Mongo tutorial

Desenvolvimento de Aplicações Web (DAW2020)
Processamento e Representação da Informação (PRI2020)

José Carlos Ramalho
Nov. 2020

Mongo vs RDMS

Mongo DB	RDBMS: MySQL, PostGres, SQLlite, ...
Database	Database
Collection	Table
Document	Record, Row, Tuple
Field	Column
Embedded documents	Table join
Primary Key (Default key _id provided by MongoDB itself)	Primary Key

Example document

```
{    "id":"FBR07-IJDL",
     "type": "article",
     "title": "An intelligent decision support system for digital
preservation",
     "authors":[
          "Miguel Ferreira",
           "Ana Alice Baptista",
           "José Carlos Ramalho"
      ],
    "year":"2007"}
```

Example document2

```
{    "id":"pri2020-e1",
     "type": "exame",
     "title": "Teste de avaliação de PRI2020" ,
     "authors":["José Carlos Ramalho" ],
     "year":"2007",
     "posts": [
          {"id": "p1", "from": "student1", "text": "Please, verify q1.",
           "comments":[{"id": "p1c1", "from": "student7", "text": "Yes!!!"} ]
     ],
}
```

Where and When to use MongoDB

- Big Data
- Content Management and Delivery
- Mobile and Social Infrastructure
- User Data Management
- Data Hub

Start

```
$ mongod --dbpath MyDataFolder &

$ mongo

   > ...
```

Dataset import

```
$ mongoimport -d database -c collection file.json
```

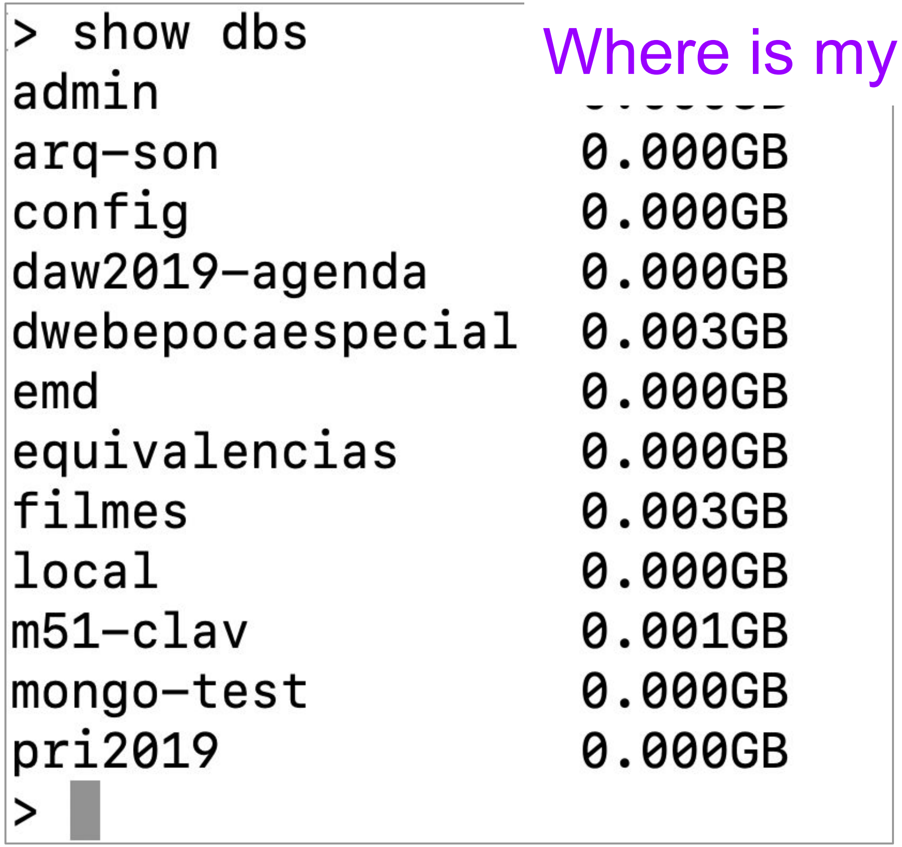
Example

```
$ mongoimport -d arq-son -c musicas arq-son.json
```

First commands

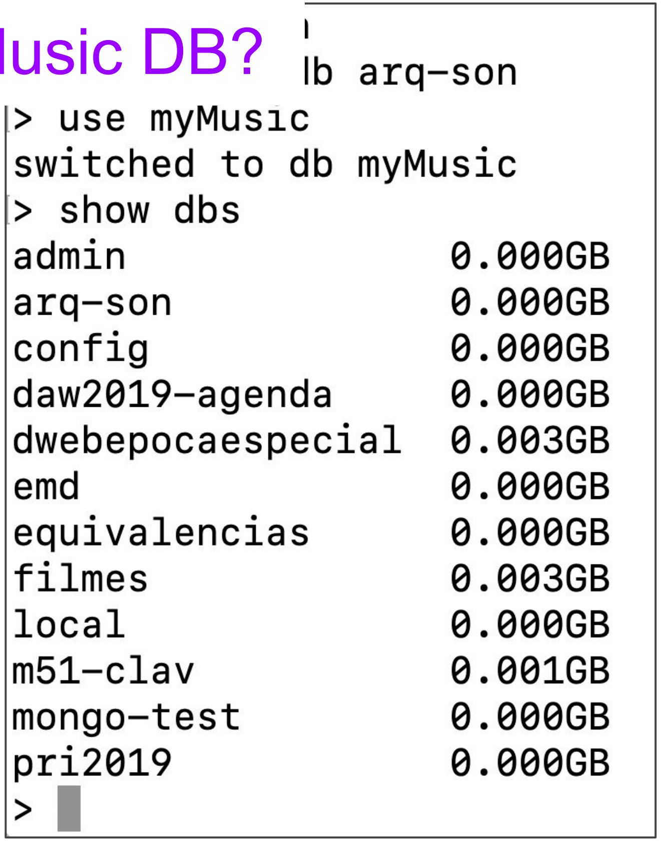
```
$ db.help()

   > ...

$ show dbs

   > ...

$ db.stats()

   > ...
```

Create / Select database



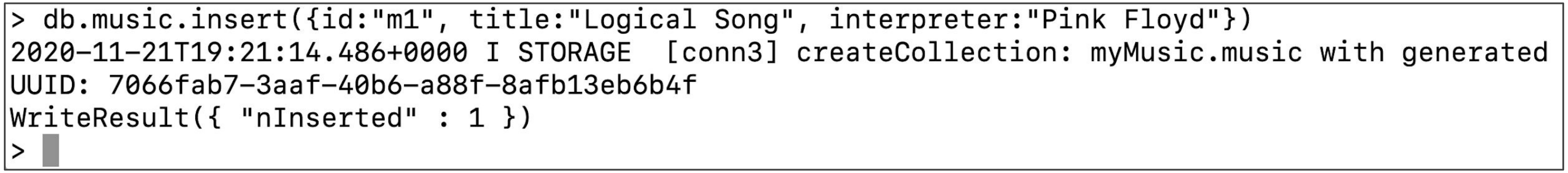
```
> show dbs
admin
arq-son              0.000GB
config               0.000GB
daw2019-agenda       0.000GB
dwebepocaespecial    0.003GB
emd                  0.000GB
equivalencias        0.000GB
filmes               0.003GB
local                0.000GB
m51-clav             0.001GB
mongo-test           0.000GB
pri2019              0.000GB
>
```

Where is myMusic DB?

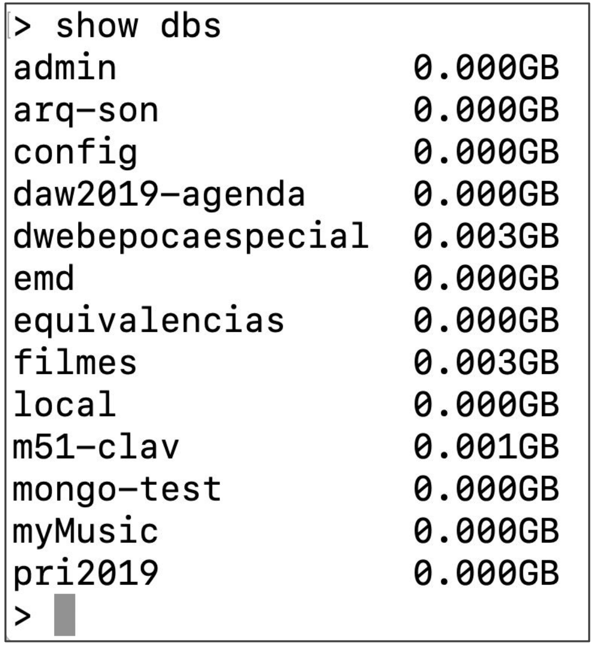


```
                 lb arq-son
> use myMusic
switched to db myMusic
> show dbs
admin                0.000GB
arq-son              0.000GB
config               0.000GB
daw2019-agenda       0.000GB
dwebepocaespecial    0.003GB
emd                  0.000GB
equivalencias        0.000GB
filmes               0.003GB
local                0.000GB
m51-clav             0.001GB
mongo-test           0.000GB
pri2019              0.000GB
>
```

Document insert / create Collection



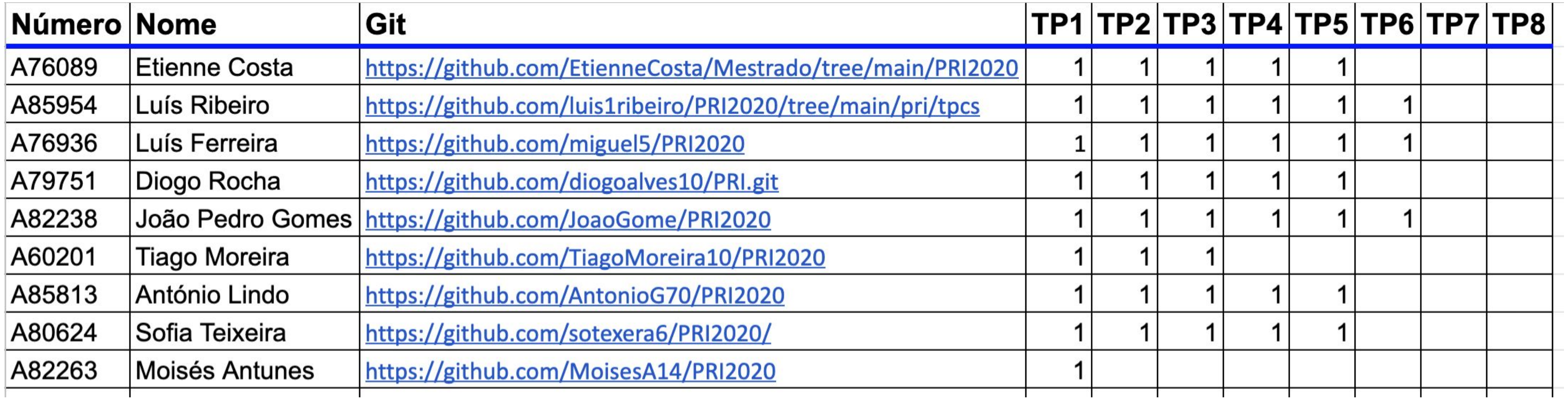
```
> db.music.insert({id:"m1", title:"Logical Song", interpreter:"Pink Floyd"})
2020-11-21T19:21:14.486+0000 I STORAGE  [conn3] createCollection: myMusic.music with generated
UUID: 7066fab7-3aaf-40b6-a88f-8afb13eb6b4f
WriteResult({ "nInserted" : 1 })
> 
```



```
> show dbs
admin                0.000GB
arq-son              0.000GB
config               0.000GB
daw2019-agenda       0.000GB
dwebepocaespecial    0.003GB
emd                  0.000GB
equivalencias        0.000GB
filmes               0.003GB
local                0.000GB
m51-clav             0.001GB
mongo-test           0.000GB
myMusic              0.000GB
pri2019              0.000GB
> 
```

```
$ db.createCollection("music")

...

$ db.createCollection("music",
{autoIndexID: true})

...
```

Exercise



Número	Nome	Git	TP1	TP2	TP3	TP4	TP5	TP6	TP7	TP8
A76089	Etienne Costa	https://github.com/EtienneCosta/Mestrado/tree/main/PRI2020	1	1	1	1	1			
A85954	Luís Ribeiro	https://github.com/luis1ribeiro/PRI2020/tree/main/pri/tpcs	1	1	1	1	1	1		
A76936	Luís Ferreira	https://github.com/miguel5/PRI2020	1	1	1	1	1	1		
A79751	Diogo Rocha	https://github.com/diogoalves10/PRI.git	1	1	1	1	1			
A82238	João Pedro Gomes	https://github.com/JoaoGome/PRI2020	1	1	1	1	1	1		
A60201	Tiago Moreira	https://github.com/TiagoMoreira10/PRI2020	1	1	1					
A85813	António Lindo	https://github.com/AntonioG70/PRI2020	1	1	1	1	1			
A80624	Sofia Teixeira	https://github.com/sotexera6/PRI2020/	1	1	1	1	1			
A82263	Moisés Antunes	https://github.com/MoisesA14/PRI2020	1							

1. Create a database and name it “PRI2020”;
2. Download this spreadsheet from BB;
3. Insert records into “work” collection (“débrouillez vous...”);
4. Confirm with listing: db.work.find(), db.work.find().pretty()

Drop database / collection

```
$ db.dropDatabase()

...

$ db.music.drop()

> true
```

insert document

Syntax

```
$ db.collectionName.insert({...})

> ...

$ db.collectionName.insert([{...}, {...}, {...}, ...])
```

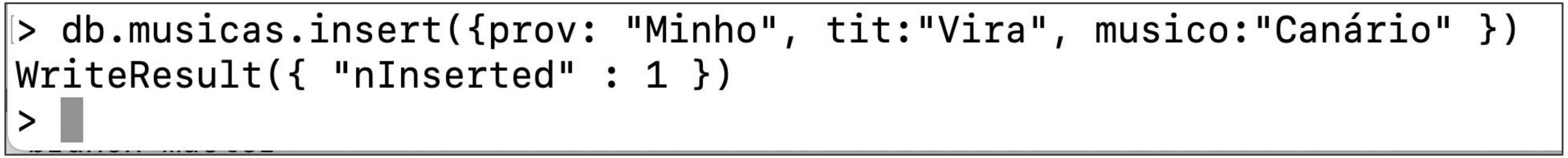
- If collection does not exist, Mongo creates it and inserts document into it;
- If document does not have an "_id" field, Mongo adds it with an unique ObjectId;
- "_id" is a 12 bytes hexadecimal number unique for every document in a collection; the 12 bytes are divided as follows:

```
id: ObjectId(4 bytes timestamp,
             3 bytes machine id,
             2 bytes process id,
             3 bytes incrementer)
```

insert document: example 1

Syntax

```
$ db.collectionName.insert({...})

> ...

$ db.collectionName.insert([{...}, {...}, {...}, ...])

> ...
```

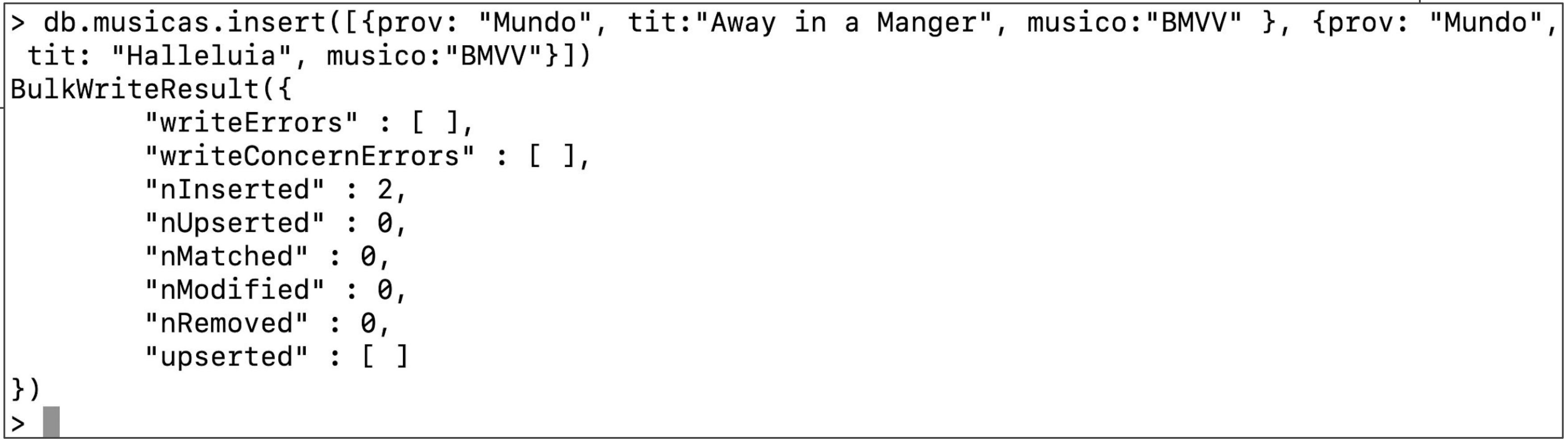


```
> db.musicas.insert({prov: "Minho", tit:"Vira", musico:"Canário" })
WriteResult({ "nInserted" : 1 })
> 
```

insert document: example 2

Syntax

```
$ db.collectionName.insert({...})

> ...

$ db.collectionName.insert([{...}, {...}, {...}, ...])

>
```



```
> db.musicas.insert([{prov: "Mundo", tit:"Away in a Manger", musico:"BMVV" }, {prov: "Mundo",
 tit: "Halleluia", musico:"BMVV"}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
>
```

insert document

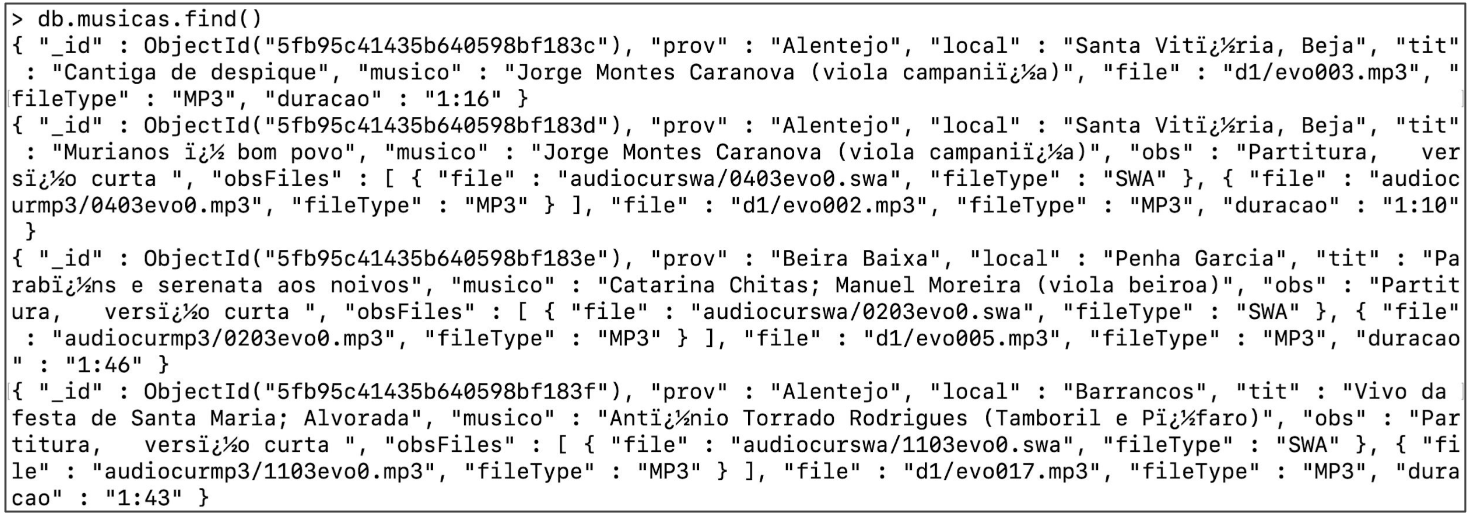
Syntax

```
$ db.collectionName.insert({...})

> ...

$ db.collectionName.insert([{...}, {...}, {...}, ...])

> ...
```

Same As

```
$ db.collectionName.insertOne()

$ db.collectionName.insertMany()
```

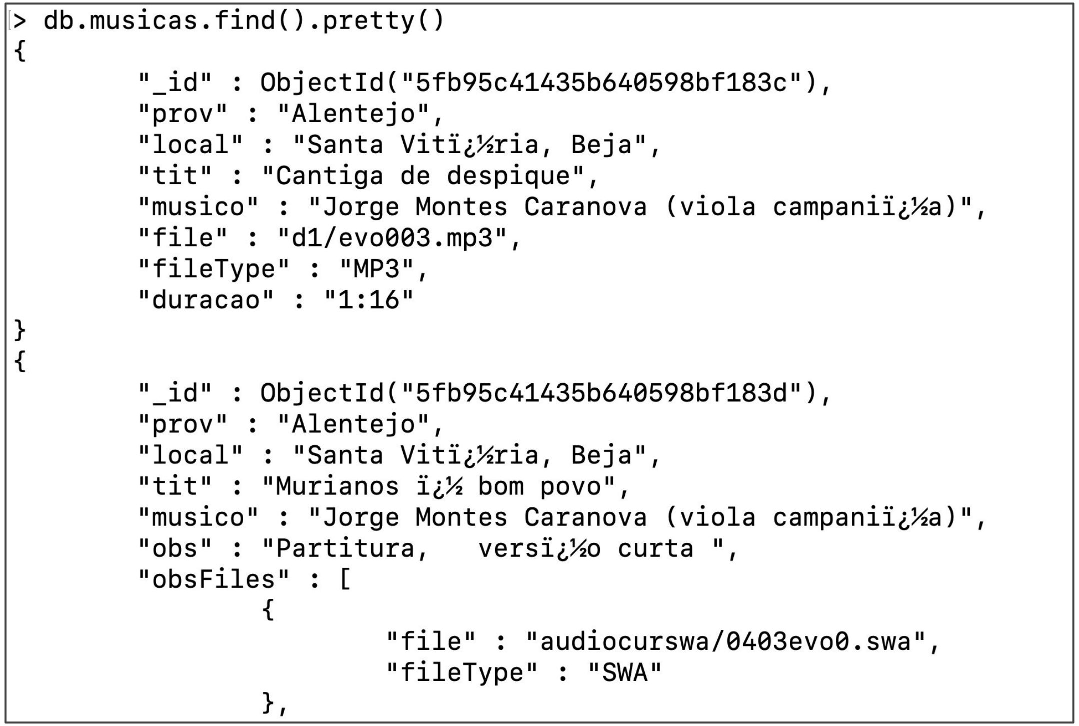
Query data: db.collectionName.find()



```
> db.musicas.find()
{ "_id" : ObjectId("5fb95c41435b640598bf183c"), "prov" : "Alentejo", "local" : "Santa Vitï¿½ria, Beja", "tit"
 : "Cantiga de despique", "musico" : "Jorge Montes Caranova (viola campaniï¿½a)", "file" : "d1/evo003.mp3", "
fileType" : "MP3", "duracao" : "1:16" }
{ "_id" : ObjectId("5fb95c41435b640598bf183d"), "prov" : "Alentejo", "local" : "Santa Vitï¿½ria, Beja", "tit"
 : "Murianos ï¿½ bom povo", "musico" : "Jorge Montes Caranova (viola campaniï¿½a)", "obs" : "Partitura,   ver
sï¿½o curta ", "obsFiles" : [ { "file" : "audiocurswa/0403evo0.swa", "fileType" : "SWA" }, { "file" : "audioc
urmp3/0403evo0.mp3", "fileType" : "MP3" } ], "file" : "d1/evo002.mp3", "fileType" : "MP3", "duracao" : "1:10"
 }
{ "_id" : ObjectId("5fb95c41435b640598bf183e"), "prov" : "Beira Baixa", "local" : "Penha Garcia", "tit" : "Pa
rabï¿½ns e serenata aos noivos", "musico" : "Catarina Chitas; Manuel Moreira (viola beiroa)", "obs" : "Partit
ura,   versï¿½o curta ", "obsFiles" : [ { "file" : "audiocurswa/0203evo0.swa", "fileType" : "SWA" }, { "file"
 : "audiocurmp3/0203evo0.mp3", "fileType" : "MP3" } ], "file" : "d1/evo005.mp3", "fileType" : "MP3", "duracao
" : "1:46" }
{ "_id" : ObjectId("5fb95c41435b640598bf183f"), "prov" : "Alentejo", "local" : "Barrancos", "tit" : "Vivo da
festa de Santa Maria; Alvorada", "musico" : "Antï¿½nio Torrado Rodrigues (Tamboril e Pï¿½faro)", "obs" : "Par
titura,   versï¿½o curta ", "obsFiles" : [ { "file" : "audiocurswa/1103evo0.swa", "fileType" : "SWA" }, { "fi
le" : "audiocurmp3/1103evo0.mp3", "fileType" : "MP3" } ], "file" : "d1/evo017.mp3", "fileType" : "MP3", "dura
cao" : "1:43" }
```

- **`find()`** method displays all documents in a non-structured way.

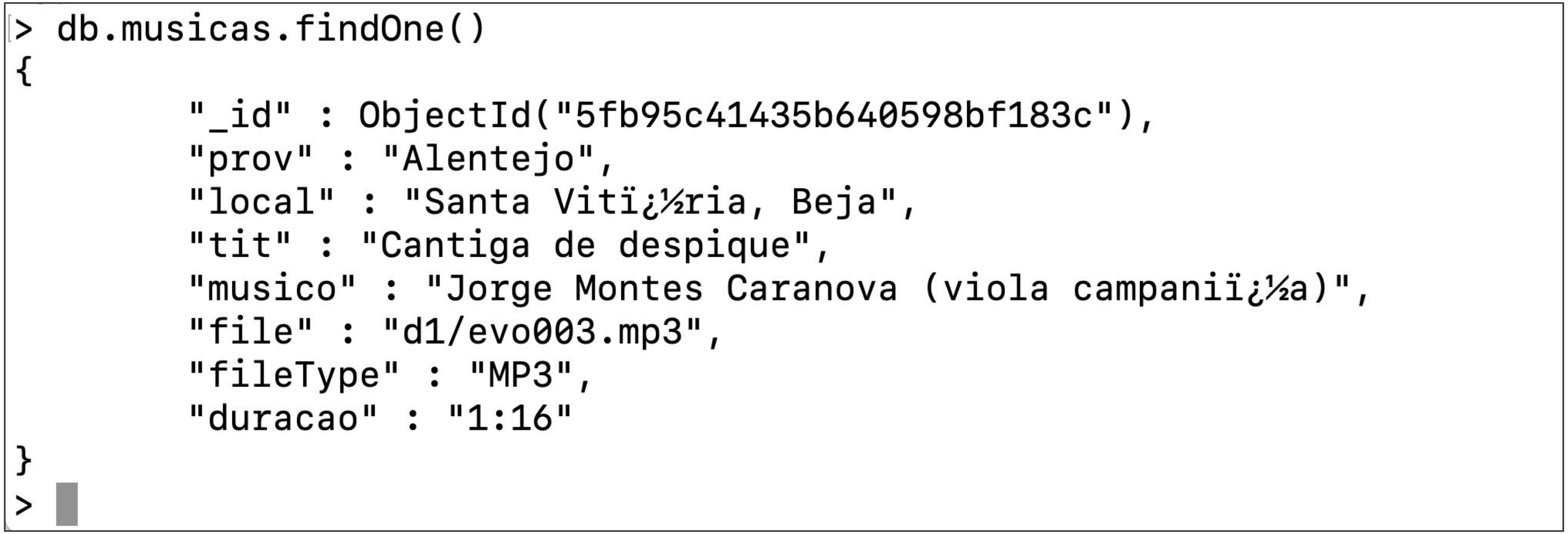
Query data: pretty()



```
> db.musicas.find().pretty()
{
        "_id" : ObjectId("5fb95c41435b640598bf183c"),
        "prov" : "Alentejo",
        "local" : "Santa Vitï¿½ria, Beja",
        "tit" : "Cantiga de despique",
        "musico" : "Jorge Montes Caranova (viola campaniï¿½a)",
        "file" : "d1/evo003.mp3",
        "fileType" : "MP3",
        "duracao" : "1:16"
}
{
        "_id" : ObjectId("5fb95c41435b640598bf183d"),
        "prov" : "Alentejo",
        "local" : "Santa Vitï¿½ria, Beja",
        "tit" : "Murianos ï¿½ bom povo",
        "musico" : "Jorge Montes Caranova (viola campaniï¿½a)",
        "obs" : "Partitura,   versï¿½o curta ",
        "obsFiles" : [
                {
                        "file" : "audiocurswa/0403evo0.swa",
                        "fileType" : "SWA"
                },
```

- **`pretty()`** method displays documents in formatted way.

Query data: findOne()



```
> db.musicas.findOne()
{
        "_id" : ObjectId("5fb95c41435b640598bf183c"),
        "prov" : "Alentejo",
        "local" : "Santa Vitï¿½ria, Beja",
        "tit" : "Cantiga de despique",
        "musico" : "Jorge Montes Caranova (viola campanii¿½a)",
        "file" : "d1/evo003.mp3",
        "fileType" : "MP3",
        "duracao" : "1:16"
}
> 
```

- **`findOne()`** method displays one document in a formatted way.

Query data: find(...)

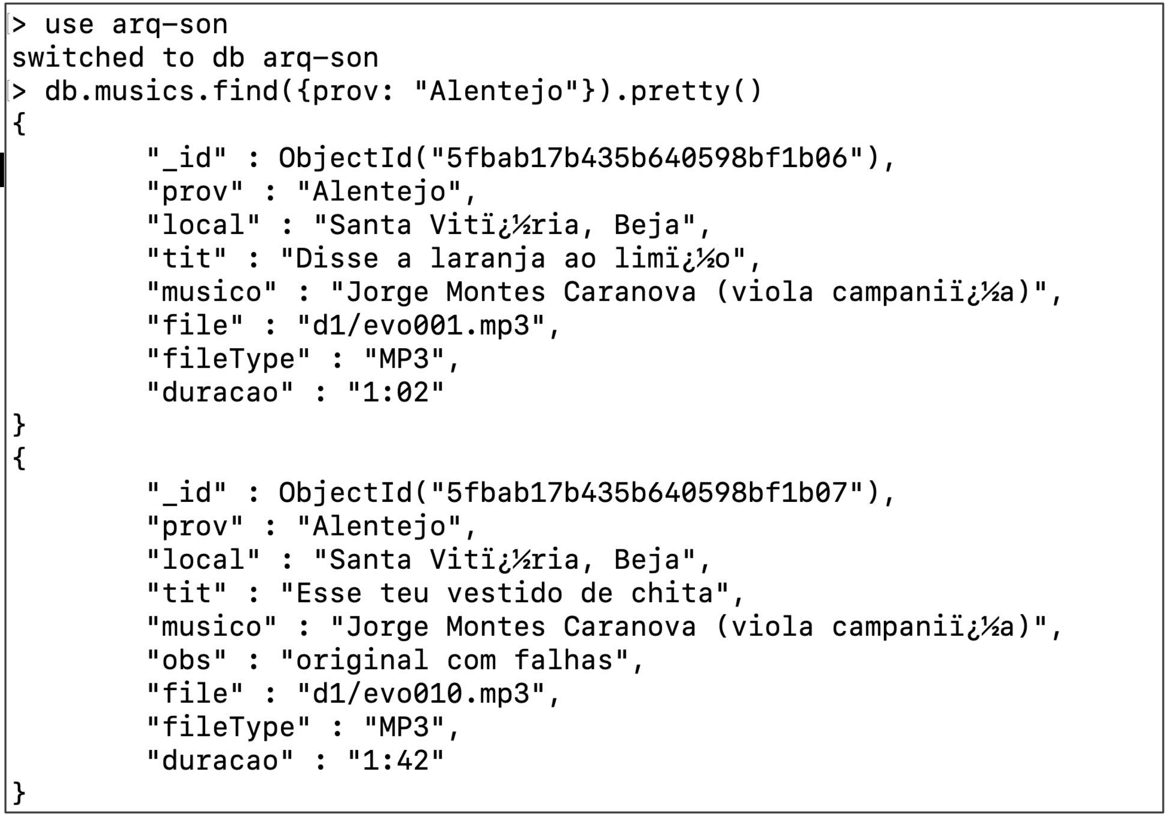
```
find(<select conditions>, <project selections>)
```

- **`<select conditions>`** Conditions that will be used to select/filter documents;
- **`<projet selections>`** Select which fields you want in the result set.

Query data: find(<select conditions>)

Equality

Query: 'I want to retrieve all

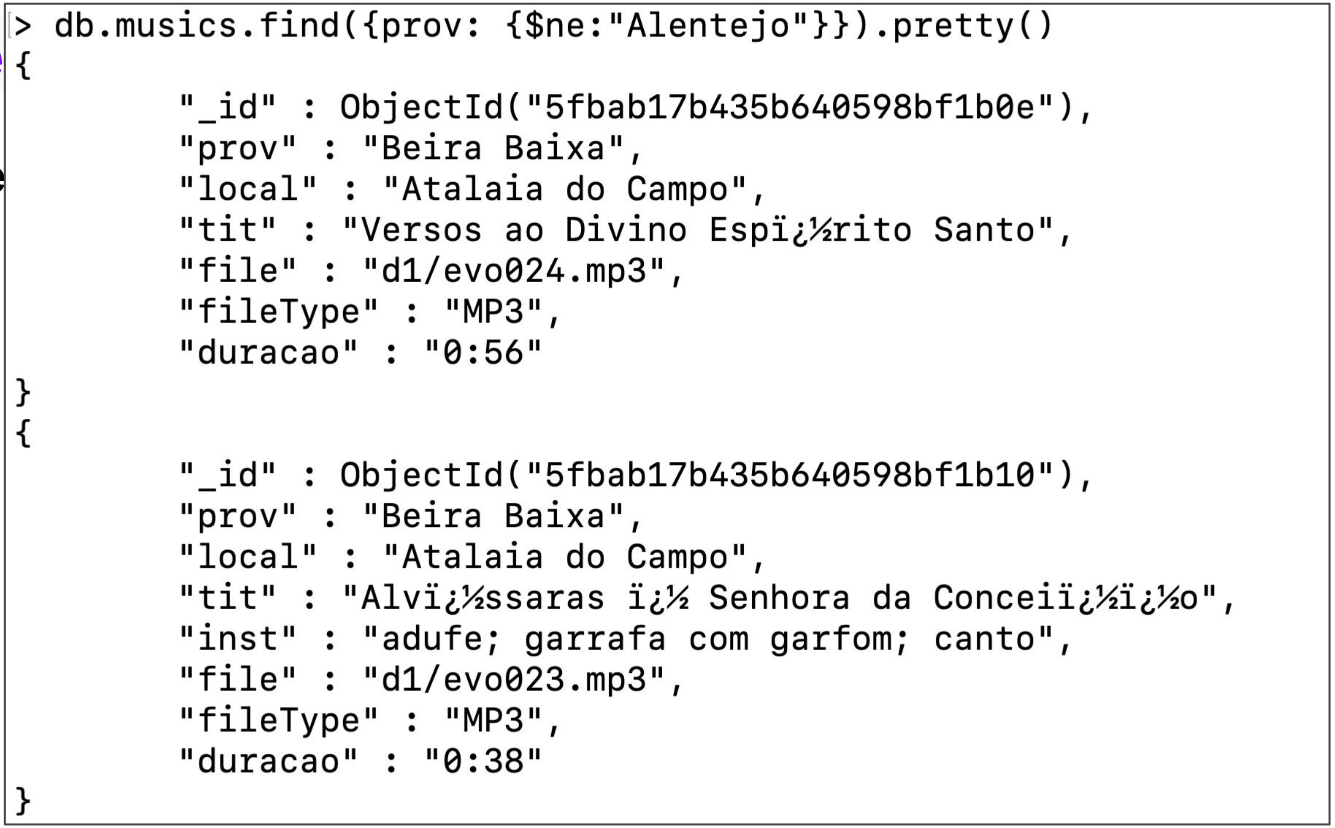


```
> use arq-son
switched to db arq-son
> db.musics.find({prov: "Alentejo"}).pretty()
{
        "_id" : ObjectId("5fbab17b435b640598bf1b06"),
        "prov" : "Alentejo",
        "local" : "Santa Vitï¿½ria, Beja",
        "tit" : "Disse a laranja ao limï¿½o",
        "musico" : "Jorge Montes Caranova (viola campaniï¿½a)",
        "file" : "d1/evo001.mp3",
        "fileType" : "MP3",
        "duracao" : "1:02"
}
{
        "_id" : ObjectId("5fbab17b435b640598bf1b07"),
        "prov" : "Alentejo",
        "local" : "Santa Vitï¿½ria, Beja",
        "tit" : "Esse teu vestido de chita",
        "musico" : "Jorge Montes Caranova (viola campaniï¿½a)",
        "obs" : "original com falhas",
        "file" : "d1/evo010.mp3",
        "fileType" : "MP3",
        "duracao" : "1:42"
}
```

Query data: find(<select conditions>)

Inequality: $ne

Query: 'I want to re
province.'

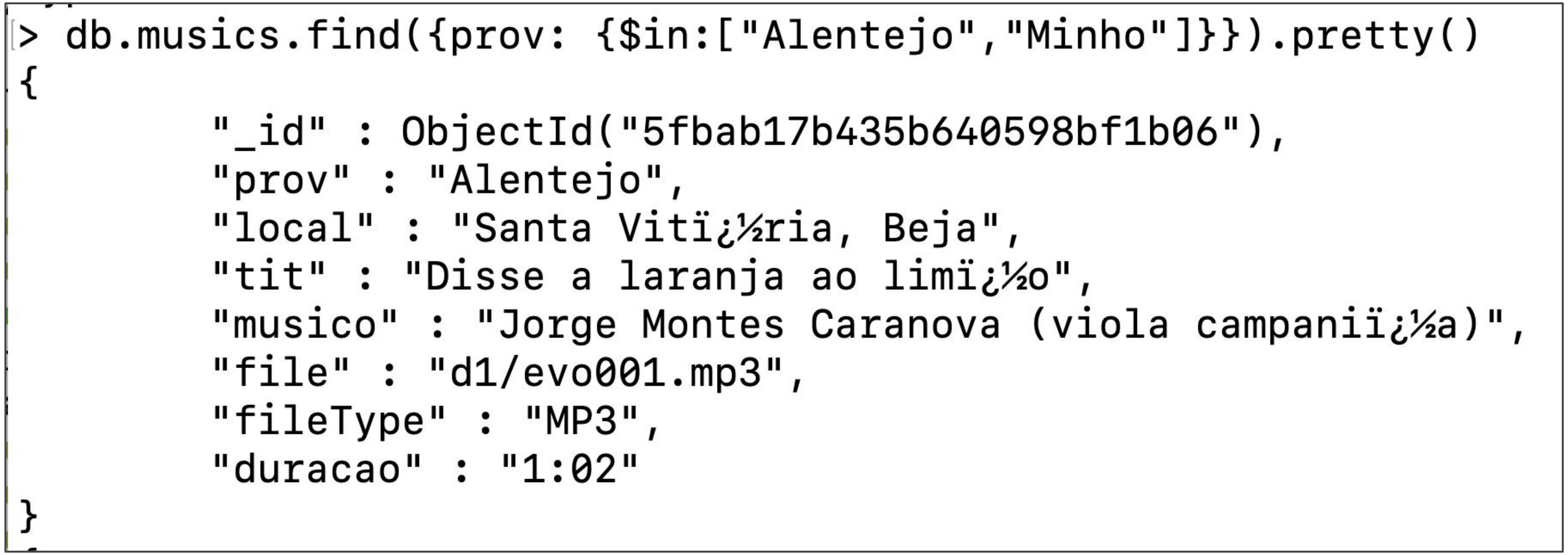


```
> db.musics.find({prov: {$ne:"Alentejo"}}).pretty()
{
        "_id" : ObjectId("5fbab17b435b640598bf1b0e"),
        "prov" : "Beira Baixa",
        "local" : "Atalaia do Campo",
        "tit" : "Versos ao Divino Espï¿½rito Santo",
        "file" : "d1/evo024.mp3",
        "fileType" : "MP3",
        "duracao" : "0:56"
}
{
        "_id" : ObjectId("5fbab17b435b640598bf1b10"),
        "prov" : "Beira Baixa",
        "local" : "Atalaia do Campo",
        "tit" : "Alvï¿½ssaras ï¿½ Senhora da Conceiï¿½ï¿½o",
        "inst" : "adufe; garrafa com garfom; canto",
        "file" : "d1/evo023.mp3",
        "fileType" : "MP3",
        "duracao" : "0:38"
}
```

Query data: find(<select conditions>)

Arrays: $in, $nin

Query: ‘I want to retrieve all documents with music from “Alentejo” and “Minho” provinces.’



```
> db.musics.find({prov: {$in:["Alentejo","Minho"]}}).pretty()
{
        "_id" : ObjectId("5fbab17b435b640598bf1b06"),
        "prov" : "Alentejo",
        "local" : "Santa Vitï¿½ria, Beja",
        "tit" : "Disse a laranja ao limï¿½o",
        "musico" : "Jorge Montes Caranova (viola campaniï¿½a)",
        "file" : "d1/evo001.mp3",
        "fileType" : "MP3",
        "duracao" : "1:02"
}
```

Query data: find(<select conditions>)

And: $and

Query: ‘I want to retrieve all documents with music from “Minho” province and played by “BMVV”.’

```
$ db.musics.find({$and:[
    {prov:"Minho"},
    {musico:"BMVV"}
]}).pretty()
```

Query data: find(<select conditions>)

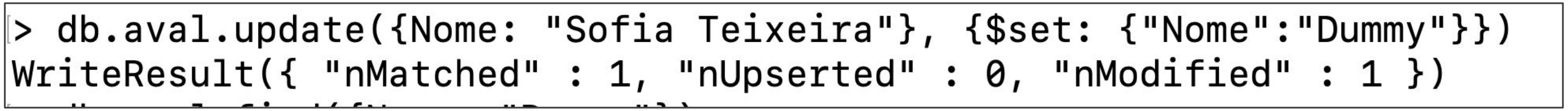
Or: $or

Query: ‘I want to retrieve all documents with music from “Minho” province or “Estremadura” province.’

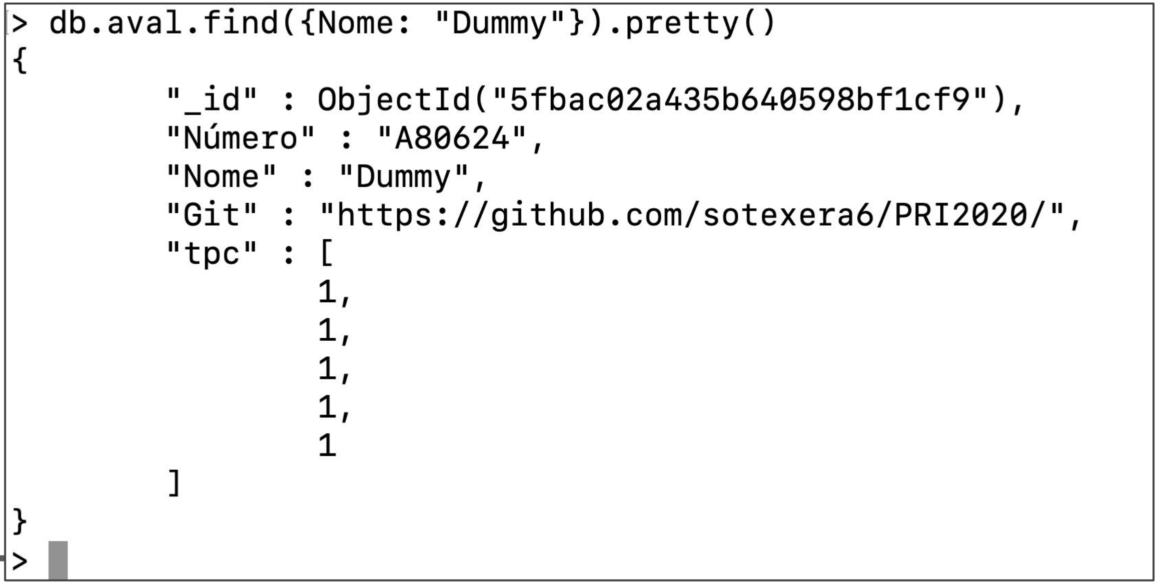
```
$ db.musics.find({$or:[
    {prov:"Minho"},
    {prov:"Estremadura"}
]}).pretty()
```

Update()

```
db.collectionName.update(<selection>, <updated data>)
```



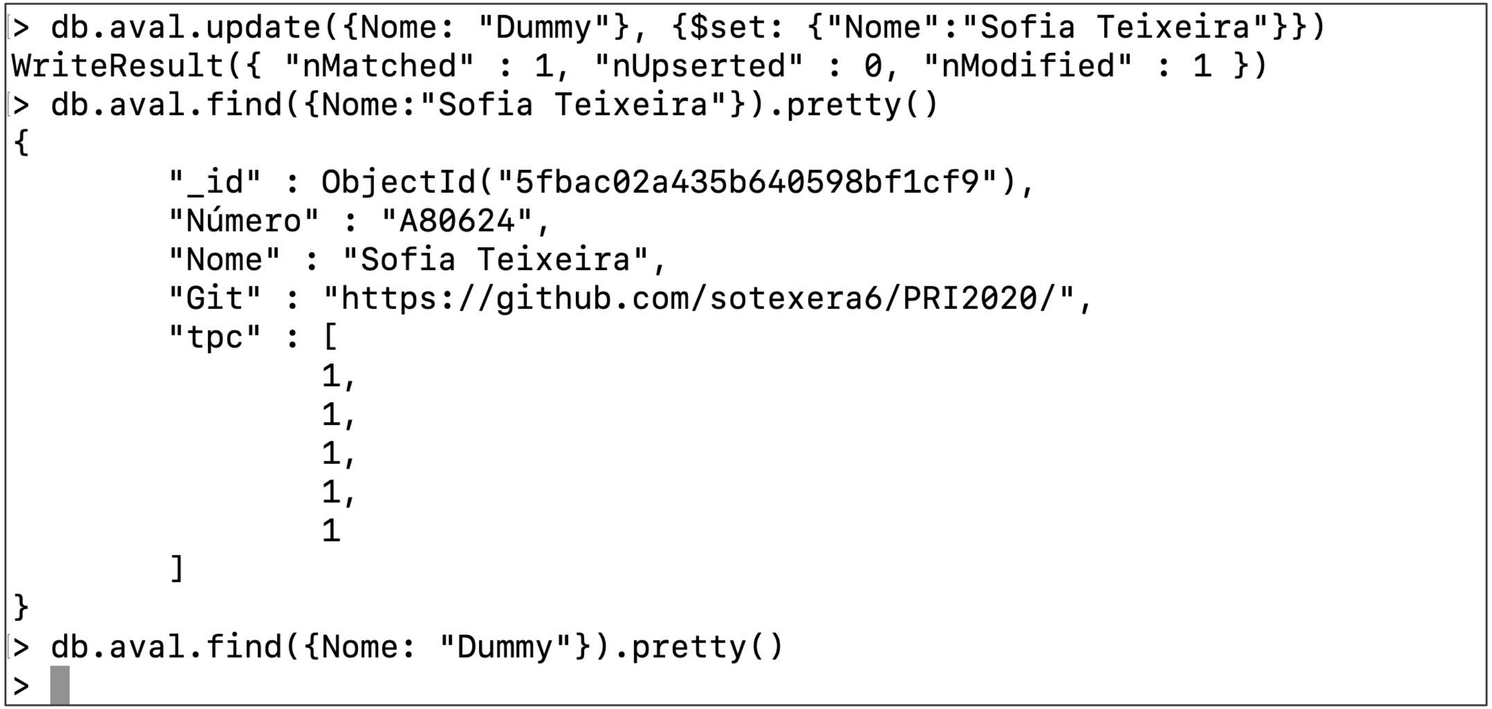
```
> db.aval.update({Nome: "Sofia Teixeira"}, {$set: {"Nome":"Dummy"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```



```
> db.aval.find({Nome: "Dummy"}).pretty()
{
        "_id" : ObjectId("5fbac02a435b640598bf1cf9"),
        "Número" : "A80624",
        "Nome" : "Dummy",
        "Git" : "https://github.com/sotexera6/PRI2020/",
        "tpc" : [
                1,
                1,
                1,
                1,
                1
        ]
}
>
```

Update(): reverse

```
db.collectionName.update(<selection>, <updated data>)
```

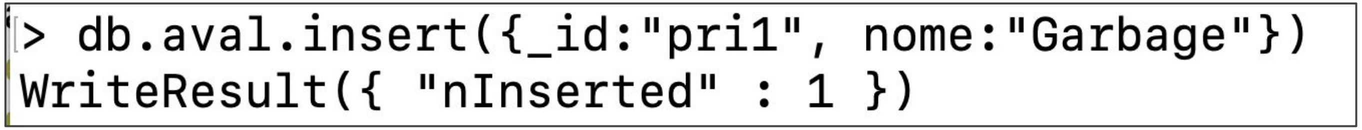


```
> db.aval.update({Nome: "Dummy"}, {$set: {"Nome":"Sofia Teixeira"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.aval.find({Nome:"Sofia Teixeira"}).pretty()
{
        "_id" : ObjectId("5fbac02a435b640598bf1cf9"),
        "Número" : "A80624",
        "Nome" : "Sofia Teixeira",
        "Git" : "https://github.com/sotexera6/PRI2020/",
        "tpc" : [
                1,
                1,
                1,
                1,
                1
        ]
}
> db.aval.find({Nome: "Dummy"}).pretty()
>
```

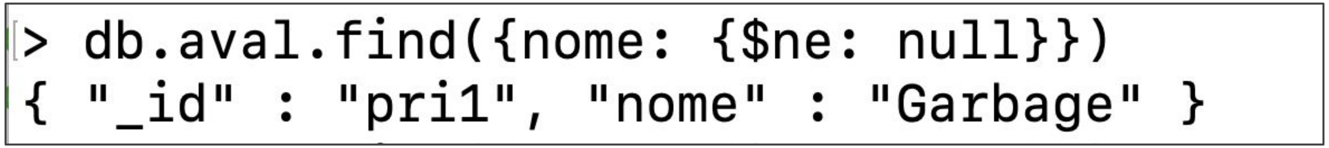
Update(): related methods

```
db.collectionName.save({_id: ObjectId()}, <newData>)

db.collectionName.findOneAndUpdate(<selection>, <updatedData>)

db.collectionName.updateOne(<selection>, <updatedData>)

db.collectionName.updateMany(<selection>, <updatedData>)
```

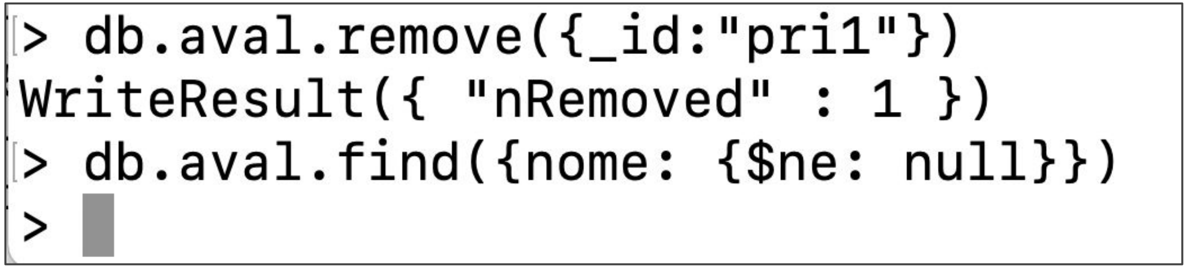
remove()



```
> db.aval.insert({_id:"pri1", nome:"Garbage"})
WriteResult({ "nInserted" : 1 })
```



```
> db.aval.find({nome: {$ne: null}})
{ "_id" : "pri1", "nome" : "Garbage" }
```



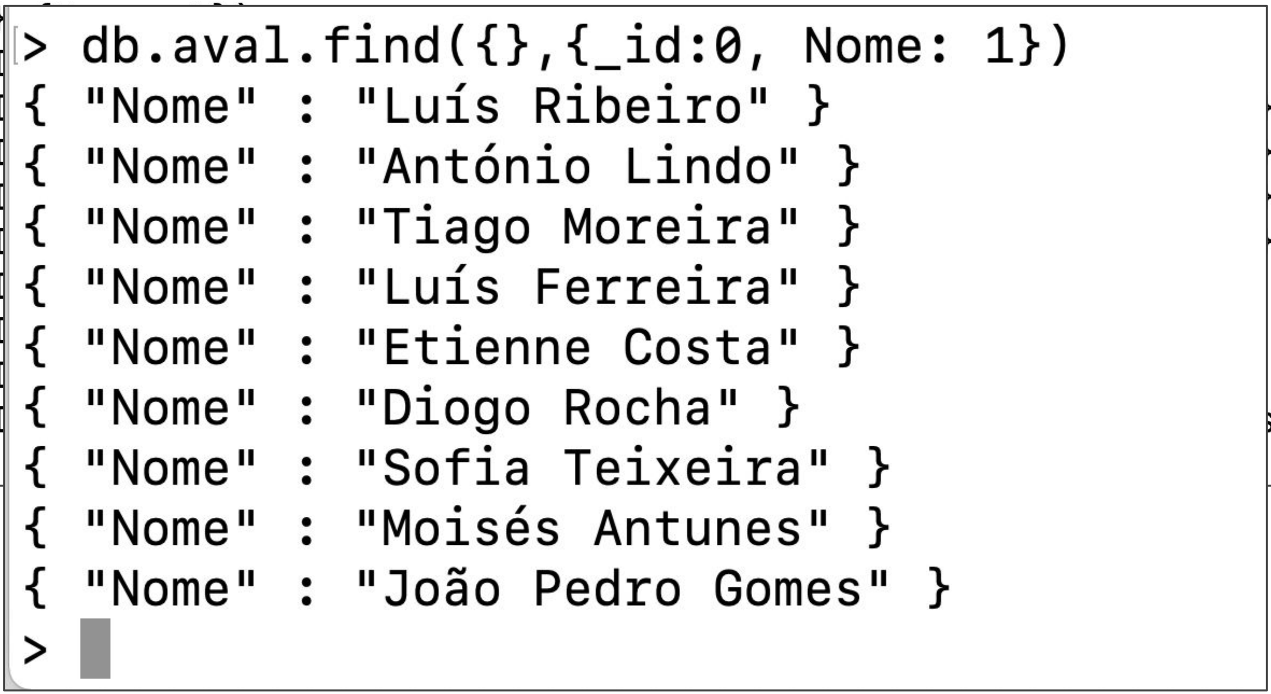
```
> db.aval.remove({_id:"pri1"})
WriteResult({ "nRemoved" : 1 })
> db.aval.find({nome: {$ne: null}})
>
```

- Removing without criteria will remove all documents;
- To remove just one add parameter: **`{justOne: 1}`**

Projection

```
$ db.collectionName.find({}, {KEY: 1})
```

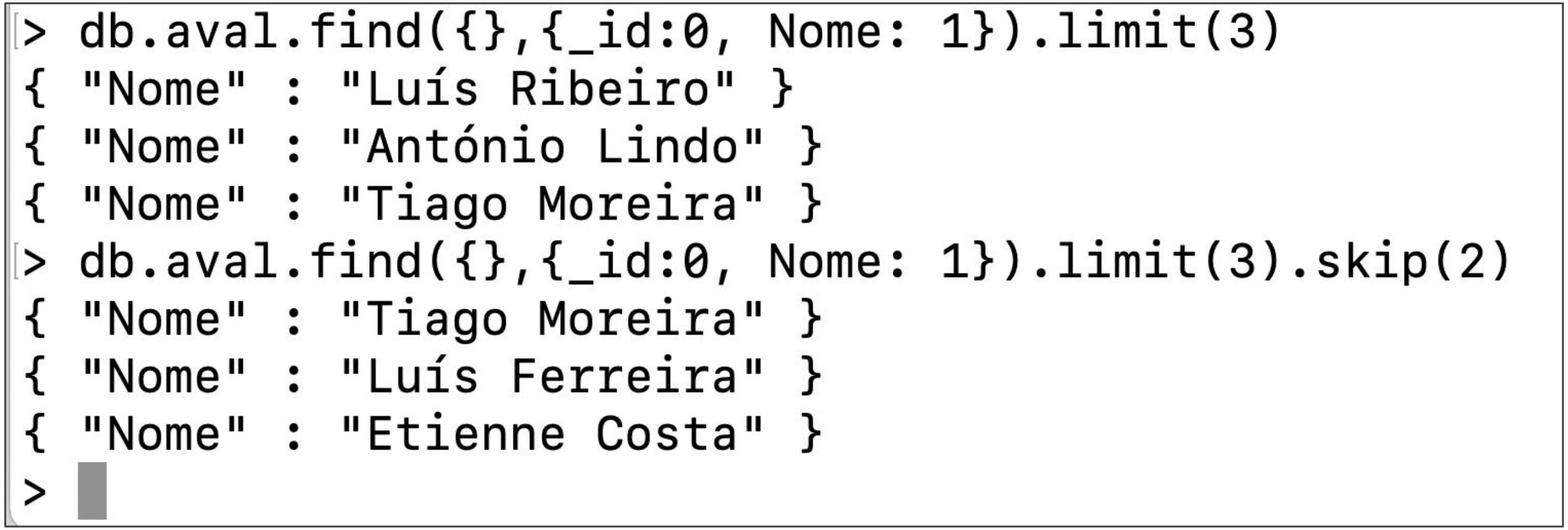
```
> db.aval.find({}
{ "_id" : ObjectI
{ "_id" : ObjectI
{ "_id" : ObjectI
{ "_id" : ObjectI
{ "_id" : ObjectI
{ "_id" : ObjectI
{ "_id" : ObjectI                                                   }
{ "_id" : ObjectI                                                   }
{ "_id" : ObjectI                                                  s" }
> 
```



```
> db.aval.find({},{_id:0, Nome: 1})
{ "Nome" : "Luís Ribeiro" }
{ "Nome" : "António Lindo" }
{ "Nome" : "Tiago Moreira" }
{ "Nome" : "Luís Ferreira" }
{ "Nome" : "Etienne Costa" }
{ "Nome" : "Diogo Rocha" }
{ "Nome" : "Sofia Teixeira" }
{ "Nome" : "Moisés Antunes" }
{ "Nome" : "João Pedro Gomes" }
> 
```

Limit and Skip

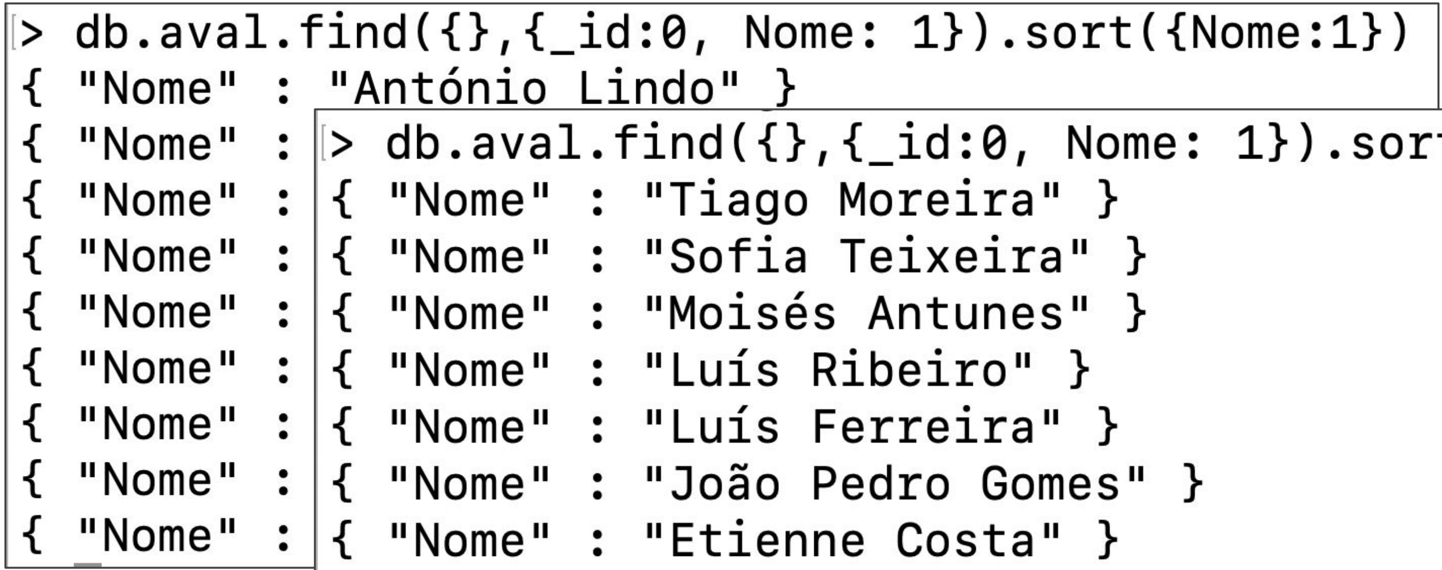
```
$ db.collectionName.find().limit(Number)
$ db.collectionName.find().limit(Number).skip(Number)
```



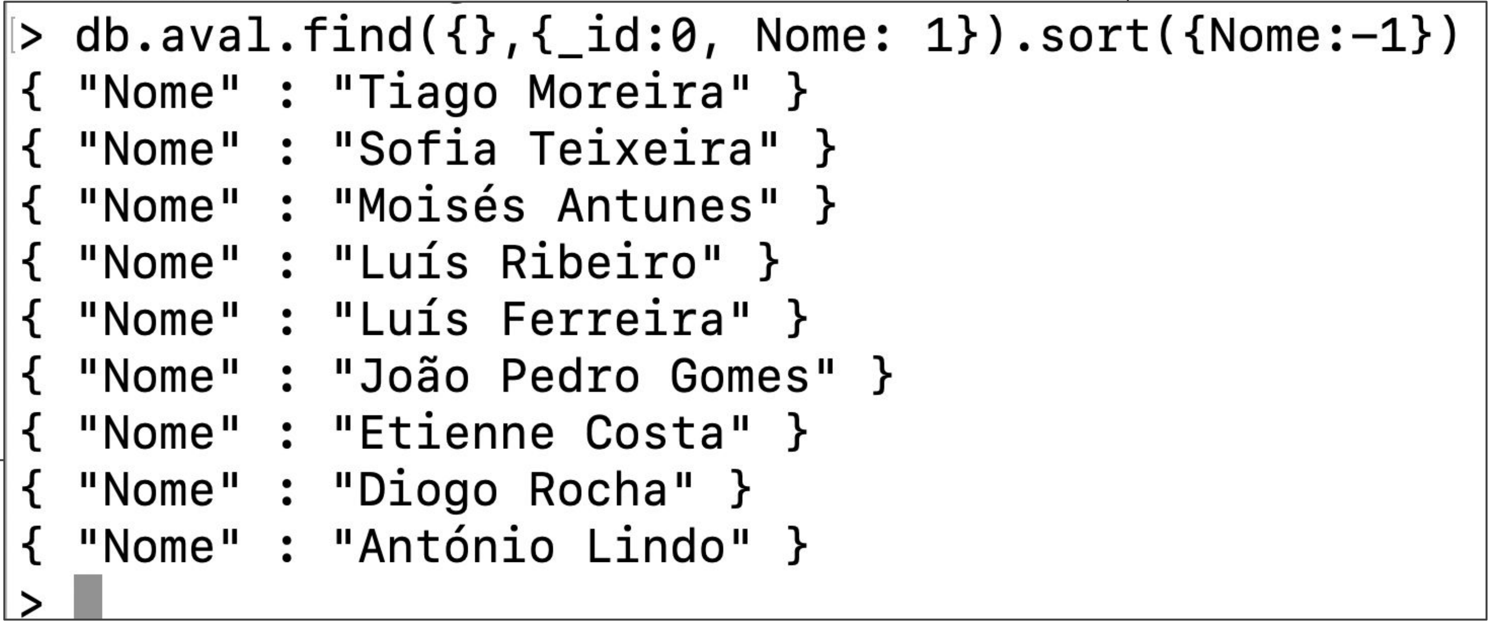
```
> db.aval.find({},{_id:0, Nome: 1}).limit(3)
{ "Nome" : "Luís Ribeiro" }
{ "Nome" : "António Lindo" }
{ "Nome" : "Tiago Moreira" }
> db.aval.find({},{_id:0, Nome: 1}).limit(3).skip(2)
{ "Nome" : "Tiago Moreira" }
{ "Nome" : "Luís Ferreira" }
{ "Nome" : "Etienne Costa" }
>
```

Sorting records

```
$ db.collectionName.find().sort({KEY: 1})
```



```
> db.aval.find({},{_id:0, Nome: 1}).sort({Nome:1})
{ "Nome" : "António Lindo" }
{ "Nome" :
{ "Nome" :
{ "Nome" :
{ "Nome" :
{ "Nome" :
{ "Nome" :
{ "Nome" :
{ "Nome" :
```



```
> db.aval.find({},{_id:0, Nome: 1}).sort({Nome:-1})
{ "Nome" : "Tiago Moreira" }
{ "Nome" : "Sofia Teixeira" }
{ "Nome" : "Moisés Antunes" }
{ "Nome" : "Luís Ribeiro" }
{ "Nome" : "Luís Ferreira" }
{ "Nome" : "João Pedro Gomes" }
{ "Nome" : "Etienne Costa" }
{ "Nome" : "Diogo Rocha" }
{ "Nome" : "António Lindo" }
>
```

Aggregation: operation pipeline

To be continued...

Connecting MongoDB with nodejs: mongoose

$

```
//Import the mongoose module
var mongoose = require('mongoose');
//Set up default mongoose connection
var mongoDB = 'mongodb://127.0.0.1/PRI2020' ;
mongoose.connect(mongoDB, {useNewUrlParser:  true, useUnifiedTopology: 
true});
//Get the default connection
var db = mongoose.connection;
//Bind connection to error event (to get notification of connection 
errors)
db.on('error', console.error.bind(console, 'MongoDB connection 
error...'));
db.once('open', function() {
```

mongoose: inserting data

```
var studentSchema = new mongoose.Schema({
   Número: String,
   Nome: String,
   Git: String,
   tpc: [Number]
});
var studentModel = mongoose.model('student', studentSchema)
var data = [
   {
     "Número": "A76089",
     "Nome": "Etienne Costa",
     "Git": "https://github.com/Eti
     "tpc": [1,1,1,1,1]
   }, ... ]
```

```
studentModel.create(data)

console.log("That's all folks...")
```

mongoose: retrieving data

```
var avalSchema = new mongoose.Schema({
   Número: String, Nome: String, Git: String, tpc: [Number]
});
var AVAL = mongoose.model('students', avalSchema)
// Retrieve all students
AVAL
   .find(function(err, docs) {
   if(err){
       console.log('Error retrieving student records: ' + err)
   }
   else{
       console.log(docs)
   }})
```

Putting all together: express

```
$ npm i express -g

$ express --view=pug studentsApp

$ cd studentsApp

$ npm i

$ npm start

...
```