

MongoDB

NoSQL: Banco de dados não relacionais

Mongo vem da expressão em inglês "Humongous".

Significados: Gigantesco, Monstruoso.

11-03-2018

Índice

NoSQL:	4
O que é um documento?	4
Download MongoDB:	5
Instalação:	5
Terminal de comandos	7
Comandos básicos:	7
Criando uma base de dados	8
Criando uma coleção:	8
Visualizando os dados de uma coleção:	9
Visualizar todas as bases de dados:	9
Selecionar uma base de dados:	10
Mostrar todas as coleções	10
Visualizar o stats de uma base de dados:	11
Visualizando stats de uma coleção:	11
DATE	12
PRETTY	13
OBJETOS E VETORES	13
CRUD (Create, Read, Update, Delete)	16
CREATE	17
UPDATE	20
DELETE	25
READ	27
Operadores Lógicos:	29
\$and:	29
\$or:	30
\$exists:	31
\$not:	32
\$nor:	33

Operadores de Comparação:	34
\$eq:	34
\$ne:	35
Operadores de Maior e Menor que.	36
\$gt:	36
\$lt:	36
\$lte:	37
\$gte:	38
Projections.	38
Visualizar atributos específicos:	38
Como omitir colunas:	39
Buscas Ordenadas:	40
LIMIT.	42
SKIP.	43
Busca por palavras específicas:	45
\$regex:	45
COUNT.	46
Associações.	46
Fim.	51

NoSQL:

É um banco de dados não relacional.

Trabalha com coleções de documentos.

NoSQL são diferentes sistemas de armazenamento que vieram para suprir necessidades em demandas onde os bancos de dados tradicionais (relacionais) são ineficazes. Muitas dessas bases apresentam características muito interessantes como altas performances, escalabilidade, replicação, suporte a dados estruturados e sub colunas.

No caso dos bancos NoSQL, toda a informação necessária estará agrupada no mesmo registro, ou seja, em vez de você ter o relacionamento entre várias tabelas para formar uma informação, ela estará em sua totalidade no mesmo registro.

Artigo completo em:

<https://imasters.com.br/artigo/17043/banco-de-dados/nosql-voce-realmente-sabe-do-que-estamos-falando/>

O que é um documento?

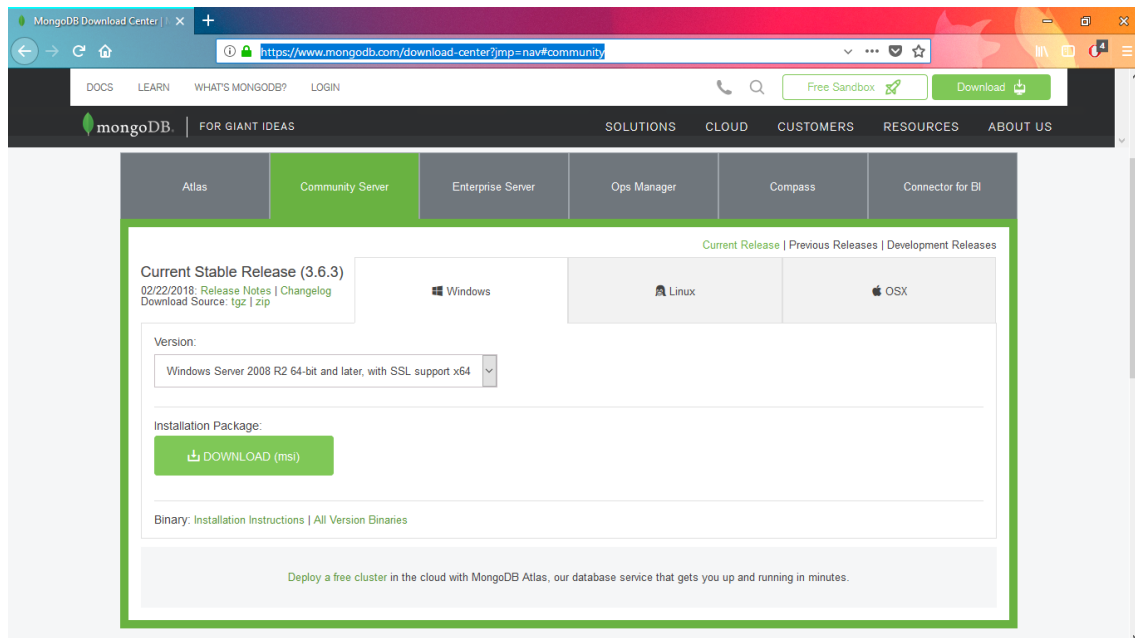
Um documento é um conjunto de chaves e valores. Documentos tem esquema dinâmico. Esquema dinâmico significa que os documentos na mesma coleção não precisam ter o mesmo conjunto de campos ou estrutura de campos comuns em documentos de uma coleção, podendo conter diferentes tipos de dados.

Artigo completo:

<https://mongodbwise.wordpress.com/2014/05/22/mongodb-guia-rapido/>

Download MongoDB:

<https://www.mongodb.com/download-center?jmp=nav#community>



Instalação:

Após a instalação é necessário criar os seguintes diretórios:

Acesse o Disco Local C.

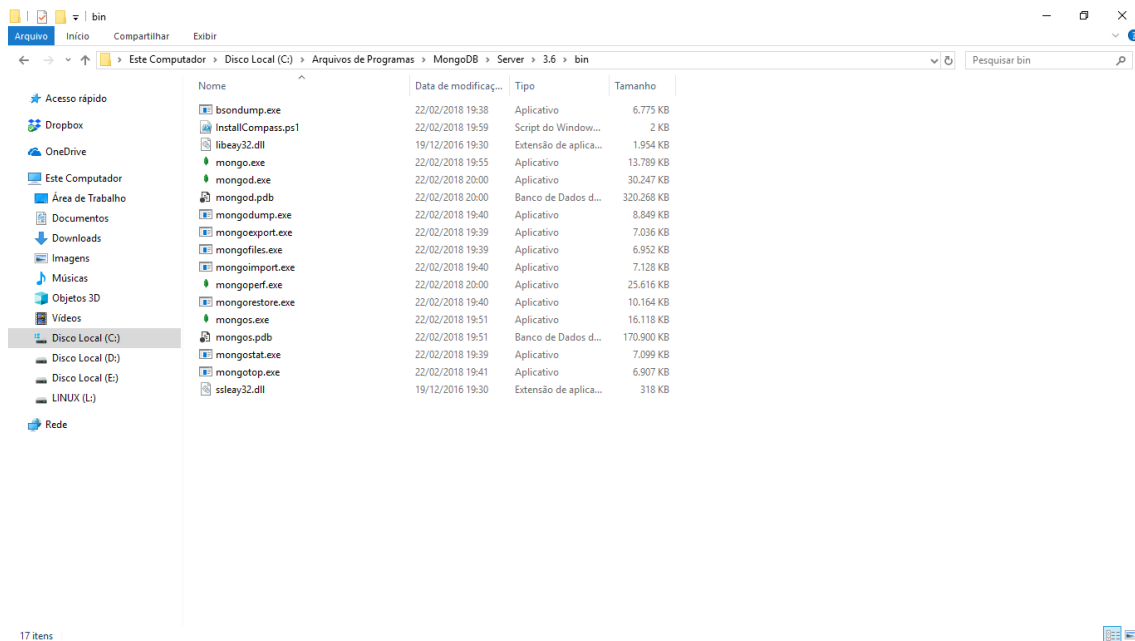
Crie uma pasta chamada "data".

Dentro da pasta da "data" crie outra chamada "db".

Feito isso...

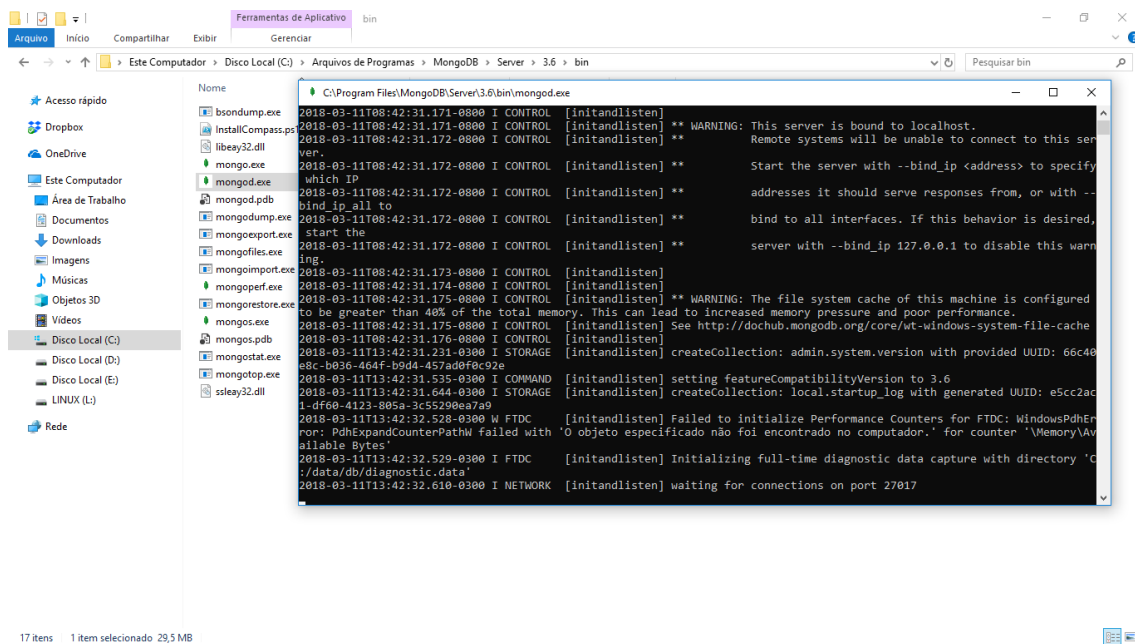
Agora será necessário ligar o servidor do mongodb.

Acesse a pasta de instalação.



Clique em “mongod.exe”.

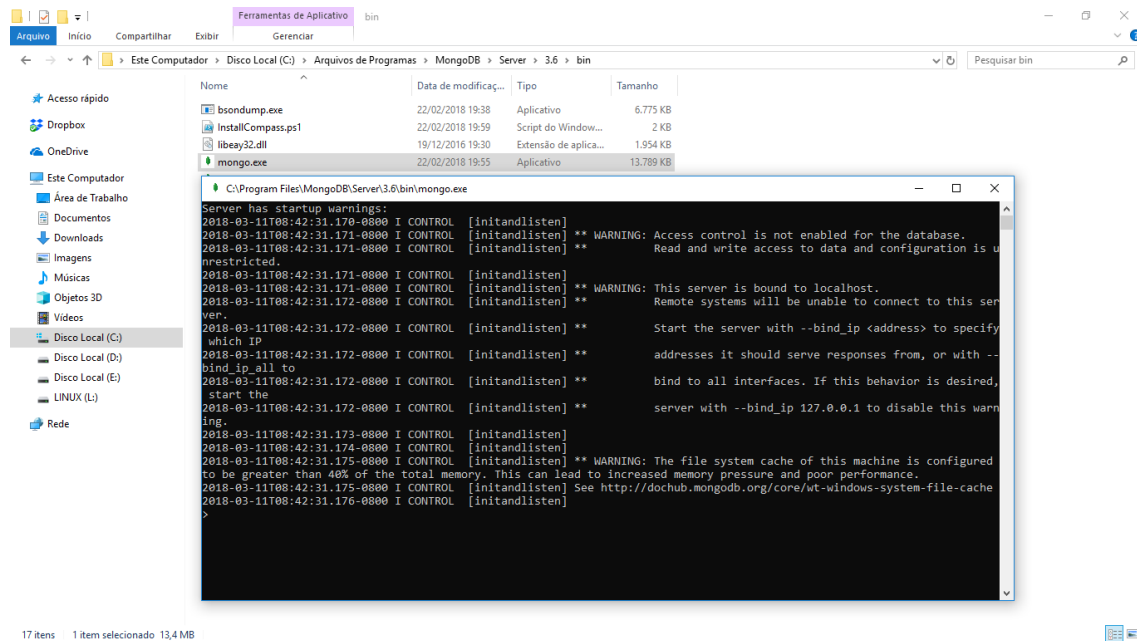
Servidor Start...



Caso este terminal seja fechado os comandos não poderão ser executados, pois o servidor é desligado.

Minimize a janela para abrir o novo terminal, onde será executado os comandos.

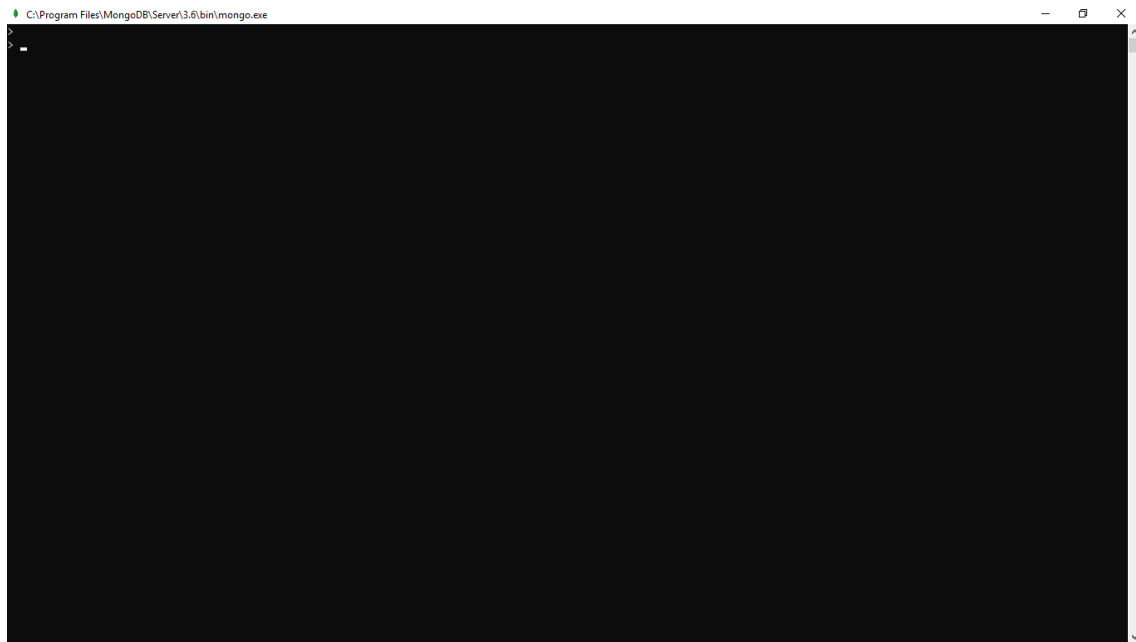
Clique em “mongo.exe”



Terminal de comandos.

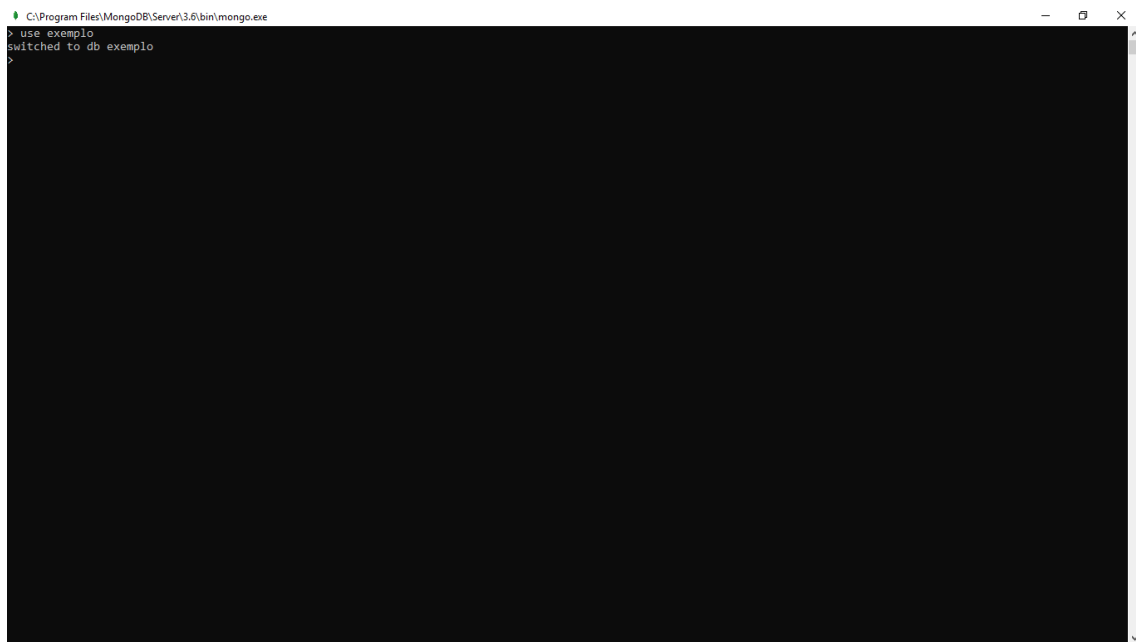
Comandos básicos:

➤ **cls** : limpa a tela.



Criando uma base de dados.

➤ **use exemplo**



Criando uma coleção:

➤ **db.usuarios.insert({cod: 1, nome: "Jane Doe"});**


```
Prompt de Comando - mongo
> db.usuarios.insert({cod: 1, nome: "Jane Doe"});
WriteResult({ "nInserted" : 1 })
>
```

Visualizando os dados de uma coleção:

➤ **db.usuarios.find();**

```
Prompt de Comando - mongo
> db.usuarios.insert({cod: 1, nome: "Jane Doe"});
WriteResult({ "nInserted" : 1 })
> db.usuarios.find();
{ "_id" : ObjectId("5aa5900fac32a6b6c6dd2d88"), "cod" : 1, "nome" : "Jane Doe" }
>
```

Visualizar todas as bases de dados:

➤ **show dbs;**

```
C:\Program Files\MongoDB\Server\3.6\bin\mongo.exe
> show dbs;
admin      0.000GB
config     0.000GB
exemplo    0.000GB
local      0.000GB
>
```

Selecionar uma base de dados:

➤ **use exemplo;**

Caso a base não exista ela será criada com este mesmo comando.

```
C:\Program Files\MongoDB\Server\3.6\bin\mongo.exe
> show dbs;
admin      0.000GB
config     0.000GB
exemplo    0.000GB
local      0.000GB
> use exemplo;
switched to db exemplo
>
```

Mostrar todas as coleções.

➤ **show collections;**

```
C:\Program Files\MongoDB\Server\3.6\bin\mongo.exe
> show dbs;
admin    0.000GB
config  0.000GB
exemplo  0.000GB
local   0.000GB
> use exemplo;
switched to db exemplo
> show collections;
usuarios
>
```

Visualizar o stats de uma base de dados:

➤ **db.stats();**

```
C:\Program Files\MongoDB\Server\3.6\bin\mongo.exe
> show dbs;
admin    0.000GB
config  0.000GB
exemplo  0.000GB
local   0.000GB
> use exemplo;
switched to db exemplo
> show collections;
usuarios
> db.stats();
{
  "db" : "exemplo",
  "collections" : 1,
  "views" : 0,
  "objects" : 1,
  "avgObjSize" : 54,
  "dataSize" : 54,
  "storageSize" : 16384,
  "numExtents" : 0,
  "indexes" : 1,
  "indexSize" : 16384,
  "fsUsedSize" : 137077035008,
  "fsTotalSize" : 209138479104,
  "ok" : 1
}
```

Visualizando stats de uma coleção:

➤ **db.usuarios.stats();**

```

Prompt de Comando - mongo
> db.usuarios.stats();
{
  "ns" : "exemplo.usuarios",
  "size" : 238,
  "count" : 4,
  "avgObjSize" : 59,
  "storageSize" : 36864,
  "capped" : false,
  "wiredTiger" : {
    "metadata" : {
      "formatVersion" : 1
    },
    "creationString" : "access_pattern_hint=none,allocation size=4KB,app_metadata=(formatVersion=1),assert=(commit_timesta
mp=none,read_timestamp=none),block_allocation=best,block_compressor=snappy,cache_resident=false,checksum=on,colgroups=,collator=,column
ns=,dictionary=0,encryption=(keyid=,name=),exclusive=false,extractor=,format=btree,huffman_key=,huffman_value=,ignore_in_memory_cache_
size=false,immutable=false,internal_item_max=0,internal_key_max=0,internal_key_truncate=true,internal_page_max=4KB,key_format=q,key_ga
p=10,leaf_item_max=0,leaf_key_max=0,leaf_page_max=32KB,leaf_value_max=64MB,log=(enabled=true),lsm=(auto_throttle=true,bloom=true,bloom
_bit_count=16,bloom_config=,bloom_hash_count=8,bloom_oldest=false,chunk_count_limit=0,chunk_max=5GB,chunk_size=10MB,merge_custom=(pref
ix=,start_generation=0,suffix=),merge_max=15,merge_min=0),memory_page_max=10m,os_cache_dirty_max=0,os_cache_max=0,prefix_compression=f
alse,prefix_compression_min=4,source=,split_deepen_min_child=0,split_deepen_per_child=0,split_pct=90,type=file,value_format=u",
    "type" : "file",
    "uri" : "statistics:table:collection-2-9130376997547728080",
    "LSM" : {
      "bloom filter false positives" : 0,
      "bloom filter hits" : 0,
      "bloom filter misses" : 0,
      "bloom filter pages evicted from cache" : 0,
      "bloom filter pages read into cache" : 0,
      "bloom filters in the LSM tree" : 0,
      "chunks in the LSM tree" : 0,
      "highest merge generation in the LSM tree" : 0,
      "queries that could have benefited from a Bloom filter that did not exist" : 0,
      "sleep for LSM checkpoint throttle" : 0,
      "sleep for LSM merge throttle" : 0,
      "total size of bloom filters" : 0
    }
  },
  "block-manager" : {

```

Inserindo dados com novas colunas em uma coleção existente:

- **db.usuarios.insert({ cod: 2, nome: "Jhon Doe", email: "jhon@gmail.com"});**
- **db.usuarios.insert({ cod: 3, nome: "Nancy", nascimento: new Date(), email: "[nancy@gmail.com](\"mailto:nancy@gmail.com\")", sexo: "F"});**

DATE.

O parametro "*new Date()*" retorna a data e a hora do Sistema Operacional.

```
Prompt de Comando - mongo
> db.usuarios.insert({cod: 2, nome: "Jhon Doe", email: "jhon@gmail.com"});
WriteResult({ "nInserted" : 1 })
> db.usuarios.insert({cod: 3, nome: "Nancy", nascimento: new Date(), email: "nancy@gmail.com", sexo: "F"});
WriteResult({ "nInserted" : 1 })
>
```

PRETTY.

Visualizando dados no formato estruturado:

➤ **db.usuarios.find().pretty();**

```
Prompt de Comando - mongo
> db.usuarios.find().pretty();
{
  "_id" : ObjectId("5aa58cb7ac32a6b6c6dd2d84"),
  "cod" : 1,
  "nome" : "Jane Doe"
}
{
  "_id" : ObjectId("5aa58cecac32a6b6c6dd2d85"),
  "cod" : 2,
  "nome" : "Jhon Doe",
  "email" : "jhon@gmail.com"
}
{
  "_id" : ObjectId("5aa58d42ac32a6b6c6dd2d86"),
  "cod" : 3,
  "nome" : "Nancy",
  "nascimento" : ISODate("2018-03-11T20:10:42.983Z"),
  "email" : "nancy@gmail.com",
  "sexo" : "F"
}
```

OBJETOS E VETORES.

Inserindo informações em atributos com vetores:

- **db.usuarios.insert({
cod: 4,
nome: "McDonalds",
email: "mcdonalds@gmail.com",
comidas: ["Hambúrguer", "Coca-Cola", "Milk-Shake", "Batata Frita"]
});**



```
Prompt de Comando - mongo
> db.usuarios.insert({
... cod: 4,
... nome: "McDonalds",
... email: "mcdonalds@gmail.com",
... comidas: ["Hambúrguer", "Coca-Cola", "Milk-Shake", "Batata Frita"]
... });
WriteResult({ "nInserted" : 1 })
>
```

Visualizando os dados de uma chave específica:

- **db.usuarios.find({cod: 4}).pretty();**

```
Prompt de Comando - mongo
> db.usuarios.find({cod: 4}).pretty();
{
  "_id" : ObjectId("5aa597ceac32a6b6c6dd2d8d"),
  "cod" : 4,
  "nome" : "McDonalds",
  "email" : "mcdonalds@gmail.com",
  "comidas" : [
    "Hamburguer",
    "Coca-Cola",
    "Milk-Shake",
    "Batata Frita"
  ]
}
```

Inserido informações em atributos com objetos:

```
> db.usuarios.insert({
  cod: 5,
  nome: "Amy Adams",
  profissao: "Atriz",
  email: "adams@email.com",
  filmes: {
    lancamento: 2016,
    titulo: "A Chegada",
    personagem: "Dr. Louise Banks",
    generos: ["ação", "misterio", "suspense"]
  }
});
```

```
Prompt de Comando - mongo
> db.usuarios.insert({
... cod: 5,
... nome: "Amy Adams",
... profissao: "Atriz",
... email: "adams@email.com",
... filmes: {
... lancamento: 2016,
... titulo: "A Chegada",
... personagem: "Dr. Louise Banks",
... generos: ["ação", "misterio", "suspense"]
... }
... });
WriteResult({"nInserted" : 1 })
>
```

Visualizando as informações:

➤ **db.usuarios.find({cod: 5}).pretty();**

```
Prompt de Comando - mongo
> db.usuarios.find({cod: 5}).pretty();
{
  "_id" : ObjectId("5aa59c5aac32a6b6c6dd2d8e"),
  "cod" : 5,
  "nome" : "Amy Adams",
  "profissao" : "Atriz",
  "email" : "adams@email.com",
  "filmes" : {
    "lancamento" : 2016,
    "titulo" : "A Chegada",
    "personagem" : "Dr. Louise Banks",
    "generos" : [
      "ação",
      "misterio",
      "suspense"
    ]
  }
}
>
```

CRUD (Create, Read, Update, Delete)

CREATE.

Tipos de insert:

1. **db.collection.insert();**

Inserir 1 ou mais documentos;

2. **db.collection.insertOne();**

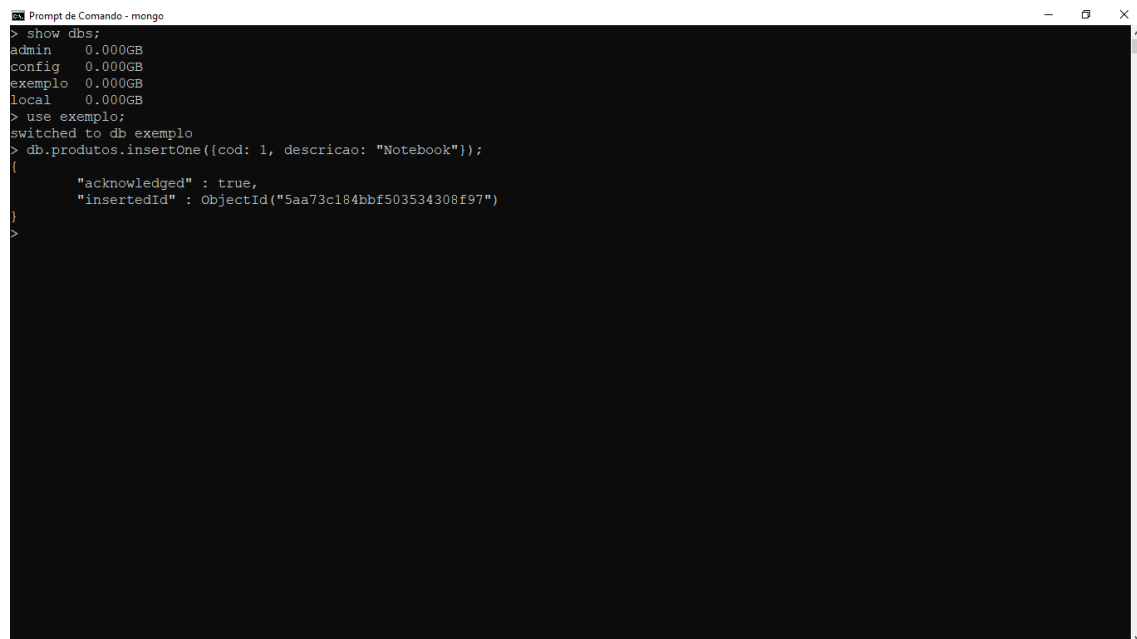
Inserir apenas 1 documento;

3. **db.collection.insertMany();**

Inserir mais de um documento.

Usando "insertOne()":

➤ **db.produtos.insertOne({cod: 1, descricao: "Notebook"});**



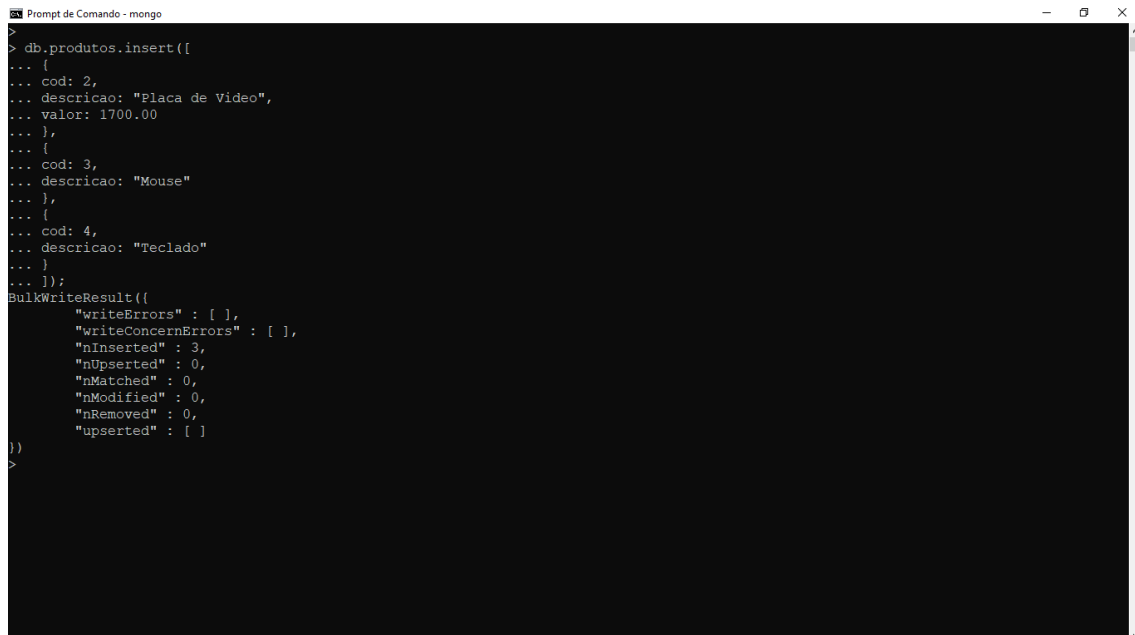
```
Prompt de Comando - mongo
> show dbs;
admin      0.000GB
config     0.000GB
exemplo    0.000GB
local      0.000GB
> use exemplo;
switched to db exemplo
> db.produtos.insertOne({cod: 1, descricao: "Notebook"});
{
  "acknowledged" : true,
  "insertedId" : ObjectId("5aa73c184bbf503534308f97")
}
```

Usando "insert()" com vários parâmetros:

É necessário abrir um vetor para inserir mais de um objeto.

➤ **db.produtos.insert([**
 {
 cod: 2,
 descricao: "Placa de Video",

```
        valor: 1700.00
    },
    {
        cod: 3,
        descricao: "Mouse"
    },
    {
        cod: 4,
        descricao: "Teclado"
    }
]);
```



```
Prompt de Comando - mongo
>
> db.produtos.insert([
... {
...   cod: 2,
...   descricao: "Placa de Video",
...   valor: 1700.00
... },
... {
...   cod: 3,
...   descricao: "Mouse"
... },
... {
...   cod: 4,
...   descricao: "Teclado"
... }
... ]);
BulkWriteResult({
  "writeErrors" : [ ],
  "writeConcernErrors" : [ ],
  "nInserted" : 3,
  "nUpserted" : 0,
  "nMatched" : 0,
  "nModified" : 0,
  "nRemoved" : 0,
  "upserted" : [ ]
})
>
```

Visualizando dados inseridos:

```
Prompt de Comando - mongo
"upserted" : [ ]
})
> db.produtos.find().pretty();
{
  "_id" : ObjectId("5aa73c184bbf503534308f97"),
  "cod" : 1,
  "descricao" : "Notebook"
}
{
  "_id" : ObjectId("5aa73e244bbf503534308f98"),
  "cod" : 2,
  "descricao" : "Placa de Video",
  "valor" : 1700
}
{
  "_id" : ObjectId("5aa73e244bbf503534308f99"),
  "cod" : 3,
  "descricao" : "Mouse"
}
{
  "_id" : ObjectId("5aa73e244bbf503534308f9a"),
  "cod" : 4,
  "descricao" : "Teclado"
}
> -
```

Usando o *"insertMany()*".

```
> db.produtos.insertMany([
  {
    cod: 5,
    descricao: "Monitor",
    tela: 15.4
  },
  {
    cod: 6,
    descricao: "Gabinete"
  },
  {
    cod: 7,
    descricao: "Fonte",
    valor: 575.86
  }
]);
```

```
Prompt de Comando - mongo
> db.produtos.insertMany([
... {
...   cod: 5,
...   descricao: "Monitor",
...   tela: 15.4
... },
... {
...   cod: 6,
...   descricao: "Gabinete"
... },
... {
...   cod: 7,
...   descricao: "Fonte",
...   valor: 575.86
... }
... ]);
{
  "acknowledged" : true,
  "insertedIds" : {
    ObjectId("5aa73fd54bbf503534308f9b"),
    ObjectId("5aa73fd54bbf503534308f9c"),
    ObjectId("5aa73fd54bbf503534308f9d")
  }
}
```

UPDATE.

1. db.collection.update();

Atualiza o primeiro documento encontrado.

2. db.collection.updateOne();

Atualiza o primeiro documento encontrado.

3. db.collection.updateMany();

Atualiza todos os documentos encontrados.

Usando "update()"

```
➤ db.usuarios.update(
    {cod: 2},
    {
      $set: {
        nome: "Jon", email: "jon@gmail.com"
      }
    }
  );
```

```
Prompt de Comando - mongo
> db.usuarios.update(
... {cod: 2},
... {
...   $set: {
...     nome: "Jon", email: "jon@gmail.com"
...   }
... }
... );
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

Visualizando as alterações:

```
Prompt de Comando - mongo
> db.usuarios.find({cod: 2}).pretty();
{
  "_id" : ObjectId("5aa591adac32a6b6c6dd2d8b"),
  "cod" : 2,
  "nome" : "Jon",
  "email" : "jon@gmail.com"
}
```

Utilizando "updateOne()":

```
➤ db.usuarios.updateOne(
    {cod: 3},
    {
      $set: {
        nome: "Dany",
        email: "dan@gmail.com",
        sexo: "M"
      }
    }
  )
```

);

```
Prompt de Comando - mongo
> db.usuarios.updateOne(
... {cod: 3},
... {
...   $set: {
...     nome: "Dany",
...     email: "dan@gmail.com",
...     sexo: "M"
...   }
... }
... );
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
>
```

Visualizando alteração:

```
Prompt de Comando - mongo
> db.usuarios.find({cod: 3}).pretty();
{
  "_id" : ObjectId("5aa591ccac32a6b6c6dd2d8c"),
  "cod" : 3,
  "nome" : "Dany",
  "nascimento" : ISODate("2018-03-11T20:30:04.151Z"),
  "email" : "dan@gmail.com",
  "sexo" : "M"
}
>
```

Nos *“update”* anteriores o comando só altera a primeira coleção que for encontrada de acordo com o parametro informado.

Agora vamos testar o *“updateMany()”*:

Insira dois usuários em que as chaves tenham valores idênticos:

➤ **db.usuarios.insertMany([**

```
{cod: 9, nome: "Mabel", online: 1},  
{cod: 10, nome: "Dipper", online: 1}
```

```
});
```



```
Prompt de Comando - mongo  
> db.usuarios.insertMany([  
... {cod: 9, nome: "Mabel", online: 1},  
... {cod: 10, nome: "Dipper", online: 1}  
... ]);  
{  
  "acknowledged" : true,  
  "insertedIds" : [  
    ObjectId("5aabfa26ee0e3a32ed5352ea"),  
    ObjectId("5aabfa26ee0e3a32ed5352eb")  
  ]  
}
```

Agora use o *"updateMany()"* para modificar todas as coleções que for igual a chave e o valor passado como parametro:

```
➤ db.usuarios.updateMany(  
    {online: 1},  
    {  
        $set: {  
            online: 0  
        }  
    }  
);
```

```
Prompt de Comando - mongo
> db.usuarios.updateMany(
... {online: 1},
... {
...   $set: {
...     online: 0
...   }
... });
{ "acknowledged" : true, "matchedCount" : 2, "modifiedCount" : 2 }
>
```

Visualizando alterações:

```
Prompt de Comando - mongo
> db.usuarios.find({online: 0}).pretty();
{
  "_id" : ObjectId("5aabfa26ee0e3a32ed5352ea"),
  "cod" : 9,
  "nome" : "Mabel",
  "online" : 0
}
{
  "_id" : ObjectId("5aabfa26ee0e3a32ed5352eb"),
  "cod" : 10,
  "nome" : "Dipper",
  "online" : 0
}
>
```

Se o parametro “\$set” não for colocado o objeto é substituído.

Exemplo:

➤ **db.usuarios.update(
 {cod: 10},
 {nome: "Jujuba"}**

);

```
Prompt de Comando - mongo
>
> db.usuarios.update(
... {cod: 10},
... {nome: "Jujuba"}
... );
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

Quando o parametro “\$set” não é informado o objeto é substituído:

```
Prompt de Comando - mongo
>
> db.usuarios.update(
... {cod: 10},
... { "_id" : ObjectId("5aabfa26ee0e3a32ed5352ea"),
...   "cod" : 9,
...   "nome" : "Mabel",
...   "online" : 0
... }
... );
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
> db.usuarios.find(
... { "_id" : ObjectId("5aabfa26ee0e3a32ed5352eb"), "nome" : "Jujuba" }
... );
{ "_id" : ObjectId("5aabfa26ee0e3a32ed5352eb"), "nome" : "Jujuba" }
>
```

O documento com “cod: 10” foi substituído por outro documento que contem apenas o “ObjectId” e a chave “nome”.

DELETE.

1. **db.collection.remove({});**

Remove todos os documentos.

2. **db.collection.deleteOne({});**

Remove o primeiro documento encontrado.

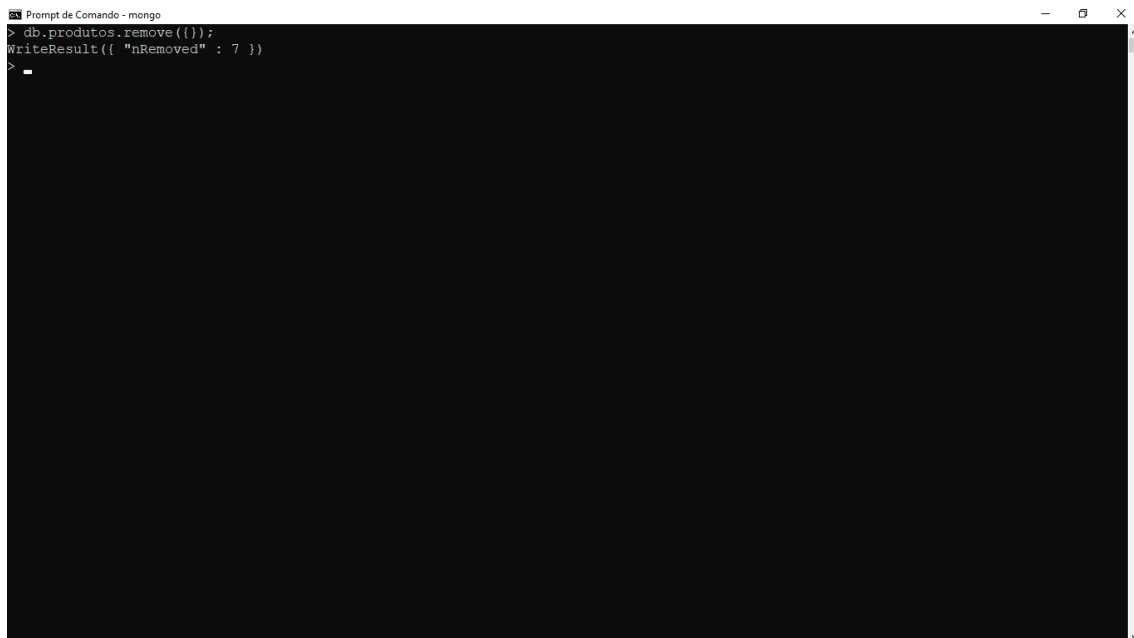
3. **db.collection.deleteMany({});**

Remove todos os documentos encontrados.

Usando o "remove({})":

➤ **db.produtos.remove({});**

Este comando irá remover todos os documentos pois não foi informado nenhum parametro de busca.

A screenshot of a terminal window titled "Prompt de Comando - mongo". The terminal shows the command `> db.produtos.remove({});` being executed, followed by the output `WriteResult({ "nRemoved" : 7 })`. The terminal background is black, and the text is white. The window has standard Windows-style window controls (minimize, maximize, close) in the top right corner.

```
Prompt de Comando - mongo
> db.produtos.remove({});
WriteResult({ "nRemoved" : 7 })
>
```

Removendo apenas o documento especificado:

➤ **db.usuarios.remove({nome: "Jujuba"});**

```
Prompt de Comando - mongo
> db.usuarios.remove({nome: "Jujuba"});
WriteResult({ "nRemoved" : 1 })
>
```

READ.

1. **db.collection.find({});**

Retorna todos os documentos da coleção.

2. **db.collection.find().pretty();**

Retorna todos os documentos em formato Legível(JSON).

3. **db.collection.find({chave: valor}).pretty();**

Retorna todos os documentos encontrados. Aceita mais de um parametro.

3. **db.collection.findOne({});**

Retorna o primeiro documento encontrado.

Usando o "findOne()";

➤ **db.alunos.findOne({});**

```
Prompt de Comando - mongo
> db.alunos.findOne({});
{
  "_id" : ObjectId("5aac99196c9822f5ce5795cb"),
  "AlunoId" : 1,
  "nome" : "Lauren",
  "endereco" : "Rua A",
  "telefone" : "(18) 9980-36954",
  "curso" : "Java"
}
```

Usando “find({chave: valor})”.

➤ **db.alunos.find({curso: “Assembly”}).pretty();**

```
Prompt de Comando - mongo
> db.alunos.find({curso: "Assembly"}).pretty();
{
  "_id" : ObjectId("5aac99196c9822f5ce5795cd"),
  "AlunoId" : 3,
  "nome" : "Joseph",
  "endereco" : "Rua B",
  "telefone" : "(18) 9980-25478",
  "curso" : "Assembly"
}
{
  "_id" : ObjectId("5aac99196c9822f5ce5795d4"),
  "AlunoId" : 10,
  "nome" : "Isaac",
  "endereco" : "Rua Z",
  "telefone" : "(18) 9980-66687",
  "curso" : "Assembly"
}
```

Usando “findOne({chave: valor})”.

➤ **db.alunos.find({curso: “Assembly”}).pretty();**

```
Prompt de Comando - mongo
> db.alunos.findOne({curso: "Assembly"});
{
  "_id" : ObjectId("5aac99196c9822f5ce5795cd"),
  "AlunoId" : 3,
  "nome" : "Joseph",
  "endereco" : "Rua B",
  "telefone" : "(18) 9980-25478",
  "curso" : "Assembly"
}
```

Operadores Lógicos:

\$and:

```
➤ db.alunos.find({
    $and: [
        {endereco: "Rua A"},
        {curso: "Java"}
    ]
}).pretty();
```

```
Prompt de Comando - mongo
> db.alunos.find({
... $and: [
... {endereco: "Rua A"},
... {curso: "Java"}
... ]
... })
... }).pretty();
{
  "_id" : ObjectId("5aac99196c9822f5ce5795cb"),
  "AlunoId" : 1,
  "nome" : "Lauren",
  "endereco" : "Rua A",
  "telefone" : "(18) 9980-36954",
  "curso" : "Java"
}
```

\$or:

```
➤ db.alunos.find({
    $or: [
        {curso: "Java"},
        {curso: "Python"}
    ]
}).pretty();
```

```
Prompt de Comando - mongo
> db.alunos.find({
... $or: [
... {curso: "Java"},
... {curso: "Python"}
... ]
... })
... }).pretty();
{
  "_id" : ObjectId("5aac99196c9822f5ce5795cb"),
  "AlunoId" : 1,
  "nome" : "Lauren",
  "endereço" : "Rua A",
  "telefone" : "(18) 9980-36954",
  "curso" : "Java"
}
{
  "_id" : ObjectId("5aac99196c9822f5ce5795ce"),
  "AlunoId" : 4,
  "nome" : "Annie",
  "endereço" : "Rua A",
  "telefone" : "(18) 9225-36954",
  "curso" : "Python"
}
{
  "_id" : ObjectId("5aac99196c9822f5ce5795d0"),
  "AlunoId" : 6,
  "nome" : "Sophie",
  "endereço" : "Rua B",
  "telefone" : "(18) 9980-36954",
  "curso" : "Java"
}
>
```

\$exists:

Este operador verifica se a chave/atributo existe na coleção.

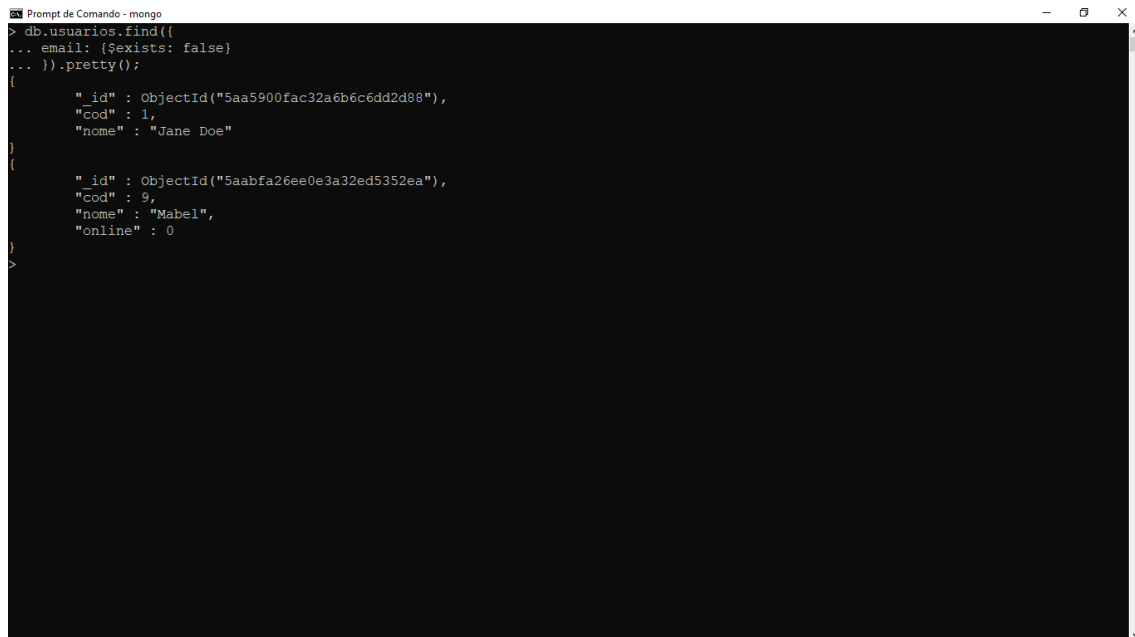
➤ **db.usuarios.find({
 nascimento: {\$exists: true}
}).pretty();**

Retorna apenas os documentos ao qual o atributo nascimento exista.

```
Prompt de Comando - mongo
> db.usuarios.find({
... nascimento: {$exists: true}
... }).pretty();
{
  "_id" : ObjectId("5aa591ccac32a6b6c6dd2d8c"),
  "Cod" : 3,
  "nome" : "Dany",
  "nascimento" : ISODate("2018-03-11T20:30:04.151Z"),
  "email" : "dan@gmail.com",
  "sexo" : "M"
}
>
```

➤ **db.usuarios.find({
 email: {\$exists: false}
}).pretty();**

Retorna apenas os documentos ao qual o atributo email não exista.



```
Prompt de Comando - mongo
> db.usuarios.find(
... email: {$exists: false}
... ).pretty();
{
  "_id" : ObjectId("5aa5900fac32a6b6c6dd2d88"),
  "cod" : 1,
  "nome" : "Jane Doe"
}
{
  "_id" : ObjectId("5aabfa26ee0e3a32ed5352ea"),
  "cod" : 9,
  "nome" : "Mabel",
  "online" : 0
}
>
```

\$not:

➤ **db.alunos.find({
 endereco: {
 \$not: {
 \$eq: "Rua A"
 }
 }
}).pretty();**

Retorna os documentos em que o endereco não seja "Rua A".


```
Prompt de Comando - mongo
> db.alunos.find({
... endereco: {
... $not: {
... $eq: "Rua A"
... }
... }
... }).pretty();
{
  "_id" : ObjectId("5aac99196c9822f5ce5795cd"),
  "AlunoId" : 3,
  "nome" : "Joseph",
  "endereco" : "Rua B",
  "telefone" : "(18) 9980-25478",
  "curso" : "Assembly"
}
{
  "_id" : ObjectId("5aac99196c9822f5ce5795cf"),
  "AlunoId" : 5,
  "nome" : "Nathan",
  "endereco" : "Rua D",
  "telefone" : "(18) 9980-11125",
  "curso" : "HTML 5 e CSS3"
}
{
  "_id" : ObjectId("5aac99196c9822f5ce5795d0"),
  "AlunoId" : 6,
  "nome" : "Sophie",
  "endereco" : "Rua B",
  "telefone" : "(18) 9980-36954",
  "curso" : "Java"
}
{
  "_id" : ObjectId("5aac99196c9822f5ce5795d3"),
  "AlunoId" : 9,
  "nome" : "Matheus",
  "endereco" : "Rua E",
  "telefone" : "(18) 9980-33338",
  "curso" : "Python"
}
```

\$nor:

```
➤ db.alunos.find({
    $nor: [
        {curso: "Java"},
        {endereco: "Rua A"}
    ]
}).pretty();
```

Retorna os documentos que não possuam um dos parâmetros passados.

```
Prompt de Comando - mongo
> db.alunos.find({
... $nor: [
... {curso: "Java"},
... {endereco: "Rua A"}
... ]
... }).pretty();
{
  "_id" : ObjectId("5aac99196c9822f5ce5795cd"),
  "AlunoId" : 3,
  "nome" : "Joseph",
  "endereco" : "Rua B",
  "telefone" : "(18) 9980-25478",
  "curso" : "Assembly"
}
{
  "_id" : ObjectId("5aac99196c9822f5ce5795cf"),
  "AlunoId" : 5,
  "nome" : "Nathan",
  "endereco" : "Rua D",
  "telefone" : "(18) 9980-11125",
  "curso" : "HTML 5 e CSS3"
}
{
  "_id" : ObjectId("5aac99196c9822f5ce5795d3"),
  "AlunoId" : 9,
  "nome" : "Matheus",
  "endereco" : "Rua E",
  "telefone" : "(18) 9980-33338",
  "curso" : "Web Design"
}
{
  "_id" : ObjectId("5aac99196c9822f5ce5795d4"),
  "AlunoId" : 10,
  "nome" : "Isaac",
  "endereco" : "Rua Z",
  "telefone" : "(18) 9980-66687",
  "curso" : "Assembly"
}
```

Operadores de Comparação:

\$eq:

```
➤ db.produtos.find({
    quantidade: {
        $not: {
            $eq: 10
        }
    }
}).pretty();
```

Este é o operador de igualdade geralmente é mais usado em conjunto com o operador \$not.

```
Prompt de Comando - mongo
> db.produtos.find({
... quantidade: {
... $not: {
... $eq: 10
... }
... }
... }).pretty();
{
  "_id" : ObjectId("5aad7c51d9d708d7640b266b"),
  "codigo" : 1,
  "descricao" : "PC",
  "valor" : 1500,
  "quantidade" : 15
}
{
  "_id" : ObjectId("5aad7c51d9d708d7640b266c"),
  "codigo" : 2,
  "descricao" : "Placa de Video",
  "valor" : 2000
}
{
  "_id" : ObjectId("5aad7c51d9d708d7640b266d"),
  "codigo" : 3,
  "descricao" : "Processador",
  "valor" : 700
}
> -
```

\$ne:

➤ **db.produtos.find({
 quantidade: {
 \$ne: 10
 }
}).pretty();**

Comando para negar igualdade, numa forma mais simplificada.

```
Prompt de Comando - mongo
> db.produtos.find({
... quantidade: {
... $ne: 10
... }
... }).pretty();
{
  "_id" : ObjectId("5aad7c51d9d708d7640b266b"),
  "codigo" : 1,
  "descricao" : "PC",
  "valor" : 1500,
  "quantidade" : 15
}
{
  "_id" : ObjectId("5aad7c51d9d708d7640b266c"),
  "codigo" : 2,
  "descricao" : "Placa de Video",
  "valor" : 2000
}
{
  "_id" : ObjectId("5aad7c51d9d708d7640b266d"),
  "codigo" : 3,
  "descricao" : "Processador",
  "valor" : 700
}
> -
```

Operadores de Maior e Menor que.

Operador Maior que:

\$gt:

```
➤ db.alunos.find({
    AlunoId: {
        $gt: 9
    }
}).pretty();
```



The screenshot shows a MongoDB command prompt window titled "Selecionar Prompt de Comando - mongo". The user enters the command `db.alunos.find({ AlunoId: { $gt: 9 } }).pretty();`. The output displays three documents from the `alunos` collection, all with `AlunoId` values greater than 9. Each document includes fields for `_id`, `AlunoId`, `nome`, `endereco`, `telefone`, and `curso`.

```
> db.alunos.find({
... AlunoId: {
...   $gt: 9
... }
... }).pretty();
{
  "_id" : ObjectId("5aac99196c9822f5ce5795d4"),
  "AlunoId" : 10,
  "nome" : "Isaac",
  "endereco" : "Rua Z",
  "telefone" : "(18) 9980-66687",
  "curso" : "Assembly"
}
{
  "_id" : ObjectId("5aac99196c9822f5ce5795d5"),
  "AlunoId" : 11,
  "nome" : "Eliza",
  "endereco" : "Rua F",
  "telefone" : "(18) 1125-36954",
  "curso" : "MySQL"
}
{
  "_id" : ObjectId("5aac99196c9822f5ce5795d6"),
  "AlunoId" : 12,
  "nome" : "Valentina",
  "endereco" : "Rua G",
  "telefone" : "(18) 9450-36451",
  "curso" : "Rust"
}
>
```

Operador Menor que:

\$lt:

```
➤ db.produtos.find({
    valor: {
        $lt: 1000
    }
}).pretty();
```

```
Prompt de Comando - mongo
> db.produtos.find({
... valor: {
... $lt: 1000
... }
... }).pretty();
{
  "_id" : ObjectId("5aad7c51d9d708d7640b266d"),
  "codigo" : 3,
  "descricao" : "Processador",
  "valor" : 700
}
{
  "_id" : ObjectId("5aad7c51d9d708d7640b266e"),
  "codigo" : 4,
  "descricao" : "Mouse",
  "valor" : 200,
  "quantidade" : 10
}
{
  "_id" : ObjectId("5aad7c51d9d708d7640b266f"),
  "codigo" : 5,
  "descricao" : "Teclado",
  "valor" : 89.99,
  "quantidade" : 10
}
>
```

Operador Menor ou igual à:

\$lte:

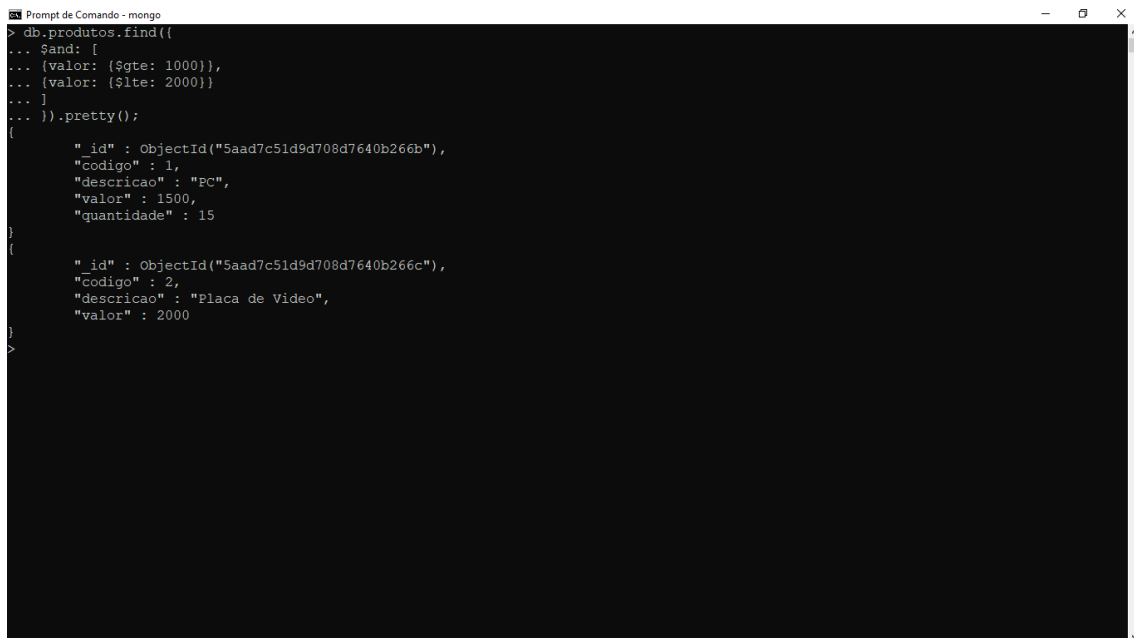
➤ **db.produtos.find({
 valor: {
 \$lte: 1500
 }
}).pretty();**

```
Prompt de Comando - mongo
> db.produtos.find({
... valor: {
... $lte: 1500
... }
... }).pretty();
{
  "_id" : ObjectId("5aad7c51d9d708d7640b266b"),
  "codigo" : 1,
  "descricao" : "PC",
  "valor" : 1500,
  "quantidade" : 15
}
{
  "_id" : ObjectId("5aad7c51d9d708d7640b266d"),
  "codigo" : 3,
  "descricao" : "Processador",
  "valor" : 700
}
{
  "_id" : ObjectId("5aad7c51d9d708d7640b266e"),
  "codigo" : 4,
  "descricao" : "Mouse",
  "valor" : 200,
  "quantidade" : 10
}
{
  "_id" : ObjectId("5aad7c51d9d708d7640b266f"),
  "codigo" : 5,
  "descricao" : "Teclado",
  "valor" : 89.99,
  "quantidade" : 10
}
>
```

Operador Maior ou igual à:

\$gte:

```
➤ db.produtos.find({
    $and: [
        {valor: {$gte: 1000}},
        {valor: {$lte: 2000}}
    ]
}).pretty();
```



```
Prompt de Comando - mongo
> db.produtos.find({
... $and: [
... {valor: {$gte: 1000}},
... {valor: {$lte: 2000}}
... ]
... }).pretty();
{
  "_id" : ObjectId("5aad7c51d9d708d7640b266b"),
  "codigo" : 1,
  "descricao" : "PC",
  "valor" : 1500,
  "quantidade" : 15
}
{
  "_id" : ObjectId("5aad7c51d9d708d7640b266c"),
  "codigo" : 2,
  "descricao" : "Placa de Video",
  "valor" : 2000
}
>
```

Projections.

Visualizar atributos específicos:

```
db.collection.find({}, {}).pretty;
```

A primeira chave determina os valores a serem encontrados.

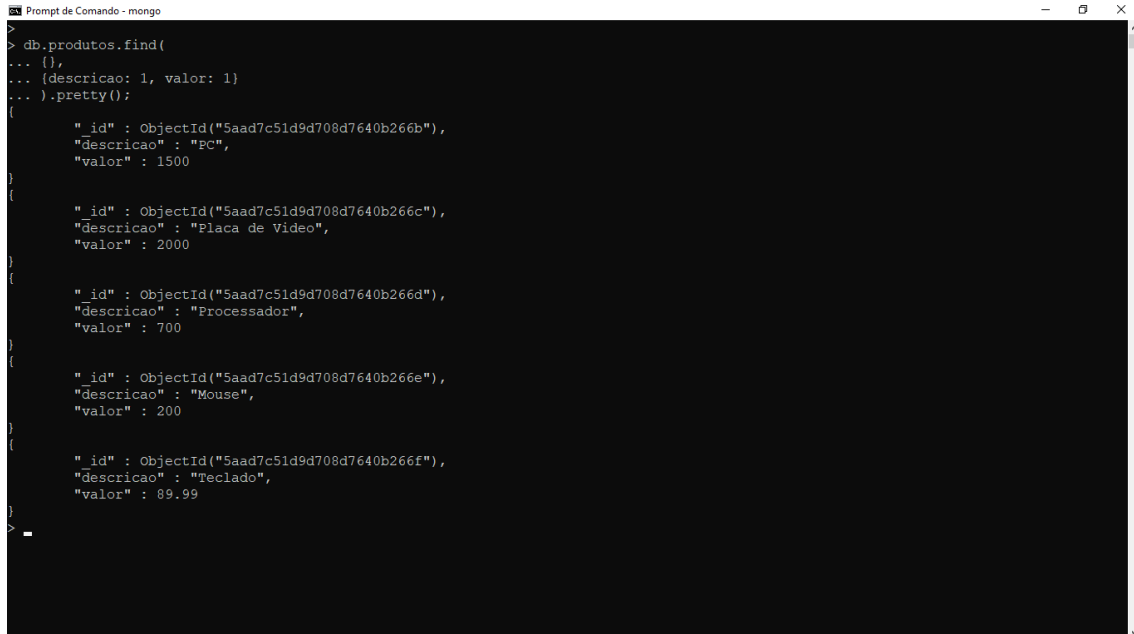
A segunda chave determina as colunas que serão mostradas.

Exemplo:

```
➤ db.produtos.find(
```

```
    {},  
    {descricao: 1, valor: 1}  
  ).pretty();
```

Apenas os atributos com valores 1 serão mostrados



```
Prompt de Comando - mongo  
> db.produtos.find(  
... {},  
... {descricao: 1, valor: 1}  
... ).pretty();  
{  
  "_id" : ObjectId("5aad7c51d9d708d7640b266b"),  
  "descricao" : "Pc",  
  "valor" : 1500  
}  
{  
  "_id" : ObjectId("5aad7c51d9d708d7640b266c"),  
  "descricao" : "Placa de Video",  
  "valor" : 2000  
}  
{  
  "_id" : ObjectId("5aad7c51d9d708d7640b266d"),  
  "descricao" : "Processador",  
  "valor" : 700  
}  
{  
  "_id" : ObjectId("5aad7c51d9d708d7640b266e"),  
  "descricao" : "Mouse",  
  "valor" : 200  
}  
{  
  "_id" : ObjectId("5aad7c51d9d708d7640b266f"),  
  "descricao" : "Teclado",  
  "valor" : 89.99  
}  
>
```

Como omitir colunas:

```
➤ db.produtos.find(  
    {},  
    {valor: 0, quantidade: 0}  
  ).pretty();
```

Os atributos com valores 0 serão escondidos.

```
Prompt de Comando - mongo
> db.produtos.find(
... {},
... {valor: 0, quantidade: 0}
... ).pretty();
{
  "_id" : ObjectId("5aad7c51d9d708d7640b266b"),
  "codigo" : 1,
  "descricao" : "Pc"
}
{
  "_id" : ObjectId("5aad7c51d9d708d7640b266c"),
  "codigo" : 2,
  "descricao" : "Placa de Video"
}
{
  "_id" : ObjectId("5aad7c51d9d708d7640b266d"),
  "codigo" : 3,
  "descricao" : "Processador"
}
{
  "_id" : ObjectId("5aad7c51d9d708d7640b266e"),
  "codigo" : 4,
  "descricao" : "Mouse"
}
{
  "_id" : ObjectId("5aad7c51d9d708d7640b266f"),
  "codigo" : 5,
  "descricao" : "Teclado"
}
>
```

Buscas Ordenadas:

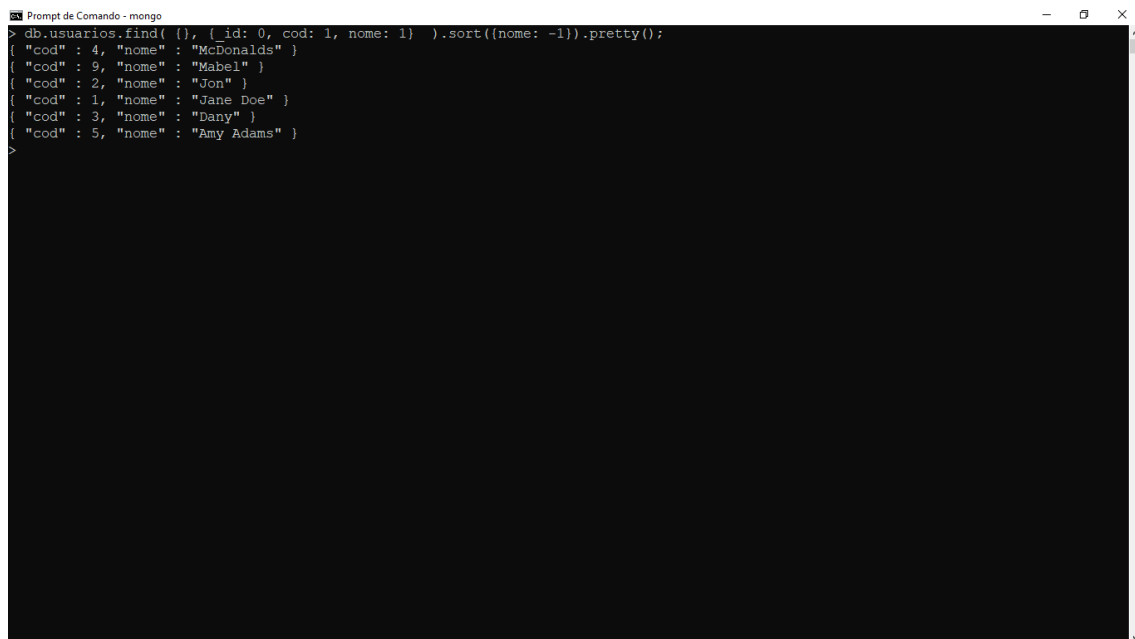
SORT.

➤ **db.usuarios.find(**
 {
 {_id: 0, cod: 1, nome: 1}
 }).sort({cod: -1}).pretty();

O parametro "-1" indica que será ordenado em ordem decrescente.

```
Prompt de Comando - mongo
> db.usuarios.find( {}, {_id: 0, cod: 1, nome: 1} ).sort({cod: -1}).pretty();
{ "cod" : 9, "nome" : "Mabel" }
{ "cod" : 5, "nome" : "Amy Adams" }
{ "cod" : 4, "nome" : "McDonalds" }
{ "cod" : 3, "nome" : "Dany" }
{ "cod" : 2, "nome" : "Jon" }
{ "cod" : 1, "nome" : "Jane Doe" }
>
```


➤ **db.usuarios.find(
 {},
 {_id: 0, cod: 1, nome: 1}
).sort({nome: -1}).pretty();**



```
Prompt de Comando - mongo
> db.usuarios.find( {}, { _id: 0, cod: 1, nome: 1 } ).sort({nome: -1}).pretty();
{ "cod" : 4, "nome" : "McDonalds" }
{ "cod" : 9, "nome" : "Mabel" }
{ "cod" : 2, "nome" : "Jon" }
{ "cod" : 1, "nome" : "Jane Doe" }
{ "cod" : 3, "nome" : "Dany" }
{ "cod" : 5, "nome" : "Amy Adams" }
```

Ordenando em ordem Crescente:

➤ **db.alunos.find(
 {},
 {_id: 0, nome: 1}
).sort({nome: 1}).pretty();**

```
Prompt de Comando - mongo
> db.alunos.find(
... {},
... {_id: 0, nome: 1}
... ).sort({nome: 1}).pretty();
{ "nome" : "Annie" }
{ "nome" : "Charlie" }
{ "nome" : "Eliza" }
{ "nome" : "Emma" }
{ "nome" : "Isaac" }
{ "nome" : "Joseph" }
{ "nome" : "Lauren" }
{ "nome" : "Matheus" }
{ "nome" : "Nathan" }
{ "nome" : "Patrick" }
{ "nome" : "Sophie" }
{ "nome" : "Valentina" }
>
```

LIMIT.

Limitando a quantidade de documentos que serão mostrados.

Exemplos:

➤ **db.usuarios.find().limit(3).pretty();**

```
Prompt de Comando - mongo
> db.usuarios.find().limit(3).pretty();
{
  "_id" : ObjectId("5aa5900fac32a6b6c6dd2d88"),
  "cod" : 1,
  "nome" : "Jane Doe"
}
{
  "_id" : ObjectId("5aa591adac32a6b6c6dd2d8b"),
  "cod" : 2,
  "nome" : "Jon",
  "email" : "jon@gmail.com"
}
{
  "_id" : ObjectId("5aa591ccac32a6b6c6dd2d8c"),
  "cod" : 3,
  "nome" : "Dany",
  "nascimento" : ISODate("2018-03-11T20:30:04.151Z"),
  "email" : "dan@gmail.com",
  "sexo" : "M"
}
>
```

➤ **db.alunos.find(
 {},
 {_id: 0, curso: 1}
).limit(5).sort({curso: 1}).pretty();**

```
Prompt de Comando - mongo
> db.alunos.find(
... {},
... { _id: 0, curso: 1}
... ).limit(5).sort({curso: 1}).pretty();
{ "curso" : ".Net Visual Studio C#" }
{ "curso" : "Assembly" }
{ "curso" : "Assembly" }
{ "curso" : "C++" }
{ "curso" : "HTML 5 e CSS3" }
```

SKIP.

Pula a quantidade de documentos que for passado como parametro.

Exemplos:

➤ **db.produtos.find().skip(2).pretty();**

```
Prompt de Comando - mongo
> db.produtos.find().skip(2).pretty();
{
  "_id" : ObjectId("5aad7c51d9d708d7640b266d"),
  "codigo" : 3,
  "descricao" : "Processador",
  "valor" : 700
}
{
  "_id" : ObjectId("5aad7c51d9d708d7640b266e"),
  "codigo" : 4,
  "descricao" : "Mouse",
  "valor" : 200,
  "quantidade" : 10
}
{
  "_id" : ObjectId("5aad7c51d9d708d7640b266f"),
  "codigo" : 5,
  "descricao" : "Teclado",
  "valor" : 89.99,
  "quantidade" : 10
}
```

➤ **db.alunos.find(**

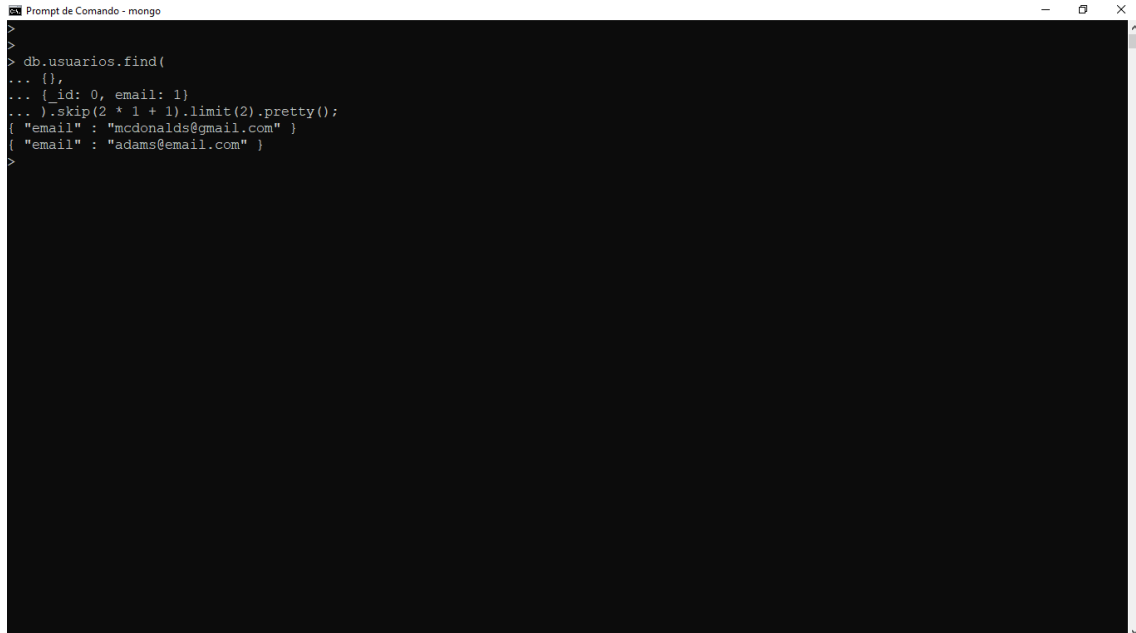
```
    {},  
    {_id: 0, nome: 1}  
  ).sort({nome: 1}).skip(10).pretty();
```

```
Prompt de Comando - mongo  
> db.alunos.find(  
... {},  
... {_id: 0, nome: 1}  
... ).sort({nome: 1}).skip(10).pretty();  
{ "nome" : "Sophie" }  
{ "nome" : "Valentina" }  
>
```

➤ **db.produtos.find(
 {},
 {_id: 0, descricao: 1}
).skip(2 * 1).limit(2).pretty();**

```
Prompt de Comando - mongo  
> db.produtos.find(  
... {},  
... {_id: 0, descricao: 1}  
... ).skip(2 * 1).limit(2).pretty();  
{"descricao" : "Processador" }  
{"descricao" : "Mouse" }  
>
```

➤ **db.usuarios.find(
 {},
 {_id: 0, email: 1}
).skip(2 * 1 + 1).limit(2).pretty();**



```
Prompt de Comando - mongo
> db.usuarios.find(
... {},
... {_id: 0, email: 1}
... ).skip(2 * 1 + 1).limit(2).pretty();
{ "email" : "mcdonalds@gmail.com" }
{ "email" : "adams@email.com" }
>
```

Busca por palavras específicas:

\$regex:

➤ **db.alunos.find({nome: {\$regex: /u/}}).pretty();**

“\$regex” é semelhante ao “like” do MySQL.

Neste exemplo busca apenas os nomes que possuam a letra “u”.

```
Prompt de Comando - mongo
> db.alunos.find([nome: {$regex: /u/}]).pretty();
{
  "_id" : ObjectId("5aac99196c9822f5ce5795cb"),
  "AlunoId" : 1,
  "nome" : "Lauren",
  "endereco" : "Rua A",
  "telefone" : "(18) 9980-36954",
  "curso" : "Java"
}
{
  "_id" : ObjectId("5aac99196c9822f5ce5795d3"),
  "AlunoId" : 9,
  "nome" : "Matheus",
  "endereco" : "Rua B",
  "telefone" : "(18) 9980-33338",
  "curso" : "Web Design"
}
>
```

Contar documentos:

COUNT.

➤ **db.alunos.find({}).count();**

```
Prompt de Comando - mongo
> db.alunos.find({}).count();
12
>
```

Associações.

Criando uma coleção de clientes:

- **db.clientes.insert({ ClienteId: 1, nome: "Marie"});**

```
Prompt de Comando - mongo
> db.clientes.insert({ ClienteId: 1, nome: "Marie"});
WriteResult({ "nInserted" : 1 })
>
```

Salve este cliente numa variável:

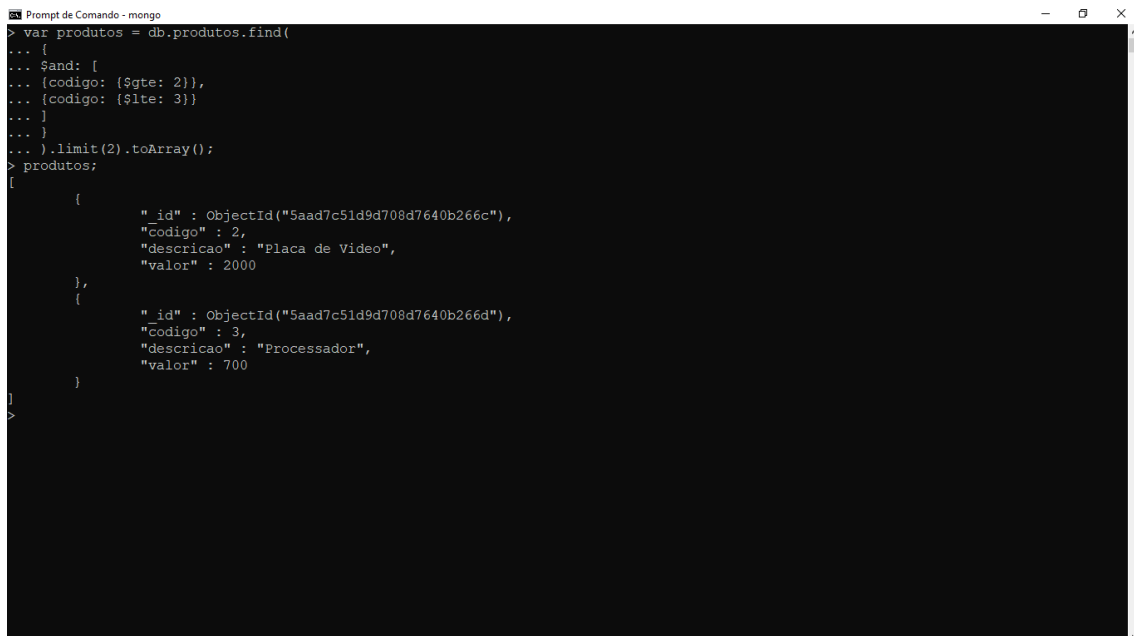
- **var cliente = db.clientes.findOne({ClienteId: 1});**

```
Prompt de Comando - mongo
> var cliente = db.clientes.findOne({ClienteId: 1});
> cliente
{
  "_id" : ObjectId("5aadb9c998484308cee789b60"),
  "ClienteId" : 1,
  "nome" : "Marie"
}
>
```

Agora é necessário selecionar os produtos comprados e guardar numa variável.

Caso retorne mais de um objeto é necessário adicionar a função *"toArray()"* para salvar todos os dados na variável.

```
➤ var produtos = db.produtos.find(  
  {  
    $and: [  
      {codigo: {$gte: 2}},  
      {codigo: {$lte: 3}}  
    ]  
  }  
)  
.limit(2).toArray();
```



```
Prompt de Comando - mongo  
> var produtos = db.produtos.find(  
... {  
... $and: [  
... {codigo: {$gte: 2}},  
... {codigo: {$lte: 3}}  
... ]  
... }  
... ).limit(2).toArray();  
> produtos;  
[  
  {  
    "_id" : ObjectId("5aad7c51d9d708d7640b266c"),  
    "codigo" : 2,  
    "descricao" : "Placa de Video",  
    "valor" : 2000  
  },  
  {  
    "_id" : ObjectId("5aad7c51d9d708d7640b266d"),  
    "codigo" : 3,  
    "descricao" : "Processador",  
    "valor" : 700  
  }  
]
```

Conferindo os dados das variáveis:


```
Prompt de Comando - mongo
> cliente
{
  "_id" : ObjectId("5aadbc998484308cee789b60"),
  "ClienteId" : 1,
  "nome" : "Marie"
}
> produtos
[
  {
    "_id" : ObjectId("5aad7c51d9d708d7640b266c"),
    "Codigo" : 2,
    "descricao" : "Placa de Video",
    "valor" : 2000
  },
  {
    "_id" : ObjectId("5aad7c51d9d708d7640b266d"),
    "Codigo" : 3,
    "descricao" : "Processador",
    "valor" : 700
  }
]
>
```

Agora é só inserir as variáveis na coleção de vendas:

```
➤ db.vendas.insert(
  {
    cliente: cliente,
    produtos: produtos
  }
);
```

```
Prompt de Comando - mongo
> db.vendas.insert(
... {
... cliente: cliente,
... produtos: produtos
... }
... );
WriteResult({ "nInserted" : 1 })
>
```

Visualizando os dados inseridos:

```
Prompt de Comando - mongo
> db.vendas.find().pretty();
{
  "_id" : ObjectId("5aadcc408484308cee789b6a"),
  "cliente" : {
    "_id" : ObjectId("5aadbc998484308cee789b60"),
    "clienteId" : 1,
    "nome" : "Marie"
  },
  "produtos" : [
    {
      "_id" : ObjectId("5aad7c51d9d708d7640b266c"),
      "codigo" : 2,
      "descricao" : "Placa de Video",
      "valor" : 2000
    },
    {
      "_id" : ObjectId("5aad7c51d9d708d7640b266d"),
      "codigo" : 3,
      "descricao" : "Processador",
      "valor" : 700
    }
  ]
}
```

Venda realizada com sucesso!

Fim.