Bahae Eddine HALIM

MySQL cheatsheet

The SQL cheat sheet provides you with the most commonly used SQL statements for your reference.

# Getting Started

Connect MySQL

```
mysql -u <user> -p

mysql [db_name]

mysql -h <host> -P <port> -u <user> -p [db_name]

mysql -h <host> -u <user> -p [db_name]
```

Backups

Create a backup

```
mysqldump -u user -p db_name > db.sql
```

Export db without schema

```
mysqldump -u user -p db_name --no-data=true --add-drop-table=false
> db.sql
```

Restore a backup

```
mysql -u user -p db_name < db.sql
```

Commons

Database	
`CREATE DATABASE db ;`	Create database
`SHOW DATABASES;`	List databases
`USE db;`	Switch to db
`CONNECT db ;`	Switch to db
`DROP DATABASE db;`	Delete db
Table	
`SHOW TABLES;`	List tables for current db
`SHOW FIELDS FROM t;`	List fields for a table
`DESC t;`	Show table structure
`SHOW CREATE TABLE t;`	Show create table sql
`TRUNCATE TABLE t;`	Remove all data in a table
`DROP TABLE t;`	Delete table
Proccess	
`show processlist;`	List processes
`kill pid;`	kill process
Other	
`exit` or `\q`	Exit MySQL session

# MySQL Examples

Managing tables

Create a new table with three columns

```
CREATE TABLE t (
     id    INT,
     name  VARCHAR DEFAULT NOT NULL,
     price INT DEFAULT 0
     PRIMARY KEY(id)
);
```

Delete the table from the database

```
DROP TABLE t ;
```

Add a new column to the table

```
ALTER TABLE t ADD column;
```

Drop column c from the table

```
ALTER TABLE t DROP COLUMN c ;
```

Add a constraint

Querying data from a table

Query data in columns c1, c2 from a table

```
SELECT c1, c2 FROM t
```

Query all rows and columns from a table

```
SELECT * FROM t
```

Query data and filter rows with a condition

```
SELECT c1, c2 FROM t
WHERE condition
```

Query distinct rows from a table

```
SELECT DISTINCT c1 FROM t
WHERE condition
```

Sort the result set in ascending or descending order

```
SELECT c1, c2 FROM t
```

Querying from multiple tables

Inner join t1 and t2

```
SELECT c1, c2
FROM t1
INNER JOIN t2 ON condition
```

Left join t1 and t1

```
SELECT c1, c2
FROM t1
LEFT JOIN t2 ON condition
```

Right join t1 and t2

```
SELECT c1, c2
FROM t1
RIGHT JOIN t2 ON condition
```

Perform full outer join

```
SELECT c1, c2
FROM t1
```

```
ALTER TABLE t ADD constraint;
```

Drop a constraint

```
ALTER TABLE t DROP constraint;
```

Rename a table from t1 to t2

```
ALTER TABLE t1 RENAME TO t2;
```

Rename column c1 to c2

```
ALTER TABLE t1 RENAME c1 TO c2 ;
```

Remove all data in a table

```
TRUNCATE TABLE t;
```

Using SQL constraints

Set c1 and c2 as a primary key

```
CREATE TABLE t(
    c1 INT, c2 INT, c3 VARCHAR,
    PRIMARY KEY (c1,c2)
);
```

Set c2 column as a foreign key

```
CREATE TABLE t1(
    c1 INT PRIMARY KEY,
    c2 INT,
    FOREIGN KEY (c2) REFERENCES t2(c2)
);
```

Make the values in c1 and c2 unique

```
CREATE TABLE t(
    c1 INT, c1 INT,
    UNIQUE(c2,c3)
);
```

Ensure c1 > 0 and values in c1 >= c2

```
CREATE TABLE t(
  c1 INT, c2 INT,
  CHECK(c1> 0 AND c1 >= c2)
);
```

Set values in c2 column not NULL

```
CREATE TABLE t(
     c1 INT PRIMARY KEY,
     c2 VARCHAR NOT NULL
);
```

Managing Views

Create a new view that consists of c1 and c2

```
CREATE VIEW v(c1,c2)
AS
SELECT c1, c2
FROM t;
```

Create a new view with check option

```
CREATE VIEW v(c1,c2)
AS
```

```
SELECT c1, c2 FROM t
ORDER BY c1 ASC [DESC]
```

Skip offset of rows and return the next n rows

```
SELECT c1, c2 FROM t
ORDER BY c1
LIMIT n OFFSET offset
```

Group rows using an aggregate function

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1
```

Filter groups using HAVING clause

```
SELECT c1, aggregate(c2)
FROM t
GROUP BY c1
HAVING condition
```

Modifying Data

Insert one row into a table

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

Insert multiple rows into a table

```
INSERT INTO t(column_list)
VALUES (value_list),
       (value_list), …;
```

Insert rows from t2 into t1

```
INSERT INTO t1(column_list)
SELECT column_list
FROM t2;
```

Update new value in the column c1 for all rows

```
UPDATE t
SET c1 = new_value;
```

Update values in the column c1, c2 that match the condition

```
UPDATE t
SET c1 = new_value,
        c2 = new_value
WHERE condition;
```

Delete all data in a table

```
DELETE FROM t;
```

Delete subset of rows in a table

```
DELETE FROM t
WHERE condition;
```

Managing triggers

Create or modify a trigger

```
CREATE OR MODIFY TRIGGER trigger_name
WHEN EVENT
ON table_name TRIGGER_TYPE
EXECUTE stored_procedure;
```

WHEN	
BEFORE	invoke before the event occurs
AFTER	invoke after the event occurs

```
FULL OUTER JOIN t2 ON condition
```

Produce a Cartesian product of rows in tables

```
SELECT c1, c2
FROM t1
CROSS JOIN t2
```

Another way to perform cross join

```
SELECT c1, c2
FROM t1, t2
```

Join t1 to itself using INNER JOIN clause

```
SELECT c1, c2
FROM t1 A
INNER JOIN t1 B ON condition
```

Using SQL Operators Combine rows from two queries

```
SELECT c1, c2 FROM t1
UNION [ALL]
SELECT c1, c2 FROM t2
```

Return the intersection of two queries

```
SELECT c1, c2 FROM t1
INTERSECT
SELECT c1, c2 FROM t2
```

Subtract a result set from another result set

```
SELECT c1, c2 FROM t1
MINUS
SELECT c1, c2 FROM t2
```

Query rows using pattern matching %, _

```
SELECT c1, c2 FROM t1
WHERE c1 [NOT] LIKE pattern
```

Query rows in a list

```
SELECT c1, c2 FROM t
WHERE c1 [NOT] IN value_list
```

Query rows between two values

```
SELECT c1, c2 FROM t
WHERE  c1 BETWEEN low AND high
```

Check if values in a table is NULL or not

```
SELECT c1, c2 FROM t
WHERE  c1 IS [NOT] NULL
```

Managing indexes

Create an index on c1 and c2 of the t table

```
CREATE INDEX idx_name
ON t(c1,c2);
```

Create a unique index on c3, c4 of the t table

```
CREATE UNIQUE INDEX idx_name
ON t(c3,c4)
```

```
AS
SELECT c1, c2
FROM t;
WITH [CASCADED | LOCAL] CHECK OPTION;
```

Create a recursive view

```
CREATE RECURSIVE VIEW v
AS
select-statement -- anchor part
UNION [ALL]
select-statement; -- recursive part
```

Create a temporary view

```
CREATE TEMPORARY VIEW v
AS
SELECT c1, c2
FROM t;
```

Delete a view

```
DROP VIEW view_name;
```

AFTER

EVENT	
INSERT	invoke for INSERT
	invoke after the event occurs
UPDATE	invoke for UPDATE
DELETE	invoke for DELETE

TRIGGER_TYPE
FOR EACH ROW
FOR EACH STATEMENT

Drop an index

```
DROP INDEX idx_name;
```

# MySQL Data Types

Strings

CHAR	String (0 - 255)
VARCHAR	String (0 - 255)
TINYTEXT	String (0 - 255)
TEXT	String (0 - 65535)
BLOB	String (0 - 65535)
MEDIUMTEXT	String (0 - 16777215)
MEDIUMBLOB	String (0 - 16777215)
LONGTEXT	String (0 - 4294967295)
LONGBLOB	String (0 - 4294967295)
ENUM	One of preset options
SET	Selection of preset options

Date & time

DATE	yyyy-MM-dd
TIME	hh:mm:ss
DATETIME	yyyy-MM-dd hh:mm:ss
TIMESTAMP	yyyy-MM-dd hh:mm:ss
YEAR	yyyy

Numeric

TINYINT x	Integer (-128 to 127)
SMALLINT x	Integer (-32768 to 32767)
MEDIUMINT x	Integer (-8388608 to 8388607)
INT x	Integer (-2147483648 to 2147483647)
BIGINT x	Integer (-9223372036854775808 to 9223372036854775807)
FLOAT	Decimal (precise to 23 digits)
DOUBLE	Decimal (24 to 53 digits)
DECIMAL	"DOUBLE" stored as string

# MySQL Functions & Operators

Strings

- ASCII()
- BIN()
- BIT_LENGTH()
- CHAR()
- CHARACTER_LENGTH()
- CHAR_LENGTH()
- CONCAT()
- CONCAT_WS()
- ELT()
- EXPORT_SET()
- FIELD()
- FIND_IN_SET()
- FORMAT()
- FROM_BASE64()
- HEX()
- INSERT()
- INSTR()
- LCASE()
- LEFT()
- LENGTH()
- LIKE
- LOAD_FILE()
- LOCATE()
- LOWER()
- LPAD()
- LTRIM()

Date and Time

- ADDDATE()
- ADDTIME()
- CONVERT_TZ()
- CURDATE()
- CURRENT_DATE()
- CURRENT_TIME()
- CURRENT_TIMESTAMP()
- CURTIME()
- DATE()
- DATE_ADD()
- DATE_FORMAT()
- DATE_SUB()
- DATEDIFF()
- DAY()
- DAYNAME()
- DAYOFMONTH()
- DAYOFWEEK()
- DAYOFYEAR()
- EXTRACT()
- FROM_DAYS()
- FROM_UNIXTIME()
- GET_FORMAT()
- HOUR()
- LAST_DAY
- LOCALTIME()
- LOCALTIMESTAMP()

Numeric

- %, MOD
- *
- +
- -
- -
- /
- ABS()
- ACOS()
- ASIN()
- ATAN()
- ATAN2(), ATAN()
- CEIL()
- CEILING()
- CONV()
- COS()
- COT()
- CRC32()
- DEGREES()
- DIV
- EXP()
- FLOOR()
- LN()
- LOG()
- LOG10()
- LOG2()
- MOD()

- LPAD()
- LTRIM()
- MAKE_SET()
- MATCH
- MID()
- NOT LIKE
- NOT REGEXP
- OCT()
- OCTET_LENGTH()
- ORD()
- POSITION()
- QUOTE()
- REGEXP
- REGEXP_INSTR()
- REGEXP_LIKE()
- REGEXP_REPLACE()
- REGEXP_SUBSTR()
- REPEAT()
- REPLACE()
- REVERSE()
- RIGHT()
- RLIKE
- RPAD()
- RTRIM()
- SOUNDEX()
- SOUNDS LIKE
- SPACE()
- STRCMP()
- SUBSTR()
- SUBSTRING()
- SUBSTRING_INDEX()
- TO_BASE64()
- TRIM()
- UCASE()
- UNHEX()
- UPPER()
- WEIGHT_STRING()

JSON

- ->
- ->>
- JSON_ARRAY()
- JSON_ARRAY_APPEND()
- JSON_ARRAY_INSERT()
- JSON_CONTAINS()
- JSON_CONTAINS_PATH()
- JSON_DEPTH()
- JSON_EXTRACT()
- JSON_INSERT()
- JSON_KEYS()
- JSON_LENGTH()
- JSON_MERGE() (deprecated)
- JSON_MERGE_PATCH()
- JSON_MERGE_PRESERVE()
- JSON_OBJECT()
- JSON_OVERLAPS() (introduced 8.0.17)
- JSON_PRETTY()
- JSON_QUOTE()
- JSON_REMOVE()
- JSON_REPLACE()
- JSON_SCHEMA_VALID() (introduced 8.0.17)
- JSON_SCHEMA_VALIDATION_REPORT() (introduced 8.0.17)
- JSON_SEARCH()

- LOCALTIME()
- LOCALTIMESTAMP()
- MAKEDATE()
- MAKETIME()
- MICROSECOND()
- MINUTE()
- MONTH()
- MONTHNAME()
- NOW()
- PERIOD_ADD()
- PERIOD_DIFF()
- QUARTER()
- SEC_TO_TIME()
- SECOND()
- STR_TO_DATE()
- SUBDATE()
- SUBTIME()
- SYSDATE()
- TIME()
- TIME_FORMAT()
- TIME_TO_SEC()
- TIMEDIFF()
- TIMESTAMP()
- TIMESTAMPADD()
- TIMESTAMPDIFF()
- TO_DAYS()
- TO_SECONDS()
- UNIX_TIMESTAMP()
- UTC_DATE()
- UTC_TIME()
- UTC_TIMESTAMP()
- WEEK()
- WEEKDAY()
- WEEKOFYEAR()
- YEAR()
- YEARWEEK()
- GET_FORMAT()

Cast

- BINARY
- CAST()
- CONVERT()

Information

- BENCHMARK()
- CHARSET()
- COERCIBILITY()
- COLLATION()
- CONNECTION_ID()
- CURRENT_ROLE()
- CURRENT_USER()
- DATABASE()
- FOUND_ROWS()
- ICU_VERSION()
- LAST_INSERT_ID()
- ROLES_GRAPHML()
- ROW_COUNT()
- SCHEMA()
- SESSION_USER()
- SYSTEM_USER()
- USER()
- VERSION()

Locking

- GET_LOCK()
- IS_FREE_LOCK()
- IS_USED_LOCK()
- RELEASE_ALL_LOCKS()
- RELEASE_LOCK()

Miscellaneous

- ANY_VALUE()
- BIN_TO_UUID()

- LOG2()
- MOD()
- PI()
- POW()
- POWER()
- RADIANS()
- RAND()
- ROUND()
- SIGN()
- SIN()
- SQRT()
- TAN()
- TRUNCATE()

Aggregate

- AVG()
- BIT_AND()
- BIT_OR()
- BIT_XOR()
- COUNT()
- COUNT(DISTINCT)
- GROUP_CONCAT()
- JSON_ARRAYAGG()
- JSON_OBJECTAGG()
- MAX()
- MIN()
- STD()
- STDDEV()
- STDDEV_POP()
- STDDEV_SAMP()
- SUM()
- VAR_POP()
- VAR_SAMP()
- VARIANCE()

Flow Control

- CASE
- IF()
- IFNULL()
- NULLIF()

Encryption and Compression

- AES_DECRYPT()
- AES_ENCRYPT()
- COMPRESS()
- MD5()
- RANDOM_BYTES()
- SHA1(), SHA()
- SHA2()
- STATEMENT_DIGEST()
- STATEMENT_DIGEST_TEXT()
- UNCOMPRESS()
- UNCOMPRESSED_LENGTH()
- VALIDATE_PASSWORD_STRENGTH()

Bit

- &
- >>
- <<
- ^
- BIT_COUNT()
- |
- ~

- JSON_SET()
- JSON_STORAGE_FREE()
- JSON_STORAGE_SIZE()
- JSON_TABLE()
- JSON_TYPE()
- JSON_UNQUOTE()
- JSON_VALID()
- JSON_VALUE() (introduced 8.0.21)
- MEMBER OF() (introduced 8.0.17)

- DEFAULT()
- GROUPING()
- INET_ATON()
- INET_NTOA()
- INET6_ATON()
- INET6_NTOA()
- IS_IPV4()
- IS_IPV4_COMPAT()
- IS_IPV4_MAPPED()
- IS_IPV6()
- IS_UUID()
- MASTER_POS_WAIT()

NAME_CONST()

SLEEP()

UUID()

UUID_SHORT()

UUID_TO_BIN()

VALUES()