

DATA SCIENCE

SQL		
Tag	Beschreibung	Beispiel
	<ul style="list-style-type: none">Aktivieren von sql-datenbanken	<ul style="list-style-type: none">Links für Download MySQL:https://dev.mysql.com/downloads/installer/(https://dev.mysql.com/downloads/mysql/)https://dev.mysql.com/downloads/workbench/ <ol style="list-style-type: none">Öffnen der sql-datei in mysqleinmal mit dem „Blitz“ durchlaufen lassen„schemas“ am linken rand des sql-fensters aktualisierendatenbank scheint nun unter den schemas auf

Aufbau von SQL-Abfragen

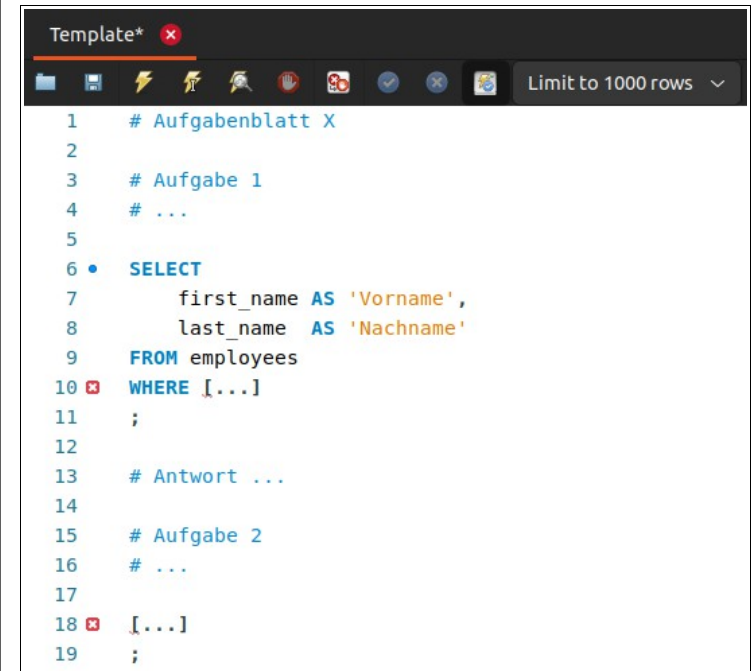
- Syntax

SELECT

Alles auswählen
[**DISTINCT**] *,

Eine konkrete Spalte auswählen
[alias.]spaltenbezeichnung **AS** "Gewünschte Bezeichnung",

Aggregatfunktionen
count([**DISTINCT**] *),
min(spaltenbezeichnung),
max(spaltenbezeichnung),
sum(spaltenbezeichnung),
...



```
1  # Aufgabenblatt X
2
3  # Aufgabe 1
4  # ...
5
6  • SELECT
7      first_name AS 'Vorname',
8      last_name  AS 'Nachname'
9  FROM employees
10 WHERE [...]
11 ;
12
13 # Antwort ...
14
15 # Aufgabe 2
16 # ...
17
18 ✖ [...]
19 ;
```

```
# Verzweigung
CASE WHEN Bedingung1
THEN Schlussfolgerung1
WHEN Bedingung2
THEN Schlussfolgerung2
ELSE Schlussfolgerung3
END AS "Gewünschte Bezeichnung"
FROM linke_tabelle alias

# Eine neue Tabelle joinen
[INNER | LEFT | RIGHT] JOIN rechte_tabelle alias2

# boolescher Ausdruck zwischen linke und rechte Tabelle
ON
alias.Schlüssel1 [< | > | = | !=] alias2.Schlüssel2
[AND | OR]
alias.Spaltenbezeichnungen [LIKE | % | _ | IN | IS NULL] Wert

# Das Ergebnis vor der Aggregation filtern
WHERE
alias.Schlüssel1 [< | > | = | !=] alias2.Schlüssel2
[AND | OR]
alias.Spaltenbezeichnungen [LIKE | % | _ | IN | IS NULL] Wert

# Das Ergebnis gruppieren
GROUP BY [alias.]Spaltenbezeichnungen

# Bedingungen, die nach der Aggregation ausgeführt wird
HAVING alias.Spaltenbezeichnungen [< | > | = | !=] Wert

# Das Ergebnis nach Spaltenbezeichnungen sortieren
ORDER BY [alias.]Spaltenbezeichnungen [DESC]
```

Alle sql-befehle

```
#=====#  
# Query 1 - SELECT  
#=====#  
  
SELECT DISTINCT  
|  
# *  
  emp_no AS 'Mitarbeiternummer',  
  first_name AS 'Vorname',  
  last_name AS 'Nachname'  
  
FROM employees;
```

```
#=====#  
# Query 3 - WHERE  
  
SELECT *  
FROM salaries  
WHERE salary <= 6000;
```

```
#=====#  
# Query 2 - LIMIT  
#=====#  
  
SELECT *  
FROM salaries  
LIMIT 10;
```

```
# Query 4 - WHERE  
  
SELECT  
  first_name,  
  last_name,  
  gender  
FROM employees  
WHERE first_name != 'Berni';
```

```

=====#
# Query 5 - AND OR NOT
=====#

SELECT *
FROM titles
WHERE
    NOT(
        title = 'Senior Engineer'
        AND
        from_date >= '1986-01-01'
    )
    OR to_date < '2000-01-01';

```

```

=====#
# Query 6 - LIKE & WILDCARDS
=====#

SELECT *
FROM employees
WHERE
    first_name LIKE 'Ma__'           # Fängt mit 'Ma' an gefolgt von 2 Buchstaben
    and last_name LIKE '%s%'        # Hat ein 's'
    and birth_date LIKE '%-02-18'    # Endet mit '02-18'
;

```

```
#=====SQL BASICS 2=====#  
#=====#
```

```
#=====#  
# Query 1 - ORDER BY  
#=====#
```

```
SELECT salary  
FROM salaries  
WHERE salary < 50000  
ORDER BY salary DESC;
```

```
# Query 3 - AGGREGATION
```

```
SELECT count(*) as 'Anzahl Mitarbeitende'  
FROM employees;
```

```
#=====#  
# Query 2 - AGGREGATION
```

```
SELECT  
    min(salary) as 'Kleinstes Gehalt',  
    max(salary) as 'Höchstes Gehalt'  
FROM salaries;
```

```
# Query 4 - AGGREGATION
```

```
SELECT  
    count(DISTINCT emp_no) as 'Anzahl Mitarbeitende',  
    sum(salary) as 'Summe aller Gehälter',  
    avg(salary) as 'Durchschnitt aller Gehälter'  
FROM salaries  
WHERE  
    from_date like '1999%'  
    or to_date like '1999%';
```

```
#=====#  
# Query 5 - GROUP BY  
  
SELECT  
    emp_no as 'Personalnummer',  
    sum(salary) 'Summem Gehälter'  
  
FROM salaries  
  
GROUP BY emp_no  
;
```

```
# Query 6 - GROUP BY  
  
SELECT  
    first_name,  
    gender,  
    count(*) as 'Anzahl'  
  
FROM employees  
  
GROUP BY first_name, gender  
  
ORDER BY count(*);
```

```
#=====#  
# Query 7 - IN  
  
SELECT *  
FROM departments  
WHERE dept_name NOT IN ('Finance', 'Marketing', 'Production', 'Sales');
```

```
# Query 8
```

```
SELECT *
```

```
FROM titles
```

```
WHERE title IN (
```

```
    SELECT title
```

```
    FROM TITLES
```

```
    WHERE title = 'Senior Engineer' and emp_no = '10004'
```

```
);
```

```
#=====
```

```
# Query 9 - BETWEEN
```

```
SELECT *
```

```
FROM employees
```

```
WHERE hire_day BETWEEN '1985-11-10' AND '1985-11-22';
```

```
# Query 10 - BETWEEN
```

```
SELECT *
```

```
FROM employees
```

```
WHERE emp_no BETWEEN 10000 AND 20000;
```



```
#=====#
```

```
# Query 11 - INNER JOIN
```

```
#=====#
```

```
SELECT
```

```
    e.first_name,
```

```
    e.last_name,
```

```
    e.gender,
```

```
    dm.dept_no,
```

```
    d.dept_name
```

```
FROM employees e
```

```
INNER JOIN dept_manager dm ON e.emp_no = dm.emp_no
```

```
INNER JOIN departments d  ON d.dept_no = dm.dept_no
```

```
WHERE dm.to_date > CURDATE();
```

```
#=====#  
# Query 12 - HAVING  
#=====#  
  
SELECT  
    e.first_name,  
    e.last_name,  
    count(*) AS 'Anzahl GE'  
  
FROM salaries s  
  
INNER JOIN employees e ON e.emp_no = s.emp_no  
  
WHERE from_date > '1990-01-01'  
  
GROUP BY e.emp_no  
  
HAVING count(*) >= 5  
  
ORDER BY count(*)
```

tabellen

```
# eigene Datenbank in SQL erstellen mit "create database 'db_name'":
create database mitarbeiter;

# eine Datenbank in SQL löschen mit "drop database 'db_name'";
# achtung: "drop database if exists 'db_name'",
# bevor man eine neue anlegt (doppelte databases nicht möglich, fehlermeldung beim anlegen der database!!!)

# in der passenden database aktiv sein mit "use 'db_name'":
use mitarbeiter;

# tabelle neu anlegen: --> "create table 'tb_name'"
#      wichtig: nullwerte ausschließen mit "not null"!!
CREATE TABLE mitarbeitende (
    emp_no INT NOT NULL,
    birth_date DATE NOT NULL,
    first_name VARCHAR(14) NOT NULL,
    last_name VARCHAR(16) NOT NULL,
    gender ENUM('M', 'F') NOT NULL,
    hire_date DATE NOT NULL,
    PRIMARY KEY (emp_no)
);

select * from mitarbeitende;
```

```
# -----  
  
# tabelle aus bestehender anlegen:  
create table gehalt as  
    select *  
    from employees.salaries;  
  
select * from gehalt;  
  
create table dept_mitarbeiter as  
    select *  
    from employees.dept_emp;  
  
create table manager as  
    select *  
    from employees.dept_manager;  
  
create table titel as  
    select *  
    from employees.titles;
```

```
#-----  
# tabelle aus datei anlegen  
# in andere database gehen --> employees aktiv machen:  
# im ausgabe-fenster "export/import" "export recordset.." anwählen,  
# als csv abspeichern, und dann in eigene database gehen --> mitarbeiter aktiv machen:  
select * from departments;  
  
# in der eigenen database mitarbeiter mit rechtsklick anwählen "table data import wizard":  
select * from abteilungen;
```

```
#-----  
# inhalte einfügen durch select:  
# bei gleicher struktur (spalten/bezeichnungen) kann ich aus anderer database mit  
# "insert into 'eigene_tab_name' select * from 'fremde_database_name.fremder_tab_name'"  
insert into mitarbeitende  
select * from employees.employees;  
  
select * from mitarbeitende;
```

```
# -----  
# tabellen nachträglich verändern mit  
# --> "alter table 'tab_name' add 'spalten_name' datentyp not null"  
# --> (löschen einer spalte mit "drop" statt "add")  
alter table mitarbeitende  
add fav_color varchar(16) not null;  
  
# datentyp ändern  
# --> mit "alter table 'tab_name' modify column 'spalten_name' neuer_datentyp not null"  
  
select * from mitarbeitende;  
  
#-----  
# felder updaten: --> "update 'tabellen_name' set 'spalten_name' where ..."  
update mitarbeitende  
set last_name = "Unterberger"  
where emp_no = 10002;
```

```
#-----  
# default-werte: --> spalte löschen und neu anlegen mit "default '...'"  
#                               wichtiger default-wert aufsteigend --> "AUTO_INCREMENT"  
alter table mitarbeitende  
drop column fav_color;  
alter table mitarbeitende  
add fav_color varchar(16) not null default "hellblau";  
  
#-----  
# index setzen: --> "create index 'index_name' on 'tabellen_name' (spalten_name)"  
# (bei häufigem gebrauch --> eigenen index setzen! (allerdings dadurch mehr daten!)  
# löschen von index: "alter table 'tabellen_name' drop index 'index_name';"  
create index idx_emp_no  
on mitarbeitende (emp_no);  
  
#-----  
# view erstellen: virtuelle tabelle mit selbst gewähltem select  
# --> "create view 'view_name' as select ... from ... where ..."  
create view mitarbeiterinnen as  
select * from mitarbeitende where gender = "F"
```

subqueries

Subqueries im SELECT-Block:

Solche Subqueries werden oft benutzt, um einen konstanten Wert schnell auszurechnen, z.B. eine Gesamtsumme, einen Mittelwert, o.Ä. Um ein Subquery im SELECT-Block zu schreiben, muss der Output **genau** eine Zeile und eine Spalte besitzen.

Syntax:

```
SELECT  
( SELECT aggregation([DISTINCT] spaltenbezeichnung)  
FROM tabellenbezeichnung  
[...]  
) AS "Gewünschte Bezeichnung"  
FROM [...]
```

Beispiel:

```
SELECT  
*,  
(SELECT count(DISTINCT emp_id) FROM employees) AS "Anzahl_MA",  
(SELECT avg(salary) FROM salaries) AS "MW_Salary",  
salary / (SELECT avg(salary) FROM salaries) * 100 AS "Salary%"  
FROM salaries  
GROUP BY emp_no ;
```


Subqueries im FROM-Block:

Diese Subqueries werden benutzt, um mit dem Zwischenergebnis der Subquery weiterzurechnen. Hier wird der Output der Subquery als eine ganz normale Tabelle, wie wir sie kennen, betrachtet.

Syntax:

```
SELECT  
[alias.]spaltenbezeichnung AS "Gewünschte Bezeichnung",  
FROM (  
  SELECT  
  spaltenbezeichnung AS "Gewünschte Bezeichnung",  
  ...  
FROM tabellenbezeichnung  
  [...]  
) alias
```

Beispiel:

```
SELECT avg(sq1.salary_pro_dep) AS "MW Salary"  
FROM (  
  SELECT de.dept_no, avg(s.salary) AS "salary_pro_dep"  
  FROM salaries s  
  INNER JOIN dept_emp de ON s.emp_no = de.emp_no  
  GROUP BY de.dept_no  
) AS sq1 ;
```

Subqueries im JOIN-Block:

Subqueries können auch miteinander und mit Tabellen verbunden werden. Die Syntax bleibt genau gleich. Natürlich kann man beliebig viele JOINS zwischen Subqueries und Tabellen erstellen.

Syntax:

```
SELECT
[...]  
FROM [Tabellenbezeichnung | Subquery] alias_1  
[INNER | LEFT | RIGHT] JOIN (  
SELECT  
Spaltenbezeichnung AS "Gewünschte Bezeichnung",  
...  
FROM Tabellenbezeichnung  
[...]  
) alias_2  
ON alias_1.Schlüssel1 [< | > | = | != | ...] alias_2.Schlüssel2
```

Beispiel:

```
SELECT *  
FROM salaries s  
INNER JOIN (  
SELECT de.dept_no, avg(s.salary) AS "salary_pro_dep"  
FROM employees e  
WHERE e.last_name LIKE 'M%'  
) AS sq1  
ON s.emp_no = sq1.emp_no ;
```

Subqueries im WHERE-Block:

Subqueries können auch zum Filtern benutzt werden. Ähnlich den Subqueries im SELECT-Block wollen wir hier wieder mit einem konstanten Wert arbeiten, d.h. eine Spalte und eine Zeile.

Syntax:

```
SELECT  
[...]  
FROM [Tabellenbezeichnung | Subquery] alias  
WHERE [alias.]Spaltenbezeichnung [< | > | = | != | ...] (  
SELECT aggregation([DISTINCT] spaltenbezeichnung)  
FROM tabellenbezeichnung  
[...]  
) AS "Gewünschte Bezeichnung"
```

Beispiel:

```
SELECT *  
FROM employees e  
INNER JOIN salaries s ON e.emp_no = s.emp_no  
WHERE s.salary > (  
SELECT avg(salary)  
FROM salaries  
) ;
```

sql_advanced befehle

```
#=====#  
# Query 1 - LEFT JOIN  
#=====#  
  
SELECT *  
  
FROM employees e  
  
LEFT JOIN dept_manager dm ON e.emp_no = dm.emp_no  
  
LEFT JOIN salaries s ON s.emp_no = e.emp_no  
  
# WHERE dm.emp_no IS NOT NULL AND to_date > CURDATE()  
;
```

```
#=====#  
# Query 3 - UNION  
#=====#  
  
SELECT emp_no  
FROM employees e  
  
UNION  
  
SELECT emp_no  
FROM salaries;
```

```
#=====#  
# Query 2 - RIGHT JOIN  
#=====#  
  
SELECT *  
  
FROM employees e  
  
RIGHT JOIN dept_manager dm ON e.emp_no = dm.emp_no  
  
RIGHT JOIN salaries s ON s.emp_no = e.emp_no  
  
WHERE dm.emp_no IS NOT NULL AND s.to_date > CURDATE()  
;
```

```
#=====#  
# Query 4 - FULL OUTER JOIN  
#=====#  
  
SELECT *  
FROM employees e  
LEFT JOIN dept_manager dm ON e.emp_no = dm.emp_no  
  
UNION # Löscht alle Duplikate  
  
SELECT *  
FROM employees e  
RIGHT JOIN dept_manager dm ON e.emp_no = dm.emp_no;
```

```
#=====#
# Query 5 - NULL
#=====#

SELECT
    e.emp_no as 'Personalnummer',
    ifnull(dm.dept_no, 'Kein Manager') as 'Department'

FROM employees e
LEFT JOIN dept_manager dm ON e.emp_no = dm.emp_no;
```

```
#=====#
# Query 6 - Operatoren
#=====#

SELECT
    salary as 'Gehalt',
    salary + 1000 as 'Gehaltserhöhung',
    salary - 1000 as 'Gehaltsminderung',
    salary * 1.20 as '20% Erhöhung',
    salary / 2    as 'Halbes Gehalt',
    salary % 2    as 'Modulo 2 Gehalt'

FROM salaries;

#=====#
```

```
#=====
```

```
# Query 7 - CASE WHEN
```

```
SELECT
```

```
    emp_no,
```

```
    first_name,
```

```
    last_name,
```

```
    CASE
```

```
        WHEN gender = 'M'
```

```
        THEN 'Männlich'
```

```
        WHEN gender = 'F'
```

```
        THEN 'Weiblich'
```

```
        ELSE 'Keine Angabe'
```

```
    END as 'Geschlecht'
```

```
FROM employees;
```

```
# Query 8 - CASE WHEN
```

```
SELECT
```

```
    salary,
```

```
    CASE
```

```
        WHEN salary > 100000
```

```
        THEN 'Spitze'
```

```
        WHEN salary < 50000
```

```
        THEN 'GERING'
```

```
        ELSE 'Mitte'
```

```
    END AS 'Geahlttsklass'
```

```
from salaries
```

```
where to_date > CURDATE();
```

```

#=====#
# Query 9 - SUBQUERIES
#=====#

SELECT avg(Anzah)

FROM (
    SELECT
        gender as 'Geschlecht',
        count(*) as 'Anzahl'

    FROM employees

    GROUP BY gender
) as Tabelle1;

```

```

#=====#
# Query 11 - INDEX
#=====#

SELECT *
FROM employees
WHERE first_name = 'Mary';

CREATE INDEX vorname_index
ON employees (first_name);

```

```

# Query 10 - SUBQUERIES

SELECT *
FROM employees
WHERE emp_no IN (
    SELECT emp_no
    FROM salaries
    WHERE salary > 100000
)
;

```

Aufbau von Datenbanken

```
# Eine DB erstellen
CREATE DATABASE Datenbankbezeichnung;

# Eine DB löschen (sollte sie existieren)
DROP DATABASE [IF EXISTS] Datenbankbezeichnung;

# Die DB verwenden. In Workbench geht das auch mit Doppelklick
# auf der Datenbank links bei den SCHEMAS.
USE datenbankbezeichnung;

# Eine neue Tabelle in der DB anlegen
CREATE TABLE tabellenbezeichnung (
  Schlüssel INT [NOT NULL] [UNIQUE] [AUTO_INCREMENT],
  Spaltenbezeichnung DATE [NOT NULL] [DEFAULT wert],
  Spaltenbezeichnung VARCHAR(16) [NOT NULL] [DEFAULT wert],
  Spaltenbezeichnung ENUM('M', 'F') [NOT NULL] [DEFAULT wert],
  PRIMARY KEY (Schlüssel),
  FOREIGN KEY (Schlüssel) REFERENCE Tabelle_2 (Schlüssel_2)
) ;

# Eine Tabelle löschen (sollte sie existieren)
DROP TABLE [IF EXISTS] Tabellenbezeichnung;

# Eine Tabelle aus einer existierenden durch einer Query anlegen
CREATE TABLE Tabellenbezeichnung2 AS
SELECT [...]
FROM Tabellenbezeichnung
[...];

# Tabellen können auch automatisch aus einer CSV- oder XLSX-Datei
# mit der Hilfe vom Table Data Import Wizard generiert werden.
```



```
# Spalte nachträglich hinzufügen
ALTER TABLE Tabellenbezeichnung
ADD Spaltenbezeichnung DATE [NOT NULL] ;

# Spalte nachträglich entfernen
ALTER TABLE Tabellenbezeichnung
DROP COLUMN Spaltenbezeichnung ;

# Spalte nachträglich umbenennen
ALTER TABLE Tabellenbezeichnung
RENAME COLUMN Spaltenbezeichnung_alt TO Spaltenbezeichnung_neu;

# Datentyp einer Spalte nachträglich ändern
ALTER TABLE Tabellenbezeichnung
MODIFY COLUMN Spaltenbezeichnung Datentyp;

# Default Wert einer Spalte nachträglich ändern
ALTER TABLE Tabellenbezeichnung
ALTER Spaltenbezeichnung SET DEFAULT 'Wert';

# Default Wert einer Spalte löschen
ALTER TABLE Tabellenbezeichnung
ALTER Spaltenbezeichnung DROP DEFAULT;

# Primärschlüssel hinzufügen
ALTER TABLE Tabellenbezeichnung
ADD PRIMARY KEY (Spaltenbezeichnung);

# Primärschlüssel löschen
ALTER TABLE Tabellenbezeichnung
DROP PRIMARY KEY;

# Feld updaten
```

```
UPDATE Tabellenbezeichnung  
SET Spaltenbezeichnung1 = Wert1, Spaltenbezeichnung2 = Wert2, ...  
[WHERE Bedingung];
```

```
# Inhalte in Tabelle durch einer Query einfügen
```

```
INSERT INTO Tabellenbezeichnung  
SELECT [...]  
FROM Tabellenbezeichnung  
[...]  
;
```

```
# Inhalte in Tabelle manuell einfügen
```

```
INSERT INTO Tabellenbezeichnung (Spalte1, Spalte2, ...)  
VALUES (Wert1, Wert2, ...);
```

```
# Einen Index erstellen
```

```
CREATE [UNIQUE] INDEX Indexbezeichnung  
ON Tabellenbezeichnung (Spalte1, Spalte2, ...);
```

```
# Einen Index löschen
```

```
ALTER TABLE Tabellenbezeichnung  
DROP INDEX Indexbezeichnung;
```

```
# Einen View erstellen
```

```
CREATE VIEW Viewbezeichnung AS  
SELECT [...]  
FROM Tabellenbezeichnung  
[...]  
;
```

```
# Einen View löschen
```

```
DROP VIEW Viewbezeichnung;
```