**Section 6: Finding Documents (19min)**

- Find Methods find() and findOne()

- Finding Documents

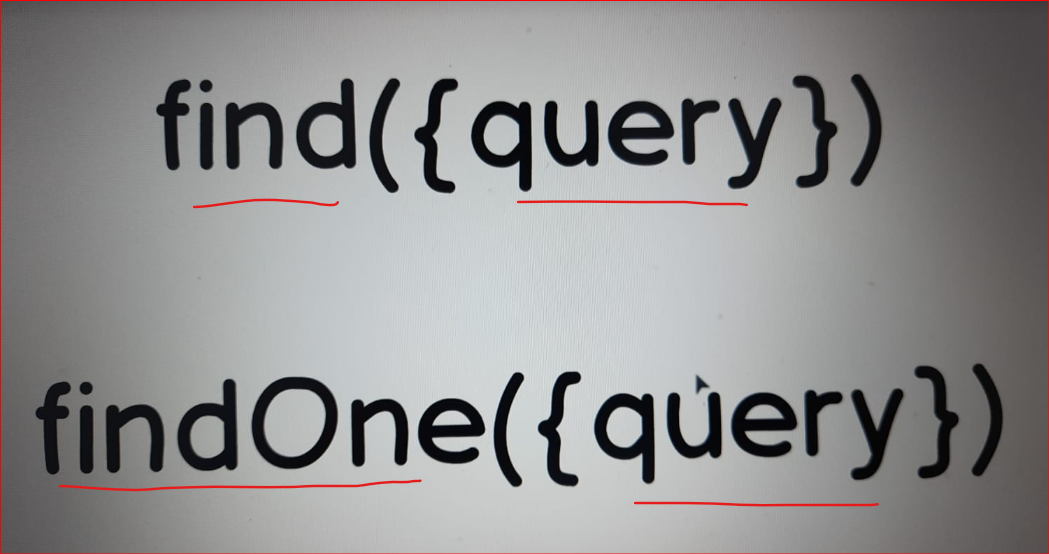
- Query Operators Overview

- Finding Documents with Query Operators

- Sort, Limit and Skip helper methods

- Sort, Limit and Skip

**24. PRACTICE - Finding Documents**



In this section we will talk about read operations

in MongoDB. And you can use two methods.

"find" and "findOne". From the name of those methods

It is clear that "findOne" will return just one document and "find" may return several documents.

In fact all that match this query.

Query is an argument that you pass to the "findOne" or "find" method.

And query is just a JavaScript object with key-value pairs. Nothing else.

And of course "find" and "findOne" methods are collection methods and need to be executed on the certain collection. In our example it is "posts" collection.

**PRACTICE - Finding Documents “2.1FindingDocuments.js”**

**// - Find Methods find() and findOne()**

**// find({Query}) Method**

db.getCollection("posts").find({}); **//find Method by default (It will show all the documents in the collection 'posts'**

**// findOne({Query}) Method**

db.getCollection("posts").findOne({});

**// Let's find specific document by it's "postId".**

**// I'll try to find this document with "postId" 3015**

db.getCollection("posts").findOne({postId: 3015}); **// or you can use postId: NumberInt(3015)**

**// Please find one document that has two comments**

db.getCollection("posts").findOne({comments: 2})

**// Please find document that has 0 comments**

db.getCollection("posts").find({comments: 0})

**// Please find all documents that were created by Emily Watson.**

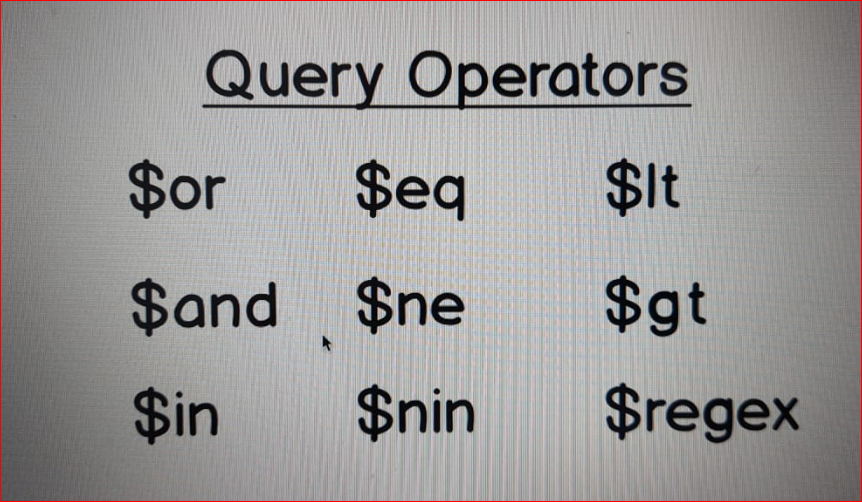
db.getCollection("posts").find({ "author.name" : "Emily Watson**"}); // if you want to access embedded property you need to add double quotes here, around the fields**

**// Please find all documents where "tags" contains "programming" string.**

db.getCollection("posts").find({ tags : "programming**"}); //You see that "tags" is an array, array of strings.**

**// It means that you can easily use such queries to find documents by array elements, by specific values or by values of the nested fields.**

**25 - Query Operators Overview**

****

**There are many different query operators that can be used in queries. And here you see some examples of them.**

* **You see that each operator is prepended by dollar sign $.**
* **And that is a signal for MongoDB Shell that you use query operator instead of just field name.**

**For example $lt operator is used when you want to find all documents with certain value of the field that is less than specified value or greater than specified value ($gt).**

**$and is used when you want to pass several conditions and only documents that match all conditions will be returned.**

**If you use $or operator then documents that match any of the conditions will be returned back.**

**Let's now try some of those query operators in action and let's do that in the next Section.**

**$in operator**

**$in operator requires an array of elements as a value.**

**And this operator is used for arrays. And it will find documents that have certain value in the array that exists in the list of the elements that you pass as a value to this $in operator.**

**26. PRACTICE - Finding Documents with Query Operators**

**// $gt**

**// Let's find all documents that have non-zero number of comments. i.e greater than 0 // 4 documents should be returned back.**

db.getCollection("posts").find({ comments : {$gt : 0}});

**// Please find all documents that have less than 5 comments.**

**// $lt**

db.getCollection("posts").find({comments : {$lt : 5}});

**// There are 5 documents which are less than 5 comments**

**// Now let's suppose that we want to find all documents that have less than 5 comments but greater than zero comments.**

**// $and**

db.getCollection("posts").find({

$and: [ // This query operator needs an array as a value, . array of conditions.

{comments: {$lt: 5}}, // I'll add two conditions to this array.

{comments: {$gt: 0}} // First condition - {comments: . . {$lt: 5}}. Then comma and next condition . . {comments: {$gt: 0}}

]

})

**// Got 2 Documents having 2 comments each(having greater than 0 and less than 5)**

**// Please use $or operator instead of $and and find all documents that are either shared,**

**// where "shared" is "true" or have "programming" tag.**

**// $or**

db.getCollection("posts").find({

$or: [

{shared: true},

{tags: "programming"}

]

})

**// we got 4 outputs, ($or operator will give either of condition to satisfy and displays the documents)**

**// $in operator**

**//Please use $in operator and this $in operator requires an array of elements as a value. And this operator is used for arrays.**

**//And it will find documents that have certain value in the array that exists in the list of the elements that you pass as a value to this $in operator.**

**// find documents where "tags" array contains either "programming" or "coding".**

db.getCollection("posts").find({

tags: {$in: [

"programming",

"coding"

]}

})

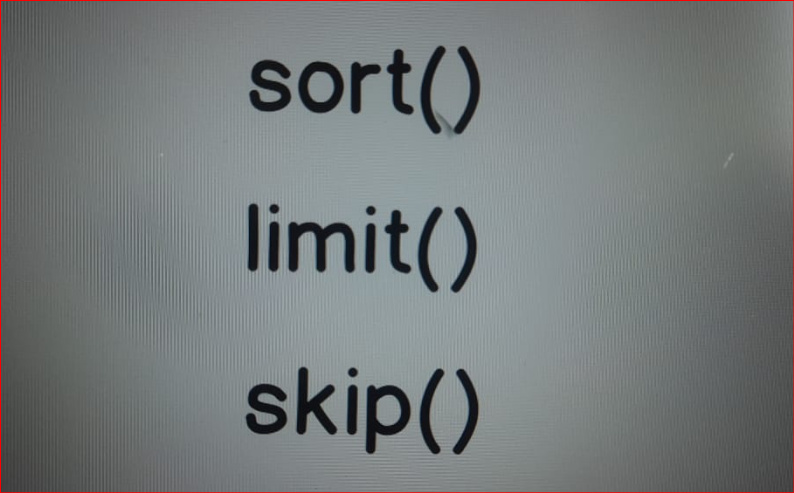
**// output:**

**// there are 3 documents which contains 'programming' or 'coding'**

**//Now you know how to use query operators in the Queries.**

**//Now let's move on and next let's talk about sort, limit and skip helper methods.**

**27. Section - Sort, Limit and Skip helper methods**

****

Let's now talk about sort, limit and skip methods. Those methods are helper methods of the "find" method.

It means that they need to be chained after the "find" method.

1. Sort will sort resulting documents.

You can sort documents by certain field and you can use either ascending or descending sort

1. Limit will limit number of the resulting documents

In the limit you should simply pass a number.

For example "limit(2)".And this will limit number of the documents in the result to 2.

1. skip will skip certain amount of the documents.

Skip same as with limit you pass a number and you will simply skip certain number of the documents.

For example five or three.

**28. PRACTICE - Sort, Limit and Skip**

**db.getCollection("posts").find({})**

**//2.Limit will limit number of the resulting documents**

**//In the limit you should simply pass a number.**

**//For example "limit(2)".And t**

**his will limit number of the documents in the result to 2.**

db.getCollection('posts').find({}).limit(2);

**//3.skip will skip certain amount of the documents.**

**//Skip same as with limit you pass a number and you will simply skip certain number of the documents.**

**//For example five or three.**

db.getCollection('posts').find({}).skip(3);

**//1.Sort will sort resulting documents.**

**//You can sort documents by certain field and you can use either ascending(use 1) or descending(use -1) sort**

**// "sort" requires an object as a value. And here in the object you need to pass key-value pair.**

**// Let's suppose that I want to sort by "comments" field.**

db.getCollection('posts').find({}).sort({comments: 1});

**// All the Documents sorted in Ascending based on Comments values**

db.getCollection('posts').find({}).sort({comments: -1});

**// All the Documents sorted in Descending based on Comments values**

**//Please try to sort by title in ascending order. Starting a,b,c...z**

db.getCollection('posts').find({}).sort({title: 1});

**// NOTE:**

**// By default documents are sorted by "\_id" and you may notice that each "ObjectID" for the next document is larger than previous "ObjectID".**

**// Please use simultaneously two methods "sort" and "skip".**

**// "skip" two elements and "sort" by "shared" field in Ascending order**

db.getCollection('posts')

.find({})

.skip(2)

.sort({shared: 1});

**// let's summarize.**

There are three helper methods for "find" method.

//They are skip, sort and limit.

//And you can use them to limit numbers of documents, to skip some documents and to sort documents.

**let's move on to the next section where we will talk about update operations.**