

MongoDB

NoSQL Database

Install Homebrew

<https://brew.sh/>



Homebrew

The missing package manager for macOS

Install MongoDB

```
$ brew install mongodb
```

<https://www.mongodb.com/download-center#community>

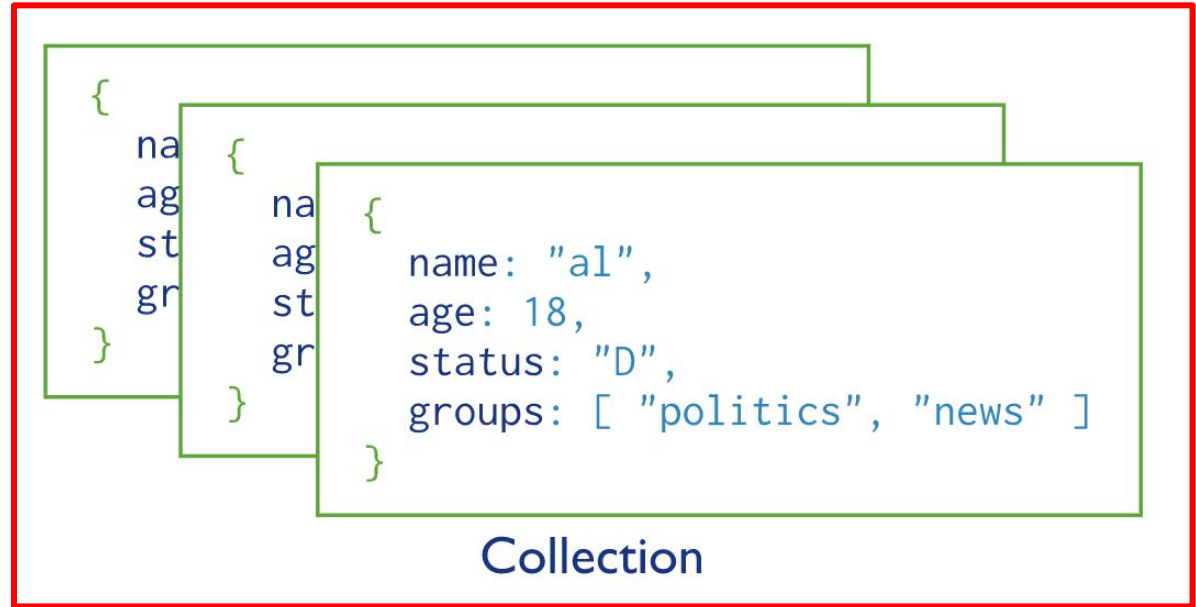


MongoDB Quick Reference

- <https://docs.mongodb.com/manual/reference/mongo-shell/>
- <https://www.opentechguides.com/how-to/article/mongodb/118/mongodb-cheatsheet.html>
- <http://tech.joshegan.com/posts/yr2017/mongodb-cheatsheet>

Database Schema

- Database
- Collections
- Documents



Starting MongoDB

```
$ brew services start mongodb
```

```
alpha [~] : brew services start mongodb
⇒ Tapping homebrew/services
Cloning into '/usr/local/Homebrew/Library/Taps/homebrew/homebrew-services' ...
remote: Counting objects: 14, done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 14 (delta 0), reused 10 (delta 0), pack-reused 0
Unpacking objects: 100% (14/14), done.
Tapped 0 formulae (43 files, 54.8KB)
⇒ Successfully started `mongodb` (label: homebrew.mxcl.mongodb)
alpha [~] : █
```

Verify MongoDB is Running

```
$ ps -ef | grep mongo
```

```
alpha [~] : ps -ef | grep mongo
  501 97217      1   0 10:37AM ??          0:01.18 /usr/local/opt/mongodb/bin/mongod --config /usr/local/etc/mongod.conf
  501 97274 91335   0 10:41AM ttys001    0:00.00 grep mongo
alpha [~] : █
```

Connect to MongoDB

\$ mongo

```
alpha [~] : mongo
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.6.3
Server has startup warnings:
2018-03-15T10:37:53.257-0700 I CONTROL [initandlisten]
2018-03-15T10:37:53.257-0700 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-03-15T10:37:53.257-0700 I CONTROL [initandlisten] **          Read and write access to data and configuration is unrestricted.
2018-03-15T10:37:53.257-0700 I CONTROL [initandlisten]
> █
```


Showing all/current db and selection

Show all databases

```
> show dbs
admin      0.000GB
config     0.000GB
local      0.000GB
> 
```

Change database

```
> use animalfarm
switched to db animalfarm
> 
```

Show all collections in current database

```
> db.getCollectionNames()
[ "dogs" ]
```

Show current database

```
> db
animalfarm
> 
```

Insert and Query data

> insert

> find

```
> db.dogs.insert({name:'fido',age:3,breed:'labrador'})
WriteResult({ "nInserted" : 1 })
> db.dogs.find()
{ "_id" : ObjectId("5aaabe405ba15e7e28267030"), "name" : "fido", "age" : 3, "breed" : "labrador" }
>
```

MongoDB collection functions

```
db.dogs.addIdIfNeeded(
db.dogs.aggregate(
db.dogs.bulkWrite(
db.dogs.constructor
db.dogs.convertToCapped(
db.dogs.convertToSingleObject(
db.dogs.copyTo(
db.dogs.count(
db.dogs.createIndex(
db.dogs.createIndexes(
db.dogs.dataSize(
db.dogs.deleteMany(
db.dogs.deleteOne(
db.dogs.diskStorageStats(
db.dogs.distinct(
db.dogs.drop(
db.dogs.dropIndex(
db.dogs.dropIndexes(
db.dogs.ensureIndex(
db.dogs.exists(
db.dogs.explain(
db.dogs.find(
>

db.dogs.findAndModify(
db.dogs.findOne(
db.dogs.findOneAndDelete(
db.dogs.findOneAndReplace(
db.dogs.findOneAndUpdate(
db.dogs.getCollection(
db.dogs.getDB(
db.dogs.getDiskStorageStats(
db.dogs.getFullName(
db.dogs.getIndexKeys(
db.dogs.getIndexSpecs(
db.dogs.getIndexes(
db.dogs.getIndices(
db.dogs.getMongo(
db.dogs.getName(
db.dogs.getPagesInRAM(
db.dogs.getPlanCache(
db.dogs.getQueryOptions(
db.dogs.getShardDistribution(
db.dogs.getShardVersion(
db.dogs.getSlaveOk(
db.dogs.getSplitKeysForChunks(

db.dogs.getWriteConcern(
db.dogs.group(
db.dogs.groupcmd(
db.dogs.hasOwnProperty
db.dogs.hashAllDocs(
db.dogs.help(
db.dogs.initializeOrderedBulkOp(
db.dogs.initializeUnorderedBulkOp(
db.dogs.insert(
db.dogs.insertMany(
db.dogs.insertOne(
db.dogs.isCapped(
db.dogs.latencyStats(
db.dogs.mapReduce(
db.dogs.pagesInRAM(
db.dogs.propertyIsEnumerable
db.dogs.prototype
db.dogs.reIndex(
db.dogs.remove(
db.dogs.renameCollection(
db.dogs.replaceOne(
db.dogs.runCommand(

db.dogs.runReadCommand(
db.dogs.save(
db.dogs.setSlaveOk(
db.dogs.setWriteConcern(
db.dogs.shellPrint(
db.dogs.stats(
db.dogs.storageSize(
db.dogs.toLocaleString
db.dogs.toString(
db.dogs.tojson(
db.dogs.totalIndexSize(
db.dogs.totalSize(
db.dogs.unsetWriteConcern
db.dogs.update(
db.dogs.updateMany(
db.dogs.updateOne(
db.dogs.validate(
db.dogs.valueOf(
db.dogs.verify(
```

Exit Shell

```
> exit
```

```
> quit()
```

Importing Data

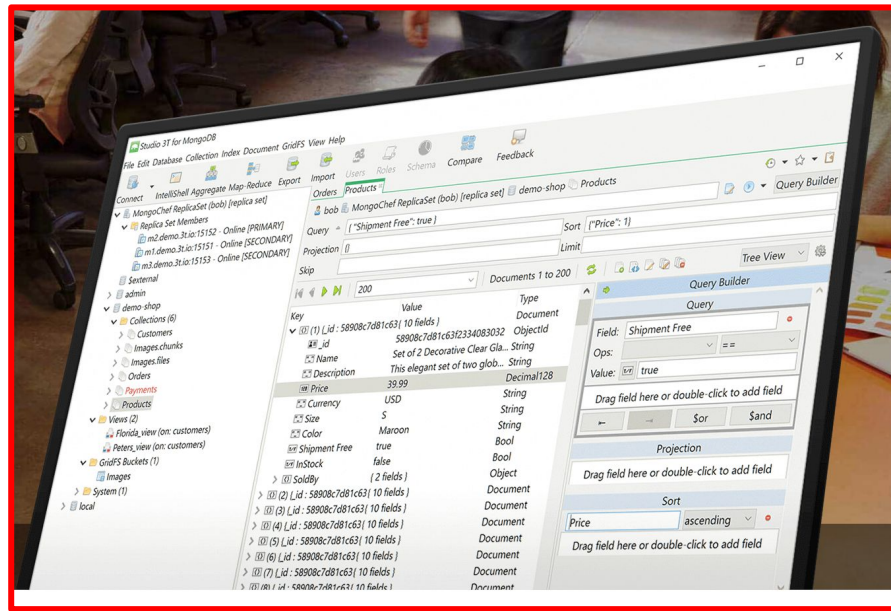
```
$ mongoimport --db test --collection restaurants --drop  
--file ~/Downloads/restaurant.json
```

```
alpha [~/Downloads] : mongoimport --db test --collection restaurants --drop --file ~/Downloads/restaurant.json  
2018-03-15T13:24:29.770-0700    connected to: localhost  
2018-03-15T13:24:29.770-0700    dropping: test.restaurants  
2018-03-15T13:24:30.291-0700    imported 25359 documents  
alpha [~/Downloads] : 
```

MongoDB GUI

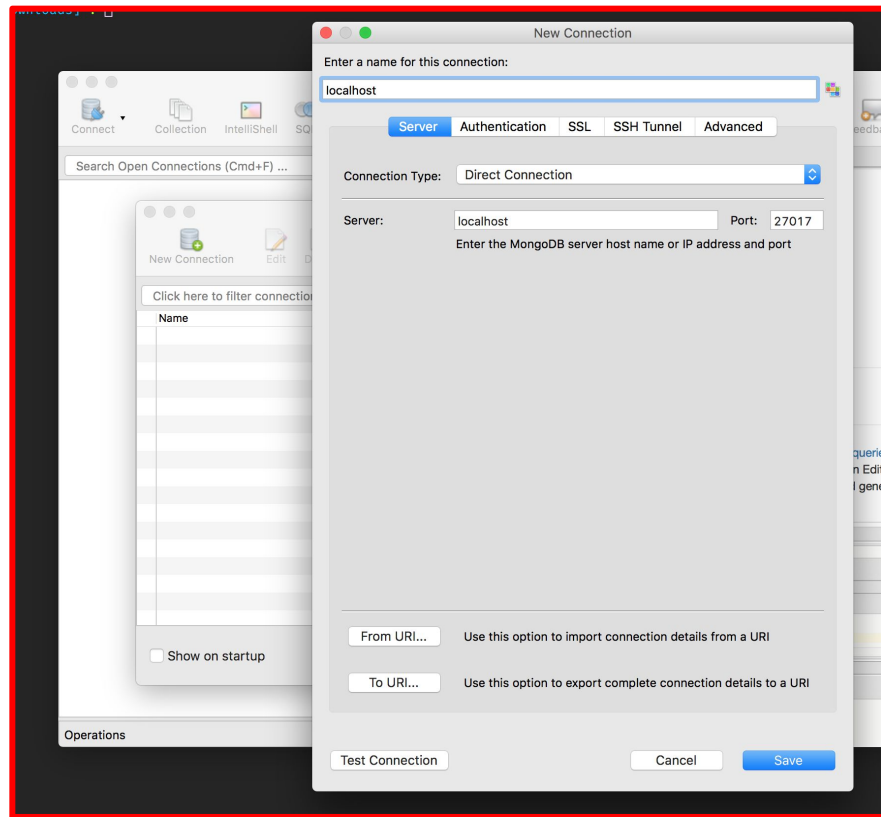
<https://studio3t.com/download-now/>

- It's free for non-commercial use
- Just register with email address



Connect to MongoDB from Studio3T

- Open Studio3T
- Click Connect
- Click New Connection
- Type localhost in top field
- Click Test Connection
- Click Save
- Click Connect
- You should now see your server and all the databases, including your restaurant data.



Viewing the Restaurant data

The screenshot shows the Studio 3T for MongoDB - Non-Commercial License interface. The top toolbar includes icons for Connect, Collection, IntelliShell, SQL, Aggregate, Map-Reduce, Export, Import, Users, Roles, Schema, Compare, and Feedback. The left sidebar shows a tree view of the database structure, with the 'restaurants' collection selected under the 'test' database. The main window displays the IntelliShell: localhost tab, showing the command `db.restaurants.find({})` and its results. The results are displayed in a table with columns for Key, Value, and Type. The table shows 10 documents, each with a unique _id and 7 fields. The bottom status bar indicates '0 items selected' and 'Count Documents' with a value of '0.004s'.

Studio 3T for MongoDB - Non-Commercial License

Connect Collection IntelliShell SQL Aggregate Map-Reduce Export Import Users Roles Schema Compare Feedback

Search Open Connections (Cn) IntelliShell: localhost

Chylds-MacBook-Pro.local (mongod-3.6.3) test

Shell Methods Reference

```
1 db.restaurants.find({})
2 |
```

Find

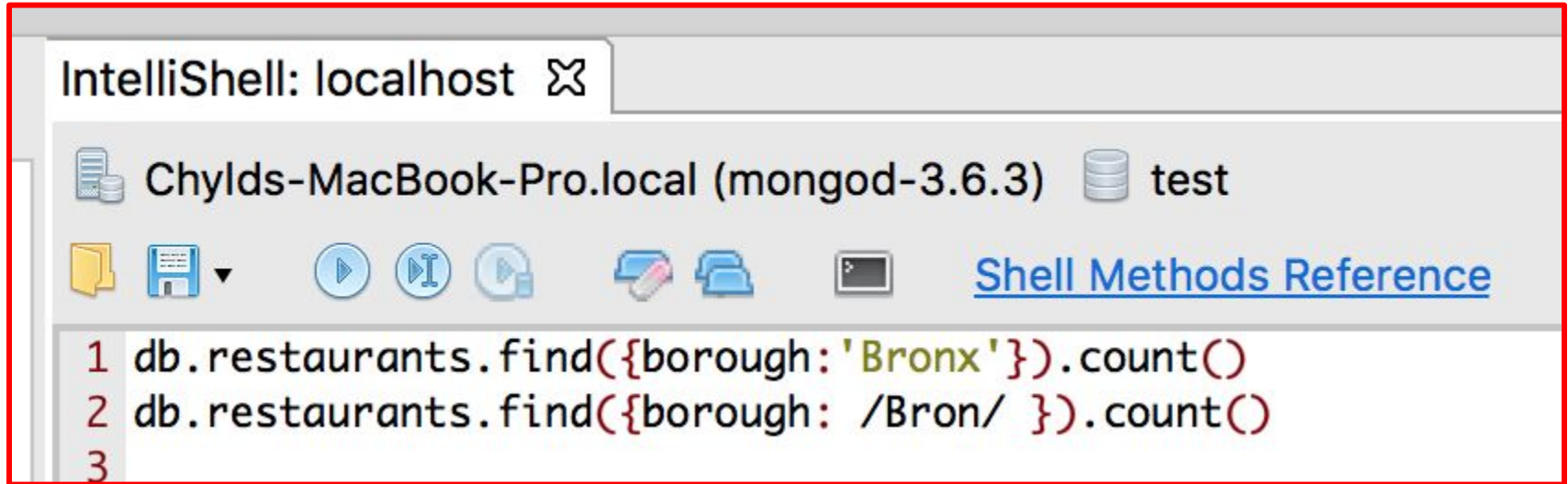
50 Documents 1 to 50 Tree View

Key	Value	Type
(1) {_id : 5aaad67dc7d922f116be7130}	{ 7 fields }	Document
(2) {_id : 5aaad67dc7d922f116be7131}	{ 7 fields }	Document
(3) {_id : 5aaad67dc7d922f116be7132}	{ 7 fields }	Document
(4) {_id : 5aaad67dc7d922f116be7133}	{ 7 fields }	Document
(5) {_id : 5aaad67dc7d922f116be7134}	{ 7 fields }	Document
(6) {_id : 5aaad67dc7d922f116be7135}	{ 7 fields }	Document
(7) {_id : 5aaad67dc7d922f116be7136}	{ 7 fields }	Document
(8) {_id : 5aaad67dc7d922f116be7137}	{ 7 fields }	Document
(9) {_id : 5aaad67dc7d922f116be7138}	{ 7 fields }	Document
(10) {_id : 5aaad67dc7d922f116be7...	{ 7 fields }	Document

Operations 0 items selected Count Documents 0.004s

Querying MongoDB

Below, we search for all the restaurants in the Bronx borough. The first is literal string, the second is a regular expression. Also, counting the results.

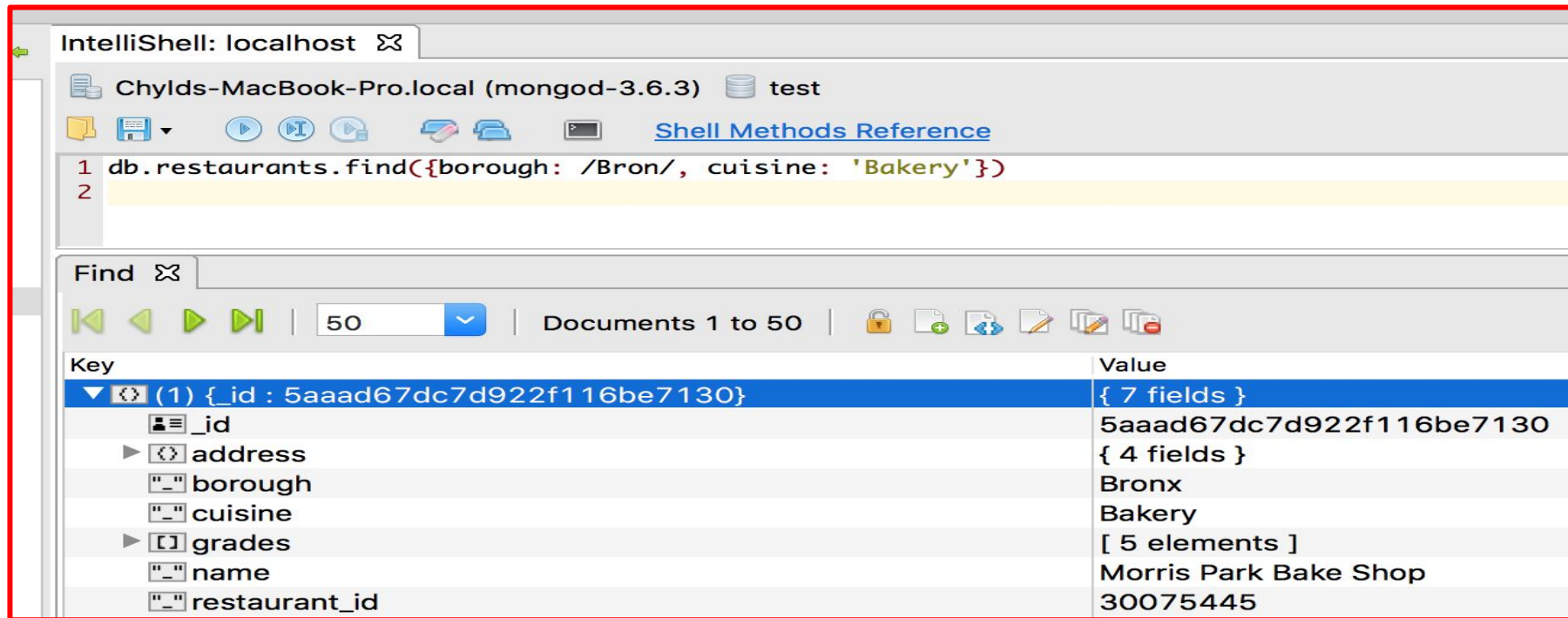


The screenshot shows the IntelliJShell interface for a MongoDB instance. The title bar reads "IntelliShell: localhost". Below the title bar, the connection details are "Chylds-MacBook-Pro.local (mongod-3.6.3)" and the database is "test". A toolbar contains icons for file operations (folders, save, play, step through, refresh), a search icon, and a link to the "Shell Methods Reference". The command prompt shows two queries:

```
1 db.restaurants.find({borough: 'Bronx'}).count()  
2 db.restaurants.find({borough: /Bron/ }).count()  
3
```

Querying MongoDB

Searching for Restaurants using multiple properties.



The screenshot displays the MongoDB IntelliShell interface. The top tab is labeled "IntelliShell: localhost". Below the tab, the connection details are "Chylds-MacBook-Pro.local (mongod-3.6.3)" and the database is "test". A toolbar contains icons for file operations and a link to the "Shell Methods Reference". The command input area shows a query: `1 db.restaurants.find({borough: /Bron/, cuisine: 'Bakery'})`. Below the command, the "Find" panel shows the results. It includes a toolbar with navigation icons, a limit of 50 documents, and a range of "Documents 1 to 50". The results are displayed in a table with "Key" and "Value" columns. The first document is expanded, showing its fields: `_id`, `address`, `borough`, `cuisine`, `grades`, `name`, and `restaurant_id`.

Key	Value
(1) {_id : 5aaad67dc7d922f116be7130}	{ 7 fields }
_id	5aaad67dc7d922f116be7130
address	{ 4 fields }
borough	Bronx
cuisine	Bakery
grades	[5 elements]
name	Morris Park Bake Shop
restaurant_id	30075445

Querying MongoDB

Searching for Restaurants using embedded properties.

The screenshot shows the MongoDB Shell (IntelliShell) interface. The command prompt shows the following query:

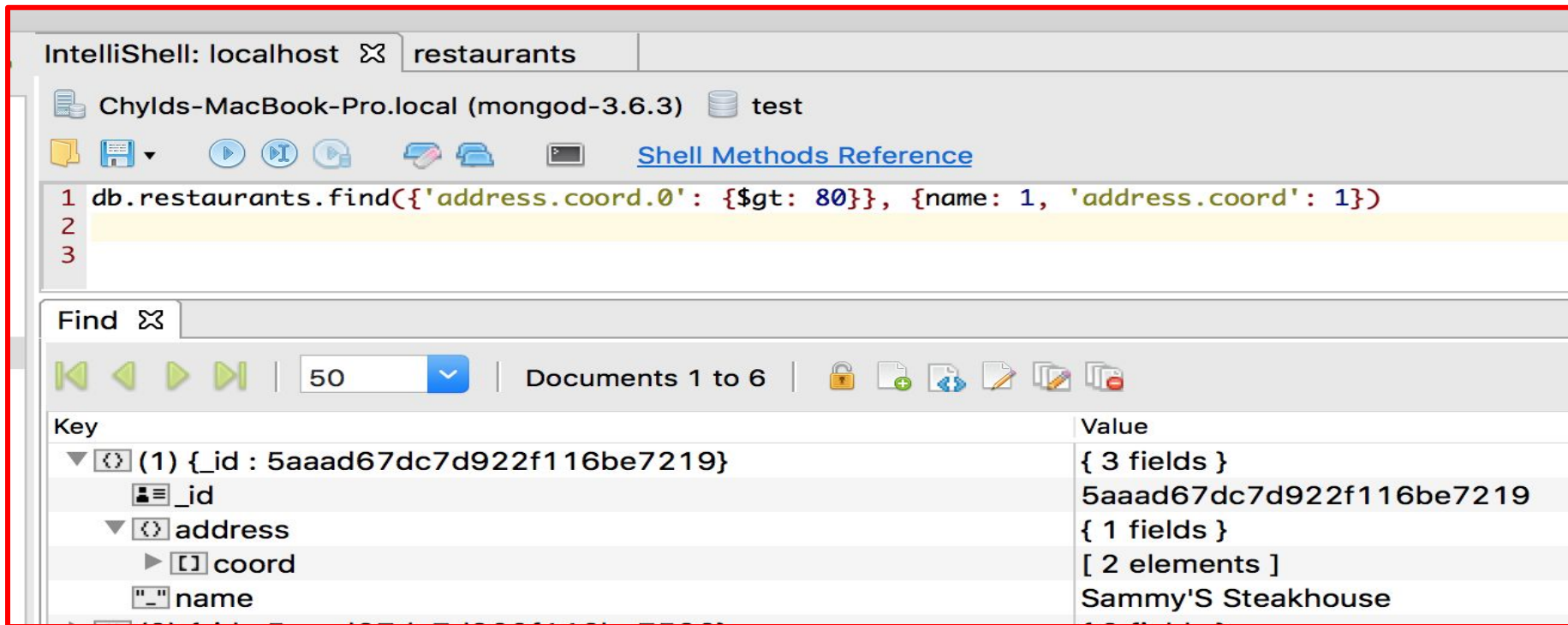
```
1 db.restaurants.find({'address.zipcode': '10462'})
2
```

The result is a single document. The interface shows the document structure with the following fields:

Key	Value
(1) {_id : 5aaad67dc7d922f116be7130}	{ 7 fields }
_id	5aaad67dc7d922f116be7130
address	{ 4 fields }
building	1007
coord	[2 elements]
street	Morris Park Ave
zipcode	10462

Querying MongoDB

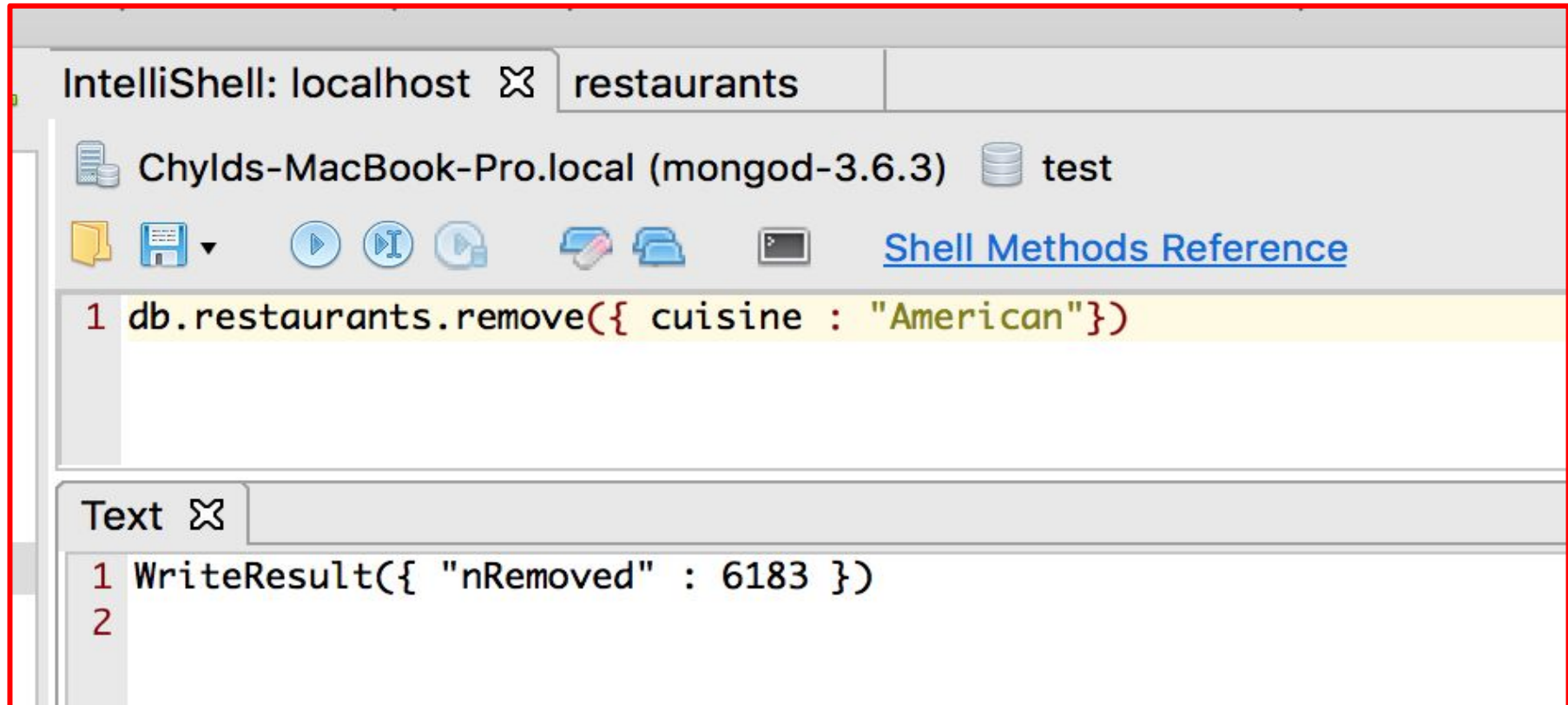
Using query operators and result projection



The screenshot shows the MongoDB IntelliShell interface. The top bar indicates the connection is to 'localhost' and the database is 'restaurants'. Below the bar, the command prompt shows the query: `db.restaurants.find({'address.coord.0': {$gt: 80}}, {name: 1, 'address.coord': 1})`. The result is displayed in a table format with columns 'Key' and 'Value'.

Key	Value
(1) {_id : 5aaad67dc7d922f116be7219}	{ 3 fields }
_id	5aaad67dc7d922f116be7219
address	{ 1 fields }
coord	[2 elements]
name	Sammy'S Steakhouse

Deleting Records



The screenshot shows the IntelliJ Shell interface with a red border. The top bar indicates the current shell is 'IntelliShell: localhost' and the selected database is 'restaurants'. Below this, the connection details are 'Chylds-MacBook-Pro.local (mongod-3.6.3)' and the database is 'test'. A toolbar with various icons (file, run, debug, etc.) is visible. The main text area contains a single line of MongoDB code: `1 db.restaurants.remove({ cuisine : "American"})`. Below the main text area, there is a 'Text' tab with a list of results: `1 WriteResult({ "nRemoved" : 6183 })` and `2`.

IntelliShell: localhost ✕ restaurants

Chylds-MacBook-Pro.local (mongod-3.6.3) test

Shell Methods Reference

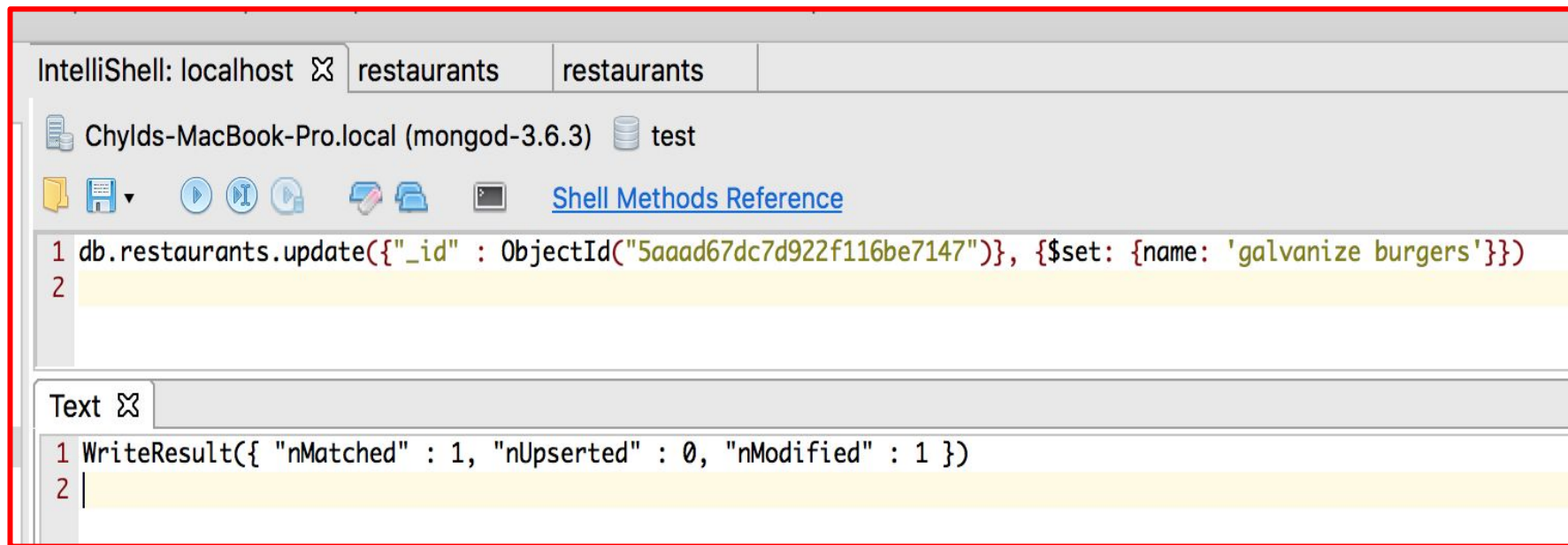
```
1 db.restaurants.remove({ cuisine : "American"})
```

Text ✕

```
1 WriteResult({ "nRemoved" : 6183 })
2
```

Updating Records

Finding an object by Id, then altering one of its properties



The screenshot shows the IntelliJ Shell interface with a red border. At the top, there are tabs for 'IntelliShell: localhost', 'restaurants', and 'restaurants'. Below the tabs, the connection information 'Chylds-MacBook-Pro.local (mongod-3.6.3)' and the database 'test' are displayed. A toolbar with various icons and a link to 'Shell Methods Reference' is visible. The main text area contains a MongoDB update command on line 1: `db.restaurants.update({"_id" : ObjectId("5aaad67dc7d922f116be7147")}, {$set: {name: 'galvanize burgers'}})`. Below this, the result is shown in the 'Text' tab on line 1: `WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })`. Line 2 in both the command and result areas is empty.

```
IntelliShell: localhost restaurants restaurants
Chylds-MacBook-Pro.local (mongod-3.6.3) test
Shell Methods Reference
1 db.restaurants.update({"_id" : ObjectId("5aaad67dc7d922f116be7147")}, {$set: {name: 'galvanize burgers'}})
2
Text
1 WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
2
```

Questions?

- How many restaurants in Queens serve Ice Cream?
- How many Cold Stone Creamery stores are in Manhattan?
- How many restaurants are in the 10001 zip code?