MySQL Cheatsheet by Example

Data Types

BIGINT(x)	integer	+/- 9,223,372,036,854,775,807
BLOB	string	up to 65,535 characters
CHAR(x)	string	up to 255 characters
DATE	date	format: YY-MM-DD
DATETIME	date	format: YY-MM-DD HH:MM:SS
DECIMAL(x,d)	double precision stored as string	
DOUBLE (x,d)	double precision floating number	
ENUM('Y','N')	string value is restricted to list	
FLOAT	floating point decimal	
INT(x)	integer	+/- 2,147,483,647
LONGBLOB	string	up to 4,294,967,295 characters
LONGTEXT	string	up to 4,294,967,295 characters
MEDIUMBLOB	string	up to 16,777,215 characters
MEDIUMINT(x)	integer	+/- 8,388,608
MEDIUMTEXT	string	up to 16,777,215 characters
SET	string	can hold any number of strings
SMALLINT(x)	integer	+/- 32,767
TEXT	string	up to 65,535 characters
TIME	time	format: HH:MM:SS
TIMESTAMP	time	format: YY-MM-DD HH:MM:SS
TINYBLOB	string	up to 255 characters
TINYINT(x)	integer	+/- 127
TINYTEXT	string	up to 255 characters
VARCHAR(x)	string	designated length up to 255 chars
YEAR	date	4-digit year – 1901 to 2155

* Can declare as UNSIGNED, which starts range at 0. SMALLINT range would therefore be 0 to 65,536
x is an integer value representing the maximum display width or number of digits displayed
d represents the number of decimal places

Data Column Attributes

AUTO_INCREMENT	next numeric value auto assigned
DEFAULT	sets default value for column
NOT NULL	null not allowed
NULL	null allowed
PRIMARY KEY	assigns primary key
UNSIGNED	starts range at 0

Create a New Table

Sample code shows how to use various data type declarations when creating a table.

```
CREATE TABLE products (
item_id             VARCHAR(50)                 NOT NULL,
retailer_sku        VARCHAR(50)                 NOT NULL,
item_name           VARCHAR(20)                 NOT NULL,
item_description    TEXT                        NOT NULL,
status_id           TINYINY(2) UNSIGNED         DEFAULT '0',
datestamp           DATE                        DEFAULT '0000-00-00',
active              ENUM('Y','N')               DEFAULT 'Y',
PRIMARY KEY (item_id),
KEY (retailer_sku),
FULLTEXT (item_name, item_description)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 ;
```

Add a Column

```
ALTER TABLE table1 ADD COLUMN col_5 MEDIUMTEXT
```

Change a Column Type

```
ALTER TABLE table1 CHANGE col_3 col_3 TEXT
```

Remove a Column

```
ALTER TABLE table1 DROP COLUMN col_4
```

Rename a Column

```
ALTER TABLE table1 CHANGE col_3 col_4 TEXT
```

Rename a TABLE

```
ALTER TABLE table1 RENAME TO table2
```

Add an Index

```
CREATE INDEX indexname ADD INDEX (col_1)
```

Optimize a Table

This can speed up database performance; especially if there have been a lot of deletes and changes made to a database. You can list any number of tables in a database and they will be optimized. This re-indexes the tables.

```
$sql = "OPTIMIZE TABLE table1, table2";
mysql_query($sql) or die(mysql_error());
```

Setting Up a Database Connection

```
$dbName = "";                       database name
$dbHost = "localhost";              localhost or a hostname
$dbUser = "";                       database user name
$dbPW = "";                         database user password

// as separate statements
$dbHandle = mysql_connect($dbHost,$dbUser,$dbPW)
  or die("Could not connect to MySQL database");
$dbSelect = mysql_select_db($dbName,$dbHandle)
  or die(mysql_error());

// combined into a single compound statement
mysql_select_db($dbName,
mysql_connect($dbHost,$dbUser,$dbPW))
 or die(mysql_error() );
```

3 Ways to Insert a New Row

The basic INSERT statement:

```
INSERT INTO products (item_id, retailer_sku, item_name,
item_description, status_id, timestamp,active)
VALUES ('A123','ZM8849','Fillrod',' 'A rod used to check fluid volume
of a tank',1,'2008-08-31','Y');
```

When you have all the values you can skip column names:

```
INSERT INTO products VALUES ('A123', 'ZM8849','Fillrod','A rod
used to check fluid volume of a tank',1,'2008-08-31','Y');
```

The following is the most flexible INSERT method. Can be used when only a few (with NOT NULL or defaults set) or all values need to be inserted. Easier to troubleshoot because column names are matched to values.

```
INSERT INTO products SET
item_id = 'A123',
retailer_sku = 'ZM8849',
item_name = 'Fillrod',
item_description = 'A rod used to check fluid volume of a tank',
status_id = 1,
datestamp = '2008-08-31',
active = 'Y';
```

Updating an Existing Row

Column name-value pairs can be used in any order. Can also be used when only a few columns need to be updated. You must know a key or column value in order to update one of more records.

```
UPDATE products SET
retailer_sku = 'ZM8849',
item_name = 'Fillrod',
item_description = 'A rod used to check fluid volume of a tank',
status_id = 1,
datestamp = '2008-08-31',
active = 'Y',
WHERE item_id = 'A123';
```

Simple Natural Language Full-Text Search
Works in MySQL 3.23.23 and newer version
First, add a FULLTEXT index to an existing table.

```
ALTER TABLE products ADD FULLTEXT (name, description);
```

Use MATCH – AGAINST syntax for searching the index. You are matching indexed columns against a search phrase. The MATCH columns must be exactly the same as the columns used when the FULLTEXT index was defined.

```
SELECT * FROM products WHERE
MATCH ( name, description ) AGAINST ('$searchPhrase');
```

Simple Boolean Full-Text Search
Works for MySQL 4.0 and newer versions. A Boolean search adds operators that alter the behavior of the search.
First, add a FULLTEXT index to an existing table. Same as above.

```
ALTER TABLE products ADD FULLTEXT (name, description);
```

Use MATCH-AGAINST IN BOOLEAN MODE.

```
SELECT * FROM products WHERE
MATCH ( name, description )
AGAINST ('$searchPhrase' IN BOOLEAN MODE);
```

Boolean Mode Operators:

```
+    when preceding a word, that word must be present in row
-    when preceding a word, that word must not be present in row
~    when preceding a word, decreases the word's contribution value
> < changes a word's contribution value.  > increases, < decreases
" "  when phrase is surrounded by quotes, a literal search for that
      phrase is performed
( )  used to group words into sub-expressions
*    a wildcard character appended to a word
```

Operator Use Examples:

```
drill press      returns rows that contain at least one of the words
"drill press"    returns rows that contain the exact phrase
+drill +press    returns rows that contain both words
+drill -press    returns rows that contain drill, but not press
drill*           returns rows with words drill, driller, drilling, etc.
```

Method to Loop Through Rows

```
$sql = "SELECT col_1, col_2 FROM table1";
$result = mysql_query($sql);
while ($row = mysql_fetch_array($result))
{
  echo $row['col_1']." ".$row['col_2']."<br>";
}
```

Commonly Used MySQL Functions

```
mysql_close              closes a connection

mysql_connect            sets up connection to database user

mysql_error              returns error message for the most recently
                         invoked API function that failed

mysql_fetch_array        fetch a result row as an array

mysql_num_rows           reports number of rows returned in a query

mysql_real_escape_string  escapes special characters in a string for
                         use in a SQL statement

mysql_select_db          sets up connection to a database
```

Sample SQL Queries
To select all columns and rows:

```
SELECT * FROM table1;
```

Returns total number of rows in table:

```
SELECT COUNT(*) FROM table1;
```

Returns number of rows with col_1 = Q:

```
SELECT COUNT(*) FROM table1 WHERE col_1 = 'Q';
```

Returns number of rows returned in query:

```
$sql = "SELECT col_1 FROM table1 WHERE col_2 = 'Y';
$results = mysql_query($sql);
$numRows = mysql_num_rows($results);
```

To calculate the next integer value for a column:

```
SELECT MAX(col_1)+1 AS col_1 FROM table1
```

To extract a single value from a table:

```
SELECT col_2 FROM table1 WHERE col_1 = '$someValue';
```

To extract unique values:

```
SELECT DISTINCT col_1 FROM table1;
```

To set an order for output:

```
SELECT * from table1 ORDER BY col_2
SELECT * from table1 ORDER BY col_2 DESC;
```

To limit the number of rows returned:

```
SELECT col_1, col_2
FROM table1 ORDER BY col_1
LIMIT 0,10;
```

Left join – returns all rows in table1
fills in null for missing values

```
SELECTp.product_name, r.retailer_name
FROM products ALIAS p LEFT JOIN retailers ALIAS r.
ON p.retailer_id = r.retailer_id
WHERE p.category_id = $categoryID.
ORDER BY p.product_name;
```

3-way join – returns only rows that match:

```
SELECT * FROM table1, table2, table3
WHERE table1.col_1 = table2.col_2
AND table2.col_2 = table3.col_2;
```

3-way left join – fills in null for missing columns:

```
SELECT * FROM (tbl1 left join tbl2 on tbl1.col_2 = tbl2.col_2)
LEFT JOIN tbl3 on tbl2.col_3 = tbl3.col_3;
```

To pull a randomly selected record:

```
SELECT col_1, col_2
FROM table1
ORDER BY RAND() LIMIT 1;
```

Returns rows within a range for column with DATE type

```
SELECT last_name, first_name, death FROM citizens
WHERE death >= "1970-01-01"
AND death < "1980-01-01";
```

MySQL One-Liners
Some of these examples are wrapped to a second line, but can be on one line.

Establishes a MySQL connection and selects the database. You can either plug in the database name, host, username and password or assign the values to variables.

```
mysql_select_db('database1',
mysql_connect('localhost','username','password'));
```

If you are not using auto_increment to establish a unique ID for each row, you can set a unique ID manually. This example grabs the next category number to be used for an INSERT statement. The second line establishes a starting value if the table is empty. You still need to INSERT the value.

```
$catID = mysql_result(mysql_query("SELECT MAX(catID)+1 AS
catID FROM categories"),0);

if ($cat == 0) $catID = 1;
```

Returns the total number of rows in a table and assigns to a variable.

```
$total = mysql_result(mysql_query("SELECT COUNT(*) as Num
FROM items"),0);
```

To extract a single value from a unique record assigns it to a variable.

```
$itemName = mysql_result(mysql_query("SELECT item_name
FROM inventory WHERE item_id='$itemID' "),0);
```

Used to increment a value with a single line of code when you do not know the value. Avoids need to read a row, increment a value and write it back.

```
mysql_query("UPDATE items SET click_count =
LAST_INSERT_ID(click_count + 1) WHERE item_id = '$itemID' ");
```