

# MySQL / MariaDB & MongoDB

Everything about MySQL / MariaDB and MongoDB

- [How to do Everything in MySQL/MariaDB](#)
- [Setup for Remote Access](#)
- [Basic MongoDB Operations](#)

# How to do Everything in MySQL/MariaDB

## Useful Tricks

### Convert a Column to Uppercase

```
UPDATE table_name SET column_name = UPPER( column_name )
```

---

### Show and Change View Definer

```
SHOW FULL TABLES IN database_name WHERE TABLE_TYPE LIKE 'VIEW';  
SHOW CREATE VIEW [view_name];  
ALTER DEFINER = '[username]'@'[host]' VIEW [view_name] AS [select statement];
```

---

### Alter Views

```
ALTER VIEW <view name> AS [view statements]
```

---

## MySQL Datatype for Different Password Hashes

It depends on the hashing algorithm you use. Hashing always produces a result of the same length, regardless of the input. It is typical to represent the binary hash result in text, as a series of hexadecimal digits. Or you can use the [UNHEX()]([http://dev.mysql.com/doc/refman/5.5/en/string-functions.html#function\\_unhex](http://dev.mysql.com/doc/refman/5.5/en/string-functions.html#function_unhex)) function to reduce a string of hex digits by half.

- MD5 generates a 128-bit hash value. You can use CHAR(32) or BINARY(16)

- SHA-1 generates a 160-bit hash value. You can use CHAR(40) or BINARY(20)
- SHA-224 generates a 224-bit hash value. You can use CHAR(56) or BINARY(28)
- SHA-256 generates a 256-bit hash value. You can use CHAR(64) or BINARY(32)
- SHA-384 generates a 384-bit hash value. You can use CHAR(96) or BINARY(48)
- SHA-512 generates a 512-bit hash value. You can use CHAR(128) or BINARY(64)
- BCrypt generates an implementation-dependent 448-bit hash value. [You might need CHAR\(56\), CHAR\(60\), CHAR\(76\), BINARY\(56\) or BINARY\(60\)](#)

NIST recommends using SHA-256 or higher for passwords. Lesser hashing algorithms have their uses, but they are [known to be crackable](#).

You should [salt](#) your passwords before applying the hashing function. Salting a password does not affect the length of the hash result.

---

## Chinese Support in MySQL

Convert entire database to UTF-8: `ALTER DATABASE databasename CHARACTER SET utf8 COLLATE utf8_unicode_ci;`

Convert entire table to UTF-8: `ALTER TABLE tablename CONVERT TO CHARACTER SET utf8 COLLATE utf8_unicode_ci;`

Convert field to UTF-8: `ALTER TABLE tablename MODIFY columnname columndef CHARACTER SET utf8 COLLATE utf8_unicode_ci;`

---

## MySQL Grant Permission

```
Grant all on {dbname}.* to 'Id'@'localhost' identified by 'password'
```

---

## Convert All Table Columns' Charset and Collation

```
ALTER TABLE <table> CONVERT TO CHARACTER SET <charset> COLLATE <collation>;
```

Convert Database to MyISAM

```
#!/bin/sh
```

```
DB=' <DB Name>'
TABLES=$( /opt/lampp/bin/mysql -uroot --skip-column-names -B -D $DB -e 'show tables' )

for T in $TABLES
do
    /opt/lampp/bin/mysql -uroot -D $DB -e "ALTER TABLE $T ENGINE=MYISAM"
done
```

# Solution to Common Problems

## MariaDB Function Error From mysqldump

Use DELIMITER keyword to change end of function delimiter, e.g.:

```
DELIMITER //
CREATE FUNCTION counter () RETURNS INT
BEGIN
    UPDATE counter SET c = c + 1;
    RETURN (SELECT c FROM counter LIMIT 1);
END;
//
CREATE FUNCTION counter2 () RETURNS INT
BEGIN
    UPDATE counter SET c = c + 2;
    RETURN (SELECT c FROM counter LIMIT 1);
END;
//
DELIMITER ;
```

## Error while sending QUERY packet

Change your maxallowedpacket by using one of the following methods:

- In mysql prompt, enter `SET GLOBAL max_allowed_packet=524288000;`
- Set `max_allowed_packet` in `my.ini`

# Setup for Remote Access

## 1. Grant Privileges

1. 

```
GRANT ALL ON <database>. * TO <user>@'%' IDENTIFIED BY '<password>';
```

## 2. Testing Remote Access

1. 

```
mysql -uroot -p -h <host/ip> <database>
```

# Basic MongoDB Operations

## Queries and Indexes

### Display query stats

```
db.<collection>.find({<query conditions>}).explain('executionStats')
```

## Basic Document Operations

### Find all documents in a collection

```
db.<collection>.find({})
```

### Sorting documents

```
db.<collection>.find({}).sort()
```

### Find one document in a collection

```
db.<collection>.findOne({})
```

### Count documents in a collection

```
db.<collection>.count()
```

### Insert a document

```
db.<collection>.insert({fieldA: 'a', fieldB: 'b'})
```

### Updating a document

```
db.<collection>.update({_id: '<document id>'}, {'$set': {'fieldA': 'value'}})
```

### Updating multiple documents

```
db.<collection>.update({<field condition>: '<condition>'}, {'$set': {'fieldA': 'value'}}, {multi
```

## Delete a document

```
db.<collection>.remove({_id: '<document id>' })
```

# Collection Operations

## Remove a collection

```
db.<collection>.drop()
```

## List all collections

```
show collections
```

# Database Operations

## List all databases

```
show databases
```

## Switch to a database

```
use <database>
```