# 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

# MongoDB Cheat Sheet

By Web Dev Simplified https://courses.webdevsimplified.com

## Terminology

| | |
|---|---|
| Database | A container for collections. This is the same as a database in SQL and usually each project will have its own database full of different collections. |
| Collection | A grouping of documents inside of a database. This is the same as a table in SQL and usually each type of data (users, posts, products) will have its own collection. |
| Document | A record inside of a collection. This is the same as a row in SQL and usually there will be one document per object in the collection. A document is also essentially just a JSON object. |
| Field | A key value pair within a document. This is the same as a column in SQL. Each document will have some number of fields that contain information such as name, address, hobbies, etc. An important difference between SQL and MongoDB is that a field can contain values such as JSON objects, and arrays instead of just strings, number, booleans, etc. |

## Basic Commands

| | |
|---|---|
| `mongosh` | Open a connection to your local MongoDB instance. All other commands will be run within this mongosh connection. |
| `show dbs` | Show all databases in the current MongoDB instance |
| `use <dbname>`<br>`use myDatabase` | Switch to the database provided by dbname<br>Switch to myDatabase |
| `db` | Show current database name |
| `cls` | Clear the terminal screen |
| `show collections` | Show all collections in the current database |
| `db.dropDatabase()` | Delete the current database |
| `exit` | Exit the mongosh session |