MySQL Einsteiger

Agenda

1. Technischer Hintergrund / Allgemeines
    - Wofür brauche ich MySQL - Anwendungsbeispiele
    - Technical Structure
    - Was ist SQL ?
    - Excel vs. MySQL
    - Wie starte ich ? (Windows: MYSQL -Server/Workbench - Installation)
    - Installation osx (Mac)
2. Grundlagen
    - Was sind Datenobjekte und welche gibt es ?
    - SQL-Syntax
    - Datenbank-und-Tabellen-verwenden
    - Struktur und Indizes von Tabellen auslesen
3. SELECT's
    - Alle Datensätze abfragen und alias für Spalte setzen
    - Rechnen mit SELECT - Beispiele
    - Beispiel und Übung - Berechnen aus Feldern
    - Beispiel mit Select where
    - Beispiel mit select where in
    - Beispiel mit select und like
    - Beispiele mit select und range
    - Beispiel mit select und not in
    - Beispiel mit select und SUBSTR,LOWER,UPPER,CONCAT
4. IS NULL / IS NOT NULL
    - Beispiel und Übung mit IS NULL / IS NOT NULL
5. DATUM (DATETIME,STR_TO_DATE oder der Unix Timestamp)
    - Beispiel mit select by date
    - Schlechtes Beispiel mit select by date
    - Beispiel mit Unix Timestamp (Umwandeln)
    - Beispiel mit Dateformat
    - Beispiel String in Date umwandeln
6. COUNT / STATISTIK
    - Zählen von Datensätzen
    - Beispiel für Statistikabfrage (MAX,MIN)
7. NUMERISCHE FUNKTIONEN
    - Abrunden / Aufrunden / Kaufmännisch Runden
8. WEITERE STRINGFUNKTIONEN
    - Überblick der Stringfunktionen
    - CHAR_LENGTH und LENGTH
    - Replace - Zeichen ersetzen

1. ORDER BY (Sortierung)
    - Beispiel und Übung order by
2. LIMIT
    - Kombiniertes Beispiel mit Order By und Limit + Übung
3. GROUP BY
    - Example GROUP BY / HAVING
4. JOINS
    - Überblick über JOINS
    - Example join - 2 tables - 1:1
    - Beispiel Join über 3 Tabellen
    - Beispiel Left join
    - Beispiel 2 Tabellen vergleichen
    - Beispiel komplexer Join
5. UPDATE
    - Update with example
6. DELETE
    - Delete with example
7. ÜBUNGEN

- Uebung - Berechnung aus Feld
- Übung order by
- Übung mit Order By und Limit
- Übung mit Select where
- Übung mit select where in
- Übung mit select und like
- Übung 1+2 - Select Range
- Übung mit select und not in
- Übung mit select by date
- Übung mit select by day
- Übung - Zählen von Datensätzen
- Übung - Statistik AVG
- Übung - NOT NULL / NULL
- Übung mit select und SUBSTR,LOWER,UPPER,CONCAT
- Übung mit unix timestamp
- Übung mit Dateformat
- Übung join - 2 tables - 1:1
- Übung - Refresher Tag 2 - morgens - SELECT
- Übung - Refresher Tag 3 - morgens - SELECT
- Optional: Übung megajoin

8. Datenbanken und Tabellen anlegen, verändern und löschen, Daten einfügen
    - Datenbanken anlegen und löschen
    - Übung - Tabelle kurs anlegen
    - Übung - Daten in kurs einfügen
    - Übung - Daten updaten
9. Datentypen
    - Integer - Ganzzahlen
    - Strings - varchar
    - Text
10. Variablen
    - Beispiel mit Abfrage und User-Variable
11. CSV export/import
    - Export von csv-daten aus der Workbench
    - Import von csv-daten mit der Workbench
    - Import von csv-daten als SQL-Befehl in der Workbench
    - Dealing with charset in Excel (CSV)
12. TIPPS & TRICKS
    - Cheatsheet - Auf dem System zurechtfinden - 1. Sichtung
    - Cheatsheet für Selects
    - Restore auf der Kommandozeile (counterpart zu backup)
    - MYSQL-Workbench disable safe-mode
    - Datenbank mit mysql workbench exportieren
    - Datenbank mit mysql workbench importieren
    - Information schema in der Workbench aktivieren
13. Optional (Views & Triggers)
    - Views
    - Triggers
14. Documentation
    - https://dev.mysql.com/downloads/windows/installer/8.0.html
    - https://www.mysqltutorial.org/
    - https://www.w3schools.com/sql/

Technischer Hintergrund / Allgemeines

Wofür brauche ich MySQL - Anwendungsbeispiele

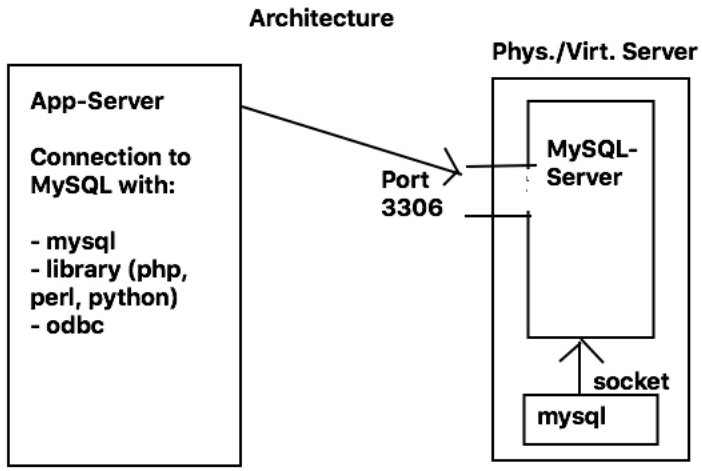
Wann ?

- Wann immer ich Daten persistent speichern möchte und von vielorts gleichzeitig darauf zugreifen möchte
- Client - Server - Architektur
- Wenn ich schnell auf eine große Menge von Daten zugreifen möchte.

Beispiele - Bereiche

1. Infrastruktur o Radius - Server
2. CMS (Content Management System)
   o Wordpress o Typo3.
3. Intranet-Anwendnungen o Zeiterfassung o Ticketsysteme (Jira -> Datenbank)
4. Shops o Amazon.de o shopify
5. Login - Userdaten verwalten (optional)
   o Apache-Webserver (Webseiten) - geschützter privater Bereich o Mailserver - postfix (Nutzerdaten in MySQL speichern)
6. Plattformen o Facebook, Twitter, Youtube, Wikipedia
7. Verarbeitung o Flugticket-Software. text-datei -> mysql -> Anfragen an ein Backend (Zusatzdaten geliefert) -> Datei geschrieben

Technical Structure



Was ist SQL ?

```
SQL - Structured Query Language -> Strukturierte Abfragesprache
Es ist eine standardisierte Programmiersprache
die zur Verwaltung relationaler Datenbanken
und zum Durchführen verschiedener Operationen mit den darin
enthaltenen Daten verwendet wird.
```

```
Will ich Daten aus einer SQL-Datenbank bekommen, brauche ich die Abfragesprache SQL
```

Excel vs. MySQL

```
Excel
====

- Tabellenkalkulation


MySQL
======

- Datenbanksystem (Client / Server)
- Viele Nutzer können gleichzeitig auf die Daten zugreifen.
```

```
Excel Nachteile
============

o kann nicht gut umgehen mit grossen Datenmengen
  O bei grossen Datenmengen langsam bis unmöglich (keine Indizierung)
o Datenintegrität (Formel löschen, Falschschreibung )
O Probleme wenn mehrere Nutzer zugreifen
O User-Management
O Komplettte Tabelle wird in den Arbeitsspeicher geladen, wenn diese geöffnet

Excel Vorteile
===========

O Daten lassen sich einfach importieren
O Schnell mal etwas aufsetzen. (Brauche nur Excel)
O Sortieren nach Spalte ist einfach
O Anpassen von Formeln etc.
O Leicht anpassbar (löschen)
…. aber nur für überschaubare Datenmengen
O Tabelle integrieren in andere Dokumente, die sich dann automatisch aktualisieren
O Bildliche Gestaltung (Farblichkeit, Diagramme), Links einfügen

MySQL - Nachteile
==============

O Komplexer aufzusetzen als Excel
O Ein wenig schwieriger Daten zu optimieren
O Hürde: SQL - Sprache erlernen müssen
O Von Hause aus kein Interface.

MySQL - Vorteile
==============

O Es wurden nur die Daten in den Arbeitsspeicher geladen, die ich abgefragt habe.
O Gute Datenintegrität aufbaubar
O Schnell
o Gut für Daten aggregieren.
O Nicht ein grosses File, sondern vielen kleine (Geschwindigkeitsvorteile)

Wann setze Excel ein ?
==================

O Kleine Projekte mit wenig Daten.
O keine komplexen Abfragen

Wann setzte ich MySQL?
====================

O Grosse Datenmengen, d.h. ich komme mit Excel nicht weiter bzw. Geht nicht oder zu langsam
```

Wie starte ich ? (Windows: MYSQL -Server/Workbench - Installation)

Step-By-Step

```
1. Lade den Installer von mysql herunter (Windows):
https://dev.mysql.com/downloads/installer/

2. Ich führe den Installer aus (diese installiert auch die MySQL Workbench)
(MySQL ist als Dienst installiert im Hintergrund)

Wichtig: Wenn ihr mit Sakila arbeitet wollt, die Documentation -> Beispieldaten installieren
(Beim Installer im Installationsprozess ausführen)

3. Suche die MySQL Workbench (unter Programme -> MySQL -> MySQL Workbench)

4. Vorkonfigurierte Verbindung durch Anklicken öffnen

5. Auf jeden ganz unten in Reiter schemas wechseln durch anklicken)
```

Testdaten -> sakila

```
Installer -> Documentation -> Examples -> und dann letzte MySQL- Version auswählen

Auch nachträglich -> einfach Installer nochmal starten und Add - Button drücken
dort dann Documentation -> Examples
```

Installation osx (Mac)

Download Installer

- Download installer under: https://dev.mysql.com/downloads/mysql/
- Use dmg - archive

Reference

- https://dev.mysql.com/doc/refman/8.0/en/macos-installation-pkg.html

Grundlagen

Was sind Datenobjekte und welche gibt es ?

Was sind Datenbankobjekte

```
Bilden eine Struktur um Daten zu speichern
```

Welche gibt es ? (Klassiker)

Ebene 1 (oberste Ebene) Datenbank (databases/schemas)

```
eine Organisationseinheit: die Datenbank
wie ein Behältnis
unter der Haube: Verzeichnis

z.B. Datenbank sakila
```

Ebene 2 - Tabellen

```
Eine Datenbank kann mehrere Tabellen haben.
Ähnlich eines Vorratsschrankes.
Im Filesystem finden man diese unterhalb der Datenbank mit
dem entsprechenden der Tabelle:
z.B. sakila\actor.ibd

Jede Tabelle hat eine feststehenden Struktur -> Columns (Felder)
```

Ebene 3 - Felder / Columns

```
Die Tabelle hat eine Struktur, die bestimmt wird durch die Columns (Felder)
Jedes Feld (Column) hat einen Datentyp, der bestimmt welche Daten dort rein dürfen.
z.B. Strings (varchar), oder Zahlen (z.B. INT -> Integer)
```

Daten -> Zeilen (ROW)

```
Daten werden zeilenweise in Tabellen geschrieben
Jede Zeile hat ein eindeutiges Merkmale (eine eindeutige Nummer) -> Primärschlüssel (Primary Key)
```

Weitere Datenbankobjekte

```
Views - Schadpotenzial -> 0%
Trigger - Schadpotential -> 50%

-> hier unkritisch, da nur neu eintrag in der Datenbank
CREATE TRIGGER before_employee_update
    BEFORE UPDATE ON employees
    FOR EACH ROW
 INSERT INTO employees_audit
 SET action = 'update',
     employeeNumber = OLD.employeeNumber,
     lastname = OLD.lastname,
     changedat = NOW();

Procedures -
CALL film_in_stock(1,1,@ausgabe);
select @ausgabe;

Function (wie systemfunktionen, nur selbst erstellt)
use sakila;
select get_customer_balance(1,'2015-06-01 12:55:12');

Events (zeitgesteuerte Ereignisse) -
Schadcodepotenzial when aktiviert -> Procedures
https://www.mysqltutorial.org/mysql-triggers/working-mysql-scheduled-event/

Wie weiss ich, dass events generell ausgeführt auf meinem System, wenn vorhanden
```

- Werden events ausgeführt

SQL-Syntax

Kommentare

```
-- Das ist ein Kommentar, der Text beschreibt etwas aber wir nicht vom SQL-Server ausgeführt
```

SQL-Schlüsselworte für Operationen

```
DDL - Daten Definition Language
DML - Data Modelling Language

DDL ändern die Struktur
=======================
CREATE DATABASE #
CREATE SCHEMA  # Datenbank
CREATE TABLE
ALTER TABLE # VErändere die Struktur
DROP

CREATE - erstellen
DROP - Löschen (komplettes Objekt)
ALTER - verändern

DML - Ändert die Inhalte und macht Abfragen
====
SELECT
DELETE - löscht Daten
INSERT - einfügen vpn Daten
UPDATE - Updaten von Daten
```

Datenbank-und-Tabellen-verwenden

```
-- Datenbanken anzeigen
show databases;
-- or:
show schemas;

-- Datenbank verwenden / wechseln
use sakila;

-- Tabellen für ausgewählte Datenbanken anzeigen
show tables;

### Struktur und Indizes von Tabellen auslesen
```

Struktur anzeigen

show fields from actor; describe actor; show create table actor;

Indizes

show indexes from actor;

-- Empfehlung, was ihr können solltet describe actor; show indexes from actor;

```
## SELECT's

### Alle Datensätze abfragen und alias für Spalte setzen


### Examples
```

select * from actor; select first_name,last_name from actor; select first_name as vorname,last_name as nachname from actor;

```
### Übung
```

1. Lasst Euch mit einem SQL-Befehl (select) alle Zeilen aus der Tabelle Film anzeigen
2. Lasst Euch aus der Tabelle nur die Spalten title und die rental_rate anzeigen und benennt die Spalten (alias) in der Ausgabe in Titel und Verleihgebuehr um.

Rechnen mit SELECT - Beispiele

Beispiel 1

```
select 1+1;
select 5*2;
select 2.4532 * 2.134;
select 5/2;
-- Gibt Ganzzahl zurück
select 15 DIV 2;
```

Beispiele 2

```
-- erlaubt sind + - / * DIV (Integer Division)
-- Integer-Division: 15/7 = 2 (nur Ganzzahl wird als Ergebnis zurückgegeben)

-- + - * / DIV(Integer)
SELECT 15 DIV 2 as Ergebnis;
```

Übung

```
Rechne 3 Beispiele:
z.B. 1+4, 2*5, 7/3
```

Beispiel und Übung - Berechnen aus Feldern

Beispiel

```
SELECT amount,1 as 'Preiserhoehung um',amount + 1 as preiserhoehung FROM payment;
```

Übung (sakila)

```
o Lese aus der Tabelle film, den rental_rate aus.
o Wir nehmen an dass dieser Netto ist.
o Berechne den Brutto-Preis rental_rate * 1,19 (1.19)
o Gebe Nettopreis, Steuersatz und Bruttopreis aus (das sind die Überschriften)
```

Beispiel mit Select where

Example (Simple)

```
SELECT * FROM actor WHERE last_name = 'AKROYD' or last_name = 'GABLE';
```

Example (mit Klammern)

```
SELECT * FROM actor WHERE (last_name = 'Akroyd' and first_name = 'Christian') or (last_name = 'Gable' and first_name =
'Christian');
```

Übung

```
* Alle Datensätze aus actor anzeigen bei denen der Nachname Akroyd oder der Vorname Christian ist.
```

Beispiel mit select where in

Example

SELECT * FROM actor WHERE first_name IN ('JOE','ED','JENNIFER');

Übung

```
* Sucht Euch 3 Citys und lasst die Datensätze aus der Tabelle city dazu ausgeben
* Frage: Welches Feld verwenden ?
* mit where ... IN umsetzen
```

Beispiel mit select und like

Example

```
SELECT * FROM actor WHERE last_name like 'D%';

-- Starts with D and ends with S
select * from actor where last_name like 'D%S';
```

Exercise

```
## Lass Euch alle actor ausgeben, deren Nachname auf N endet.
```

Beispiele mit select und range

Beispiel 1

```
SELECT * FROM actor where actor_id > 100 and actor_id < 150;

## 10 und 50 inkludiert
select * from actor where actor_id >= 10 and actor_id <= 50;
```

Beispiel 2

```
SELECT * FROM actor where actor_id between 100 and 150;
```

Übung 1

```
- Gebe alle Adressen (address) aus, wo der postal_code > 10000 und postal_code < 20000 ist
oder < 90000 und > 50000
```

Übung 2

```
- Lasst euch Filme (film) anzeigen deren Wiederbeschaffungspreis (replacement_cost) zwischen 18.99 und 20.99 ist. (inkl)
```

Übung 3

```
Lass Euch alle Einträge aus dem inventory mit between im Bereich 10 bis 100 anzeigen (inkl 10 und 100) - inventory_id
```

Beispiel mit select und not in

Beispiel:

```
SELECT * FROM actor WHERE last_name NOT IN ('AKROYD','DAVIS');
```

Übung:

```
Lasst Euch alle Adressen (address) anzeigen, die nicht im District QLD oder Alberta sind
```

Beispiel mit select und SUBSTR,LOWER,UPPER,CONCAT

Example 1

```
## Zusammenkleben
SELECT concat(title,' ',release_year,' ',description) as listeneintrag FROM film;
SELECT concat ('hans und lotta',' ','haben glueck');

## Teilstring
SELECT substr('Zusammengesetztes Wort',1,2);

## Kleinschreibung
select lower('SONNENSCHEIN');
select lower(last_name) from actor;

## Grossschreibung
select upper('klein');

## Kombiniert
SELECT concat('Ausgabe: ',substr(description,1,20)) as listeneintrag FROM film;
SELECT upper(substr(description,1,20)) from film;
SELECT lower(substr(description,1,20)) from film;
```

Example 2

```
select 'test' ' ist gut ' 'verlaufen';
```

Exercise

```
-- Db: sakila
-- Tabelle: actor
Gebt folgende Felder aus (in einer Abfrage)

o Ausgabe Spalte 1: last_name gross geschrieben
o Ausgabe Spalte 2: first_name klein geschrieben
o Ausgabe Spalte 3: ausgabe mit zusammengeklebt last_name + ' ' + first_name (ohne +)
o Ausgabe Spalte 4: erste beiden Buchstaben vom Nachnamen.
```

```
HINT: -> UPPER, LOWER, CONCAT, SUBSTR
```

Documentation

- https://dev.mysql.com/doc/refman/8.0/en/string-functions.html
- https://www.w3schools.com/sql/func_mysql_concat.asp
- https://www.mysqltutorial.org/sql-concat-in-mysql.aspx

IS NULL / IS NOT NULL

Beispiel und Übung mit IS NULL / IS NOT NULL

Example

```
SELECT * FROM rental WHERE return_date IS NOT NULL;
SELECT * FROM rental WHERE return_date IS NULL;
```

Exercise

```
## Database: sakila
## Tabelle: staff

Übung 1:

a) Gebt alle Datensätze aus in denen das Passwort NICHT GESETZT ist

b) Gebt alle Datensätze aus in denen das Passwort GESETZT ist
```

DATUM (DATETIME,STR_TO_DATE oder der Unix Timestamp)

Beispiel mit select by date

Example

```
USE sakila;
SELECT * FROM payment WHERE payment_date > '2005-05-24 22:00:01' and payment_date < '2005-05-25 22:00:01'
```

Übung:

```
Gebt alle Eintrage aus der Tabelle rental aus, bei der die Rückgabe zwischen
>= 2005-03-06 00:00:01   und <  2005-08-01
der letzte eintrag wäre 2005-07-31 23:59:59
return_date
```

Schlechtes Beispiel mit select by date

Example

```
## Es kann kein Index (Eintrag im Schlagwortverzeichnis verwendet werden)
## Alle Rückgabe im Jahr 2005
select YEAR(rental_date),return_date from rental where YEAR(rental_date) = 2005;

## Besser: Index kann verwendet werden.
SELECT YEAR(return_date),return_date
FROM rental
WHERE rental_date >= '2005-01-01 00:00:01'
AND rental_date <= '2005-12-31 23:59:59';
```

Example 2:

```
select YEAR(return_date),MONTH(return_date),DAY(return_date) from rental;
```

Exercise

```
Lasst euch alle Rückgaben aus Rental anzeigen, die zwischen (inkl) dem 27. und 30. zurückgegeben wurden
(egal welchen Monats)
HINT: DAY
```

Beispiel mit Unix Timestamp (Umwandeln)

Was ist das ?

- Das Datum wird dargestellt als Sekunden seit 01.01.1970

Warum ?

- Mit dem Unix Timestamp lassen sich einfache Vergleiche machen
- und Auswahlen treffen (z.B. Range - Abfrage)

Beispiele

```
SELECT unix_timestamp('2022-01-04');
-- Datum auslesen und in unix timestamp umwandeln
SELECT last_update,unix_timestamp(last_update) from actor;
-- Aktuelles Datum als unix timestamp (inkl Uhrzeit) - Systemzeit des Servers/Rechners
-- Rechner auf dem MySQL - Server läuft
select unix_timestamp();

-- Unix timestamp in einer Datum umwandeln
SELECT from_unixtime('1648202962');
```

Exercise

```
Frage aus rental alle Datensätze ab bei denen
o Das return_date nicht null (NOT NULL) ist
o Gib als einziges Feld das return_date umgewandelt zum Unix timestamp an.
o HINT: UNIX_TIMESTAMP()
```

Beispiel mit Dateformat

Example

```
SELECT date_format(last_update,'%M %Y') as meindatum FROM actor;
```

Exercise

```
Gib für rental_date aus der tabelle  rental das rental_date für jeden Datensatz wie folgt aus:
Tag-der-Woche Jahr-2-stellig

Weitere Spalten werden nicht benötigt.
```

- https://www.w3schools.com/sql/func_mysql_date_format.asp

Beispiel String in Date umwandeln

Beispiel:

```
## Das Ziel ist immer das bekannte Format
## YYYY-MM-DD, aber das macht selbständig unter Haube,
## Wenn ich ihm sage, wie mein Ursprungsformat aussieht

SELECT STR_TO_DATE("August 10 2017", "%M %d %Y");
```

Übung

```
Wandle das Datum in ein MYSQL-Datumsformat um mit STR_TO_DATE
12.04.2022
```

Referenz:

- https://www.w3schools.com/sql/func_mysql_str_to_date.asp

COUNT / STATISTIK

Zählen von Datensätzen

Example

```
SELECT count(*) as howmany FROM actor;
select count(*) from actor where last_name like 'D%';
```

Exercise

```
Wieviele Filme mit der film_id 15 gibt es im Inventar (über alle Stores hinweg)
in der tabelle inventory
```

Beispiel für Statistikabfrage (MAX,MIN)

Beispiel

```
SELECT MAX(replacement_cost) as am_teuersten,MIN(replacement_cost) FROM sakila.film;
```

Exercise

```
Db: sakila
Tabelle: film
Field/Column: rental_rate

Was sind die durchschnittlichen Leihgebühren für alle Film, die mit dem Titel "T" anfangen.
AVG() wird hier benötigt
```

NUMERISCHE FUNKTIONEN

Abrunden / Aufrunden / Kaufmännisch Runden

Example

```
-- Kaufmännisch runden
SELECT ROUND(1.56,1);
-- Abschneiden
SELECT truncate(1.56,1);
-- FLOOR() -> abrunden zum nächsten Integer
SELECT FLOOR(12.56);
-- CEIL() - Aufrunden zum nächsten Integer
SELECT CEIL(12.3);
```

Exercise

```
1. Übung

-- Db: sakila
-- Table: film
-- Field: length

Lass Dir die durchscnittliche Länge aller Filme, kaufmännisch gerundet auf volle Minuten anzeigen

HINT: AVG(),ROUND()


2. Übung

-- Db.: sakila
-- Table: film

Berechne die Summe aller replacement_cost(s) in film und runde dieser auf die nächste voller Zahl ohne Nachkommastellen auf.
HINT: SUM(),CEIL()
```

WEITERE STRINGFUNKTIONEN

Überblick der Stringfunktionen

- https://dev.mysql.com/doc/refman/8.0/en/string-functions.html

CHAR_LENGTH und LENGTH

Was ist der Unterschied ?

- CHAR_LENGTH - zeigt Anzahl der Zeichen an
- LENGTH - Anzahl der Bytes

Beispiel actor

```
select last_name, length(last_name),char_length(last_name) from actor;
```

Exercise

```
-- Db: sakila
-- Table: actor

Übung:

Bitte vorname und nachname als zusammengesetzten String aus (durch Leerzeichen getrennt),
z.B. PETERSON,DAN und gib als 2. Feld die Länge des Strings aus.
z.B. PETERSON,DAN 12

HINT: Verwende zum Zusammenkleben möglich CONCAT_WS
```

- https://www.w3schools.com/sql/func_mysql_concat_ws.asp

Replace - Zeichen ersetzen

Beispiel

```
SELECT REPLACE('Hallo Weltenbummler Weltenentdecker','Welt','Erd');
SELECT REPLACE('Jones','o','e');

select first_name,'=>' as ' wird zu : ', replace(first_name,'CHRISTIAN','PETER') from actor where first_name like 'C%';
```

Beispiel (Erweitert)

```
select first_name,'wird zu:',replace(first_name,'CHRISTIAN','PETER') as modified from actor where
replace(first_name,'CHRISTIAN','PETER') != first_name;

## performanter:
select first_name,'wird zu:',replace(first_name,'CHRISTIAN','PETER') as modified from actor where first_name = 'CHRISTIAN';
```

Übung:

```
Gebe all Adressen aus, bei denen die Adresse das Wort Drive enthält
(vorne, hinten, Mitte)

Zeige folgende Felder an:

Feld1: address_id,
Feld2: address,
Feld3: (modifizierte Adresse: ersetze Drive -> Weg) als neue_adresse ausgeben

Hint: replace()
```

Übung (hat nix mit dem Thema zu tun)

```
-- Db: sakila
-- table: film

-- Lese alle replacement_cost aus (Feld 1), (Feld 2:) multipliziere diese mit 2 ( * 2 ) und Ersetze das Decimaltrennzeichen "." ->
durch. ","
-- Beispiel

-- 2.99  5,98
```

Reference:

- https://www.w3schools.com/sql/func_mysql_replace.asp

ORDER BY (Sortierung)

Beispiel und Übung order by

Beispiel

```
SELECT first_name,last_name FROM actor ORDER BY last_name DESC, first_name;
```

Übung (schlechtes Beispiel, anderes wäre besser)

```
1. Wir sortieren alle Einträge film - Tabelle
2. rental_rate absteigend, Titel aufsteigend
```

LIMIT

Kombiniertes Beispiel mit Order By und Limit + Übung

What does it do ?

```
From your results of your query only shows a subset
```

Beispiel

```
## Eine Zahl: d.h. beginnend ab 0. Datensatz (Datensatz Numero 1) und 3 Ergebnisse
## Long version
SELECT * FROM actor ORDER BY last_name DESC LIMIT 0,3;
## Short version
```

```
SELECT * FROM actor ORDER BY last_name DESC LIMIT 3;

## Beginne mit Datensatz 3 und dann 10 Ergebnisse.
SELECT * FROM actor WHERE last_name like 'W%' ORDER BY last_name DESC, first_name LIMIT 2,10;
```

Übung

```
1. Zeige von den Filmen (sakila.film) die teuersten (rental_rate) zuerst, davon

Variante 1:
- Zeige die ersten 20

Variante 2: Zeige 10 ab dem 11. Film
```

GROUP BY

Example GROUP BY / HAVING

What does it do ?

```
Group By aggregates data (data sets)
Example: So if something appears multiple times, e.g a first name,
it will only appears once.
```

Syntax

```
SELECT .. FROM .. WHERE .. GROUP BY .. HAVING .. ORDER BY .. LIMIT
```

Example GROUP BY

```
SELECT last_name,COUNT(last_name) as cnt FROM actor GROUP BY last_name;
```

Exercise

```
-- DB: sakila
-- Table: film

-- Gruppiere alle filme nach rental_rate und gibt die jeweilige anzahl aus
-- d.h.
-- z.B.
-- Preis Anzahl
-- 4.99    336
-- 2.99    323
-- 0.99    341
```

What does HAVING do

```
Step 1:
First MySQL reads all the data based on WHERE

Step 2:
Then it aggregates the data (GROUP BY) and filters it by HAVING
```

Example Having (Simple)

- Simple: WHERE for GroupBy (because where does not work here)
- Example

```
SELECT last_name, COUNT(last_name)
FROM actor
GROUP BY last_name
HAVING count(last_name) > 2

## or use it with alias from the field
select last_name, count(last_name) as gezaehlt
from actor
group by last_name
having gezaehlt > 2;
```

Example Having (a bit more complicated)

```
## Step 1: Get all data with last_name like 'W%'
## Step 2: Group by last_name
## Step 3: Only return dataset where count(last_name) --> aggregated data (HAVING) = 2

select last_name,count(last_name) from actor where last_name like 'W%' group by last_name having count(last_name) = 2;
```

Exercise

```
Gruppiere alle actor
o mit nachname, und anzahl nachname (felder die wir ausgeben)
o Gebe nur die actor aus, deren nachname mit J anfängt
```

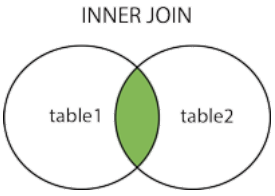
JOINS

Überblick über JOINS

What is a JOIN for ?

- combines rows from two or more tables
- based on a related column between them.

MySQL/MariaDB (Inner) Join



MySQL/MariaDB (Inner) Join (explained)

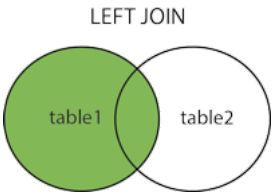
- Inner Join and Join are the same
- Returns records that have matching values in both tables
- Inner Join, Cross Join and Join
  - are the same in MySQL

In Detail: [INNER] JOIN

- Return rows when there
  - is a match in both tables
- Example

```
SELECT actor.first_name, actor.last_name, film.title
FROM film_actor
INNER JOIN actor ON film_actor.actor_id = actor.actor_id
INNER JOIN film ON film_actor.film_id = film.film_id;
```

MySQL/MariaDB Left Join



MySQL/MariaDB Left (outer) Join (explained)

- Return all records from the left table
- *AND* the matched records from the right table
- The result is NULL on the right side
  - if there are no matched columns on the right
- Left Join and Left Outer Join are the same

In Detail: Left Join

- Return all rows from the left side
  - even if there is not result on the right side
- Example

```
SELECT
c.customer_id,
c.first_name,
c.last_name,
a.actor_id,
a.first_name,
a.last_name
```

```
FROM customer c
LEFT JOIN actor a
ON c.last_name = a.last_name
ORDER BY c.last_name;
```

Example join - 2 tables - 1:1

Example Join über 2 Tabellen (1:1)

```
SELECT a.*,c.* FROM customer c JOIN address a ON c.address_id = a.address_id;
SELECT c.first_name,c.last_name,a.postal_code FROM customer c JOIN address a ON c.address_id = a.address_id;
-- Komplexe Alternative, ich würde davon abraten, weil zuviel Schreibarbeit
SELECT customer.first_name,customer.last_name,address.postal_code FROM customer JOIN address ON customer.address_id =
address.address_id;
```

Exercise über 2 Tabellen (1:1)

```
DB: sakila
Tables: store,address
Feld: address_id
Zeige alle Adressen der Stores an (Alle Felder von store und von address)
```

Beispiel Join über 3 Tabellen

Example

```
-- Schritt 1 - Erstmal JOIN über 2 Tabellen

select a.last_name,fa.film_id from actor a JOIN film_actor fa ON a.actor_id = fa.actor_id;

-- Schritt 2 (Schritt 1 erweitert)
-- Jetzt der JOIN über 3 Tabellen
SELECT a.last_name,fa.film_id,f.title
FROM actor a
JOIN film_actor fa
ON a.actor_id = fa.actor_id
JOIN film f
ON fa.film_id = f.film_id;
```

Exercise for Step 1 (Schritt 1)

```
Db: sakila
Tables: film, film_category

Zeigt alle category_id ' s aus der Tabelle film_category für den jeweiligen film an.
```

Exercise for Step 2 (Schritt 2)

```
-- Erweiterung von Schritt 1
Db: sakila
Tables: film, film_category, category

Erweitere die Abfrage aus Step 1:
o zeige jetzt auch noch mit Hilfe eines weiteren Joins die Kategorie (name) an.
```

Beispiel Left join

Example Left Join über 2 Tabellen (1:1)

```
SELECT a.*,c.last_name FROM address a LEFT JOIN customer c ON a.address_id = c.address_id;
SELECT a.*,c.last_name FROM address a LEFT JOIN customer c ON a.address_id = c.address_id WHERE c.last_name IS NULL;
```

Exercise über 2 Tabellen (1:1)

```
DB: sakila
Tables: address,store
Feld: address_id

Zeige all Adressen an (address) und bei den Adressen, wo es keinen Store gibt NULL
(LEFT JOIN)
```

Beispiel 2 Tabellen vergleichen

Zeige alle Zeilen an, bei dem einen Wert in einem Feld in beiden Tabellen vorkommt.

```
-- Wert soll jeweils im Feld last_name und first_name vorkommen
-- Vorbereitung (Simulation)
create table testtabelle as select * from actor where last_name like 'A%';
SELECT a.*,t.* FROM actor a JOIN testtabelle t ON a.last_name = t.last_name AND a.first_name = t.last_name;
```

Übung:

```
Lasse alle Kunden ausgeben, die auch gleichzeitig Schauspieler sind (anhand Vorname, Nachname)
Hint: actor, customer, first_name, last_name
```

Beispiel komplexer Join

Beispiel

```
select f.rental_rate,count(f.rental_rate)
from film f
join film_category fc
on f.film_id = fc.film_id
join category c
on fc.category_id = c.category_id
WHERE c.name ='Comedy'
AND f.rental_rate > 0.99
GROUP BY f.rental_rate
ORDER BY f.rental_rate DESC;
```

Übung

```
Gib alle Filme aus, die der category Family angehören
und deren rental_rate = 2.99 ist.
Gruppiere sie nach den replacement_cost.
Gibt folgende Felder aus:
   replacment_cost und die Anzahl replacement_cost
```

UPDATE

Update with example

Syntax

```
UPDATE tabellenname SET feld1=wert1,feld2=wert2 WHERE bedingung
z.B.
UPDATE actor SET first_name = 'ABC' where actor_id = 1;
```

Reference:

- https://www.w3schools.com/sql/sql_update.asp

DELETE

Delete with example

Example

```
DELETE FROM actor WHERE actor_id = 201;
```

Combined exercise

```
-- db: sakila
-- table: actor
--
-- 1. Neuen Datensatz mit last_name, first_name einfügen mit beliebigem Vor- und Nachnamen
-- (sollte id 201 sein)
-- 2. Ändern den neuen Datensatz -> neuer Vorname.
-- 3. Wir löschen den neuen Datensatz
```

Reference

- https://www.w3schools.com/sql/sql_delete.asp

ÜBUNGEN

Uebung - Berechnung aus Feld

Übung order by

Beispiel

```
SELECT first_name,last_name FROM actor ORDER BY last_name DESC, first_name;
```

Übung (schlechtes Beispiel, anderes wäre besser)

```
1. Wir sortieren alle Einträge film - Tabelle
2. rental_rate absteigend, Titel aufsteigend
```

Übung mit Order By und Limit

What does it do ?

```
From your results of your query only shows a subset
```

Beispiel

```
## Eine Zahl: d.h. beginnend ab 0. Datensatz (Datensatz Numero 1) und 3 Ergebnisse
## Long version
SELECT * FROM actor ORDER BY last_name DESC LIMIT 0,3;
## Short version
SELECT * FROM actor ORDER BY last_name DESC LIMIT 3;

## Beginne mit Datensatz 3 und dann 10 Ergebnisse.
SELECT * FROM actor WHERE last_name like 'W%' ORDER BY last_name DESC, first_name LIMIT 2,10;
```

Übung

```
1. Zeige von den Filmen (sakila.film) die teuersten (rental_rate) zuerst, davon

Variante 1:
- Zeige die ersten 20

Variante 2: Zeige 10 ab dem 11. Film
```

Übung mit Select where

Example (Simple)

```
SELECT * FROM actor WHERE last_name = 'AKROYD' or last_name = 'GABLE';
```

Example (mit Klammern)

```
SELECT * FROM actor WHERE (last_name = 'Akroyd' and first_name = 'Christian') or (last_name = 'Gable' and first_name =
'Christian');
```

Übung

```
* Alle Datensätze aus actor anzeigen bei denen der Nachname Akroyd oder der Vorname Christian ist.
```

Übung mit select where in

Example

SELECT * FROM actor WHERE first_name IN ('JOE','ED','JENNIFER');

Übung

```
* Sucht Euch 3 Citys und lasst die Datensätze aus der Tabelle city dazu ausgeben
* Frage: Welches Feld verwenden ?
* mit where ... IN umsetzen
```

Übung mit select und like

Example

```
SELECT * FROM actor WHERE last_name like 'D%';

-- Starts with D and ends with S
select * from actor where last_name like 'D%S';
```

Exercise

```
## Lass Euch alle actor ausgeben, deren Nachname auf N endet.
```

Übung 1+2 - Select Range

Beispiel 1

```
SELECT * FROM actor where actor_id > 100 and actor_id < 150;

## 10 und 50 inkludiert
select * from actor where actor_id >= 10 and actor_id <= 50;
```

Beispiel 2

```
SELECT * FROM actor where actor_id between 100 and 150;
```

Übung 1

```
- Gebe alle Adressen (address) aus, wo der postal_code > 10000 und postal_code < 20000 ist
oder < 90000 und > 50000
```

Übung 2

```
- Lasst euch Filme (film) anzeigen deren Wiederbeschaffungspreis (replacement_cost) zwischen 18.99 und 20.99 ist. (inkl)
```

Übung 3

```
Lass Euch alle Einträge aus dem inventory mit between im Bereich 10 bis 100 anzeigen (inkl 10 und 100) - inventory_id
```

Übung mit select und not in

Beispiel:

```
SELECT * FROM actor WHERE last_name NOT IN ('AKROYD','DAVIS');
```

Übung:

```
Lasst Euch alle Adressen (address) anzeigen, die nicht im District QLD oder Alberta sind
```

Übung mit select by date

Example

```
USE sakila;
SELECT * FROM payment WHERE payment_date > '2005-05-24 22:00:01' and payment_date < '2005-05-25 22:00:01'
```

Übung:

```
Gebt alle Eintrage aus der Tabelle rental aus, bei der die Rückgabe zwischen
>= 2005-03-06 00:00:01   und <  2005-08-01
der letzte eintrag wäre 2005-07-31 23:59:59
return_date
```

Übung mit select by day

Example

```
## Es kann kein Index (Eintrag im Schlagwortverzeichnis verwendet werden)
## Alle Rückgabe im Jahr 2005
select YEAR(rental_date),return_date from rental where YEAR(rental_date) = 2005;

## Besser: Index kann verwendet werden.
SELECT YEAR(return_date),return_date
FROM rental
WHERE rental_date >= '2005-01-01 00:00:01'
AND rental_date <= '2005-12-31 23:59:59';
```

Example 2:

```
select YEAR(return_date),MONTH(return_date),DAY(return_date) from rental;
```

Exercise

```
Lasst euch alle Rückgaben aus Rental anzeigen, die zwischen (inkl) dem 27. und 30. zurückgegeben wurden
(egal welchen Monats)
HINT: DAY
```

Übung - Zählen von Datensätzen

Example

```
SELECT count(*) as howmany FROM actor;
select count(*) from actor where last_name like 'D%';
```

Exercise

```
Wieviele Filme mit der film_id 15 gibt es im Inventar (über alle Stores hinweg)
in der tabelle inventory
```

Übung - Statistik AVG

Beispiel

```
SELECT MAX(replacement_cost) as am_teuersten,MIN(replacement_cost) FROM sakila.film;
```

Exercise

```
Db: sakila
Tabelle: film
Field/Column: rental_rate

Was sind die durchschnittlichen Leihgebühren für alle Film, die mit dem Titel "T" anfangen.
AVG() wird hier benötigt
```

Übung - NOT NULL / NULL

Example

```
SELECT * FROM rental WHERE return_date IS NOT NULL;
SELECT * FROM rental WHERE return_date IS NULL;
```

Exercise

```
## Database: sakila
## Tabelle: staff

Übung 1:

a) Gebt alle Datensätze aus in denen das Passwort NICHT GESETZT ist

b) Gebt alle Datensätze aus in denen das Passwort GESETZT ist
```

Übung mit select und SUBSTR,LOWER,UPPER,CONCAT

Example 1

```
## Zusammenkleben
SELECT concat(title,' ',release_year,' ',description) as listeneintrag FROM film;
SELECT concat ('hans und lotta',' ','haben glueck');

## Teilstring
SELECT substr('Zusammengesetztes Wort',1,2);

## Kleinschreibung
select lower('SONNENSCHEIN');
select lower(last_name) from actor;

## Grossschreibung
select upper('klein');

## Kombiniert
SELECT concat('Ausgabe: ',substr(description,1,20)) as listeneintrag FROM film;
SELECT upper(substr(description,1,20)) from film;
SELECT lower(substr(description,1,20)) from film;
```

Example 2

```
select 'test' ' ist gut ' 'verlaufen';
```

Exercise

```
-- Db: sakila
-- Tabelle: actor
Gebt folgende Felder aus (in einer Abfrage)

o Ausgabe Spalte 1: last_name gross geschrieben
o Ausgabe Spalte 2: first_name klein geschrieben
o Ausgabe Spalte 3: ausgabe mit zusammengeklebt last_name + ' ' + first_name (ohne +)
o Ausgabe Spalte 4: erste beiden Buchstaben vom Nachnamen.

HINT: -> UPPER, LOWER, CONCAT, SUBSTR
```

Documentation

- https://dev.mysql.com/doc/refman/8.0/en/string-functions.html
- https://www.w3schools.com/sql/func_mysql_concat.asp
- https://www.mysqltutorial.org/sql-concat-in-mysql.aspx

Übung mit unix timestamp

Was ist das ?

- Das Datum wird dargestellt als Sekunden seit 01.01.1970

Warum ?

- Mit dem Unix Timestamp lassen sich einfache Vergleiche machen
- und Auswahlen treffen (z.B. Range - Abfrage)

Beispiele

```
SELECT unix_timestamp('2022-01-04');
-- Datum auslesen und in unix timestamp umwandeln
SELECT last_update,unix_timestamp(last_update) from actor;
-- Aktuelles Datum als unix timestamp (inkl Uhrzeit) - Systemzeit des Servers/Rechners
-- Rechner auf dem MySQL - Server läuft
select unix_timestamp();

-- Unix timestamp in einer Datum umwandeln
SELECT from_unixtime('1648202962');
```

Exercise

```
Frage aus rental alle Datensätze ab bei denen
o Das return_date nicht null (NOT NULL) ist
o Gib als einziges Feld das return_date umgewandelt zum Unix timestamp an.
o HINT: UNIX_TIMESTAMP()
```

Übung mit Dateformat

Example

```
SELECT date_format(last_update,'%M %Y') as meindatum FROM actor;
```

Exercise

```
Gib für rental_date aus der tabelle  rental das rental_date für jeden Datensatz wie folgt aus:
Tag-der-Woche Jahr-2-stellig

Weitere Spalten werden nicht benötigt.
```

- https://www.w3schools.com/sql/func_mysql_date_format.asp

Übung join - 2 tables - 1:1

Example Join über 2 Tabellen (1:1)

```
SELECT a.*,c.* FROM customer c JOIN address a ON c.address_id = a.address_id;
SELECT c.first_name,c.last_name,a.postal_code FROM customer c JOIN address a ON c.address_id = a.address_id;
-- Komplexe Alternative, ich würde davon abraten, weil zuviel Schreibarbeit
SELECT customer.first_name,customer.last_name,address.postal_code FROM customer JOIN address ON customer.address_id =
address.address_id;
```

Exercise über 2 Tabellen (1:1)

```
DB: sakila
Tables: store,address
Feld: address_id
Zeige alle Adressen der Stores an (Alle Felder von store und von address)
```

Übung - Refresher Tag 2 - morgens - SELECT

Übung 1:

```
actor: Datensätze ausgeben, wo der Vorname mit C beginnt
und hier nur die ersten 10 sortiert nach Nachname
- nur die Felder Vorname, Nachname ausgeben
```

Übung 2:

```
addresses:
- Alle Adressen die im district California,Attika,Nantou sind
- Sortierung nach postal_code (aufsteigend)
- nur die ersten 10 davon anzeigen
```

Übung - Refresher Tag 3 - morgens - SELECT

Übung 1:

```
-- Datenbank: sakila
-- Table: payment
-- - Gebe den höchsten, den niedrigsten und den mittleren Zahlbetrag (payment) aus
-- für alle Zahlungen (payment_date) vor dem 01.07.2005 (2005-07-01)
-- RUNDE den mittleren Zahlbetrag kaufmännisch (ROUND) mit 2 Nachkommastellen.


-- Hinweis:
-- AVG, MIN, MAX, ROUND
-- und die Tabelle payment soll verwendet werden.
```

- https://www.w3schools.com/sql/func_mysql_round.asp

Übung 2:

```
Datenbank: sakila.
GEBE alle Sprachen aus, die nicht deutsch oder englisch sind.

Tabelle: language
```

Optional: Übung megajoin

```
-- Grosse übung
Gebt alle Verleihdatensätze (rental) aus und zwar
o mit dem zugehörigen Kunden (Vorname, Nachname)
o klebt beides zusammen (concat) -> als langname
o mit dem zugehörigen mitarbeiter (der hat es verliehn) -> nur den vornamen ausgaben als
mitarbeiter_vorname
o gebt den zugehörigen film title aus (inventory -> film_id)
o in welchem ort wurde der Film ausgeliehn ? -> inventory -> store -> address store_id -> address
```

Datenbanken und Tabellen anlegen, verändern und löschen, Daten einfügen

Datenbanken anlegen und löschen

Beispiel

```
## Datenbank anlegen
create schema training;
create database training;

## Datenbank löschen
drop schema training;
drop database training;
```

Übung.

```
a.) Erstelle die Datenbank testdaten und lösche sie danach wieder
b.) Lege die Datenbank training an. (wenn noch nicht vorhanden)
```

Übung - Tabelle kurs anlegen

Schritt 1:

```
Tabelle 'kurs' anlegen mit dem Feld

kurs_id
  o smallint oder smallint(10) - Auswahl in MySQL workbench
  o Haken bei PK (primary key - Primärschlüssel - nur 1x pro Tabelle möglich)
  o Haken bei NN (not null, kurs_id immer gesetzt sein muss)
  o Haken bei AI (Auto_increment - automatisches Hochzählen bei Anlage eines neuen Datensatzes)
  o Haken bei UN (Unsigned - d.h. nur positiver Wertbereich 0 bis ....)

## Warum smallint
## Wir nehmen an, dass wir nicht mehr als 65000 kurs haben, jeder Kurs hat
## eine eindeutige id, tinyint mit 0 bis 255 (unsigned wäre uns zu klein)

Dann kommt -> Apply

## Er zeigt mir dann das SQL-Statement an:
## Backticks macht er immer, brauchen wir aber nicht `
## nur benötigt wenn Leerzeichne im Tabellennamen -> ABER NICHT empfohlen.

CREATE TABLE `training`.`kurs` (
  `kurs_id` SMALLINT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`kurs_id`));

Dann -> 2. Apply


Dann -> Finish (Jetzt führt er es erst aus)
```

Schritt 2: Weitere Felder

```
titel - varchar(60) - NOT NULL (NN)
startdatum - DATE - NOT NULL (NN) - Default 2
tage - TINYINT - NOT NULL (NN)

ALTER TABLE kurs
ADD COLUMN `titel` VARCHAR(60) NOT NULL AFTER `kurs_id`,
ADD COLUMN `startdatum` DATE NOT NULL AFTER `titel`,
ADD COLUMN `tage` TINYINT NOT NULL DEFAULT 2 AFTER `startdatum`;
```

Übung - Daten in kurs einfügen

```
INSERT INTO kurs (titel,startdatum) VALUES ('HipHop Extended','2022-10-01');
INSERT INTO kurs (titel,startdatum) VALUES ('HipHop Extended','2022-10-01'),('HipHop Extended 2','2022-10-01'),('HipHop Extended
3','2022-10-01');
```

Übung - Daten updaten

Beispiel:

```
update kurs set startdatum='2023-07-01' where titel like 'Erste%';
```

Übung

```
1. Ändere den Datensatz mit der kurs_id 1 und ändere dort das startdatum.

2. Ändere einen Datensatz mit Hilfe des Titel und verändere dort die Tage
where titel =
```

Datentypen

Integer - Ganzzahlen

Overview

```
TINYINT
  unsigned: 0-255
  signed: -127 bis +128
```

```
SMALLINT
A small integer. The signed range is -32768 to 32767. The unsigned range is 0 to 65535.

MEDIUMINT
A medium-sized integer. The signed range is -8388608 to 8388607. The unsigned range is 0 to 16777215.

INT
4.3 Milliarden (unsigned)

BIGINT
2^64 - 1
```

Reference:

- https://dev.mysql.com/doc/refman/8.0/en/integer-types.html

Strings - varchar

Wie ?

```
varchar(n)

n von 0 bis 65535
```

Beispiel

```
Beispiel:

varchar(45) - 45 Zeichen
```

Text

- https://www.mysqltutorial.org/mysql-text/

Variablen

Beispiel mit Abfrage und User-Variable

```
## IN VARIABLE SCHREIBEN
## Geht aber nur, wenn wir genau ein Ergebnis haben
select count(*) into @howmany from actor where last_name like 'W%'
order by last_name desc, first_name;
-- select @howmany

select first_name,last_name,@howmany from actor where last_name like 'W%'
order by last_name desc, first_name;
```

CSV export/import

Export von csv-daten aus der Workbench

Walkthrough

```
1. Wir haben daten ausgegeben (z.B. select * from actor oder select last_name,first_name from actor where last_name like 'D%';)
2. Im Ausgabefenster oben rechts, export - button klicken
3. Speicherort festlegen
```

Exercise

```
Exportiere (aus workbench) im csv format die tabelle film mit folgenden Kriterien.
- Alle Filme die mit dem titel T anfangen.
- Ausgabe nur: film_id,title, rental_rate.
```

Import von csv-daten mit der Workbench

Prerequisites

- You must in table and have the data listed
- Eventually you need structure already

```
CREATE TABLE `film3` (
  `film_id` int unsigned NOT NULL AUTO_INCREMENT,
  `title` varchar(128) NOT NULL,
  `rental_rate` decimal(4,2) NOT NULL,
  PRIMARY KEY (`film_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Exercise

```
Erstelle eine neue Tabelle:
CREATE TABLE `film3` (
  `film_id` int unsigned NOT NULL AUTO_INCREMENT,
  `title` varchar(128) NOT NULL,
  `rental_rate` decimal(4,2) NOT NULL,
  PRIMARY KEY (`film_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci


Importiere die csv-daten aus der letzten Übung
```

Reference:

- https://www.mysqltutorial.org/import-csv-file-mysql-table/#:~:text=Importing%20CSV%20file%20using%20MySQL%20Workbench&text=Open%20table%20to%20which%20the%20data%20is%20loaded.&text=F

Import von csv-daten als SQL-Befehl in der Workbench

Example

```
LOAD DATA INFILE 'c:/tmp/discounts_2.csv'
INTO TABLE discounts
FIELDS TERMINATED BY ',' ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(title,@expired_date,amount)
SET expired_date = STR_TO_DATE(@expired_date, '%m/%d/%Y');
```

Adjusting to our example (MySQL 8, SQL executed in Workbench)

```
## Vorarbeiten, Tabelle erstellen
CREATE TABLE `mitarbeiter` (
  `mitarbeiter_id` int unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(45) NOT NULL,
  `vorname` varchar(45) NOT NULL,
  `geburtsdatum` date DEFAULT NULL,
  PRIMARY KEY (`mitarbeiter_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

Nachname, Vorname, Geburtsdatum
-- IGNORE 1 ROWS - simple does not use this first line
Mustermann,Max,12.03.1976

SHOW VARIABLES LIKE 'secure_file_priv';

-- path must be like so: c:/.... (so no backslashes)
-- geburtsdatum is a field that must exist in structure / table
LOAD DATA INFILE 'c:/ProgramData/MySQL/MySQL Server 8.0/Uploads/mysql - daten.csv'
INTO TABLE mitarbeiter
FIELDS TERMINATED BY ',' ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(name, vorname, @geburtsdatum)
SET geburtsdatum = STR_TO_DATE(@geburtsdatum, '%d.%m.%Y');
```

```
## Das funktioniert nicht: Error 1290
LOAD DATA INFILE 'C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\mysql - daten.csv'
INTO TABLE mitarbeiter
FIELDS TERMINATED BY ',' ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(name, vorname, @geburtsdatum)
SET geburtsdatum = STR_TO_DATE(@geburtsdatum, '%d.%m.%Y');
```

Reference:

- https://www.mysqltutorial.org/import-csv-file-mysql-table/
- https://www.w3schools.com/sql/func_mysql_str_to_date.asp

Dealing with charset in Excel (CSV)

Prerequisites:

```
It is important that the charset of your origin csv file fits
(best option sholud be utf8) - if you are able to define the charset for export
```

Specific for cyrillian language

- https://community.jaspersoft.com/wiki/how-import-csv-document-cyrillic-symbols-excel-and-correctly-render-them-program

Reference:

- https://www.ias.edu/itg/content/how-import-csv-file-uses-utf-8-character-encoding-0

TIPPS & TRICKS

Cheatsheet - Auf dem System zurechtfinden - 1. Sichtung

Zurechtfinden

```
-- Welche Datenbanken gibt es:
show databases;

-- Welche Tabelle gibt es in einer Datenbank
-- use <datenbankname>
use sakila;
show tables;

## Wie ist die Struktur eine Tabelle / welche Felder
-- describe <tabellenname> z.B. actor
describe actor;
## Welche Indizies gibt es in einer Tabelle ?
show indexes from actor;

## Alle procedures/functions einer Datenbank auslesen
select * from information_schema.routines where routine_schema = 'sakila';
```

Cheatsheet für Selects

SELECT .. FROM .. WHERE .. ORDER BY .. LIMIT

```
select * from actor;
select feld1,feld2,feld3 from actor;

-- Mit Feldalias für die Ausgabe
select feld1 as ueberschrift_ausgabe,feld as feldueberschrift from actor;
```

WHERE

```
-- WHERE feldname =/like
select * from actor where last_name = 'AKROYD';
select * from actor where last_name like 'A%';
select * from actor where last_name like 'A%' and first_name like 'C%';
select * from actor where (last_name = 'Akroyd' and first_name = 'Christian') or (last_name = 'Gable' and first_name =
'Christian');

-- WHERE feldname IN / NOT IN
SELECT * FROM actor WHERE first_name IN ('JOE','ED','JENNIFER');
SELECT * FROM actor WHERE last_name NOT IN ('AKROYD','JONES');

-- WHERE feldname < > <= >= BETWEEN

SELECT * FROM actor where actor_id > 100 and actor_id < 150;
select * from actor where actor_id >= 10 and actor_id <= 50;
SELECT * FROM actor where actor_id between 100 and 150;

-- WHERE feldname IS NULL / IS NOT NULL
SELECT * FROM rental WHERE return_date IS NOT NULL;
SELECT * FROM rental WHERE return_date IS NULL;

-- WHERE feldname <= '2022-01-02 10:00:01' (DATUM)
SELECT * FROM sakila.payment WHERE payment_date > '2005-05-24 22:00:01' and payment_date < '2005-05-25 22:00:01'
```

WHERE -> DATUMSABFRAGE (NUR BEI WENIG DATEN < 1.000.0000 als Richtwert)

```
-- möglichst im where keine Funktion für Feldname
select YEAR(return_date),return_date from rental where YEAR(return_date) <= 2005;
```

DATUMS-FUNKTIONEN

```
select YEAR(return_date),MONTH(return_date),DAY(return_date) from rental;
SELECT date_format(last_update,'%M %Y') as meindatum FROM actor;

-- Unix timestamp
SELECT unix_timestamp('2022-01-04');
```

```
SELECT last_update,unix_timestamp(last_update) from actor;
SELECT unix_timestamp(); -- aktuelles Datum als unix timestamp
```

STRING-FUNKTIONEN

```
SELECT concat(title,' ',release_year,' ',description) as listeneintrag FROM film;
SELECT upper(title) from film;
SELECT lower(title) from film;
SELECT lower(substr(description,1,20)) from film;
```

ZÄHLEN / STATISTIK

```
SELECT count(*) as howmany FROM actor;
select count(*) from actor where last_name like 'D%';

-- Höchste Kosten (MAX), Geringste Kosten (MIN) anzeigen
SELECT MAX(replacement_cost) as am_teuersten,MIN(replacement_cost) FROM sakila.film;
```

SORTIERUNG / LIMIT

```
-- WHERE/ORDER BY
select FELD from TABELLE WHERE BEDINGUNG ORDER BY FELD1,FELD2
select first_name,last_name from actor where last_name like 'A%' order by first_name,last_name

-- LIMIT
select FELD from tabelle limit 0,30; -- Ab Datensatz 1 der Ergebnismenge
select * from actor limit 30 -- Ab Datensatz

-- WHERE/ORDER BY/LIMIT
SELECT feld FROM tabelle WHERE bedingung ORDER BY FELD1,FELD2 LIMIT 0,10
SELECT feld FROM tabelle WHERE bedingung ORDER BY FELD1,FELD2 LIMIT 5,10 -- Ab dem 6. Datensatz der Ergebnismenge
```

RECHNEN / RUNDEN

```
SELECT amount,1 as 'Preiserhoehung um',amount + 1 as preiserhoehung FROM payment;

-- Kaufmännisch runden
SELECT ROUND(1.56,1);
-- Abschneiden
SELECT TRUNCATE(1.56,1);
-- FLOOR() -> abrunden zum nächsten Integer
SELECT FLOOR(12.56);
-- CEIL() - Aufrunden zum nächsten Integer
SELECT CEIL(12.3);
```

GROUP (GRUPPIEREN) - SELECT .. FROM .. WHERE .. GROUP BY .. HAVING .. ORDER BY .. LIMIT

```
SELECT last_name,COUNT(last_name) as cnt FROM actor GROUP BY last_name;

-- Nachnamen Gruppieren und alle Nachnamen anzeigen die mehr als 2 Mal vorkommen
SELECT last_name, COUNT(last_name)
FROM sakila.actor
GROUP BY last_name
HAVING count(last_name) > 2
```

JOINS - SELECT .. FROM .. JOIN ... ON ... JOIN ... ON ... WHERE .. GROUP BY .. HAVING .. ORDER BY .. LIMIT

```
-- JOIN über 2 Tabellen
SELECT a.*,c.* FROM customer c JOIN address a ON c.address_id = a.address_id;
SELECT c.first_name,c.last_name,a.postal_code FROM customer c JOIN address a ON c.address_id = a.address_id;

-- JOIN über 3 Tabellen

-- Schritt 1: 2 Tabellen
SELECT a.last_name,fa.film_id FROM actor a JOIN film_actor fa ON a.actor_id = fa.actor_id;

-- Schritt 2: auf 3 Tabellen erweitern
SELECT a.last_name,fa.film_id,f.title
FROM actor a
JOIN film_actor fa
ON a.actor_id = fa.actor_id
JOIN film f
ON fa.film_id = f.film_id;
```

Restore auf der Kommandozeile (counterpart zu backup)

```
## Schritt 1: cmd.exe
## in cmd
## cd <in das bin-verzeichnis von mysql>
mysql -uroot -ppassword testdaten < C:\Users\Admin\Documents\dumps\Dump20220408.sql
```

MYSQL-Workbench disable safe-mode

```
You also can disable safe mode in MySQL Workbench, go to Edit -> Preferences -> SQL Editor, and uncheck "Safe Updates" check box

After that reconnect.


Referenz:
https://www.geeksengine.com/database/manage-table/safe-
update.php#:~:text=You%20also%20can%20disable%20safe,%22Safe%20Updates%22%20check%20box.
```

Datenbank mit mysql workbench exportieren

```
Menu -> Server -> Data Export

Datenbank auswählen.
Objects to export -> Alles auswählen (ist sicherer)
Export -> Export to self-contained file (besser transportabel, wenn die komplette datenbank wieder eingespielt)
Haken bei Include Create Schema

-> Start export


Sicherung liegt in Documents/dumps  -< mit Zeitstempel und Endung .sql
```

Datenbank mit mysql workbench importieren

```
Menu -> Server -> Data Import

Import from self-contained file
File auswählen (sql-file)

Start Import
```

Information schema in der Workbench aktivieren

```
Edit -> Preferences -> SQL Editor and then check the box "Show Metadata and Internal Schemas"

## Optional (Views & Triggers)

### Views


### Einfaches Beispiel
```

use sakila; CREATE VIEW actor_vorname AS SELECT first_name AS vorname FROM actor; select * from actor_vorname; select * from actor_vorname where vorname like 'A%';

```
### Über View updaten wenn Updatable
```

use sakila; UPDATE actor_vorname SET vorname='BRANGELINA' WHERE vorname = 'ANGELINA';

```
### Neue Felder, sind NICHT automatisch im View:
```

alter table actor add column city varchar(45) default 'Hildesheim';

city is not nicht im view

select * from actor_b;

erst wenn ich das create oder alter statement ausführen

alter view actor_b as select * from actor where last_name like 'B%'; select * from actor_b;

```
### Übung
```

Ausgehend von der Tabelle: film

A. Erstelle ein view, das nur den title und die 2 ersten Buchstaben des Titels anzeigt, mit dem Name film_short.

Nenne die felder: title_lang und title_short

B. Gibt mit hilfe des title_lang - feldes Titel aus die mit "A%" beginnen

```
### Triggers


### Create the structure
```

create table countries (country_id int auto_increment, name varchar(50) not null, primary key(country_id));

INSERT INTO countries (name) values ('Germany'), ('Austria');

create table country_stats(country_id int, year int, population int, primary key (country_id, year), foreign key(country_id) references countries(country_id));

INSERT INTO country_stats (country_id, year, population) values (1,2020,100000);

create table population_logs(log_id int auto_increment, country_id int not null, year int not null, old_population int not null, new_population int not null, updated_at timestamp default current_timestamp, primary key(log_id));

```
### Create the trigger
```

create trigger before_country_stats_update before update on country_stats for each row insert into population_logs(country_id, year, old_population, new_population) values(old.country_id, old.year, old.population, new.population);

```
### Create trigger (the same) but with BEGIN/END - Block
```

delimiter // create trigger before_country_stats_update before update on country_stats for each row

```
BEGIN
SET @anfang = 1;
insert into population_logs(
    country_id,
    year,
    old_population,
    new_population
)
values(
    old.country_id,
    old.year,
    old.population,
    new.population
);
END//
```

```
### Run a test
```

update country_stats set population = 1352617399 where country_id = 1 and year = 2020;

-- what's the new result

select * from population_logs;

```


### Exercise
```

-- Database: training CREATE DATABASE IF NOT EXISTS training; use training;

CREATE TABLE animals (id mediumint(9) NOT NULL AUTO_INCREMENT, name char(30) NOT NULL, PRIMARY KEY (`id`));

CREATE TABLE animal_count (animals int);

INSERT INTO animal_count (animals) VALUES(0);

CREATE TRIGGER increment_animal AFTER INSERT ON animals FOR EACH ROW UPDATE animal_count SET animal_count.animals = animal_count.animals+1;

-- Jetzt testen SELECT * FROM animal_count; INSERT INTO animals (name) VALUES('aardvark'); INSERT INTO animals (name) VALUES('baboon'); SELECT * FROM animal_count;

-- Reference: -- https://mariadb.com/kb/en/trigger-overview/

```
## Documentation
```