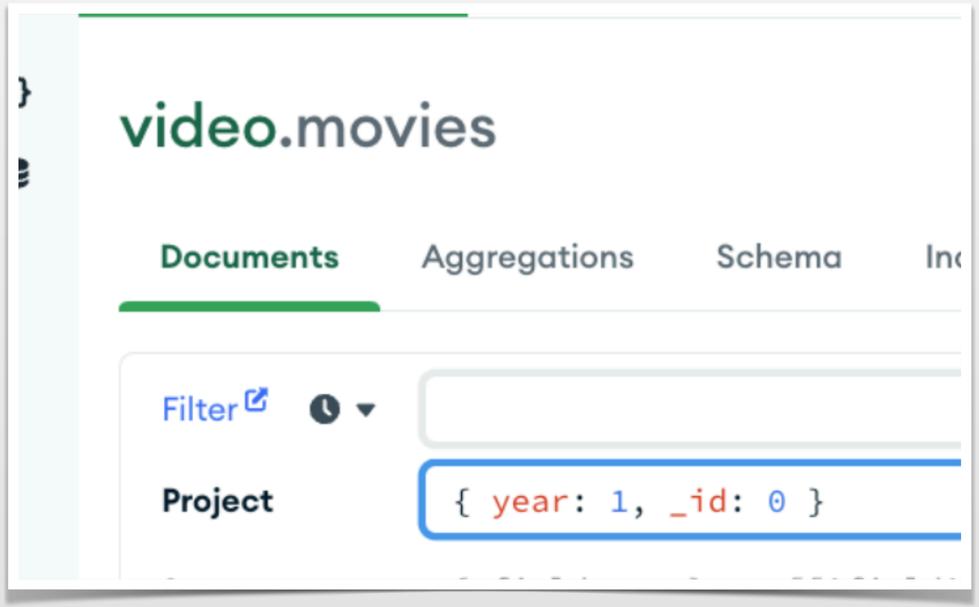
M3 - W1 - MongoDB Operations

miguel.garrido@ironhack.com

Projections

Allows you to control **which fields are returned** when querying a collection.



Sort Query

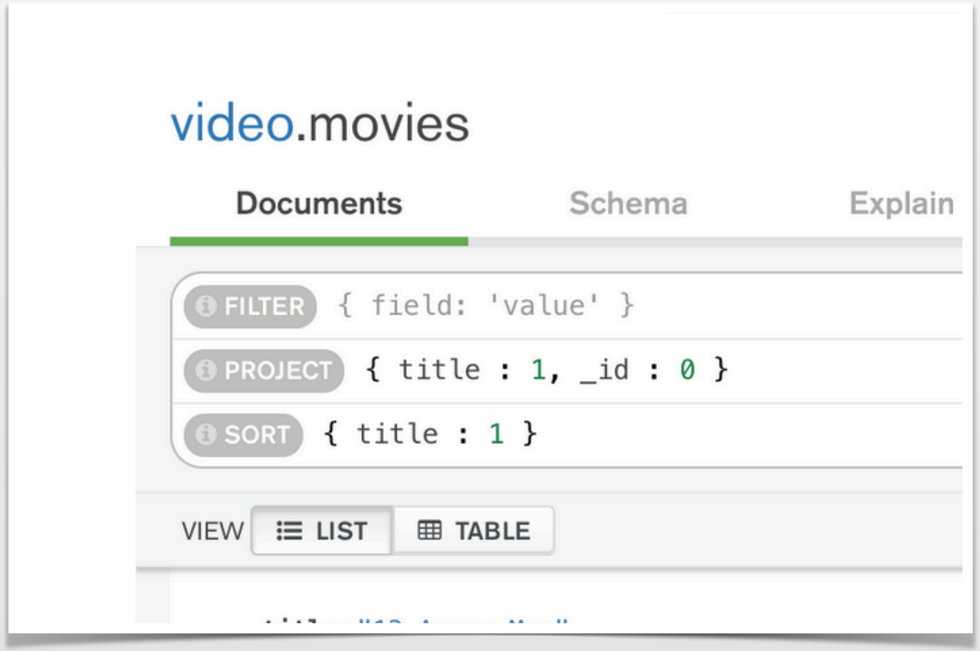
If the query bar has the Sort option, you can specify the sort order of the documents returned in the result.

Ascending = 1

Descending = -1

Example:

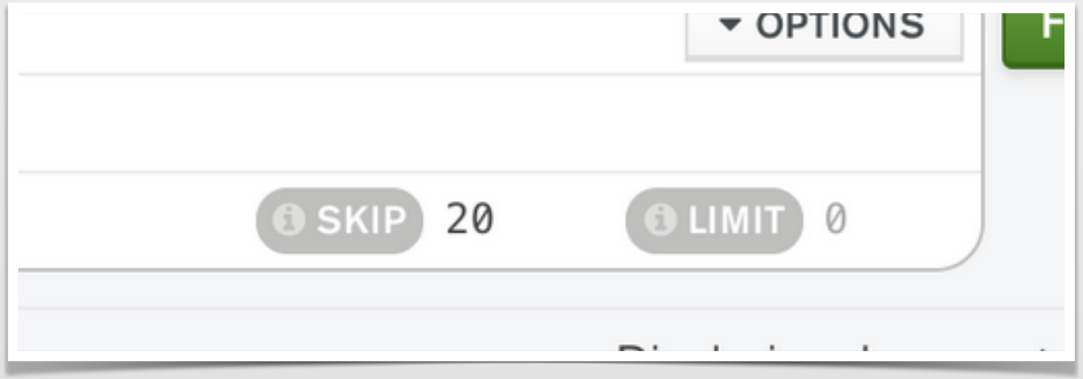
```
{ title: 1 }
```



Skip & Limit

Skip option specifies the first n-number of documents to skip before returning the result set.

Limit specifies the maximum number of documents to return in a query.



Logical Operators

A **logical operator** is a primitive construct in computer science that defines how truth values-of-propositions or conditions interrelate.

In essence, it is the **grammar of condition interaction**.

Logical Operators

Conditions $and, $or, $nor:

```
{ $cond: [ { <exp1> }, { <exp2> }, ... { <expN> } ] }
```

Examples:

```
{ $and: [ { year: "2000" }, { rate: "8.5" } ] }
```

```
{ $or: [ { year: "2000" }, { rate: "8.5" } ] }
```

Logical Operators - implicit AND

MongoDB applies an implicit **AND operation** when specifying a comma separated list of expressions.

```
{ year: "2000", rate: "8.5" }
```

Logical Operators

Conditions $eq, $ne, $gt, $gte, $lt, $lte, $exists

```
{ <field>: { $cond: <value> } }
```

Examples:

```
{ capacity : { $gte: 40 } }
```

```
{ capacity : { $lte: 20 } }
```

Logical Operators

Name	Description	Syntax
`$eq`	Matches values that are **equal** to a specified value	`{ field : { $eq: value } }`
`$ne`	Matches all values that are **not equal** to a specified value	`{ field : { $ne: value } }`
`$gt`	Matches values that are **greater than** a specified value	`{ field : { $gt: value } }`
`$gte`	Matches values that are **greater than or equal** to a specified value	`{ field : { $gte: value } }`
`$lt`	Matches values that are **less than** a specified value	`{ field : { $lt: value } }`
`$lte`	Matches values that are **less than or equal** to a specified value	`{ field : { $lte: value } }`

Logical Operators for Arrays

Conditions $in, $nin, $all

```
{ field: { $cond: [ value1, value2, ... , valueN ] } }
```

Examples:

```
{ tags: { $in: ["great food", "incredible chef"] } }
```

```
{ tags: { $nin: ["too loud"] } }
```

Logical Operators for Arrays

Conditions $elemMatch

```
{ field: { $elemMatch: { <query1>, <query2>, ... } } }
```

Examples:

```
{ grades: { $elemMatch: { grade: "A", score: { $gte: 15 } } } }
```

```
{
  grades: {
    $elemMatch: {
      grade: "A",
      score: {
        $gte: 15
      }
    }
  }
}
```

Logical Operators for Arrays

Name	Description	Syntax
`$nin`	Matches none of the values specified in an array	`{ field: { $nin: [ value1, value2, ..... , valueN ] } }`
`$in`	Matches any of the values specified in an array.	`{ field: { $in: [ value1, value2, ..... , valueN ] } }`
`$all`	Matches arrays that contain all elements specified in the query.	`{ field: { $all: [ value1, value2, ..... , valueN ] } }`
`$elemMatch`	Selects documents if an element in the array field matches all the specified `$elemMatch` conditions.	`{ field: { $elemMatch: { <query1>, <query2>, ... } } }`

Element Query Operators

Name	Description	Syntax
`$exists`	Matches documents that have the specified field.	`{ field: { $exists: <boolean> } }`
`$type`	Selects documents if a field is of the specified type.	`{ field: { $type: <BSON type> } }`

Let’s practice some queries