

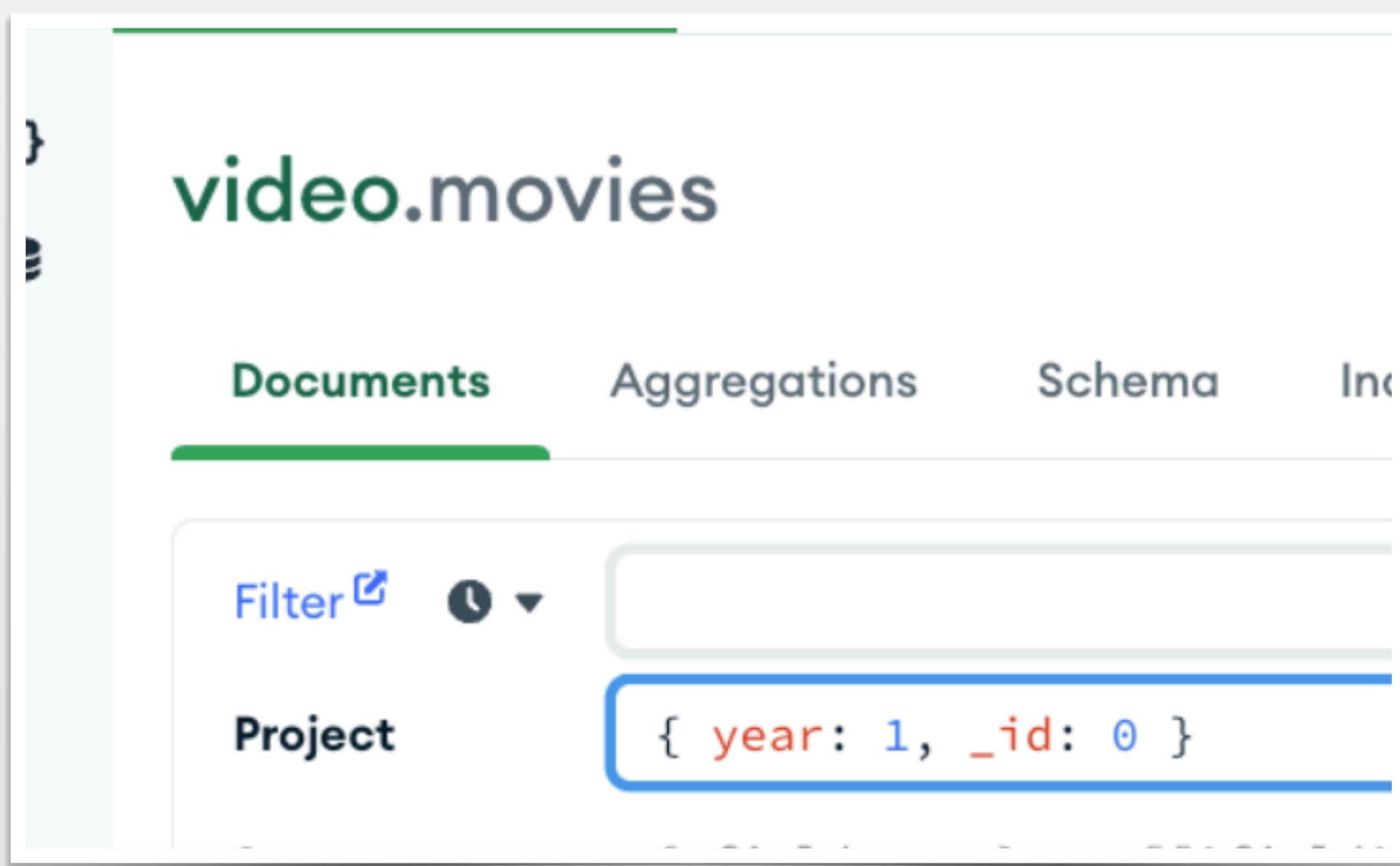
MongoDB Operations

miguel.garrido@ironhack.com



Projections

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



The screenshot shows the MongoDB Compass interface. The left sidebar has a tree view with a single node labeled 'video.movies'. The main area has a top navigation bar with tabs: 'Documents' (which is selected and highlighted in green), 'Aggregations', 'Schema', and 'Index'. Below the navigation is a search bar with a 'Filter' button and a dropdown. Underneath is a 'Project' section containing the following JSON projection:

```
{ year: 1, _id: 0 }
```

Sort Query

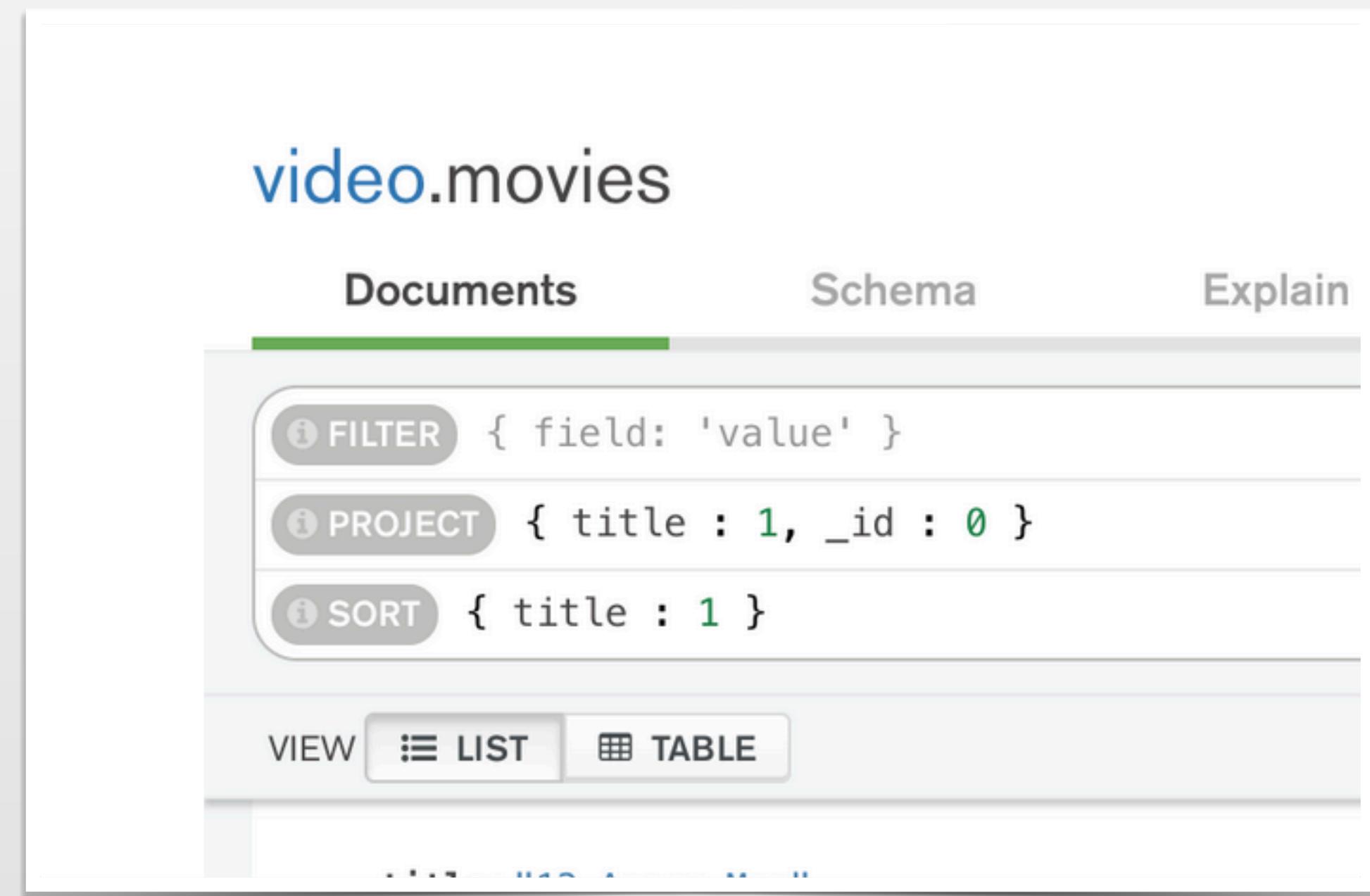
If the query bar has the Sort option, you can specify the sort order of the documents returned in the result.

Ascending = 1

Descending = -1

Example:

```
{ title: 1 }
```



The screenshot shows the MongoDB Compass interface with the collection `video.movies`. The `Documents` tab is selected. A query explain plan is displayed, showing the following stages:

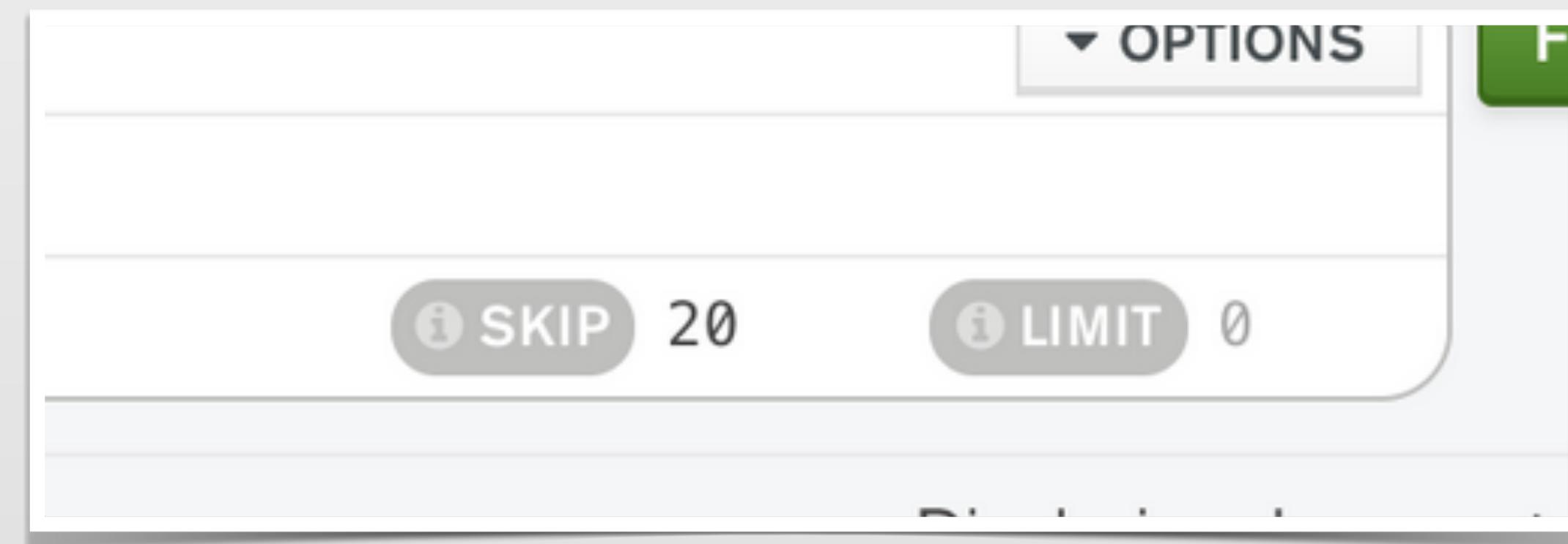
- FILTER** { field: 'value' }
- PROJECT** { title : 1, _id : 0 }
- SORT** { title : 1 }

At the bottom, there are buttons for `VIEW`, `LIST` (which is selected), and `TABLE`.

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
<code>\$eq</code>	Matches values that are equal to a specified value	<code>{ field : { \$eq: value } }</code>
<code>\$ne</code>	Matches all values that are not equal to a specified value	<code>{ field : { \$ne: value } }</code>
<code>\$gt</code>	Matches values that are greater than a specified value	<code>{ field : { \$gt: value } }</code>
<code>\$gte</code>	Matches values that are greater than or equal to a specified value	<code>{ field : { \$gte: value } }</code>
<code>\$lt</code>	Matches values that are less than a specified value	<code>{ field : { \$lt: value } }</code>
<code>\$lte</code>	Matches values that are less than or equal to a specified value	<code>{ field : { \$lte: value } }</code>



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



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

