Cheatography

# MongoDB Cheat Sheet "issr Master" Cheat Sheet

by Hany Samir (hanysamir) via cheatography.com/198431/cs/41986/

## Basinc MongoDB commands

| Commmand | Description | Example |
|---|---|---|
| mongod | start the mongo service | mongod --dbpath /home/mongo --logpath /home/mongo/mongod.log --bind_ip 0.0.0.0 |
| mongosh | command run mongodb shell | mongosh |
| show dbs | list databases | show dbs |
| use db_name | Switch to database db | use company |
| db | show current database name | db |
| show collections | List current database collections | show collections |

## Insert

| syntax | Description | Example |
|---|---|---|
| insertOne (data, options) | insert one document | db.people.insertOne({ "ssn":1, "name":"Ali","age":20}) |
| insertMany([{document1},{document2}]) | insert many documents, square brackets are mandatory | "db.sales.insertMany([ { ""item"": ""abc"", ""price"": 10}, { ""item"": ""xyz"", ""price"": 5 } ])" |

## Delete

| syntax | Description | Example |
|---|---|---|
| deleteOne-(filter, options) | delete the first matching document from collection | db.project.deleteOne( { "pnumber": 1 } ) |
| deleteMany(filter, options) | delete all matching documents from the project collection | db.project.deleteMany( { "plocation": "Stafford" } ) |

## Update

| syntax | Description | Example |
|---|---|---|
| updateOne(-filter, data, options) | update one document | db.sales.updateOne({"item":"abc"}, { "-$set" : {"price": 22} } ) |
| updateMany(filter, data, options) | update matching documents. | db.sales.updateMany ( { ""_id"":{""$-gte"":14, ""$lte"":16} } , { $unset : { ""-price"" : "" ""} } ) |

$set to update the field value
$unset will remove the field

## Index

| syntax | Description | Example |
|---|---|---|
| create-Index( {keys} , {options} ) | Creates indexes on collections. | db.project.createIndex( { "pnumber": 1, "Pname": -1 } , { "name": "pno_name" , "unique": true} ) |
| dropIndex-(index) | Delete index | db.project.dropIndex("pno_name") |
| getInd-exes() | List indexes on collection. | db.project.getIndexes() |

if index name was not specified withing options, the name will be generated automatically.
1 = ascending order
-1 = descending order

## Find

| syntax | Description | Example |
|---|---|---|
| findOne(filter, options) | find first matching document | db.employee.findOne({"dno": 8}, {"_id":0,"fname":1,"lname":1}) |
| find(filter, options) | find all matching documents | db.employee.find({"dno": 8},{"_id":0,"fname":1,"lname":1}) |




Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

# MongoDB Cheat Sheet "issr Master" Cheat Sheet

by Hany Samir (hanysamir) via cheatography.com/198431/cs/41986/

## Query methods

| syntax | Description | Example |
|---|---|---|
| sort({field_name:1 or -1}) | find first matching document | db.sales.find().sort({"price": 1}) |
| limit(n) | Limits the number of documents | db.sales.find().limit(3) |
| skip(n) | Skips over the specified number of documents | db.sales.find().skip(3) |

1 = ascending order
-1 = descending order

## Aggregation Pipeline

| syntax | Description | Example |
|---|---|---|
| $group | The $group stage separates documents into groups according to a "group key". The output is one document for each unique group key. | db.employee.aggregate([ { "$group" : { "_id" : null, "averageSalary": { "$avg": "$salary" } } } ]) |
| **Accumulator Operator** | | |
| $avg | Returns an average of numerical values. | |
| $max | Returns the highest expression value for each group. | |
| $min | Returns the lowest expression value for each group. | |
| $sum | Returns a sum of numerical values. Ignores non-numeric values. | |

## Aggregation Pipeline (cont)

| | | |
|---|---|---|
| $lookup | Performs a left outer join to a collection in the same database to filter in documents from the "joined" collection for processing. The $lookup stage adds a new array field to each input document. The new array field contains the matching documents from the "joined" collection. The $lookup stage passes these reshaped documents to the next stage. | db.orders.aggregate( [ { $lookup: { from: "inventory", localField: "item", foreignField: "sku", as: "inventory_docs" } } ] ) |
| **The $lookup takes a document with these fields:** | | |
| | from | Specifies the collection in the same database to perform the join with. |
| | localField | Specifies the field from the documents input to the $lookup stage. |
| | foreignField | Specifies the field from the documents in the from collection. |
| | as | Specifies the name of the new array field to add to the input documents. |




Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

## Logical operators

| | |
|---|---|
| $and:[-array] | AND operation between all array of expressions |
| $or:[a-rray] | OR operation between all array of expressions |
| $nor:[-array] | All array expressions must fail. |
| $not:Expr | Performs a logical NOT operation on the specified <op-erator-expression> |

## Comparison operators

| syntax | Description |
|---|---|
| $eq | Matches values that are equal to a specified value. |
| $ne | Matches all values that are not equal to a specified value. |
| $gt | Matches values that are greater than a specified value. |
| $gte | Matches values that are greater than or equal to a specified value. |
| $lt | Matches values that are less than a specified value. |
| $lte | Matches values that are less than or equal to a specified value. |
| $in | Matches any of the values specified in an array |
| $nin | Matches none of the values specified in an array. |



Sponsored by **Readable.com**
Measure your website readability!
https://readable.com