MongoDB Compass: Visual interface to MongoDB. Configure a database and explore the data it contains

MongoDB Atlas: Database as a service offering

There is also something called as the Mongo Shell

In this course we will use the Mongo shell to connect to a Atlas Cluster and run queries

With Atlas I can set up database in the cloud so that I don't have to configure the fine details using the shell

Atlas manages the details of creating clusters for you.. Just like AWS in a way

Similar to AWS, Atlas also stores your data in cloud in replica sets

To connect to the mongo cluster:

```
mongo "mongodb+srv://cluster0-jxeqq.mongodb.net/test" --username m001-student -password
m001-mongodb-basics
```

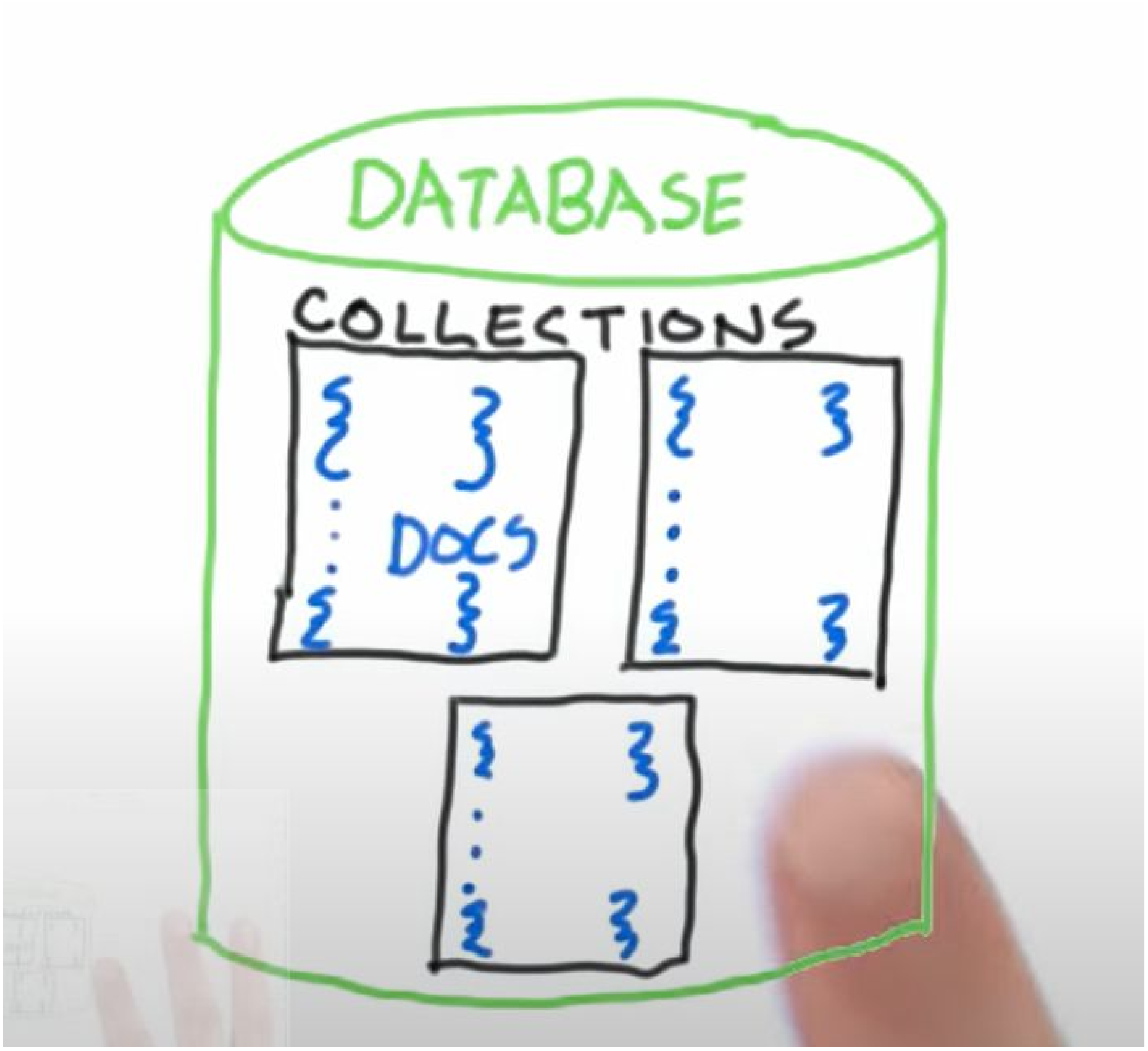
```
mongo
"mongodb://cluster0-shard-00-00-jxeqq.mongodb.net:27017,cluster0-shard-00-01-jxeqq.mongo
db.net:27017,cluster0-shard-00-02-jxeqq.mongodb.net:27017/test?replicaSet=Cluster0-shard-0"
--authenticationDatabase admin --ssl --username m001-student --password
m001-mongodb-basics
```

MongoDB is a DOCUMENT Database
So the MonGODB query language requires a little understanding of the JSON file format

Json Example:

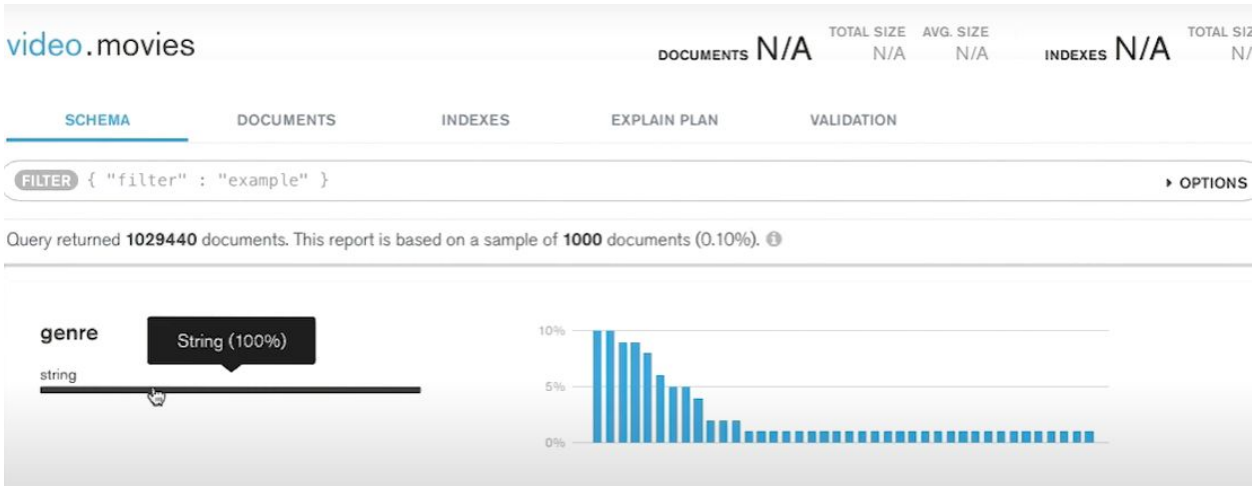
```
{
        “title”:”Mahadev”,
        “rating”:10,
        “cast”: [“Mohit Raina”, “Mouni Roy”, “Sonarika Bhadoriya”,{“name”:”Pooja
Banerjee”,”birthname”:”Puja Banerjee”}],
}
```

Date is not directly supported by JSON because it will be written as a string only



The hierarchy for database, collections and docs
We can give specific access to users on the Database and collections level but not at the doc level currently
Each Database and collection combination define a namespace
Accessed by Database.Collectioname

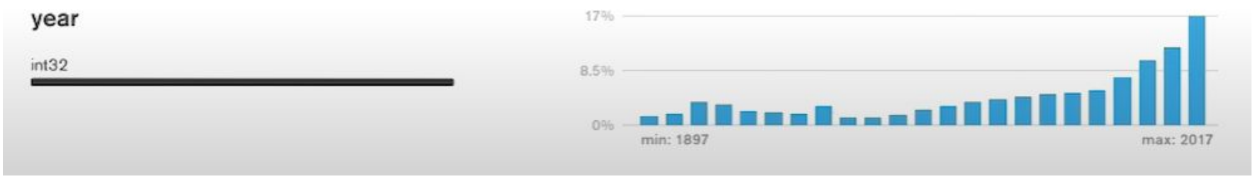
So basically one Json structure is a document.. And many documents can be a part of a collection and many collections can be a part of a database



Query returned **1029440** documents. This report is based on a sample of **1000** documents (0.10%).

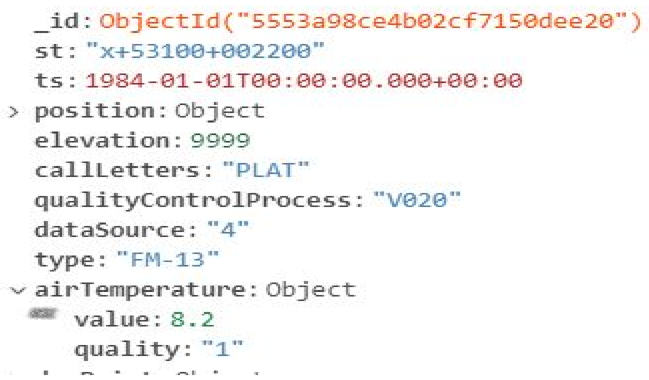
So compass has this cool feature that tells us how many documents in a particular collection have a particular key value present
So all documents in the movies collection have the genre key
This view also gives a distribution of the particular field on the right side

Like number of fields for which particular year



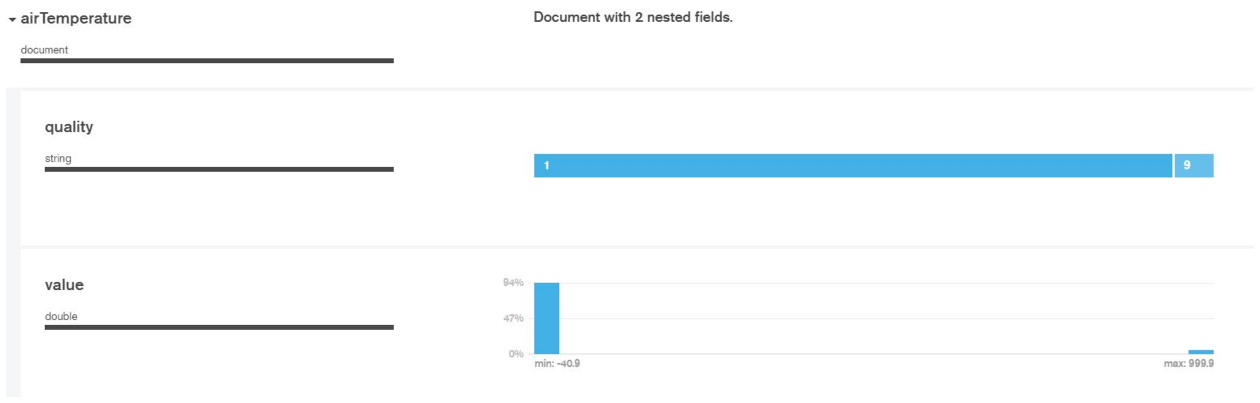
Here we can see more movies near 2017 year

I can ofcourse nest documents inside of documents:
For example this document with the field air temperature



```
  _id: ObjectId("5553a98ce4b02cf7150dee20")
  st: "x+53100+002200"
  ts: 1984-01-01T00:00:00.000+00:00
> position: Object
  elevation: 9999
  callLetters: "PLAT"
  qualityControlProcess: "V020"
  dataSource: "4"
  type: "FM-13"
v airTemperature: Object
    value: 8.2
    quality: "1"
```

This is depicted in compass as follows:



Because of the flexible data model of MongoDB, all documents do not have to contain all the fields

An array field in a document can also be an array of documents!

How to query a MongoDB database?

{'end station name': '6 Ave & Broome St'}
This query will give all documents where end station name is 6 Ave
So this is a equality query

For a range of comparison like greater than x and smaller than y.. The format is as follows:
'birth year': {$gte: 1970,$lt: 1975}
So gte is greater than or equal to and lt is less than
In Mongo keys like $gte and $lt are operators. It defines many such operators

If we want both of these together then as follows:
{'end station name': '6 Ave & Broome St','birth year': {$gte: 1970,$lt: 1975}}

CRUD: Create Read Update Delete

We created a new cluster in our MongoDB login called sandbox and we gave it access from everywhere and read write privileges in the database access layer. This is so that we can learn easily

The connection string used was: in the cmd because now we are using the mongo shell

mongo "mongodb+srv://sandbox.xulan.mongodb.net/test" --username m001-student --password m001-mongodb-basics

Now if we do “show dbs” in this we can see what databases are already present in this cluster.

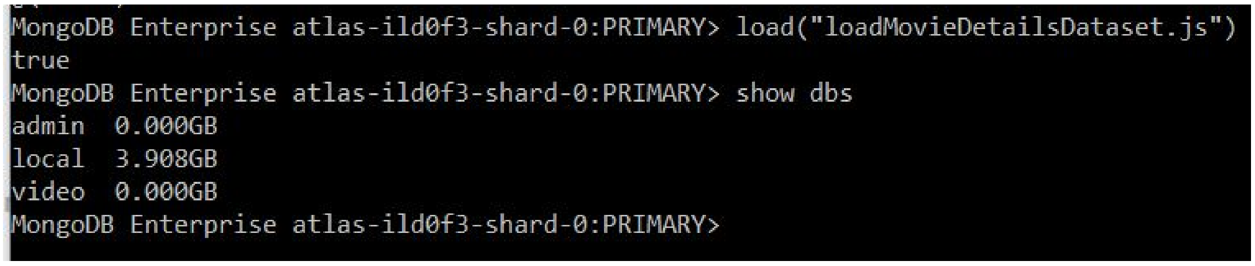
Now we want to add a video database:

CMD command - pwd gives the current directory that we are in

So wherever I launch the mongo shell from.. It will have access to the files in that directory. It will be the working directory for that mongo shell instance

The mongo shell is a fully functional javascript interpreter

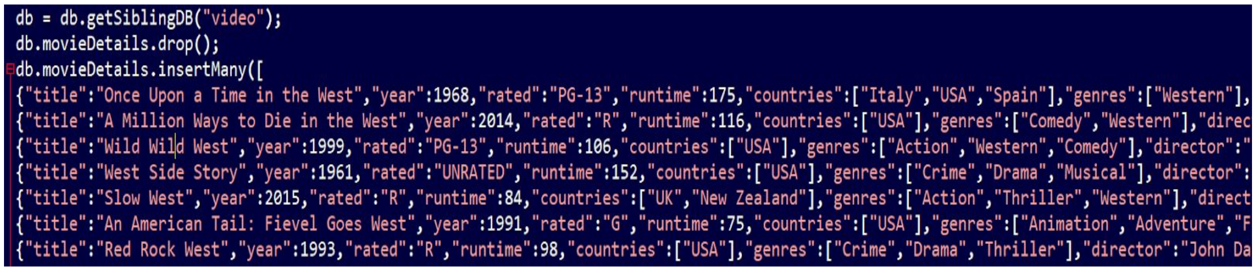
So basically we go into the folder where we unzipped the JS file and



```
MongoDB Enterprise atlas-ild0f3-shard-0:PRIMARY> load("loadMovieDetailsDataset.js")
true
MongoDB Enterprise atlas-ild0f3-shard-0:PRIMARY> show dbs
admin  0.000GB
local  3.908GB
video  0.000GB
MongoDB Enterprise atlas-ild0f3-shard-0:PRIMARY>
```

Now the video database is loaded in our database

The JS file contains the following code:



```
db = db.getSiblingDB("video");
db.movieDetails.drop();
db.movieDetails.insertMany([
{"title":"Once Upon a Time in the West","year":1968,"rated":"PG-13","runtime":175,"countries":["Italy","USA","Spain"],"genres":["Western"],
{"title":"A Million Ways to Die in the West","year":2014,"rated":"R","runtime":116,"countries":["USA"],"genres":["Comedy","Western"],"direc
{"title":"Wild Wild West","year":1999,"rated":"PG-13","runtime":106,"countries":["USA"],"genres":["Action","Western","Comedy"],"director":"
{"title":"West Side Story","year":1961,"rated":"UNRATED","runtime":152,"countries":["USA"],"genres":["Crime","Drama","Musical"],"director":
{"title":"Slow West","year":2015,"rated":"R","runtime":84,"countries":["UK","New Zealand"],"genres":["Action","Thriller","Western"],"direct
{"title":"An American Tail: Fievel Goes West","year":1991,"rated":"G","runtime":75,"countries":["USA"],"genres":["Animation","Adventure","F
{"title":"Red Rock West","year":1993,"rated":"R","runtime":98,"countries":["USA"],"genres":["Crime","Drama","Thriller"],"director":"John Da
```

Just a few documents with the structure {},{},{}

To use a database from the cmd we can say
Use videos (database name)

And now to view which collections it has:
Show collections

If I want to look at the data, then **db.collectionname.find()**
Or **db.collectionname.find().pretty()**

The pretty command will just format the data better!

Right now we connected the shell to our Atlas cluster
We can also connect the Compass to our Atlas cluster

We use the primary cluster name as the Hostname

The method to insert data through Mongo shell is called as **insertOne()**. InsertOne() is a collection level function

We can directly insert values in a collection through the Mongoshell by the inserOne and insertMany() functions as follows:

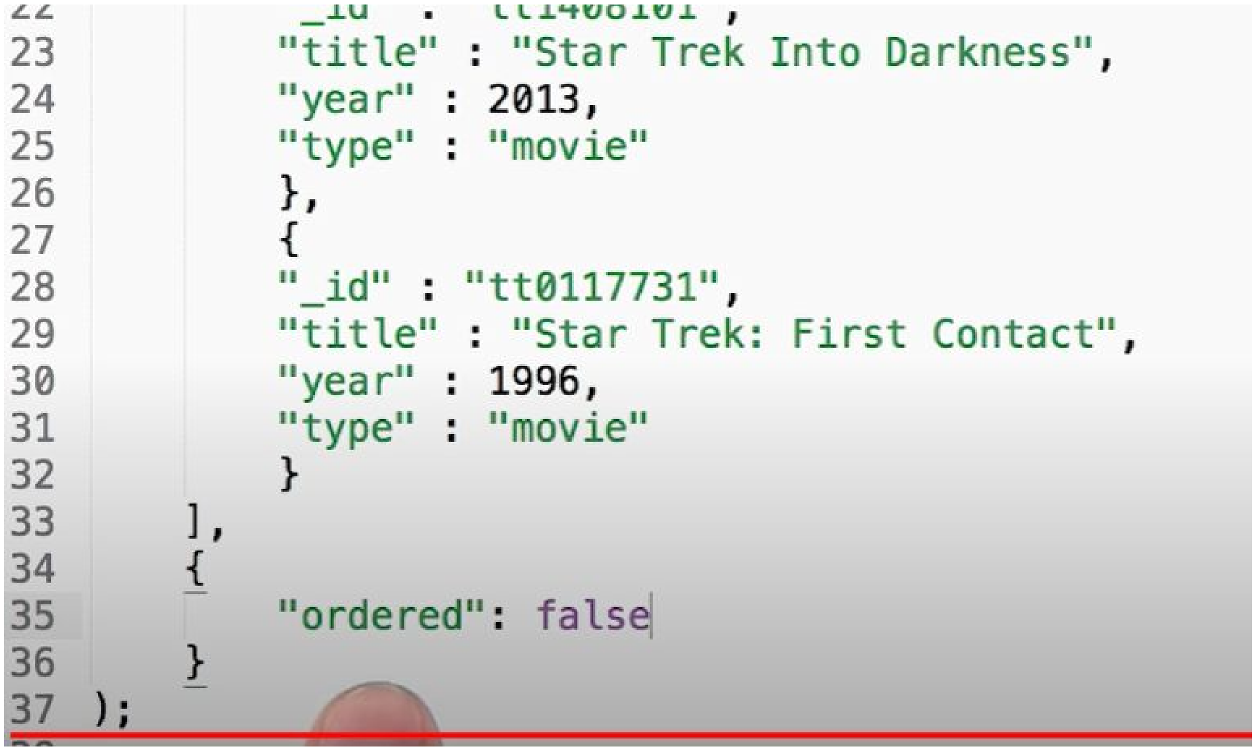
db.movieScratch.insertOne({"title":"Star Trek","year":1968})

Here we used database first like use video and then ran this.. So it inserted an entry!

ONE IMP THING:

Every document in a particular collection has an _id field. If we do not supply this field on our own then Mongo creates it for us while inserting any document! The id is like the primary key and has to be unique

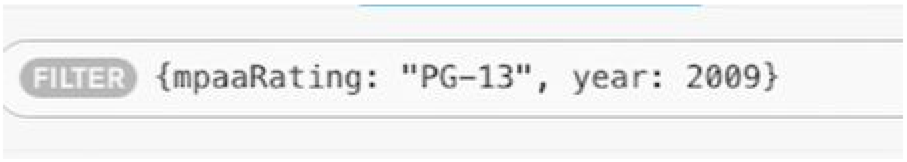
insertMany() function will keep inserting documents until it encounters an error.. Then it will stop and throw an exception



```
            "title" : "Star Trek Into Darkness",
            "year" : 2013,
            "type" : "movie"
            },
            {
            "_id" : "tt0117731",
            "title" : "Star Trek: First Contact",
            "year" : 1996,
            "type" : "movie"
            }
        ],
        {
            "ordered": false
        }
);
```

We can give another parameter in {} saying ordered is false.. Then it will insert all the others which do not generate an error! Like sql server basically

READ:
 In compass to query I just have to put the filter in the schema tab and I can get the results



```
{mpaaRating: "PG-13", year: 2009}
```

To do this in the mongo shell we have to use find() function!

If I want to search for things which are embedded in a field
Like angle and direction in the wind field Then I use the . dot operator

{wind.direction.angle:300}

If I want to do a count in the mongo shell:

 db.movieDetails.find({"awards.wins":2,"awards.nominations":2}).count()

Reading documents with Array fields:

So normally if I want to find if an element exists in an array field then:

db.movies.find({“cast”:“Shahrukh Khan”})

But the different part is that if I include 2 or more elements here, then it will only search for documents which have only both these values and not the rest

db.movies.find({“cast”:[“Shahrukh Khan”,”Salman Khan”]})

To search for a particular field at a particular position:
db.movies.find({“cast.0”:[“Shahrukh Khan”]})
Thus the position where I want the name to be is enclosed in the key!

Although we also have the operator in here!:

```
{ field: { $in: [<value1>, <value2>, ... <valueN> ] } }
```

{"actors":{$in:["Jeff Bridges","Anna Felix","Robert Downey Jr."]}}

So basically if I search in a [] it will only search for arrays with those elements/element

Which movies list Western as second genre:
{"genres.1":"Western"}

MongoDB will return the results to the client. Our mongo shell is a client.

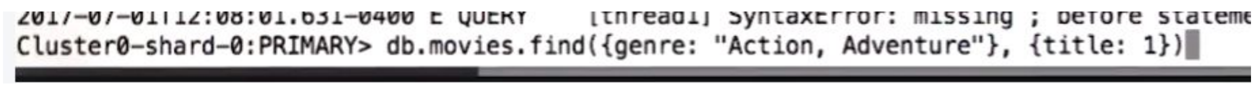
Cursor is used to iterate through the client

In search the cursor iterates a default of 20 times, that means it gives 20 results.

Projections:

Projections reduce network overhead and processing requirements by limiting the fields that are returned in the results doc.

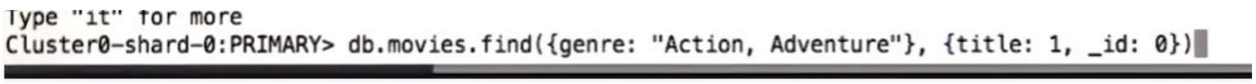
Projection can be defined as a field to the find() argument. By default Mongo returns all the fields in a doc



```
2017-07-01T12:08:01.631-0400 E QUERY    [thread1] SyntaxError: missing ; before stateme
Cluster0-shard-0:PRIMARY> db.movies.find({genre: "Action, Adventure"}, {title: 1})
```

This will just return the title in all docs

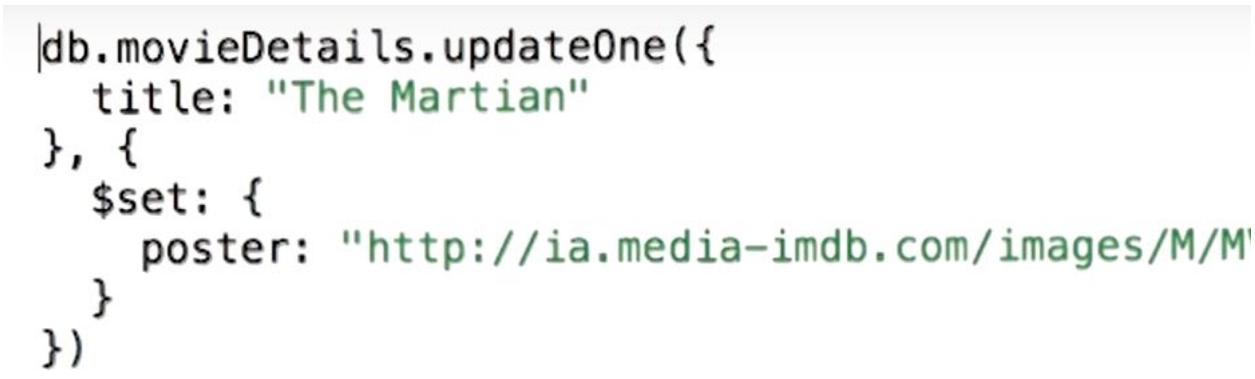
So we can include a field using 1 and exclude a field using 0!!



```
Type "it" for more
Cluster0-shard-0:PRIMARY> db.movies.find({genre: "Action, Adventure"}, {title: 1, _id: 0})
```

I can also exclude a few fields on purpose.. With only 0’s in the {}

UPDATE:



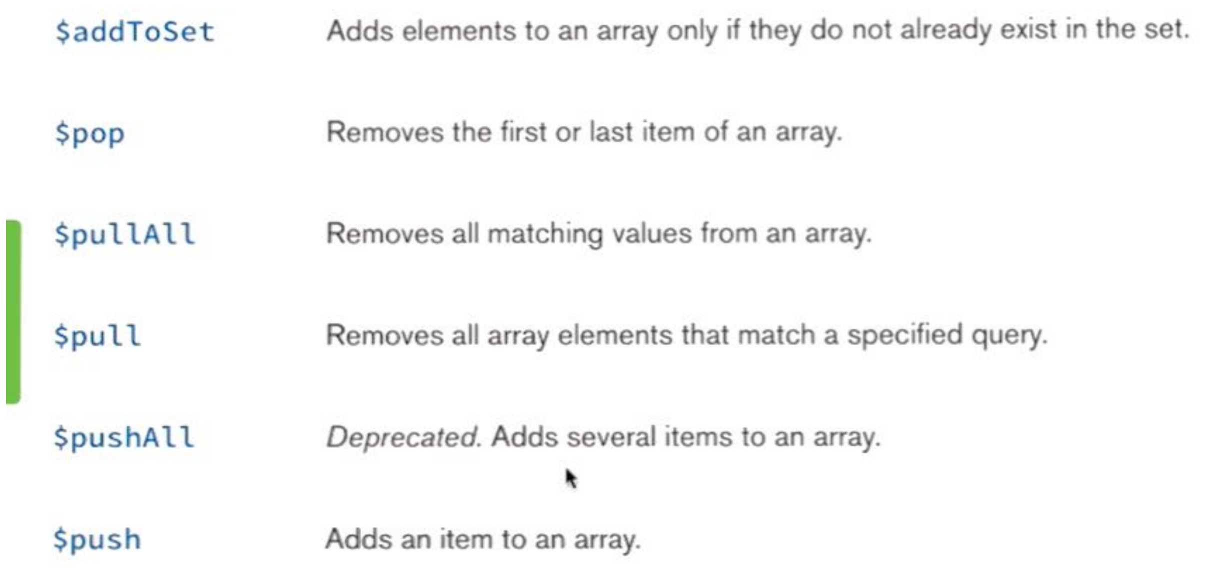
```
db.movieDetails.updateOne({
  title: "The Martian"
}, {
  $set: {
    poster: "http://ia.media-imdb.com/images/M/M
  }
})
```

Updateone will update the first doc that it finds!

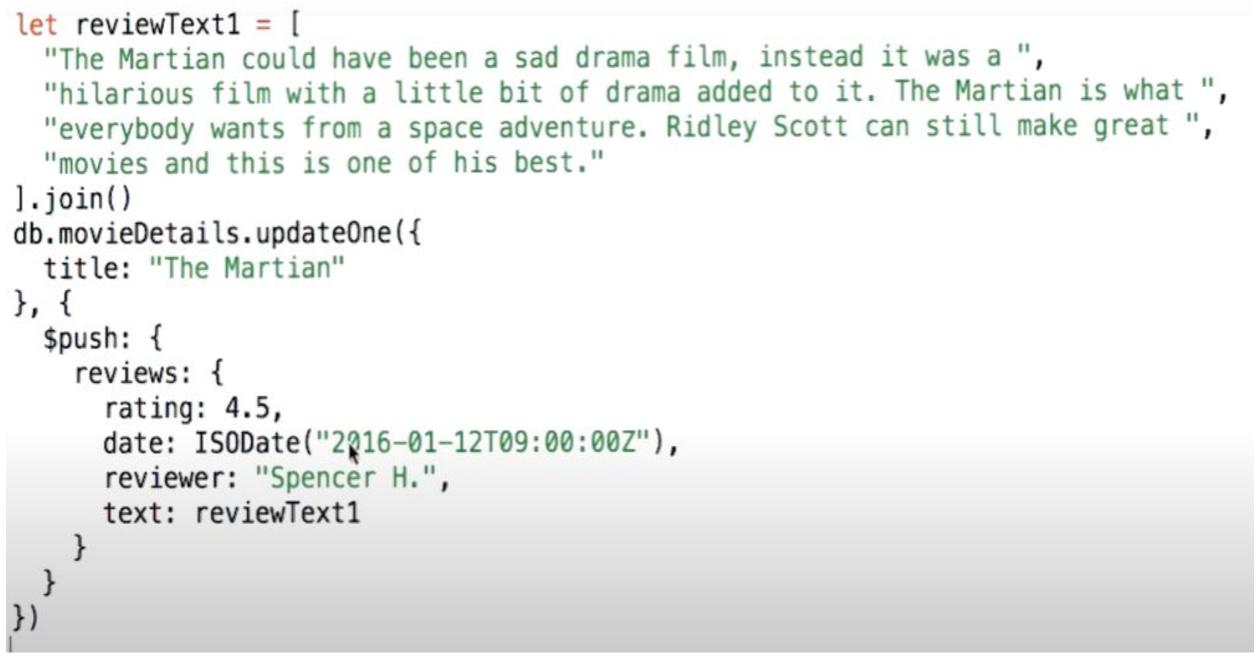
db.movieDetails.updateOne({"title":"The Girl with the Dragon Tattoo","year":2009},{$set:{"poster":"uchihaibuilding","metacritic":82}})

db.movieDetails.updateMany({"poster":"uchihaibuilding"},{$set:{"metacritic":82,"type":"movies"}})

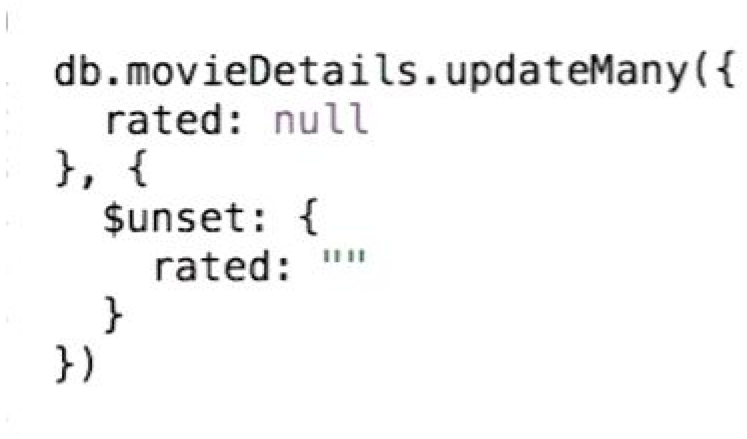
So basically I can update many entries at a time in the {} bracket for set and I can also search with many entries



`$addToSet`	Adds elements to an array only if they do not already exist in the set.
`$pop`	Removes the first or last item of an array.
`$pullAll`	Removes all matching values from an array.
`$pull`	Removes all array elements that match a specified query.
`$pushAll`	*Deprecated.* Adds several items to an array.
`$push`	Adds an item to an array.



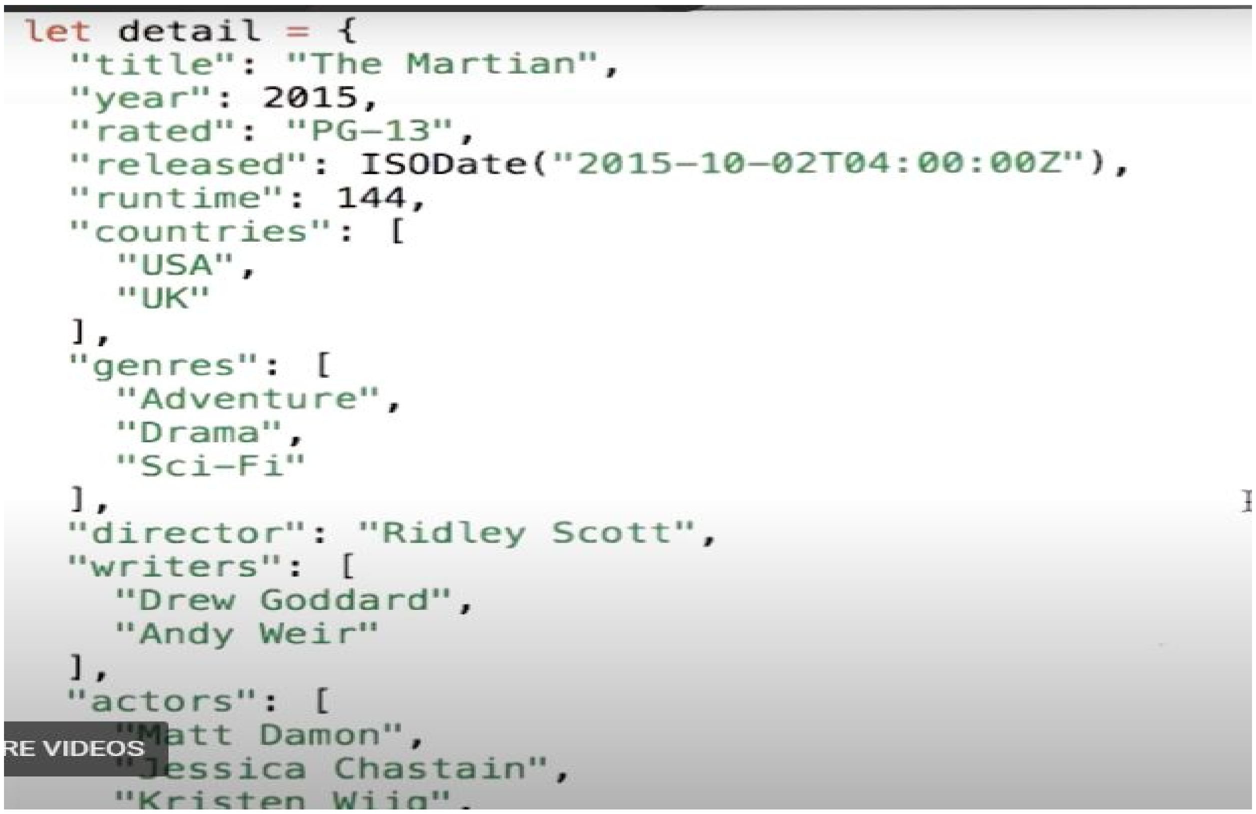
```
let reviewText1 = [
  "The Martian could have been a sad drama film, instead it was a ",
  "hilarious film with a little bit of drama added to it. The Martian is what ",
  "everybody wants from a space adventure. Ridley Scott can still make great ",
  "movies and this is one of his best."
].join()
db.movieDetails.updateOne({
  title: "The Martian"
}, {
  $push: {
    reviews: {
      rating: 4.5,
      date: ISODate("2016-01-12T09:00:00Z"),
      reviewer: "Spencer H.",
      text: reviewText1
    }
  }
})
```



```
db.movieDetails.updateMany({
  rated: null
}, {
  $unset: {
    rated: ""
  }
})
```

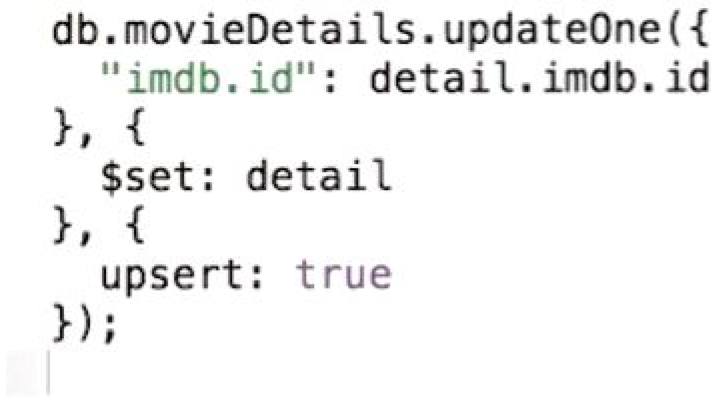
Unset operator will eliminate all fields which have been listed in it.. For docs which match the criteria. Remove the specified fields from the docs

Let detail be the data that I want to insert in the table



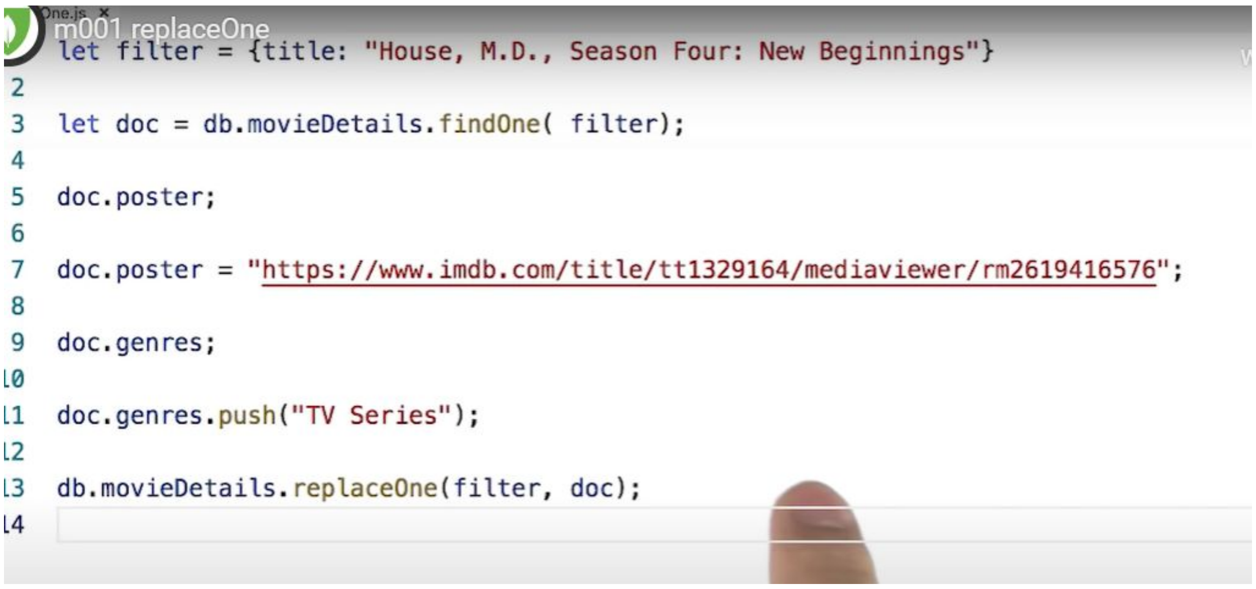
```
let detail = {
  "title": "The Martian",
  "year": 2015,
  "rated": "PG-13",
  "released": ISODate("2015-10-02T04:00:00Z"),
  "runtime": 144,
  "countries": [
    "USA",
    "UK"
  ],
  "genres": [
    "Adventure",
    "Drama",
    "Sci-Fi"
  ],
  "director": "Ridley Scott",
  "writers": [
    "Drew Goddard",
    "Andy Weir"
  ],
  "actors": [
    "Matt Damon",
    "Jessica Chastain",
    "Kristen Wiig",
```

Now I can use the upsert functionality of MongoDB to insert these new entries in my dataset



```
db.movieDetails.updateOne({
  "imdb.id": detail.imdb.id
}, {
  $set: detail
}, {
  upsert: true
});
```

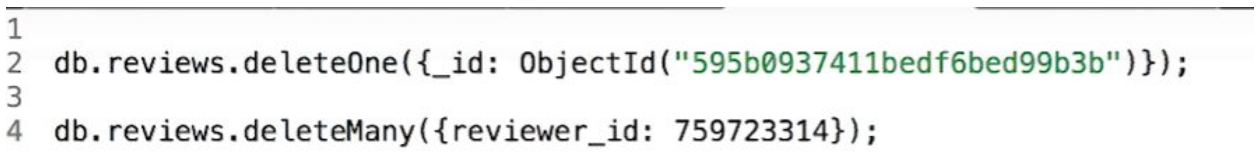
Now check the id everytime and then update the document if the id exists! And the 3rd option of upsert = True inserts a document with the particular id if that id does not exist!



```
let filter = {title: "House, M.D., Season Four: New Beginnings"}

let doc = db.movieDetails.findOne( filter);

doc.poster;

doc.poster = "https://www.imdb.com/title/tt1329164/mediaviewer/rm2619416576";

doc.genres;

doc.genres.push("TV Series");

db.movieDetails.replaceOne(filter, doc);
```

Basically create a variable filter which has the filter criteria for the document I want
Then taking all that info in the doc variable
The poster and genre commands are just to check the value before modifying.. And then use replace to replace this modified doc with the old one

DELETE



```

db.reviews.deleteOne({_id: ObjectId("595b0937411bedf6bed99b3b")});

db.reviews.deleteMany({reviewer_id: 759723314});
```

Similar to update there are two delete operators, delete one and delete many

Advanced search operators:

db.movieDetails.find({"runtime": {$gte:100,$lt:120}}).count()

Now if I do not want to print all the fields but only want to print a few of them then:

db.movieDetails.find({"runtime": {$gte:100,$lt:120}},{"title":1,"_id":0})

db.movieDetails.find({"runtime": {$gte:100,$lt:120},"tomato.meter":{$gte:95}},{"title":1,"_id":0})

If I want to include all fields except a particular value.. Like <>... then I use the $ne operator. Not equal to

db.movieDetails.find({"runtime": {$gte:150,$lt:200},"tomato.meter":{$gte:95},"rated":{$ne:"UNRATED"}},{"title":1,"_id":0})

There are also $in and $nin operators. In and not in which accept an array

There is also the $exists operator which basically checks if a field exists or not
So some examples as follows:

db.movieDetails.find({"director":{$exists:false}}).count()
Ans - 0

Here the director field is not 100%but still we get the ans as 0. This is because all the other fields are declared with null!!
So exists will only find the fields which are NOT DEFINED!

db.movieDetails.find({"director":null}).count()
Ans - 167

Now this gives values because director null will search for entries where it is actually null and where it is not defined also

Lets check this with rated!:

db.movieDetails.find({"rated":{$exists:false}}).count() - 1162
db.movieDetails.find({"rated":null}).count() - 1555

So basically the null value search includes null + undefined
And exists searches only for undefined

$TYPE:

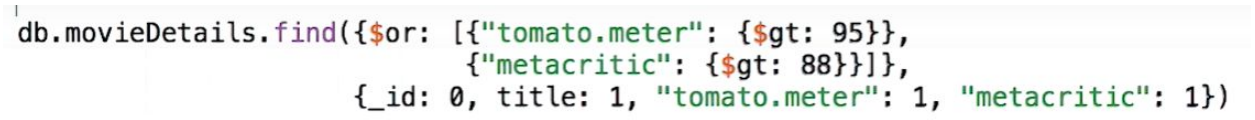
We can filter for documents that have a particular type for a field using type

db.movieDetails.find({"rated":{$type:"string"}}).count()

$OR and $AND:

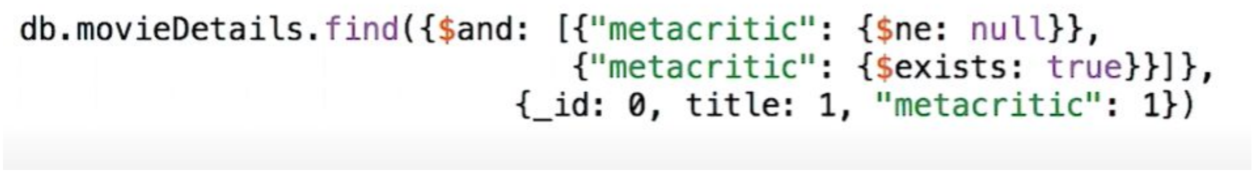
Or operator, the values can be written in a list.

And is not needed for different fields because by default the find in MongoDB is AND operator. So we only need and for querying the same operator.



```
db.movieDetails.find({$or: [{"tomato.meter": {$gt: 95}},
                            {"metacritic": {$gt: 88}}]},
                     {_id: 0, title: 1, "tomato.meter": 1, "metacritic": 1})
```

db.movieDetails.find({$or:[{"tomato.meter":{$gt:88}},{"metacritic":{$gt:88}}]}).count()

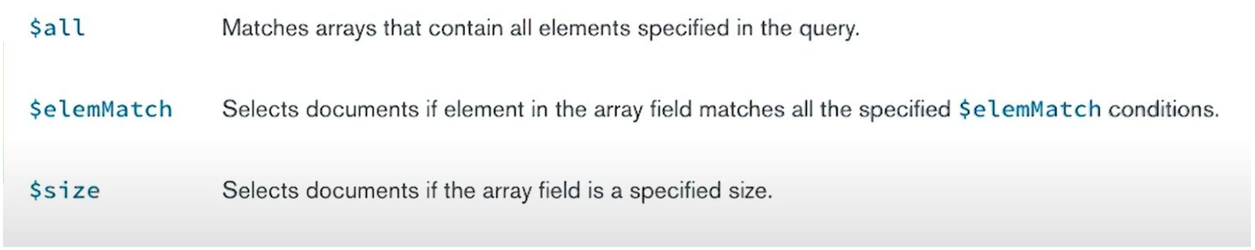


```
db.movieDetails.find({$and: [{"metacritic": {$ne: null}},
                             {"metacritic": {$exists: true}}]},
                          {_id: 0, title: 1, "metacritic": 1})
```

$and - multiple criteria on the same field

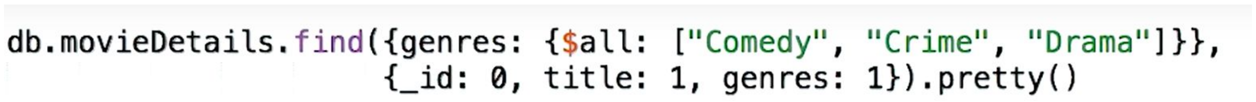
{$or:[{"watlev":"always dry"}, {"depth":0}]}

Array operators:



$all	Matches arrays that contain all elements specified in the query.
$elemMatch	Selects documents if element in the array field matches all the specified $elemMatch conditions.
$size	Selects documents if the array field is a specified size.

$all



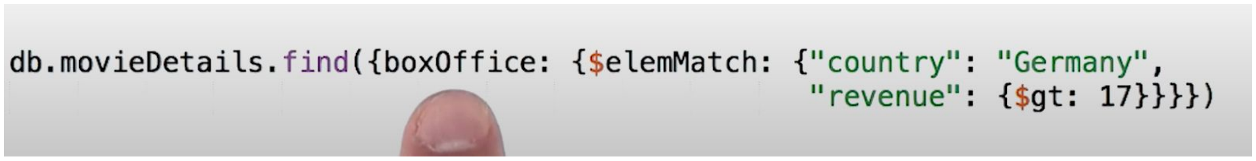
```
db.movieDetails.find({genres: {$all: ["Comedy", "Crime", "Drama"]}},
                     {_id: 0, title: 1, genres: 1}).pretty()
```

{"sections":{$all:["AG1", "MD1","OA1"]}}

$size:
Finds elements with that many number of elements in the array

{sections:{$size:2}}

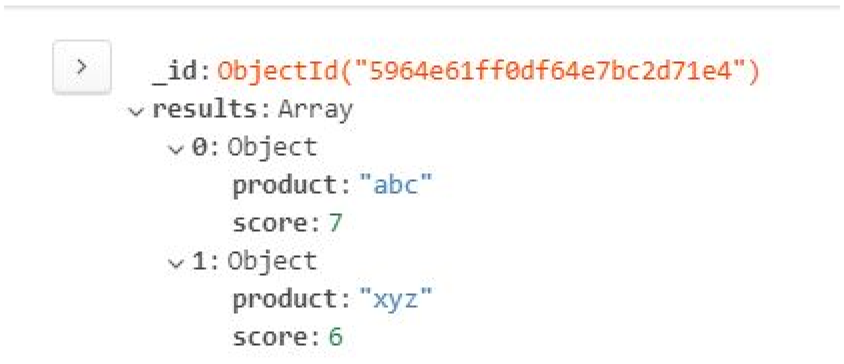
ElemMatch:
If I want to match various fields of the same element then I need the elematch function



```
db.movieDetails.find({boxOffice: {$elemMatch: {"country": "Germany",
                                               "revenue": {$gt: 17}}}})
```

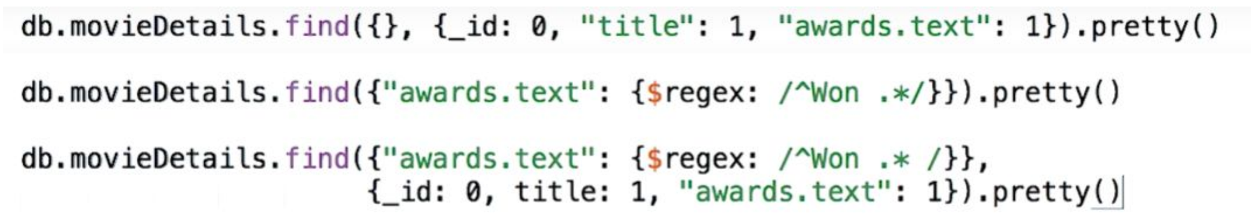
{results:{$elemMatch:{"product":"abc","score":7}}}

Here the product and score are inside the results field as follows:

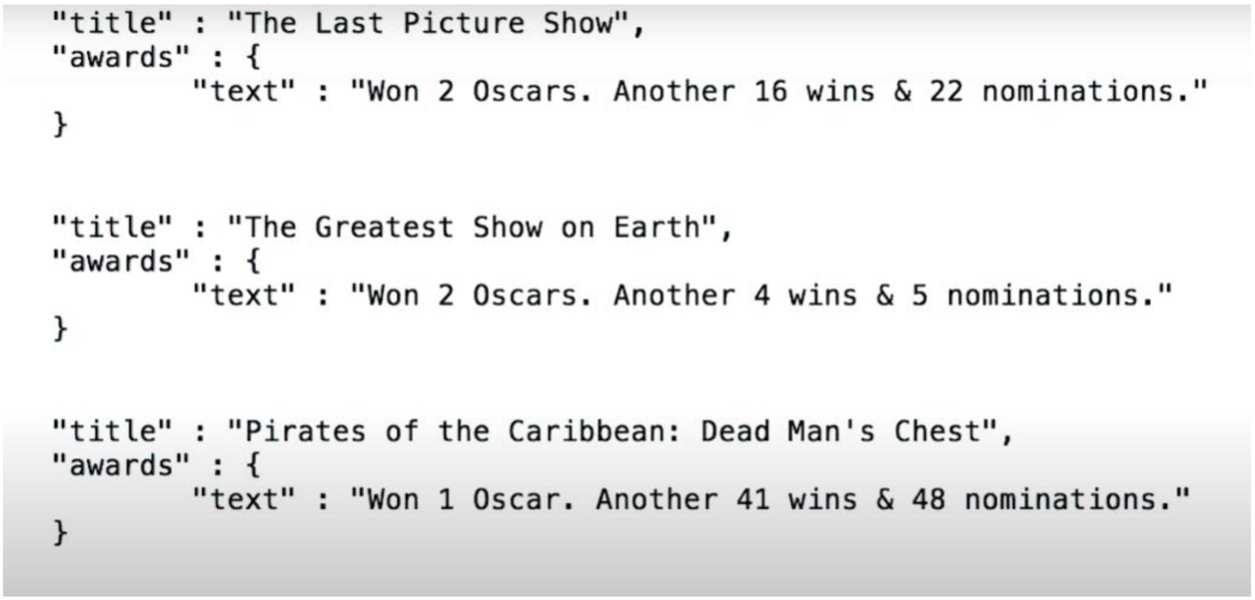


```
  _id: ObjectId("5964e61ff0df64e7bc2d71e4")
v results: Array
  v 0: Object
      product: "abc"
      score: 7
  v 1: Object
      product: "xyz"
      score: 6
```

We also have a $regex operator to use regular expressions and match fields with string values.
More comprehensive text search



```
db.movieDetails.find({}, {_id: 0, "title": 1, "awards.text": 1}).pretty()

db.movieDetails.find({"awards.text": {$regex: /^Won .*/}}).pretty()

db.movieDetails.find({"awards.text": {$regex: /^Won .* /}},
                     {_id: 0, title: 1, "awards.text": 1}).pretty()
```



```
  "title" : "The Last Picture Show",
  "awards" : {
          "text" : "Won 2 Oscars. Another 16 wins & 22 nominations."
  }


  "title" : "The Greatest Show on Earth",
  "awards" : {
          "text" : "Won 2 Oscars. Another 4 wins & 5 nominations."
  }


  "title" : "Pirates of the Caribbean: Dead Man's Chest",
  "awards" : {
          "text" : "Won 1 Oscar. Another 41 wins & 48 nominations."
  }
```

How many documents in the citibike.trips collection have the key tripduration set to null? Ignore any documents that do not contain the tripduration key.

{$and: [{tripduration:null},{tripduration:{$exists:true}}]}

How many movies match the following criteria? - The cast includes either of the following actors: "Jack Nicholson", "John Huston". - The viewerRating is greater than 7. - The mpaaRating is "R".

{$or:[{cast:"Jack Nicholson"},{cast:"John Huston"}], viewerRating:{$gt:7}, mpaaRating:"R"}