SkillNewton

MySQL Cheat Sheet

MySQL commands on one page: the CLI, tables and data types, CRUD, joins, GROUP BY, indexes, users and grants, backups and transactions.

> This MySQL cheat sheet gathers the commands you use to create, query, secure and back up a database, from the `mysql` command-line client to `mysqldump`. It targets MySQL 8 and also works on MariaDB except where a row says otherwise. It suits students building their first PHP and MySQL project and developers who manage a small production database. Each row is a statement you can run as it is, after changing the example names.

How to use it

1. **Connect.** Open the client with `mysql -u root -p`, or use the SQL tab in phpMyAdmin, MySQL Workbench or a similar tool.
2. **Create a practice database.** Run the CREATE DATABASE and CREATE TABLE examples first so the other queries have something to work on.
3. **Practise safely.** Before any UPDATE or DELETE, run the same WHERE clause as a SELECT, and take a mysqldump backup before changing a live database.
4. **Check performance.** Put `EXPLAIN` in front of slow queries and add indexes on the columns used in WHERE, JOIN and ORDER BY.

Connecting and the mysql client

`mysql -u root -p`	Log in as root; you are prompted for the password.
`mysql -h db.example.com -P 3306 -u app -p shop`	Connect to a remote server, port and database.
`SELECT VERSION();`	Shows the server version (MariaDB versions include "MariaDB").
`SHOW DATABASES;`	Lists databases you can see.
`USE shop;`	Switches to a database.
`SHOW TABLES;`	Lists tables in the current database.
`DESCRIBE products;`	Columns, types, keys and defaults.
`SHOW CREATE TABLE products\G`	Full CREATE statement; `\G` prints the result vertically.
`SOURCE /path/to/file.sql;`	Runs the statements in a file.
`exit (or \q)`	Leaves the client.

Databases, tables and constraints

`CREATE DATABASE shop CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;`	New database with full Unicode support (including emoji).
`DROP DATABASE shop;`	Deletes the database and every table in it. No undo.
`id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY`	Auto-numbered primary key.

`email VARCHAR(255) NOT NULL UNIQUE`	Required and no duplicates.
`created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP`	Filled in automatically on insert.
`CHECK (price >= 0)`	Rejects invalid values (enforced in MySQL 8.0.16+ and MariaDB 10.2+).
`FOREIGN KEY (customer_id) REFERENCES customers(id) ON DELETE CASCADE`	Links tables; deleting a customer deletes their orders. Use `RESTRICT` to block instead.
`ALTER TABLE products ADD COLUMN stock INT NOT NULL DEFAULT 0;`	Adds a column.
`ALTER TABLE products MODIFY name VARCHAR(200) NOT NULL;`	Changes a column definition.
`ALTER TABLE products RENAME COLUMN name TO title;`	Renames a column (MySQL 8.0+, MariaDB 10.5+).
`TRUNCATE TABLE logs;`	Empties a table quickly and resets AUTO_INCREMENT.
`DROP TABLE IF EXISTS temp_import;`	Deletes a table if it exists.

```
CREATE TABLE customers (
  id         INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  name       VARCHAR(100) NOT NULL,
  email      VARCHAR(255) NOT NULL UNIQUE,
  created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE orders (
  id          INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  customer_id INT UNSIGNED NOT NULL,
  total       DECIMAL(10,2) NOT NULL CHECK (total >= 0),
  status      ENUM('pending','paid','cancelled') NOT NULL DEFAULT 'pending',
  ordered_at  DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (customer_id) REFERENCES customers(id) ON DELETE RESTRICT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Data types

`TINYINT, SMALLINT, INT, BIGINT`	Whole numbers of increasing range. Add `UNSIGNED` for non-negative values.
`DECIMAL(10,2)`	Exact numbers, such as money. Never use FLOAT for prices.
`FLOAT, DOUBLE`	Approximate numbers for measurements.
`BOOLEAN`	Alias for TINYINT(1): 0 is false, 1 is true.
`CHAR(2)`	Fixed length, for codes such as state or country.
`VARCHAR(255)`	Variable-length text up to the given number of characters.
`TEXT, MEDIUMTEXT`	Long text such as article bodies.
`DATE, TIME, DATETIME`	Date only, time only, or both (YYYY-MM-DD HH:MM:SS).

`TIMESTAMP`	Stored in UTC and converted to the session time zone. Range ends in 2038.
`ENUM('pending', 'paid')`	One value from a fixed list.
`JSON`	JSON documents. In MariaDB this is an alias for LONGTEXT with a validity check.

CRUD: insert, select, update, delete

`INSERT INTO customers (name, email) VALUES ('Asha', 'asha@example.com');`	Adds one row.
`INSERT INTO customers (name, email) VALUES ('Ravi', 'ravi@example.com'), ('Meera', 'meera@example.com');`	Adds several rows in one statement.
`SELECT * FROM customers;`	Every column of every row. Name columns in real code.
`SELECT name, email FROM customers WHERE id = 1;`	Chosen columns for one row.
`SELECT LAST_INSERT_ID();`	ID generated by the last insert on this connection.
`UPDATE customers SET name = 'Asha Rao' WHERE id = 1;`	Changes matching rows. Without WHERE it changes every row.
`DELETE FROM customers WHERE id = 1;`	Removes matching rows. Without WHERE it removes every row.
`SET SQL_SAFE_UPDATES = 1;`	Blocks UPDATE and DELETE that have no key-based WHERE or LIMIT in this session.

Filtering, sorting and LIMIT

`WHERE status = 'paid' AND total > 500`	Combine conditions with AND / OR; use brackets to group.
`WHERE city IN ('Pune', 'Mumbai')`	Matches any value in the list.
`WHERE total BETWEEN 100 AND 500`	Inclusive range.
`WHERE name LIKE 'As%'`	`%` matches any characters, `_` exactly one.
`WHERE phone IS NULL`	NULL checks need IS NULL / IS NOT NULL, never = NULL.
`SELECT DISTINCT city FROM customers;`	Unique values only.
`ORDER BY total DESC, name ASC`	Sort by several columns.
`LIMIT 10`	First 10 rows.
`LIMIT 10 OFFSET 20`	Rows 21 to 30, for page 3 of 10 per page.
`SELECT name AS customer_name`	Column alias in the result.

Joins

`SELECT o.id, c.name FROM orders o JOIN customers c ON c.id = o.customer_id;`	INNER JOIN: only rows with a match in both tables.
`SELECT c.name, o.id FROM customers c LEFT JOIN orders o ON o.customer_id = c.id;`	LEFT JOIN: every customer, with NULLs where there is no order.

`... LEFT JOIN orders o ON o.customer_id = c.id WHERE o.id IS NULL;`	Customers who have never ordered.
`RIGHT JOIN`	Mirror of LEFT JOIN. Usually clearer to swap the tables and use LEFT JOIN.
`CROSS JOIN`	Every combination of rows from both tables.
`FROM employees e JOIN employees m ON m.id = e.manager_id`	Self join: a table joined to itself.
`SELECT ... UNION SELECT ...`	Stacks results and removes duplicates; `UNION ALL` keeps them and is faster.

Aggregates, GROUP BY and HAVING

`SELECT COUNT(*) FROM orders;`	Number of rows.
`COUNT(column)`	Counts non-NULL values only.
`SUM(total), AVG(total), MIN(total), MAX(total)`	Totals, averages and extremes.
`SELECT status, COUNT(*) FROM orders GROUP BY status;`	One row per group.
`SELECT customer_id, SUM(total) AS spent FROM orders GROUP BY customer_id HAVING spent > 5000;`	HAVING filters groups after aggregation; WHERE filters rows before.
`GROUP_CONCAT(name ORDER BY name SEPARATOR ', ')`	Joins values from a group into one string.
`ROUND(AVG(total), 2)`	Rounds the result to two decimals.
`COALESCE(SUM(total), 0)`	Returns 0 instead of NULL when there are no rows.

Indexes and EXPLAIN

`CREATE INDEX idx_orders_customer ON orders (customer_id);`	Speeds up lookups and joins on that column.
`CREATE INDEX idx_orders_status_date ON orders (status, ordered_at);`	Composite index: helps queries filtering on status, or status and date (left-most columns first).
`CREATE UNIQUE INDEX uq_users_email ON users (email);`	Index that also forbids duplicates.
`ALTER TABLE articles ADD FULLTEXT INDEX ft_body (title, body);`	Full-text index for word searches.
`WHERE MATCH(title, body) AGAINST ('php sessions')`	Full-text search using that index.
`SHOW INDEX FROM orders;`	Lists the indexes on a table.
`DROP INDEX idx_orders_customer ON orders;`	Removes an index.
`EXPLAIN SELECT * FROM orders WHERE customer_id = 7;`	Shows the query plan: check `key` (index used) and `rows` (rows examined).

`EXPLAIN ANALYZE SELECT ...;`	Runs the query and reports actual timings (MySQL 8.0.18+; MariaDB uses `ANALYZE SELECT`).
`WHERE ordered_at >= '2026-01-01'`	Index-friendly. Wrapping the column in a function, such as `YEAR(ordered_at) = 2026`, usually stops the index being used.

Users and privileges

`CREATE USER 'app'@'localhost' IDENTIFIED BY 'change-this-password';`	New user who can connect only from the same machine. Use a strong, unique password.
`CREATE USER 'report'@'10.0.0.%' IDENTIFIED BY '...';`	User allowed from a range of hosts.
`GRANT SELECT, INSERT, UPDATE, DELETE ON shop.* TO 'app'@'localhost';`	Least privilege: the web app can read and write data but not change the schema.
`GRANT SELECT ON shop.* TO 'report'@'10.0.0.%';`	Read-only access.
`SHOW GRANTS FOR 'app'@'localhost';`	Lists a user's privileges.
`REVOKE DELETE ON shop.* FROM 'app'@'localhost';`	Removes a privilege.
`ALTER USER 'app'@'localhost' IDENTIFIED BY 'new-password';`	Changes a password.
`DROP USER 'report'@'10.0.0.%';`	Deletes a user.
`SELECT user, host FROM mysql.user;`	Lists accounts on the server.

Backups, restore and transactions

`mysqldump -u root -p --single-transaction --routines --triggers shop > shop.sql`	Consistent backup of one InnoDB database without locking it. On recent MariaDB the tool is also called `mariadb-dump`.
`mysqldump -u root -p --no-data shop > schema.sql`	Structure only, no rows.
`mysqldump -u root -p --all-databases > all.sql`	Every database on the server.
`mysqldump -u root -p shop | gzip > shop.sql.gz`	Compressed backup (Linux and macOS).
`mysql -u root -p shop < shop.sql`	Restores into an existing database.
`gunzip < shop.sql.gz | mysql -u root -p shop`	Restores a compressed backup.
`START TRANSACTION;`	Begins a transaction (InnoDB tables).
`COMMIT;`	Saves every change in the transaction.
`ROLLBACK;`	Undoes every change since START TRANSACTION.
`SAVEPOINT step1; ROLLBACK TO step1;`	Undo part of a transaction.
`SELECT stock FROM products WHERE id = 5 FOR UPDATE;`	Locks the row until commit so two buyers cannot take the last item.

```
START TRANSACTION;

UPDATE accounts SET balance = balance - 500 WHERE id = 1;
UPDATE accounts SET balance = balance + 500 WHERE id = 2;

-- Check both updates affected one row each, then:
COMMIT;
-- or, if anything went wrong:
-- ROLLBACK;
```

Online version, with the matching free guide: skillnewton.com/resources/cheat-sheets/mysql-cheat-sheet/
Learn this hands-on: PHP Web Development: Foundation & Practical Implementation – skillnewton.com/courses/php-foundation-course/