

Der **SELECT** – Befehl ist der Hauptbefehl von SQL. Mit ihm lassen sich beliebige Felder einer oder mehrere Tabellen aus der Datenbank in einer Ergebnistabelle ausgeben. Aufbau:

SELECT → Angabe alle Felder, die angezeigt werden sollen (die gewünschten Daten)

FROM → Angabe der Tabelle(n) in denen diese Felder stehen

WHERE → Bedingungen, die die angezeigten Daten erfüllen müssen

Alle Spalten einer Tabelle ausgeben:

SELECT * FROM KUNDE;

Spalten für die Ergebnistabelle auswählen:

**SELECT name, vorname, strasse, plz, ort
FROM KUNDE;**

Angabe von Bedingungen:

**SELECT name, vorname
FROM KUNDE
WHERE ort = 'Bonn';**

Eine Bedingung muss erfüllt sein

**SELECT name, vorname, strasse, plz, ort
FROM KUNDE
WHERE name = 'Kaufmann' AND vorname = 'Andreas';**

**SELECT name, vorname, strasse, plz, ort
FROM KUNDE
WHERE ort = 'Hamburg' OR ort = 'Bonn';**

Eine der beiden Bedingungen muss erfüllt sein

**SELECT name, vorname, email, strasse, plz, ort
FROM KUNDE
WHERE email IS NOT NULL
AND plz STARTING WITH '50';**

Beide Bedingungen müssen erfüllt sein

Aufgaben:

1. Listen Sie alle Artikel der Tabelle 'artikel' auf, deren Nettopreis höher als 100 € liegt.

ARTIKELNR	BEZEICHNUNG	HERSTELLER	NETTOPREIS	MWST	BESTAND	MINDESTBESTAND	KATEGORIE	BESTELLVORSCHLAG
1	106075	1	137.93	2	100	10	1	1
2	106055	1	172.14	2	100	10	1	1
3	7535	2	111.21	2	100	10	1	1

2. Listen Sie alle Mitarbeiter auf, die in der Abteilung 2 beschäftigt sind.

MITARBEITERNR	NAME	VORNAME	STRASSE	PLZ	ORT	GEH
4	Gerhard	David	Nibelungenstraße 21	52115	Bonn	250
5	Weinert	Eduard	Quellenweg 57	53177	Bonn	200

3. Listen Sie alle Artikel auf, die zur Kategorie Grafikkarten (Kategorienr. 3) gehören und teurer als 100 € sind.

ARTIKELNR	BEZEICHNUNG	HERSTELLER	NETTOPREIS	MWST	BESTAND	MIN
13	Milenium G650	5	137.07	2	100	10
14	Milenium P750	5	197.41	2	100	10
15	Palnelia 512	5	309.48	2	100	10

4. Geben Sie alle Kunden aus, deren Kundennummer größer als 50 ist und die nicht in Köln wohnen

('nicht' wird in SQL mit != ausgedrückt).

name	vorname	strasse	plz	ort	kundennr
Kaminski	Melvin	Neue ABC-Straße 504	22607	Hamburg	51
Schäfer	Natalie	Mittelstraße 50	53225	Bonn	52
Persky	Lisa	Keplerstraße 48	40489	Düsseldorf	53
Steuer	Paul	Langweidenstraße 67	60386	Frankfurt Main	54
Barron	Marion	Im Fuldchen 7	60435	Frankfurt Main	56
Mull	Martin	Rautendeleinweg 29	40593	Düsseldorf	57

Vergleichsoperatoren:

```
SELECT * FROM KUNDE
WHERE name IN ('Maier', 'Mayer', 'Meier', 'Meyer');
```

```
SELECT * FROM MITARBEITER
WHERE gehalt BETWEEN 2000 AND 3000;
```

```
SELECT * FROM KUNDE
WHERE name LIKE 'M__er';
```

```
SELECT * FROM MITARBEITER
WHERE gehalt >= 2000 AND gehalt <=3000;
```

```
SELECT * FROM KUNDE
WHERE name IS NOT NULL;
```

```
SELECT * FROM KUNDE
WHERE name BETWEEN 'C' AND 'M';
```

*NULL bedeutet, dass hier keine
Angaben über den Inhalt vorliegen*

Like-Operator: (Groß- und Kleinschreibung wird nicht beachtet)

Alle Datensätze im Feld name, die mit a oder A
beginnen:

```
SELECT * FROM personen
WHERE name LIKE 'a%';
```

Alle Datensätze, die mit a enden:

```
SELECT * FROM personen
WHERE name LIKE '%a';
```

Alle Datensätze, die ein a beinhalten (kann auch
am Anfang oder am Ende stehen):

```
SELECT * FROM personen
WHERE name LIKE '%a%';
```

Datensätze mit der Buchstabenkombination
d(irgendein Zeichen)na egal:

```
SELECT * FROM personen
WHERE name LIKE 'd_na%';
```

5. Lassen Sie die Daten eines Kunden anzeigen, dessen Namen Sie nicht mehr genau wissen,
nur so viel: Die Anfangsbuchstaben lauten 'Sc' – letzter Buchstabe 'i'.

KUNDENNR	NAME	VORNAME	STRASSE	PLZ	ORT	TELEFON_GESCH	TELEFON_PRIVAT	EMAIL	ZAHLUNGSART
90	Scolari	Peter	Ella-Kay-Straße 85	13357	Berlin	18574639	26574619	peter.scolari@on- line.de	R

KUNDENNR	NAME	VORNAME	STRASSE	PLZ	ORT	TELEFON_G
2	Adler	Felix	Goethestraße 4	30453	Hannover	9856023452
53	Persky	Lisa	Keplerstraße 48	40489	Düsseldorf	19562109
57	Mull	Martin	Rautendeinweg 29	40593	Düsseldorf	10923874
60	Zagorsky	Annette	Isarstraße 75	30559	Hannover	286957483
72	Hart	Melissa	Gurkenweg 4	30655	Hannover	94768592
86	Mandan	Robert	Kösener Weg 8	40489	Düsseldorf	56473329

6. Lassen Sie die Kundendaten der Kunden
anzeigen, die im PLZ-Bereich zwischen
30000 und 41000 wohnen.

Das Feld PLZ hat den Datentyp *char* !

7. Geben Sie die Kundennummer und das Bestelldatum aus
der Tabelle 'Bestellungen' an, wo das Bestelldatum zwischen
dem 02.01.2022 und 04.01.2022 liegt.

Werte von Datum und String – Feldern immer in ''

KundenNr	Bestelldatum
1	2022-01-02
2	2022-01-02
3	2022-01-02
4	2022-01-02
5	2022-01-02
6	2022-01-03
7	2022-01-03
8	2022-01-03
9	2022-01-04
10	2022-01-04
11	2022-01-04
12	2022-01-04
13	2022-01-04
14	2022-01-04

8. Geben Sie alle Spalten der Artikel an, die einen
Nettopreise zwischen 200 und 250 € haben.

ARTIKELNR	BEZEICHNUNG	HERSTELLER	NETTOPREIS	MWS
25	Aureon 5.1 Universe	6	206.03	2
33	LaserJet 1150	4	240.51	2

9. Lassen Sie alle Informationen der Artikel anzeigen, in dessen Bezeichnung das Wort 'Stylus' vorkommt.

ARTIKELNR	BEZEICHNUNG	HERSTELLER	NETTOPREIS	MWST	BESTAND	MINDESTBESTAND	KATEGORIE	BESTELLVORSCHLAG
36	Stylus C44 Plus	8	46.55	2	100	10	7	0
37	Stylus Photo R300	8	150.00	2	100	10	7	1

Ausgabe sortieren mit ORDER BY

Die Ausgabe von Daten in einer fachlich oder ergonomisch sinnvollen Reihenfolge ist eine oft gestellte Forderung an eine Auswertung.

```
SELECT * FROM KUNDE
ORDER BY ort DESC, plz;
```

```
SELECT * FROM KUNDE
ORDER BY ort ASC, plz;
```

Feld 'ort' beginnend mit größtem Wert (letzter Buchstabe im Alphabet) **descending**(absteigend), oder beginnend mit kleinstem Wert (erster Buchstabe im Alphabet) **ascending** (aufsteigend)

1. Sortierkriterium: **ort** bei gleichen Ortsnamen
 2. Sortierkriterium: **name** bei gleichen Namen
 3. Sortierkriterium: **vorname**
- Anzeige: absteigend (Orte mit Z stehen oben)

```
SELECT name, vorname, strasse, plz, ort
FROM KUNDE
WHERE ort = 'Hamburg' OR ort = 'Bonn'
ORDER BY ort DESC, name, vorname;
```

10. Listen Sie alle Artikel. Sortieren Sie zuerst nach der Kategorie und dann alphabetisch (Bezeichnung) auf → gleiche Kategorien werden alphabetisch sortiert.

11. Listen Sie alle Mitarbeiter sortiert nach ihrem Gehalt und dann der Abteilung auf. Das Gehalt soll absteigend sortiert werden → Mitarbeiter mit gleichem Gehalt werden nach der Abteilung sortiert.

12. Listen Sie alle Kunden, die per Nachname (Zahlungsart='N') bezahlen, nach Postleitzahlenbezirken sortiert auf.

Aggregatfunktionen

Die Daten, die nach einer SELECT – Abfrage angezeigt wurden, waren immer Originaldaten, also Texte oder Zahlen, die in den Tabellen so abgelegt waren.

Mithilfe von Aggregatfunktionen werden nun Auswertungen über diese Daten erstellt.

Die wichtigsten:

Da die Ergebnisse in einer Tabelle angezeigt werden, kann mit **AS** ein Feldname angegeben werden.

```
SELECT    COUNT(*)      AS Anzahl_der_Artikel,
          SUM(nettopreis) AS Summe_aller_Preise,
          AVG(nettopreis) AS Durchschnittspreis,
          MAX(nettopreis) AS hoechster_Preis,
          MIN(nettopreis) AS niedrigster_Preis
FROM ARTIKEL
```

Anzahl_der_Artikel	Summe_aller_Preise	Durchschnittspreis	hoechster_Preis	niedrigster_Preis
50	9514.88	190.297600	1766.38	12.07

Die Funktion Count(*) zählt Datensätze:

SELECT COUNT(*) FROM KUNDE;

COUNT(*)
100

Die Tabelle **KUNDE** hat 100 Datensätze

SELECT COUNT(Telefon_Gesch) FROM KUNDE;

COUNT(Telefon_Gesch)
91

Hier werden nur die Datensätze gezählt, bei denen im Feld **Telefon_Gesch** ein Eintrag ist.

Besser wäre hier ein eindeutiger Feldname

**SELECT COUNT(Telefon_Gesch) AS Kunden_mit_Geschaeftstelefon
FROM KUNDE;**

KUNDEN_MIT_GESCHAEFTSTELEFON
91

Kunden ohne Geschäftstelefon können auch durch Subtraktion ermittelt werden:

**SELECT COUNT(*) - COUNT(Telefon_Gesch) AS
Kunden_Ohne_Geschäftstelefon
FROM KUNDE;**

Kunden_Ohne_Geschäftstelefon
9

Wie viele verschiedene Städte gibt es, in denen Kunden wohnen?

SELECT COUNT(DISTINCT ort) FROM kunde

**DISTINCT eliminiert
Duplikate → es werden nur
verschiedene Orte gezählt**

Die Funktion SUM() addiert die Werte eines bestimmten Feldes:

Select SUM(Rechnungsbetrag) FROM BESTELLUNG;

SUM
62372,73

Select SUM(Rechnungsbetrag) FROM BESTELLUNG

WHERE BESTELLDATUM = '2022-01-10'

SUM(Rechnungsbetrag)
1283.35

Hier werden nur die Rechnungsbeträge vom 10.01.2022 addiert.

13. Wie groß ist die höchste Bestellmenge in der Tabelle 'posten'?

MAX(bestellmenge)
10

14. Wie viel wird im Durchschnitt pro Artikel bestellt?

AVG(bestellmenge)
1.5020

15. Welcher Kunde steht alphabetisch am Anfang der Liste?

MIN(name)
Adler

**16. Wie viele Kunden wohnen in Köln? Der Feldname soll 'Anzahl
Kölner Kunden' lauten.**

Anzahl Kölner Kunden
29

Gruppieren (Zusammenfassen) mit GROUP BY

Die Aggregatfunktionen liefern ihre Ergebnisse bezogen auf die ganze Tabelle, also *eine* Zahl.

Wenn aber von jeder Stadt die Anzahl der Kunden gezählt werden soll, werden mehrere Ergebnisse geliefert. Dazu muss aber vorher die Tabelle in Gruppen eingeteilt werden:

GROUP BY ort In dem Feld ort werden alle **gleichen** Einträge zu einer Gruppe zusammen gefasst

Ist eine Tabelle in Gruppen zerlegt worden, so werden die Aggregatfunktionen immer pro Gruppe ausgeführt

Beispiel:

```
SELECT COUNT(*), ort
FROM KUNDE
GROUP BY ort
```

Düsseldorf → 3
Frankfurt Main → 6
Hamburg → 1

Datensätze werden nicht insgesamt (über die ganze Tabelle), sondern pro gleichem Ort gezählt.

STRASSE	PLZ	ORT
Keplerstraße 48	40489	Düsseldorf
Rautendeleinweg 29	40593	Düsseldorf
Kösener Weg 8	40489	Düsseldorf
Max-Hartig-Weg 20	60386	Frankfurt Main
Im Fuldchen 7	60435	Frankfurt Main
Voltastraße 53	60388	Frankfurt Main
Fuldaer Straße 73	60436	Frankfurt Main
Langweidenstraße 67	60386	Frankfurt Main
Görlitzer Straße 41	60313	Frankfurt Main
Fabriciusstieg 47	20251	Hamburg

17. Lassen Sie jeden Ort mit der Anzahl der dort lebenden Kunden anzeigen. Nennen Sie die Felder *Wohnen_in* und *ort*. Es soll nach der Anzahl absteigend sortiert werden.

Wohnen_in	ort
29	Köln
20	Hamburg
9	Bonn
8	Berlin
6	Wiesbaden
6	Frankfurt Main
4	Leverkusen
4	Aachen
3	Düsseldorf

18. Lassen Sie den Maximal-, Minimal- und Durchschnittspreis der Artikel für jede Kategorie anzeigen.

Geben Sie den Feldern mit den berechneten Werten aussagekräftige Namen.

kategorie	Durchschnittspreis	hoechster_Preis	niedrigster_Preis
1	233.912000	602.59	111.21
2	366.550000	1507.76	47.41
3	162.066000	309.48	81.03
4	96.550000	197.41	55.17
5	87.066000	206.03	33.62
6	44.480000	128.45	17.23
7	243.270000	645.69	33.62
8	520.568000	1766.38	93.97
9	29.310000	59.48	13.79
10	36.493333	72.41	12.07

19. Lassen Sie die Anzahl aller Artikel der Kategorie 1 anzeigen

SUM(bestand)

500

20. Geben Sie die Summe der Liefermenge für **jede einzelne** Bestellnummer aus.

21. Geben Sie der durchschnittlichen Nettopreis **aller** Artikel aus.
Geben Sie dem Feld mit dem berechneten Wert einen aussagekräftige Namen.

durchschnittlicher_Preis

190.297600

22. Lassen Sie die Anzahl der Artikel pro Kategorie ausgeben, die teurer als 50 € sind.

Pro Kategorie ein Ergebnis → Gruppieren nach der Kategorie
→ für **jede** Kategorie werden die Artikel gezählt

kategorie	COUNT(*)
1	5
2	4
3	5
4	5
5	3
6	1
7	4
8	5
9	1
10	1

23. Wie hoch ist der Durchschnittsverdienst der Angestellten pro Abteilungen?

abteilung	AVG(gehalt)
1	7166.666667
2	2250.000000
3	2350.000000
4	2350.000000
5	2500.000000
6	1900.000000

Pro Abteilung ein Ergebnis → Gruppieren nach der Abteilung → Durchschnittsverdienst **pro** Abteilung

Having:

Das Ergebnis einer Berechnung durch eine Aggregatfunktion wird durch 'having' eingeschränkt (nicht mit WHERE). Having folgt immer dem group by. Count() wird hier wie ein Spaltenname verwendet.

SELECT ort, COUNT(*)
FROM KUNDE
GROUP BY ort
HAVING COUNT(*) >= 10;

→ Es werden Kundendatensätze **pro Ort** gezählt.
→ **Aber** es werden nur Orte ausgegeben, zu denen **mehr als 9** Kundendatensätze gezählt wurden

ort	COUNT(*)
Hamburg	20
Köln	29

24. Lassen Sie sich die Bestellnummern von allen Bestellungen aus der Tabelle 'posten' ausgeben, bei denen 5 Artikel oder mehr bestellt wurden.
Sorgen Sie für eine Ausgabe der Größe nach.

bestellnr	COUNT(*)
74	19
81	10
75	9
76	5
77	5
78	5
72	5

25. Lassen Sie alle Kundennummern von den Kunden ausgeben, die mehr als 3 Bestellungen aufgegeben haben.

KUNDENNR	COUNT(*)
1	4
23	4

Abfragen über mehrere Tabellen

Die bisher durchgeführten Auswertungen fanden immer auf *einer* Tabelle statt. Inhaltlich zusammenhängende Informationen sind oft auf mehrere Tabellen verteilt und werden bei der SELECT – Abfrage wieder zusammen gesetzt.



Alle Bestelldaten vom Kunden Felix Adler sollen angezeigt werden

```
SELECT kunde.NAME, bestellung.BESTELLDATUM
FROM bestellung, kunde
WHERE
kunde.NAME = 'Adler' AND
kunde.VORNAME = 'Felix'
```

NAME	BESTELLDATUM
Adler	2022-01-02
Adler	2022-01-02
Adler	2022-01-02
Adler	2022-01-02
Adler	2022-01-02
Adler	2022-01-03
Adler	2022-01-03
Adler	2022-01-03
Adler	2022-01-04
Adler	2022-01-04
Adler	2022-01-04

Fehler: Jedes Bestelldatum wird dem Kunden Adler zugeordnet! → SQL weiß nicht welche Datensätze der beiden Tabellen zusammen gehören

Um zusammen gehörende Datensätze zu erkennen sind die Tabellen miteinander verknüpft. Diese Verknüpfung muss im SQL-Statement mit aufgeführt werden. Dazu muss man wissen, über welche Schlüsselfelder die Tabellen verknüpft sind.

KUNDENNR	NAME	VORNAME	STRASSE	PLZ	ORT
1	Loewe	Arthur	Sebastianstraße 134	50737	Köln
2	Adler	Felix	Goethestraße 4	30453	Hannover

Nur dort, wo in beiden Schlüsselfeldern der gleiche Wert steht, gehören die Datensätze zusammen.

BESTELLNR	KUNDENNR	BESTELLDATUM	LIEFERDATUM	RECHNUNGSBETRAG
1	1	2022-01-02	2022-01-11	359.68
2	2	2022-01-02	2022-01-11	107.99
65	2	2022-01-24	2022-01-27	3798.00

Bedingung in SQL:
WHERE
Fremdschlüsselwert =
Primärschlüsselwert

```
SELECT kunde.NAME, bestellung.BESTELLDATUM
FROM bestellung, kunde
WHERE
kunde.NAME = 'Adler' AND
kunde.VORNAME = 'Felix' AND
bestellung.KUNDENNR = kunde.KUNDENNR
```

NAME	BESTELLDATUM
Adler	2022-01-02
Adler	2022-01-24

Da das Feld *kundenr* in beiden Tabellen enthalten ist, muss es jeweils mit dem Namen der Tabelle qualifiziert werden (Punktnotation <Tabellenname>.<Attribut>). Der Ausdruck *kundenr* = *kundenr* wäre falsch und führt bei der Ausführung der Query zu einer Fehlermeldung wie "Spalte nicht eindeutig definiert".

26. Geben Sie die Höhe der gesamten Rechnungsbeträge des Kunden Felix Adler aus. Geben Sie einen passenden Feldnamen an.

Gesamtrechnungsbetrag

3905.99

27. Wie viele Produkte von den einzelnen Herstellern sind im Angebot? Listen Sie den Herstellernamen und die Anzahl ihrer Produkte (die Bezeichnungen müssen gezählt werden).

Die Schlüssel: Tabelle Artikel: *hersteller* (FK)

Tabelle Hersteller: *herstellernr* (PK)

Falsch: ohne Tabellenverknüpfung:

name	COUNT(bezeichnung)	1
Samsung		50
Terratec		50
Microsoft		50
Sony	Keine Zuordnung der Datensätze	50
Canon		50
HP		50
Epson		50
Belinea		50
Matrox		50
Fujitsu		50

Richtig: mit Tabellenverknüpfung:

name	COUNT(bezeichnung)	1
Microsoft		13
Samsung		12
HP	Zuordnung der Datensätze	5
Matrox		5
Terratec		5
Canon		4
Epson		3
Belinea		2
Fujitsu		1

28. Geben Sie zu jedem Mitarbeiter die Bezeichnung seiner Abteilung an. Die Tabellen 'mitarbeiter' und 'abteilung' sind über die Abteilungsnummer miteinander verknüpft.

name	vorname	bezeichnung
Ross	Hagen	Geschäftsführung
Roberts	Patrick	Geschäftsführung
Hummer	Stefan	Geschäftsführung
Gerhard	David	Support
Weinert	Eduard	Support
Michaels	Connie	Rechnungswesen
Osser	Bernd	Rechnungswesen
Koppes	Karin	Einkauf
Wilding	Alexander	Einkauf
Schmidt	Peter	Vertrieb
Müller	Ole	Vertrieb
Meier	Wilhelm	Vertrieb
Schiff	Martin	Vertrieb
Lehne	Luise	Verwaltung
Remsen	Kevin	Verwaltung

29. Um festzustellen, ob die Beispielfirma irgendwelche Ladenhüter in ihrem Lager beherbergt, listen Sie für alle Artikel auf, wie viele von ihnen bestellt wurden und sortieren Sie die Ausgabe absteigend.

BEZEICHNUNG	SUM(POSTEN.BESTELLMENGE)
SC 152A	24
SM-352	16
SD-616	15
CanoScan 5000F	14
Officalet 7140xi	13
CanoScan LiDE 20	13
SC-152A schwarz	13
SM-352	12

Der innere Verbund (INNER JOIN)

Der innere Verbund gibt nur Datensätze aus, die in beiden Tabellen vorhanden sind. Wenn alle Kunden mit ihren Bestellungen aufgelistet werden sollen, werden nur die Kunden aufgeführt, die schon etwas bestellt haben → man erhält dasselbe Ergebnis wie mit der WHERE-Bedingung.

WHERE: **SELECT spaltenliste**
 FROM tabellenliste
 WHERE tabellenname1.primärschlüssel = tabellenname2.fremdschlüssel;

INNER JOIN: **SELECT spaltenliste**
 FROM tabellenname1
 [INNER] JOIN tabellenname2 ON
 tabellenname1.primärschlüssel = tabellenname2. Fremdschlüssel
 [INNER] JOIN tabellenname3 ON
 tabellenname2. primärschlüssel = tabellenname3.fremdschlüssel ;

Beispiel: Die Geschäftsführung der Beispielfirma möchte wissen, wann welcher Kunde etwas bestellt hat. Die dazu benötigten Tabellen kunde und bestellung sind über die Kundennummer miteinander verknüpft.

```
SELECT KUNDE.name, KUNDE.vorname, BESTELLUNG.bestelldatum
FROM KUNDE
INNER JOIN BESTELLUNG ON
KUNDE.kundennr = BESTELLUNG.kundennr;
```

Tabellenverknüpfung
mit inner join

```
SELECT KUNDE name, KUNDE.vorname, BESTELLUNG.bestelldatum
FROM KUNDE, BESTELLUNG
WHERE
KUNDE.kundennr = BESTELLUNG.kundennr;
```

Die gleiche
Tabellenverknüpfung
mit where

Lösen Sie die Aufgaben 27 – 29 mit inner join.

Der äußere Verbund (LEFT JOIN / RIGHT JOIN)

Es sollen alle Kunden und ihre Bestellungen aufgelistet werden, selbst wenn die Kunden nichts bestellt haben – vielleicht weil sie gerade erst einen Katalog angefordert haben.

In diesem Fall existiert zu diesem Kunden ein Datensatz in der Tabelle 'kunde' aber nicht in der Tabelle 'bestellungen'; trotzdem sollen alle Kunden angezeigt werden.

Das ist weder mit WHERE noch mit INNER JOIN möglich. Dafür wird ein äußerer Verbund hergestellt, der OUTER JOIN. Hier können Sie angeben, welche der Tabellen alle Spalten aUSgeben soll → Bei LEFT JOIN werden alle Spalten der Tabelle angezeigt, die 'links' bzw. direkt hinter 'from' steht.

SQL-Syntax:

```
SELECT KUNDE.name, KUNDE.vorname, BESTELLUNG.rechnungsbetrag
FROM KUNDE
LEFT JOIN BESTELLUNG ON KUNDE.kundennr = BESTELLUNG.kundennr;
```

30. Lassen Sie die Namen der Mitarbeiter anzeigen, die ein Jobticket besitzen.

name	mitarbeiternr
Ross	1
Hummer	3

31. Lassen Sie die Namen der Mitarbeiter anzeigen, die ein Jobticket besitzen. Es sollen jedoch alle Namen der Mitarbeiter angezeigt werden.

name	mitarbeiternr
Ross	1
Roberts	NULL
Hummer	3
Gerhard	NULL
Weinert	NULL
Michael	NULL

32. Lassen Sie die Namen der Kunden ausgeben, die den Artikel 'CanoScan LiDE 20' bestellt haben.

name	bezeichnung
Schäfer	CanoScan LiDE 20
Falkner	CanoScan LiDE 20
Fabrizi	CanoScan LiDE 20
Kampmann	CanoScan LiDE 20
Piekarsky	CanoScan LiDE 20
Palk	CanoScan LiDE 20

33. Die Summe der Liefermenge und der Bestellmenge sollen für jede Kategorie (Bezeichnung) ausgegeben werden.

bezeichnung	bestellt	geliefert
Drucker	46	46
Festplatten	34	34
Grafikkarten	29	29
Keyboards	30	30
Laufwerke	80	80
Monitore	34	34
Mäuse	15	15
Scanner	38	38
Software	28	28
Soundkarten	37	214

Unterabfragen

Den maximalen Rechnungsbetrag einer Bestellung herauszufinden:

SELECT MAX(rechnungsbetrag) FROM BESTELLUNG

MAX
4235,97

Wenn der Kunde dieser Bestellung ausgegeben werden soll, würde das mit dem folgenden SQL-Statement nicht funktionieren, weil eine Funktion nur weitere Felder in der SELECT – Zeile zulässt, wenn diese gruppiert werden.

**SELECT MAX(rechnungsbetrag), kundennr
FROM BESTELLUNG**

Mit einer Gruppierung werden alle Datensätze angezeigt, weil sich die Abfrage jetzt auf die entsprechende Kundennummer bezieht. Und jede Kundennummer hat **einen** Rechnungsbetrag (der max-Wert macht hier also keinen Sinn).

**SELECT MAX(rechnungsbetrag), kundennr
FROM BESTELLUNG
GROUP BY kundennr**

MAX(rechnungsbetrag)	kundennr
359.68	1
3798.00	2
322.99	3
736.36	4
109.00	5
497.00	6
267.99	7
193.99	8
278.99	9
388.00	10

weitere Möglichkeit:

**SELECT kundennr
FROM Bestellung
WHERE rechnungsbetrag = MAX(Verkaufspreis)**

führt zu einer Fehlermeldung, weil eine Aggregatfunktion nicht in der WHERE-Klausel

auftreten darf.

Abhilfe: den maximalen Rechnungsbetrag in einer Unterabfrage ermitteln und dann zu diesem Betrag die anderen Kundendaten anzeigen lassen.

```
SELECT kundennr, bestelldatum, rechnungsbetrag
FROM BESTELLUNG
WHERE rechnungsbetrag = 4235,97
(
SELECT MAX(rechnungsbetrag) FROM BESTELLUNG
);
```

Es wird zuerst die Unterabfrage ausgeführt.
Die Hauptabfrage arbeitet dann mit dem Ergebnis der Unterabfrage.

Ergebnis:

kundennr	bestelldatum	rechnungsbetrag
63	2018-01-25	4235.97

Arbeiten mit Variablen

Es können auch mehrere SQL – Abfragen hintereinander getätigt werden. Der größte Rechnungsbetrag kann zuerst in die Variable **@max** gespeichert werden und dann in der nächsten Abfrage benutzt werden.

```
SELECT MAX(rechnungsbetrag) INTO @max FROM BESTELLUNG;
```

```
SELECT KUNDENNR, RECHNUNGSBETRAG, BESTELLDATUM FROM bestellung
```

```
WHERE bestellung.RECHNUNGSBETRAG = @max;
```

34. Ermitteln Sie das Eintrittsdatum des Mitarbeiters, der zuletzt zur Firma gekommen ist.

name	MAX(eintrittsdatum)
Roberts	2018-08-17

35.

Es sollen alle Bestellungen mit ihrer Bestellnummer und ihrem Rechnungsbetrag aufgelistet werden, die einen überdurchschnittlichen Rechnungsbetrag haben (Es muss erst der durchschnittliche Rechnungsbetrag ausgerechnet werden).

bestellnr	rechnungsbetrag
21	599.05
23	626.37
24	457.99
29	797.99
36	736.36
49	626.37
55	457.99
59	1252.73
65	3798.00
69	462.00
72	4235.97
73	768.00
74	2442.26
75	1506.26
76	1429.38
77	1699.98
78	839.99
81	1034.98
82	1052.36
85	2049.00
86	2230.01
87	699.00
93	457.99
96	1908.00
99	1198.00

36.

Es sollen Vorname und Name des zuerst eingestellten Mitarbeiters ausgegeben werden (der Mitarbeiter mit dem kleinsten Datumswert).

name	vorname
Meier	Wilhelm

Numerische, Zeichenketten und Datums – Funktionen:

Die Schreibweise für ein Datum kann unterschiedlich sein. Zum Beispiel verwendet MySQL standardmäßig die Form YYYY-MM-DD. Bei anderen DBMS wird das Datumsformat bei der Installation über die Sprachauswahl festgelegt.

CURRENT_DATE gibt das aktuelle Datum zurück (im Datumsformat der DB).

DATEDIFF(datum2, datum1) gibt die Differenz zweier Daten in Tagen zurück.

Bestellungen sollen spätestens nach 14 Tagen erledigt sein. Es müssen alle Datensätze der Tabelle Bestellung daraufhin überprüft werden.

```
SELECT bestellnr,
DATEDIFF(CURRENT_DATE, bestelldatum)
AS Tage_seit_Bestellung FROM
BESTELLUNG
WHERE lieferdatum IS NULL
```

Die Funktionen **DAY()**, **MONTH()** und **YEAR()** liefern die entspr. Infos aus einem Datum.

Anzahl der Bestellungen vom Januar 2022:

```
SELECT COUNT(*) FROM BESTELLUNG
WHERE MONTH(bestelldatum) = 1 AND
YEAR(bestelldatum) = 2022
```

Mit **CHAR_LENGTH** (Spaltenname) wird die Länge einer Zeichenkette angegeben.

```
SELECT name, CHAR_LENGTH(name)
FROM MITARBEITER;
```

Man kann mit den Werten, die von einer SELECT-Abfrage zurück gegeben werden rechnen.

```
SELECT bezeichnung, nettopreis * 1.19 AS Endpreis
FROM ARTIKEL;
```

Wie hoch ist der **Gesamtwert** aller Artikel?

```
SELECT SUM(bestand * nettopreis) as Artikel_Werte
FROM Artikel;
```

37. Einer der Geschäftsführer der Beispielfirma hat gehört, dass Artikel, deren Bezeichnung mehr als 17 Buchstaben lang ist, von Kunden ungern gekauft werden. Er möchte daher wissen, welche Artikelnamen länger als 17 Zeichen sind.

Bezeichnung	CHAR_LENGTH(Bezeichnung)
Aureon 5.1 Fun Games	20
Aureon 5.1 Universe	19
Natural MultiMedia	18
MultiMedia Keyboard refresh	27
WL Optical Desktop to Bluetooth	31
Basic Optical Mouse SB	22
Wireless IntelMouse Explorer BT	31
Windows 2020 Professional	25
Windows 2020 Home Edition	25
Visual Studio NET 2020	22

38. Wie können Sie feststellen, wie viele Bestellungen bisher insgesamt an jedem zweiten Tag eines Monats insgesamt eingegangen sind?

Bestellungen_am_2ten

10

39. Es sollen die Bestellnummern der Bestellungen, deren Lieferdatum mehr als 10 Tage nach dem Bestelldatum war ausgegeben werden. Das aktuelle Datum sowie das Bestelldatum sollen angezeigt werden. Die Feldnamen sollen wie unten gezeigt benannt werden.

bestellnr	heutiges_Datum	Differenz_Liefer_Bestelldatum	bestelldatum
11	2021-04-19	12	2022-01-04
14	2021-04-19	11	2022-01-04
18	2021-04-19	11	2022-01-07
109	2021-04-19	11	2022-01-31