

MySQL Cheat Sheet

MySQL is a popular open-source relational database management system known for its ease of use and scalability. Sometimes, you will need a little help while working on a project. That's why we created this MySQL Cheat Sheet.

Instructions for installing MySQL are available at:
<https://dev.mysql.com>

CONNECTING TO A MYSQL SERVER

Connect to a MySQL server with a username and a password using the mysql command-line client.

MySQL will prompt for the password:

```
mysql -u [username] -p
```

To connect to a **specific database** on a MySQL server using a username and a password:

```
mysql -u [username] -p [database]
```

To **export data** using the mysqldump tool:

```
mysqldump -u [username] -p \[database] > data_backup.sql
```

To exit the client:

```
quit or exit
```

For a full list of commands:

```
help
```

CREATING AND DISPLAYING DATABASES

To **create** a database:

```
CREATE DATABASE zoo;
```

To **list all the databases** on the server:

```
SHOW DATABASES;
```

To **use** a specified database:

```
USE zoo;
```

find the current database
SELECT database();

To **delete** a specified database:

```
DROP DATABASE zoo;
```

To **list all tables** in the database:

```
SHOW TABLES;
```

to show the columns from table
SHOW COLUMN FROM <tablename>

OR

DESC <tablename>
or
DESCRIBE <name>

To **get information about a specified table**:

```
DESCRIBE animal;
```

It outputs column names, data types, default values, and more about the table.

CREATING TABLES

To **create a table**:

```
CREATE TABLE habitat (  
  id INT,  
  name VARCHAR(64)  
);
```

Use **AUTO_INCREMENT** to increment the ID automatically with each new record. An AUTO_INCREMENT column must be defined as a **primary or unique key**:

```
CREATE TABLE habitat (  
  id INT PRIMARY KEY AUTO_INCREMENT,  
  name VARCHAR(64)  
);
```

To create a table with a foreign key:

```
CREATE TABLE animal (  
  id INT PRIMARY KEY AUTO_INCREMENT,  
  name VARCHAR(64),  
  species VARCHAR(64),  
  age INT,  
  habitat_id INT,  
  FOREIGN KEY (habitat_id)  
    REFERENCES habitat(id)  
);
```

MODIFYING TABLES

Use the ALTER TABLE statement to modify the table structure.

To **change a table name**:

```
ALTER TABLE animal RENAME pet;
```

To **add a column** to the table:

```
ALTER TABLE animal  
ADD COLUMN name VARCHAR(64);
```

To change a column name:

```
ALTER TABLE animal  
RENAME COLUMN id TO identifier;
```

To change a column data type:

```
ALTER TABLE animal  
MODIFY COLUMN name VARCHAR(128);
```

To delete a column:

```
ALTER TABLE animal  
DROP COLUMN name;
```

To **delete a table**:

```
DROP TABLE animal;
```

QUERYING DATA

To **select data from a table**, use the SELECT command.

An example of a single-table query:

```
SELECT species, AVG(age) AS average_age  
FROM animal  
WHERE id != 3  
GROUP BY species  
HAVING AVG(age) > 3  
ORDER BY AVG(age) DESC;
```

An example of a multiple-table query:

```
SELECT city.name, country.name  
FROM city  
[INNER | LEFT | RIGHT] JOIN country  
  ON city.country_id = country.id;
```

Use +, -, *, / to do some basic math.

To get the number of seconds in a week:

```
SELECT 60 * 60 * 24 * 7; -- result: 604800
```

AGGREGATION AND GROUPING

- **AVG**(expr) – **average** value of expr for the group.
- **COUNT**(expr) – **count** of expr values within the group.
- **MAX**(expr) – **maximum** value of expr values within the group.
- **MIN**(expr) – **minimum** value of expr values within the group.
- **SUM**(expr) – **sum** of expr values within the group.

To count the rows in the table:

```
SELECT COUNT(*)  
FROM animal;
```

To count the non-NULL values in a column:

```
SELECT COUNT(name)  
FROM animal;
```

To count unique values in a column:

```
SELECT COUNT(DISTINCT name)  
FROM animal;
```

GROUP BY

To count the animals by species:

```
SELECT species, COUNT(id)  
FROM animal  
GROUP BY species;
```

To get the average, minimum, and maximum ages by habitat:

```
SELECT habitat_id, AVG(age),  
  MIN(age), MAX(age)  
FROM animal  
GROUP BY habitat_id;
```

INSERTING DATA

To **insert data** into a table, use the INSERT command:

```
INSERT INTO habitat VALUES  
(1, 'River'),  
(2, 'Forest');
```

You may **specify the columns** in which the data is added. The remaining columns are filled with default values or NULLs.

```
INSERT INTO habitat (name) VALUES  
( 'Savanna');
```

UPDATING DATA

To **update the data in a table**, use the UPDATE command:

```
UPDATE animal  
SET  
  species = 'Duck',  
  name = 'Quack'  
WHERE id = 2;
```

we should use SELECT and
WHERE to find the data and
then update it to avoid any unneeded
changes

DELETING DATA

To **delete data from a table**, use the DELETE command:

```
DELETE FROM animal  
WHERE id = 1;
```

This deletes all rows satisfying the WHERE condition.

To delete all data from a table, use the TRUNCATE TABLE statement:

```
TRUNCATE TABLE animal;
```

CASTING

From time to time, you need to change the type of a value.

Use the CAST () function to do this.

In MySQL, you can cast to these data types:

CHAR	NCHAR	BINARY	DATE	DATETIME
DECIMAL	DOUBLE	FLOAT	REAL	SIGNED
UNSIGNED TIME	YEAR	JSON	spatial_type	

To get a number as a signed integer:

```
SELECT CAST(1234.567 AS signed);  
-- result: 1235
```

To change a column type to double:

```
SELECT CAST(column AS double);
```

MySQL Cheat Sheet

TEXT FUNCTIONS

FILTERING THE OUTPUT

To fetch the city names that are not Berlin:

```
SELECT name
FROM city
WHERE name != 'Berlin';
```

TEXT OPERATORS

To fetch the city names that start with a 'P' or end with an 's':

```
SELECT name
FROM city
WHERE name LIKE 'P%' OR name LIKE '%s';
```

To fetch the city names that start with any letter followed by 'ublin' (like Dublin in Ireland or Lublin in Poland):

```
SELECT name
FROM city
WHERE name LIKE '_ublin';
```

CONCATENATION

Use the `CONCAT()` function to concatenate two strings:

```
SELECT CONCAT('Hi ', 'there!');
-- result: Hi there!
```

If any of the string is NULL, the result is NULL:

```
SELECT CONCAT(Great ' ', 'day', NULL);
-- result: NULL
```

MySQL allows specifying a separating character (separator) using the `CONCAT_WS()` function. The separator is placed between the concatenated values:

```
SELECT CONCAT_WS(' ', 1, 'Olivier',
'Norris'); -- result: 1 Olivier Norris
```

OTHER USEFUL TEXT FUNCTIONS

To get the count of characters in a string:

```
SELECT LENGTH('LearnSQL.com'); CHAR_LENGTH()
-- result: 12
```

To convert all letters to lowercase:

```
SELECT LOWER('LEARNSQL.COM'); LCASE()
-- result: learnsql.com
```

To convert all letters to uppercase:

```
SELECT UPPER('LearnSQL.com'); UCASE()
-- result: LEARNSQL.COM
```

To get just a part of a string:

```
SELECT SUBSTRING('LearnSQL.com', 9);
-- result: .com
SELECT SUBSTRING('LearnSQL.com', 1, 5);
-- result: Learn
```

To replace a part of a string:

```
SELECT REPLACE('LearnSQL.com', 'SQL',
'Python');
-- result: LearnPython.com
```

NUMERIC FUNCTIONS

To get the remainder of a division:

```
SELECT MOD(13, 2); -- result: 1
```

To round a number to its nearest integer:

```
SELECT ROUND(1234.56789); -- result: 1235
```

To round a number to three decimal places:

```
SELECT ROUND(1234.56789, 3);
-- result: 1234.568
```

To round a number up:

```
SELECT CEIL(13.1); -- result: 14
SELECT CEIL(-13.9); -- result: -13
```

The `CEIL(x)` function returns the smallest integer not less than x. To round the number down:

```
SELECT FLOOR(13.8); -- result: 13
SELECT FLOOR(-13.2); -- result: -14
```

The `FLOOR(x)` function returns the greatest integer not greater than x. To round towards 0 irrespective of the sign of a number:

```
SELECT TRUNCATE(13.56, 0); -- result: 13
SELECT TRUNCATE(-13.56, 1); -- result: -13.5
```

To get the absolute value of a number:

```
SELECT ABS(-12); -- result: 12
```

To get the square root of a number:

```
SELECT SQRT(9); -- result: 3
```

USEFUL NULL FUNCTIONS

To fetch the names of the cities whose rating values are not missing:

```
SELECT name
FROM city
WHERE rating IS NOT NULL;
```

COALESCE(x, y, ...)

To replace NULL in a query with something meaningful:

```
SELECT domain,
COALESCE(domain, 'domain missing')
FROM contacts;
The COALESCE() function takes any number of arguments and returns the value of the first argument that is not NULL.
```

NULLIF(x, y)

To save yourself from *division by 0* errors:

```
SELECT last_month, this_month,
this_month * 100.0
/ NULLIF(last_month, 0)
AS better_by_percent
FROM video_views;
```

The `NULLIF(x, y)` function returns NULL if x equals y, else it returns the value of x value.

DATE AND TIME

There are 5 main time-related types in MySQL:

DATE TIME DATETIME TIMESTAMP YEAR

DATE – stores the year, month, and day in the YYYY-MM-DD format.

TIME – stores the hours, minutes, and seconds in the HH:MM:SS format.

DATETIME – stores the date and time in the YYYY-MM-DD HH:MM:SS format. The supported range is '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.

TIMESTAMP – stores the date and time. The range is '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC. MySQL converts TIMESTAMP values from the current time zone to UTC for storage, and back from UTC to the current time zone for retrieval.

YEAR – stores the year in the YYYY format.

INTERVALS

An interval is the duration between two points in time.

To define an interval: **INTERVAL 1 DAY**

This syntax consists of the INTERVAL keyword, a value, and a time part keyword (YEAR, QUARTER, MONTH, WEEK, DAY, HOUR, MINUTE, SECOND, MICROSECOND).

You may combine different INTERVALs using the + or – operator:

INTERVAL 1 YEAR + INTERVAL 3 MONTH

You may also use the standard SQL syntax:

INTERVAL '1-3' YEAR_MONTH

-- 1 year and 3 months

INTERVAL '3-12' HOUR_MINUTE

-- 3 hours 12 minutes

WHAT TIME IS IT?

To answer this question, use:

- `CURRENT_TIME` or `CURTIME` – to get the current time.
- `CURRENT_DATE` or `CURDATE` – to get the current date.
- `NOW()` or `CURRENT_TIMESTAMP` – to get the current timestamp with both of the above.

CREATING VALUES

To create a date, time, or datetime, write the value as a string and cast it to the proper type.

```
SELECT CAST('2021-12-31' AS date),
CAST('15:31' AS time),
CAST('2021-12-31 23:59:29' AS datetime);
```

You may skip casting in simple conditions; the database knows what you mean.

```
SELECT airline, flight_no, departure_time
FROM airport_schedule
WHERE departure_time < '12:00';
```

EXTRACTING PARTS OF DATES

To extract a part of a date, use the functions `YEAR`, `MONTH`, `WEEK`, `DAY`, `HOUR`, and so on.

```
SELECT YEAR(CAST('2021-12-31' AS date));
-- result: 2021
SELECT MONTH(CAST('2021-12-31' AS date));
-- result: 12
SELECT DAY(CAST('2021-12-31' AS date));
-- result: 31
```

DATE ARITHMETICS

To add or subtract an interval from a DATE, use the `ADDDATE()` function:

```
ADDDATE('2021-10-31', INTERVAL 2 MONTH);
-- result: '2021-12-31'
ADDDATE('2014-04-05', INTERVAL -3 DAY);
-- result: '2014-04-02'
```

To add or subtract an interval from a TIMESTAMP or DATETIME, use the `TIMESTAMPADD()` function:

```
TIMESTAMPADD(MONTH, 2,
'2014-06-10 07:55:00');
-- result: '2014-08-10 07:55:00'
TIMESTAMPADD(MONTH, -2,
'2014-06-10 07:55:00');
-- result: '2014-04-10 07:55:00'
```

To add or subtract TIME from a DATETIME, use the `ADDTIME()` function:

```
ADDTIME('2018-02-12 10:20:24', '12:43:02');
-- result: '2018-02-12 23:03:26'
ADDTIME('2018-02-12 10:20:24', '-12:43:02');
-- result: '2018-02-11 21:37:22'
```

To find the difference between two dates, use the `DATEDIFF()` function:

```
DATEDIFF('2015-01-01', '2014-01-02');
-- result: 364
```

To find the difference between two times, use the `TIMEDIFF()` function:

```
SELECT TIMEDIFF('09:30:00', '07:55:00');
-- result: '01:35:00'
```

To find the difference between two datetimes (in a given unit of time), use the `TIMESTAMPDIFF()` function. Here's an example with the difference given in weeks:

```
SELECT TIMESTAMPDIFF(
WEEK, '2018-02-26', '2018-03-21'
); -- result: 3
```