MySQL is the most popular, Oracle backed, open-source Structured Query Language. It is a Relational Database Management System. It is a component of the LAMP [Linux – Apache – MySQL – PHP/Python/Perl] Application Stack. Various other database-driven applications also implement it. MySQL 8.0 is the latest version available now. This MYSQL cheat sheet assumes that MySQL is already installed, and there is a MySQL Server accessible for connection.

MySQL Cheat Sheet

1. MySQL Prompts

Mysql>	This prompt indicates ready for a new query.
->	Next line of a multi-line query
`>	Continues to wait for identifier completion beginning with.`
'>	Continues to wait to the end of the string that was begun with '
">	Continues to wait to the end of the string that was begun with "
/*>	Waits for the completion of a comment begun with /*

2. Data Types

VARCHAR	A string of variable length up to 65535.
CHAR	A fixed-length character variable. T is declared with the number of characters. Ex. CHAR(15)
TEXT	Used to store long-form text strings. Generally used to store article bodies.
TINY TEXTTINYTEXT	It is used to store short strings of information. It stores up to 255 bytes or 255 characters along with 1 byte as overhead.
MEDIUMTEXT	It is useful for storing larger text strings like whitepapers, etc. These data objects can be of 16MB size.
LONGTEXT	It is used for storing extreme long text strings. This can be of 4GB size.

BLOB	These are binary strings which are treated as numeric values. They are used to store datafiles like images and videos.
BIT	Stores bit values.
BOOL | BOOLEAN	It holds either a 0 or a nonzero value. The value 0 is considered false, a non-zero value is considered TRUE.
TINYINT	It holds an integer value. The range of a signed small integer is -128 to 127, The unsigned field is 0-255.
SMALLINT	It holds an integer value. The range of a signed small integer is -32768 to 32767, The unsigned range is 0-65535
MEDIUMINT	It holds an integer value. The range of a signed small integer is -8388608 to 8388607, The unsigned range is 0-16777215
INT	Used to store an integer value. The range of a signed small integer is -2147483648 to 2147483647, The unsigned range is 0-4294967295
BIGINT	It is used to hold a big integer. Its signed range is -2E63 to 2E63 -1. The unsigned range is 0 to 2E64-1.
FLOAT	It is a single precision floating-point number. Its permissible values are -3.402823466E+38 to -1.175494351E-38, 0, and 1.175494351E-38 to 3.402823466E+38.
DOUBLE	It is a double precision floating-point number. Its permissible values are -1.7976931348623157E+308 to -2.2250738585072014E-308, 0, and 2.2250738585072014E-308 to 1.7976931348623157E+308.

DECIMAL	Used to store an exact fixed-point decimal value. The maximum number of digits is 65, while the maximum number of the decimal is 30.
ENUM	It is a string object. Its value is selected from a list of benefitsvalues. It uses numeric indexes to represent string values.
JSON	Defines JSON datatype to store JSON documents.
SET	It is a string object that enables us to store zero or more values from a list of pre-specified values when the table is created.
DATE	Renders a datatype as a date value. – YYYY-MM-DD
DATETIME	It is a combination of datetime. It displays datetime values in YYYY-MM-DD hh:mm:ss format.
TIMESTAMP	This datatype holds a combination date and time within the range 1970-01-01 00:00:00 UTC to 2038-01-19 03:14:07 UTC
TIME	Returns the current time.

3. Date – Time Functions

TIMEDIFF()	Calculates the difference between two in terms of time between two time or datetime values.
TIMESTAMPDIFF()	Returns the difference between two date or date-time values.
CURDATE()	Gives the current date.
NOW()	Returns the date and time of statement execution.
DATADIFF()	Returns the number of days between the two dates
DATE_FORMAT()	Reformats a date value based on the specified format.
DAY()	Returns the day of the specified date.
DAYNAME()	Returns the day name for the specified date.

DAYOFWEEK()	Returns the day of the week index for a specified date.
MONTH()	Returns the month number of the specified date.
STR_TO_DATE()	Converts a string into date-time based on a specified format.
SYSDATE()	Returns the system configured date.
WEEK()	It returns the week number of a specified date.
WEEKDAY()	Returns the index of the weekday of a specified date.
YEAR()	Returns the year from a specified date.

4. Working with Tables

CREATE	`CREATE table (<table_name> <field1(type)>, <field2, (type)>…);`	Used to create a new table
INSERT	`INSERT INTO <table_name>(field1, field2, …) VALUES(value1, value 2,…)`	This command is used to insert one or more rows of data into the table.
UPDATE	`UPDATE <table_name> SET field_name1 = value 1, field_name2 = value 2, [WHERE <condition>];`	Used to modify existing data in a table. The WHERE command here is optional here.

SELECT	`SELECT <field_list> FROM <table_name>;`	Used to select list values from a table.
SELECT DISTINCT	`SELECT DISTINCT <field_list> FROM <table_name>;`	Used to select only unique values from a list.
WHERE	`SELECT <filed_list> FROM <table_name> WHERE <condition>;`	The WHERE clause allows to select values based on specified conditions.
ORDER BY	`SELECT <field_list> FROM <table_name> ORDER BY <field_name> [ASC|DESC];`	The ORDER BY clause is used to sort the queried result set in either ascending or Descending Order. By default, it will be Ascending Order.
AND	`… expression_1 AND expression_2`	This is a logical operator. It is mainly used with the WHERE clause. Here, the query statement is executed only when both the expressions are true.
OR	`… expression_1 OR expression_2`	This is a logical operator. It is mainly used with the WHERE clause. Here, the query statement is executed only when one or both the expressions are true.
IN	`SELECT <field_list> FROM <table_name> WHERE <expression|column_1> IN (value1, value2, …)`	The IN operator is used with the WHERE clause. It enables to determine if a specified value in a list matches in another set of values.

BETWEEN	`… expression [NOT] BETWEEN expression_1 AND expression_2`	This operator is also used with the WHERE clause. It is used to specify whether a value is in a specified range.
LIMIT	`SELECT <field_list> FROM <table_name> LIMIT <first_row_offset> <number of rows to be returned>;`	This clause is used with SELECT statement to specify the number of rows to be returned.
IS NULL	`Value IS NULL`	It is used to test if a value is NULL or not. If it is NULL, the expression returns true, else false.
INNER JOIN	`SELECT <field_list> FROM <table1_name> INNER JOIN <table2_name> ON <join_condition>`	This is a filter clause that matches each row in one table with every row in the other table thus enabling to query only those rows that have corresponding columns from both tables.
LEFT JOIN	`SELECT <field_names> FROM <table1_name> LEFT JOIN <table2_name> ON join_condition;`	It allows you to query data from multiple tables. It matches each row from the first table to each row in the second table on the join_condition
RIGHT JOIN	`SELECT <field_names> FROM <table1_name> LEFT JOIN <table2_name> ON join_condition;`	Same as LEFT_JOIN except that the table manipulation is in reverse order. It matches each row from the second table to each row in the first table on the join_condition

CROSS JOIN	`SELECT * FROM <table 1_name> CROSS JOIN <table2_n ame>;`	It returns the cartesian product of rows from the joined tables.
GROUP BY	`SELECT <field1, fiel d2, field3…> FROM <table1_name> WHERE <condition/exp ression> GROUP BY <field1, fi eld2, field3…>`	This command enables to group rows into subgroups based on column or expression values.
HAVING	`SELECT <field1, fiel d2, field3…> FROM <table1_name> WHERE <condition/exp ression> GROUP BY <field1, fi eld2, field3…> HAVING <group_condit ion>`	It is generally used with the GROUP BY clause. It is used to specify filter conditions for group of rows.
ROLL UP	`SELECT <field_name1 >, SUM(column_name) <fi eld_name2> FROM <table_name> GROUP BY <field_name 1>WITH ROLLUP;`	Used to generate the subtotals as well as grandtotals of fieldvalues.
EXISTS	`SELECT <field_list> FROM <table_list> WHERE [NOT] EXISTS(s ubquery);`	It is a Boolean operator that returns either true or false. It is generally used to determine if a query/subquery has returned any number of rows.

INTERSECT	`(SELECT <field_list> FROM <table1_name>) INTERSECT (SELECT <field_list> FROM <table2_name>)`	This is a set operator which returns only distinct rows of two or more queries.
UNION	`SELECT <field_list> UNION [DISTINCT | ALL] SELECT <field_list> UNION [DISTINCT | ALL]`	This command combines two or more result sets from multiple SELECT queries and returns a single result set.
UPDATE JOIN	`UPDATE <table1>, <table2>. [INNER JOIN | LEFT JOIN] table1.common_field = table2.common_field SET table1.field1 = newvalues … WHERE <condition>`	JOIN clauses when used with the UPDATE statement is called as UPDATE JOIN.
DELETE	`DELETE FROM <table_name> WHERE <condition>`	This command is used to delete data from table.
DELETE JOIN	`DELETE <field1>,<field2> FROM <table1> INNER JOIN <table2> ON table1.key = table2.key WHERE <condition>;`	This command is used to delete data from multiple tables using the JOIN statement.

ON DELETE CASCADE	`SELECT <table_name> FROM <referential_constraints> WHERE <constraint_schema = 'database_name'> AND referenced_table_name = 'parent_table' AND delete_rule = 'CASCADE'`	This command enables deleting data from child table automatically when data from master table is deleted.
REPLACE	`REPLACE <table_name> (<field1>, <field2>,…) VALUES (<value1>, <value2>, …)`	This command is used to inert or update data in a table.

5. String Functions

ASCII()	`ASCII('string');`	Returns the ASCII value of the left most character of the string passed as an argument.
BIN()	`BIN (number);`	Returns the string representation of the binary value of the number passed as an argument.
CHARACTER_LENGTH() | CHAR_LENGTH()	`CHARACTER_LENGTH ('string');`	Returns the length of the string passed as an argument.

LOWER()	`LOWER('string');`	Changes all the characters of the argument string into lower case.
UPPER()	`UPPER('string');`	Changes all the characters of the argument string into lower case.
CONCAT()	`CONCAT(''string 1','string 2',...)`	Appends one string at the end of the other string. Any number of string's can be passed as an argument. If any argument is NULL, then the function will return NULL.
LENGTH()	`LENGTH('string');`	Returns the length of the string.

LEFT()	`LEFT ('string','number of characters to be returned');`	Returns a specified number of characters from left most side of the argument string.
RIGHT()	`RIGHT('string','number of characters to be returned');`	Returns a specified number of characters from right most side of the argument string.
TRIM()	`TRIM(' string ');`	Removes all leading and trailing spaces from a string.
LTRIM()	`LTRIM(' string ');`	Removes all leading space from a character string.

RTRIM()	`RTRIM('string ');`	Removes all trailing space from a character string.
FORMAT()	`FORMAT(number, number of decimal places);`	Formats a number as '#,###,###.##', rounded to a specified number of decimal places and returns the result as a string.
SUBSTRING() | SUBSTR()	`SUBSTR('string', position, length);`	Returns a substring from an argument string, starting from a specified position, of a specified length.
SUBSTRING_INDEX()	`SUBSTRING_INDEX('string','delimiter', count)`	Returns a substring from an argument string before a 'count' number of occurrences of a specified delimiter. If the 'count' specified in the argument is positive, then all characters left of the string is returned, if it is negative, then all characters right of the string is returned. Further, it performs a case sensitive search for the specified delimiter.

STRCMP()	`STRCMP('string 1','string 2');`	It compares two strings passed in the argument and returns 0 if both are equal, -1 if the first string is smaller than the second and 1 if the first string is greater than the second.
LIKE()	`'String' LIKE 'patter n';`	This command is used to match a string with a pattern on a per character basis. It returns 1 if there is a pattern, and 0 if there is no pattern.
POSITION() |LOCATE()	`LOCATE('substring','string');` `POSITION('substring' IN 'string');`	Returns the position of a substring in a string.

REVERSE()	`REVERSE('string');`	Reverses the order of the string argument.
REGEXP()	`'string' REGEXP 'pattern';`	Returns 1 if the string argument matches a specified pattern, else returns 0.
LOAD_FILE()	`LOAD_FILE(file_name);`	Returns the contents of a file as a string.

6. Math Functions

ABS()	`ABS()`	Returns the absolute value of a number passed to it as an argument
TRUNCATE()	TRUNCATE(number_to be truncated, number of decimal places to be truncated to)	It trims a number to a specified number of decimal places passed to it as an argument.
ROUND()	`ROUND(number)`	It rounds a number to a specified number of decimal places passed to it as an argument.

MOD()	`MOD(number)`	Returns the remainder of a number after dividing it with another number.
CEIL() | CEILING()	`CEIL(number)`	It returns the integer value which is equal to or is the next greater integer value of the specified number. Ex CEIL(3.5) – Returns 4.
CONV()	`CONV(number, from the base, to base)`	Converts a number of one base to a number of another base.
FLOOR()	`FLOOR(number)`	It returns the integer value which is equal to or is the next smaller integer value of the specified number. Ex FLOOR(3.15) – Returns 3. FLOOR(-3.15) – Returns -4.
PI()	`PI()`	Returns the value of PI.
RAND()	`RAND()`	Returns a random floating-point number within the range 0 - 1.
SIGN()	`SIGN(number)`	Returns the sign of a number passed as an argument.
SQRT()	`SQRT()`	Returns the square root of a number passed as an argument.
RADIANS()	`RADIANS(number value in degrees)`	Returns an argument number in terms of radians

MIN()	`MIN(expression)` `SELECT MIN(marks) as minimum_marks FROM <marks_table>`	Returns the minimum value in a set of values. Here, 'marks' refers to the field name.
MAX()	`MAX(expression)` `SELECT MIN(marks) as minimum_marks FROM <marks_table>`	Returns the maximum value in a set of values. Here, 'marks' refers to the field name.

7. Information Functions

CONNECTION_ID()	`SELECT CONNECTION_ID`	Returns the Connection ID for a particular connection
CURRENT_USER()	`SELECT CURRENT_USER();`	It returns the combination of the username and the hostname used by the MySQL account server to authenticate the current user.
DATABASE()	`SELECT DATABASE();`	Returns the default database name.
SCHEMA()		Same as DATABASE()

FOUND_ROWS()	`SELECT FOUND_ROWS();`	Returns the number of rows found by a query without actually running the query. In order to refer to this value, later on, it needs to be saved.
ROW_COUNT()	`SELECT ROW_COUNT();`	It returns the number of affected rows following a statement execution.
LAST_INSERT_ID()	`UPDATE sequence SET p_id = LAST_INSERT_ID (p_id +1) SELECT LAST_INSERT_ID();`	It returns a 64-bit value which is an automatically generated value that has been successfully inserted for an AUTO_INCREMENT column. This value remains unchanged if no new rows are inserted successfully.
USER(), SESSION_USER() SYSTEM_USER	`SELECT USER();`	Returns the MySQL username and Hostname.
VERSION()	`SELECT VERSION ();`	Returns the MySQL Version number.

CHARSET()	`SELECT CHARSET(USER());`	Returns the character set of the string argument.
ENCRYPT()	`SELECT ENCRYPT('string');`	Returns a binary string.
DECODE()		Decodes or decrypts an encrypted string.
MD5()	`SELECT MD5('password');`	It calculates an MD5 128-bit checksum for the string and returns a string of 32 hexadecimal digits.
PASSWORD()	`SELECT PASSWORD('password_text')`	Uses a cleartext password str to return a hashed password string.
COMPRESS()	`SELECT COMPRESS('string_to_compress')`	This command is used to compress a string and returns a binary string.

UNCOMPRESS()	`SELECT UNCOMPRESS('String_word');`	This command is used to uncompress a string that is compressed using the COMPRESS() command.
SLEEP()	`SELECT SLEEP(2000);`	This command is used to pause for a specified number of seconds in the argument.
DEFAULT()	`UPDATE <table_name1> SET v=DEFUALT (v)+1 where id <10;`	Returns the default value for a field. If no default value is specified, it returns an error.

8. Working With Databases

Show databases;	To list all the existing databases
Create Database <database_name>	Creates a database of name <database_name>
Use <database_name>	Selects the <database_name> to be used.
Grant all on <database_name> to <mysql_user> @ <client_host>	Used to grant all permission on <database_name> to the mysql_user and the host_server
Drop <database_name>	Delete database_name

9. Database Connection

mysql -h host_name -u user_name -p	Here the host_name refers to the name of the host where MySQL server is running. User_name is the user name of the MySQL account.
QUIT | \q	Disconnects from the server.

10. PHP -> MYSQL

`<?php` `$host = 'localhost';` `$dbname='<database_name>';` `$username='<username>';` `$password='<password>';` `?>`	Included in the dbconfig.php file. It contains all the configured parameters for database configuration.
`<?php` `require_once 'dbconfig.php';` `try` `{` `$conn = new PDO("mysql:host=$host;dbname=$dbname", $usernam e, $password);` `}` `?>`	This file can be named as phpmysqlconnect.php. Here the code for establishing database connection has been included. More code can be included here in order to handle exceptions in case the connection is not successful.

11. Python – MySQL

Pip	`Pip install mysql -connector -pyt hon`	Enables installation of MySQL Python connector on any Operating System. This includes Linux, Unix, macOS & Windows.
	`Pip uninstall mysql -connector -p ython`	This command will enable you to uninstall the current MySQL Connector/Python.

Connect()	``` Import mysql.connector From mysql.connector import error Def connect(): Try: Conn = mysql.connector.connect(ho st = 'localhost', database = 'pyt hon_mysql', user = 'root', passwo rd = 'Securepass1!') if conn.is_connected(): print('Connection Successful') except Error as e: print(e) finally: if conn is not None and conn.is_c onnected(): conn.close() ```	Here the mysql.connector and Error objects are first imported from the Python package. The connect() function is used to connect to the MySQL server. The connect() function needs specification for 4 parameters. They are: • Host • Database • User • Password The is_connected() function is used to check if the connection has been established successfully. Close() function is used to close the database connection.

12. Node.js – MySQL

`` Let mysql = require (‘mysql’) ```	Imports the mysql module
`` let connection = mys ql.createConnection ({ host: 'localhost', user: 'root', password: '<password > ', database: ‘database_ name’ }); ```	Connects to the MySQL Database by calling the creatconnection() method.

`connection.end(function(err) { if (err) { return console.log('error:' + err.message); } console.log('Close database connection.'); });`	The end() method allows the execution of all remaining queries before closing the database connection.
Connection.destroy()	This method is used to close the connection immediately. It further does not allow any triggers for the connection.

13. JDBC – MySQL

`Import java.sql.Connection; Import java.sql.DriverManager Import java.sql.SQLException`	The three classes Connection, Driver Manager and SQL Exception need no be imported into the from the java.sql.*package.

`` Connection conn = null; try { String url = "<jdbc_mys ql_localhost>"; String user = "root"; String password = "<pas sword>"; conn = DriverManager.ge tConnection(url, user, password); } catch(SQLException e) { System.out.println(e.ge tMessage()); } finally { try{ if(conn ! null) conn.close() } catch(SQLException ex){ System.out.println(ex.g etMessage()) } } ```	The getConnection() method is required to get the Connection Object.

Download MySQL Cheat Sheet from Here (https://drive.google.com/file/d/1rmgDqThL-sD3sEQCrLuzVJh2meQoydaO/view)

Conclusion

Today, MySQL is the most reliable and largely used database in the market. MySQL 8.0 is the latest version available now. It has designed improvements for Database Administrators and developers to develop and deploy high-end applications on highly powerful frameworks and hardware platforms. You can download MySQL 8.0 from the following link: https://www.mysql.com/downloads/ (https://www.mysql.com/downloads/). Also, find more information from its official site: https://www.mysql.com/ (https://www.mysql.com/).

People are also reading:

- MySQL Create Database (https://hackr.io/blog/mysql-create-database)
- MySQL vs MongoDB (https://hackr.io/blog/mongodb-vs-mysql)
- PostgreSQL vs MySQL (https://hackr.io/blog/postgresql-vs-mysql)
- MariaDB vs MySQL (https://hackr.io/blog/mariadb-vs-mysql)
- Linux Cheat Sheet (https://hackr.io/blog/linux-cheat-sheet)
- GIT Cheat Sheet (https://hackr.io/blog/git-cheat-sheet)