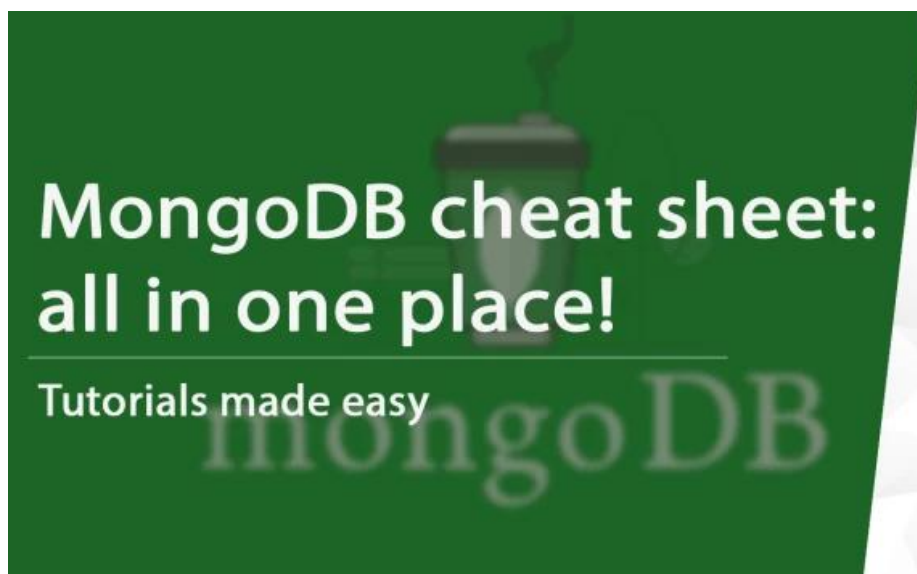# The Ultimate MongoDB Cheat Sheet: Mastering NoSQL in One Place

Maximize your efficiency with our ultimate MongoDB cheat sheet! Covering everything from basic syntax to advanced functions, this resource is designed to help you streamline your NoSQL workflow and become a MongoDB master.

List of content you will read in this article:

MongoDB is a data storage platform that uses a similar data model to JSON. It is hosted on multiple servers and allows users to build quick, real-time applications that access their data. MongoDB is a distributed system which doesn't use any specific database software.

MongoDB provides a wide range of analytical tools for data profiling, load analysis, and monitoring. MongoDB can use these tools to perform data mining, Big Data, and Online Analytic Processing. MongoDB's Replica Management feature makes scaling up your system easy and cost-effective.

MongoDB is a viable choice for enterprise and internet application developers seeking flexibility and scalability. It's particularly helpful for programmers of many disciplines to create scalable applications through agile methods.

This post provides a list of the most popular and widely used MongoDB commands that will come in handy for beginners. We covered almost all the useful commands, but before we start, let's review MongoDB's features, pros, and cons.

Features of MongoDB
Before going through the MongoDB cheat sheet, we identify some key features of MongoDB:

- Schema-less Database: In a MongoDB database, any collection can hold multiple documents. These documents can contain different numbers of fields, sizes, and content. A schema-less database doesn't have any prescribed structure. It's possible to have multiple documents with similar content as MongoDB databases support. This leads to a wide range of options for database creation because each variation can be different from the next.
- Document Oriented: Data stored in MongoDB tables, or RDBMS, is instead organized in documents. Each document contains a key-value pair in fields instead of rows and columns. This makes data stored in documents more accessible and flexible than in RDBMS tables. Every document also has an individual ID that can't be changed.
- Indexing: The database employs two primary indexes to index every field in a MongoDB document. This speeds up searching and compression significantly. If indexes aren't used, documents are searched using a specified query. This process takes longer and doesn't perform as well.
- Scalability: MongoDB increases its size by dividing data across multiple servers. Called horizontal scaling, this technique partitions the information into smaller sections and distributes them across many physical servers. This allows them to access more resources as new machines are added to the database.
- Replication: The MongoDB database offers high availability and redundancy through replication. It creates copies of the data and sends them to multiple servers so that if one fails, it can retrieve the data from a different one.
- Aggregation: This feature of MongoDB allows performing operations on grouped data and showing single or computed results like the SQL GROUPBY clause.
- High Performance: In addition to its high performance, MongoDB also has features such as indexing, replication, and scalability, which make it better than other databases.
- File storage: GridFS can replicate and balance data on multiple machines. It uses MongoDB as a file system. A file will be stored as a grid file system in MongoDB.
- Sharding: MongoDB's sharding allows users to split up a collection among many shards (machines) and let it grow beyond resource limitations. Sharding is the process of splitting up data among machines. With data distributed across our machines, we may be able to store and handle more data without upgrading our

Pros and Cons of MongoDB

All NoSQL databases have some advantages and disadvantages. In this part, we'll take a quick look at the pros and cons of MongoDB.

Pros of MongoDB

1. A document can have as many fields as the user wishes in MongoDB, which allows users to store any data.
2. Most programming languages support documents as native data types, which allows the addition of data to documents.
3. Like Hadoop Distributed File System (HDFS), MongoDB has its file system called GridFS. As a result, MongoDB's 16 MB per document BSON size limit is mainly used to store files that exceed the size limit of the file system.
4. MongoDB is compatible with Hadoop, Spark, and other data processing frameworks like SQL.

Cons of MongoDB

1. The automatic failover strategy is not instantaneous - it may take up to a minute to switch from one master node to another. However, it promises continuity despite not being instantaneous. Despite the fast speed of writing data to the database, the amount of data is limited. So, data must be recorded on the master, and it takes time to add new information.
2. Data consistency may be affected by MongoDB's lack of foreign-key integrity.
3. A database administrator must configure network connections to MongoDB databases since user authentication isn't enabled by default in MongoDB databases.
4. Also, ransomware has forced database administrators to enable this setting following attacks.

Now that you know the features, pros, and cons of MongoDB, you can have the cheat sheet of MongoDB commands to use when you need it.

Getting started with MongoDB

Before anything, you need to install MongoDB Shell mongoose. After installing, log in and enter MongoDB Shell mongoose. To do this, use the following command:

```
$ mongosh
```

//OR

```
$ mongosh "mongodb://localhost:27017"
```

Database Commands

To view all databases:

```
show dbs
```

To create a new database or switch databases:

```
use dbName
```

To view the current Database:

```
db
```

To drop a Database:

```
db.dropDatabase()
```

To search in a Database:

```
db.comments.find({lang:'Python'})
```

To backup MongoDB database:

```
$ mongodump -d oldDatabaseName path
```

To restore the MongoDB database:

```
$ mongorestore -d newDatabaseName path/oldDatabaseName.bson
```

Collection Commands
To show Collections:

```
show collections
```

To create a collection named 'clo':

```
db.createCollection('clo')
```

To delete a collection named 'clo':

```
db.clo.drop()
```

To create a statistical structure:

```
db.docx.stats()
```

(Note: you can use this command to copy a pointer into a user-specified memory location)

To find out how much total storage space in bytes is allocated to the document for document storage:

```
db.docx.storageSize()
```

To report the total size used by the indexes in a collection document:

```
db.docx.totalIndexSize()
```

To calculate the total size in bytes of the set data plus the size of each index:

```
db.docx.totalSize()
```

To list a Collection's Records:

```
db.collectionname.find()
```

To list Records with Matching Values of Specific Fields:

```
collectionname.find({"field2": "secondmatching value"})
```

To Match multiple Values:

```
collectionname.find({"field2": "second  matching  value",  "field3":
"thirdmatchingvalue"})
```

To find a Single Record:

```
collectionname.findOne({"field2": "content"})
```

Row Commands
To show all Rows of a Collection:

```
db.comments.find()
```

To show all Rows of a Collection (Prettified):

```
db.comments.find().pretty()
```

To find the first row matching the object:

```
db.comments.findOne({name: 'Harry'})
```

To insert only one Row:

```
db.comments.insert({
    'name': 'Harry',
    'lang': 'JavaScript',
    'member_since': 5
 })
```

To insert multiple Rows:

```
db.comments.insertMany([{
    'name': 'Harry',
    'lang': 'JavaScript',
    'member_since': 5
    },
    {'name': 'Rohan',
    'lang': 'Python',
    'member_since': 3
    },
    {'name': 'Lovish',
    'lang': 'Java',
    'member_since': 4
}])
```

To limit the number of rows in the output:

```
db.comments.find().limit(2)
```

To count the number of rows in the output:

```
db.comments.find().count()
```

To update a row:

```
db.comments.updateOne({name: 'Shubham'},

{$set: {'name': 'Harry',

    'lang': 'JavaScript',

    'member_since': 51

}}, {upsert: true})
```

To delete Row:

```
db.comments.remove({name: 'Harry'})
```

To sort rows, you can use multiple methods:

```
 # asc

 db.docx.find().sort({ title: 5 }).pretty()

# desc

 db.docx.find().sort({ title: -5}).pretty()
```

To count the number of Rows:

```
db.docx.find().count()
```

To limit the number of rows:

```
db.docx.find().limit(5).pretty()
```

Document commands
To find documents using operators:

```
$gt greater than db.docx.find({class:{$gt:'T'}
```

```
$gte greater than equals db.docx.find({class:{$gt:'T'}
```

```
$lt lesser than db.docx.find({class:{$lt:'T'}
```

```
$lte lesser than equals db.docx.find({class:{$lte:'T'}
```

```
$exists does an attribute exist or not db.docx.find({class:{$gt:'T'}
```

```
$regex    Matching    pattern    in    pearl-style
db.docx.find({name:{$regex:'^USS\\sE'}})
```

```
$type search by type of an element db.docx.find({name : {$type:4}})
```

To delete a document:

```
db.docx.deleteOne({"_id" : 6})
```

(Note: you can use both deleteOne and deleteMany commands for this purpose.
Because both of them take a filter document as their first parameter.)

To update one document:

```
db.docx.updateOne({"_id": 2}, {$set: {"title": 'revised title'}})
```

(Note: as their first parameter, updateOne and updateMany take a filter document, and as the second parameter, they take a modifier document that describes changes to make.)

To update multiple documents:

```
db.docx.update({"category": "Information"}, {$set: {"category": 'Sports'}})
```

Increment Operator of Mongodb:

```
db.comments.update({name: 'Rohan'},

{$inc:{

    member_since: 2

}})
```

Rename Operator of Mongodb:

```
db.comments.update({name: 'Rohan'},

{$rename:{

    member_since: 'member'

}})
```

Less than/Greater than/ Less than or Eq/Greater than or Eq Commands:

```
db.comments.find({member_since: {$lt: 90}})

db.comments.find({member_since: {$lte: 90}})

db.comments.find({member_since: {$gt: 90}})

db.comments.find({member_since: {$gte: 90}})
```

Index commands
To list indexes:

```
db.docx.getIndexes()
```

To create Index:

```
db.docx.createIndex({"name": 2})                // single field index

db.docx.createIndex({"name": 2, "date": 2})     // compound index

db.docx.createIndex({foo: "text", bar: "text"}) // text index

db.docx.createIndex({"$**": "text"})            // wildcard text index

db.docx.createIndex({"userMetadata.$**": 1})    // wildcard index
```

To drop an index:

```
db.docx.dropIndex("name_3")
```

To hide or unhide indexes:

```
db.docx.hideIndex("name_3")
```

```
db.docx.unhideIndex("name_3")
```

To create a compound index:

```
db.docx.ensureIndex({name : 3, operator : 1, class : 0})
```

To drop a compound index:

```
db.docx.dropIndex({name : 3, operator : 1, class : 0})
```

Aggregation commands
 To sum up values:

```
db.docx.aggregate([{$group : {_id : "$operator", num_docx : {$sum :
"$value"}}}])
```

To calculate average values:

```
db.docx.aggregate([{$group : {_id : "$operator", num_docx : {$avg :
"$value"}}}])
```

To find min or max values:

```
db.docx.aggregate([{$group : {_id : "$operator", num_docx : {$min :
"$value"}}}])
```

To push values to a result array:

```
db.docx.aggregate([{$group : {_id : "$operator", classes : {$push:
"$value"}}}])
```

To push values to a result array without duplicates:

```
db.docx.aggregate([{$group : {_id : "$operator", classes : {$addToSet :
"$value"}}}])
```

To get the first or last document:

```
db.docx.aggregate([{$group : {_id : "$operator", last_class : {$last :
"$value"}}}])
```

Managing commands

To check the MongoDB version:

```
db.version()
```

To check if MongoDB is running:

```
$ pgrep -fa -- -D | grep mongo
```

To restart MongoDB on Linux:

```
db.shutdownServer()
```

```
$ sudo systemctl start mongodb
```

To restart MongoDB on Mac OSX:

```
$ brew services stop mongodb-community@yourMongodbVersionNumber
```

To restart MongoDB on Windows:

```
Winkey + R

Type "services.msc"

Search "MondoDb"

Click "restart"
```

To change the user password:

```
db.changeUserPassword(username, NewPassword)
```

To exit from mongosh:

```
quit()
```

To list MongoDB roles:

```
db.getRoles(
    {
        rolesInfo: 1
    }
)
```

To create a MongoDB user:

```
db.createUser(
{
user: "userName",
pwd: "userPassword",
roles:[{role: "roleName" , db:"databaseName"}]
}
)
```

To delete MongoDB user:

```
db.dropUser(userName)
```

Conclusion
One of the world's most widely used and popular document databases is MongoDB. Full-text search, data aggregation, and many other features made MongoDB a powerful data storage platform. In this post, we identified MongoDB, its features, pros, and cons. We also provided a useful cheat sheet that will come in handy for different purposes. Now you're ready to head out with what we covered in this post.

People also read:

- [Docker Cheat Sheet](#)
- [MySQL cheat sheet](#)
- [SQL Cheat Sheet](#)
- [Python Cheat Sheet](#)