# **MongoDB: The database for modern applications**

"MongoDB is a general purpose, document-based, distributed database built for modern application developers and for the cloud era .... MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time" - MongoDB

### **Used by millions of developers to power the world's most innovative products and services**

****

Let's download and install MongoDB (<https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/>):

brew tap mongodb/brew

brew install mongodb-community@5.0

Once install is complete, we can run Mongo with the following command:

brew services start mongodb-community@5.0

Great! Now let's drop into the mongo shell:

mongo

We are now interacting with the [mongo shell](https://docs.mongodb.com/manual/mongo/). The mongo shell is an interactive JavaScript interface to MongoDB. We can use the mongo shell to perform CRUD operations as well as MongoDB administrative operations.

## **Working with MongoDB**

Once we're connected to a MongoDB server if we'd like to know what [database](https://docs.mongodb.com/manual/reference/glossary/#term-database) we're connected to we can issue the following command:

db

db will return the current [database](https://docs.mongodb.com/manual/reference/glossary/#term-database) we're connected to.

To switch [databases](https://docs.mongodb.com/manual/reference/glossary/#term-database) we can issue the following command:

use <database>

Note! You can switch to a non-existing [database](https://docs.mongodb.com/manual/reference/glossary/#term-database) and when you first store data in that [database](https://docs.mongodb.com/manual/reference/glossary/#term-database), the [database](https://docs.mongodb.com/manual/reference/glossary/#term-database) gets created.

To list all [databases](https://docs.mongodb.com/manual/reference/glossary/#term-database):

show dbs

## [**MongoDB CRUD Operations**](https://docs.mongodb.com/manual/crud/)

We are going to learn how to perform all MongoDB CRUD operations using the mongo shell. Let's start by creating a users [database](https://docs.mongodb.com/manual/reference/glossary/#term-database):

use myGreatUserDatabase

### [**Insert a Single Document**](https://docs.mongodb.com/manual/tutorial/insert-documents/)

Let's now create a new users [collection](https://docs.mongodb.com/manual/reference/glossary/#term-collection) and insert a new user!

db.users.insertOne( { name: "Benny", age: 28, status: "active" } )

* db represents our current [database](https://docs.mongodb.com/manual/reference/glossary/#term-database) (which in our case should be myGreatUserDatabase)
* users is the new [collection](https://docs.mongodb.com/manual/reference/glossary/#term-collection) we created

Remember: A [collection](https://docs.mongodb.com/manual/reference/glossary/#term-collection) is a grouping of MongoDB documents

* { name: "Benny", age: 28, status: "active" } is the [document](https://docs.mongodb.com/manual/reference/glossary/#term-document)

We can also [insert many documents](https://docs.mongodb.com/manual/reference/method/db.collection.insertMany/#db.collection.insertMany) at once!

db.users.insertMany([

{ name: "Claire", age: 28, status: "active" },

{ name: "Joey", age: 28, status: "active" },

{ name: "Abe", age: 22, status: "pending" },

{ name: "Sunny", age: 23, status: "pending" },

{ name: "Lizzy", age: 28, status: "active" },

{ name: "Julie", age: 21, status: "active" }

])

### [**Read Operations**](https://docs.mongodb.com/manual/crud/#read-operations)

Let's say we would like to retrieve the user [document](https://docs.mongodb.com/manual/reference/glossary/#term-document) that we just inserted into our users [collection](https://docs.mongodb.com/manual/reference/glossary/#term-collection). How do we do this?

db.users.find( { name: "Benny" } )

You can see we specify the constraint that we are looking for, so the constraint is: find all users in the users [collection](https://docs.mongodb.com/manual/reference/glossary/#term-collection) where name is "Benny". We can pass in additional constraints if we would like or we can remove all constraints if we'd like to see all users in the users [collection](https://docs.mongodb.com/manual/reference/glossary/#term-collection):

db.users.find( {} )

Note: Once a [document](https://docs.mongodb.com/manual/reference/glossary/#term-document) is created in a Mongo database, [Mongo assigns each document a unique identifier](https://docs.mongodb.com/manual/reference/glossary/#term-id), hence the \_id you see. The \_id field is immutable

What if we would like to only return user names and omit everything else e.g. age, status? We can do this with a [projection](https://docs.mongodb.com/manual/reference/method/db.collection.find/#find-projection). [Projection](https://docs.mongodb.com/manual/reference/method/db.collection.find/#find-projection) is the second argument in the query command below:

db.users.find( { age: 28 }, { name: 1 } )

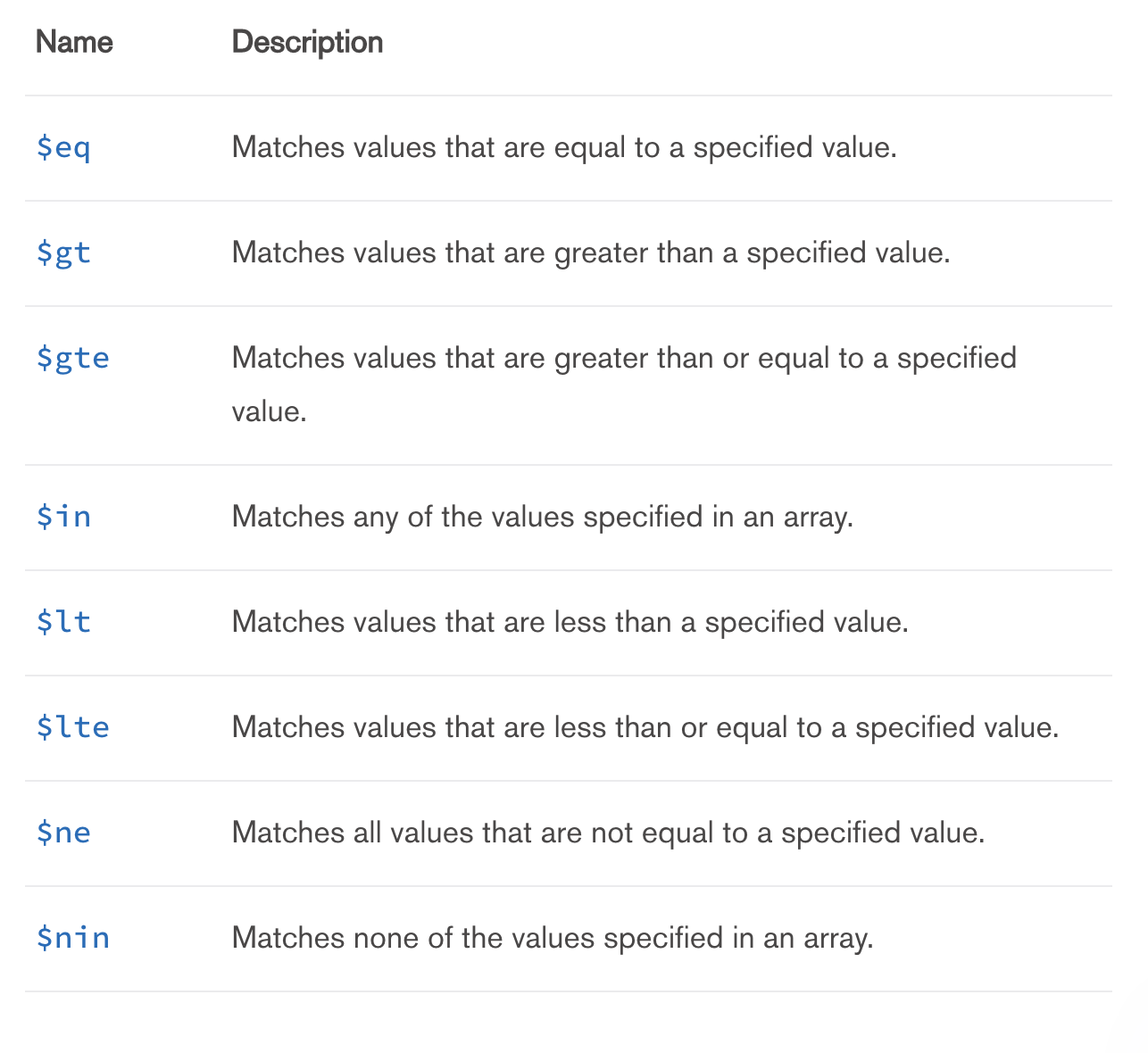
The above will return all users within the users [collection](https://docs.mongodb.com/manual/reference/glossary/#term-collection) in the myGreatUserDatabase that have age 28 and will only return their name

or maybe we would like name and age but don't care for status:

db.users.find( { age: 28 }, { name: 1, age: 1 } )

### [**Comparison Query Operators**](https://docs.mongodb.com/manual/reference/operator/query-comparison/#query-selectors-comparison)

We have a host of comparison query operators available for us to use:



db.users.find( { age: { $gt: 25 } } )

Specify AND Conditions

Find all users who have status of pending and age less than 25:

db.users.find( { status: "pending", age: { $lt: 25 } } )

Specify OR Conditions

Find all users with status of pending or age less than 25.

db.users.find( { $or: [ { status: "pending" }, { age: { $lt: 25 } } ] } )

### [**Update a Single Document**](https://docs.mongodb.com/manual/tutorial/update-documents/)

Inorder to update a document in a [collection](https://docs.mongodb.com/manual/reference/glossary/#term-collection) in MongoDB, we need to first find the [document](https://docs.mongodb.com/manual/reference/glossary/#term-document) we would like to update, then using the $set operator, we specify what field we would like to update and to what. And finally, we use the $currentDate operator to update the value of the lastModified field to the current date (if the lastModified field does not exist then one will be created).

db.users.updateOne( { name: "Benny" }, { $set: { name: "Ben", age: 29 }, $currentDate: { lastModified: true } } )

[Update Multiple Documents](https://docs.mongodb.com/manual/reference/method/db.collection.updateMany/#db.collection.updateMany)

db.users.updateMany( { "age": { $lt: 27 } }, { $set: { name: "Ben", age: 29 }, $currentDate: { lastModified: true } } )

[Replace a Document](https://docs.mongodb.com/manual/reference/method/db.collection.replaceOne/#db.collection.replaceOne)

You can replace an entire document. Just keep in mind the \_id of a document is immutable - you cannot change it, however, you can change everything else.

db.users.replaceOne( { name: "Ben" }, { name: "Benny", age: 39, status: "active" } )

### [**Delete a Document**](https://docs.mongodb.com/manual/tutorial/remove-documents)

To delete one document that matches a condition:

db.users.deleteOne( { name: "Joey" } )

[Delete All Documents that Match a Condition](https://docs.mongodb.com/manual/reference/method/db.collection.deleteMany/#db.collection.deleteMany)

db.users.deleteMany( { status: "pending" } )

To delete all documents in a [collection](https://docs.mongodb.com/manual/reference/glossary/#term-collection):

db.users.deleteMany({})

## **Resources**

* <https://docs.mongodb.com/manual/reference/glossary>