# MYSQL CHEAT SHEET

comparitech

## DATA QUERY LANGUAGE

### MYSQL FUNCTIONS

#### String Functions

| | |
|---|---|
| ASCII(character) | ASCII value of *character* |
| CHAR_LENGTH(string) | Returns the length of *string* (in characters) |
| CHARACTER_LENGTH(string) | As above |
| CONCAT(value1, value2, …) | Joins two or more expressions together |
| CONCAT_WS (separator, value1, value2, …) | Joins two or more expressions together with a *separator* |
| FIELD(value, list) | Returns the index position of a *value* in a *list* |
| FIND_IN_SET(string, string list) | Returns the position of *string* within a *string list* |
| FORMAT (number, decimal places) | Formats a number to "9,999,999.99", rounded to the given *decimal places* |
| INSERT(string, start, length, string2) | Inserts *string2* into **string1** at **start** position, replacing the number of characters in *length* |
| INSTR(string1, string2) | The first position of *string2* in **string1** |
| LCASE(string) | Converts *string* to lower-case |
| LEFT(string, length) | Leftmost *length* number of characters from *string* |
| LENGTH(string) | Returns the length of *string* (in bytes) |
| LOCATE(string1, string2, start) | The first *string1* in *string2* after *start* |
| LOWER(string) | Converts *string* to lower-case |
| LPAD(string1, length, string2) | Left-pads string1 with string2 up to *length* |
| LTRIM(string) | Removes leading spaces from a string |
| MID(string, start, length) | Substring of *length* from the *start* point in *string* |
| POSITION(string1 IN string2) | Position of the first *string1* in *string2* |
| REPEAT(string, number) | Repeats *string* the *number* of times |
| REPLACEstring1, string2, string3) | Replaces *string2* in *string1* with *string3* |
| REVERSE(string) | Reverses the character order in *string* |
| RIGHT(string, number) | A substring counting from the right |
| RPAD(string1, length, string2) | Right-pads string1 with string2 up to *length* |
| RTRIM(string) | Trims trailing spaces from *string* |
| SPACE(number) | Returns a string of *number* space characters |
| STRCMP(string1, string2) | Compares two strings |
| SUBSTR(string, start, length) | Extracts *length* characters in *string* from *start* |
| SUBSTRING(string, start, length) | As above |
| SUBSTRING_INDEX (string, character, number) | Cuts off *string* when *character* has occurred the *number* of times |
| TRIM(string) | Removes leading and trailing spaces from *string* |
| UCASE(string) | Converts *string* to upper-case |
| UPPER(string) | Converts *string* to upper-case |

#### Date Functions

| | |
|---|---|
| ADDDATE(date, INTERVAL value unit) | Increases the *date* |
| DATE_ADD(date, INTERVAL value unit) | As above |
| CURDATE() | Returns the current date |
| CURRENT_DATE() | As above |
| CURRENT_TIME() | Returns the current time |
| CURTIME() | As above |
| CURRENT_TIMESTAMP() | Returns the current date and time |
| LOCALTIME() | As above |
| LOCALTIMESTAMP() | As above |
| NOW() | As above |
| SYSDATE() | As above |
| DATE(string) | Extracts the date from *string* |
| DATEDIFF(date1, date2) | Gives the number of days between two dates |
| DATE_FORMAT(date, format) | Formats *date* according to the *format* code |
| DATE_SUB(date, INTERVAL number unit) | Subtracts a period from *date* |
| DAY(date) | Returns the day of the month in *date* |
| DAYOFMONTH(date) | As above |
| DAYNAME(date) | Returns the weekday name for a given date |
| DAYOFWEEK(date) | The weekday number for a given date |
| DAYOFYEAR(date) | Returns the day of the year as a number |
| EXTRACT(unit FROM date) | Extracts a datetime *unit* from *date* |
| FROM_DAYS(number) | Returns a date from a number |
| HOUR(datetime) | Returns the hour part of *datetime* |
| LAST_DAY(datetime) | Gives the last day of the month for *datetime* |
| MAKEDATE(year, day) | Creates and returns a date |
| MAKETIME(hours, minutes, seconds) | Creates and returns a time |
| MICROSECOND(datetime) | Returns the microsecond part of *datetime* |
| MINUTE(datetime) | Returns the minute part of *datetime* |
| MONTH(date) | Returns the month number of *date* |
| MONTHNAME(date) | Returns the name of the month for *date* |
| PERIOD_ADD(date, number) | Adds *number* to the *date* format YYMM |
| PERIOD_DIFF(date1, date2 | Difference between *date1* and *date2* |
| QUARTER(date) | Returns the quarter number of *date* |
| SECOND(datetime) | Returns the seconds part of *datetime* |
| SEC_TO_TIME(number) | Converts *number* of seconds into a time |
| STR_TO_DATE(date, format) | Gives a date based on a string and a format |
| SUBDATE(date, INTERVAL value unit) | Subtracts the interval from *date* |
| SUBTIME(datetime, number) | Subtracts time *number* from *datetime* |
| TIME(datetime) | Extracts the time part of *datetime* |
| TIME_FORMAT(time, format) | Formats a time |
| TIME_TO_SEC(time) | Converts *time* to number of seconds |
| TIMEDIFF(time1, time2) | Returns the difference between two times |
| TIMESTAMP(date, time) | Create a datetime from *date* and *time* |
| TO_DAYS(date) | The number of days since year 0 to *date* |
| WEEK(date) | Returns the week number for *date* |
| WEEKDAY(date) | Returns the weekday number for *date* |
| WEEKOFYEAR(date) | Returns the week number for *date* |
| YEAR(date) | Returns the year part of *date* |
| YEARWEEK(date) | Returns the year and week number for *date* |

Date and time units used in functions:

- MICROSECOND
- SECOND
- MINUTE
- HOUR
- DAY
- WEEK
- MONTH
- QUARTER
- YEAR
- SECOND_MICROSECOND
- MINUTE_MICROSECOND
- MINUTE_SECOND
- HOUR_MICROSECOND
- HOUR_SECOND
- HOUR_MINUTE
- DAY_MICROSECOND
- DAY_SECOND
- DAY_MINUTE
- DAY_HOUR
- YEAR_MONTH

Date formatting codes for date functions:

| | |
|---|---|
| %a | Abbreviated weekday name (Sun to Sat) |
| %b | Abbreviated month name (Jan to Dec) |
| %c | Numeric month name (0 to 12) |
| %D | Ordinal day of the month |
| %d | Day of the month as a numeric value (01 to 31) |
| %e | Day of the month as a numeric value (0 to 31) |
| %f | Microseconds (000000 to 999999) |
| %H | Hour (00 to 23) |
| %h | Hour (00 to 12) |
| %I | Hour (00 to 12) |
| %i | Minutes (00 to 59) |
| %j | Day of the year (001 to 366) |
| %k | Hour (0 to 23) |
| %l | Hour (1 to 12) |
| %M | Month name in full (January to December) |
| %m | Month name as a numeric value (00 to 12) |
| %p | AM or PM |
| %r | Time in 12 hour AM or PM format (hh:mm:ss AM/PM) |
| %S | Seconds (00 to 59) |
| %s | Seconds (00 to 59) |
| %T | Time in 24 hour format (hh:mm:ss) |
| %U | Week. Sunday is the first day of the week (00 to 53) |
| %u | Week. Monday is the first day of the week (00 to 53) |
| %V | Week. Sunday is the first day of the week (01 to 53). Used with %X |
| %v | Week. Monday is the first day of the week (01 to 53). Used with %X |
| %W | Weekday name in full (Sunday to Saturday) |
| %w | Day of the week where Sunday=0 and Saturday=6 |
| %X | Year for the week. Sunday is the first day of the week. Used with %V |
| %x | Year for the week. Monday is the first day of the week. Used with %V |
| %Y | Year as a numeric, 4-digit value |
| %y | Year as a numeric, 2-digit value |

## Numeric Functions

| | |
|---|---|
| ABS(number) | Converts a negative number into positive |
| ACOS(number) | Returns the arc cosine of *number* |
| ASIN(number) | Returns the arc sine of *number* |
| ATAN(number) | Returns the arc tangent of *number* |
| ATAN(number1, number2) | Returns the arc tangent of two numbers |
| ATAN2(number1, number2) | As above |
| AVG(expression) | Average value of a list or a column in a SELECT |
| CEIL(number) | Rounds up *number* with decimal places |
| CEILING(number) | As above |
| COS(number) | Returns the cosine of *number* |
| COT(number) | Returns the cotangent of *number* |
| COUNT(column) | Count of records returned by a SELECT |
| DEGREES(number) | Converts *number* in radians to degrees |
| DIV | SELECT number1 DIV number2 |
| EXP(number) | Returns e raised to the power of *number* |
| FLOOR(number) | Rounds down *number* with decimal places |
| GREATEST(list) | Returns the greatest value in *list* or a column |
| LEAST(list) | Returns the smallest value of *list* or a column |
| LN(number) | Returns the natural logarithm of *number* |
| LOG(number) | As above |
| LOG10(number) | Gives the natural logarithm of *number* to base 10 |
| LOG2(number) | Returns the natural logarithm of *number* to base 2 |
| MAX(list) | Returns the maximum value in *list* or a column |
| MIN(list) | Returns the minimum value in *list* or a column |
| MOD(number1, number2) | The remainder of *number1* divided by *number2* |
| PI() | Returns the value of PI |
| POW(number1, number2) | Returns number1 to the power number2 |
| POWER(number1, number2) | As above |
| RADIANS(number) | Converts a degree value into radians |
| RAND() | Returns a random number |
| RAND(number) | Returns a repeatable random number |
| ROUND(number1, number2) | Rounds *number1* to *number2* decimal places |
| SIGN(number) | Returns the sign of *number* as 1 or -1 |
| SIN(number) | Returns the sine of *number* |
| SQRT(number) | Returns the square root of *number* |
| SUM(expression) | The sum of a column in a SELECT |
| TAN(number) | Returns the tangent of *number* |
| TRUNCATE(number) | Removes the decimal places of *number* |

## SELECT STATEMENT OPTIONS

```
SELECT *
FROM   table

SELECT DISTINCT column1
FROM   table

SELECT *
FROM   table
WHERE column7 = value

SELECT *
FROM  table
WHERE column7 = value
AND      column3 IN (value1, value2, value3 …)

SELECT *
FROM  table
WHERE column7 = value
OR        column4 LIKE pattern

SELECT *
FROM  table
WHERE column7 = value
AND      (column3 IN (value1, value2, value3 …)
          OR column4 LIKE pattern)

SELECT table1.column3,
           table2.column3,
           table2.column5
FROM   table1,
           table2
WHERE table1.column7 = value
AND      table2.column4 = table1.column1

SELECT *
FROM  table
WHERE column7 BETWEEN value1 AND value2

SELECT column1
FROM  table
WHERE column8 IS NULL

SELECT *
FROM  table
WHERE column7 BETWEEN value1 AND value2
AND column8 IS NOT NULL

SELECT a.column1,
           a.column2,
           b.column7
FROM   table1 AS a,
           table2 AS b
WHERE a.column1 = b.column1
AND      a.column3 = value1

SELECT a.column1 AS heading1,
           a.column4 AS heading2,
           b.column7 AS heading3
FROM   table1 AS a,
           table2 AS b
WHERE a.column3 = value
AND      b.column4 = a.column1

SELECT a.column1,
           a.column2
FROM table AS a
INNER JOIN table2 AS b
ON a.column1 = b.column7
WHERE a.column4 = value

SELECT *
FROM  table1
LEFT JOIN (table2, table3)
ON (table2.column4 = table1.column1
     AND table3.column7 = table2.column1)

SELECT column7
FROM   table 1
LEFT JOIN table2
USING (column1, column2)

SELECT column7
FROM   table1
WHERE EXISTS
    (SELECT token
     FROM  table2
     WHERE table2.column1 = value
      AND      table2.column7 = table1.column4)

SELECT column7
FROM   table1
WHERE NOT EXISTS
    (SELECT token
     FROM  table2
     WHERE table2.column1 = value
      AND      table2.column7 = table1.column4)

SELECT column1
FROM   table1
WHERE column2 > ALL (SELECT column5
                              FROM   table2)

SELECT column1
FROM   table1
WHERE column2 > ANY (SELECT column5
                              FROM   table2)

SELECT column1
FROM   table1
WHERE column2 <> SOME (SELECT column5
                                  FROM   table2)

SELECT *
FROM  table
WHERE column3 IN (SELECT column5
                              FROM   table2)

SELECT column1
FROM  table
WHERE column3 NOT IN (SELECT column5
                              FROM   table2)

SELECT table1.column4,
           table_name.name2
FROM table1,
         (SELECT column7 AS name1,
                    column2 AS name2
          FROM   table2) AS table_name
WHERE table_name.name1 = table1.column1

SELECT column4
FROM   table1
WHERE column7 = value
UNION
SELECT column2
FROM   table2
WHERE column3 = value

SELECT column4
FROM   table1
WHERE column7 = value
UNION ALL
SELECT column2
FROM   table2
WHERE column3 = value

SELECT column1,
           column2
FROM   table
ORDER BY column2

SELECT column1,
           aggregate_function(column3)
FROM   table
GROUP BY column1

SELECT column1,
           aggregate_function(column3)
FROM   table
WHERE column_name <> value
GROUP BY column1
HAVING aggregate_function (column3) > 7
```

## DATA DESCRIPTION LANGUAGE

### DATA TYPES

| | |
|---|---|
| CHAR(size) | Fixed length string (up to 255 characters) |
| VARCHAR(size) | Variable length string (up to 255 characters) |
| TINYTEXT | A string up to 255 characters |
| TEXT | A string up to 65,535 bytes |
| MEDIUMTEXT | A string up to 16,777,215 characters |
| LONGTEXT | A string up to 4,294,967,295 characters |
| BLOB | Binary large objects (up to 65,535 bytes) |
| MEDIUMBLOB | Binary large objects (up to 16,777,215 bytes) |
| LONGBLOB | Binary large objects (up to 4,294,967,295 bytes) |
| ENUM(x,y,z,etc.) | A list up to 65,535 values |
| SET | A list up to 64 values |
| TINYINT(size) | -128 to 127 normal. 0 to 255 UNSIGNED |
| SMALLINT(size) | -32768 to 32767 normal. 0 to 65535 UNSIGNED |
| MEDIUMINT(size) | -8388608 to 8388607 |
| INT(size) | -2147483648 to 2147483647 |
| BIGINT(size) | -9223372036854775808 to 9223372036854775807 |
| FLOAT(size,d) | A small number – d = decimal places |
| DOUBLE(size,d) | A large number – d = decimal places |
| DECIMAL(size,d) | A DOUBLE stored as a string – d = decimal places |
| DATE() | YYYY-MM-DD |
| DATETIME() | YYYY-MM-DD HH:MI:SS |
| TIME() | HH:MI:SS |
| TIMESTAMP() | Number of seconds since '1970-01-01 00:00:00' UTC |
| YEAR() | Year in two-digit or four-digit format |

### OBJECT DEFINITION STATEMENTS

```
CREATE TABLE table_name
(column_name_1 data_type,
 column_name_2 data_type,
…
column_name_n data_type)

CREATE TABLE table_name
(column_name_1 data_type NOT NULL AUTO_INCREMENT,
 column_name_2 data_type NOT NULL,
…
column_name_n data_type
PRIMARY KEY (column_name_1)
)

CREATE INDEX index_name ON
table_name (column_1, … column_n)

CREATE UNIQUE INDEX ON
table_name (column)

CREATE VIEW view_name AS
select_statement

CREATE OR REPLACE VIEW viewname AS
select_statement
```

```
CREATE TRIGGER trigger_name
BEFORE | AFTER
INSERT | UPDATE | DELETE
ON table_name FOR EACH ROW
FOLLOWS | PRECEDES
BEGIN
  SQL statements
END

CREATE PROCEDURE procedure_name
          (optional parameter_list)
procedure_code

CREATE FUNCTION function_name
          (parameter_list)
RETURNS data_type
function_code

ALTER TABLE table_name
ADD column_name data_type


ALTER TABLE table_name
DROP COLUMN column_name


ALTER VIEW view_name AS
select_statement

DROP TABLE table_name

DROP INDEX index_name

DROP VIEW view_name

DROP TRIGGER trigger_name

DROP PROCEDURE procedure_name

DROP FUNCTION function_name

RENAME TABLE table_name TO new_table_name

TRUNCATE TABLE table_name
```

## DATA MANIPLATION LANGUAGE

```
INSERT INTO table_name
         (column1, column2, …)
VALUES
        (value1, value2, …)

INSERT INTO table_name
VALUES (value1, value2, …)

INSERT INTO table_name
         (column1, column2, …)
select_statement

INSERT INTO table_name
         (column1, column2, …)
VALUES
        (value1, value2, …)
ON DUPLICATE KEY UPDATE column1 = value

INSERT INTO table_name
SET column1 = value1,
     column2 = value2
     …

INSERT IGNORE INTO table_name
         (column1, column2, …)
VALUES
        (value1, value2, …)

REPLACE INTO table_name
         (column1, column2, …)
VALUES
        (value1, value2, …)

UPDATE table_name
SET column1 = value1,
     column2 = value2
     …
WHERE where_condition

UPDATE IGNORE table_name
SET column1 = value1,
     column2 = value2
     …
WHERE where_condition

DELETE FROM table_name

DELETE * FROM table_name

DELETE FROM table_name
WHERE where_condition
```