Basic Commands

Start database server

```
$ mongod --dbpath data/db
$ mongo ds137281.mlab.com:37281/db-name -u UN -p PW
```

Import/Export data

```
$ mongoexport -d <db> -c <col> -o ~/data.json
$ mongoimport -d <db> -c <col> --drop -f ~/data.json
```

Start Client

```
$ mongo
```

Show databases *mongo shell*

```
> show dbs
```

Use database named "mydb" *mongo shell*

```
> use mydb
```

Delete database *mongo shell*

```
> db.dropDatabase();
```

Show collections in current database *mongo shell*

```
> show collections
> db.getCollectionNames();
```

CRUD operations

Creating Documents

Insert one or more new documents into a collection

`.insert(<doc or array of docs> [,opts])`

- The *doc* is an object literal

Example:

Lorem ipsum dolor sit amet, consectetur adipiscing elit

```
db.authors.insert({email:'joe@box.com', name:'Joe'})
```

See also: insert(), insertOne(), insertMany()

Reading Documents

Find and filter one or more documents from collection.

`.find(query, [,projection])`

- Use *query object* to filter the selection
- Use *projection* to specify the fields to return

Examples:

Find all documents where `email` is `joe@box.com`

```
db.authors.find( {email:'joe@box.com'} )
```

Find document by _id

```
db.authors.find({
  _id: ObjectId("5963965386d224366e0b9fa5")
})
```

Find all authors where language is French and active is true

```
db.authors.find({
  language: "French",
  active: true
})
```

Find all authors; return only the name and _id fields (aka projection)

```
db.authors.find{ {}, {name: 1})
```

See also: find(), findOne()

Query Selectors

Query selectors provide additional ways to filter data

Find in date range

$gte - greater than or equal to
$lte - less than or equal to

```
db.authors.find({
  createdAt: {
    $gte: ISODate("2017-01-01"),
    $lte: ISODate("2017-12-31")
  }
})
```

Find all authors where number of likes is between 10 and 50

```
db.authors.find({
  likes: { $gt: 10, $lt: 50 }
})
```

Find authors with one of several languages

```
db.authors.find({
  language: { $in: ['French', 'English'] }
})
```

See also: $eq, $gt, $lt, $in, $and, $exists, $elemMatch

Query Results

Database queries return a cursor with zero or more results whose content can only be read once, so you may want to save to variable. You can do things like sort, limit, and offset.

Examples:

Get the 10 most recent authors

```
db.authors.find()
  .sort( {createdAt: -1} )
  .limit(10)
```

Get the next 10 most recent author posts

```
db.authors.find()
  .sort({createdAt: -1})
  .limit(10)
  .skip(10);
```

See also: sort(), limit() ,min(), max(), skip(), count(), map(), forEach()

Updating documents

Update one or more documents matching query

`.update( <query>, update: {}, options: {})`

- *query*: filter determining which documents to update
- *update*: operations to apply to update
- *options*: { upsert: boolean, multi: boolean}
  - *upsert*: if true, then will create a document if no match found
  - *multi*: if true, updates 1 or more documents

Example:

Update email of user with a string of the user's id

```
db.authors.update({
    _id: ObjectId("507c35dd8fada716c89d0013")
  },{
    email: "newaddress@box.com"
});
```

Update ONE document matching query

`.findAndModify({query, sort, update, new, upsert})`

- *query*: filter determining which document to update
- *sort:* if multiple document found, sort to determine which is updated
- *update*: operations to apply to update
- new: if true, returns modified document rather than original
- *upsert*: if true, then will create a document if no match found

Example:

Update email of user with a string of the user's id

```
db.authors.findAndModify(
  query: { name: "Ben" },
  sort: { rating: 1 },
  update: { $inc: { score: 1 } },
  upsert: true
);
```

See also: update(), updateOne(), updateMany(), findOneAndUpdate(), findAndModify(), findOneAndReplace(), replaceOne(),

Deleting Documents

Permanently delete one or more documents from a collection

`.deleteOne( <query>)`
`.deleteMany( <query> )`

- *query*: filter determining which documents to update

Examples:

Delete a single document by id

```
db.authors.deleteOne({
  _id: ObjectID("someObjectIdString")
})
```

Deleting where email is empty

```
db.authors.deleteMany( {email: {$exists: false}} )
```

See also: deleteOne(), deleteMany(), remove(), findOneAndDelete()