MySQL Commands

MySQL uses commands to communicate with the MySQL database by creating queries with data and performing specific tasks and functions. The commands are instructions coded into SQL (structured query language) statements. To write a query requires a set of predefined code that is understandable to the database.

MySQL supports all SQL Standard types of data in several categories including **Numeric**, **Date and Time**, **String**, and **Spatial** data types. The string data types include **Character string** and **Byte string**. MySQL also implements spatial extensions as a subset of SQL with **Geometry Types** environment following the *Open Geospatial Consortium (OGC)* specification.

MySQL Cheat Sheet

Below are some of the most commonly used MySQL commands and statements that help users work with MySQL more easily and effectively. In this article, we present briefly the most commonly used commands – including MySQL command-line client commands – and the commands for working with databases, tables, indexes, views, triggers, procedures, and functions.

MySQL command-line client Commands

Below is a list of MySQL command-line client commands:

```
mysql -u [username] -p;      # Connect to MySQL server

mysql -u [username] -p [database];   # Connect to MySQL Server

exit;                                # Exit mysql command-line client

mysqldump -u [username] -p [database] > data_backup.sql; # Export data
using mysqldump tool

mysql> system clear;  # Clear MySQL screen console for Linux
```

The MySQL command-line client commands are available on Linux for clearing the MySQL screen console window, and there is no client command available on Windows OS.

MySQL Commands for Working with Databases

Below are MySQL commands used for working with databases:

```
CREATE DATABASE [IF NOT EXISTS] database_name; # Create a database in the
server

SHOW DATABASE; # Show all available databases

USE database_name; # Use a database with a specified name

DROP DATABASE [IF EXISTS] database_name; # Drop a database with a specified
name
```

MySQL Commands for Working with Tables

Here are MySQL commands for working with tables in a database:

```
CREATE TABLE [IF NOT EXISTS] table_name(column_list,...); # Create a new
table

SHOW TABLES; # Show all tables in the database

DROP TABLE [IF EXISTS] table_name; # Drop a table from the database
```

Commonly Used MySQL Commands

Below is a list of the most commonly used MySQL commands for database developers and database administrators using MySQL databases:

ALTER

```
ALTER TABLE table_name ADD [COLUMN] column_name;

ALTER TABLE table_name DROP [COLUMN] column_name;

ALTER TABLE table_name MODIFY column_name type;

ALTER TABLE table_name MODIFY column_name type NOT NULL ...;

ALTER TABLE table_name CHANGE old_column_name new_column_name type;
ALTER TABLE table_name CHANGE old_column_name new_column_name type NOT NULL
...;

ALTER TABLE table_name MODIFY column_name type FIRST;

ALTER TABLE table_name MODIFY column_name type AFTER another_column;

ALTER TABLE table_name CHANGE old_column_name new_column_name type FIRST;

ALTER TABLE table_name CHANGE old_column_name new_column_name type AFTER
another_column;

ALTER TABLE table_name ALTER column_name SET DEFAULT ...;

ALTER TABLE table_name ALTER column_name DROP DEFAULT;

ALTER TABLE table_name ADD new_column_name type;

ALTER TABLE table_name ADD new_column_name type FIRST;

ALTER TABLE table_name ADD new_column_name type AFTER another_column;

ALTER TABLE table_name ADD INDEX [name](column, ...);

ALTER TABLE table_name ADD PRIMARY KEY (column_name,...);

ALTER TABLE table_name DROP PRIMARY KEY;
```

SELECT

```
SELECT * FROM table_name;

SELECT * FROM table1, table2, …;

SELECT column_name FROM table_name;

SELECT column1, column2, ... FROM table_name;

SELECT column1, column2, ... FROM table1, table2, …;

SELECT select_list FROM table_name WHERE condition;

SELECT select_list FROM table GROUP BY column1, column2, ...;
SELECT select_list FROM table GROUP BY column_name HAVING condition;

SELECT COUNT(*) FROM table_name;

SELECT DISTINCT (column_name) FROM    table_name;

SELECT select_list FROM table ORDER BY column_name;

SELECT select_list FROM table ORDER BY column1 ASC [DESC], column2 ASC
[DESC];

SELECT column_name AS alias_name, expression AS alias, ... FROM table_name;

SELECT select_list FROM table_name WHERE column LIKE '%pattern%';

SELECT select_list FROM table_name WHERE column RLIKE 'regular_expression';
```

SELECT – JOIN

```
SELECT select_list FROM table1 INNER JOIN table2 ON condition;

SELECT select_list FROM table1 LEFT JOIN table2 ON condition;

SELECT select_list FROM table1 RIGHT JOIN table2 ON condition;

SELECT select_list FROM table1 CROSS JOIN table2;
```

DESCRIBE

```
DESCRIBE table_name;

DESCRIBE table_name column_name;
```

INSERT INTO

```
INSERT INTO table (column_list) VALUES(value_list);

INSERT INTO table (column_list) VALUES(list1), (list2), ...;
```

UPDATE

```
UPDATE table_name SET column1 = value1, ...;

UPDATE table_name SET column_1 = value_1, ... WHERE condition;

UPDATE table1, table2 INNER JOIN table1 ON table1.column1 = table2.column2
SET column1 = value1, WHERE condition;
```

DELETE

```
DELETE FROM table_name;

DELETE FROM table_name WHERE condition;

DELETE table1, table2 FROM table1 INNER JOIN table2 ON table1.column1=
table2.column2 WHERE condition;
```

INDEX

```
CREATE INDEX index_name ON table_name (column,...);

DROP INDEX index_name;

CREATE UNIQUE INDEX index_name ON table_name (column,...);
```

VIEW

```
CREATE VIEW [IF NOT EXISTS] view_name AS  select_statement;

CREATE VIEW [IF NOT EXISTS] view_name AS select_statement WITH CHECK
OPTION;

CREATE OR REPLACE view_name AS select_statement;

DROP VIEW [IF EXISTS] view_name;

DROP VIEW [IF EXISTS] view1, view2, ...;

RENAME TABLE view_name TO new_view_name;
```

```
SHOW FULL TABLES [{FROM | IN } database_name] WHERE table_type = 'VIEW';
```

TRIGGER

```
CREATE TRIGGER trigger_name {BEFORE | AFTER} {INSERT | UPDATE| DELETE } ON
table_name FOR EACH ROW trigger_body;

SHOW TRIGGERS [{FROM | IN} database_name] [LIKE 'pattern' | WHERE
search_condition];

DROP TRIGGER [IF EXISTS] trigger_name;
```

PROCEDURE

```
DELIMITER $$ CREATE PROCEDURE procedure_name (parameter_list) BEGIN body;
END $$ DELIMITER;

DROP PROCEDURE [IF EXISTS] procedure_name;

SHOW PROCEDURE STATUS [LIKE 'pattern' | WHERE search_condition];
```

FUNCTION

```
DELIMITER $$ CREATE FUNCTION function_name(parameter_list) RETURNS datatype
[NOT] DETERMINISTIC BEGIN -- statements END $$ DELIMITER;

DROP FUNCTION [IF EXISTS] function_name;

SHOW FUNCTION STATUS [LIKE 'pattern' | WHERE search_condition];
```

Users and Privileges

```
CREATE USER 'user'@'localhost';

GRANT ALL PRIVILEGES ON base.* TO 'user'@'localhost' IDENTIFIED BY
'password';

GRANT SELECT, INSERT, DELETE ON base.* TO 'user'@'localhost' IDENTIFIED BY
'password';
```

```
REVOKE ALL PRIVILEGES ON base.* FROM 'user'@'host';

REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'user'@'host';

FLUSH PRIVILEGES;

SET PASSWORD = PASSWORD('new_pass');

SET PASSWORD FOR 'user'@'host' = PASSWORD('new_pass');

SET PASSWORD = OLD_PASSWORD('new_pass');

DROP USER 'user'@'host';
```