Delete

Each of these commands is run on a specific collection

`db.<collectionName>.<command>`

deleteOne

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

Delete the first document that matches the filter object
Delete the first user with an age of 20

deleteMany

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

Delete all documents that matches the filter object
Delete all users with an age of 12

Complex Filter Object

Any combination of the below can be use inside a filter object to make complex queries

$eq

```
db.users.find({ name: { $eq: "Kyle" } })
```

Check for equality
Get all users with the name Kyle

$ne

```
db.users.find({ name: { $ne: "Kyle" } })
```

Check for not equal
Get all users with a name other than Kyle

$gt / $gte

```
db.users.find({ age: { $gt: 12 } })
db.users.find({ age: { $gte: 15 } })
```

Check for greater than and greater than or equal to
Get all users with an age greater than 12
Get all users with an age greater than or equal to 15

$lt / $lte

```
db.users.find({ age: { $lt: 12 } })
db.users.find({ age: { $lte: 15 } })
```

Check for less than and less than or equal to
Get all users with an age less than 12
Get all users with an age less than or equal to 15

$in

```
db.users.find({ name: { $in: ["Kyle", "Mike"] } })
```

Check if a value is one of many values
Get all users with a name of Kyle or Mike

$nin

```
db.users.find({ name: { $nin: ["Kyle", "Mike"] } })
```

Check if a value is none of many values
Get all users that do not have the name Kyle or Mike

$and

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

Check that multiple conditions are all true
Get all users that have an age of 12 and the name Kyle
This is an alternative way to do the same thing. Generally you do not need $and.

$or

```
db.users.find({ $or: [{ age: 12 }, { name: "Kyle" }] })
```

Check that one of multiple conditions is true
Get all users with a name of Kyle or an age of 12

$not

```
db.users.find({ name: { $not: { $eq: "Kyle" } } })
```

Negate the filter inside of $not
Get all users with a name other than Kyle

$exists

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

Check if a field exists
Get all users that have a name field

$expr

```
db.users.find({ $expr: { $gt: ["$balance", "$debt"] } })
```

Do comparisons between different fields
Get all users that have a balance that is greater than their debt

Complex Update Object

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

$set

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

Update only the fields passed to $set. This will not affect any fields not passed to $set.

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

---

$inc

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

Increment the value of the field by the amount given

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

$rename

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

Rename a field

Rename the field age to years for all users

$unset

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

Remove a field

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

$push

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

Add a value to an array field

Add John to the friends array for all users

$pull

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

Remove a value from an array field

Remove Mike from the friends array for all users

Read Modifiers

Any combination of the below can be added to the end of any read operation

sort

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

Sort the results of a find by the given fields

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

---

limit

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

Only return a set number of documents

Only return the first 2 users

---

skip

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

Skip a set number of documents from the beginning

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