SQL cheat sheet · MariaDB

Basics · Database · Tables · Changing data · Help

Holger Sebastião · www.sebastiao.org
Version 1.0 · 2026-10-25

Terms — relational database

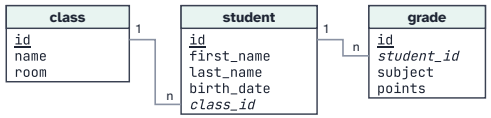
Table student

id	first_name	last_name	birth_date	class_id
1	Ali	Yilmaz	2008-03-14	1
2	Lea	Schmitt	2007-11-02	2
3	Tom	Becker	2008-06-30	1

primary key — *yellow: one row = one record, each column = one attribute (field)* — *foreign key → class.id*

Primary key	makes every row unique, never duplicated, never empty
Foreign key	refers to the primary key of another table
DBMS	the program that manages the database: here MariaDB

Our sample database — school



underlined = primary key, *italic* = foreign key. A class has many students, a student has many grades.

Writing rules

- Every statement ends with `;`
- Keywords in capitals (`SELECT`), names in lower case: easier to read.
- Text and dates in single quotes: `'Ali'`, `'2008-03-14'`
- Comment: `-- to end of line` (space after `--`)
- Names without special characters or spaces: `student`, `class_id`

Where do I write SQL?

- **phpMyAdmin** (e.g. with XAMPP): choose the database → **SQL** tab → enter the statement → Go.
- **Command line:** log in with `mariadb -u root -p` (with XAMPP: `mysql -u root -p`), quit with `exit`.

Creating a database

```
CREATE DATABASE school
  CHARACTER SET utf8mb4
  COLLATE utf8mb4_unicode_ci;
USE school;   -- work in school from now on
```

`SHOW DATABASES;`	all databases
`SHOW TABLES;`	all tables in the database
`DESCRIBE student;`	columns and data types
`DROP DATABASE school;`	deletes **everything**, without asking

`utf8mb4` stores every character (é, ü, emojis). `unicode_ci` sorts by Unicode rules, `ci`: case doesn't matter.

Data types

TYPE	FOR · EXAMPLE
`INT`	whole number: `42`
`DECIMAL(6,2)`	money, 6 digits, 2 after the point: `1234.50`
`DOUBLE`	decimal number (measurement): `3.14`
`VARCHAR(50)`	text up to 50 characters: `'Ali'`
`CHAR(5)`	text, always exactly 5: `'63450'`
`TEXT`	long text
`DATE`	date: `'2026-10-25'`
`TIME`	time: `'07:45:00'`
`DATETIME`	date and time: `'2026-10-25 07:45:00'`
`BOOLEAN`	true/false, stored as 1/0: `TRUE`

Decimals with a **point**: `3.5`. Dates always `'YYYY-MM-DD'`, not `'10/25/2026'`.

Postcodes and phone numbers are text, not numbers: otherwise a leading 0 disappears.

NULL means: no value

`NULL` is **not** `0` and not `''` (empty text), but "unknown". Fields without `NOT NULL` may stay empty. Test with `IS NULL` (page 2).

Creating a table — CREATE TABLE

```
CREATE TABLE class (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(10) NOT NULL UNIQUE,
  room VARCHAR(10)
);

CREATE TABLE student (
  id INT PRIMARY KEY AUTO_INCREMENT,
  first_name VARCHAR(50) NOT NULL,
  last_name VARCHAR(50) NOT NULL,
  birth_date DATE,
  class_id INT,
  FOREIGN KEY (class_id)
    REFERENCES class(id)
);
```

`AUTO_INCREMENT`	MariaDB counts the id up itself: 1, 2, 3 …
`NOT NULL`	required field, must not be empty
`UNIQUE`	no value twice
`DEFAULT 0`	default if nothing is given
`FOREIGN KEY`	foreign key: only ids that exist in `class`

Order: create `class` first, then `student`.

Changing and deleting tables

```
-- new column
ALTER TABLE student ADD email VARCHAR(100);
-- change the data type
ALTER TABLE student MODIFY email TEXT;
-- delete a column
ALTER TABLE student DROP COLUMN email;
-- delete the table including data
DROP TABLE grade;
```

Implementing relationships

1:n foreign key on the n side. **n:m** junction table:

```
CREATE TABLE membership (
  student_id INT,
  club_id INT,
  PRIMARY KEY (student_id, club_id),
  FOREIGN KEY (student_id)
    REFERENCES student(id),
  FOREIGN KEY (club_id) REFERENCES club(id)
);
```

Inserting data — INSERT

```
INSERT INTO class (name, room)
VALUES ('11IT1', 'B204'),
       ('12IT1', 'B206');

INSERT INTO student (first_name, last_name,
  birth_date, class_id)
VALUES ('Ali', 'Yilmaz', '2008-03-14', 1);
```

Leave out the `id`: `AUTO_INCREMENT` assigns it. Columns and values in the **same order**.

Changing data — UPDATE

```
UPDATE student
SET class_id = 2, last_name = 'Kaya'
WHERE id = 1;
```

Deleting data — DELETE

```
DELETE FROM student
WHERE id = 3;
```

Without `WHERE`, `UPDATE` changes and `DELETE` deletes **all** rows. There is no undo. Tip: first check with `SELECT … WHERE …` which rows are affected.

`DELETE FROM t;`	all rows gone, table stays
`TRUNCATE t;`	as above, id starts at 1 again
`DROP TABLE t;`	table and structure gone

Typical error messages

`1064`	typo in the statement: check comma, bracket, `;` or `'`. The message shows the place **after** the mistake.
`1452`	foreign key points to an id that doesn't exist
`1451`	row is still needed by another table: delete there first
`1062`	duplicate value in `PRIMARY KEY` or `UNIQUE`
`1146`	table doesn't exist: check the name or `USE`

SQL cheat sheet · MariaDB

Queries · Filtering · Aggregating · Joining · Help

Holger Sebastião · www.sebastiao.org
Version 1.0 · 2026-10-25

Structure of a query — SELECT

```
SELECT   columns          -- what?
FROM     table            -- from where?
WHERE    condition        -- which rows?
GROUP BY column           -- form groups
HAVING   condition        -- which groups?
ORDER BY column           -- sort
LIMIT    count;           -- how many?
```

Only `SELECT` and `FROM` are required. The **order** is fixed. Memory aid: **S**ome **F**rench **W**aiters **G**row **H**ealthy **O**range **L**emons.

Choosing columns

```
SELECT * FROM student;      -- all columns

SELECT first_name, last_name
FROM student;

SELECT DISTINCT class_id    -- no duplicates
FROM student;

SELECT last_name AS Name    -- alias
FROM student;
```

Sorting and limiting

```
SELECT first_name, last_name
FROM student
ORDER BY last_name ASC, first_name ASC
LIMIT 10;
```

`ASC` ascending (A–Z, 1–9, default), `DESC` descending. If two rows are equal, the second column decides. `LIMIT 10`: only the first 10 rows.

Useful functions

`CONCAT(first_name, ' ', last_name)`	join text
`UPPER(name)` · `LOWER`	upper · lower case
`LENGTH(name)`	length
`CURDATE()` · `NOW()`	today · now
`YEAR(birth_date)`	only the year

```
SELECT first_name,
  TIMESTAMPDIFF(YEAR, birth_date,
                CURDATE()) AS years
FROM student;     -- age in years
```

Filtering rows — WHERE

```
SELECT first_name, last_name
FROM student
WHERE class_id = 1
  AND birth_date >= '2008-01-01';
```

`= <>`	equal, not equal (also `!=`)
`< > <= >=`	less, greater …
`AND`	both conditions must be true
`OR`	at least one must be true
`NOT`	reverses
`BETWEEN 10 AND 15`	from 10 to 15, both limits **included**
`IN ('Maths', 'English')`	one from the list
`IS NULL`	field is empty
`IS NOT NULL`	field is filled

! `= NULL` **never** finds anything. Always `IS NULL`.

AND before OR: use brackets when mixing conditions:

```
WHERE (subject = 'Maths'
       OR subject = 'English')
  AND points < 5
```

Searching text — LIKE

```
SELECT * FROM student
WHERE last_name LIKE 'Sm%';
```

`%`	any number of characters, even none
`_`	exactly one character
`'Sm%'`	starts with Sm: Smith, Smart
`'%son'`	ends in son: Johnson
`'%e%'`	contains an e somewhere
`'J_nes'`	Jones, Janes, not Joanes

Calculating in a query

```
SELECT subject, points,
  ROUND((17 - points) / 3, 1) AS school_grade
FROM grade;
```

Per row with `+ - * /`. The table itself stays the same.

Counting and calculating

```
SELECT COUNT(*) AS number
FROM student
WHERE class_id = 1;
```

`COUNT(*)`	number of rows
`COUNT(column)`	number of filled fields (without NULL)
`SUM(points)`	sum
`AVG(points)`	average
`MIN(points)`	smallest value
`MAX(points)`	largest value
`ROUND(AVG(points), 1)`	round to 1 decimal place

Forming groups — GROUP BY

```
SELECT subject, ROUND(AVG(points), 1) AS avg
FROM grade
GROUP BY subject
HAVING AVG(points) >= 8
ORDER BY avg DESC;
```

SUBJECT	AVG
Computer science	11.4
English	9.2
German	8.0

`GROUP BY subject`: one result row **per subject**, the function calculates per group.

WHERE or HAVING? `WHERE` filters rows **before** grouping, `HAVING` filters groups **afterwards**. `COUNT`, `AVG` … are only allowed in `HAVING`.

! Every column in the `SELECT` must be in the `GROUP BY` or inside a function. Otherwise MariaDB reports no error but shows a random value.

Subquery

```
SELECT first_name, last_name
FROM student
WHERE id IN (SELECT student_id
             FROM grade
             WHERE points = 15);
```

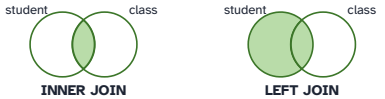
The inner query runs first and returns a list of ids.

Joining tables — JOIN

```
SELECT student.first_name, class.name
FROM student
INNER JOIN class
  ON student.class_id = class.id;
```

FIRST_NAME	NAME
Ali	11IT1
Lea	12IT1
Tom	11IT1

After `ON` it is always: **foreign key = primary key**. `table.column` says which table the column comes from.



INNER JOIN: only students who have a class.
LEFT JOIN: **all** students; without a class, `name` is `NULL`.

Three tables

```
SELECT class.name, student.last_name,
       grade.subject, grade.points
FROM grade
INNER JOIN student
  ON grade.student_id = student.id
INNER JOIN class
  ON student.class_id = class.id
WHERE class.name = '11IT1'
ORDER BY student.last_name;
```

Typical mistakes

- Comma before `FROM`: `SELECT first_name, FROM`
- Text without quotes: `WHERE subject = Maths`
- `JOIN` without `ON`: every row combined with every other.
- Column called `id` in two tables: write `table.id`.
- Reserved words as names, e.g. `order`, `key`, `group`.