

MONGODB CHEAT SHEET



As part of our MongoDB Guide, we've compiled this cheat sheet of common and not-so-common MongoDB commands.

(This article is part of our [MongoDB Guide](#). Use the right-hand menu to navigate.)

Table of Contents

[Pretty Print](#)

[Create Collection](#)

[Create Indexes](#)

[Create index](#)

[Create sparse index](#)

[Create compound index](#)

[Create geo index](#)

[Create partial index](#)

[Add and Delete Data](#)

[Add one data record](#)

[Add array \(many\) records](#)

[Delete data](#)

[Query MongoDB Documents](#)

[Search by attribute](#)

[Search by geolocation](#)

[Search greater or less than](#)

[Search not equal to](#)

[Search and return only certain fields](#)

[Search by regular expression](#)

[Find by elements in array](#)

[Replication and Sharding](#)

[Enable replication](#)

[Add shard](#)

[Show Server Memory](#)

[Aggregate functions](#)

[MapReduce](#)

Pretty Print

Add pretty() to the end of any search:

```
db.collection.find({}).pretty()
```

Create Collection

```
db.createCollection("<collection name>")
```

Create Indexes

Create index

The format is {attribute : sortOrder} where sort order is -1 (descending) or 1 (ascending).

```
db.cars.createIndex( { make: 1 } )
```

Create sparse index

This does not index documents that do not have this attribute.

```
db.products.createIndex({product:-1},{sparse: true})
```

Create compound index

```
db.cars.createIndex( { make: 1 , model: -1} )
```

Create geo index

```
db.address.createIndex( { "location": "2dsphere"} )
```

Create partial index

This indexes those documents that match the search criteria.

```
db.students.createIndex(  
{ age: 1},  
{ partialFilterExpression: { age: { $gt: 14}}}
```

)

Add and Delete Data

Add one data record

```
db.cars.insert()
```

Add array (many) records

```
db.cars.insertMany()
```

Delete data

```
db.cars.remove({"make": "Chevrolet"})
```

Query MongoDB Documents

Search by attribute

```
db.cars.find({"make": "Mercedes"})
```

Search by geolocation

```
db.address.find ({
  location: {
    $near: {
      $geometry: {
        type: "Point" ,
        coordinates:
      },
      $maxDistance: 4,
      $minDistance: 0
    }
  }
})
```

Search greater or less than

```
db.sales.find({"price": {$gt: 100}})
```

Search not equal to

```
db.cars.find({make: {$not: {$eq: "Mercedes"}}})
```

Search and return only certain fields

This is called projection.

Show only the make field:

```
db.cars.find({make: "Mercedes"}, {vin: -1})
```

Show every field except the make field:

```
db.cars.find({make: "Mercedes"}, {vin: 0})
```

Search by regular expression

You can use the slash or quote marks as a delimiter.

```
db.cars.find({"make": {$regex: /^M.*}/})
```

Case insensitive:

```
db.cars.find({"make": {$regex: /^merc.*/ , $options: "i"}} )
```

Find by elements in array

This matches documents that contain all of these array elements:

```
db.manufacturers.find({"name": {$all: }} )
```

Match on any element in the array:

```
db.manufacturers.find({"name": "Ford"} )
```

Replication and Sharding

Enable replication

Connect to a config server then:

```
rs.initiate()
```

And then show replication status:

```
rs.status()
```

Add shard

Connect to query router (mongos) server. 27018 in this example is the port number of a shard server

```
sh.addShard( "ShardReplSet/172.31.47.43:27018")
```

Show Server Memory

```
db.serverStatus().mem
```

Aggregate functions

This is the same as the SQL `select count(*) from words group by word`. This is the word count program.

```
db.words.aggregate()
```

MapReduce

You can do the same word count function using the map operation:

```
db.words.mapReduce(
function() { emit(this.word,1); },
function(key, values) {return Array.sum(values)}), {
query:{},
} out:"total_matches"
).find()
```