Banco de dados NoSQL - MongoDB (Parte 1)

Prof. Gustavo Leitão

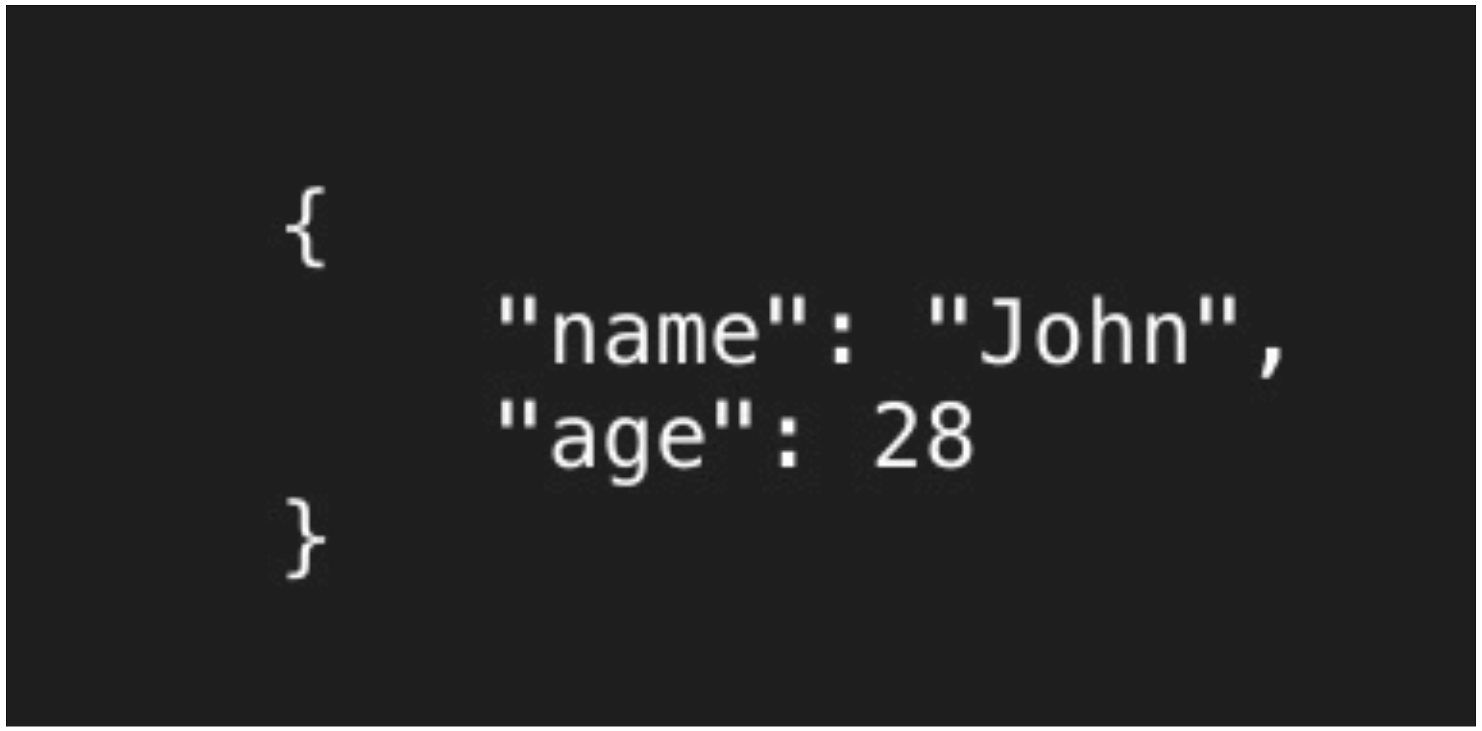
AGENDA

- Introdução ao MongoDB
- Instalação com Docker
- Comandos básicos
- Operações

- MongoDB é um banco NoSQL orientado a Documento.
- Começou a ser desenvolvido em 2007 pela 10gen, mas só em 2009 passou a ser open source
- Escrito em C++
- Possui suporte a diversas plataformas: windows, linux, solaris, freebsd, macOS

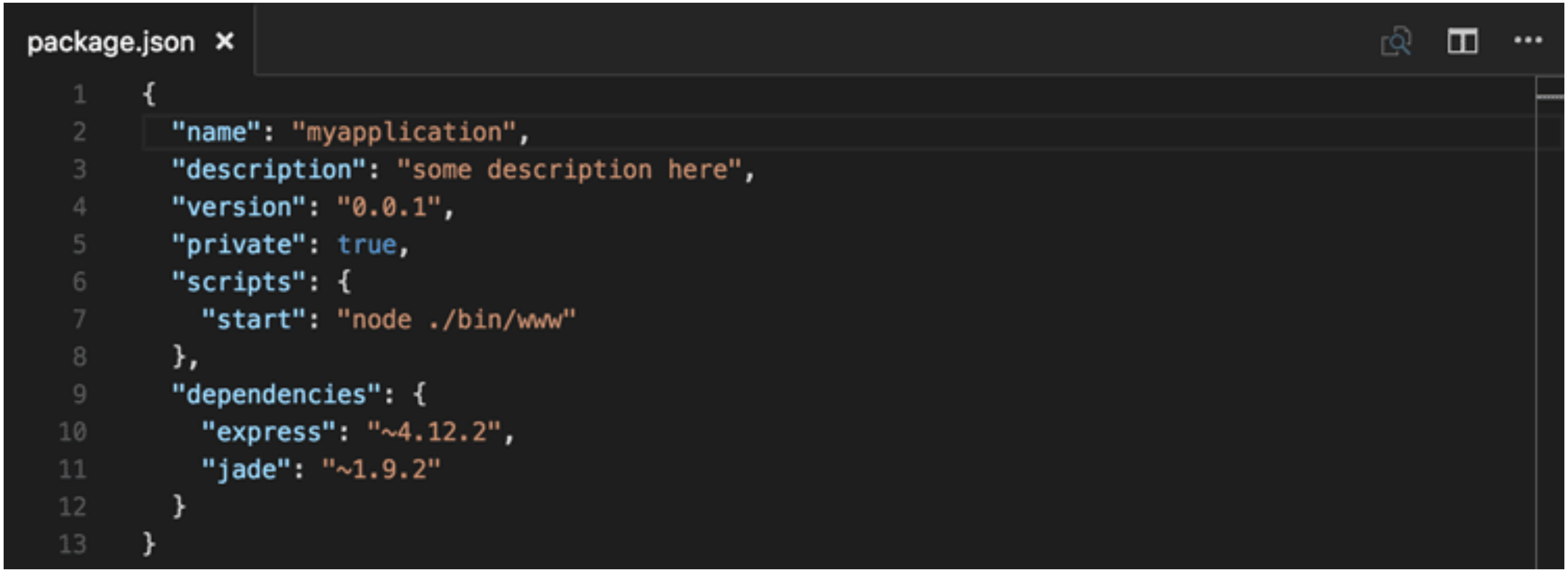
- MongoDB possui Schema-free
- Possui escalabilidade horizontal
  - Alta performance
  - Alta disponibilidade

- Documentos são armazenados serializados em BSON (Binary JSON)



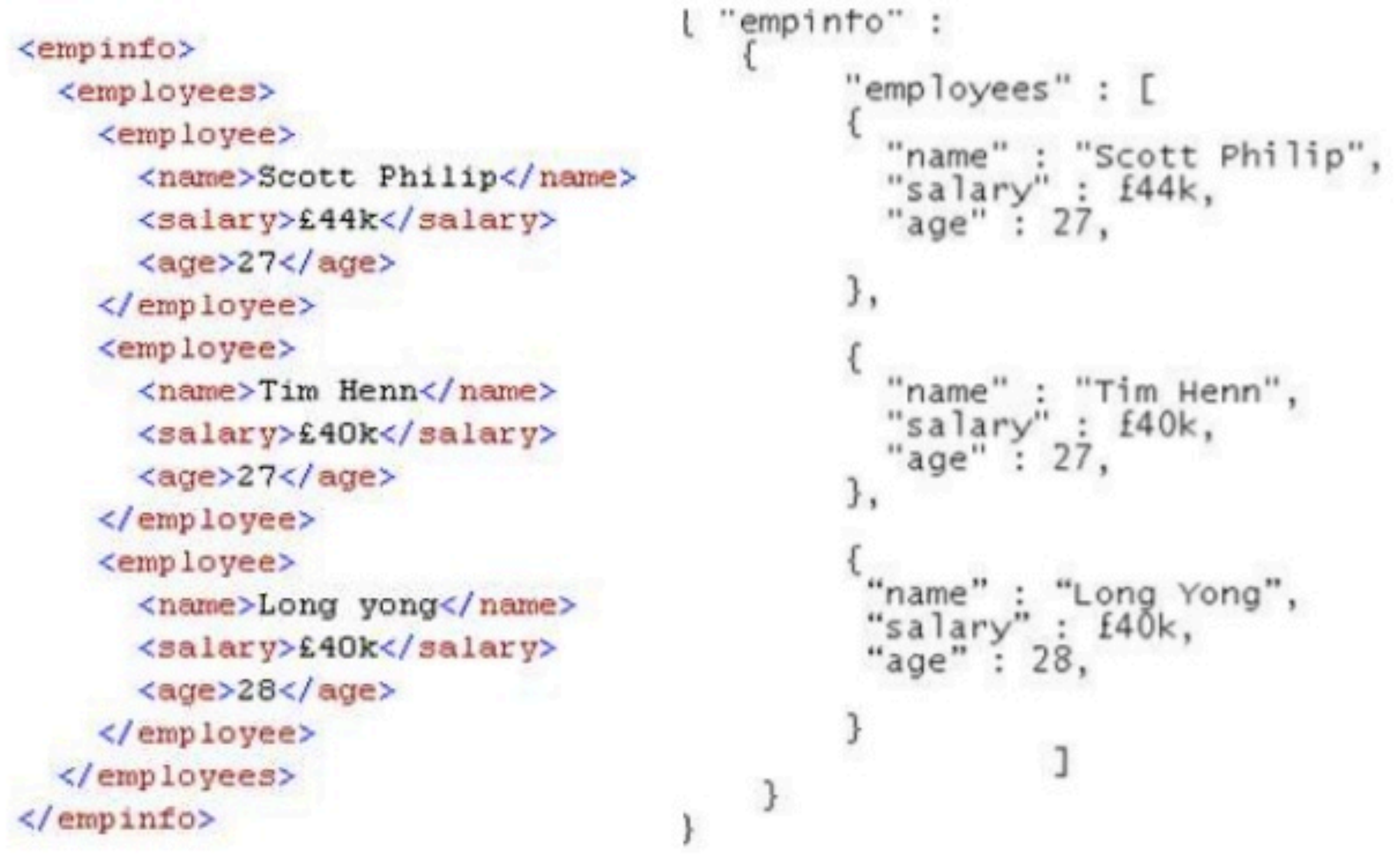
```
{
    "name": "John",
    "age": 28
}
```

mongoDB®


package.json ×

```
{
  "name": "myapplication",
  "description": "some description here",
  "version": "0.0.1",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "express": "~4.12.2",
    "jade": "~1.9.2"
  }
}
```

XML vs JSON



```xml
<empinfo>
  <employees>
    <employee>
      <name>Scott Philip</name>
      <salary>£44k</salary>
      <age>27</age>
    </employee>
    <employee>
      <name>Tim Henn</name>
      <salary>£40k</salary>
      <age>27</age>
    </employee>
    <employee>
      <name>Long yong</name>
      <salary>£40k</salary>
      <age>28</age>
    </employee>
  </employees>
</empinfo>
```

```json
{ "empinfo" :
   {
        "employees" : [
        {
         "name" : "Scott Philip",
         "salary" : £44k,
         "age" : 27,

        },

        {
         "name" : "Tim Henn",
         "salary" : £40k,
         "age" : 27,
        },

        {
        "name" : "Long Yong",
        "salary" : £40k,
        "age" : 28,

        }
                   ]
    }
}
```

mongoDB®

SGBD	MongoDB
Database	Database
Table	Collection
Row	Document
Coluna	Field
Primary Key	Primary Key (_id)
Index	Index

Instalação

https://www.docker.com/

INSTALAÇÃO

Instalando o MongoDB

```
docker run -p 27017:27017 --name nosql-mongo -v /home/
mongo:/data/ -d mongo
```

INSTALAÇÃO

Acessando interactive shell do mongo

```
docker exec -it nosql-mongo mongo
```

Primeiros comandos…

Primeiros Comandos

Exibindo bancos de dados

```
show dbs
```

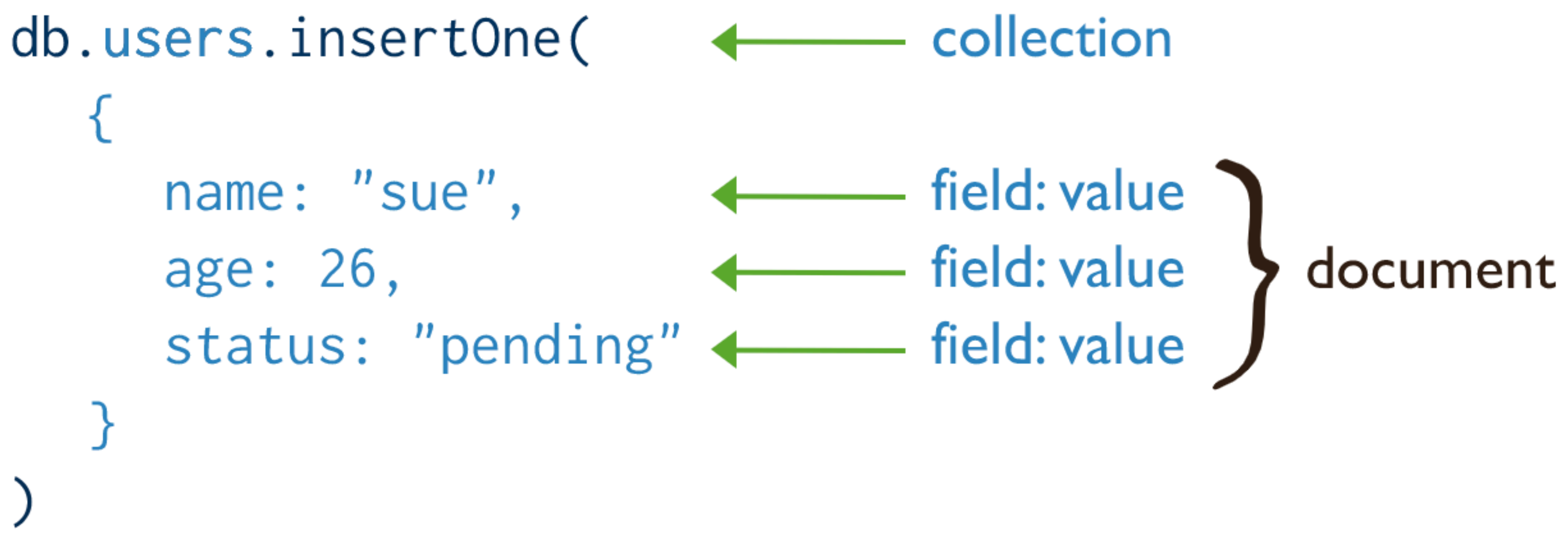
Primeiros Comandos

Para utilizar o database test (caso não exista será criado automaticamente)

```
use test
```

Primeiros Comandos

Inserindo um documento…



```
db.users.insertOne(
   {
      name: "sue",
      age: 26,
      status: "pending"
   }
)
```

Primeiros Comandos

Inserindo um documento…

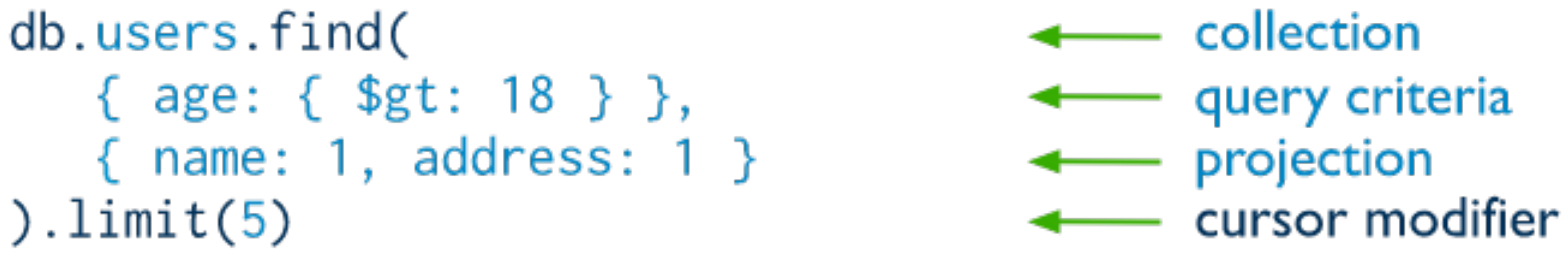
```
db.usuarios.insertOne({nome: 'newton',
    email: 'newton@gmail.com',
 idade: 53})
```

Primeiros Comandos

Buscando todos os documentos

```
db.usuarios.find();
```

Primeiros Comandos

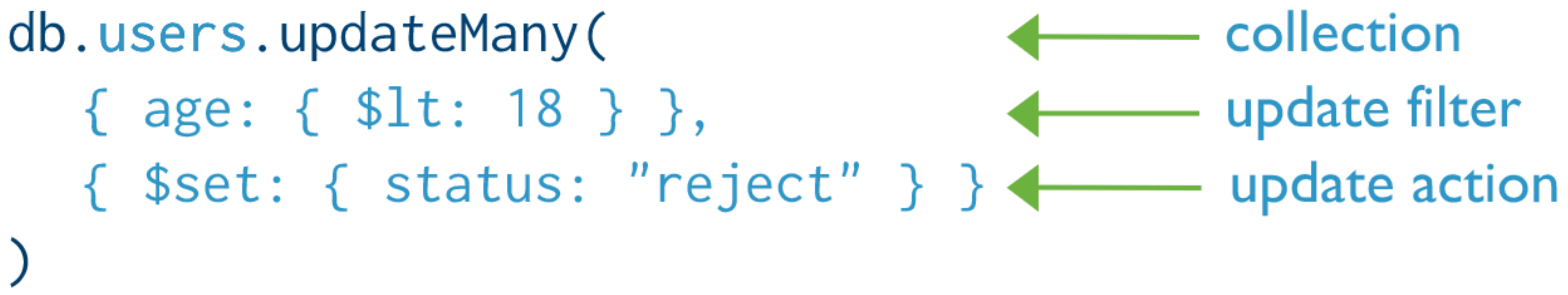


```
db.users.find(
   { age: { $gt: 18 } },
   { name: 1, address: 1 }
).limit(5)
```

Buscando um só documento

```
db.usuarios.findOne();
```

Primeiros Comandos



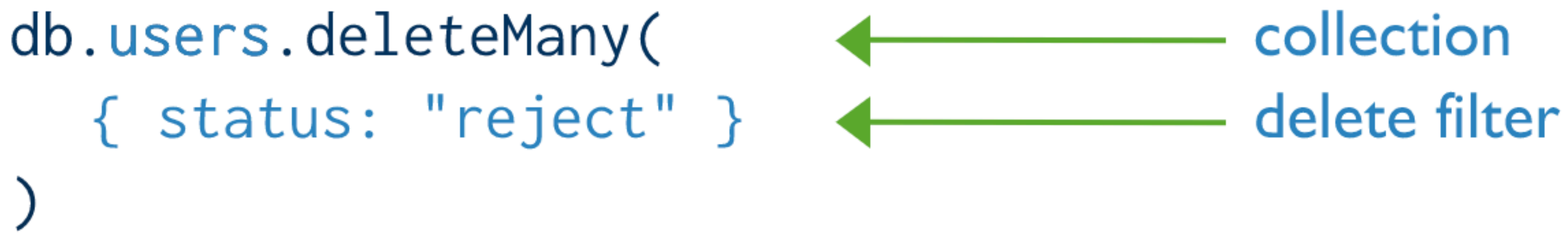
```
db.users.updateMany(
   { age: { $lt: 18 } },
   { $set: { status: "reject" } }
)
```

Atualizando o e-mail de newton

```
db.usuarios.updateMany({nome: "newton"}, {$set: {email:
"newton@outlook.com"}})
```

Não é possível atualizar o _id

Primeiros Comandos



```
db.users.deleteMany(
   { status: "reject" }
)
```

Deletando newton

```
db.usuarios.deleteMany({nome: "newton"})
```

Primeiros Comandos

Importando uma base de dados para teste…

```
exit; //sair do shell do mongo
docker exec -it bigdata-mongo /bin/bash

mongoimport --db test --collection restaurants --drop
--file /data/primer-dataset.json
```

https://github.com/gustavoleitao/mongo-dataset

Primeiros Comandos

Mostrando coleções

```
use test
show collections
```

Primeiros Comandos

Contanto número de elementos de uma coleção

```
db.restaurants.count();
```

Trabalhando com o exemplo…

Buscando por todos os restaurantes

To return all documents in a collection, call the **`find()`** method *without* a criteria document. For example, the following operation queries for all documents in the **`restaurants`** collection.

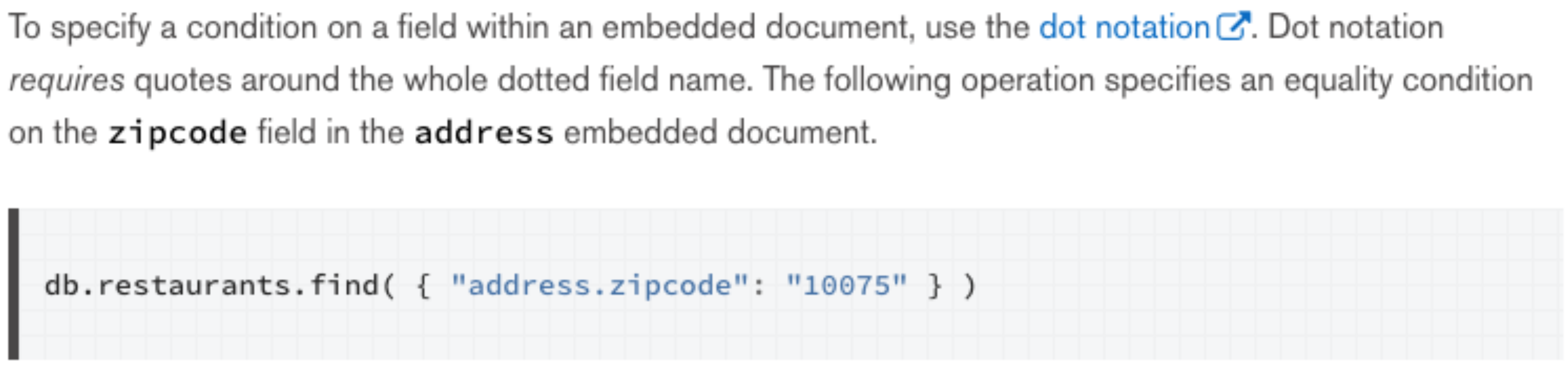
```
db.restaurants.find()
```

Adicionando condicionais

The following operation finds documents whose **`borough`** field equals **`"Manhattan"`**.

```
db.restaurants.find( { "borough": "Manhattan" } )
```

Buscando por um campo em um documento interno



To specify a condition on a field within an embedded document, use the dot notation. Dot notation *requires* quotes around the whole dotted field name. The following operation specifies an equality condition on the `zipcode` field in the `address` embedded document.

```
db.restaurants.find( { "address.zipcode": "10075" } )
```

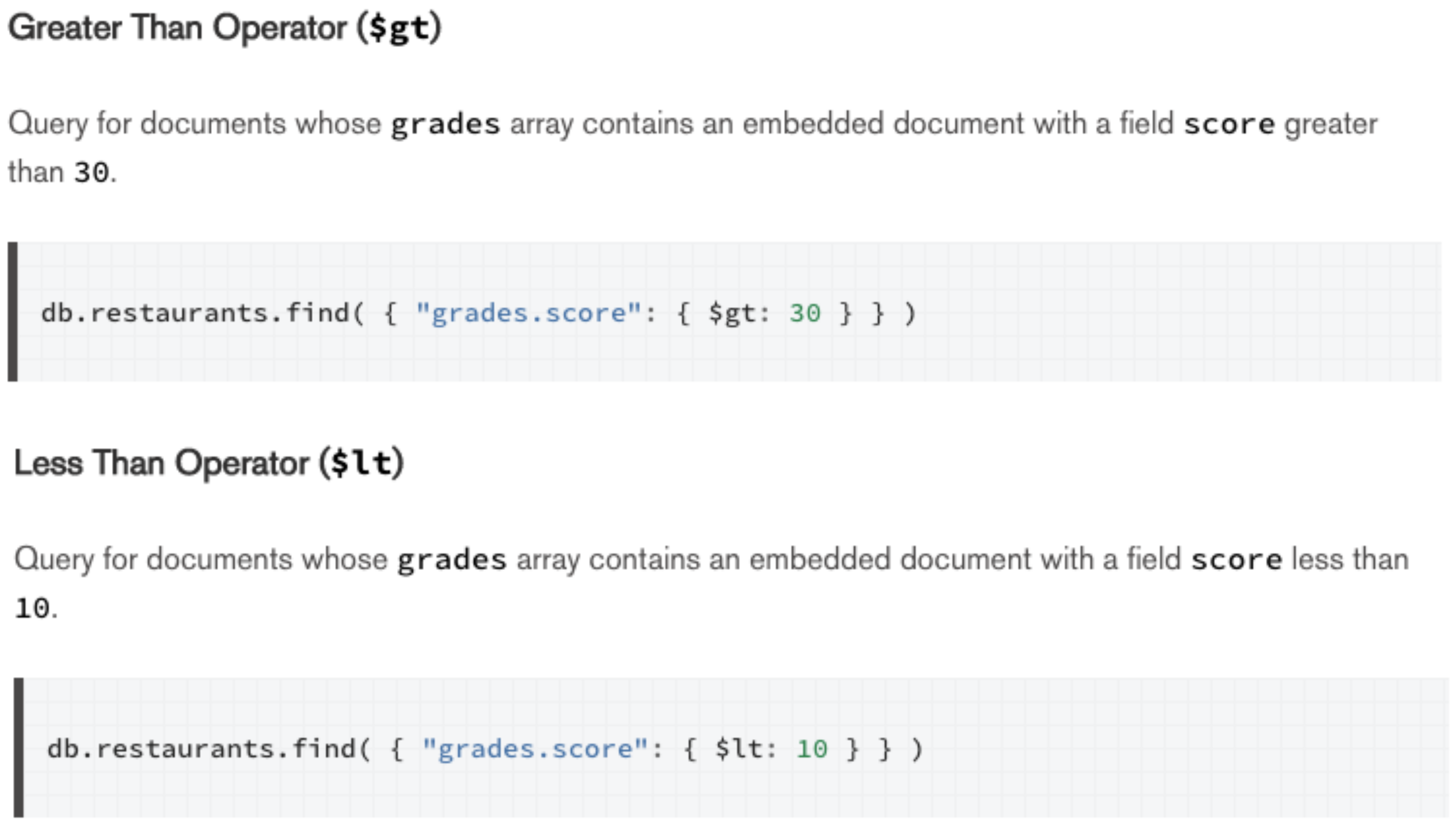
Contando quantidade de retorno

```
db.collection.find({}).count()
```

Limitando quantidade retornada

```
db.collection.find({}).limit(<number>)
```

Utilizando operadores matemáticos…



Greater Than Operator (`$gt`)

Query for documents whose `grades` array contains an embedded document with a field `score` greater than `30`.

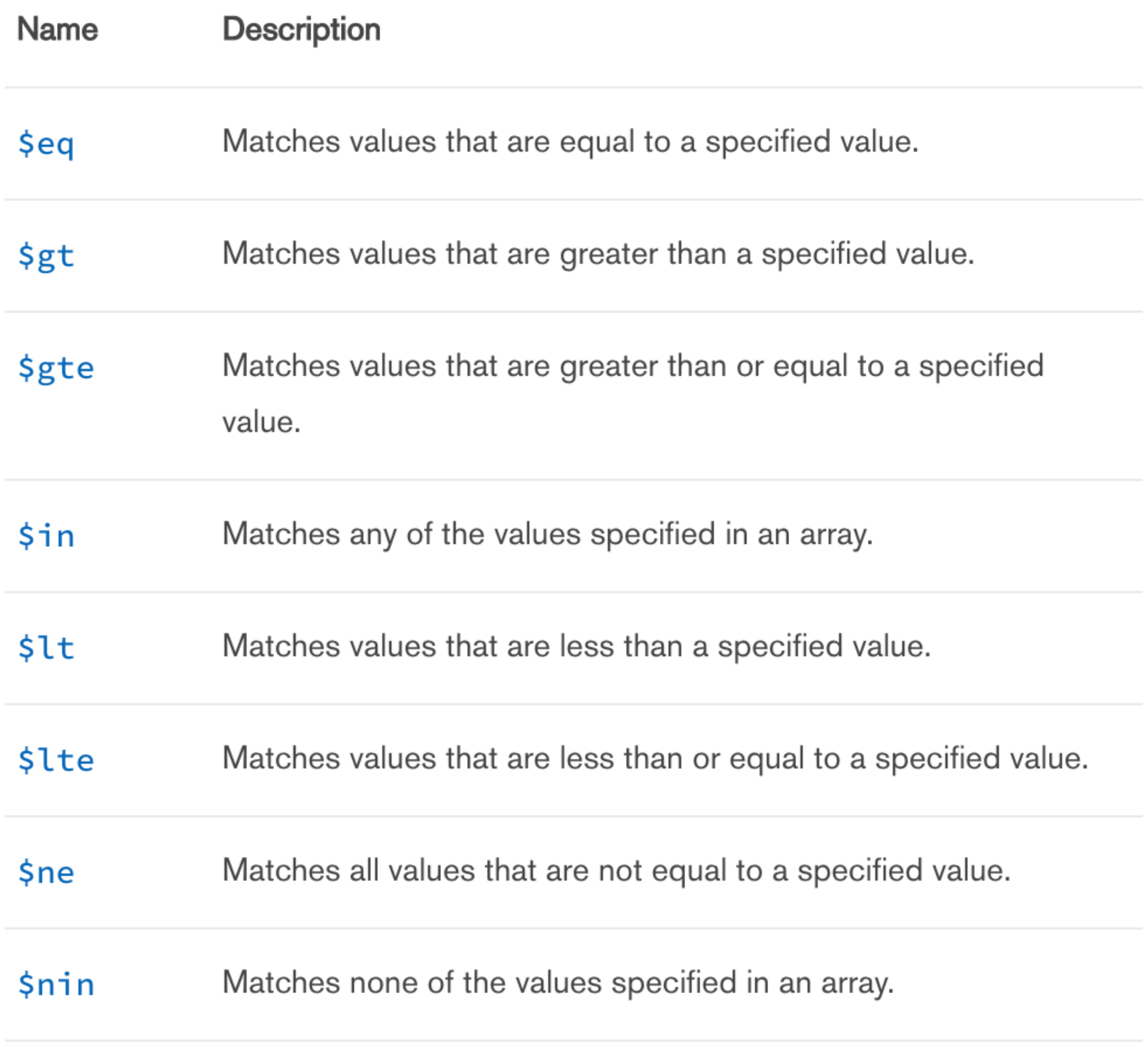
```
db.restaurants.find( { "grades.score": { $gt: 30 } } )
```

Less Than Operator (`$lt`)

Query for documents whose `grades` array contains an embedded document with a field `score` less than `10`.

```
db.restaurants.find( { "grades.score": { $lt: 10 } } )
```

Operadores



Name	Description
`$eq`	Matches values that are equal to a specified value.
`$gt`	Matches values that are greater than a specified value.
`$gte`	Matches values that are greater than or equal to a specified value.
`$in`	Matches any of the values specified in an array.
`$lt`	Matches values that are less than a specified value.
`$lte`	Matches values that are less than or equal to a specified value.
`$ne`	Matches all values that are not equal to a specified value.
`$nin`	Matches none of the values specified in an array.

Exercício

- Quantos são padaria? (cuisine: Bakery)
- Há quantos restaurantes na rua "Morris Park Ave’’?

Exercício

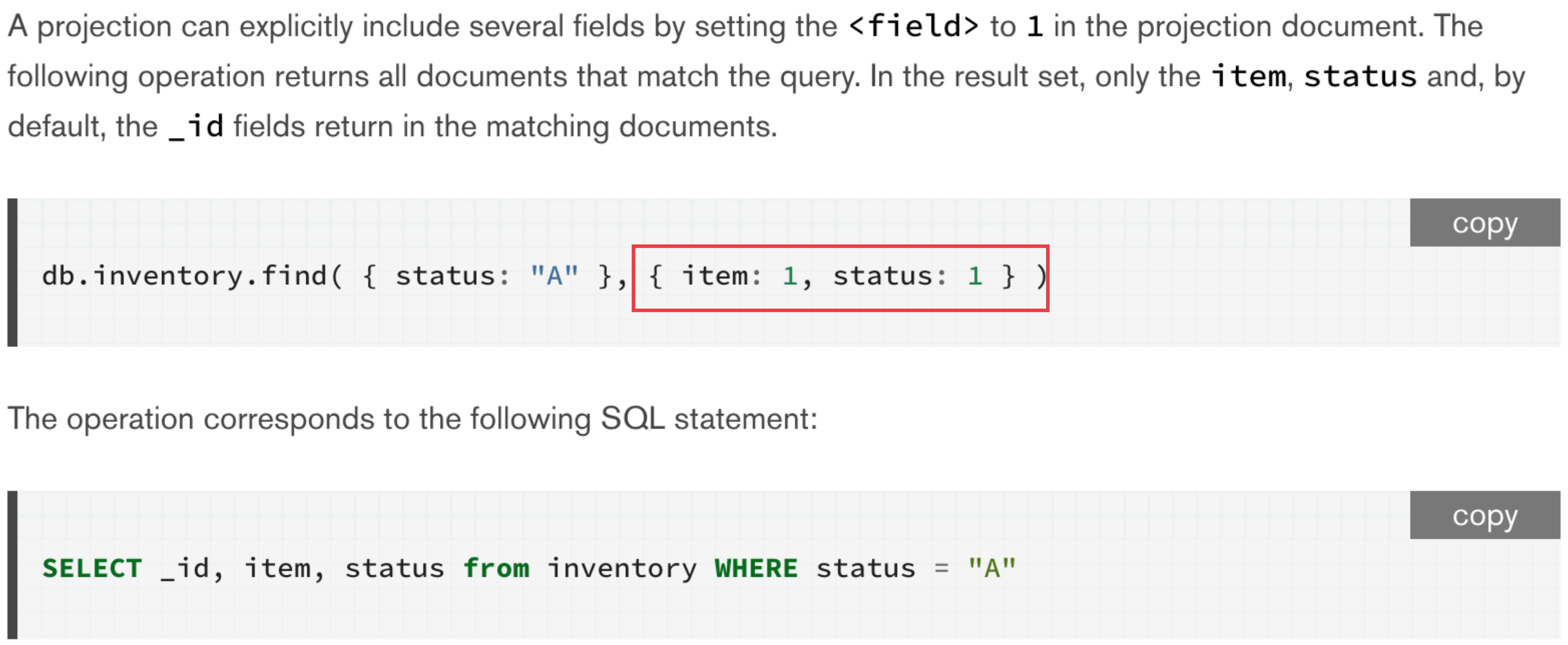
- Quantos são padaria? (cuisine: Bakery)

```
db.restaurants.find({"cuisine": "Bakery"}).count()
```

Há quantos restaurantes na rua "Morris Park Ave”?

```
db.restaurants.find({"address.street": "Morris Park
Ave"}).count()
```

Retornando apenas alguns campos



A projection can explicitly include several fields by setting the `<field>` to `1` in the projection document. The following operation returns all documents that match the query. In the result set, only the `item`, `status` and, by default, the `_id` fields return in the matching documents.

```
db.inventory.find( { status: "A" }, { item: 1, status: 1 } )
```

The operation corresponds to the following SQL statement:

```
SELECT _id, item, status from inventory WHERE status = "A"
```

Exercício

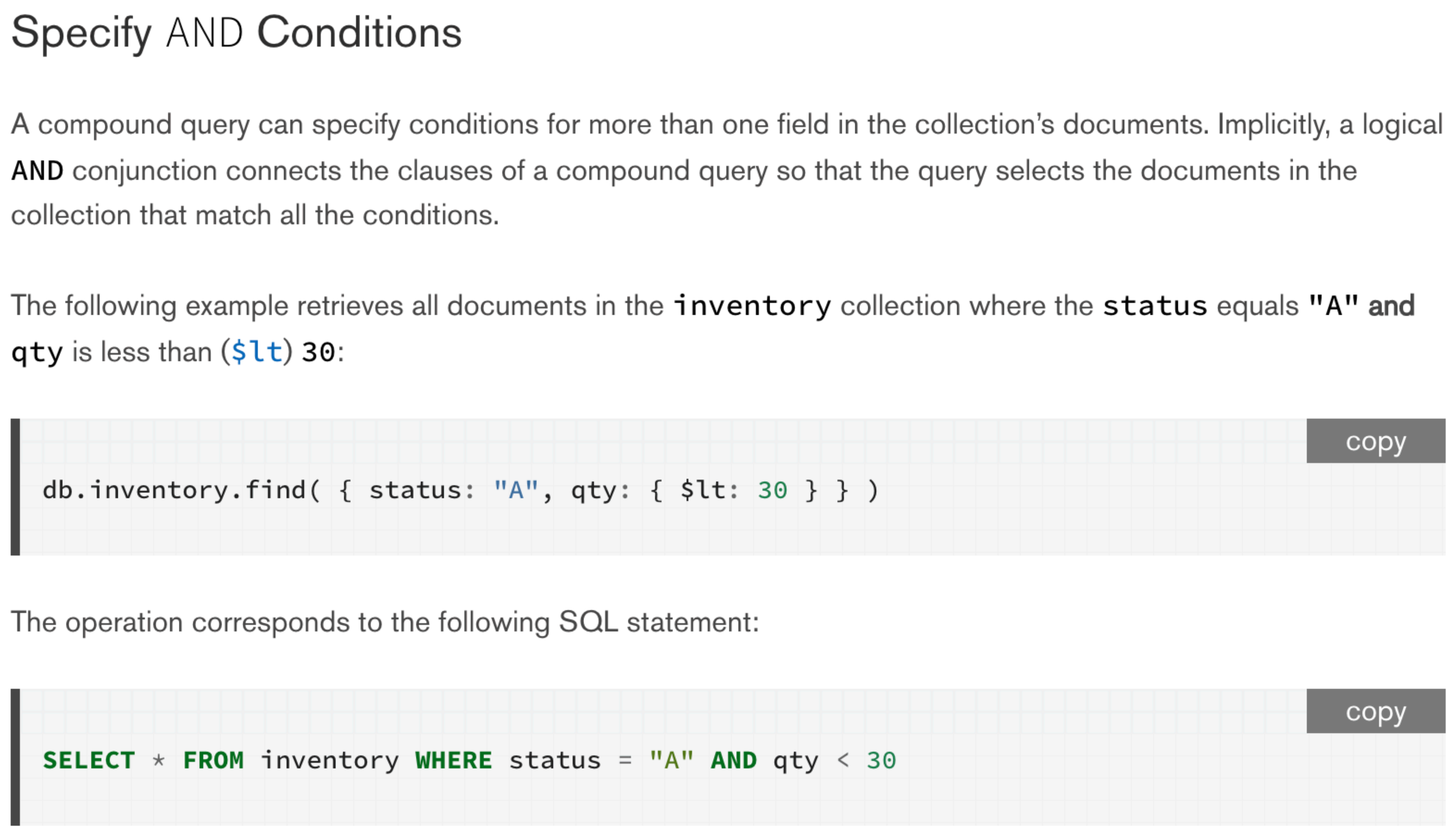
- Selecione apenas o nome dos restaurantes italianos. (cuisine: Italian)

Exercício

- Selecione apenas o nome dos restaurantes italianos. (cuisine: Italian)

```
db.restaurants
.find({“cuisine": "Italian"}, {"name": 1, "_id":
0})
```

Utilizando operadores lógicos…



Specify AND Conditions

A compound query can specify conditions for more than one field in the collection's documents. Implicitly, a logical `AND` conjunction connects the clauses of a compound query so that the query selects the documents in the collection that match all the conditions.

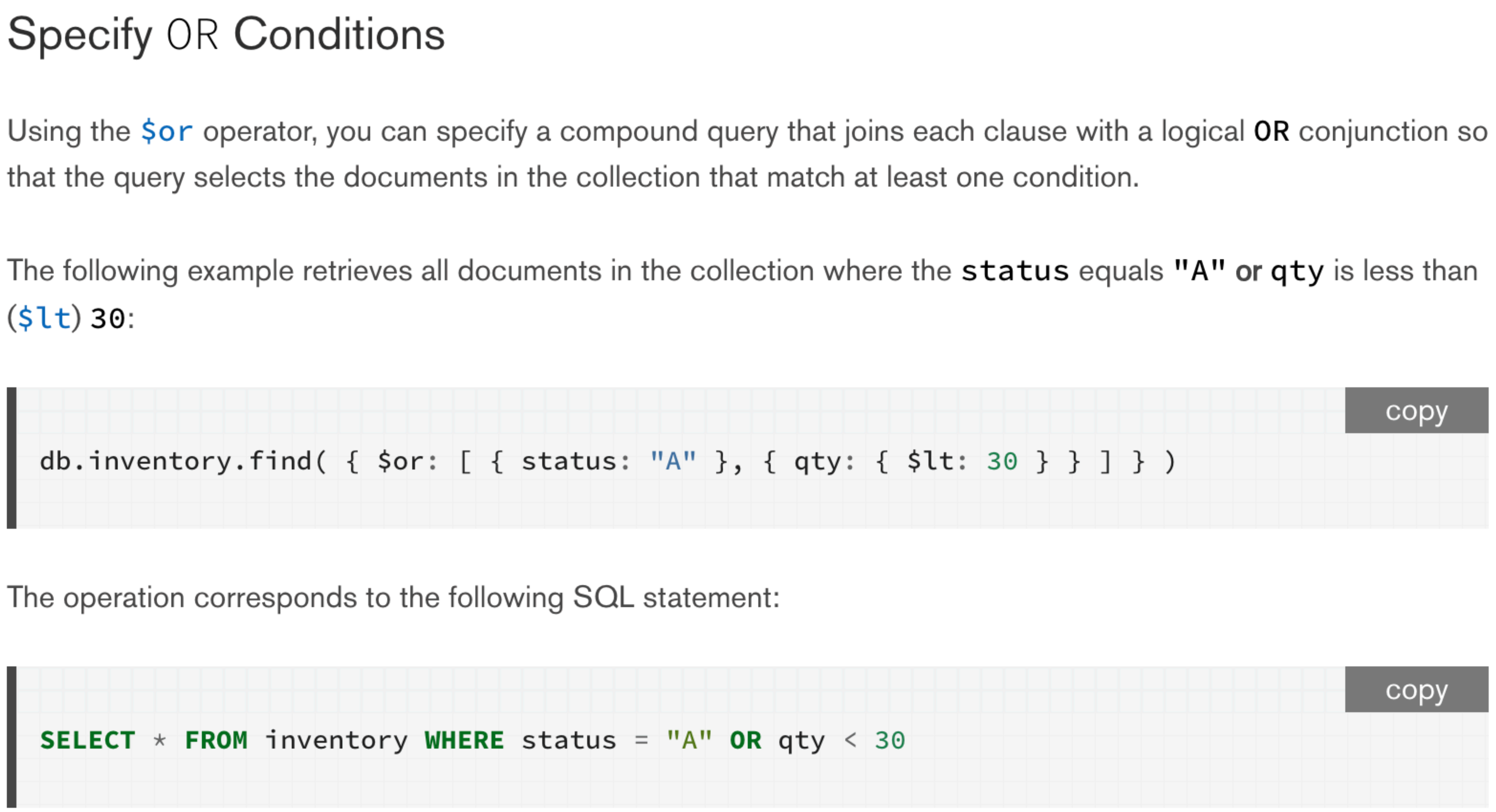
The following example retrieves all documents in the `inventory` collection where the `status` equals `"A"` **and** `qty` is less than (`$lt`) `30`:

```
db.inventory.find( { status: "A", qty: { $lt: 30 } } )
```

The operation corresponds to the following SQL statement:

```
SELECT * FROM inventory WHERE status = "A" AND qty < 30
```

Utilizando operadores lógicos…



Specify `OR` Conditions

Using the `$or` operator, you can specify a compound query that joins each clause with a logical `OR` conjunction so that the query selects the documents in the collection that match at least one condition.

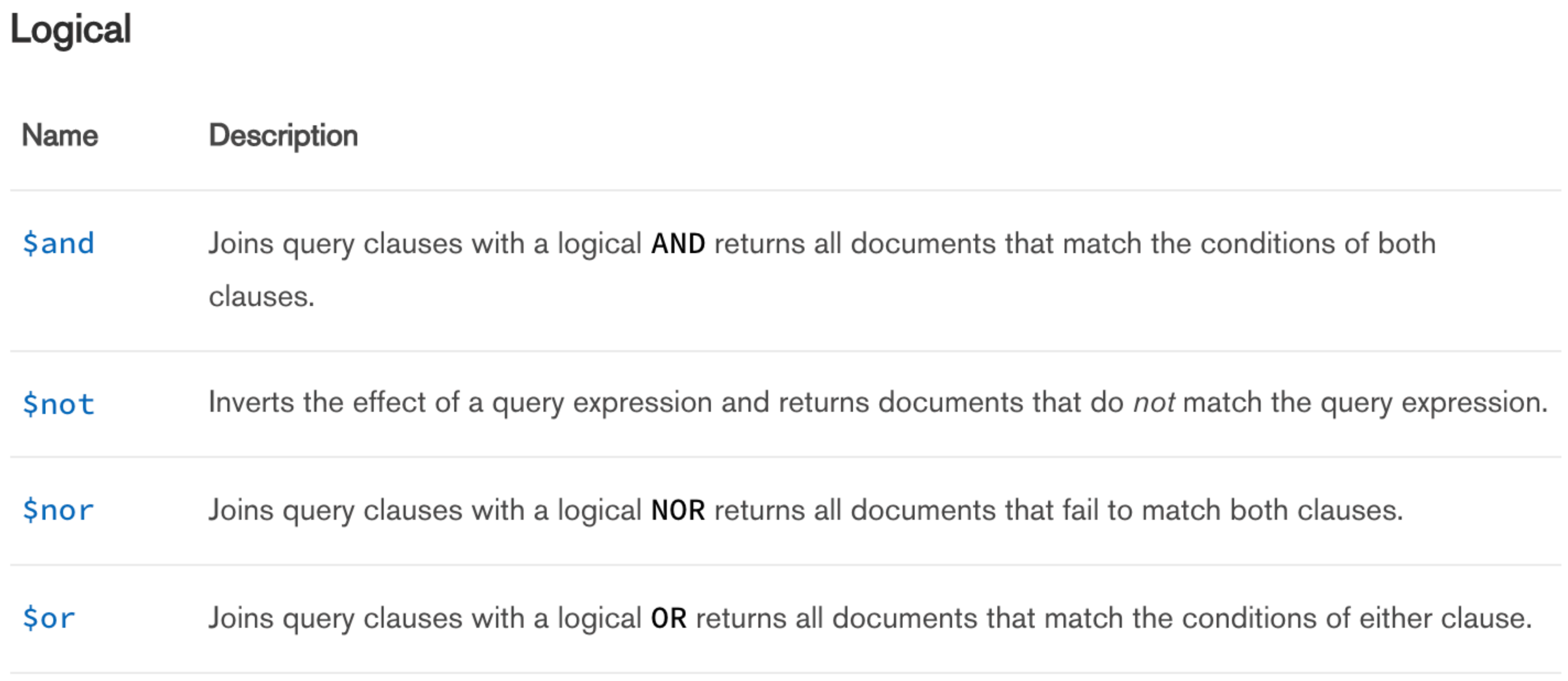
The following example retrieves all documents in the collection where the `status` equals `"A"` **or** `qty` is less than (`$lt`) `30`:

```
db.inventory.find( { $or: [ { status: "A" }, { qty: { $lt: 30 } } ] } )
```

The operation corresponds to the following SQL statement:

```
SELECT * FROM inventory WHERE status = "A" OR qty < 30
```

Utilizando operadores lógicos…



Logical

Name	Description
`$and`	Joins query clauses with a logical **AND** returns all documents that match the conditions of both clauses.
`$not`	Inverts the effect of a query expression and returns documents that do *not* match the query expression.
`$nor`	Joins query clauses with a logical **NOR** returns all documents that fail to match both clauses.
`$or`	Joins query clauses with a logical **OR** returns all documents that match the conditions of either clause.

Exercício

- Quais restaurantes são de cozinha italiana e possui *zipcode* 10075?
- Quais os nomes dos restaurantes que são de cozinha italiana ou irlandesa? (Italian and Irish)
- Quantos restaurantes possuem na base de dados sem nome?

Exercício

- Quais restaurantes são de cozinha italiana e possui *zipcode* 10075?

```
db.restaurants.find({cuisine:
"Italian","address.zipcode": “10075"});
```

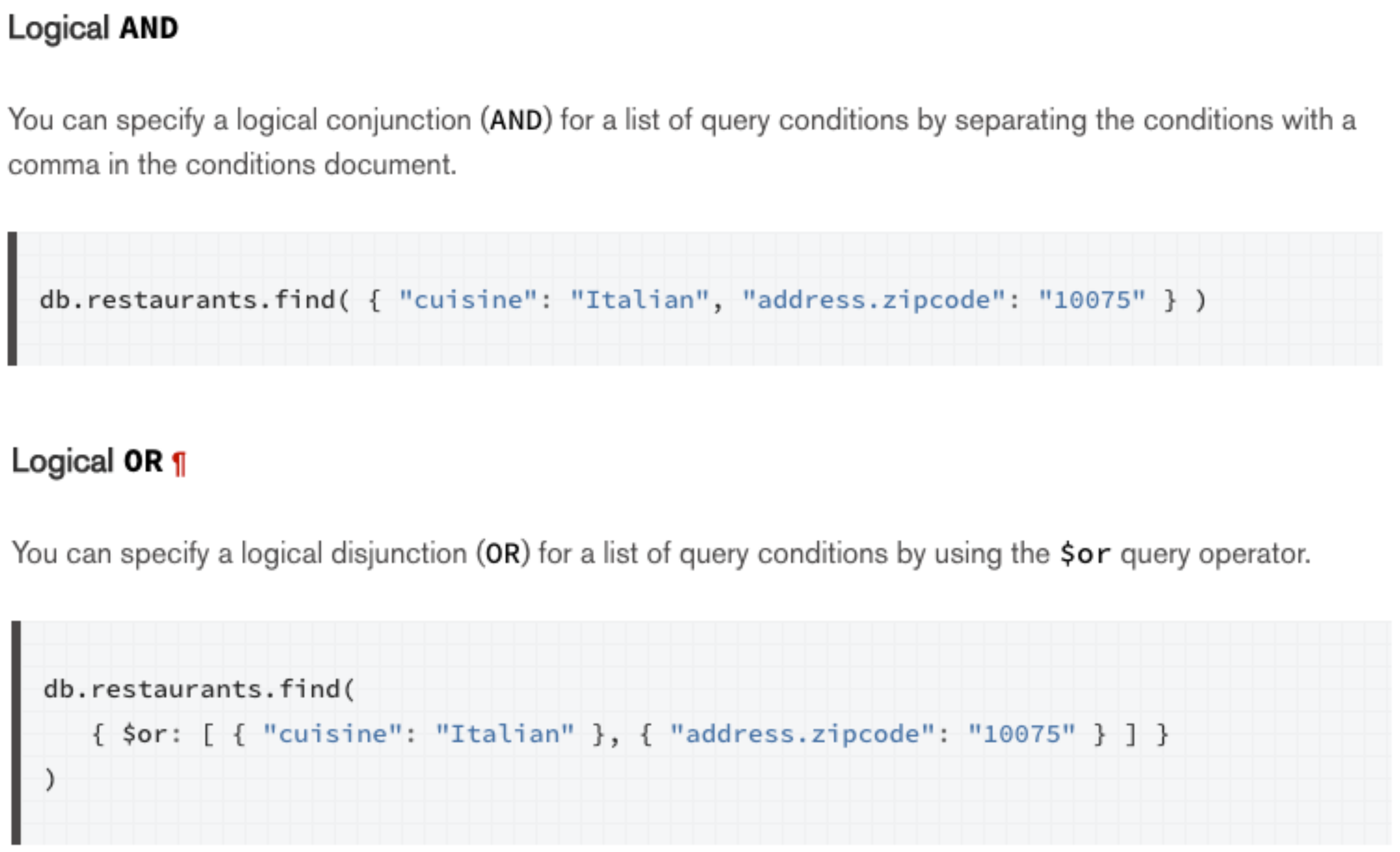
- Quais os nomes dos restaurantes que são de cozinha italiana ou irlandesa? (Italian and Irish)

```
db.restaurants.find({$or: [{"cuisine": "Italian"},
{"cuisine": "Irish"}]})
```

- Quantos restaurantes possuem na base de dados sem nome?

```
db.restaurants
.find({"name": “”})
.count()
```

Utilizando operadores lógicos…



Logical **AND**

You can specify a logical conjunction (**AND**) for a list of query conditions by separating the conditions with a comma in the conditions document.

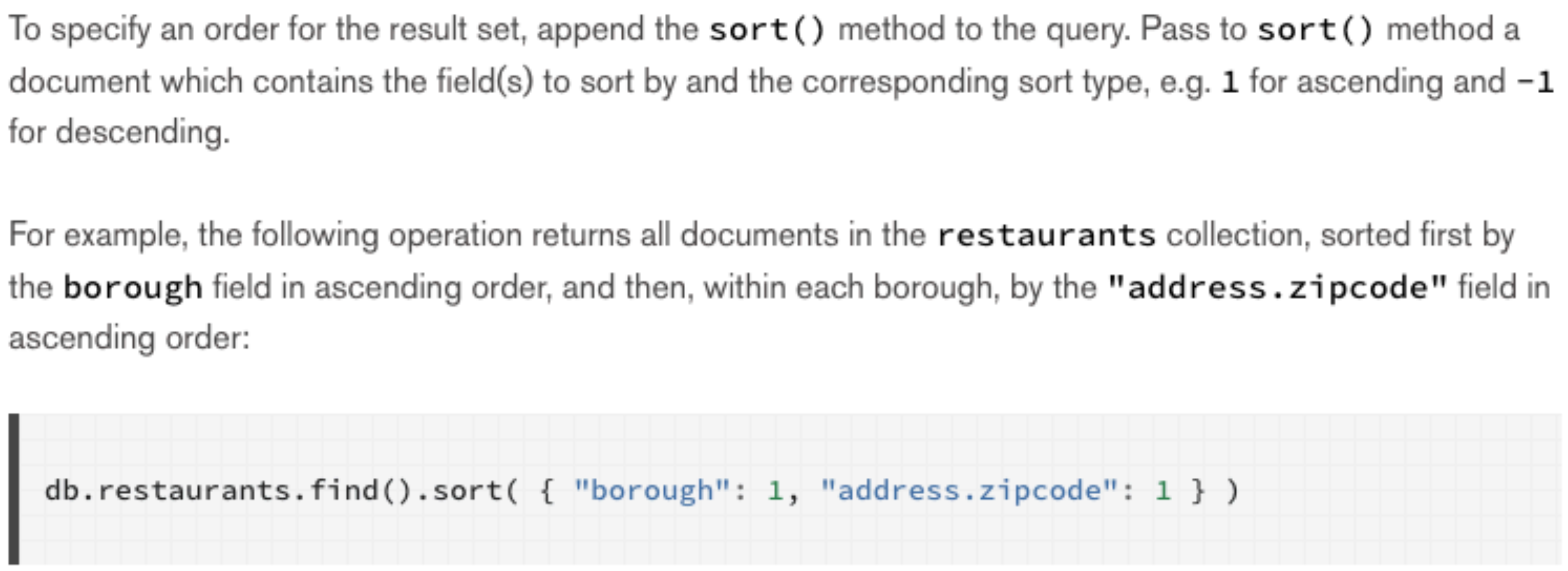
```
db.restaurants.find( { "cuisine": "Italian", "address.zipcode": "10075" } )
```

Logical **OR** ¶

You can specify a logical disjunction (**OR**) for a list of query conditions by using the `$or` query operator.

```
db.restaurants.find(
   { $or: [ { "cuisine": "Italian" }, { "address.zipcode": "10075" } ] }
)
```

Ordenando os resultados



To specify an order for the result set, append the **`sort()`** method to the query. Pass to **`sort()`** method a document which contains the field(s) to sort by and the corresponding sort type, e.g. **`1`** for ascending and **`-1`** for descending.

For example, the following operation returns all documents in the **`restaurants`** collection, sorted first by the **`borough`** field in ascending order, and then, within each borough, by the **`"address.zipcode"`** field in ascending order:

```
db.restaurants.find().sort( { "borough": 1, "address.zipcode": 1 } )
```

Exercício

- Selecione todos os restaurantes com nome definidos (diferente de vazio) ordenado por nome.

Exercício

- Selecione todos os restaurantes com nome definidos (diferente de vazio) ordenado por nome.

```
db.restaurants
.find({"name": {$ne: "" }})
.sort({name: 1})
```

Paginando o resultado

Limitando quantidade retornada

```
db.collection.find().sort().limit(<number>)
```

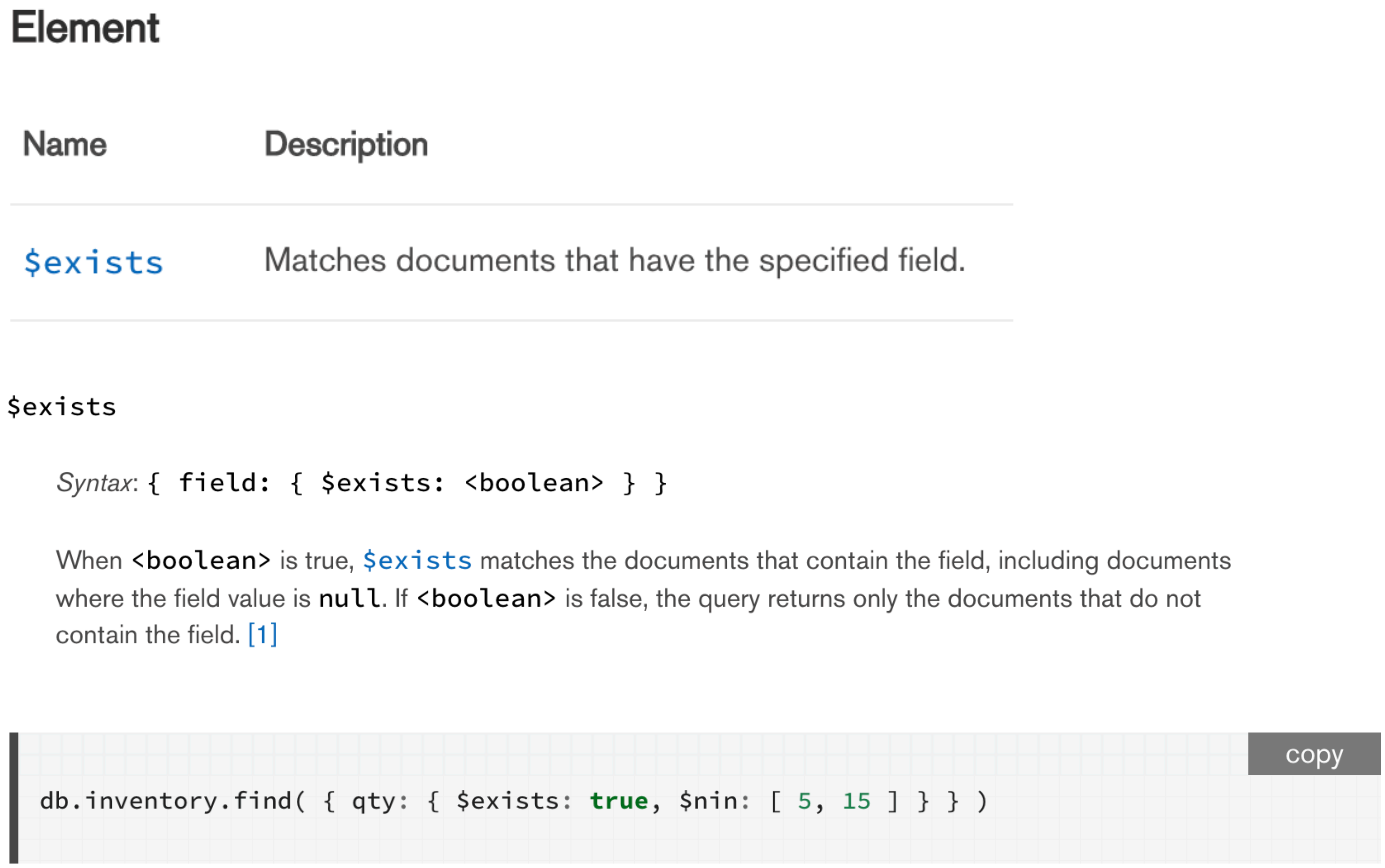
Pulando documentos

```
db.collection.find().sort().skip(<number>)
```

Paginação:

```
db.collection.find().sort({})
.skip(<pagina-1 * qnt-por-pagina>)
.limit(<qnt-por-pagina>)
```

Verificando existência de um campo



Element

Name	Description
`$exists`	Matches documents that have the specified field.

`$exists`

Syntax: `{ field: { $exists: <boolean> } }`

When `<boolean>` is true, `$exists` matches the documents that contain the field, including documents where the field value is `null`. If `<boolean>` is false, the query returns only the documents that do not contain the field. [1]

```
db.inventory.find( { qty: { $exists: true, $nin: [ 5, 15 ] } } )
```

Exercício

- Quantos restaurantes possuem coordenadas definidas?

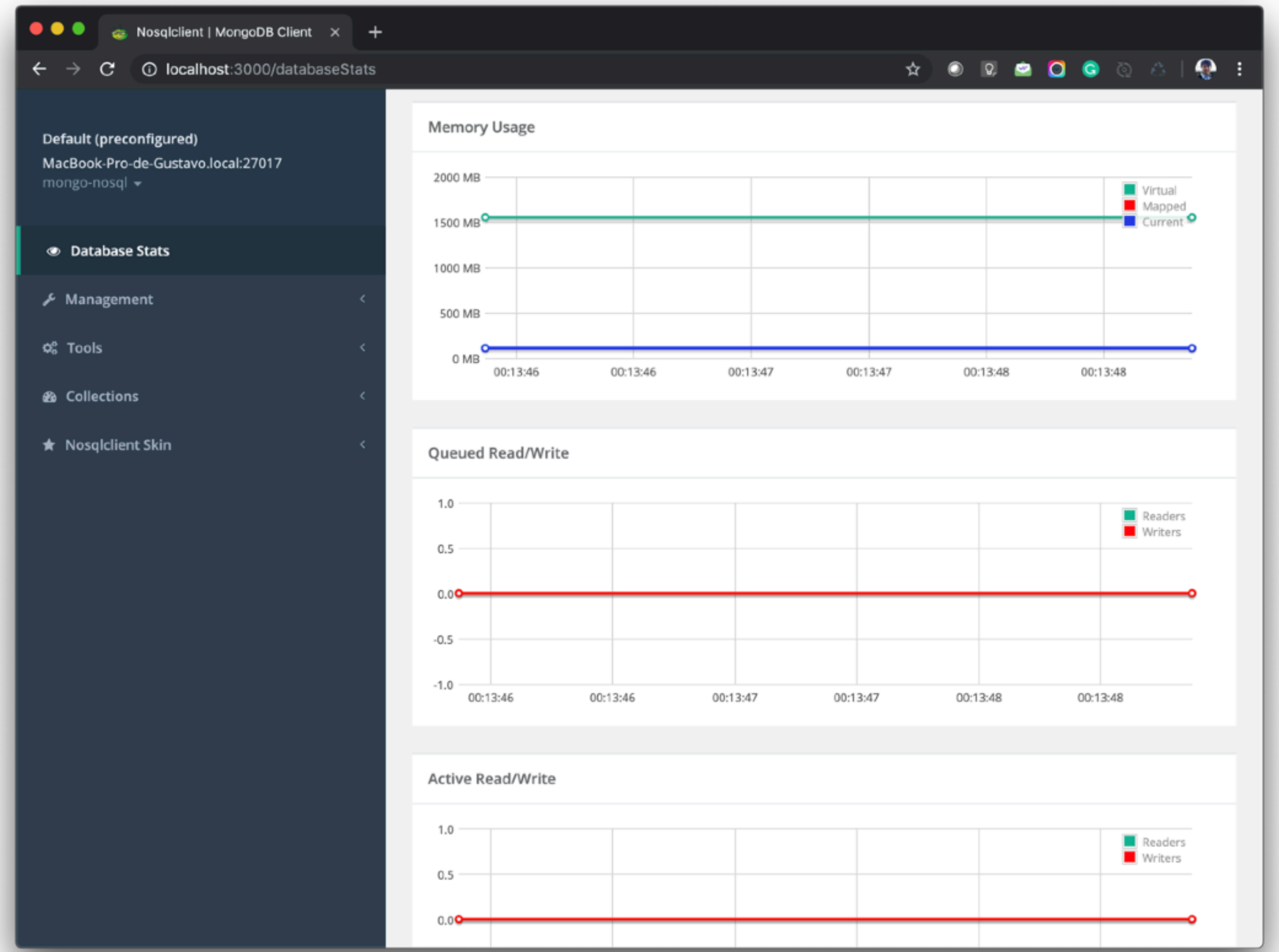
Exercício

- Quantos restaurantes possuem coordenadas definidas?

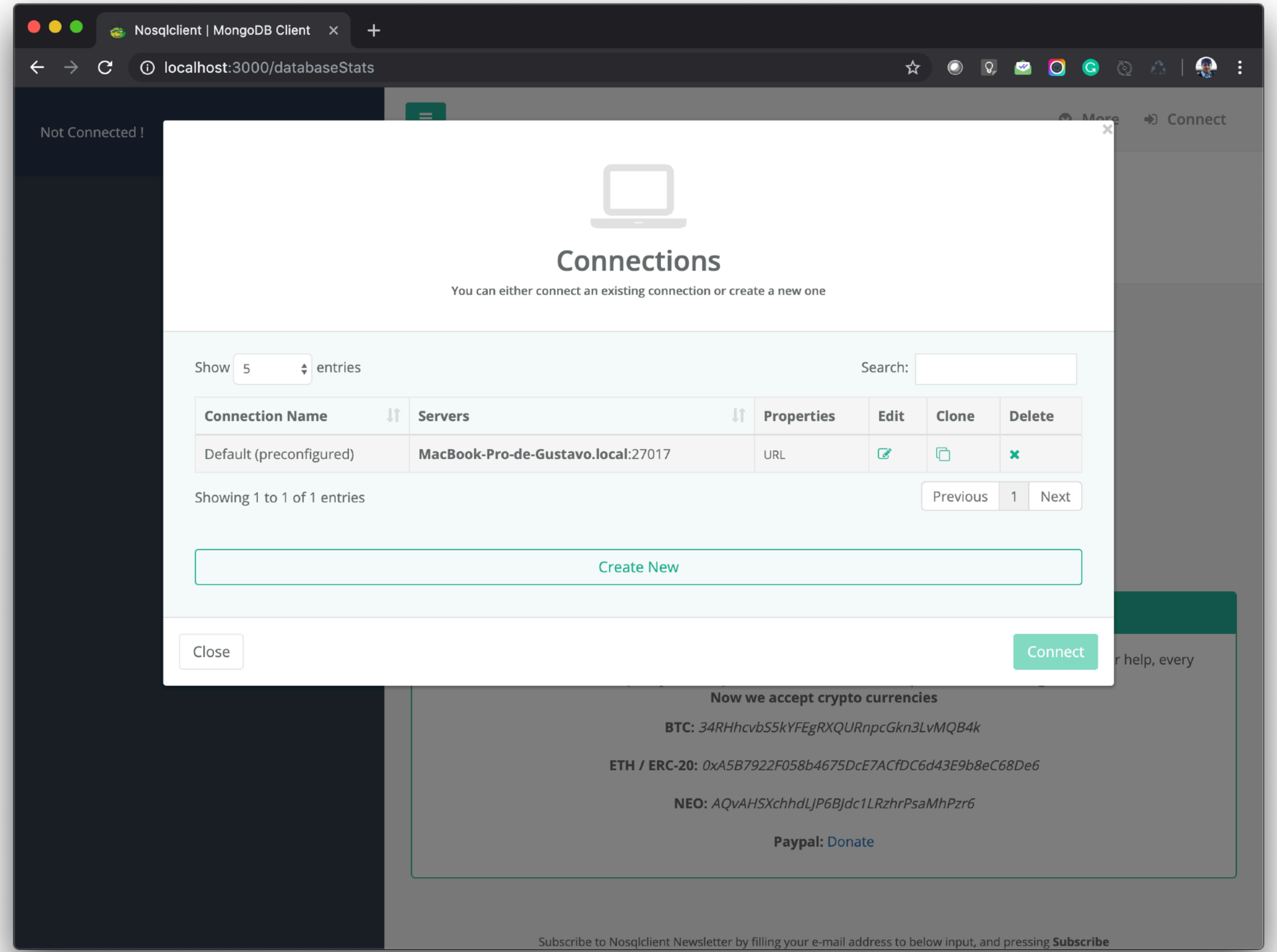
```
db.restaurants
.find({"address.coord": {$exists: true}})
.count()
```

Cliente grafico

Mongo Cliente



```
docker run -d -p 3000:3000 mongoclient/mongoclient
```



Connections

You can either connect an existing connection or create a new one

Show 5 entries

Search:

Connection Name	Servers	Properties	Edit	Clone	Delete
Default (preconfigured)	**MacBook-Pro-de-Gustavo.local**:27017	URL			✖

Showing 1 to 1 of 1 entries

r help, every

Now we accept crypto currencies

BTC: *34RHhcvbS5kYFEgRXQURnpcGkn3LvMQB4k*

ETH / ERC-20: *0xA5B7922F058b4675DcE7ACfDC6d43E9b8eC68De6*

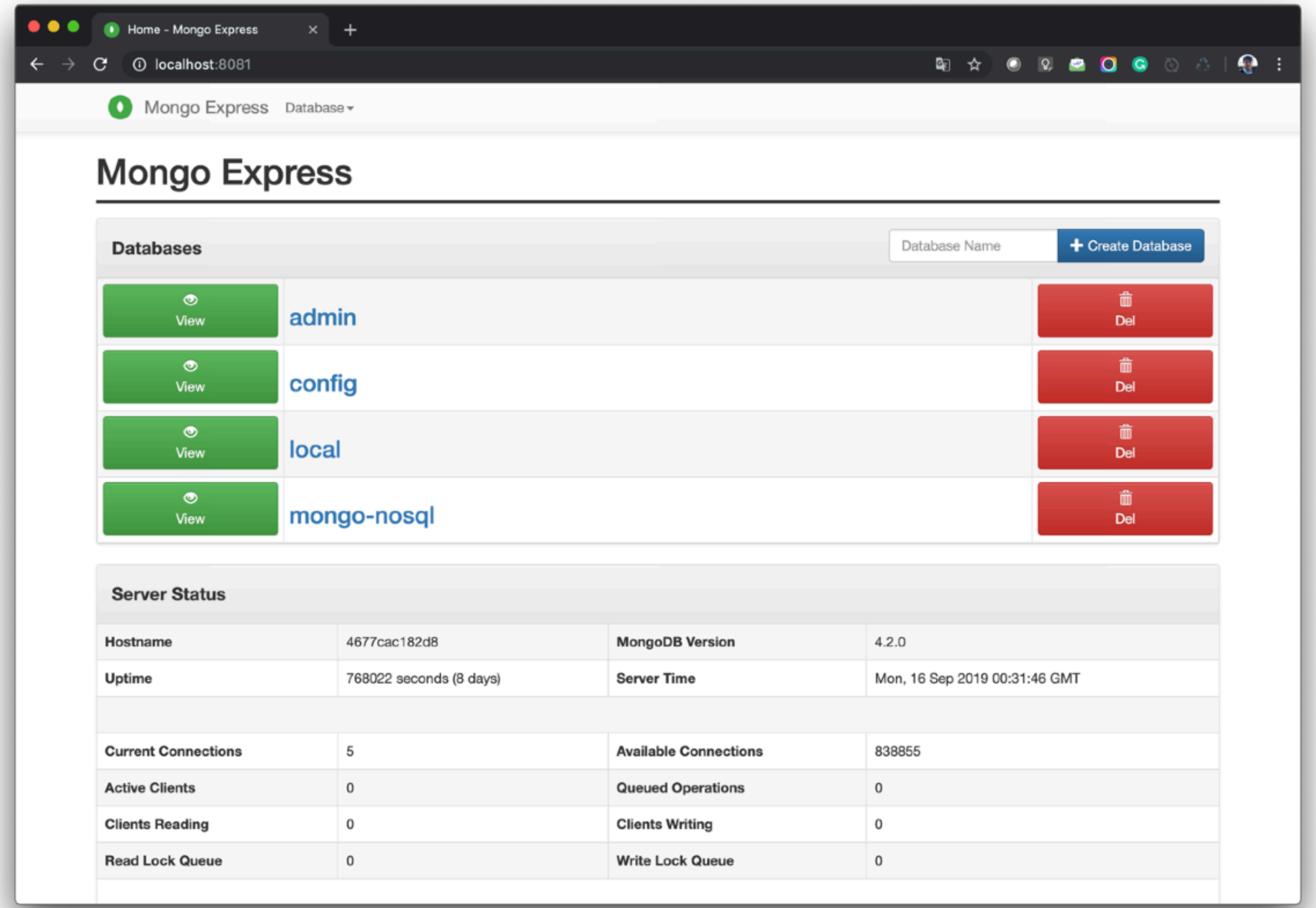
NEO: *AQvAHSXchhdLJP6BJdc1LRzhrPsaMhPzr6*

Paypal: Donate

Subscribe to Nosqlclient Newsletter by filling your e-mail address to below input, and pressing **Subscribe**

Cliente grafico

Mongo Express



Mongo Express

Databases

- admin
- config
- local
- mongo-nosql

Server Status

Hostname	4677cac182d8	MongoDB Version	4.2.0
Uptime	768022 seconds (8 days)	Server Time	Mon, 16 Sep 2019 00:31:46 GMT
Current Connections	5	Available Connections	838855
Active Clients	0	Queued Operations	0
Clients Reading	0	Clients Writing	0
Read Lock Queue	0	Write Lock Queue	0

```
docker run -e ME_CONFIG_MONGODB_SERVER=$hostname
-p 8081:8081  -d mongo-express
```

Banco de dados NoSQL - MongoDB (Parte 1)

Prof. Gustavo Leitão