Querying Data from MongoDB

WRITING YOUR FIRST MONGO QUERY

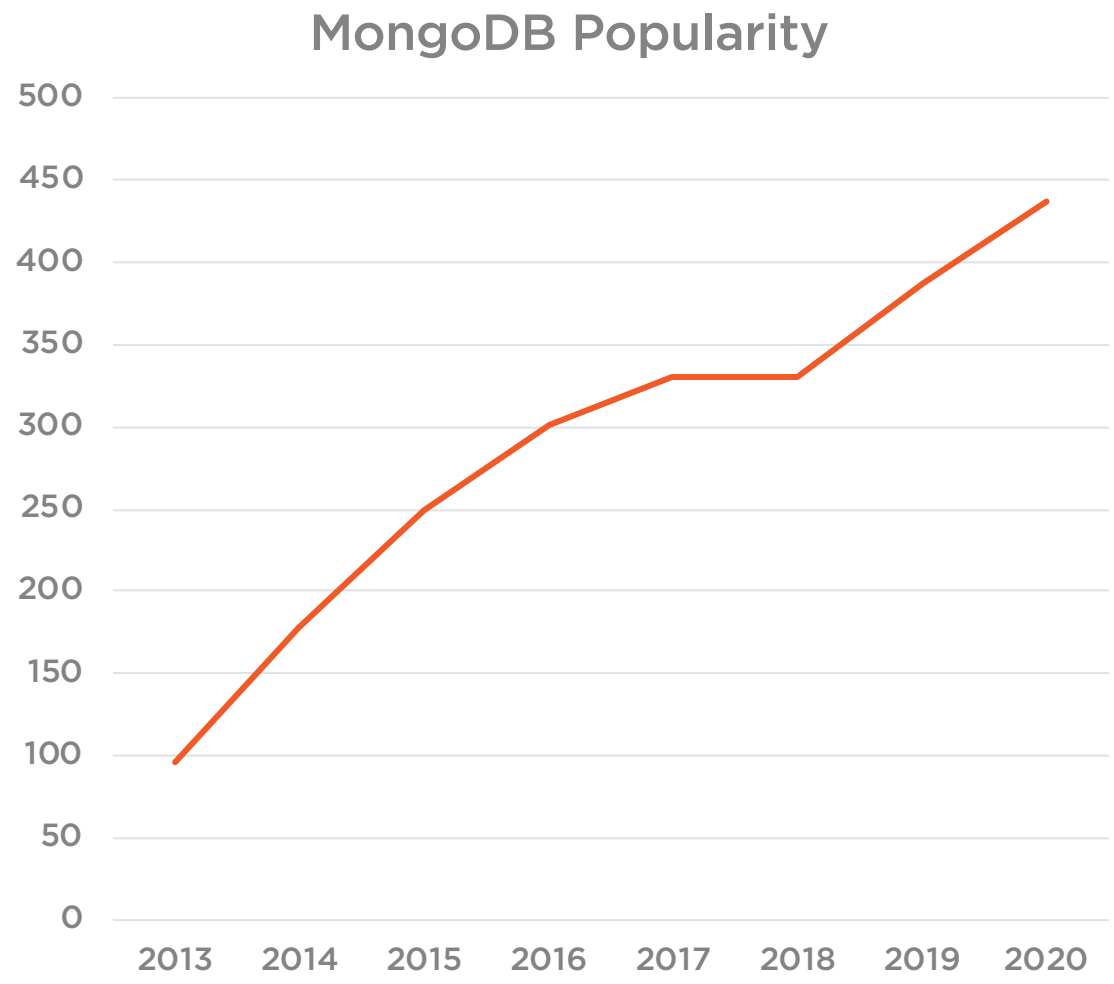
Dan Geabunea
SENIOR SOFTWARE DEVELOPER

@romaniancoder www.romaniancoder.com

Ranks 5th in the most popular database management systems

The most popular document database

https://db-engines.com/en/ranking



https://db-engines.com/en/ranking

Course Overview

- Connect to a Mongo server and query documents from any collection
- Master filtering and learn about the most used query operators
- Create queries based on nested documents or arrays
- Create queries by composing multiple conditions
- Tackle edge cases when querying null or missing fields

You will be proficient
querying data in MongoDB

Writing Your First Mongo Query

Overview

Basic MongoDB concepts

The shell and GUI alternatives

Querying all documents from a Mongo database collection

The sample data used in this course

Demo: Writing our first queries

Course Pre-requisites

JavaScript / JSON

Data engineering literacy

Basic familiarity with document databases

Learn by Doing

Install MongoDB Server

https://docs.mongodb.com/manual/installation/

Import the sample data

https://github.com/dangeabunea/pluralsight-mongodb-queries

Learning Checks

Basic MongoDB Concepts

MongoDB

Database management system built for scalability and flexibility which uses a document-oriented model

Relational Model

Id	Name	Age	Rank
1	John	36	Captain
2	Anna	45	Captain
3	Joe	29	Attendant
4	Stan	38	Attendant

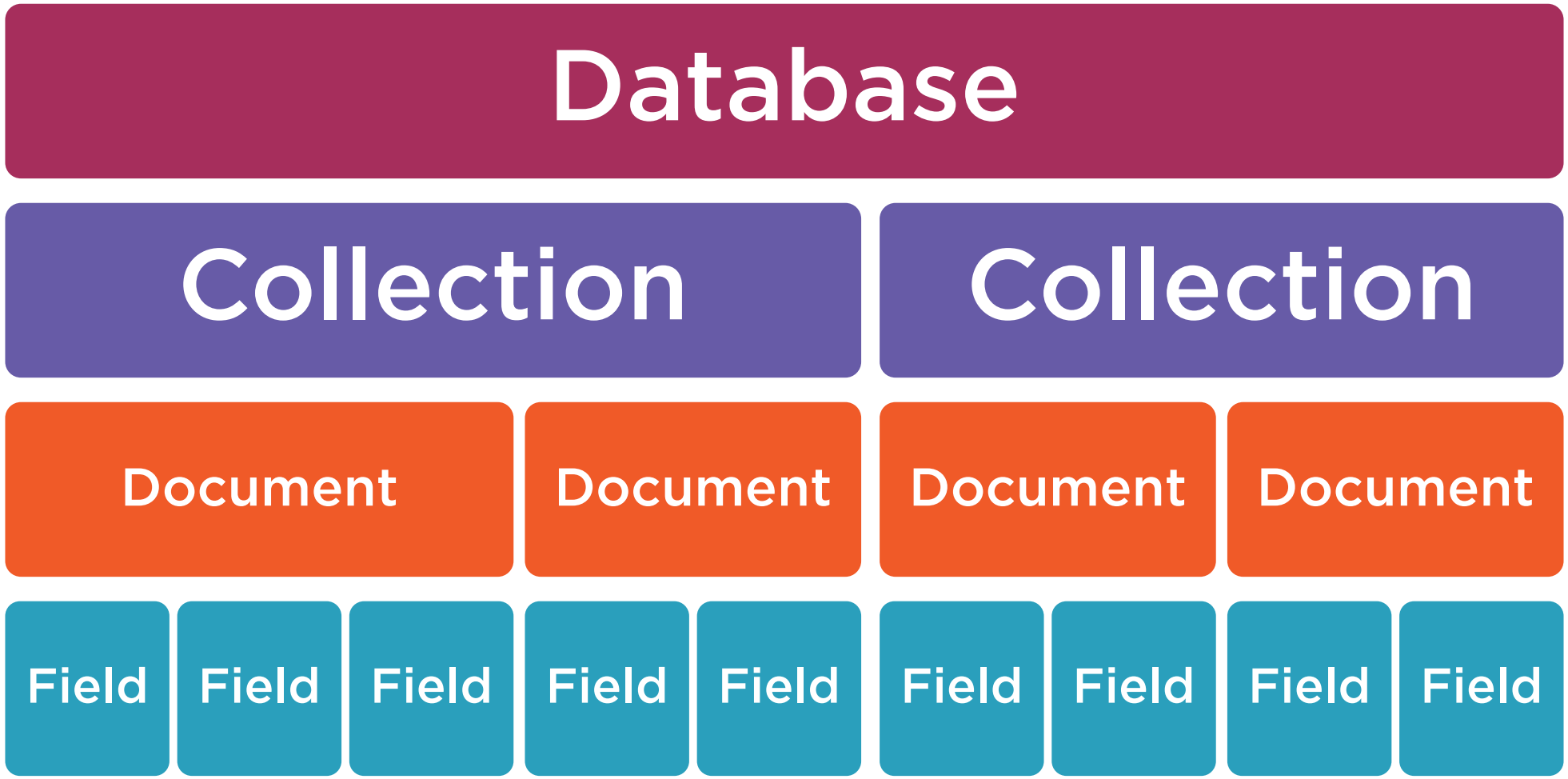
Table

Rows

Columns

Primary keys

MongoDB Model



Document

JSON

```
{
    “_id”: ObjectId("5e77b531b75ee55b200200cb"),
    “name”: “John Doe”,
    “age”: 36,
    “onDuty”: false,
    “address”: {
        “city”: “Los Angeles”,
        “street”: “627 Evergreen Lane”
    },
    “upcomingTrainings”: [{“title”: “Landing Procedures”, “durationMinutes”: 120}]
}
```

“Wait a minute…I heard that MongoDB stores data in something called BSON!”

You

BSON

MongoDB represents JSON documents in a binary form called BSON internally

It enriches JSON with additional data types that Mongo uses

It is a very fast and efficient form for storing data

You don’t have to worry about BSON or the conversion process since you will be working with JSON documents

Schemaless

```
{
    “_id”: ObjectId("bf8e2fca-a9b3-4cf2-950f-d0d7d9868580"),
    “name”: “John Doe”,
    “age”: 36
}
{
    “_id”: ObjectId("0c0ab305-c894-44e8-89e2-f4b43421f712"),
    “name”: “Anna Smith”,
    “onDuty”: true
}
```

“So I can just put anything inside a collection and Mongo won’t complain?”

You

Mongo does not enforce a schema, but documents inside the same collection should have a similar structure

Few Relationships

Nested Documents

```
{
    “_id”: ObjectId("5e77b531b75ee55b200200cb"),
    “name”: “John Doe”,
    “age”: 36,
    “onDuty”: false,
    “address”: {
        “city”: “Los Angeles”,
        “street”: “627 Evergreen Lane”
    },
    “upcomingTrainings”: [{“title”: “Landing Procedures”, “durationMinutes”: 120}]
}
```

There are cases where it makes sense to link documents with relationships, but try and keep those situations to a minimum

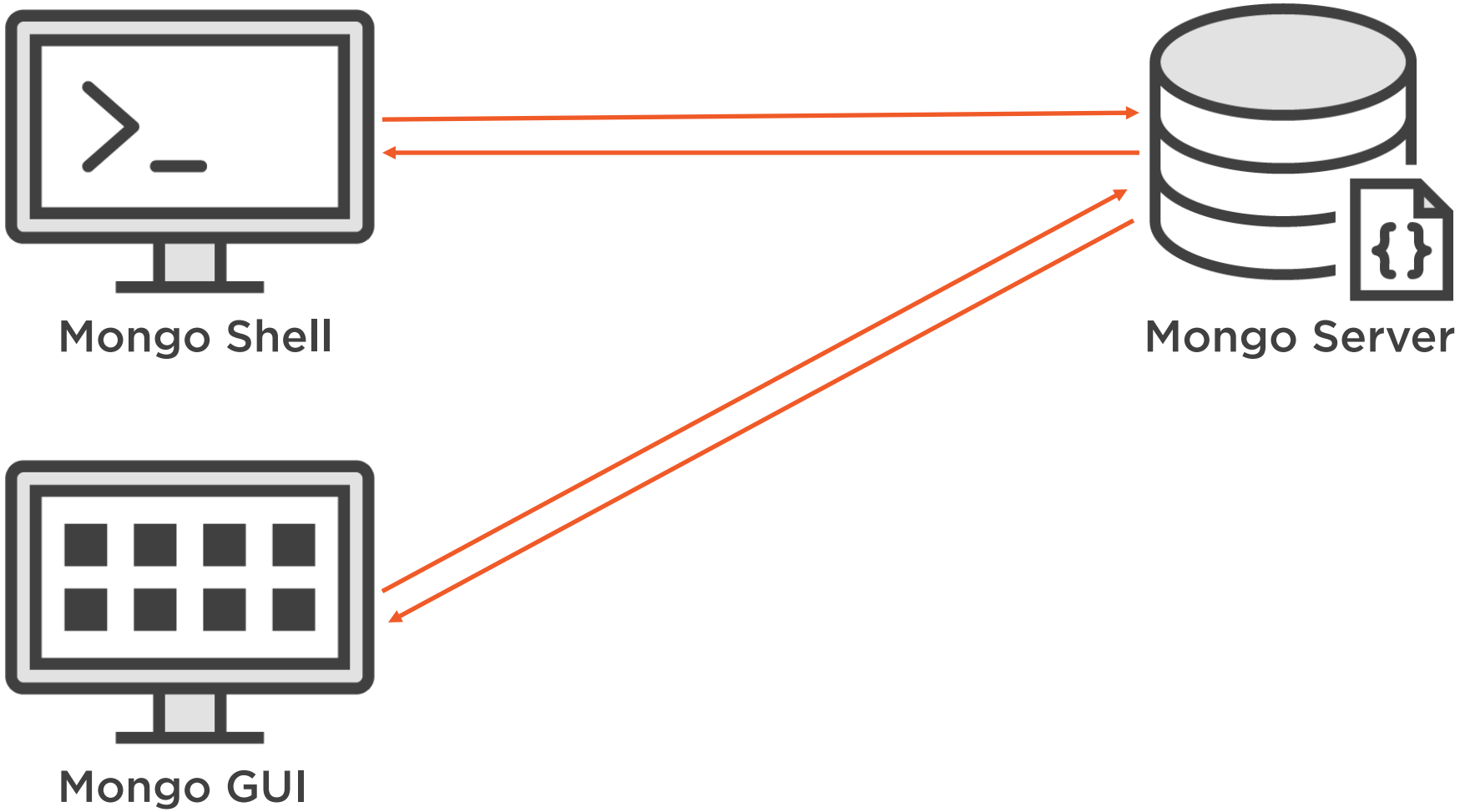
Shell and GUI Alternatives

Mongo Shell

The Mongo shell is an interactive JavaScript interface to MongoDB. You can use the mongo shell to query and update data as well as perform administrative operations

https://docs.mongodb.com/manual/mongo/

Query Flow



Firing up the Shell

```
> mongo

MongoDB shell version v4.2.2

connecting to:
mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb

Implicit session: session { "id" : UUID("ccc10eb9-adc3-440f-80eb-96c09fbbb9a5") }

MongoDB server version: 4.2.2

>

CTRL+C

bye
```

Connect to a Particular Db Server

```
> mongo --host localhost --port 27017

MongoDB shell version v4.2.2

connecting to:
mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb

>
```

```
> mongo “localhost:27017”

MongoDB shell version v4.2.2

connecting to:
mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb

>
```

Viewing and Selecting Databases

```
> mongo

MongoDB shell version v4.2.2

> db

test

> show dbs

flightmgmt 0.000GB
admin 0.000GB
local 0.000GB
test 0.000GB

> use flightmgmt

switched to db flightmgmt
```

Viewing Database Collections

```
> use flightmgmt

switched to db flightmgmt

> show collections

crew

aircraft
```

Shell Features & Configuration

You can use the up/down arrow keys to scroll through the command history

You can use TAB to trigger autocomplete or to list the completion options

You can edit the prompt display by playing with the .mongorc.js. This is typically found under your user folder.

The .mongorc.js File

Display Database and Host

```
> use test
switched to db flightmgmt
>
test@DESKTOP-R6VQM3R >
```

.mongorc.js

```
host = db.serverStatus().host;
prompt = function() {
    return db+"@"+host+" > ";
}
```

Mongo GUI

Mongo Compass

https://www.mongodb.com/products/compass

Robo 3T

https://robomongo.org/download

“Should I use the shell or go straight for the GUIs?”

You

Become comfortable with the shell, but also use GUIs when it makes sense

Querying All Documents from a Collection

Aircraft

- _id
- model
- capacity
- range
- minRunwayLength

Querying All Items

Display All Aircraft

```
> use flightmgtm

switched to db flightmgmt

> db.aircraft.count()

2

> db.aircraft.find()

{ "_id" : ObjectId("5e77b3fed0fc70bbe1d74677"), "code" : "eede6be6-f716-4e2e-bf81-
885f0a16a50c", "model" : "Boeing 737-9
00", "minRunwayLength" : 2975, "range" : 5600, "capacity" : 215 }
{ "_id" : ObjectId("5e77b3fed0fc70bbe1d74678"), "code" : "51192f6b-9c26-4ef9-b843-
cf241f326091", "model" : "Embraer E-17
5", "minRunwayLength" : 1261, "range" : 4000, "capacity" : 80 }
```

Querying All Items

Large Collections

```
> db.aircraft.find()

…

{ "_id" : ObjectId("5e77b3fed0fc70bbe1d74677"), "code" : "eede6be6-f716-4e2e-bf81-
885f0a16a50c", "model" : "Boeing 737-9
00", "minRunwayLength" : 2975, "range" : 5600, "capacity" : 215 }
{ "_id" : ObjectId("5e77b3fed0fc70bbe1d74678"), "code" : "51192f6b-9c26-4ef9-b843-
cf241f326091", "model" : "Embraer E-17
5", "minRunwayLength" : 1261, "range" : 4000, "capacity" : 80 }

Type “it” for more

> it

{ "_id" : ObjectId("5e77b3fed0fc70bbe1d7467e"), "code" : "00126a63-f342-4ccd-ba86-
4a7beecf10c0", "model" : "Airbus A350"
, "minRunwayLength" : 3200, "range" : 15000, "capacity" : 300 }
```

Querying All Items

Formatting the Output

```
> db.aircraft.find().pretty()

{
    "_id" : ObjectId("5e77b3fed0fc70bbe1d74677"),
    "model" : "Boeing 737-900",
    "minRunwayLength" : 2975,
    "range" : 5600,
    "capacity" : 215
}
{
    "_id" : ObjectId("5e77b3fed0fc70bbe1d74678"),
    "model" : "Embraer E-175",
    "minRunwayLength" : 1261,
    "range" : 4000,
    "capacity" : 80
}
```

The Collection Find Method

```
find(query, projection): cursor
```

INPUT:

query	(Optional) Filtering using query operators
projection	(Optional) Specify which fields to display/hide

OUTPUT:

cursor	Cursor to the documents that match the query criteria

Projection

Limit the amount of data sent from the database by eliminating or including specific fields

Document Projection Structure

```
find({}, {field1: val1, field2: val2})
```

INPUT:

field	The name of the field to be included or excluded
val	1 for inclusion, 0 for exclusion

Include / Exclude Fields

```
> db.aircraft.find({}, {model: 1, range: 1})

{ "_id" : ObjectId("5e77b3fed0fc70bbe1d74677"), "model" : "Boeing 737-900", "range"
: 5600 }
{ "_id" : ObjectId("5e77b3fed0fc70bbe1d74678"), "model" : "Embraer E-175", "range"
: 4000 }

> db.aircraft.find({}, {code: 0, capacity: 0, range: 0})

{ "_id" : ObjectId("5e77b3fed0fc70bbe1d74677"), "model" : "Boeing 737-900",
"minRunwayLength" : 2975 }
{ "_id" : ObjectId("5e77b3fed0fc70bbe1d74678"), "model" : "Embraer E-175",
"minRunwayLength" : 1261 }

> db.aircraft.find({}, {model: 1, _id: 0})

{ "model" : "Boeing 737-900" }
{ "model" : "Embraer E-175" }
```

You can not include and exclude fields in the same projection, except the _id field

Cursor

A virtual object where MongoDB stores the documents returned by the find method

```
db.aircraft.find().pretty()
```

- Format documents to make them readable

```
db.aircraft.find().limit(5)
```

- Limit the output of the find method to 5 documents

```
db.aircraft.find().skip(3)
```

- Skip the first 3 documents returned by the find method

```
db.aircraft.find().sort({…})
```

- Sort the output of the find method using sorting criteria

```
db.aircraft.find().count()
```

- Count the output of the find method

Pagination

```
> db.aircraft.find().limit(2)

{ "_id" : ObjectId("5e77b3fed0fc70bbe1d74677"), "code" : "eede6be6-f716-4e2e-bf81-
885f0a16a50c", "model" : "Boeing 737-9
00", "minRunwayLength" : 2975, "range" : 5600, "capacity" : 215 }
{ "_id" : ObjectId("5e77b3fed0fc70bbe1d74678"), "code" : "51192f6b-9c26-4ef9-b843-
cf241f326091", "model" : "Embraer E-17
5", "minRunwayLength" : 1261, "range" : 4000, "capacity" : 80 }

> db.aircraft.find().skip(2).limit(2)

{ "_id" : ObjectId("5e77b3fed0fc70bbe1d7467b"), "code" : "4f356f56-84dd-484f-a5f7-
b960dfba5823", "model" : "Boeing 747",
"minRunwayLength" : 3100, "range" : 14000, "capacity" : 467 }
{ "_id" : ObjectId("5e77b3fed0fc70bbe1d7467c"), "code" : "a3faaef2-fe54-4949-928f-
be93584da471", "model" : "Airbus A319"
, "minRunwayLength" : 2255, "range" : 6900, "capacity" : 124 }

> db.aircraft.find().skip(4).limit(2)
```

MongoDB does not guarantee the order of the returned documents unless you use sort()

The Cursor Sort Method

```
find({},{}).sort({field1: val1, field2: val2})
```

INPUT:

field	The name of the field(s) you want to sort by
val	1 for ascending order, -1 for descending order

Sorting

```
> db.aircraft.find({}, {model: 1, _id: 0}).sort({model: 1})

{ "model" : "Airbus A350" }
{ "model" : "Boeing 737-400" }

> db.aircraft.find({}, {model: 1, range: 1, _id: 0}).sort({range: -1})

{ "model" : "Boeing 747", "range" : 14000 }
{ "model" : "Airbus A319", "range" : 6900 }

> db.aircraft.find({}, {model: 1, range: 1, _id: 0}).sort({model: 1, range: -1})

{ "model" : "Airbus A319", "range" : 6900 }
{ "model" : "Airbus A320", "range" : 6000 }
{ "model" : "Airbus A350", "range" : 15000 }
```

The Sample Data Used in This Course

Flight Management Database

Aircraft

Flights

Flight Management Database

Aircraft	Flights
code	type
model	delayed
minRunwayLength	departureDate
range	distanceKm
capacity	departure (city, country, location, etc.)
	destination (city, country, location, etc.)
	aircraftCode
	crew

```
mongoimport --file C:\aircraft.json --db flightmgmt --collection aircraft --drop

mongoimport --file C:\flights.json --db flightmgmt --collection flights --drop
```

Practice Makes Perfect

Two json files can be imported to obtain the sample data:

- Course Assets
- GitHub: https://github.com/dangeabunea/pluralsight-mongodb-queries

Demo

Writing Your First Mongo Queries

- Connect to a database
- Import the sample data
- Write queries against it

Summary

Connect to a Mongo database

Navigate through a database server

Fetch documents from any collection

- Sorting
- Paging
- Projections

“These queries are basic and not very useful for complex projects.”

You

Up Next:
Understanding Query Filters and Query Operators