

MongoDB Cheat Sheet 2022

a) Check `monosh` Version

```
mongosh --version
```

b) Start the Mongo Shell

```
mongosh "YOUR_CONNECTION_STRING" --username YOUR_USER_NAME
```

c) Show Current Database

```
Db
```

d) Show All Databases

```
show dbs
```

e) Create Or Switch Database

```
use blog
```

```
db.stats() // return a document that reports on the state of the current database.
```

f) Drop Database

```
db.dropDatabase()
```

```
db.myCollection.drop() //Drop a collection
```

g) Create Collection

```
db.createCollection('posts') //Create Collection randomly without Schema Validation.
```

```
db.createCollection('posts', {validator: {$jsonSchema: {bsonType: "object", required: ["title", "text", "creator"], properties} }}) // Create Collection using \$jsonSchema Validation
```

NB> To add document validation to an existing collection, use [collMod](#) command with the [validator](#) option, ie;

```
db.runCommand({collMod: 'posts', validator: {$jsonSchema: {bsonType: "object", required: ["title", "text", "creator"], properties} }})
```

h) Show Collections

```
show collections
```

i) Insert Document

```
db.posts.insertOne({  
  title: 'Post 1',  
  body: 'Body of post.',  
  category: 'News',  
  likes: 1,  
  tags: ['news', 'events'],  
  date: Date()  
})
```

j) Insert Multiple Documents

>> When inserting multiple documents, we use an array of BSON documents(objects)

```
db.posts.insertMany([  
  {  
    title: 'Post 2',  
    body: 'Body of post.',  
    category: 'Event',  
    likes: 2,  
    tags: ['news', 'events'],  
    date: Date()  
  },  
  {  
    title: 'Post 3',  
    body: 'Body of post.',
```

```

        category: 'Tech',
        likes: 3,
        tags: ['news', 'events'],
        date: Date()
    },
    {
        title: 'Post 4',
        body: 'Body of post.',
        category: 'Event',
        likes: 4,
        tags: ['news', 'events'],
        date: Date()
    },
    {
        title: 'Post 5',
        body: 'Body of post.',
        category: 'News',
        likes: 5,
        tags: ['news', 'events'],
        date: Date()
    }
])

```

k) Find All Documents

`db.posts.find()` //Displays the first 20 documents according to MongoDB Shell.

`db.posts.find().pretty()` //Display data in a good format.

`Db.posts.find().toArray()` //Displays all the documents in an array.

l) Find Documents with Query

```
db.posts.find({ category: 'News' })
```

m) Sort Documents

Ascending

```
db.posts.find().sort({ title: 1 })
```

Descending

```
db.posts.find().sort({ title: -1 })
```

db.collection.aggregate() >> aggregation operations process the data records/documents and return computed results. It collects values from various documents and groups them together and then performs different types of operations on that grouped data like sum, average, minimum, maximum, etc to return a computed result. It is similar to the aggregate function of SQL.

n) Count Documents

```
db.posts.find().count()
```

```
db.posts.find({ category: 'news' }).count()
```

o) Limit Documents

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

p) Chaining

```
db.posts.find().limit(2).sort({ title: 1 })
```

q) Find One Document

```
db.posts.findOne({ likes: { $gt: 3 } })
```

r) Update Document

```
db.posts.updateOne({ title: 'Post 1' },  
{
```

```
$set: {  
  category: 'Tech'  
}  
})
```

s) Update Document or Insert if not Found

```
db.posts.updateOne({ title: 'Post 6' },  
{  
  $set: {  
    title: 'Post 6',  
    body: 'Body of post.',  
    category: 'News'  
  }  
},  
{  
  upsert: true  
})
```

t) Increment Field (`\$inc`)

```
db.posts.updateOne({ title: 'Post 1' },  
{  
  $inc: {  
    likes: 2  
  }  
})
```

u) Update Multiple Documents

```
db.posts.updateMany({}, {  
  $inc: {
```

```
    likes: 1
  }
})
```

v) Rename Field

```
db.posts.updateOne({ title: 'Post 2' },
{
  $rename: {
    likes: 'views'
  }
})
```

w) Delete a Document

```
db.posts.deleteOne({ title: 'Post 6' })
```

x) Delete Multiple Documents

```
db.posts.deleteMany({ category: 'Tech' })
```

➤ *To remove document that not matches condition you can do,*

```
db.inventory.deleteMany( { type : { $ne: "food" } } )
```

or ➤ *Delete All from collection except fieldname with:*

```
db.inventory.deleteMany ( { type : { $nin: ["Apple", "Mango"] } } )
```

```
db.collection.deleteMany ( { "fieldName" : { $ne : "value" } })
```

y) Greater & Less Than

```
db.posts.find({ views: { $gt: 2 } })
```

```
db.posts.find({ views: { $gte: 7 } })
```

```
db.posts.find({ views: { $lt: 7 } })
```

```
db.posts.find({ views: { $lte: 7 } })
```

<https://gist.github.com/mourice-oduor/04f188b2ee0c0087e1f86d7756439211>