-\infty, a] : a \in \mathbb{R}\}, E_2 = \{(-\infty, a) : a \in \mathbb{R}\}, E_3 = \{(a, b) \subset \mathbb{R} : a \leq b\}, \text{etc.}$$

- Die *Borel- $\sigma$ -Algebra* ist die kleinste  $\sigma$ -Algebra die alle Teilmengen von  $\Omega$  besitzt.
- Für den weiteren Verlauf wird nicht weiter darauf eingegangen, welcher Erzeuger die *Borel- $\sigma$ -Algebra* erzeugt.

**Verteilung einer Zufallsvariable** Es besteht folgendes Problem. Wie kann sicherstellt werden, dass die Eigenschaften der Wahrscheinlichkeitsverteilung des Erzeugerraums  $(\Omega, \mathcal{A}, P)$  erhalten bleiben oder sich in gewissem Maße übertragen. Hierfür wird für  $X$  die *Messbarkeitseigenschaft* verlangt.

Diese besagt im Kontext der Maß Theorie

### Definition XI.6: Messbarkeitseigenschaft

Eine Abbildung  $f$  heißt *messbar*, wenn das Urbild jeder Menge  $B \in \mathcal{B}$  unter  $f$  ein Element aus  $\mathcal{A}$  ist.

Formaler bedeutet diese

$$f^{-1}(B) \in \mathcal{A} \quad \forall B \in \mathcal{B} \quad (2.2)$$

#### Bemerkung Messbarkeitseigenschaft

- Der Ausdruck  $f^{-1}(B)$  bezeichnet dabei keine Abbildung, sondern die Urbildmenge.
- Alternative Schreibweisen sind  $\{\omega \in \Omega : f(\omega) = x\} \in \mathcal{A}$  für  $\forall x \in \mathbb{R}$ .
- Diese Anforderung stellt für die Definition oder Konzeption einer Zufallsvariable sicher, dass das Urbild eines Ereignisses in  $\mathcal{B}$  auch ein Ereignis in  $\mathcal{A}$  unter  $\Omega$  ist.

Hintergrund ist, diese Mengengebilde  $\{\omega \in \Omega : X(\omega) = x\} \in \mathcal{A}$  wird für das W-Maß  $P_X$  benötigt. Denn im Bezug auf die ZV  $X$  bleibt die Frage bestehen, wie kann die Wahrscheinlichkeitsverteilung von  $\Omega$  über  $X$  auf  $P_X$  übertragen. Die Antwort: XI.6 - Messbarkeitseigenschaft (S.: 247). Damit diese Definition der *Verteilung von X* greift, muss zur Definition einer Zufallsvariable die Messbarkeitseigenschaft (messbar) ergänzt werden.

### Definition XI.8: Zufallsvariable und Messbarkeitseigenschaft

Sei  $(\Omega, \mathcal{A}, P)$  ein W-Raum und  $(\mathbb{R}, \mathcal{B})$  ein Messraum, so heißt die die  $\Omega - \mathbb{R}$ -messbare Abbildung

$$X : \Omega \rightarrow \mathbb{R}$$

**Zufallsvariable.**

### Definition XI.9: Verteilung einer Zufallsvariable/ Wahrscheinlichkeitsfunktion

Sei  $(\Omega, \mathcal{A}, P)$  und  $(\mathbb{R}, \mathcal{B})$  ein Messraum und  $X : \Omega \rightarrow \mathcal{B}$  eine Zufallsvariable, so heißt das Bildmaß

$$P_X(B) := P(X^{-1}(B))$$

Verteilung von  $X$  oder auch *Wahrscheinlichkeitsverteilung* im diskreten oder *Dichtefunktion* im stetigen Fall.

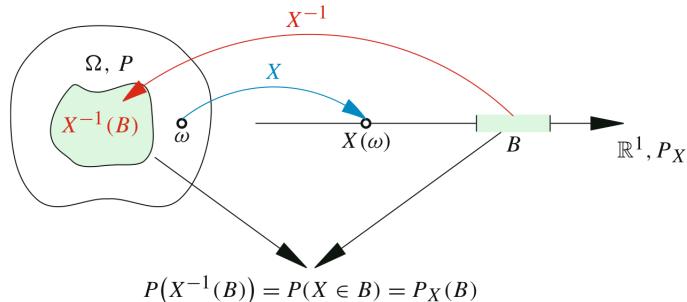


Abbildung 253: Die Zufallsvariable  $X$  bildet  $\Omega$  in  $\mathbb{R}$  ab und überträgt die Wahrscheinlichkeiten der Ereignisse

- Wegen der *Messbarkeitseigenschaft* ist das Bildmaß  $P_X$  auf  $\Omega$  gilt  $P_X(\{\Omega\}) = P(\Omega) = 1$
- Eigenschaften der Verteilung  $P_X$  werden als Eigenschaften von  $X$  deklariert. Z. B.: Wenn  $P_X$  diskret ist, ist auch  $X$  diskret. Wenn  $P_X = \mathcal{N}(\mu, \sigma^2)$ , so wird

$$X \sim \mathcal{N}(\mu, \sigma^2)$$

notiert. Das Symbol  $\sim$  wird nicht einheitlich ausgedrückt. Für den weiterem Umgang, wird hier angenommen, dass es für *verteilt nach* steht.

- Andere Schreibweisen für das W-Maß  $P_X$  sind

$$\begin{aligned} P(X \in B) &:= P_X(B) := P(\{\omega \in \Omega : X(\omega) \in B\}), \\ P(X = x) &:= P_X(x) := P(\{\omega \in \Omega : X(\omega) = x\}), \\ P(X \leq x) &:= P_X((-\infty, x]) := P(\{\omega \in \Omega : X(\omega) \leq x\}), \text{ etc.} \end{aligned}$$

29

Auf der Verteilung von  $X$  aufbauend wird die *Verteilungsfunktion von  $X$*  definiert.

### Definition XI.10: Verteilungsfunktion von $X$

Ist  $X$  eine Zufallsvariable auf einen WS-Raum  $(\Omega, \mathcal{F}, \mathbb{P})$ , so heißt die durch

$$F_X(x) := \mathbb{P}(X \leq x), x \in \mathbb{R}$$

definierte Abbildung  $F_X : \mathbb{R} \rightarrow [0, 1]$  die *Verteilungsfunktion von  $X$* . Diese besitzt die Eigenschaft *monoton wachsend, rechtsseitig stetig* und  $F_X$  kommt von 0 und geht nach 1.

<sup>29</sup> Kommt die Frage auf, ist der Ausdruck  $P(X \leq x)$  überhaupt möglich, wenn nicht sichergestellt werden kann, ob die Zufallsvariable mindestens eine ordinale Skalierung besitzt? Die Antwort: Ja. Das W-Maß gibt nur wieder, wie die Verteilung von  $X$  aussieht und die Realisationen in  $\mathbb{R}$  zu verwerten sind. Dabei gibt es eine gegebene Ordnung in  $\mathbb{R}$ . Ob diese Sinn ergibt, hängt von der Skala ab. Dies ist Thema im Kapitel *Descriptive Statistik*.

## 2.2.2 Eigenschaften und Besonderheiten

**Diskret ZV** Handelt es sich bei  $X$  um eine stetige oder diskrete ZV ist damit die *Wahrscheinlichkeitsfunktion* oder *Dichtefunktion* gemeint. Dies Eigenschaft wird sprachlich auf die Abbildung  $X$  gemünzt.

### Definition XI.11: Diskrete Wahrscheinlichkeitsverteilung einer diskreten zufälligen Variablen

Eine *diskrete* Zufallsvariable  $X$  besitzt endlich oder abzählbar unendlich viele Realisationen  $x_i$ , die mit Wahrscheinlichkeiten

$$p_i = P(X = x_i) > 0$$

angenommen werden. Für  $x_i$  gilt

$$\sum_{i=1}^{\infty} P(X = x_i) = 1.$$

Die Angaben aller  $p_i$  heißen *Wahrscheinlichkeitsverteilung von  $X$* .

**Verteilung im Erzeugerraum bekannt** Im Fall, dass  $\Omega$  sich nicht bestimmen lässt oder  $P$  sich nicht abbilden lässt, ist das Wissen gut, dass für den W-Raum von  $X$  es ausreicht, wenn die  $F_X$  bekannt ist. Dies erlaubt, dass direkt im aufgespannten Raum zu bleiben.

**Kanonische Konstruktion** Lässt sich  $\Omega$  nicht definieren oder ist zu kompliziert darzustellen. Kann für  $\Omega$  gleich mit  $\mathbb{R}$  gesetzt werden. Für  $\mathcal{A}$  wird  $\mathcal{B}$  gesetzt. Für die Abbildung von  $\Omega$  nach  $\mathbb{R}$  wird die Identität gewählt.

Die Art des Aufbaus des neuen W-Raums nennt sich **Kanonische Konstruktion** bei  $X(x) := x$  ist. Dies ist nicht untypisch in der Statistik, dass der W-Raum  $(\Omega, \mathcal{A}, P)$  in den Hintergrund gerät.

**Verteilungen** Verteilungen wie Bernoulli, Poisson, Normalverteilt, etc. bestimmen sich aus den gegebenen Parametern des zu Grunde liegenden Sachverhalt der W-Raums.

**Nicht messbare Abbildung** Sei  $X = Y = \{1, 2\}$  und  $\mathcal{A} = \{\emptyset, X\}$  (Kleinste  $\sigma$ -Algebra) und  $\mathcal{B} = \mathcal{P}(X)$ .

Die Frage: Ist  $\mathcal{A} - \mathcal{B} - f$  messbar, wenn  $f(1) := 2$  und  $f(2) := 1$  definiert ist?

Antwort: Nein, die Abbildung ist nicht messbar. Für  $f^{-1}(\{2\})$  müsste das Ereignis  $\{1\}$  in  $\mathcal{A}$  liegen.<sup>30</sup>

Hinweis: Für *messbare* ZV wird im diskreten Fall die Potenzmenge für  $\Omega$  herangezogen.

## 2.2.3 Beispiele

**Bsp. 1** Der Erzeugerraum wird für den stochastischen zweifachen Würfelwurf

$$\Omega = \{1, 2, 3, 4, 5, 6\}^2 = \{\omega = (\omega_i)_{i \in \{1,2\}} : \omega_i \in \{1, 2, 3, 4, 5, 6\}\}$$

Ein Ereignis  $A = \{2, 4, 6\}$  könnte die Menge der geraden Zahlen sein. Bei der Zufallsvariable  $X$  interessiert *Die Summe der Augenzahlen*. Die Abbildungsvorschrift sieht dabei wie folgt aus

$$X = \begin{cases} 1 & , \text{ wenn } (1, 1) \\ 2 & , \text{ wenn } (1, 2) \\ \vdots & \vdots \\ 36 & \text{ wenn } (6, 6) \end{cases}$$

<sup>30</sup>Referenz: Matheraum/nicht messbare Funktion

Das W-Maß  $P$  auf  $\Omega$  definiert sich gleichverteilt:  $P(\{\omega\}) = \frac{1}{36}$ . Für  $P_X$  leitet sich aus der XI.6 - Messbarkeitseigenschaft (S.: 247) ab, die Kombinationen von  $\omega$  welche die Summenzahl  $x_i$  ergeben, werden aufaddiert. Als Beispiel:

$$\begin{aligned} X &= 4 - \text{Summe 4 der Augenzahlen zweier Würfel} \\ \rightarrow P_X(\{X = 4\}) &= P(\{(1, 3), (2, 2), (3, 1)\}) \\ &= \frac{1}{36} + \frac{1}{36} + \frac{1}{36} + \frac{1}{36} \\ &= \frac{4}{36} \end{aligned}$$

**Bsp. 2** Gegeben sei eine Population von Menschen mit  $\Omega = \{\text{Silvia, Manfred, ...}\}$  von diesen können Informationen (Plural) gezogen werden.

- Körpergewicht  $\mathbb{R}_>$ . Diese genau Wahrscheinlichkeitsverteilung  $P_X$  ergibt sich aus den konkreten Urbildern zu jeden Wert von  $x$ .
- Die Information Parteizugehörigkeit kann ebenfalls als Zufallsvariable abgebildet werden. Die Parteien werden einer belieben Zahl zugeordnet. Je nach dem wie in der Population die Personen den Parteien zugeordnet sind, bestimmen die Urbilder über der Verteilung der ZV.

### 2.3 Momente einer Zufallsvariable

Momente sind Kenngrößen von ZV. Die zwei wichtigsten sind der Erwartungswert und die Varianz. Im Allgemeinen bestimmen die beiden Momente eine Verteilung einer ZV nicht eindeutig. Es gibt jedoch eine Menge von Verteilungen welche sich eindeutig durch ihre Momente bestimmen lassen.

**Erwartungswert** Der Erwartungswert nimmt somit die zwei Komponenten, Ergebnisraum und W-Maß eines W-Raums und bildet eine eigenen Größe.

**Definition XI.12: Erwartungswert (stetig)**

Ist  $X : \Omega \rightarrow \mathbb{R}$  eine quasiintegrierbare reelle Zufallsvariable, so heißt

$$E[X] := \int X dP = \int x dP_X(x)$$

der Erwartungswert von  $X$ .

Ist  $X$  diskret, so definiert sich der Erwartungswert wie folgt:

**Definition XI.13: Erwartungswert (Diskret)**

Für eine reelle Zufallsvariable  $X$  auf einer endlichen W-Raum  $(\Omega, \mathcal{A}, P)$  heißt

$$E[X] := E[X]_P := \sum_{\omega \in \Omega} X(\omega) P_X(\{\omega\})$$

der Erwartungswert von  $X$ .

Das  $P_X(\{\omega\})$  angegeben wurde, bezieht sich darauf, dass zu jedem  $\omega$  es unterschiedliche oder auch gleiche  $x$  abgebildet werden können.

**Varianz** Diese gibt die Streuung der Realisationen um den Erwartungswert wieder.

#### Definition XI.14: Varianz (diskret)

Für eine reelle Zufallsvariable  $X$  auf einer endlichen W-Raum  $(\Omega, \mathcal{A}, P)$  heißt

$$Var [X] := E [X - E[X]]^2$$

die Varianz von  $X$ .

## 3 Inferenz Statistik

### 3.1 Statisches Model

In der W-Theorie ist das Kernstück der W-Raum. In der Inferenz Statistik befasst man sich mit den Ungenauigkeiten der Welt, dafür wird der W-Raum zum Statistischen Model (SM) angepasst. Ebenso werden neue Begrifflichkeiten aus Aufgebauten Konzepte benötigt.

**Begriffliche Einführung** Das SM vereinfacht oder passt das vorherige Verständnis eines W-Raum an. Für die Modellierung eines W-Raum wird eine konkrete W-Verteilung/ W-Maß festgelegt. Dies wird im SM durch eine Tupel von W-Maße ersetzt. Dieses Tupel ist

$$(P_\vartheta^X)_{\vartheta \in \Theta} \text{ -- die parametrische Familie möglicher Verteilungen von } X.$$

Es wird angenommen, dass die *wahre* Verteilung zu diesem Tupel gehört. Im Allgemeinen wird von einem *Parameter* gesprochen. Dieser kann auch mehrdimensional sein, wie auch bei der Normalverteilung mit  $(\mu, \sigma^2)$ . Ebenso wird im Allgemeinen von einer ZV gesprochen, wenn auch hier eine mehrdimensional ZV vorliegt.

#### Definition XI.7: Parameterraum

Die Menge aller möglichen Werte des Parameters  $\vartheta$  wird *Parameterraum*  $\Theta$  genannt.

In der Regel ist der Parameterraum endlich und ein Vektor  $\Theta = (\Theta_1, \dots, \Theta_n)$  mit den Parametern  $\Theta_1, \dots, \Theta_n$  des Parameterraums  $\mathbb{R}^n$ . In der Stichprobe liegt ein unbekannter aber fester Parameter  $\vartheta \in \Theta$  zugrunde. Dieser Parameter wird aus der Stichprobe geschätzt. Sei  $X \sim \mathcal{N}(\mu, \sigma^2)$  wird aus der Stichprobe zu  $X \Theta = (\mu, \sigma^2) \in \Theta \subset \mathbb{R} \times \mathbb{R}^+$  geschätzt.

Im diesem anfänglichen Verständnis, dass die Verteilung und Verteilungsfunktion von  $X$  auf eine Familie von Verteilungen begründet ist, ergibt sich auch die neue Begrifflichkeit der *Grundgesamtheit*. Das Konzept der Zufallsvariable wir im Rahmen der Inferenzstatistik umschlossen.

#### Definition XI.8: Grundgesamtheit

Die *Grundgesamtheit* umfasst die Zufallsvariable  $X$  mit der Verteilungsfunktion  $F_\vartheta(x)$ , wobei  $\vartheta \in \Theta$  für die Parameter der Verteilungsfunktion steht.

In der Inferenzstatistik ist der Startpunkt die Realisation einer ZV. Um die Probleme von

- Abhängigkeiten,
- Inkonsistenz bei der Erhebung der "Realisationen",
- etc.

wird das Konzept der *Stichproben* eingeführt.

Für die Realisationen einer ZV wird jetzt ein Begrifflichkeit eingeführt, welches beschreibt, was passiert, wenn mehrere Realisationen einer ZV vorliegen. Nun wird jedoch nicht davon gesprochen, dass alle Realisationen einer

**Herleitung der Definition** Damit die Formalitäten stimmen, ist wichtig zu erwähnen, der Erzeugerraum mit dem W-Maß auf  $(\Omega, \mathcal{A})$  ebenfalls eine Familien von W-Verteilungen unterliegt für die gilt

$$P_\vartheta \text{ für alle } \vartheta \in \Theta$$

In dieser Familie liegt die Verteilung wahre Verteilung auf  $\Omega$ . Davon abgeleitet, ist

$$P_\vartheta^X \text{ für alle } \vartheta \in \Theta$$

**Beispiel 0-1-wertige Zufallsvariable** Es sei  $n$  0-1-wertige Zufallsvariablen  $X_1, \dots, X_n$  auf  $(\Omega, \mathcal{A}, P_p)$  gegeben. Diese sind unter  $P_p$  stochastisch unabhängig. Wie oben hergeleitet, gilt für das W-Maß bezüglich des Messbarkeits-Eigenschaft, für die Zufallsvariablen

$$P_p(X_i = 1) = p, \quad \forall i = 1, \dots, n. \quad (3.1)$$

#### Definition XI.15: Statistisches Modell

(a) Sei  $(\mathbb{R}, \mathcal{B})$  ein Messraum und  $(P_\vartheta^X)_{\vartheta \in \Theta}$  eine Familien von W-Verteilungen auf diesen Messraum, dann nennt man das Triplet

$$(\mathbb{R}, \mathcal{B}, (P_\vartheta^X)_{\vartheta \in \Theta})$$

ein *parametrisches statistisches Modell*.

(b) Sei  $(\mathbb{R}, \mathcal{B})$  ein Messraum und  $\mathbb{P}$  eine Menge von W-Verteilungen auf diesen Messraum, dann nennt man das Triplet

$$(\mathbb{R}, \mathcal{B}, \mathbb{P})$$

ein *nicht-parametrisches statistisches Modell*.

In der Regel ist  $\vartheta \in \Theta \subset \mathbb{R}^k$  eine ein reeller Vektor.

Ein *nicht-parametrisches statistisches Modell* ist ein Triplet gleich eines *parametrisches statistisches Modell* mit der Ausnahme von Familie von W-Verteilungen. In diesem Fall besteht dieses auch aus einer Familie von W-Verteilungen, diese sind aber nicht von  $\vartheta \in \Theta$  abhängig, sondern von einer Menge  $\mathbb{Q}$  von W-Verteilungen auf  $(\mathbb{R}, \mathcal{A})$ :

$$\mathbb{Q} = \left\{ \bigotimes_{i=1}^n Q_i : Q_i \text{ ist eine W-Verteilung auf } (\mathbb{R}, \mathcal{A}) \right\}$$

31

**Weitere Beispiele für ein parametrisches statistisches Modell** Es ist zu vermerken, dass es einen darauf zu achten ist, ob bei der Schreibweise mehrere Zufallsvariablen zu einer zusammengefasst werden, dann wird die  $\otimes$ -Schreibweise verwendet. Im Allgemeinen Fall kann für die Zufallsvariable als *Ergebnisraum* das Symbole  $M$  verwendet werden.

**Normalverteilung** Gegeben  $X = (X_1, \dots, X_n)$  stochastisch unabhängige Zufallsvariablen, ein SM, wenn angenommen wird, dass  $X_i$  normalverteilt ist, sieht wie folgt aus:

$$\left( \mathbb{R}, \mathcal{A}B^n, \bigotimes_{i=1}^n \mathcal{N}(\mu, \sigma^2)_{\mu \in \mathbb{R}, \sigma^2 \in (0, \infty]} \right)$$

**Binomialverteilung** Sei  $M = \{0, 1, \dots, n\} \subset \mathbb{R}$ ,  $\mathcal{A} = \mathcal{P}(M)$  und  $P_p = Bi(n, p)$  mit  $p \in (0, 1)$ .

**Binomialverteilung** Sei  $M = \{0, 1\}^n \subset \mathbb{R}^n$ ,  $\mathcal{A} = \mathcal{P}(M)$  und  $P_p = \bigotimes_{i=1}^n Bi(1, p)$  mit  $p \in (0, 1)$ .

Wenn von  $X \sim \mathcal{N}(\mu, \sigma^2)$  mit  $(\mu, \sigma^2) \in \mathbb{R} \times \mathbb{R}^+$  gesprochen wird, dann handelt es sich um ein (*parametrisches*) *statistisches Modell*.

<sup>31</sup> Das Symbol  $\otimes$  steht für das *Produkt*- $\sigma$ -Algebra und kommt aus der Maßtheorie. Hier wird sie nicht weiter definiert. Für den weiteren Verlauf ist wichtig zu wissen, im mehrdimensionalen Fall einer Zufallsvariable wird diese Schreibweise verwendet. Liegt eine Dimension vor, kann die Familienschreibweise verwendet werden.

**Begrifflichkeiten des Stichprobenraum** Die neue Begrifflichkeit eines W-Raums, *statistisches Model* lässt jetzt die Anstrengung der Schließenden Statistik besser verstehen. Diese versucht Anhand von der Realisierungen  $x$  von  $X$  den Parameter  $\vartheta$  zu bestimmen. Die Problematik ist, dass

$$P_{\Theta}(X = x) > 0$$

für jedes  $\Theta \in \Theta$  ist. Dies bedeutet, jedes  $x$  könnte von  $X$  zu jedem  $\Theta$  erzeugt worden sein. Deshalb kann nur von *Schätzung* gesprochen werden und der Zusammenhang zwischen  $x$  und  $\Theta$  unterliegt einer strukturellen Unsicherheit.

Die Essenz ist daher, anhand einer bekannten Stichproben  $x$  einen unbekannten Parameter  $\Theta$  zu schätzen.  
Dafür konstruiert man einen *Schätzer*

Unsicher was davor steht

**Stichprobenfunktion**

**Schätfunktion, Schätzer**

**Stichprobenraum**

### 3.2 Schätzen

### 3.3 Testen

## 4 Modellierung statistischer Zusammenhänge

Unter der Modellierung Funktionszusammenhänge - impliziert, expliziert, deterministisch und stochastisch Korrelation - Lediglich die Richtung des Zusammenhang klären. bvg Regression - Abhängigkeit von deterministischen Variablen auf eine zufällige Variable Um was geht es, es geht darum, wie die Themenblöcke, Schätzen, Testen und Regression abgebildet werden kann.

## 5 Regression

### 5.0.1 Beziehungen

Es gibt mehrere Möglichkeiten Beziehungen mathematisch darzustellen.

**kausale Inferenz** Um Kausalität zwischen  $X$  und  $Y$  darzustellen, muss die Ursache zeitlich vor der Wirkung sein. Zweitens muss die Wirkung und Ursache mit einander zusammenhängen. Drittens müssen weitere plausible Erklärungen ausgeschlossen werden.

### 5.1 Deskriptive Statistik

In der Deskriptiven Statistik wird der Gesichtspunkt geändert. Die Realisationen der Zufallsvariablen liegen vor. Es wird jetzt kein Rückschluss auf die wahren W-Raum gezogen, sondern eher eine Bewertung und Darstellung der Daten vollzogen.

Achtung: Die Begrifflichkeiten spiegeln sich und es gibt viele Gleichheiten. Es handelt sich aber hier nicht um das Feld der Induktiven Statistik.

**Merkmal** Anders als im vorherigen Kapitel ist ein Merkmal zwar das Kernelement der Descriptiven Statistik, jedoch nicht so strikt definiert wie die Zufallsvariable. Der Begriff der ZV wird jedoch oft als Synonym verwendet.

#### Definition XI.16: Merkmal

Ein **Merkmal** ist eine Abbildung von  $\Omega$  in eine Menge  $M$ :

$$X : \Omega \rightarrow M$$

Wenn die Menge  $M$  ohne weitere Struktur ist, so ist  $X$  ein **nominales Merkmal**. Ist  $X(\Omega)$  eine geordnete Menge, so ist  $X$  ein **ordinales Merkmal**. Ist  $X(\Omega) \subset \mathbb{R}$  und übernimmt die  $X(\Omega)$  die euklidische Metrik, so ist  $X$  ein **quantitatives** oder **kardinale Merkmal**.

Die Skalierung von Merkmalen erlaubt Vergleichsmöglichkeiten:

**Nominal**  $X(\omega_1) = X(\omega_2)$  oder  $X(\omega_1) \neq X(\omega_2)$

**Ordinal**  $X(\omega_1) \preceq X(\omega_2)$  oder  $X(\omega_1) \simeq X(\omega_2)$

**Kardinal**  $X(\omega_1) - X(\omega_2)$

Wie schon erwähnt, gibt es ähnliche Konzept, welche sich schwach zuordnen.

Merkmal $X$	Zufallsvariable $X$
$\Omega$ ist die Grundgesamtheit	$\Omega$ ist die Ergebnismenge und gehört zu einem W-Raum
$M$ ist der Merkmalsraum	$\mathbb{R}$ ist Bildraum
$X(\omega)$ ist die Ausprägung	$X(\omega)$ ist die Realisation
$X$ besitzt eine Häufigkeitsverteilung	$X$ besitzt eine W- Maß (W-verteilung)
$X$ besitzt eine empirische Verteilungsfunktion	$X$ besitzt eine theoretische Verteilungsfunktion

## **Teil XII**

# **Neues Familienmitglied-Zeitstrahl**

Das Kapitel beschäftigt sich mit dem Thema der Integration eines neuen Familienmitglied in eine bestehende zweier-Familien Struktur. Dabei werden die einzelnen Komponenten unter der Überschrift Neues-Familienmitglied Zeitstrahl (Neues-Familienmitglied Zeitstrahl) zusammengefasst.

Im folgenden werden drei Schwerpunkte behandelt:

- Physische Entwicklungslimitation (PEL),
- Dynamische Beziehungsstrafe (DBS) und
- Zeitlose Komponente (ZK).

Im ersten Abschnitt, PEL werden die mentalen und physischen Besonderheiten (mpB)<sup>32</sup> von Menschen in der Entwicklung eines Neugeborenen bis hin zur einem erwachsenen Menschen<sup>33</sup> aufgedeckt. Es wird dabei nicht auf Vollständigkeit abgezielt, sondern das Ziel verfolgt, gewonnene Informationen über diese Entwicklung unter der Zielseitung des NFMZ integrieren zu können.

Der Abschnitt DBS beschäftigt sich mit der Fragestellung, wie die PEL sich auf die Beziehungen zwischen den bestehenden FM und dem Neuen-Familienmitglied (NFM) auswirken. Der Schwerpunkt liegt darauf, Beziehungslevels von den gruppierten PELs abzuleiten. Diese Levels werden genutzt, um Meilensteine für die FM zu legen, um Verhaltensänderungen zu bewirken oder kommende Verhaltenanpassungen zu indizieren.

Im Abschnitt ZK werden Prinzipien, Wertvorstellungen und Heuristiken festgehalten, welche keine bestimmte zeitliche Komponenten beinhalten oder besondere Herleitung besitzen. Ein intrinsische Struktur zwischen diesen den einzelnen Komponenten wird nicht angestrebt, selbst wenn sie zu erkennen ist.

## 1 Physische Entwicklungslimitationen

Jegliche mpB sind einem zeitlichen Verlauf angeordnet. Jegliche Anhäufungen von mpB in einem Zeitintervall weisen dabei auf keine Besonderheit in diesem Lebensabschnitt hin.<sup>34</sup>

Für die weiteren Abschnitte werden Subbezeichnungen für verschiedene Zeiträume des *NFM* nach der Kassenärztlichen Bundesvereinigung herangeführt.

**Neugeborenes** [0, 28Lebenstag]

**Saugling** [0, 365Lebenstag]

**Kleinkind** [1Lebensjahr, 3Lebensjahr]

**Kind** [4Lebensjahr, 12Lebensjahr]

**Jugentlicher** [13Lebensjahr, 17Lebensjahr]

**Erwachsener** [18Lebensjahr, . . .)<sup>35</sup>

- 0 Jahre - Geburt
- Erster Atemzug: Zur Geburt wird durch einen Adrenalienschub die ungenutzten Lungen zur Atmung animiert.
  - Muskeln Kontraktion: Muskeln müssen erste jetzt komplett die Funktion übernehmen.
  - Herz: Dies hat zwei Löcher. Diese schließen sich in den ersten Tagen. Grund dafür ist, dass die Lungen jetzt mit Blut versorgt werden müssen, welches sie vorher nicht mussten.
  - Erster Stuhlgang: Dieser besteht aus einer grünen Flüssigkeit, welcher sich aus der Verdauungsflüssigkeit ohne jegliche Nahrung besteht.

<sup>32</sup>Beide Eigenschaften werden ebenfalls auf physische Merkmalsausprägen zurückgeführt. In diesem Kontext werden mentale und physische Besonderheiten als Emergenz einer bio-chemischen Merkmalsausprägung bezeichnet.

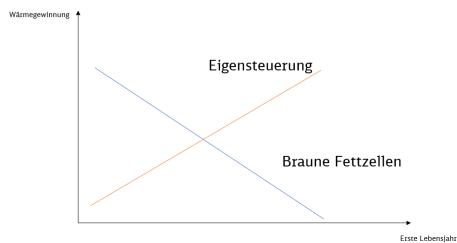
<sup>33</sup>Der relevante Zeitraum für erwachsene Menschen beträgt hier 18. bis 20. Jahren

<sup>34</sup>Der NFMZ ist ein Werkzeug, welches hilft Informationen über die Entwicklung von *NFM* in die aufgezeigte Struktur zu bringen. Quellen welche herangezogen werden, um Inhalte zu liefern, sind zufällig, daher überträgt sich diese Zufälligkeit auf auf die extrahierten und hier eingebrochenen PEL.

<sup>35</sup>In der ersten Bezeichnung

## 0 Jahre - Erste Wochen

- Temperatur
  - Körper des *NFM* muss sich an eine neue Außentemperatur anpassen. Es kommt zu einem Temperaturabsturz von  $38^{\circ}\text{C}$  zu durchschnittlich  $18^{\circ}\text{C}$ .
  - Der Hypothalamus ist bei neu geborenen nicht vollständig entwickelt, eine eigene Temperaturregulierung ist noch nicht möglich.
  - Zusätzlich besitzen *NFM* eine große Körperoberfläche im Verhältnis zum Körpervolumen. Dies verursachte einen hohen Temperaturverlust.
  - Gewinnung von Wärme aus Muskelkontraktion ist im ersten Jahr nicht möglich.
  - Die Wärmentwicklung wird durch Braunefettzellen gesteuert. Diese Wärmequelle nimmt im ersten Lebensjahr ab, bis der Körper eigenständig die Wärmeregulierung übernehmen kann.



36

### • Immunsystem

- Das Immunsystem wird bis das Stillen abgeschlossen ist, von der Muttermilch unterstützt. Antikörper werden über die Milch übertragen.
- Gleichzeitig trägt der enge Bund dazu bei, dass die Mutter den gleichen Erregern ausgesetzt ist.

### • Plötzlicher Kindstod

- Genaue physische Ursachen sind für dieses Phänomen nicht ableitbar.
- Empfehlungen wie: *Säuglinge* im ersten Lebensjahr sollen immer auf dem Rücken einschlafen, und es soll kontrolliert werden, dass dies auch durch die Nacht hin passiert. Es ist dabei möglich, dass auf dem Bauch zuerst eingeschlafen wird und später der Säugling umgedreht wird.
- Tagsüber ist für die Ausbildung der Rückenmuskulatur entscheidend, dass der Säugling auf dem Bauch liegt.

## 0 Jahre - 6 Wochen

- Iterationen mit unvertrauten Umgebungen sind eine hohe Lernherausforderung. Beispiel: Ein Besuch im Supermarkt zählt zu solchen Lerneindrücken.
- Die Hörqualität wird mit weiterem Alter nur noch abnehmen, und ist somit nie wieder so gut, wie als Neugeborenes.
- Retina und die Muskulatur zum Einstellen der Linse ist noch nicht entwickelt und trainiert. Diese führt dazu, dass schwarz-weiß und unscharf gesehen wird.
- Nach ca. 2 Monaten können Farben auseinander gehalten werden.
- Nach ca. 4 Monaten können Gesichter auseinander gehalten werden.
- In den ersten drei Monaten herrscht eine Wachstumsrate von 25 % je Monat. Ab 6 Monaten kann sich ein Säugling meist drehen.
- Ab 6 Monaten kann sich ein Säugling meist drehen. Dies erhöht das Risiko für einen plötzlichen Kindstod.

## 0 Jahre - 8 Monate

- Alle Sinne sind vollständig ausgebildet.

<sup>36</sup> Empfehlung: Generell fühlen sich Säuglinge gut angezogen bei einer Raumtemperatur von etwa 24 bis 25 Grad Celsius am wohlsten. Herrschen aber in den Räumen der Wohnung „normale“ Temperaturen, das heißt zwischen 20 und 23 Grad Celsius im Wohnzimmer und 17 bis 20 Grad im Schlafzimmer, sind ein Body oder ein Shirt aus Baumwolle genau das Richtige zum Unterziehen für das Baby. Darüber eignet sich dann ein recht enganliegender leichter Pullover, beziehungsweise ein nicht allzu dickes Sweatshirt optimal.

- Mit dem Tastsinn werden weitere Lernprozesse angestoßen.
- Weil die Dichte im Mund an Sinneszellen sehr hoch ist, wird dieser genutzt, um eine höhere Lernqualität zu erhalten. Der Mund besitzt spezielle Enzyme, welche Bakterien angreifen.
- Mit dem 9 Lebensmonat startet die Aushärtung der Kopfform und Knochenbildung. Dies führt dazu, dass Asymmetrien sich ab diesen Zeitpunkt verfestigen. Bis zum Ende des ersten Jahres ist dieser Prozess abgeschlossen.

1 Jahr	<ul style="list-style-type: none"> <li>• Durch die Möglichkeit des Krabbelns startet die Entwicklung von einem Baby zu einem Kleinkind.</li> <li>• Fähigkeit des Sprechens wird entwickelt</li> </ul>
11 Jahren	<ul style="list-style-type: none"> <li>• Hormonbildung startet.</li> <li>• Die sexuelle Entwicklung startet.</li> </ul>

37

## 2 Dynamische Beziehungsstrukturen

•

## 3 Zeitlose Komponenten

- "Mit den Geschenken der FM schafft *NFM* seinen eigenen Weg. Diese Geschenken können auch negativer Natur sein, da der gewünschte Effekt nicht eintritt. *With all that is given to you, you make it your own (Marvel, Shing Chan)*
- Umgang mit Geld: Unklar, wie dies genau passen soll.
- Ein Ziel ist: Die Strategien, die von FM selber erlernt wurden, dem *NFM* zu vermitteln. Diese können vom *NFM* angenommen, modifiziert oder auch abgelehnt zu werden.
- Ausbildung: Verweis: Conan & Harvard. Zwecke der Ausbildung. Es gibt Phasen der Ausbildung, in welcher das *NFM* experimentieren soll, um zu lernen, worin es nicht gut ist und was ein wirklich reizt.

## 4 Mathematik in der Welt der Töne

### 4.1 Einführung

- Fundierung der Tonsysteme als Gefüge von Tonbeziehungen
- Kepler: Harmonische Proportionen konsonanter Intervalle aus der Konstruierbarkeit regelmäßiger Viel-ecke
- Euler: Ästhetische Phänomene der Musik: Intervalle, Akkorde, Rythmen und Formproportionen
- Ab dem 19 Jh. kam die Erkenntnis, dass Musik mit Zahlen beschrieben, aber nicht hergeleitet werden kann. Heute Musiktheorie versucht geordnete Strukturen zu beschreiben
- Ab dem 20 Jh. wurde versucht eine mathematische Musiktheorie zu entwickeln.

---

<sup>37</sup>Quelle 1: [Link](#), Quelle 2: [Living Body Documentation](#), Quelle 3:

## **Teil XIII**

## **Anhang**

## Anhang A

# Abkürzungsverzeichnis

### **Symbole**

- NFM* Neues-Familienmitglied. 1, 257–259  
*R<sup>2</sup>* Empirisches Bestimmtheitsmaß. 1, 221  
**.pbix** Power BI Desktop File. 1  
**.qvd** QlikView Data. 1, 156  
**.qvf** Qlik Sense Desktop App File. 1, 161

### **A**

- ABB** Arbeitsbereicht. 1  
**AD** Active Directory. 1, 47  
**AGen2** Azure Data Lake Storage Gen2. 1, 52  
**ALM** Application Lifecycle Management. 1, 76, 77, 83  
**API** Application programming interface. 1, 76, 83  
**AWS** Amazon Web Service. 1  
**AZM** Arbeit Zeitmanagement. 1

### **B**

- Bahn Enterprise Architecture Management** Bahn Enterprise Architecture Management. 1  
**BB** Beförderungsbedingungen. 1  
**BCM** Bahn Content Management. 1  
**BEG** Bayrische Eisenbahngesellschaft. 1  
**BKU** Bürokommunikation Unternehmensweit. 1

### **C**

- CD** Continous Development. 1, 76  
**CDS** Common Data Service/ Dataverse. 1, 69, 70, 72  
**CI** Continous Integration. 1, 76  
**cmd** command-line interpreter. 1, 142  
**CNN** Convolutional Neural Networks. 1  
**CPP** C++. 1, 210, 267  
**csv** comma-separated values. 1, 21, 182

### **D**

- DAX** Data Analysis Expressions. 1, 21, 34, 37, 40, 55, 56  
**DB** Deutsche Bahn AG Konzern. 1, 73  
**DB MC** DB Modular Cloud. 1  
**dbo** DataBase Owner. 1  
**DBS** DB Sicherheit. 1  
**DBS** Dynamische Beziehungsstufe. 1, 257

**DBSE** Externe zivile Prüfer der DB Sicherheit. 1

**DFO** DAX-Filter-Option. 1, 55, 56, 61

**DLP** Data Loss Prevention Policies. 1, 69

**DNN** Deep Neural Network. 1

## E

**EBL** Eisenbahnbetriebsleiter. 1

**EC2** Amazon Elastic Computing Cloud. 1, *Glossar:* Amazon Elastic Computing Cloud

**EDA** Exploratory Data Analysis. 1, 194

**EfA** Endgeräte für Alle. 1

**ETL** Extract, Transform, Load. 1, 21, 52, 182

**EuLe** Erlös- und Leistungsdatenbank. 1

## F

**FGMT** Folgegeneration MT. 1, 263

**FIS-DV** Fahrgastinformationssystem Datenversorgung. 1

**FM** Familienmitglieder. 1, 257, 259

**FP-Nr** Fahrausweisprüfer Nummer. 1

**FV** Fernverkehr. 1

## G

**GML** General Linear Model. 1, 229–231, 233

## H

**HTML** Hypertext Markup Language. 1

**HTTP** Hypertext Transfer Protocol. 1, *Glossar:* Python

## I

**i.i.d** unabhängig und identisch verteilt (engl.) independent and identically distributed. 1, 230, 231

**IDE** Integrated development environment. 1, 148

**IFM** infoscore Forderungsmanagement GmbH (Aktuell: paigo). 1

**iMan** Zentrales Benutzermanagement. 1

**IoT** Internet of Things. 1

**iPD** integriertes Planungs- und Dispositionssystem. 1

## J

**JS** JavaScript. 1, 158

**JSON** JavaScript Object Notation. 1, *Glossar:* JSON

**json** JavaScript Object Notation. 1, *Glossar:* JSON

## K

**K.I.** Künstliche Intelligenz. 1, 239

**KiN-A** Kundenbetreuer im Nahverkehr. 1

## L

**LRM** Lineares Regressionsmodell. 1, 229, 230

**LV** Leistungsvereinbarung. 1

## M

**ML** Machine Learning. 1

**ML** Machine Learning. 1, 221, 225, 239

**MLE** Maximum Likelihood Estimation. 1  
**MLR** Marketing Leiter Runde. 1  
**MOSAIK** Mobile Strategie im Nahverkehr (vorherige Bezeichnung: Folgegeneration MT (FGMT) Kontroll App für Prüfer). 1  
**mpB** mentalen und physischen Besonderheiten. 1, 257  
**MSE** Mittlere quadratischer Fehler. 1, 221, 236  
**MVG** Münchener Verkehrsgesellschaft mbH. 1  
**MZVT** MOSAIK Zahlungsverkehrsterminal. 1

## N

**Na** Not available. 1, 210, *Glossar:* Na (R)  
**NaN** Not a Number. 1, 210, 211, 213, *Glossar:* NaN  
**NFMZ** Neues-Familienmitglied Zeitstrahl. 1, 257  
**NN** (Artificial) Neural Networks. 1

## O

**ODBC** Open Database Connectivity (Connection). 1, 156, *Glossar:*  
**OE** Organisationseinheit. 1  
**OfW** Ohne festen Wohnsitz. 1  
**OLS** Ordinary Least Square Estimation. 1  
**OLT** Online Ticket. 1  
**OOP** Objekt-Orientierte-Programmierung. 1, 89, 102, 167

## P

**P.RM-SBM-M1** Kundeninteraktion. 1  
**P.RM-SBM-M11** Fahrgast- und Verbundmarketing. 1  
**P.RM-SBM-M12** Prüfdienst. 1  
**P.RM-SBM-M13** Fahrgastinformation. 1  
**PBDP** Power BI Deployment Pipeline. 1  
**PDOS** Prüfdienst ohne Sicherheit (Externe zivile Prüfer der DB Sicherheit). 1  
**PEL** Physische Entwicklungslimitation. 1, 257  
**PKF** Pivot-Koordinaten-Filter. 1, 55–57, 59–61, 63  
**PP** Power Platform. 1, 71, 72, 74, 76, 77  
**PQE** Power Query Engine. 1, 65–67

## Q

**QMC** Qlik Management Console. 1, 158

## R

**R** R - Statistical Programming Language. 1, 210, 211  
**RDG** Rechtsdienstleistungsgesetz. 1  
**RFU** Rahmen des jährlichen Fortbildungsunterrichts. 1  
**RiL** Richtlinie. 1  
**RIS** Reise- und Informationssystem. 1  
**RKI** Robert-Koch-Institute. 1, 195  
**RL** Reinforcement Learning. 1  
**RTA** Reisenden-Ticket-Automat. 1

## S

**S3** Simple Storage Services. 1  
**SBF** Schutzbedarfseinstellung. 1  
**SBM** S-Bahn München. 1  
**SDVS** Securitas Deutsche Verkehrssicherheit und Service GmbH (Externe Prüfer von Securitas). 1

**SECU** Externe Prüfer von Securitas. 1  
**SEV** Schienenersatzverkehr. 1  
**SFTP** Secure File Transfer Protocol. 1  
**SM** Statistisches Model. 1, 251, 252  
**SQE** erklärenden Quadradsumme. 1, 221, 236  
**SQL** Structured Query Language. 1, 126, 156, 268, *Glossar*: SQL  
**SQT** totale Quadradsumme. 1, 221, 236

## T

**TV** Tarfiverstoß (ursprünglich Vertragsstrafe). 1

## U

**UI** User Interface. 1, 77  
**URL** Uniform Resource Locator. 1, *Glossar*: URL  
**UX** User Experience. 1

## V

**VAS** Vertriebs- und Abrechnungssoftware. 1  
**VB** Verkehrsvertrieb (VB). 1  
**VBA** Visual Basic Application. 1, 89–91, 97, 100, 103–105  
**VCS** Version Controll System. 1, 136, 137, 140, 266  
**VDV** Verband Deutscher Verkehrsunternehmen. 1

## W

**W-** Wahrscheinlichkeit(s). 1, 243, 245–255

## X

**xlsx** Microsoft Excel Open XML Spreadsheet with Macros. 1  
**xlsx** Microsoft Excel Open XML Spreadsheet. 1, 21, 181

## Z

**ZK** Zeitlose Komponente. 1, 257  
**ZV** Zufallsvariable ( $k = 1$ )/ Zufallsvektor ( $k > 1$ ) oder Zufallsgröße. 1, 243, 245–247, 249–251, 254  
**ZVT** Zahlung-Verkehr-Terminal. 1

## ¶

**ÖPNV** Öffentlicher Personennahverkehr. 1

# Anhang B

## Glossar

### A

**Amazon Elastic Computing Cloud** Amazon Elastic Compute Cloud (Amazon EC2) bietet eine skalierbare Rechenkapazität in der Amazon-Web-Services(AWS)-Cloud. Amazon EC2 beseitigt die Notwendigkeit, im Voraus in Hardware investieren zu müssen. Daher können Sie Anwendungen schneller entwickeln und bereitstellen. Sie können Amazon EC2 verwenden, um so viele oder so wenige virtuelle Server zu starten, wie Sie benötigen, Sicherheit und Netzwerk zu konfigurieren und den Speicher zu verwalten. Amazon EC2 können Sie auf- oder abwärts skalieren, um auf geänderte Anforderungen oder Datenverkehrsspitzen zu reagieren. Dies reduziert die Notwendigkeit, den Datenverkehr vorauszusagen. . 1

**Anwendungsentwicklerin** Diese Person, wird durch die Anwendungsverantwortliche, in der Entwicklungsumgebung der Maker-Rolle zugewiesen. Diese ist für die Entwicklung der Anwendung oder des Flows zuständig.



. 1

**Anwendungsverantwortliche** Diese Person hat die Projekt/Stakeholder und Entwicklungsverantwortung. Sie erfasst die Anforderungen aus dem Geschäftsbereich. Die Rolle wird der Person zu gewiessen, welche die *Entwicklungsumgebung* über das Digitalportal bestellt.



. 1, 265

**Application Programming Interface** Ein Programmschnittstelle ist eine Interaktionsmöglichkeit, die ein Programm anderen Programmen bietet auf bestimmte Teile der Software und oder Daten zuzugreifen. . 1

## B

**Berechnende Spalte** Diese ergänzen Tabellen, welche im Datenmodell eingefügt sind. Dabei werden Daten erzeugt und auch im Arbeitsspeicher geladen. Diese Spalten lassen keine Aggregations zu und beziehen sich ausschließlich auf jede Zeile in einer Tabelle. . 1, 37

**Build Artifacts** In Kurz: Environment + Complied Output = Artifact. In Lang: Die Umgebung mit alle Werkzeugen, Abhängigkeiten, welche benötigt werden, um die Build (oder Image oder Source) plus tatsächlich compilierte resultate. Kommt es zu einem Fehler, ist alles an einem Ort, um unabhängig von den Konfigurationen, eine Fehleranalyse durchzuführen. Artifact kann auch nur "etwas produziertes" bedeuten. Somit ist *libs/runnables - Compiled Artifacts; Image - Build Step*. . 1, 78

## E

**Enviromentverantwortliche/Mutli-Stage Verantwortliche** Diese Person hat die DB-Goverance und Prozessverantwortung. Sie überwacht die Hostung verschiedener Applikationen/ Flows in der Produktionsumgebung. Die Rolle wird der Person zu gewiessen, welche die *Multi-Stage Umgebung* über das Digital-portal bestellt.



. 1

## G

**Git** Git is a free and open-source distribution VCS. It trackes changes in source code during software development. . 1, 78, 136, 137, 139, 142, 148, 149, 152

**Git Repository** Repositories in GIT contain a collection of files of various different versions of a Project. These files are imported from the repository into the local server of the user for further updatons and modifications in the content of the file. A VCS or the Version Control System is used to create these versions and store them in a specific place termed as a repository *GeeksforGeeks o. D.* . 1, 138

**Git Repository** Repositories in GIT contain a collection of files of various different versions of a Project. These files are imported from the repository into the local server of the user for further updatons and modifications in the content of the file. A VCS or the Version Control System is used to create these versions and store them in a specific place termed as a repository *GeeksforGeeks o. D.* . 1, 137, 139, 140, 143, 144

## I

**Intellisense** Dies ist eine Anwendung zur Auto-Vervollständigung . 1, 91, 97

## J

**JSON** JavaScript Object Notation strukturiert Daten. Dieses Datenformat hat sich in der Webentwicklung verbreitet. Bei Anfragen von Rest-API kann man ein JSON Format erwarten. Ebenso finden JSON-Formate

in Log und Confi-Dateien ihren Nutzen. Es gibt 6 Datenformate, welche in einer JSON Datei abgespeichert werden können. String, Boolean, Integer, null, Array und Object . 1

## M

**M Formula Language** Power Query nutzt die Sprache M. Diese ist eine funktional Sprache ähnlich zu F. Es können mit den Ausdrücken Daten gefiltert und transformiert werden. . 1, 65, 66, 121

**Measure** Dies ist eine berechnenden DAX-Funktion. Dies ist zu unterscheiden zu den *Berechneten Feld* Funktionen, die für Tabellen die nicht im Datenmodell enthalten sind, zu nutzen sind. . 1, 37, 42, 43, 57–59, 61, 62

## N

**Na (R)** In R existiert zu NULL und NaN auch Na. Es handelt sich hier um einen logischen Operator. Die Besonderheit ist, dass dieser, wenn abgefragt, Na zurückgibt.  $Na = Na \rightarrow Na$ . Dieser Operator kann in verschiedenen Datentypen: Array, Vektor, Matrix, etc. vorkommen. <sup>1</sup> . 1

**NaN** NaN ist eine numerische Fehlerabfrage. Es ist ein besonder *float*-Zeiger. Dieser kann aus verschiedensten Gründen auftreten. Der Unterschied zu None ist, dass None bewusst definiert und ausgegeben wird - ähnlich zu Null. Bei NaN handelt sich sich um ein Float wert, der aus den verschiedensten Gründen entstehen kann, immer aber eine Sackgasse representiert, weil von NaN zu anderen Werten nicht transformiert werden kann - Virusbefall von Daten.

Aus diesem Grund ist ein Vergleich von zwei NaN Ausgaben auch nicht zwangsläufig gleich, weil die interne Kodierung von Fall zu Fall unterschiedlich ist. Es kann ebenso sein, dass die Sprache NaN direkt definiert - wie in Python, dass eine Gleichungs-Abfrage  $NaN == NaN \rightarrow false$  zurückgibt. Dies würde dann bedeuten, selbst wenn der Fehlercode der gleiche ist, würde *false* zurückgegeben. Was die Identität *id()* angeht, sind Variablen mit *np.nan* gleiche Objekte. Zwischen Dataframe erstellt NaN, NumPy, Math, Etc. Können und sind meistens die Objekt-Identitäten anderen, weshalb eine Abfragen von Dataframe erstellten NaN und NumPy NaN nicht in *true* sondern *false* resultiert. . 1, 210, 211

**None** None ist ähnlich zu NULL. Es ist ein eigener Datentype: *NoneType*. Dieser Datentyp kann nur einen Wert besitzen *None*. Zum Vergleich Integer kann Werte von  $-3, 2, 10, \dots$  besitzen. Aus diesem Grund ist  $None == None true$ . Ebenso ist *None is None true*, weil es nur ein Objekt gibt, welches diesen Wert besitzt. In Python werden Varialben nicht zuerst deklariert. Diese bedeutet, diese halten immer eine Wert in dem entsprechenden Typ. Funktionen, welche keinen Rückgabewert wieder geben, nutzen *None*: *def no-return(void): pass* → *print(no-return):None*. Die normale Ausgabe der Funktion zeigt nicht an. Diese hat damit zu tun, dass Python dies unterdrückt.

Die Instanz *None* gibt es in Python nur einmal, wird mehrere Objekte *None* zugewiesen, so zeigen alle Objekte, auf die selbe Instanz. <sup>2</sup> . 1, 210, 211

**NULL (R)** In CPP oder R wird *NULL* verwendet, wenn Variablen deklariert werden. Der Pointer zeigt auf *NULL*, und die Variablen bleiben vorher leer: *int height;* . 1, 210, 211

## O

**Open Database Connectivity (or Connector)** Es handelt sich hierbei um eine API Verbindung zu einem DBMS. Das Ziel ist, dass eine Datenverbindung zu Datenbanksystem hergestellt werden kann, die unterschiedlicher Natur und oder auf unterschiedlichen Plattformen laufen. . 1

## P

**Python** The Hypertext Transfer Protocol is an application layer protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web, where hypertext documents include hyperlinks to other resources that the user can easily access, for example by a mouse click or by tapping the screen in a web browser. *Wikipedia2 o. D.* . 1

**Python** Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. *Wikipedia1 o. D.* . 1

---

<sup>1</sup>[Link](#) [Link](#)

<sup>2</sup>[Link](#)

## **Q**

**Qlik Sense** Qlik Sense ist eine Business Analytics Software, welche die QIX-Engine Technologie nutzt. . 1, 157, 158, 161, 164

## **S**

**scikit learn** Es enthält sich um ein Packet, welches Algorithmen für maschinelles Lernen in Python beinhalten.

Fragestellungen aus dem Data Scince Bereich können damit beantwortet werden. Ebenso enthält scikit learn zusätzliche Funktionen die bei Preprocessing, Feature Extraction, Evaluation und weiteres helfen. Dier Name setzt sich aus Science und Toolkit zusammen. Abgekürzt heißt es sklearn. . 1, 226

**Shell** Eine Shell ist ein Befehl Interpolator (Commandline Interpretar). Diese gibt Zugang zum Betriebssystem.

Im Regelfall wird dieser über ein command-line or graphical user interface gesteuert. . 1, 136

**SQL** SQL ist eine Sprache welche strukturierte Abfragen an Datenbanken stellt. Diese Sprache wird von Datenbanken wie MySQL, Oracle, Mircosoft SQL Server, etc. verwendet. . 1

**SQLite** SQLite ist ein relationales Datenbanksystem. . 1, 128

**Statistik** Das Feld der Statistik ist ein Teilbereich der Stochastik. Der Schwerpunkt der Statistik ist, Datenmenge auszuwerten, zu interpretieren und wie diese verteilt sind. . 1, 241

**Stochastik** Unter dem Feld der Stochastik wird die **Wahrscheinlichkeitstheorie** und die **Statistik** zusammengefasst. Der Begriff Stochastik bedeutet *Kunst des Vermutens*. Zwei Schwerpunkt des Wahrscheinlichkeitslehre ist, zu modellieren, wie wahrscheinlich es ist das ein Ereignis eintritt und welche Ereignise es geben kann. Statistik hat den Schwerpunkt anhand von Daten Rückschlüsse auf Charakteristika einer wahrscheinlichkeitstheoretischen Modells zu ziehen. . 1, 241

## **U**

**URL** A Uniform Resource Locator (URL), colloquially termed a web address, is a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it. A URL is a specific type of Uniform Resource Identifier (URI), although many people use the two terms interchangeably. URLs occur most commonly to reference web pages (http), but are also used for file transfer (ftp), email (mailto), database access (JDBC), and many other applications. [Wikipedia](#) 3 o. D. . 1

## **W**

**Wahrscheinlichkeitstheorie** Die Wahrscheinlichkeitstheorie, auch Wahrscheinlichkeitsrechnung oder Probabilitistik, ist ein Teilgebiet der Mathematik, das aus der Formalisierung, der Modellierung und der Untersuchung von Zufallsgeschehen hervorgegangen ist. . 1, 241

# Anhang C

## Literatur

- Arens, Tilo (2018). *Mathematik*. Springer Spektrum.
- GeeksforGeeks (o. D.). URL: <https://www.geeksforgeeks.org/what-is-a-git-repository/>.
- Henze, Norbert (2018). *Stochastik für Einsteiger*. Springer Spektrum.
- log.1 (o. D.). URL: <https://www.datacamp.com/community/tutorials/logistic-regression-R>.
- log.10 (o. D.). URL: <https://towardsdatascience.com/machine-learning-102-logistic-regression-with-polynomial-features-98a208688c17>.
- log.11 (o. D.). URL: [https://en.wikipedia.org/wiki/Logistic\\_regression#Logistic\\_model](https://en.wikipedia.org/wiki/Logistic_regression#Logistic_model).
- log.12 (o. D.). URL: [https://rufiismada.files.wordpress.com/2012/02/regression\\_\\_linear\\_models\\_in\\_statistics.pdf](https://rufiismada.files.wordpress.com/2012/02/regression__linear_models_in_statistics.pdf).
- log.13 (o. D.). URL: [https://stat.ethz.ch/education/seminsters/WS\\_2006\\_07/asr/Stat.pdf](https://stat.ethz.ch/education/seminsters/WS_2006_07/asr/Stat.pdf).
- log.14 (o. D.). URL: [https://statmath.wu.ac.at/courses/heather\\_turner/glmCourse\\_001.pdf](https://statmath.wu.ac.at/courses/heather_turner/glmCourse_001.pdf).
- log.15 (o. D.). URL: <https://stat.ethz.ch/~stahel/courses/cheming/nlreg.pdf>.
- log.16 (o. D.). URL: <https://iopscience.iop.org/article/10.1088/1755-1315/267/4/042077/pdf#:~:text=In%20polynomial%20logistic%20regression%2C%20the,over%2Dfitting%20phenomenon%20will%20occur..>
- log.17 (o. D.). URL: <https://towardsdatascience.com/logistic-and-decision-boundary-eab6e00c1e8#:~:text=The%20fundamental%20application%20of%20logistic,classes%20or%20multi%2Dclass%20classification..>
- log.2 (o. D.). URL: <https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python>.
- log.3 (o. D.). URL: <https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc>.
- log.4 (o. D.). URL: [https://sebastianraschka.com/faq/docs/logistic\\_regression\\_linear.html](https://sebastianraschka.com/faq/docs/logistic_regression_linear.html).
- log.5 (o. D.). URL: <https://medium.com/analytics-vidhya/logistic-regression-b35d2801a29c>.
- log.6 (o. D.). URL: <https://youtu.be/EVEXFACiqKE>.
- log.7 (o. D.). URL: [https://en.wikipedia.org/wiki/Polynomial\\_regression](https://en.wikipedia.org/wiki/Polynomial_regression).
- log.8 (o. D.). URL: <https://stats.stackexchange.com/questions/92001/polynomial-term-in-logistic-regression>.
- log.9 (o. D.). URL: <https://iopscience.iop.org/article/10.1088/1755-1315/267/4/042077/pdf>.
- Notation. Abbildung (o. D.). URL: <https://www.stksachs.uni-leipzig.de/files/media/pdf/lehrbuecher/mathematik/propaed/abbildungen.pdf>.
- Pyplot-Tutorial (o. D.). URL: <https://matplotlib.org/stable/tutorials/introductory/pyplot.html>.
- Python Data Science Handbook (o. D.). URL: [C:/Users/PaulJulitz/OneDrive%20-%20Deutsche%20Bahn/B%C3%BCcher/Data%20Science/G%20-%20Guide%20for%20Data%20Science/\(Py\)/\(G-DSci\)%20Python%20Data%20Science%20Handbook\\_Plas.pdf](C:/Users/PaulJulitz/OneDrive%20-%20Deutsche%20Bahn/B%C3%BCcher/Data%20Science/G%20-%20Guide%20for%20Data%20Science/(Py)/(G-DSci)%20Python%20Data%20Science%20Handbook_Plas.pdf).
- stackoverflow (o. D.). URL: [https://stackoverflow.com/questions/45034549/difference-between-git-gui-git-bash-git-cmd#:~:text=\(Command%20Line%20prompt\)%20is%20the,cmd%20to%20run%20git%20commands..](https://stackoverflow.com/questions/45034549/difference-between-git-gui-git-bash-git-cmd#:~:text=(Command%20Line%20prompt)%20is%20the,cmd%20to%20run%20git%20commands..)
- Wikipedia1 (o. D.). URL: [https://en.wikipedia.org/wiki/Python\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Python_(programming_language)).
- Wikipedia2 (o. D.). URL: [https://en.wikipedia.org/wiki/Hypertext\\_Transfer\\_Protocol](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol).
- Wikipedia3 (o. D.). URL: <https://en.wikipedia.org/wiki/URL>.

# Anhang D

## Definitions

### List of Integrated Definition

VIII.1 empirisches Bestimmtheitsmaß . . . . .	221
VIII.2 I . . . . .	221
IX.1 Generalisiertes lineares Modell . . . . .	229
IX.2 Regressionsmodell . . . . .	231
IX.3 linearisierbar . . . . .	232
IX.4 Logistisches Modell . . . . .	233
IX.5 Bestimmtheitsmaß . . . . .	236
XI.1 Abbildung . . . . .	242
XI.2 Wahrscheinlichkeitsraum . . . . .	243
XI.3 Ergebnismenge/raum . . . . .	243
XI.4 Ereignismenge . . . . .	244
XI.5 Drei Axiome von Kolmogorov/ Wahrscheinlichkeitsmaß . . . . .	245
XI.6 Zufallsvariable . . . . .	246
XI.7 Borelsche Sigma-Algebra . . . . .	247
XI.8 Zufallsvariable und Messbarkeitseigenschaft . . . . .	247
XI.9 Verteilung einer Zufallsvariable/ Wahrscheinlichkeitsfunktion . . . . .	248
XI.10 Verteilungsfunktion von X . . . . .	248
XI.11 Diskrete Wahrscheinlichkeitsverteilung einer diskreten zufälligen Variablen . . . . .	249
XI.12 Erwartungswert (stetig) . . . . .	250
XI.13 Erwartungswert (Diskret) . . . . .	250
XI.14 Varianz (diskret) . . . . .	251
XI.15 Statistisches Modell . . . . .	252
XI.16 Merkmal . . . . .	254

### List of Loose Definition

VIII.1 Parameterfunktion . . . . .	222
IX.1 Lineares Regressionsmodell . . . . .	232
XI.1 Maßraum . . . . .	243
XI.2 Sigma-Algebra . . . . .	244
XI.3 Maß . . . . .	245
XI.4 Alternative Definition I der Zufallsvariable . . . . .	246
XI.5 Alternative Definition II der Zufallsvariable . . . . .	246
XI.6 Messbarkeitseigenschaft . . . . .	247
XI.7 Parameterraum . . . . .	251
XI.8 Grundgesamtheit . . . . .	251