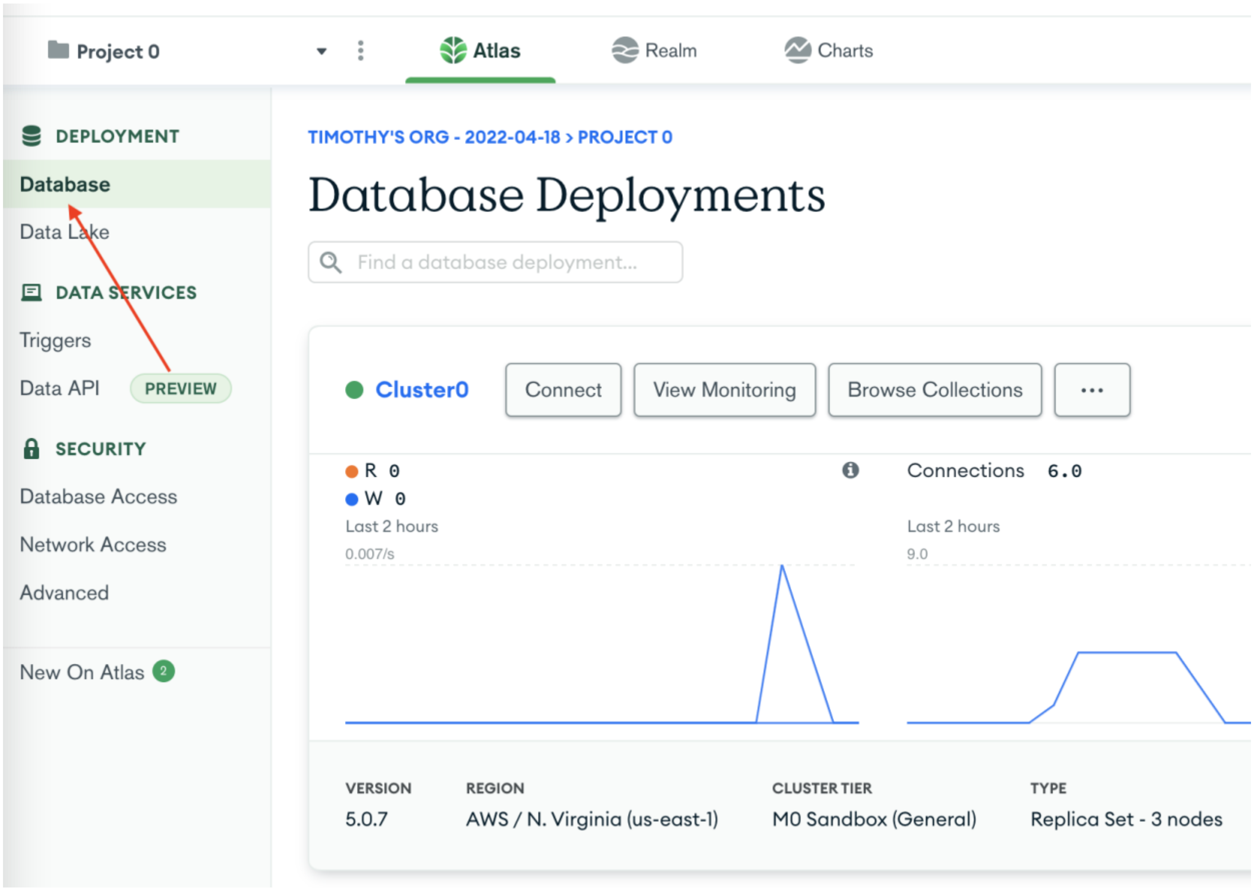
Basic MongoDB Queries

The purpose of this tutorial is to get you started with the basics of connecting to a MongoDB cluster and performing some basic operations. It shows you how to:

1. Connect to a MongoDB database using **`mongosh`**
2. Perform basic MongoDB operations to interact with a MongoDB database and its data.

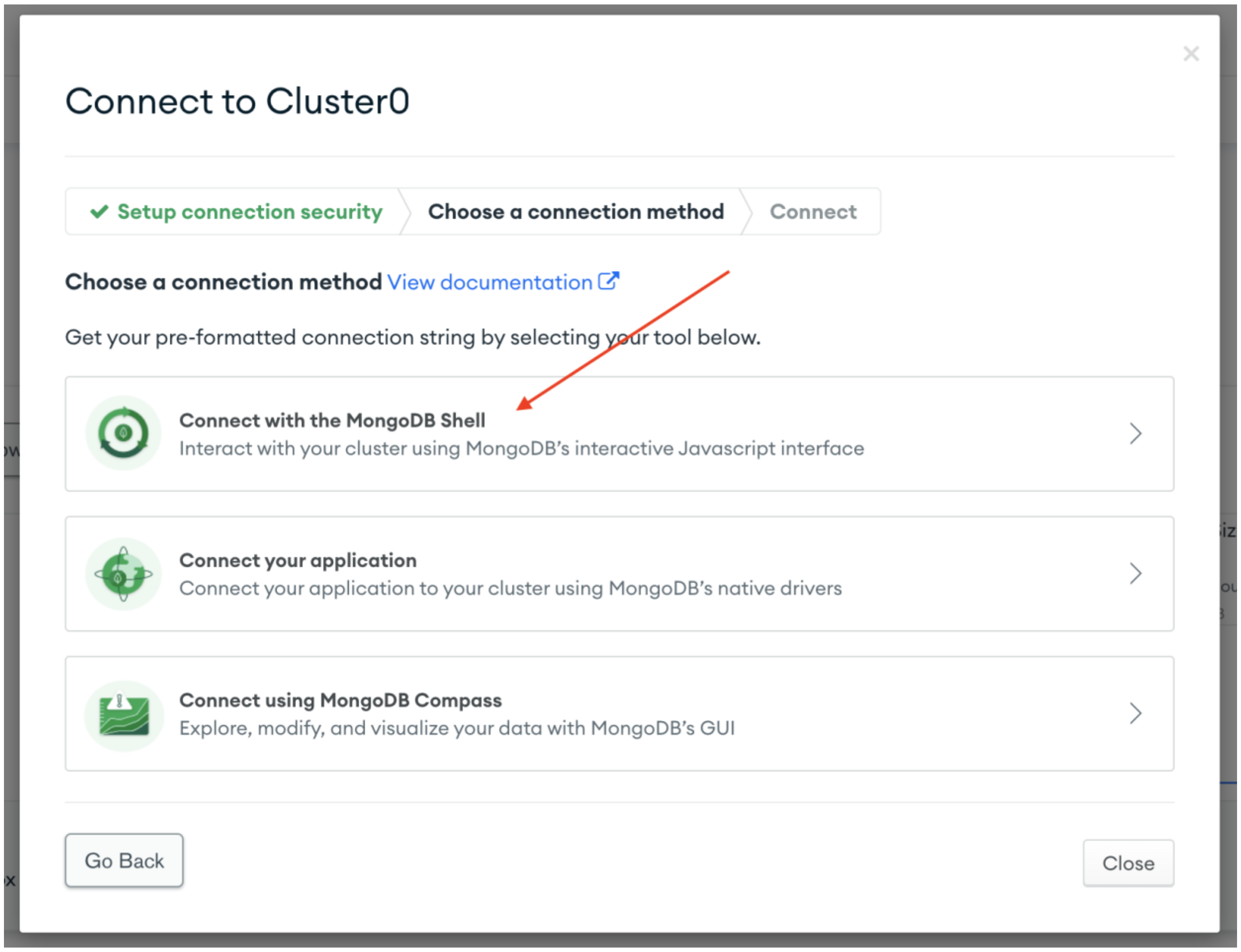
Part 1: Connecting to the cluster

At this point, you should already have created an account and MongoDB cluster on Atlas MongoDB. Now, you need to connect to the cluster to explore MongoDB queries. To do this, you'll need to specify the cluster's name and the username and password of the account you created. You can easily find the connection information from the Atlas dashboard when you are logged into Atlas. Hopefully, you remember your username and password as this will be important to connect to the database. To get the connection information, you need to visit the **Database Deployments** page in the Atlas dashboard:



VERSION	REGION	CLUSTER TIER	TYPE
5.0.7	AWS / N. Virginia (us-east-1)	M0 Sandbox (General)	Replica Set - 3 nodes

Next, you want to click on the **Connect** button. This will allow you to get the connection information for a variety of ways to connect to the cluster. We will choose the **MongoDB Shell** connection method in this first example.



Connect to Cluster0

✓ Setup connection security | Choose a connection method | Connect

Choose a connection method View documentation

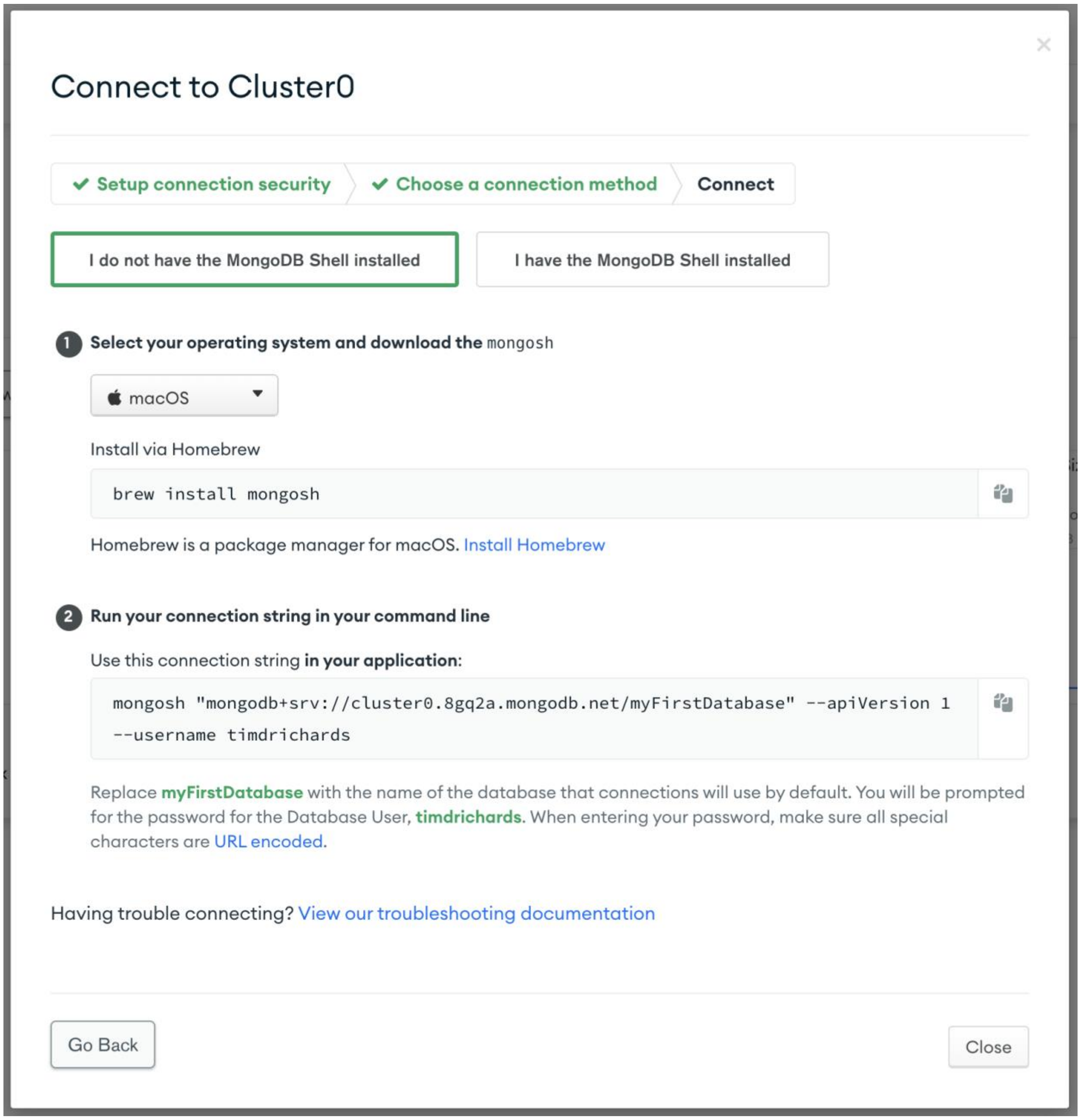
Get your pre-formatted connection string by selecting your tool below.

Connect with the MongoDB Shell
Interact with your cluster using MongoDB's interactive Javascript interface

Connect your application
Connect your application to your cluster using MongoDB's native drivers

Connect using MongoDB Compass
Explore, modify, and visualize your data with MongoDB's GUI

This will open another dialog box where you will be able to download the **`mongosh`** executable and use the provided *connection string* to connect to the cluster. First, you want to download and install the **`mongosh`** executable according to the platform you are working on. Note, on a Mac, you will need to first install the homebrew package manager. After you have installed **`mongosh`**, you can run it from a terminal/console using the command shown in step 2 of the following figure. Note, the connection information will be slightly different since it will depend on your cluster ID, database name, and username.



Connect to Cluster0

✓ Setup connection security ✓ Choose a connection method Connect

I do not have the MongoDB Shell installed | I have the MongoDB Shell installed

1 **Select your operating system and download the** `mongosh`

macOS

Install via Homebrew

```
brew install mongosh
```

Homebrew is a package manager for macOS. Install Homebrew

2 **Run your connection string in your command line**

Use this connection string **in your application**:

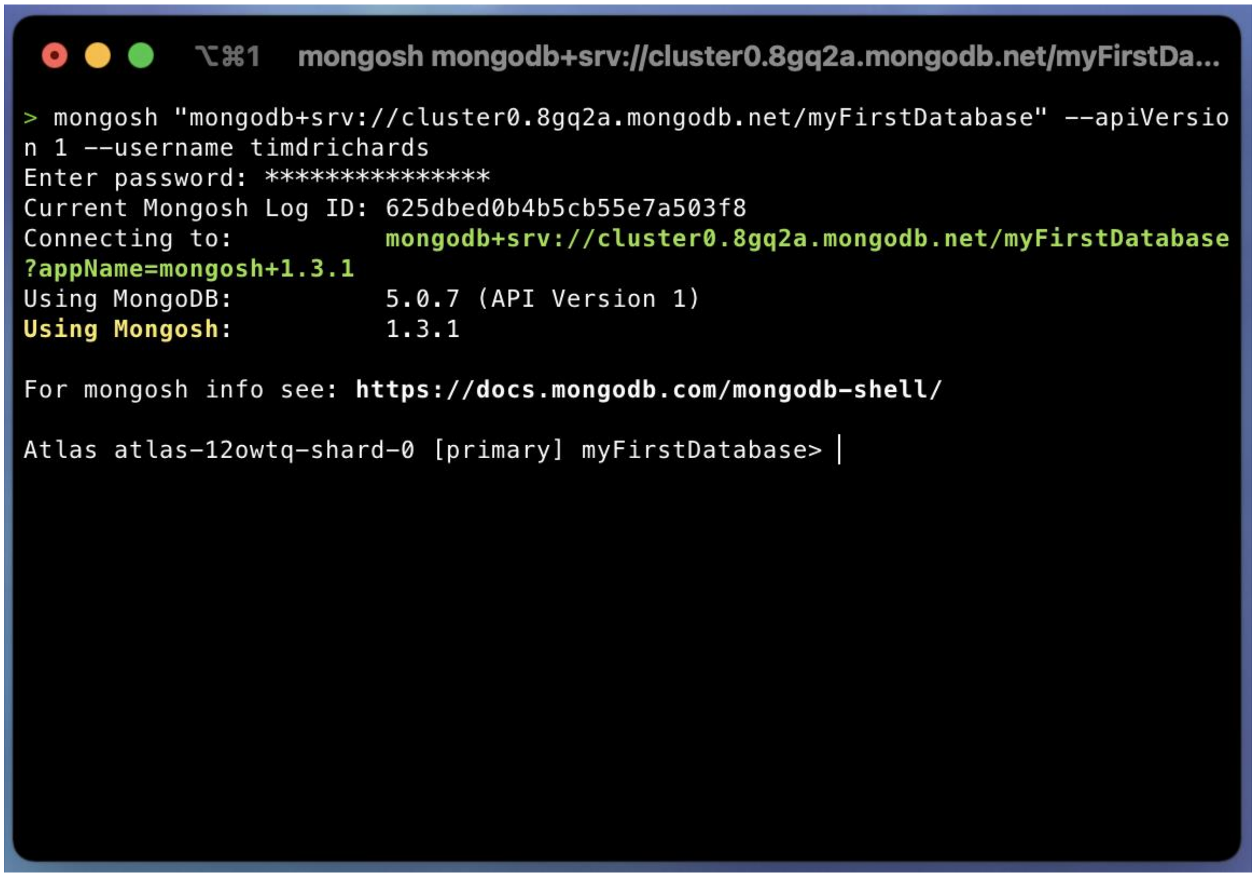
```
mongosh "mongodb+srv://cluster0.8gq2a.mongodb.net/myFirstDatabase" --apiVersion 1
--username timdrichards
```

Replace **myFirstDatabase** with the name of the database that connections will use by default. You will be prompted for the password for the Database User, **timdrichards**. When entering your password, make sure all special characters are URL encoded.

Having trouble connecting? View our troubleshooting documentation

Go Back | Close

After you have completed the above steps, you are ready to open a terminal and connect to your MongoDB in the cloud. Although it will look different depending on your platform, it will generally look like this:



```
⌥⌘1   mongosh mongodb+srv://cluster0.8gq2a.mongodb.net/myFirstDa...

> mongosh "mongodb+srv://cluster0.8gq2a.mongodb.net/myFirstDatabase" --apiVersio
n 1 --username timdrichards
Enter password: ***************
Current Mongosh Log ID: 625dbed0b4b5cb55e7a503f8
Connecting to:          mongodb+srv://cluster0.8gq2a.mongodb.net/myFirstDatabase
?appName=mongosh+1.3.1
Using MongoDB:          5.0.7 (API Version 1)
Using Mongosh:          1.3.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> |
```

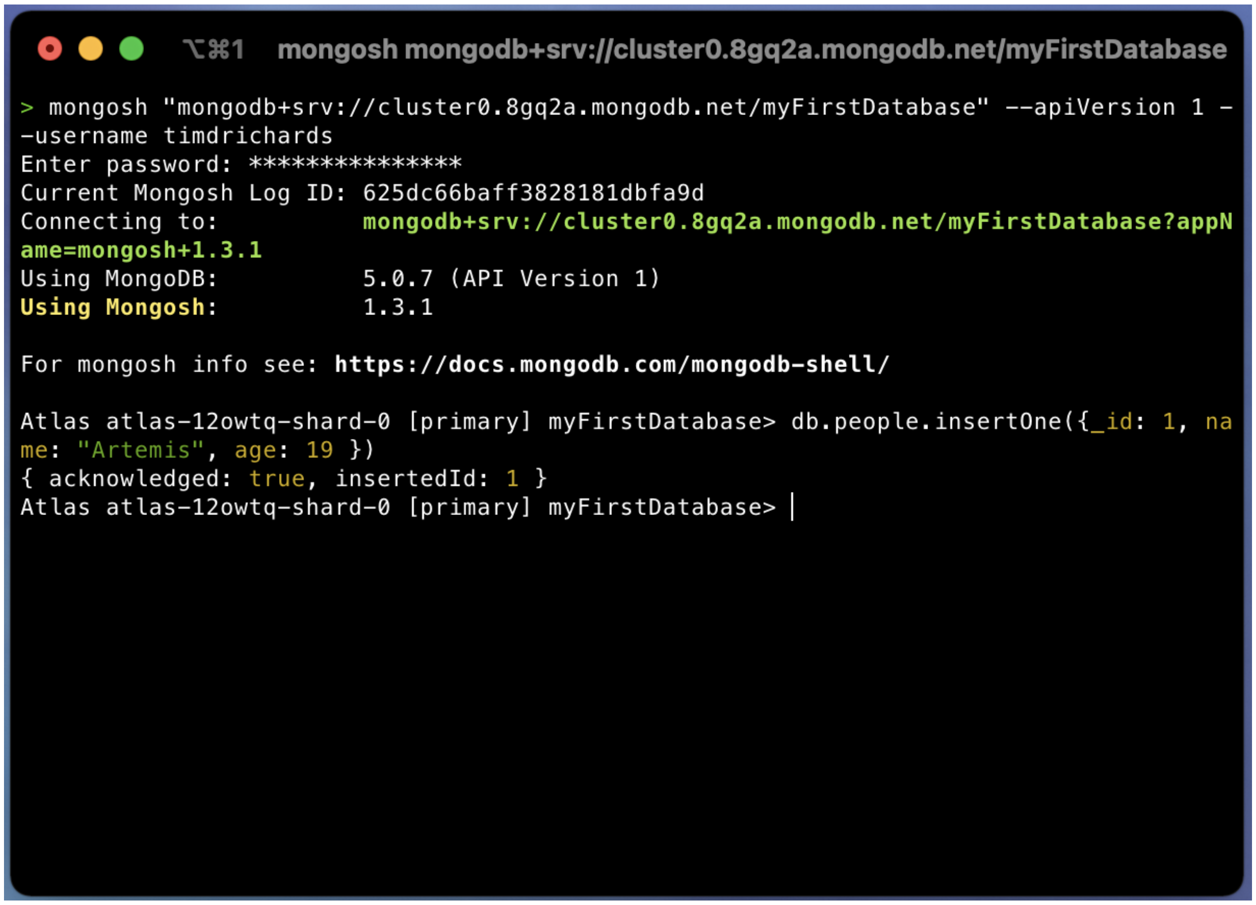
Part 2: Running some basic queries

Before we get started with querying a database, it is important to understand the basic structure of a Mongo database. A Mongo database is made up of the following three important parts:

- **Documents**: a document is a JSON-like structure that consists of zero or more ordered field-value pairs. For our purposes, documents are synonymous with JSON structures.
- **Collections**: a collection is essentially an array of documents. It is analogous to a table in a relational database that stores rows/records. Queries are performed on collections.
- **Database**: a database is a set of collections.

With the basic Mongo structure in mind and assuming you were able to connect to your Atlas cluster successfully, we can now perform some basic CRUD operations to see what it is like to work with a NoSQL database.

Creating Data: insertOne()
First, let us add some data to the database we are connected to. In the figure above this is designated as the *myFirstDatabase* database. Fortunately, the language that is used to communicate to a MongoDB database is JavaScript. The following command inserts a single record into the people collection:

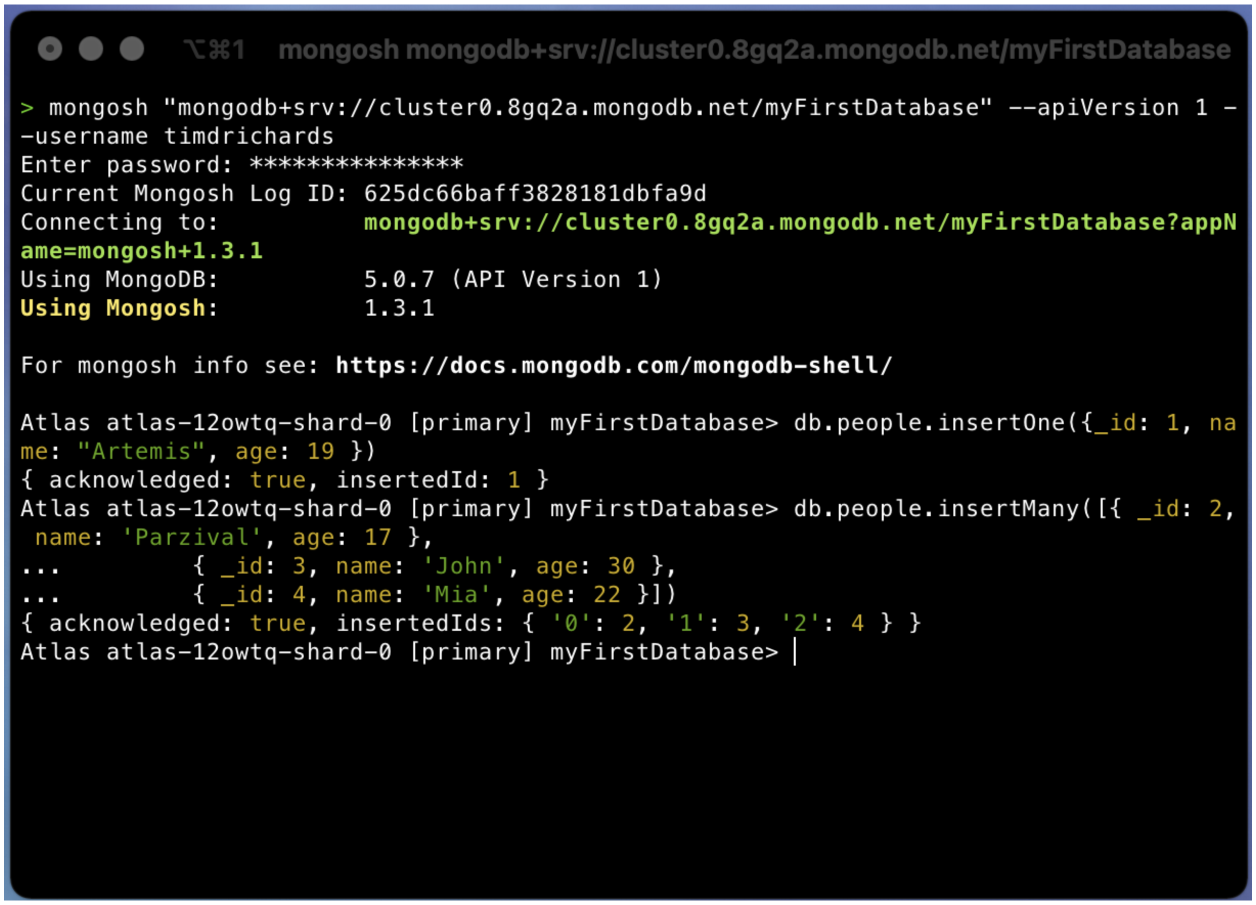

⌥⌘1 mongosh mongodb+srv://cluster0.8gq2a.mongodb.net/myFirstDatabase

```
> mongosh "mongodb+srv://cluster0.8gq2a.mongodb.net/myFirstDatabase" --apiVersion 1 -
-username timdrichards
Enter password: ***************
Current Mongosh Log ID: 625dc66baff3828181dbfa9d
Connecting to:          mongodb+srv://cluster0.8gq2a.mongodb.net/myFirstDatabase?appN
ame=mongosh+1.3.1
Using MongoDB:          5.0.7 (API Version 1)
Using Mongosh:          1.3.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> db.people.insertOne({_id: 1, na
me: "Artemis", age: 19 })
{ acknowledged: true, insertedId: 1 }
Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> |
```

This inserted a document into the *people* collection in the *myFirstDatabase* database referenced by **db**. You may wonder where the *people* collection came from. It turns out, in Mongo you can create a new collection simply be referencing a new collection by name. In this case, the **`people`** collection didn't exist until we referenced it using **`db.people`**. To insert a new document we use the **`insertOne()`** method on a collection.

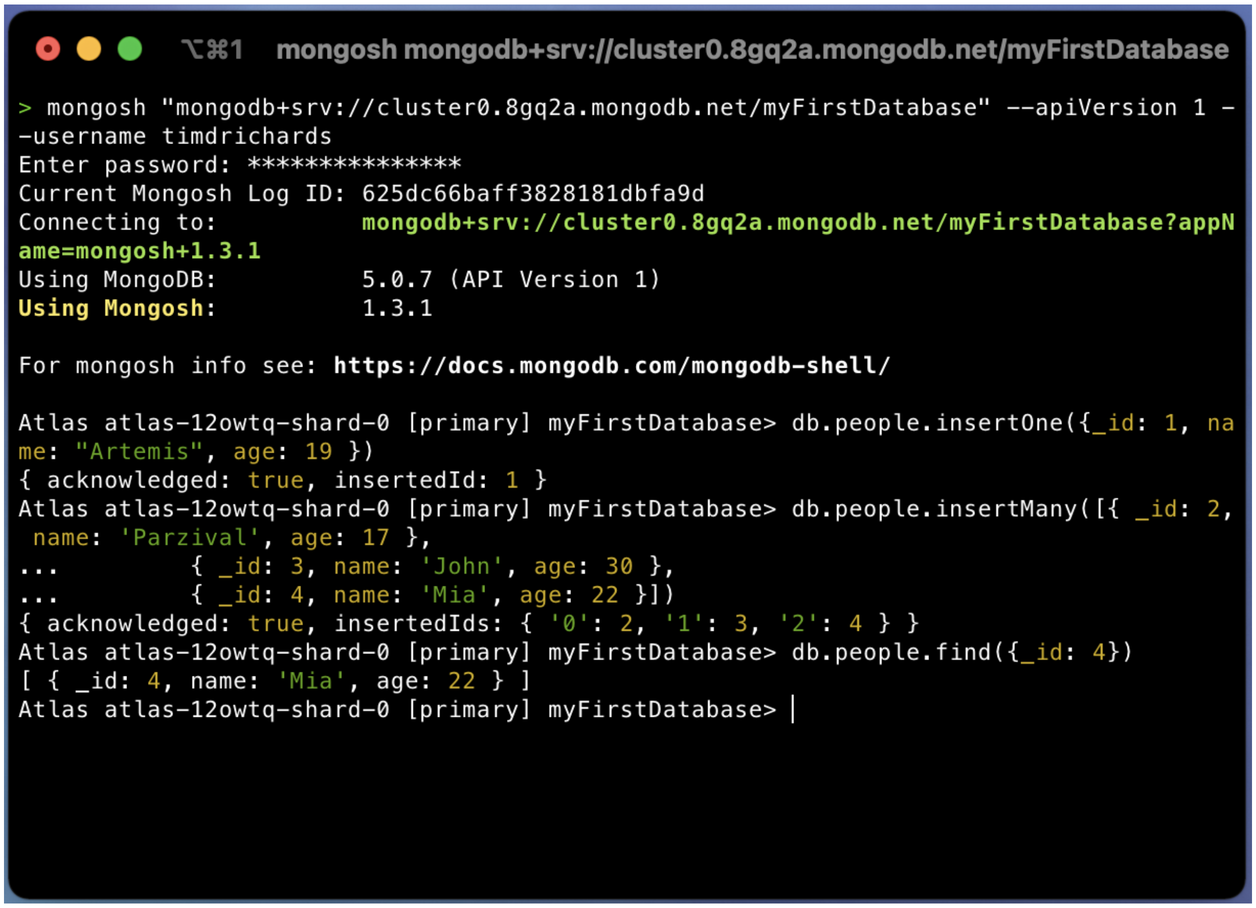
You will also note that we used _id as the property for the identifier used to identify the person. There is special significance associated with the _id property. In short, it is used internally by the database to identify and reference documents uniquely. The true details of this is beyond the scope of this tutorial and course.

Creating Data: insertMany()
If we have more than one document to add to a collection we can use the **`insertMany()`** method on the collection. In this case, we provide an array of documents to be inserted. The use of this is clear:



```
⌥⌘1   mongosh mongodb+srv://cluster0.8gq2a.mongodb.net/myFirstDatabase

> mongosh "mongodb+srv://cluster0.8gq2a.mongodb.net/myFirstDatabase" --apiVersion 1 -
-username timdrichards
Enter password: ***************
Current Mongosh Log ID: 625dc66baff3828181dbfa9d
Connecting to:          mongodb+srv://cluster0.8gq2a.mongodb.net/myFirstDatabase?appN
ame=mongosh+1.3.1
Using MongoDB:          5.0.7 (API Version 1)
Using Mongosh:          1.3.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> db.people.insertOne({_id: 1, na
me: "Artemis", age: 19 })
{ acknowledged: true, insertedId: 1 }
Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> db.people.insertMany([{ _id: 2,
 name: 'Parzival', age: 17 },
...         { _id: 3, name: 'John', age: 30 },
...         { _id: 4, name: 'Mia', age: 22 }])
{ acknowledged: true, insertedIds: { '0': 2, '1': 3, '2': 4 } }
Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> |
```

Reading Data: find()
To read data from the database we use the **find()** method. This method is given a query represented as a document and returns an array of documents that are found. Here is an example of finding a person whose _id is 4:

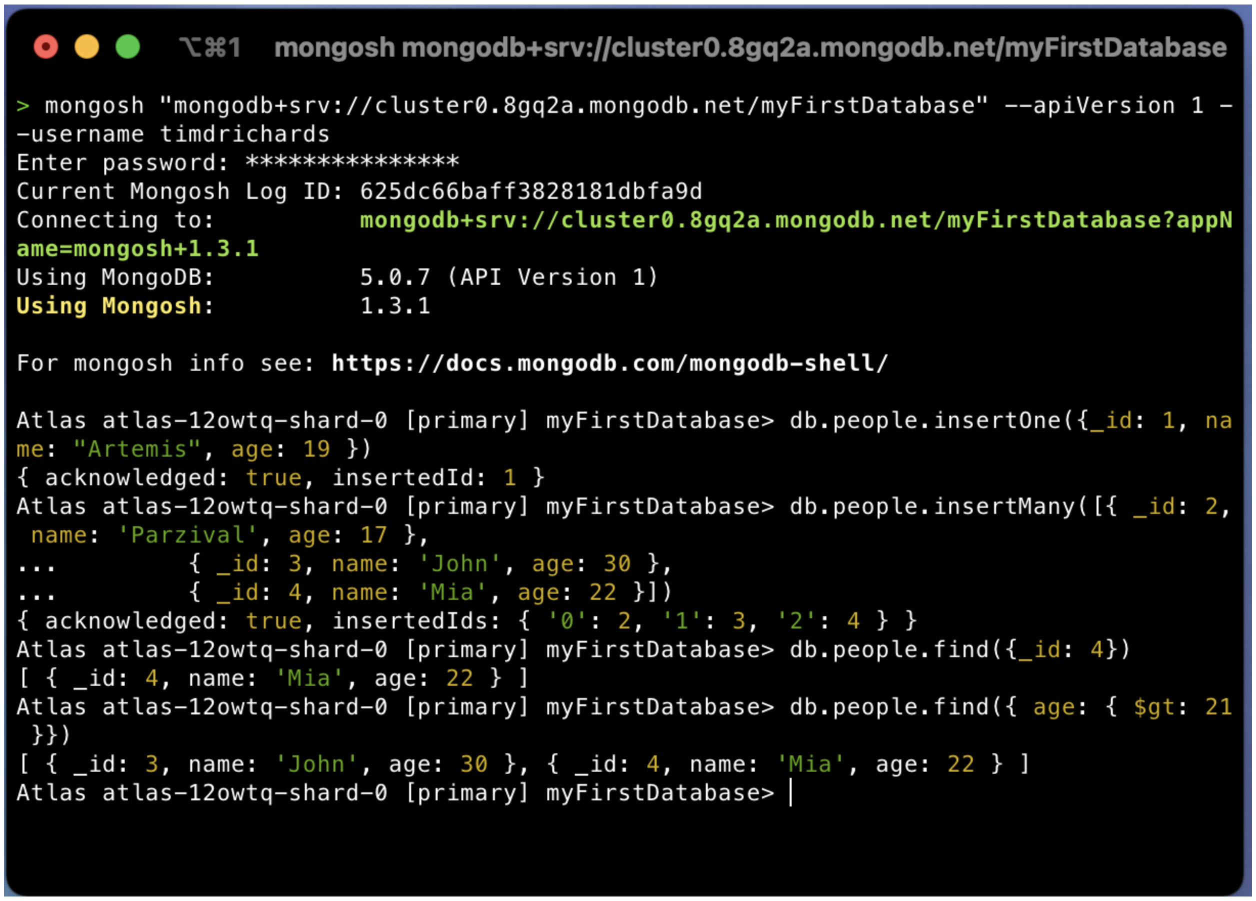

⌥⌘1 mongosh mongodb+srv://cluster0.8gq2a.mongodb.net/myFirstDatabase

```
> mongosh "mongodb+srv://cluster0.8gq2a.mongodb.net/myFirstDatabase" --apiVersion 1 -
-username timdrichards
Enter password: ***************
Current Mongosh Log ID: 625dc66baff3828181dbfa9d
Connecting to:          mongodb+srv://cluster0.8gq2a.mongodb.net/myFirstDatabase?appN
ame=mongosh+1.3.1
Using MongoDB:          5.0.7 (API Version 1)
Using Mongosh:          1.3.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> db.people.insertOne({_id: 1, na
me: "Artemis", age: 19 })
{ acknowledged: true, insertedId: 1 }
Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> db.people.insertMany([{ _id: 2,
 name: 'Parzival', age: 17 },
...         { _id: 3, name: 'John', age: 30 },
...         { _id: 4, name: 'Mia', age: 22 }])
{ acknowledged: true, insertedIds: { '0': 2, '1': 3, '2': 4 } }
Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> db.people.find({_id: 4})
[ { _id: 4, name: 'Mia', age: 22 } ]
Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> |
```

As you can see, the result is an array of documents matching the query criteria that was provided. We can also query for people whose age is great than 21. In this case, instead of a matching value, you provide an object with a single **$gt** field and a value that you want to match against. Here is the query:

```
db.people.find({age: {$gt: 21}})
```

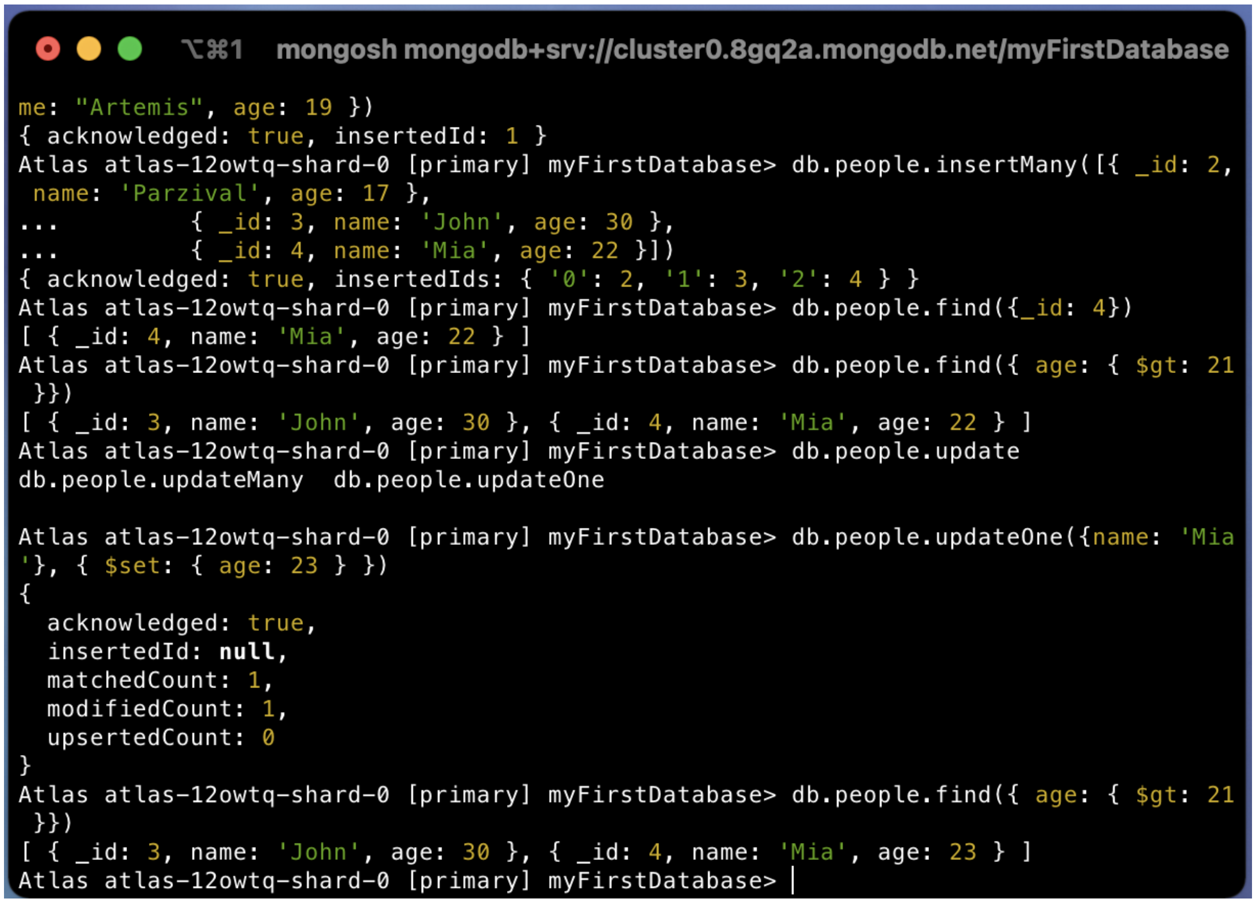
Rather than matching the value exactly, we restrict the values to only those ages that are less than 21. The execution of this query is shown in the following figure.



```
⌥⌘1   mongosh mongodb+srv://cluster0.8gq2a.mongodb.net/myFirstDatabase

> mongosh "mongodb+srv://cluster0.8gq2a.mongodb.net/myFirstDatabase" --apiVersion 1 -
-username timdrichards
Enter password: ***************
Current Mongosh Log ID: 625dc66baff3828181dbfa9d
Connecting to:          mongodb+srv://cluster0.8gq2a.mongodb.net/myFirstDatabase?appN
ame=mongosh+1.3.1
Using MongoDB:          5.0.7 (API Version 1)
Using Mongosh:          1.3.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> db.people.insertOne({_id: 1, na
me: "Artemis", age: 19 })
{ acknowledged: true, insertedId: 1 }
Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> db.people.insertMany([{ _id: 2,
 name: 'Parzival', age: 17 },
...         { _id: 3, name: 'John', age: 30 },
...         { _id: 4, name: 'Mia', age: 22 }])
{ acknowledged: true, insertedIds: { '0': 2, '1': 3, '2': 4 } }
Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> db.people.find({_id: 4})
[ { _id: 4, name: 'Mia', age: 22 } ]
Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> db.people.find({ age: { $gt: 21
 }})
[ { _id: 3, name: 'John', age: 30 }, { _id: 4, name: 'Mia', age: 22 } ]
Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> |
```

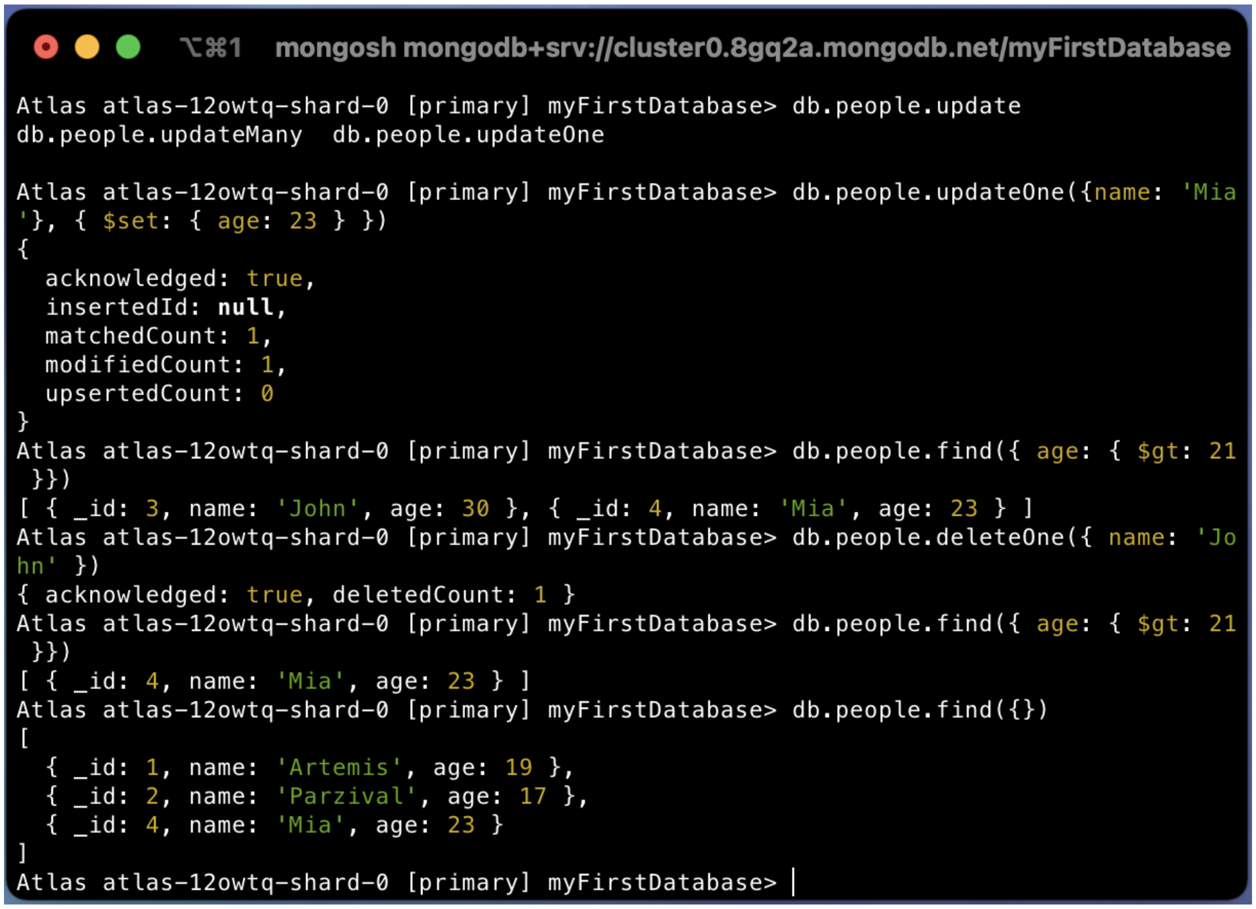
There are many different operators that you can use for restricting the results you are searching for. We refer you to the MongoDB documentation to see how this is done (or this Quick Reference).

Updating Data: updateOne() / updateMany()
To update data requires you to provide search criteria and data to be updated. We saw how we search for data, but how do we update it? It should not be surprising that we need to provide both the criteria to search for as well as the data to be updated. Here is an example of how we update the age of a person in our very small database:


⌥⌘1 mongosh mongodb+srv://cluster0.8gq2a.mongodb.net/myFirstDatabase

```
me: "Artemis", age: 19 })
{ acknowledged: true, insertedId: 1 }
Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> db.people.insertMany([{ _id: 2,
 name: 'Parzival', age: 17 },
...         { _id: 3, name: 'John', age: 30 },
...         { _id: 4, name: 'Mia', age: 22 }])
{ acknowledged: true, insertedIds: { '0': 2, '1': 3, '2': 4 } }
Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> db.people.find({_id: 4})
[ { _id: 4, name: 'Mia', age: 22 } ]
Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> db.people.find({ age: { $gt: 21
 }})
[ { _id: 3, name: 'John', age: 30 }, { _id: 4, name: 'Mia', age: 22 } ]
Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> db.people.update
db.people.updateMany  db.people.updateOne

Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> db.people.updateOne({name: 'Mia
'}, { $set: { age: 23 } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> db.people.find({ age: { $gt: 21
 }})
[ { _id: 3, name: 'John', age: 30 }, { _id: 4, name: 'Mia', age: 23 } ]
Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> 
```

Deleting Data: deleteOne() / deleteMany()
Deleting data is just like find except we are finding the data to delete. Here is an example of that:


⌥⌘1 mongosh mongodb+srv://cluster0.8gq2a.mongodb.net/myFirstDatabase

```
Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> db.people.update
db.people.updateMany  db.people.updateOne

Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> db.people.updateOne({name: 'Mia
'}, { $set: { age: 23 } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> db.people.find({ age: { $gt: 21
 }})
[ { _id: 3, name: 'John', age: 30 }, { _id: 4, name: 'Mia', age: 23 } ]
Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> db.people.deleteOne({ name: 'Jo
hn' })
{ acknowledged: true, deletedCount: 1 }
Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> db.people.find({ age: { $gt: 21
 }})
[ { _id: 4, name: 'Mia', age: 23 } ]
Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> db.people.find({})
[
  { _id: 1, name: 'Artemis', age: 19 },
  { _id: 2, name: 'Parzival', age: 17 },
  { _id: 4, name: 'Mia', age: 23 }
]
Atlas atlas-12owtq-shard-0 [primary] myFirstDatabase> |
```