

WIRTSCHAFTSINFORMATIK 1

DATENBANKEN – AGGREGATFUNKTIONEN UND GRUPPIERUNG

PROF. DR. BERND BLÜMEL, PROF. DR. CHRISTIAN BOCKERMANN, PROF.
DR. VOLKER KLINGSPOR

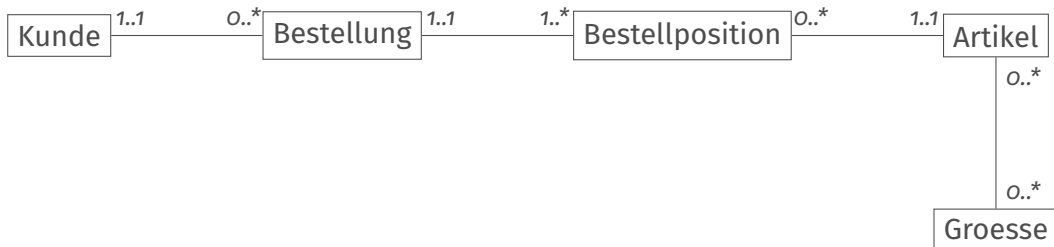
HOCHSCHULE BOCHUM

SOMMERSEMESTER 2025

Inhalt

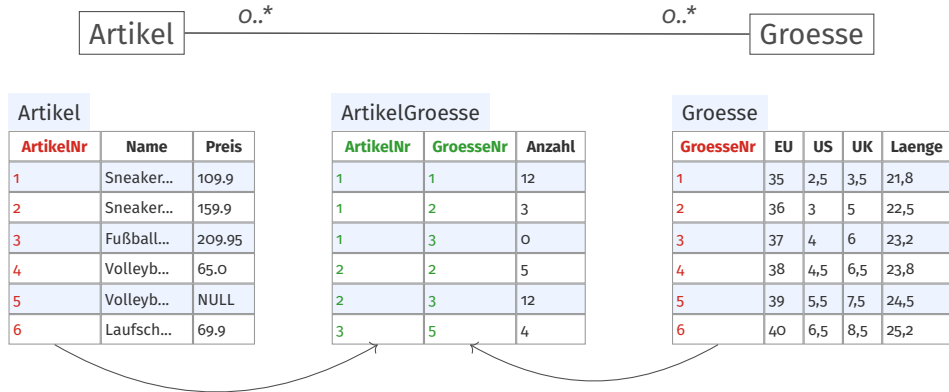
- 1 Wiederholung
- 2 Betrachtungen zum Modell
- 3 Aggregatfunktionen in SQL
- 4 Gruppierung
- 5 Weitere SQL-Befehle
- 6 Zusammenfassung und Ausblick

Modell aus der Vorlesung



Modell aus der Übung





- Wir erstellen eine (**Verbindungstabelle**).
- Die Tabelle enthält die Primärschlüssel der Grundtabellen als **Fremdschlüssel**.



Artikel		
ArtikelNr	Name	Preis
1	Sneaker...	109.9
2	Sneaker...	159.9
3	Fußball...	209.95
4	Volleyb...	65.0
5	Volleyb...	NULL
6	Lausch...	69.9

ArtikelGroesse		
ArtikelNr	GroesseNr	Anzahl
1	1	12
1	2	3
1	3	0
2	2	5
2	3	12
3	5	4

Groesse				
GroesseNr	EU	US	UK	Laenge
1	35	2,5	3,5	21,8
2	36	3	5	22,5
3	37	4	6	23,2
4	38	4,5	6,5	23,8
5	39	5,5	7,5	24,5
6	40	6,5	8,5	25,2

- Wir erstellen eine (**Verbindungstabelle**).
- Die Tabelle enthält die Primärschlüssel der Grundtabellen als **Fremdschlüssel**.
- Diese Fremdschlüssel sind gleichzeitig der **zusammengesetzte Primärschlüssel** der neuen Tabelle.
- Die Verbindungstabelle darf weitere Attribute enthalten.

Artikel

ArtikelNr	Name	Preis
1	Sneaker...	109.9
2	Sneaker...	159.9
3	Fußball...	209.95
4	Volleyb...	65.0
5	Volleyb...	NULL
6	Laufsch...	69.9

ArtikelGroesse

ArtikelNr	GroesseNr	Anzahl
1	1	12
2	2	5
1	2	3
1	3	0
2	3	12
3	5	4

Groesse

GroesseNr	EU	US	UK	Laenge
1	35	2,5	3,5	21,8
2	36	3	5	22,5
3	37	4	6	23,2
4	38	4,5	6,5	23,8
5	39	5,5	7,5	24,5
6	40	6,5	8,5	25,2

```
Select * from Artikel
join ArtikelGroesse on (Artikel.ArtikelNr = ArtikelGroesse.ArtikelNr)
join Groesse on (ArtikelGroesse.GroesseNr = Groesse.GroesseNr)
```

ArtikelNr	Name	Preis	ArtikelNr	GroesseNr	Anzahl	GroesseNr	EU	US	UK	Laenge
1	Sneaker Gazelle	109.9	1	1	12	1	35	2,5	3,5	21,8
2	Sneaker Stan Smith	159.9	2	2	5	2	36	3	5	22,5
1	Sneaker Gazelle	109.9	1	2	3	2	36	3	5	22,5
1	Sneaker Gazelle	109.9	1	3	0	3	37	4	6	23,2
2	Sneaker Stan Smith	159.9	2	3	12	3	37	4	6	23,2
3	Fußballschuh King Ultimate	209.95	3	5	4	5	39	5,5	7,5	24,5

Was ist der Nettopreis der Artikel?

```
Select Name, Preis, Preis / 1.19 as Nettopreis from Artikel
```

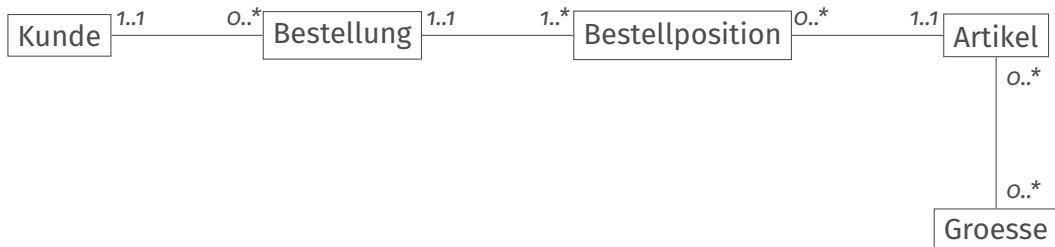
Name	Preis	Nettopreis
Sneaker Gazelle	109.9	92.3529411764706
Sneaker Stan Smith	159.9	134.36974789915968
Fußballschuh King Ultimate	209.95	176.42857142857142
Volleyballschuhe Upcourt 5	65.0	54.6218487394958
Volleyballschuhe Gel-Furtherup Damen	NULL	NULL
Laufschuh Electrify Nitro 2 Herren	69.9	58.73949579831933

Wie ist der Gesamtpreis jedes Artikels in den Bestellungen?

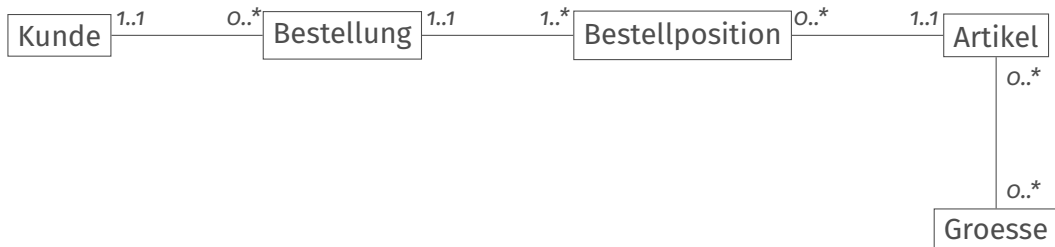
```
Select *, Bestellposition.Anzahl * Artikel.Preis As Gesamtpreis  
from Bestellposition  
join Artikel on (Bestellposition.ArtikelNr = Artikel.ArtikelNr)
```

BestellpositionNr	BestellungNr	ArtikelNr	Anzahl	ArtikelNr	Name	Preis	Gesamtpreis
1	1	1	2	1	Sneaker Gazelle	109.9	219.8
2	1	2	1	2	Sneaker Stan Smith	159.9	159.9
3	2	3	1	3	Fußballschuh King Ultimate	209.95	209.95
4	3	3	2	3	Fußballschuh King Ultimate	209.95	419.9
5	4	4	3	4	Volleyballschuhe Upcourt 5	65.0	195.0
6	5	4	1	4	Volleyballschuhe Upcourt 5	65.0	65.0

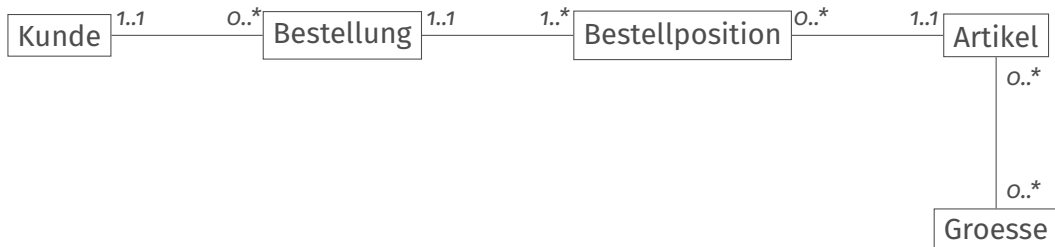
Betrachtungen zum Modell



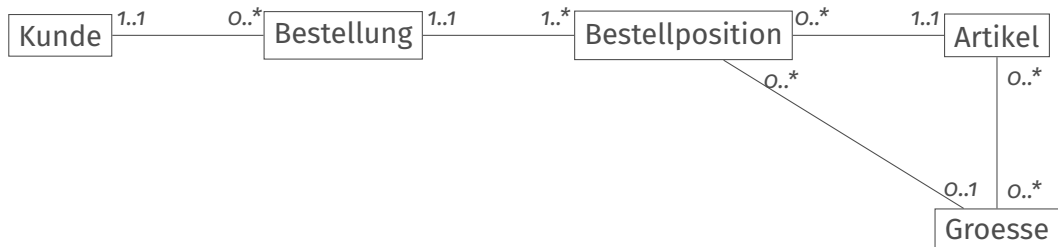
- Wird die Größe der bestellten Artikel gespeichert?



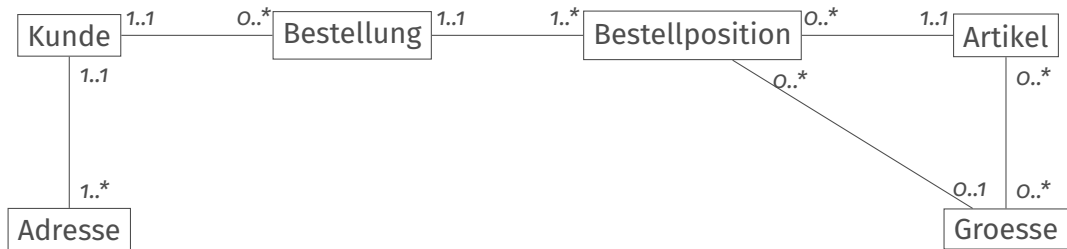
- Wird die Größe der bestellten Artikel gespeichert?
- Was passiert mit den Bestellungen, wenn sich Artikelpreise ändern?



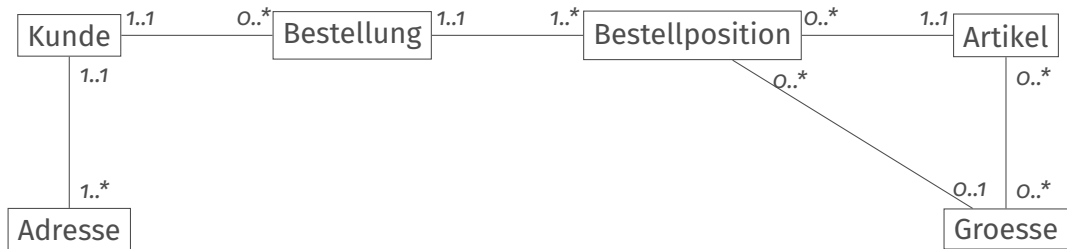
- Wird die Größe der bestellten Artikel gespeichert?
- Was passiert mit den Bestellungen, wenn sich Artikelpreise ändern?
- Haben Kunden nur eine Adresse und was passiert mit den Bestellungen, wenn Kunden umziehen?



- Speicherung der Größe einer Bestellposition als neue Beziehung



- Speicherung der Größe einer Bestellposition als neue Beziehung
- Speichern der Adresse als eigene Entität mit Beziehung zum Kunden



- Speicherung der Größe einer Bestellposition als neue Beziehung
- Speichern der Adresse als eigene Entität mit Beziehung zum Kunden
- Speichern aller zum Bestellzeitpunkts gültigen Daten (z.B. Preis und Liefer-/Rechnungsadresse) in der Bestellung (im Modell nicht sichtbar)

Aggregatfunktionen in SQL

Was können wir mit SQL bisher?

- Suche über verschiedene Tabellen
- Berechnung neuer Attribute basierend auf den Wert der aktuelle Zeile



Sneaker Shop

Hochschule Bochum
Bochum University
of Applied Sciences



Warenkorb: 3 Artikel, 672,00 €

Warenkorb

Warenkorb

Position	Artikel	Anzahl	Beschreibung	Einzelpreis	Preis	
1	1	2	Nike SB Dunk Low April Skateboards	186,00 €	372,00 €	X
2	2	1	Pharrell x NMD_S1 Mahbs 'Earth Strata'	300,00 €	300,00 €	X
				Gesamt:	672,00 €	

Zur Bestellung

Shop-Daten

shop.hsbo.de
Version: 0.0.1



Sneaker Shop

Hochschule Bochum
Bochum University
of Applied Sciences



Warenkorb: 3 Artikel, 672,00 €

Warenkorb

Warenkorb

Position	Artikel	Anzahl	Beschreibung	Einzelpreis	Preis	
1	1	2	Nike SB Dunk Low April Skateboards	186,00 €	372,00 €	X
2	2	1	Pharrell x NMD_S1 Mahbs 'Earth Strata'	300,00 €	300,00 €	X
Gesamt:					672,00 €	

Zur Bestellung

Select Anzahl * Preis

Shop-Daten

shop.hsbo.de
Version: 0.0.1



Sneaker Shop

Hochschule Bochum
Bochum University
of Applied Sciences



Warenkorb: 3 Artikel, 672,00 €

Warenkorb

Warenkorb

Position	Artikel	Anzahl	Beschreibung	Einzelpreis	Preis	
1	1	2	Nike SB Dunk Low April Skateboards	186,00 €	372,00 €	X
2	2	1	Pharrell x NMD_S1 Mahbs 'Earth Strata'	300,00 €	300,00 €	X
				Gesamt:	672,00 €	

Zur Bestellung

Select ?

Shop-Daten

shop.hsbo.de
Version: 0.0.1

Es fehlt:

- Aggregation von Daten mehrerer Zeilen
- Beispiel: Berechne den Gesamtpreis einer Bestellung!

Zunächst: **Aggregatfunktionen** in der Tabelle Artikel

ArtikelNr	Name	Preis
1	Sneaker Gazelle	109.9
2	Sneaker Stan Smith	159.9
3	Fußballschuh King Ultimate	209.95
4	Volleyballschuhe Upcourt 5	65.0
5	Volleyballschuhe Gel-Furtherup Damen	NULL
6	Laufschuh Electrify Nitro 2 Herren	69.9

Bestimme den niedrigsten, den höchsten und den durchschnittlichen Preis sowie die Summe aller Preise

```
Select min(Preis), max(Preis), avg(Preis), sum(Preis) from Artikel
```

min(Preis)	max(Preis)	avg(Preis)	sum(Preis)
65.0	209.95	122.92999999999999	614.65

ArtikelNr	Name	Preis
1	Sneaker Gazelle	109.9
2	Sneaker Stan Smith	159.9
3	Fußballschuh King Ultimate	209.95
4	Volleyballschuhe Upcourt 5	65.0
5	Volleyballschuhe Gel-Furtherup Damen	NULL
6	Laufschuh Electrify Nitro 2 Herren	69.9

Wie viele Artikel habe ich?

Wie viele unterschiedliche Preise habe die Artikel?

Wie viele Artikel haben einen Preis?

```
Select count(*), count(distinct Preis), count(Preis) from Artikel
```

count(*)	count(distinct Preis)	count(Preis)
6	5	5

Wie ist der Gesamtpreis der Bestellung Nr. 1?

Bestellung

BestellungNr	Datum	KundeNr
1	2022-02-12	1
2	2022-03-28	2
3	2022-05-12	3
4	2022-08-01	1
5	2022-09-01	3

BestellPosition

BestellpositionNr	BestellungNr	ArtikelNr	Anzahl
1	1	1	2
2	1	2	1
3	2	3	1
4	3	3	2
5	4	4	3
6	5	4	1

Artikel

ArtikelNr	Name	Preis
1	Sneaker...	109.9
2	Sneaker...	159.9
3	Fußball...	209.95
4	Volleyb...	65.0
5	Volleyb...	NULL
6	Laufsch...	69.9

Wie ist der Gesamtpreis der Bestellung Nr. 1?

Bestellung

BestellungNr	Datum	KundeNr
1	2022-02-12	1
2	2022-03-28	2
3	2022-05-12	3
4	2022-08-01	1
5	2022-09-01	3

BestellPosition

BestellpositionNr	BestellungNr	ArtikelNr	Anzahl
1	1	1	2
2	1	2	1
3	2	3	1
4	3	3	2
5	4	4	3
6	5	4	1

Artikel

ArtikelNr	Name	Preis
1	Sneaker...	109.9
2	Sneaker...	159.9
3	Fußball...	209.95
4	Volleyb...	65.0
5	Volleyb...	NULL
6	Laufsch...	69.9

Die Tabelle Bestellung wird nicht benötigt!

Wie ist der Gesamtpreis der Bestellung Nr. 1?

BestellPosition			
BestellpositionNr	BestellungNr	ArtikelNr	Anzahl
1	1	1	2
2	1	2	1
3	2	3	1
4	3	3	2
5	4	4	3
6	5	4	1

Artikel		
ArtikelNr	Name	Preis
1	Sneaker...	109.9
2	Sneaker...	159.9
3	Fußball...	209.95
4	Volleyb...	65.0
5	Volleyb...	NULL
6	Laufsch...	69.9

Zunächst: Welche Artikel sind in der Bestellung mit der Nr. 1?

```
Select * from  
BestellPosition join Artikel on (BestellPosition.ArtikelNr = Artikel.ArtikelNr)  
where BestellPosition.BestellungNr = 1
```

BestellpositionNr	BestellungNr	ArtikelNr	Anzahl	ArtikelNr	Name	Preis
1	1	1	2	1	Sneaker Gazelle	109.9
2	1	2	1	2	Sneaker Stan Smith	159.9

```
Select * from  
BestellPosition join Artikel on (BestellPosition.ArtikelNr = Artikel.ArtikelNr)  
where BestellPosition.BestellungNr = 1
```

BestellpositionNr	BestellungNr	ArtikelNr	Anzahl	ArtikelNr	Name	Preis
1	1	1	2	1	Sneaker Gazelle	109.9
2	1	2	1	2	Sneaker Stan Smith	159.9

Wie ist der Gesamtpreis der Bestellung Nr. 1?

(Anzahl * Preis) berechnen und das Produkt aufsummieren

```
Select * from  
BestellPosition join Artikel on (BestellPosition.ArtikelNr = Artikel.ArtikelNr)  
where BestellPosition.BestellungNr = 1
```

BestellpositionNr	BestellungNr	ArtikelNr	Anzahl	ArtikelNr	Name	Preis
1	1	1	2	1	Sneaker Gazelle	109.9
2	1	2	1	2	Sneaker Stan Smith	159.9

Wie ist der Gesamtpreis der Bestellung Nr. 1?

(Anzahl * Preis) berechnen und das Produkt aufsummieren

```
Select sum(BestellPosition.Anzahl * Artikel.Preis)  
from BestellPosition join Artikel on (BestellPosition.ArtikelNr = Artikel.ArtikelNr)  
where BestellPosition.BestellungNr = 1
```

sum(BestellPosition.Anzahl * Artikel.Preis)
379.70000000000005

Aggregatfunktionen

- berechnen einen Wert über alle gefundenen Datensätze/Tupel
- liefern als Ergebnis nur ein Tupel zurück
- können die Anzahl, das Minimum und Maximum, den Durchschnitt und die Summe einer Spalte (oder eines mathematischen Ausdrucks) berechnen

Gruppierung

Kunde					
KundeNr	Nachname	Vorname	PLZ	Ort	Strasse
1	Müller	Werner	44789	Bochum	Wittener Str. 79
2	Thomas	Walter	36272	Niederaula	Lerchenweg 13b
3	Maier	Ella	53347	Alfter	Amselweg 28
4	Schmidt	Ulrike	53347	Alfter	Höhenweg 3
5	Meyer	Klaus	44801	Bochum	Wittener Str. 101
6	Maier	Heike	44789	Bochum	Wittener Str. 79

- Wie können wir die Anzahl der Kunden in den jeweiligen Orten berechnen?

Select count(*) from Kunde
where Ort = 'Bochum'

count(*)
3

Select count(*) from Kunde
where Ort = 'Niederaula'

count(*)
1

Select count(*) from Kunde
where Ort = 'Alfter'

count(*)
2

...

KundeNr	Nachname	Vorname	PLZ	Ort	Strasse
3	Maier	Ella	53347	Alfter	Amselweg 28
4	Schmidt	Ulrike	53347	Alfter	Höhenweg 3
1	Müller	Werner	44789	Bochum	Wittener Str. 79
5	Meyer	Klaus	44801	Bochum	Wittener Str. 101
6	Maier	Heike	44789	Bochum	Wittener Str. 79
2	Thomas	Walter	36272	Niederaula	Lerchenweg 13b

```
Select Ort, count(*) from Kunde group by Ort
```

KundeNr	Nachname	Vorname	PLZ	Ort	Strasse
3	Maier	Ella	53347	Alfter	Amselweg 28
4	Schmidt	Ulrike	53347	Alfter	Höhenweg 3
1	Müller	Werner	44789	Bochum	Wittener Str. 79
5	Meyer	Klaus	44801	Bochum	Wittener Str. 101
6	Maier	Heike	44789	Bochum	Wittener Str. 79
2	Thomas	Walter	36272	Niederaula	Lerchenweg 13b

```
Select Ort, count(*) from Kunde group by Ort
```

- **group by** fasst alle Tupel mit demselben Attributwert zu einer Teilmenge zusammen.

	KundeNr	Nachname	Vorname	PLZ	Ort	Strasse
count(*) {	3	Maier	Ella	53347	Alfter	Amselweg 28
	4	Schmidt	Ulrike	53347	Alfter	Höhenweg 3
count(*) {	1	Müller	Werner	44789	Bochum	Wittener Str. 79
	5	Meyer	Klaus	44801	Bochum	Wittener Str. 101
count(*) {	6	Maier	Heike	44789	Bochum	Wittener Str. 79
	2	Thomas	Walter	36272	Niederaula	Lerchenweg 13b

```
Select Ort, count(*) from Kunde group by Ort
```

- **group by** fasst alle Tupel mit demselben Attributwert zu einer Teilmenge zusammen.
- Die Aggregatfunktion wird dann auf jede Teilmenge getrennt angewendet.

Ort	count(*)
Alfter	2
Bochum	3
Niederaula	1

KundeNr	Nachname	Vorname	PLZ	Ort	Strasse
3	Maier	Ella	53347	Alfter	Amselweg 28
4	Schmidt	Ulrike	53347	Alfter	Höhenweg 3
1	Müller	Werner	44789	Bochum	Wittener Str. 79
5	Meyer	Klaus	44801	Bochum	Wittener Str. 101
6	Maier	Heike	44789	Bochum	Wittener Str. 79
2	Thomas	Walter	36272	Niederaula	Lerchenweg 13b

Diagram illustrating the grouping of data by location (Ort) and the application of the count(*) aggregate function. The left table shows the result of the aggregation, and the right table shows the original data. Brackets and arrows indicate the mapping from the original data rows to the aggregated counts.

```
Select Ort, count(*) from Kunde group by Ort
```

- **group by** fasst alle Tupel mit demselben Attributwert zu einer Teilmenge zusammen.
- Die Aggregatfunktion wird dann auf jede Teilmenge getrennt angewendet.
- Das Attribut, nach dem gruppiert wird, kann mit ausgegeben werden.

Ort	count(*)
Alfter	2
Bochum	3

count(*)

count(*)

X

count(*)

KundeNr	Nachname	Vorname	PLZ	Ort	Strasse
3	Maier	Ella	53347	Alfter	Amselweg 28
4	Schmidt	Ulrike	53347	Alfter	Höhenweg 3
1	Müller	Werner	44789	Bochum	Wittener Str. 79
5	Meyer	Klaus	44801	Bochum	Wittener Str. 101
6	Maier	Heike	44789	Bochum	Wittener Str. 79
2	Thomas	Walter	36272	Niederaula	Lerchenweg 13b

```
Select Ort, count(*) from Kunde group by Ort having count(*) >= 2
```

Ort	count(*)
Alfter	2
Bochum	3

Diagram illustrating the aggregation process. The left table shows the result of a `count(*)` operation grouped by `Ort`. The right table shows the original data. Brackets on the right table group rows by `Ort`: Alfter (rows 3, 4) and Bochum (rows 1, 5, 6). Arrows point from these groups to the `count(*)` values in the left table. A red 'X' is placed between the two `count(*)` labels, indicating a filter operation.

KundeNr	Nachname	Vorname	PLZ	Ort	Strasse
3	Maier	Ella	53347	Alfter	Amselweg 28
4	Schmidt	Ulrike	53347	Alfter	Höhenweg 3
1	Müller	Werner	44789	Bochum	Wittener Str. 79
5	Meyer	Klaus	44801	Bochum	Wittener Str. 101
6	Maier	Heike	44789	Bochum	Wittener Str. 79
2	Thomas	Walter	36272	Niederaula	Lerchenweg 13b

```
Select Ort, count(*) from Kunde group by Ort having count(*) >= 2
```

- **having** filtert *nach* der Aggregation alle Ergebnistupel heraus, die nicht der Bedingung entsprechend.

Mit dem erlernten SQL-Befehlen können wir den Umsatz des Unternehmens aus verschiedenen Perspektiven betrachten.

- Umsatz der verschiedenen Bestellungen
- Umsatz der verschiedenen Kunden
- Umsatz der verschiedenen Artikel
- Umsatz in den verschiedenen Orten
- Umsatz in den verschiedenen Monaten

Mit dem erlernten SQL-Befehlen können wir den Umsatz des Unternehmens aus verschiedenen Perspektiven betrachten.

- Umsatz der verschiedenen Bestellungen
 - Umsatz der verschiedenen Kunden
 - Umsatz der verschiedenen Artikel
 - Umsatz in den verschiedenen Orten
 - Umsatz in den verschiedenen Monaten
-
- Die Berechnung des Umsatzes ist dabei immer gleich
 - Die Spalte, über die gruppiert wird, verändert sich

Bestellungen mit Artikeln

BestellpositionNr	BestellungNr	ArtikelNr	Anzahl	ArtikelNr	Name	Preis
1	1	1	2	1	Sneaker Gazelle	109.9
2	1	2	1	2	Sneaker Stan Smith	159.9
3	2	3	1	3	Fußballschuh King Ultimate	209.95
4	3	3	2	3	Fußballschuh King Ultimate	209.95
5	4	4	3	4	Volleyballschuhe Upcourt 5	65.0
6	5	4	1	4	Volleyballschuhe Upcourt 5	65.0

```
Select BestellPosition.BestellungNr, sum(BestellPosition.Anzahl * Artikel.Preis)
from BestellPosition join Artikel on (BestellPosition.ArtikelNr = Artikel.ArtikelNr)
group by BestellPosition.BestellungNr
```

BestellungNr	sum(BestellPosition.Anzahl * Artikel.Preis)
1	379.70000000000005
2	209.95
3	419.9
4	195.0
5	65.0

Kunden mit ihren Bestellungen und den dazugehörigen Artikeln

KundeNr	Nachname	Vorname	BestellungNr	Datum	KundeNr	BestellpositionNr	BestellungNr	ArtikelNr	Anzahl	ArtikelNr	Preis
1	Müller	Werner	1	2022-02-12	1	1	1	1	2	1	109.9
1	Müller	Werner	1	2022-02-12	1	2	1	2	1	2	159.9
2	Thomas	Walter	2	2022-03-28	2	3	2	3	1	3	209.95
3	Maier	Ella	3	2022-05-12	3	4	3	3	2	3	209.95
1	Müller	Werner	4	2022-08-01	1	5	4	4	3	4	65.0
3	Maier	Ella	5	2022-09-01	3	6	5	4	1	4	65.0

```
Select Kunde.KundeNr, Kunde.Nachname, sum(BestellPosition.Anzahl * Artikel.Preis)
from Kunde join Bestellung on (Kunde.KundeNr = Bestellung.KundeNr)
join BestellPosition on (Bestellung.BestellungNr = BestellPosition.BestellungNr)
join Artikel on (BestellPosition.ArtikelNr = Artikel.ArtikelNr)
group by Kunde.KundeNr
```

KundeNr	Nachname	sum(BestellPosition.Anzahl * Artikel.Preis)
1	Müller	574.7
2	Thomas	209.95
3	Maier	484.9

Kunden mit ihren Bestellungen und den dazugehörigen Artikeln

KundeNr	Nachname	Vorname	BestellungNr	Datum	KundeNr	BestellpositionNr	BestellungNr	ArtikelNr	Anzahl	ArtikelNr	Preis
1	Müller	Werner	1	2022-02-12	1	1	1	1	2	1	109.9
1	Müller	Werner	1	2022-02-12	1	2	1	2	1	2	159.9
2	Thomas	Walter	2	2022-03-28	2	3	2	3	1	3	209.95
3	Maier	Ella	3	2022-05-12	3	4	3	3	2	3	209.95
1	Müller	Werner	4	2022-08-01	1	5	4	4	3	4	65.0
3	Maier	Ella	5	2022-09-01	3	6	5	4	1	4	65.0

```

Select Kunde.Ort, sum(BestellPosition.Anzahl * Artikel.Preis)
from Kunde join Bestellung on (Kunde.KundeNr = Bestellung.KundeNr)
join BestellPosition on (Bestellung.BestellungNr = BestellPosition.BestellungNr)
join Artikel on (BestellPosition.ArtikelNr = Artikel.ArtikelNr)
group by Kunde.Ort
    
```

Ort	sum(BestellPosition.Anzahl * Artikel.Preis)
Alfter	484.9
Bochum	574.7
Niederaula	209.95

Kunden mit ihren Bestellungen und den dazugehörigen Artikeln

KundeNr	Nachname	Vorname	BestellungNr	Datum	KundeNr	BestellpositionNr	BestellungNr	ArtikelNr	Anzahl	ArtikelNr	Preis
1	Müller	Werner	1	2022-02-12	1	1	1	1	2	1	109.9
1	Müller	Werner	1	2022-02-12	1	2	1	2	1	2	159.9
2	Thomas	Walter	2	2022-03-28	2	3	2	3	1	3	209.95
3	Maier	Ella	3	2022-05-12	3	4	3	3	2	3	209.95
1	Müller	Werner	4	2022-08-01	1	5	4	4	3	4	65.0
3	Maier	Ella	5	2022-09-01	3	6	5	4	1	4	65.0

```
Select Month(Bestellung.Datum) as Monat, sum(BestellPosition.Anzahl * Artikel.Preis)
from Kunde join Bestellung on (Kunde.KundeNr = Bestellung.KundeNr)
join BestellPosition on (Bestellung.BestellungNr = BestellPosition.BestellungNr)
join Artikel on (BestellPosition.ArtikelNr = Artikel.ArtikelNr)
group by Monat
```

Monat	sum(BestellPosition.Anzahl * Artikel.Preis)
02	379.70000000000005
03	209.95
05	419.9
08	195.0
09	65.0

Weitere SQL-Befehle

```
CREATE TABLE tabellenname (  
    spalte1 datatype,  
    spalte2 datatype,  
    ...  
);
```

```
CREATE TABLE tabellenname (  
    spalte1 datatype,  
    spalte2 datatype,  
    ...  
);
```

Constraints:

NOT NULL Attribut darf nicht NULL sein

```
CREATE TABLE tabellenname (  
    spalte1 datatype,  
    spalte2 datatype,  
    ...  
);
```

Constraints:

NOT NULL Attribut darf nicht NULL sein

UNIQUE Attribut muss eindeutig sein (darf nicht mehrfach vorkommen)

```
CREATE TABLE tabellenname (  
    spalte1 datatype,  
    spalte2 datatype,  
    ...  
);
```

Constraints:

NOT NULL Attribut darf nicht NULL sein

UNIQUE Attribut muss eindeutig sein (darf nicht mehrfach vorkommen)

PRIMARY KEY Kennzeichnet den Primärschlüssel (impliziert NOT NULL und UNIQUE)

```
CREATE TABLE tabellenname (  
    spalte1 datatype,  
    spalte2 datatype,  
    ...  
);
```

Constraints:

NOT NULL Attribut darf nicht NULL sein

UNIQUE Attribut muss eindeutig sein (darf nicht mehrfach vorkommen)

PRIMARY KEY Kennzeichnet den Primärschlüssel (impliziert NOT NULL und UNIQUE)

FOREIGN KEY Kennzeichnet einen Fremdschlüssel

```
CREATE TABLE tabellenname (  
    spalte1 datatype,  
    spalte2 datatype,  
    ...  
);
```

Constraints:

NOT NULL Attribut darf nicht NULL sein

UNIQUE Attribut muss eindeutig sein (darf nicht mehrfach vorkommen)

PRIMARY KEY Kennzeichnet den Primärschlüssel (impliziert NOT NULL und UNIQUE)

FOREIGN KEY Kennzeichnet einen Fremdschlüssel

CHECK Erlaubt es, beliebige Bedingungen (z.B. Anzahl ≥ 1) zu definieren

```
CREATE TABLE tabellenname (  
    spalte1 datatype,  
    spalte2 datatype,  
    ...  
);
```

Constraints:

NOT NULL Attribut darf nicht NULL sein

UNIQUE Attribut muss eindeutig sein (darf nicht mehrfach vorkommen)

PRIMARY KEY Kennzeichnet den Primärschlüssel (impliziert NOT NULL und UNIQUE)

FOREIGN KEY Kennzeichnet einen Fremdschlüssel

CHECK Erlaubt es, beliebige Bedingungen (z.B. Anzahl ≥ 1) zu definieren

DEFAULT Definiert einen Standardwert beim Einfügen eines Datensatzes, wenn das Attribut nicht gesetzt wird

```
CREATE TABLE tabellenname (  
    spalte1 datatype,  
    spalte2 datatype,  
    ...  
);
```

Constraints:

NOT NULL Attribut darf nicht NULL sein

UNIQUE Attribut muss eindeutig sein (darf nicht mehrfach vorkommen)

PRIMARY KEY Kennzeichnet den Primärschlüssel (impliziert NOT NULL und UNIQUE)

FOREIGN KEY Kennzeichnet einen Fremdschlüssel

CHECK Erlaubt es, beliebige Bedingungen (z.B. Anzahl ≥ 1) zu definieren

DEFAULT Definiert einen Standardwert beim Einfügen eines Datensatzes, wenn das Attribut nicht gesetzt wird

CREATE INDEX Kennzeichnet, dass über das Attribut schnell gesucht werden soll

```
CREATE TABLE Bestellposition (  
    BestellpositionNr INTEGER,  
    BestellungNr INTEGER,  
    ArtikelNr INTEGER NOT NULL,  
    Anzahl INTEGER NOT NULL DEFAULT 1,  
    PRIMARY KEY (BestellpositionNr AUTOINCREMENT)  
    FOREIGN KEY (BestellungNr) REFERENCES Bestellung(BestellungNr),  
    FOREIGN KEY (ArtikelNr) REFERENCES Artikel(ArtikelNr)  
)
```

```
UPDATE Tabelle  
SET spalte1 = wert1, spalte2 = wert12, ...  
WHERE Bedingung;
```

Beispiel:

```
UPDATE Kunde  
SET Ort = "Alfter", PLZ = "53347";
```

```
INSERT INTO Tabelle (spalte1, spalte2, spalte3, ...)  
VALUES (wert11, wert12, wert13, ...), (wert21, wert22, wert23, ...), ...;
```

Beispiel:

```
INSERT INTO BestellPosition (BestellungNr, ArtikelNr)  
VALUES (3, 5), (3, 6);
```

Werte eines Tupels, die nicht durch das INSERT gesetzt werden, werden per AUTO-INCREMENT, mit dem DEFAULT-Wert oder mit NULL belegt.

```
DELETE FROM Tabelle  
WHERE Bedingung;
```

Beispiel:

```
DELETE FROM Groesse  
WHERE EU="40";
```

Achtung: Wird keine Bedingung angegeben, werden alle Tupel aus der Tabelle gelöscht.

Eine Tutorial mit noch viel mehr Beispielen finden Sie unter
<https://www.w3schools.com/sql/default.asp>

Die Zugriffsrechte auf die Daten können detailliert eingestellt werden. Für jeden Benutzer kann für

- alle Datenbanken
- jede einzelne Datenbank
- jede Tabelle
- jede Spalte

festgelegt werden, welche Operationen, insbesondere Lesen und Schreiben, erlaubt sind.

Zusammenfassung und Ausblick

- Detailbetrachtungen im Modell
- Aggregatfunktionen
- Gruppierung
- Tabellen erstellen mit Constraints

- Weitere Arten von Beziehungen?
- Wie speichern wir diese in Tabellen?
- Wie sieht die Klausuraufgabe aus?