Befehle CheatSheet:

https://www.sqltutorial.org/wp-content/uploads/2016/04/SQL-cheat-sheet.pdf

Allgemeiner Hinweis:
Sollte versucht werden einen Namen zu wählen, der einem reservierten Keyword entspricht (Spalte “Alter” im zweiten Beispiel), muss der Name in Backquotes ` (Shift + Taste rechts neben Fragezeichen) eingekapselt sein! (Gilt nicht nur beim Erstellen, sondern auch Einfügen, Selektieren, etc.)

Konkrete Beispiele:

Beispiel 1: Erstellung einer Datenbank:

```
CREATE DATABASE neue_datenbank
```

Beispiel 2: Erstellung einer Tabelle (**ohne** Primärschlüssel, 4 Spalten mit den wichtigsten Typen):

```
CREATE TABLE neue_tabelle (Name VARCHAR(50),
                           Beschreibung TEXT,
                           Geburtstag DATE,
                           `Alter` INT)
```

Hinweis: Bei VARCHAR (variable characters) muss in Klammern die Angabe zur Maximallänge erfolgen

Beispiel 3: Erstellung einer Tabelle (**mit** Primärschlüssel, selbst erhöhend):

```
CREATE TABLE neue_tabelle2 (ID INT AUTO_INCREMENT PRIMARY KEY,
                            Name VARCHAR(50))
```

Andere Variante: Primärschlüssel nach Angabe der Spalten festlegen

```
CREATE TABLE neue_tabelle2 (ID INT AUTO_INCREMENT,
                            Name VARCHAR(50),
                            PRIMARY KEY (id))
```

Selbst erhöhend: Entsprechende Spalte (hier ID) wird bei neuen Einträgen immer um 1 erhöht (vom letzten Eintrag), wenn nicht explizit eine ID angegeben wird. Sollte beim Primary Key eigentlich immer drin sein.

Beispiel 4: Einfügen eines Eintrags in eine Tabelle (basierend auf Beispiel 3):

```
INSERT INTO neue_tabelle2 (ID, Name)
VALUES (999, "Peter")
```

Hinweis: Da ID automatisch erhöht wird, kann ID (und Wert 999) auch weggelassen werden. Wert wird dann letzter vorheriger Eintrag + 1.

Beispiel 5: Einfügen mehrere Einträge in eine Tabelle (basierend auf Beispiel 2):

```
INSERT INTO neue_tabelle (Name, Beschreibung, Geburtstag, `Alter`)
VALUES ("Peter", "Ist lustig", '1970-01-01', 51),
("Peter", "Ist nicht mehr so lustig", '1980-08-08', 41),
("Hans", "Peter", '1990-09-09', 31)
```

Hinweis: Datumsangaben in einzelne Anführungszeichen ‘ schreiben. Format kann abweichen (z.B. YYYY/MM/DD, YY-MM-DD, YYYYMMDD…)
Siehe dazu: https://mariadb.com/kb/en/date-and-time-literals/

Beispiel 6: Auswählen aller Infos aller Einträge (basierend auf Beispiel 5):

```
SELECT * FROM neue_tabelle
```

Name	Beschreibung	Geburtstag	Alter
Peter	Ist lustig	1970-01-01	51
Peter	Ist nicht mehr so lustig	1980-08-08	41
Hans	Peter	1990-09-09	31

Beispiel 7: Abfrage aller Infos aller Einträge, sortiert nach Alter, von jung bis alt (basierend auf Beispiel 5):

```
SELECT * FROM neue_tabelle
ORDER BY `Alter` ASC
```

ASC / DESC =
Ascending / Descending

Name	Beschreibung	Geburtstag	Alter ▲ 1
Hans	Peter	1990-09-09	31
Peter	Ist nicht mehr so lustig	1980-08-08	41
Peter	Ist lustig	1970-01-01	51

Beispiel 8: Abfrage bestimmter Infos aller Einträge mit Einschränkung Alter > 40 (basierend auf Beispiel 5):

```
SELECT Name, Beschreibung FROM neue_tabelle
WHERE `Alter` > 40
```

Name	Beschreibung
Peter	Ist lustig
Peter	Ist nicht mehr so lustig

Dinge mit Sekundärschlüssel:

Beispiel 9: Erstellung mehrerer Tabellen (können mit Semikolon ; getrennt hintereinander als “ein Befehl” eingegeben werden), dabei referenzieren “Projekte” und “Teilnahme” die vorherigen Tabellen als Sekundärschlüssel):

```
CREATE TABLE Entwickler (Personalnummer INT AUTO_INCREMENT PRIMARY KEY,
                         Vorname VARCHAR(50),
                         Nachname VARCHAR(50));

CREATE TABLE Führungskräfte (Personalnummer INT AUTO_INCREMENT PRIMARY
KEY,
                             Vorname VARCHAR(50),
                             Nachname VARCHAR(50));

CREATE TABLE Fachgebiete (Skill_ID INT AUTO_INCREMENT PRIMARY KEY,
                          Bezeichnung VARCHAR(100));

CREATE TABLE Projekte (Projekt_ID INT AUTO_INCREMENT PRIMARY KEY,
                       Skill_ID INT,
                       Personalnummer INT,
                       Foreign Key (Skill_ID) REFERENCES Fachgebiete(Skill_ID),
                       Foreign Key (Personalnummer) REFERENCES
                                                  Führungskräfte(Personalnummer));

CREATE TABLE Teilnahme (Projekt_ID INT,
                        Personalnummer INT,
                        Foreign Key (Projekt_ID) REFERENCES Projekte(Projekt_ID),
                        Foreign Key (Personalnummer) REFERENCES
                                                  Entwickler(Personalnummer))
```

Füllen mit Einträgen für weitere Beispiele:

```
INSERT INTO Entwickler (Personalnummer, Vorname, Nachname)
VALUES (1001234, "Klaus", "Abel"),
       (1001235, "John", "Smith"),
       (1001236, "Rebecca", "White");

INSERT INTO Führungskräfte (Personalnummer, Vorname, Nachname)
VALUES (19001, "Tom", "Cook"),
       (18002, "Pit", "Gates"),
       (17003, "Marija", "Musk");

INSERT INTO Fachgebiete (Skill_ID, Bezeichnung)
VALUES (1, "Data Science"),
       (2, "Kryptographie"),
       (3, "AI"),
       (4, "Robots");

INSERT INTO Projekte (Projekt_ID, Skill_ID, Personalnummer)
VALUES (1, 2, 19001),
       (2, 3, 19001),
       (3, 1, 17003);

INSERT INTO Teilnahme (Projekt_ID, Personalnummer)
VALUES (1, 1001234),
       (2, 1001235),
       (2, 1001234),
       (3, 1001236)
```

Entwickler

Personalnummer	Vorname	Nachname
1001234	Klaus	Abel
1001235	John	Smith
1001236	Rebecca	White

Führungskräfte

Personalnummer	Vorname	Nachname
17003	Marija	Musk
18002	Pit	Gates
19001	Tom	Cook

Fachgebiete

Skill_ID	Bezeichnung
1	Data Science
2	Kryptographie
3	AI
4	Robots

Projekte

Projekt_ID	Skill_ID	Personalnummer
1	2	19001
2	3	19001
3	1	17003

Teilnahme

Projekt_ID	Personalnummer
1	1001234
2	1001235
2	1001234
3	1001236

Beispiel 10: Abfrage aller Projekte (aufsteigend anhand Projekt_ID) mit Vor- & Nachname der zugehörigen Führungskraft (basierend auf Beispiel 9):

```
SELECT p.Projekt_ID, f.Vorname, f.Nachname FROM Führungskräfte as f
JOIN Projekte as p ON f.Personalnummer = p.Personalnummer
ORDER BY Projekt_ID ASC
```

Hier werden die Tabellen Projekte und Führungskräfte miteinander verbunden (JOIN). Die Personalnummer beider Tabellen werden dabei gleichgesetzt.

Anmerkung: Tabelle Projekte wird als “p” und Führungskräfte als “f” benannt. Mit p.Spaltenname können dann die Spalten angesprochen werden. Dies ist nötig, wenn Spalten gleich heißen (siehe Zeile 2), im SELECT wäre dies nicht nötig, aber schadet auch nicht.

Projekt_ID ▲ 1	Vorname	Nachname
1	Tom	Cook
2	Tom	Cook
3	Marija	Musk

Beispiel 11: Abfrage aller Namen der Entwickler und Bezeichnung der Fachgebiete, in denen diese tätig sind (basierend auf Beispiel 9):

```
SELECT e.Vorname, e.Nachname, f.Bezeichnung FROM Entwickler as e
JOIN Teilnahme as t ON e.Personalnummer = t.Personalnummer
JOIN Projekte as p ON p.Projekt_ID = t.Projekt_ID
JOIN Fachgebiete as f ON f.Skill_ID = p.Skill_ID
```

Hier werden bis auf Führungskräfte alle Tabellen miteinander verbunden.
In “Entwickler” stehen die benötigten Namen + Personalnummer.
In “Teilnahme” stehen die Personalnummern der Entwickler und Projekt_IDs.
In “Projekte” stehen die Projekt_IDs und Fachgebiete (Skill_IDs).
In “Fachgebiete” steht dann die Bezeichnung der Skill_IDs.

Vorname	Nachname	Bezeichnung
Klaus	Abel	Kryptographie
Klaus	Abel	AI
John	Smith	AI
Rebecca	White	Data Science