

WIRTSCHAFTSINFORMATIK 1

DATENBANKEN – AGGREGATFUNKTIONEN UND GRUPPIERUNG

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

HOCHSCHULE BOCHUM

WINTERSEMESTER 2024/2025

Inhalt

- 1 Wiederholung: Funktionen und Aggregationen
- 2 Wiederholung: Gruppierung
- 3 Datumsfunktionen
- 4 Weitere SQL-Befehle
- 5 Zusammenfassung und Ausblick

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	
1	1	1	2	1	Sneaker Gazelle	109.9	21
2	1	2	1	2	Sneaker Stan Smith	159.9	15
3	2	3	1	3	Fußballschuh King Ultimate	209.95	20
4	3	3	2	3	Fußballschuh King Ultimate	209.95	41
5	4	4	3	4	Volleyballschuhe Upcourt 5	65.0	19
6	5	4	1	4	Volleyballschuhe Upcourt 5	65.0	65

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

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

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

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

Wiederholung: 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?

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	
3	Maier	Ella	53347	Alfter	Amselweg
4	Schmidt	Ulrike	53347	Alfter	Höhenweg
1	Müller	Werner	44789	Bochum	Wittener S
5	Meyer	Klaus	44801	Bochum	Wittener S
6	Maier	Heike	44789	Bochum	Wittener S
2	Thomas	Walter	36272	Niederaula	Lerchenwe

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

KundeNr	Nachname	Vorname	PLZ	Ort	
3	Maier	Ella	53347	Alfter	Amselweg
4	Schmidt	Ulrike	53347	Alfter	Höhenweg
1	Müller	Werner	44789	Bochum	Wittener S
5	Meyer	Klaus	44801	Bochum	Wittener S
6	Maier	Heike	44789	Bochum	Wittener S
2	Thomas	Walter	36272	Niederaula	Lerchenwe

```
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	
count(*)	3	Maier	Ella	53347	Alfter	Amselweg
	4	Schmidt	Ulrike	53347	Alfter	Höhenweg
count(*)	1	Müller	Werner	44789	Bochum	Wittener S
	5	Meyer	Klaus	44801	Bochum	Wittener S
count(*)	6	Maier	Heike	44789	Bochum	Wittener S
	2	Thomas	Walter	36272	Niederaula	Lerchenwe

```
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	
3	Maier	Ella	53347	Alfter	Amselweg
4	Schmidt	Ulrike	53347	Alfter	Höhenweg
1	Müller	Werner	44789	Bochum	Wittener S
5	Meyer	Klaus	44801	Bochum	Wittener S
6	Maier	Heike	44789	Bochum	Wittener S
2	Thomas	Walter	36272	Niederaula	Lerchenwe

Diagram showing the mapping from the original table to the grouped table. Brackets on the right group rows with the same 'Ort' value. Arrows on the left point from these groups to the 'Ort' and 'count(*)' columns of the grouped table.

```
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. 10
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

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. 10
6	Maier	Heike	44789	Bochum	Wittener Str. 79
2	Thomas	Walter	36272	Niederaula	Lerchenweg 13b

Diagram illustrating the aggregation process. Brackets on the right group the rows of the 'Kunde' table by 'Ort'. Brackets on the left group the rows of the 'Ort' table by 'Ort'. A red 'X' is placed between the 'Bochum' row of the 'Ort' table and the 'count(*)' label, indicating that the 'Bochum' group is filtered out because its count (3) is less than the 'having count(*) >= 2' condition.

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

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

Umsatzabfrage für den Sneaker-Shop

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
1	Müller	Werner	1	2022-02-12	1	1	1	1
1	Müller	Werner	1	2022-02-12	1	2	1	2
2	Thomas	Walter	2	2022-03-28	2	3	2	3
3	Maier	Ella	3	2022-05-12	3	4	3	3
1	Müller	Werner	4	2022-08-01	1	5	4	4
3	Maier	Ella	5	2022-09-01	3	6	5	4

```
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	
1	Müller	Werner	1	2022-02-12	1	1	1	1
1	Müller	Werner	1	2022-02-12	1	2	1	2
2	Thomas	Walter	2	2022-03-28	2	3	2	3
3	Maier	Ella	3	2022-05-12	3	4	3	3
1	Müller	Werner	4	2022-08-01	1	5	4	4
3	Maier	Ella	5	2022-09-01	3	6	5	4

```
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

Datumsfunktionen

Datumsangaben in SQL

Bisherige Arten von Spalten/Attributen:

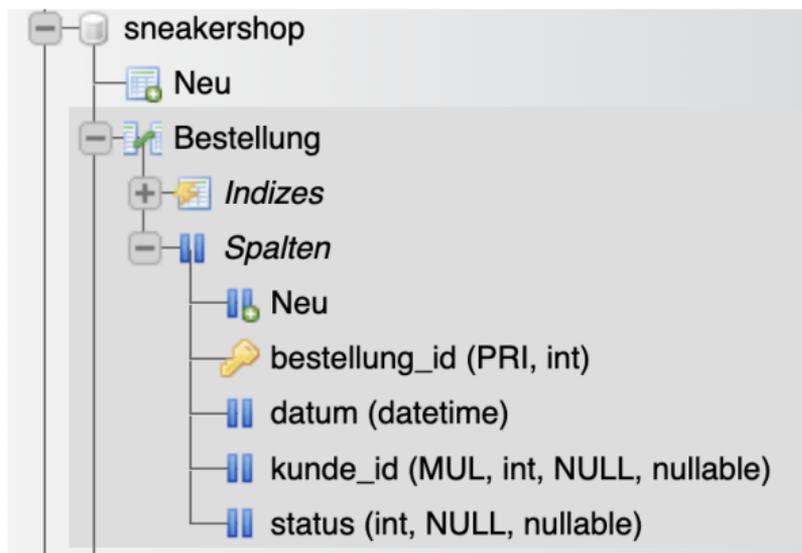
- `String` – Texte in Spalten
- `Int` – ganze Zahlen (1,2,3,..)
- `Double` – Fließkommazahlen (3.14159...)

Datumsspalten in SQL

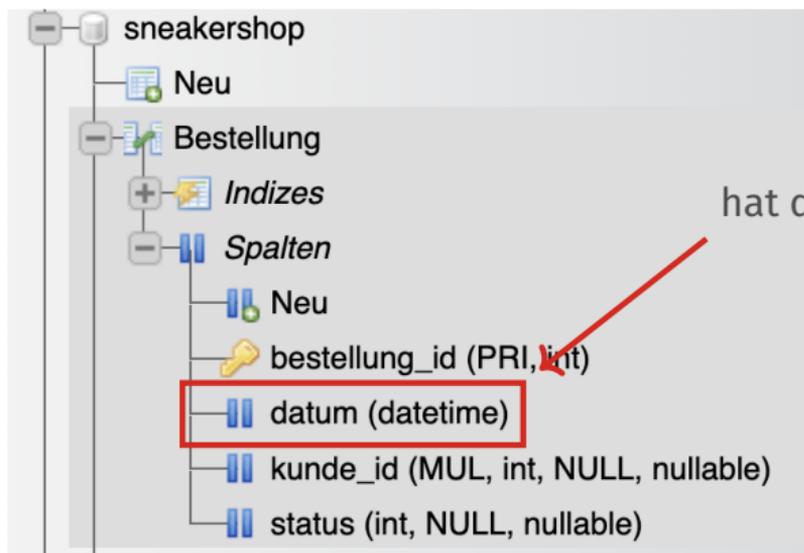
SQL enthält verschiedene Typen für Datumsangaben:

- `Date` – Datum (Tag)
- `Datetime` – Datum und Uhrzeit

Beispiel: **Tabelle Bestellung im Sneakershop**



Beispiel: Tabelle Bestellung im Sneakershop



Spalte Datum
hat den Typ datetime

Datum und Uhrzeit von Bestellungen im Sneakershop:

```
SELECT bestellung_id,datum FROM Bestellung
```

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

Schreibweise für Daten in SQL

Jahr-Monat-Tag Stunde:Minute:Sekunde

Beispiel:

Bestellungen aus dem Mai 2022:

```
SELECT * FROM Bestellung
WHERE Datum >= '2022-05-01' AND Datum < '2022-06-01'
```

Schreibweise für Daten in SQL

Jahr-Monat-Tag Stunde:Minute: Sekunde

Beispiel:

Bestellungen aus dem Mai 2022:

```
SELECT * FROM Bestellung
WHERE Datum >= '2022-05-01' AND Datum < '2022-06-01'
```

Bestellungen vom 1.5.2022 zwischen 0:00 Uhr und 13:00 Uhr:

```
SELECT * FROM Bestellung
WHERE Datum >= '2022-05-01 00:00:00'
AND Datum < '2022-05-01 13:00:00'
```

Datumsfunktionen in SQL

- YEAR(. .) – berechnet das Jahr aus einem Datum
- MONTH(. .) – berechnet den Monat
- WEEKOFYEAR(. .) – berechnet die Kalenderwoche

```
SELECT BestellungNr, Datum, YEAR(Datum), DAYOFWEEK(Datum) FROM Bestellung
```

BestellungNr	Datum	YEAR(Datum)	WEEKOFYEAR(Datum)
1	2022-02-12	2022	06
2	2022-03-28	2022	13
3	2022-05-12	2022	19
4	2022-08-01	2022	31
5	2022-09-01	2022	35

Kunden mit ihren Bestellungen und den dazugehörigen Artikeln

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

```
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	379.700000000000005
03	209.95
05	419.9
08	195.0
09	65.0

Kombiniert: **Datumsfunktion** und **Group By**

Umsatz pro Kalenderwoche für das Jahr 2022:

```
SELECT WEEKOFDAY(Datum) AS KalenderWoche,  
       SUM(BestellPosition.Anzahl * Artikel.Preis) AS Umsatz  
  
FROM   Bestellung  
JOIN   BestellPosition  
       ON (Bestellung.BestellungNr = BestellPosition.BestellungNr)  
JOIN   Artikel ON (BestellPosition.ArtikelNr = Artikel.ArtikelNr)  
  
WHERE YEAR(Datum) = 2022  
GROUP BY KalenderWoche
```

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?