MongoDB Sheet

Show All Databases

```
show dbs
```

Exit the mongosh session

```
exit
```

Show Current Database

```
db
```

Create Or Switch Database

```
**use <database name>**
```

```
example: use users
```

Drop

```
db.dropDatabase()
```

Create Collection

```
db.createCollection('users')
```

Show Collections

```
show collections
```

insertOne

```
db.users.insertOne({ name: “mamun” })
```

main data for the database

```
db.users.insertMany([{
  name: "sohan"
},
{
  name: "chaudhuree",
  age: 19,
  address:{street:"uttara sector 4"},
  hobbies:["Running"]
},
{name:"kabir"},
{name:"shahriar"},
{
  name:"jeshan",
  age:26,
  hobbies:["Weight Lifting","Bowling"],
  address:{street:"balighata bazar",city: "joypurhat panchbibi"}
},
{
  name:"habib",
  age:41,
  hobbies:["Swimming","Bowling"],
  address:{street:"savar 435",city: "savar dhaka"}
},
{
  name: "rakib",
  age: null
},
{
  name:"dipto",
  cashIn:100,
  cashOut:200
},
{
  name:"gourab",
  cashIn:500,
  cashOut:800
}
])
```

Get all documents

```
db.users.find()
```

Find all documents that match the filter object

> ***Get all users*** *with the name sohan*

```
db.users.find({ name: “sohan” })
```

> *in order to search from* ***object***

```
db.users.find({"object element in quatation":"search string"})
```

example:

```
db.users.find({ “address.street”: “balighata bazar” })
```

in order to search in array follow the link below

> ***find in an array***

Find all documents but only return the field specified in the select object

> *Get all users with the name chaudhuree but only return their name, age, and _id*

```
db.users.find({ name: “chaudhuree” }, { name: 1, age: 1 })
```

> *Get all users and return all columns except for age*

```
db.users.find({}, { age: 0 })
```

findOne

> *Get the first user with the name chaudhuree*

```
db.users.findOne({ name: “chaudhuree” })
```

countDocuments

> *Get the number of users with the name chaudhuree*

```
db.users.countDocuments({ name: “chaudhuree” })
```

Update

updateOne

> *Update the first user with an age of 20 to the age of 21*

```
db.users.updateOne({ age: 20 }, { $set: { age: 21 } })
```

> *Update the first user with adding the new value of notes*

```
db.users.updateOne({ age: 20 }, { $set: { notes: "work hard" } })
```

updateMany

> *Update all users with an age of 12 by adding 3 to their age*

```
db.users.updateMany({ age: 12 }, { $inc: { age: 3 } })
```

replaceOne

Replace the first document that matches the filter object with the exact object passed as the second parameter. This will **completely overwrite** the entire object and not just update individual fields.

> *Replace the first user with an age of 12 with an object that has the age of 13 as its only field*

```
db.users.replaceOne({ age: 12 }, { age: 13 })
```

Difference between replace and update

With replaceOne() you can only replace the entire document, while updateOne() allows for updating fields.

Since replaceOne() replaces the entire document - fields in the old document not contained in the new will be lost. With updateOne() new fields can be added without losing the fields in the old document.

> *example:*

```
db.users.insertOne({my_test_key3 : 3333})
```

using replace one:

```
db.users.replaceOne({my_test_key3 : 3333},{ my_test_key4 : 4444})
```

results in:

```
{ "_id" : ObjectId("0123456789abcdef01234567"),
my_test_key4 : 4444 }
```

using updateOne:

```
updateOne( {my_test_key3 : 3333} ,
{ $set:{my_test_key4 : 4444}} )
```

results in:

```
{ "_id" : ObjectId("0123456789abcdef01234567"),
my_test_key3 : 3333,
my_test_key : 4444
}
```

Delete

deleteOne

> *Delete the first user with an age of 20*

```
db.users.deleteOne({ age: 20 })
```

deleteMany

> *Delete all users with an age of 12*

```
db.users.deleteMany({ age: 12 })
```

Read Modifiers

sort

> *Get all users sorted by name in alphabetical order and then if any names are the same sort by age in reverse order*

```
db.users.find().sort({ name: 1, age: -1 })
```

limit

> *Only return the first 2 users*

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

skip

> *Skip the first 4 users when returning results. This is great for pagination when combined with limit.*

```
db.users.find().skip(4)
```

Complex Filter Object

$eq

> *Get all users with the name chaudhuree*

```
db.users.find({ name: { $eq: “chaudhuree” } })
```

$ne

> *Get all users with a name other than Kyle*

```
db.users.find({ name: { $ne: “chaudhuree” } })
```

$gt / $gte

> *Get all users with an age greater than 12*

```
db.users.find({ age: { $gt: 12 } })
```

> *Get all users with an age greater than or equal to 15*

```
db.users.find({ age: { $gte: 15 } })
```

$lt / $lte

> *Get all users with an age less than 12*

```
db.users.find({ age: { $lt: 12 } })
```

> *Get all users with an age less than or equal to 15*

```
db.users.find({ age: { $lte: 15 } })
```

$in

> *Get all users with a name of chaudhuree or sohan*

```
db.users.find({ name: { $in: [“chaudhuree”, “sohan”] } })
```

$nin

> *Get all users that do not have the name chaudhuree or sohan*

```
db.users.find({ name: { $nin: [“chaudhuree”, “sohan”] } })
```

$and

> *Get all users that have an age of 12 and the name chaudhuree*

```
db.users.find({ $and: [{ age: 12 }, { name: “Kyle” }] })
```

> *This is an alternative way to do the same thing. Generally you do not need $and.*

```
db.users.find({ age: 12, name: “Kyle” })
```

$or

> *Get all users with a name of chaudhuree or an age of 12*

```
db.users.find({ $or: [{ age: 12 }, { name: “chaudhuree” }] })
```

$not

> *Get all users with a name other than chaudhuree*

```
db.users.find({ name: { $not: { $eq: “chaudhuree” } } })
```

$exists

> *Get all users that have a name field*

```
db.users.find({ name: { $exists: true } })
```

$expr

Do comparisons between different fields

> *Get all users that have a cashIn that is greater than their cashOut*

```
db.users.find({ $expr: { $gt: [“$cashIn”, “$cashOut”] } })
```

> *** note: infront of cashIn and cashOut we have to use **$** sign*

Complex Update Object

Any combination of the below can be use inside an update object to make complex updates

$set

> *Update the name of the first user with the age of 12 to the value Hi*

```
db.users.updateOne({ age: 12 }, { $set: { name: “Hi” } })
```

$inc

> *Add 2 to the age of the first user with the age of 12*

```
db.users.updateOne({ age: 12 }, { $inc: { age: 2 } })
```

$rename

> *Rename the field age to years for all users*

```
db.users.updateMany({}, { $rename: { age: “years” } })
```

$unset

> *Remove the age field from the first user with an age of 12*

```
db.users.updateOne({ age: 12 }, { $unset: { age: “” } })
```

$push

> *Add John to the friends array for all users*

```
db.users.updateMany({}, { $push: { friends: “annie” } })
```

$pull

> *Remove Mike from the friends array for all users*

```
db.users.updateMany({}, { $pull: { friends: “annie” } })
```