

**Mestrado em Engenharia Informática**

**Unidade Curricular Complementar**

**Base de Dados NoSQL**

Ficha de Exercícios 04

4º Ano, 1º Semestre

Ano letivo 2020/2021

Diogo Alexandre Rodrigues Lopes

PG42823

Joel Costa Carvalho

PG42837

Ana Margarida da Rocha Ferreira

PG44412

Bruno Xavier Brás dos Santos

PG44414

**Conteúdo**

[Tarefa 1. Customers](#_heading=h.30j0zll) **3**

[Tarefa 2. restaurants.json](#_heading=h.2zf81hd9ntx1) **6**

# **Tarefa 1.Customers**

1. **Listar todas as bases de dados após a instalação do container com a imagem do MongoDB.**

$ show dbs

1. **Criar uma base de dados denominada “customers”.**

$ use customers

A base de dados só é criada após a inserção do primeiro documento.

1. **Verificar a criação da base de dados.**

Confirmação da criação: *switched to db customers*

$ show collections - vazio porque ainda não contém coleções

1. **Criar uma coleção denominada “customers”.**

$ db.createCollection("customers")

1. **Validar a criação da coleção.**

Confirmação da criação: *{ "ok" : 1 }*

$ show collections

1. **Criar um cliente com os seguintes características:** 
   1. **first\_name: “John”, last\_name:”Doe”, age: 30**

$ db.customers.insert({first\_name: "John", last\_name: "Doe", age: 30})

1. **Introduzir 2 clientes na coleção criada com as seguintes características:**
   1. **first\_name: “Steven”, last\_name:”Williams”, gender: “male”**
   2. **first\_name: “Mary”, last\_name: “Troy”, age: 19**

$ var alinea7 = [{first\_name: "Steven", last\_name: "Williams", gender: "male"}, {first\_name: "Mary", last\_name: "Troy", age: 19}]

$ db.customers.insert(alinea7)

1. **Introduzir mais um cliente com as seguintes características:**
   1. **first\_name: “Ric”, last\_name: “Foe”, address: {street: “4 main st”, city: “Boston”}**

$ db.customers.insert({first\_name: "Ric", last\_name: "Foe", address: {street: "4 main st", city: "Boston"}})

1. **Criar um cliente com as seguintes características:**
   1. **first\_name: “Ana”, last\_name: “Durant”, degree:[“phD”,”Msc”], address: {street: “4 Square Garden”, city: “New York”}, age: 32**

$ db.customers.insert({first\_name: "Ana", last\_name: "Durant", degree:["phD","Msc"], address:{street: "4 Square Garden", city: "New York"}, age: 32})

1. **Criar um cliente com as seguintes características:**
   1. **first\_name: “Natalia”, last\_name:”Will”, age: 44, gender: “female”**

$ db.customers.insert({first\_name: "Natalia", last\_name: "Will", age: 44, gender: "female"})

1. **Listar todos os clientes.**

$ db.customers.find()

1. **Listar todos os clientes usando a função pretty().**

$ db.customers.find().pretty()

1. **Efetuar uma atualização ao cliente ‘Ric’, colocar idade 45.**

$ db.customers.update({first\_name: "Ric"}, {$set: {age: 45}})

1. **Encontrar todos os clientes que tenham ‘Will’ no último nome.**

$ db.customers.find({"last\_name": /Will/})

1. **Efetuar uma atualização ao cliente ‘Steven’, colocar idade 35.**

$ db.customers.update({first\_name: "Steven"}, {$set: {age: 35}})

1. **Verificar se a idade da cliente ‘Ana’ é superior a 30 e se sim aumentar a idade em 10 anos.**

$ db.customers.find({first\_name: "Ana", age:{$gt:30}}).pretty()

$ db.customers.update({first\_name: "Ana"}, {$set: {age: 42}})

1. **O cliente ‘Ric’ quer que a sua idade seja removida da base de dados.**

$ db.customers.update({first\_name: "Ric"}, {$unset :{age: ""}})

1. **Procurar um cliente com o primeiro nome: "Jimmy" e atualizar, ou criar, caso não exista, com as seguintes características.:**
   1. **first\_name: "Jimmy", last\_name: "Connors", age: 25, gender: male**

$ db.customers.update({first\_name: "Jimmy"}, {$set: {first\_name: "Jimmy", last\_name: "Connors", age: 25, gender: "male"}}, {upsert:true})

1. **Procurar todos os clientes com idade superior ou igual a 25.**

$ db.customers.find({age:{$gte:25}})

1. **Procurar todos os clientes sexo masculino.**

$ db.customers.find({gender: "male"})

1. **Apagar o cliente cujo primeiro nome é "Mary".**

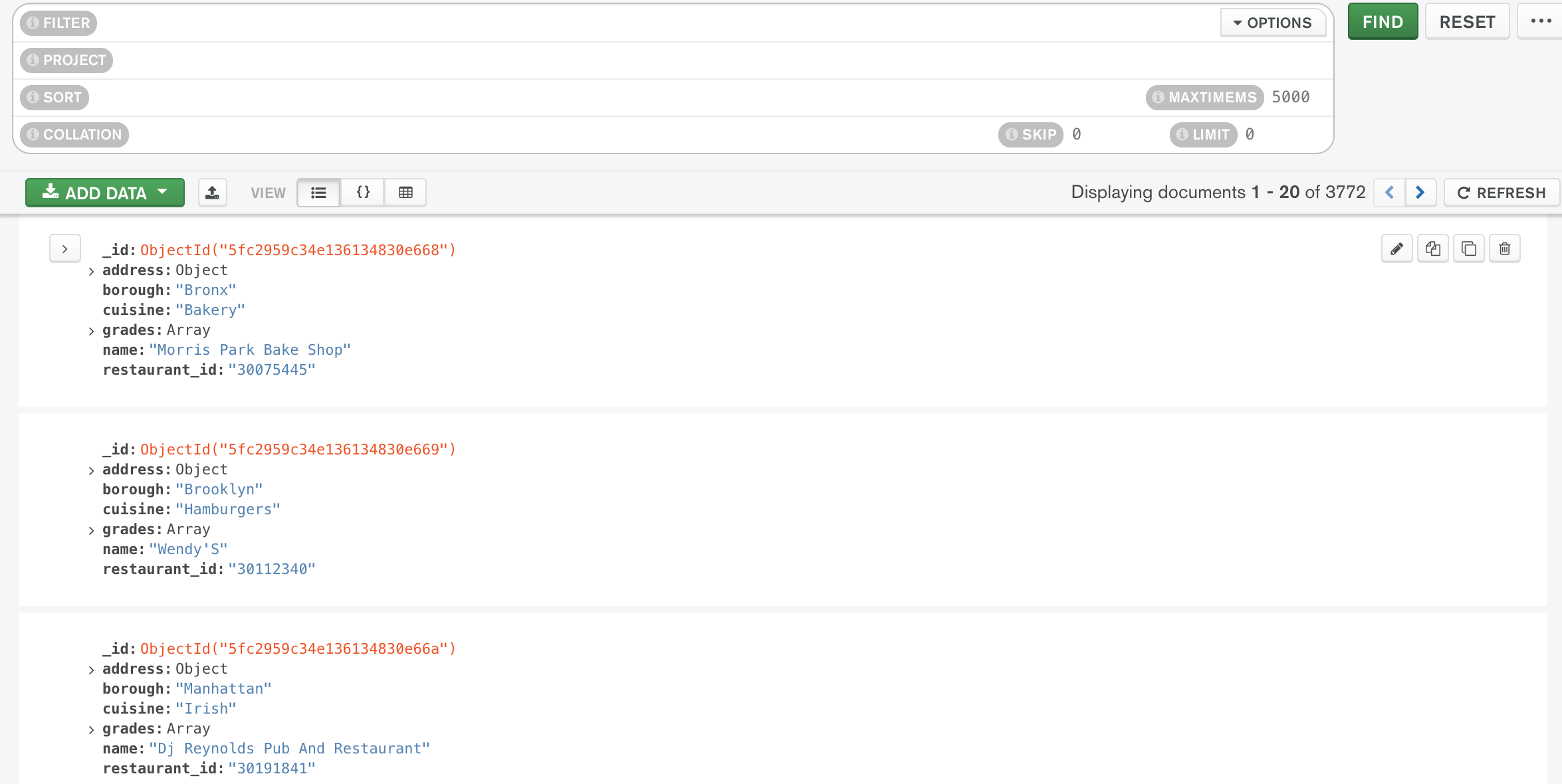
$ db.customers.remove({first\_name: "Mary"})

1. **Encontrar os clientes com o nome "Ana" ou "Ric".**

$ db.customers.find({$or : [{ first\_name: "Ana"}, {first\_name: "Ric"}]}).pretty()

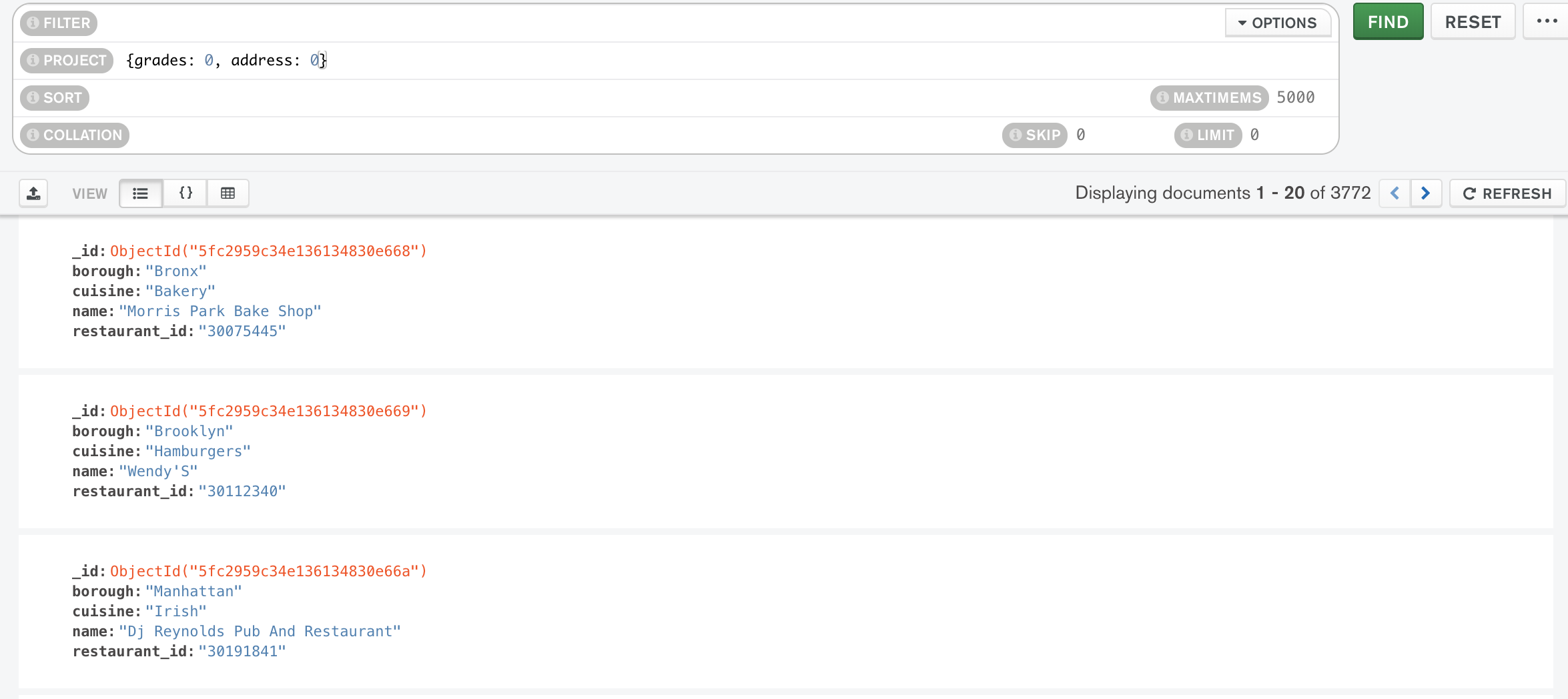
# **Tarefa 2.restaurants.json**

1. **Liste todos os documentos na coleção restaurants.**

****

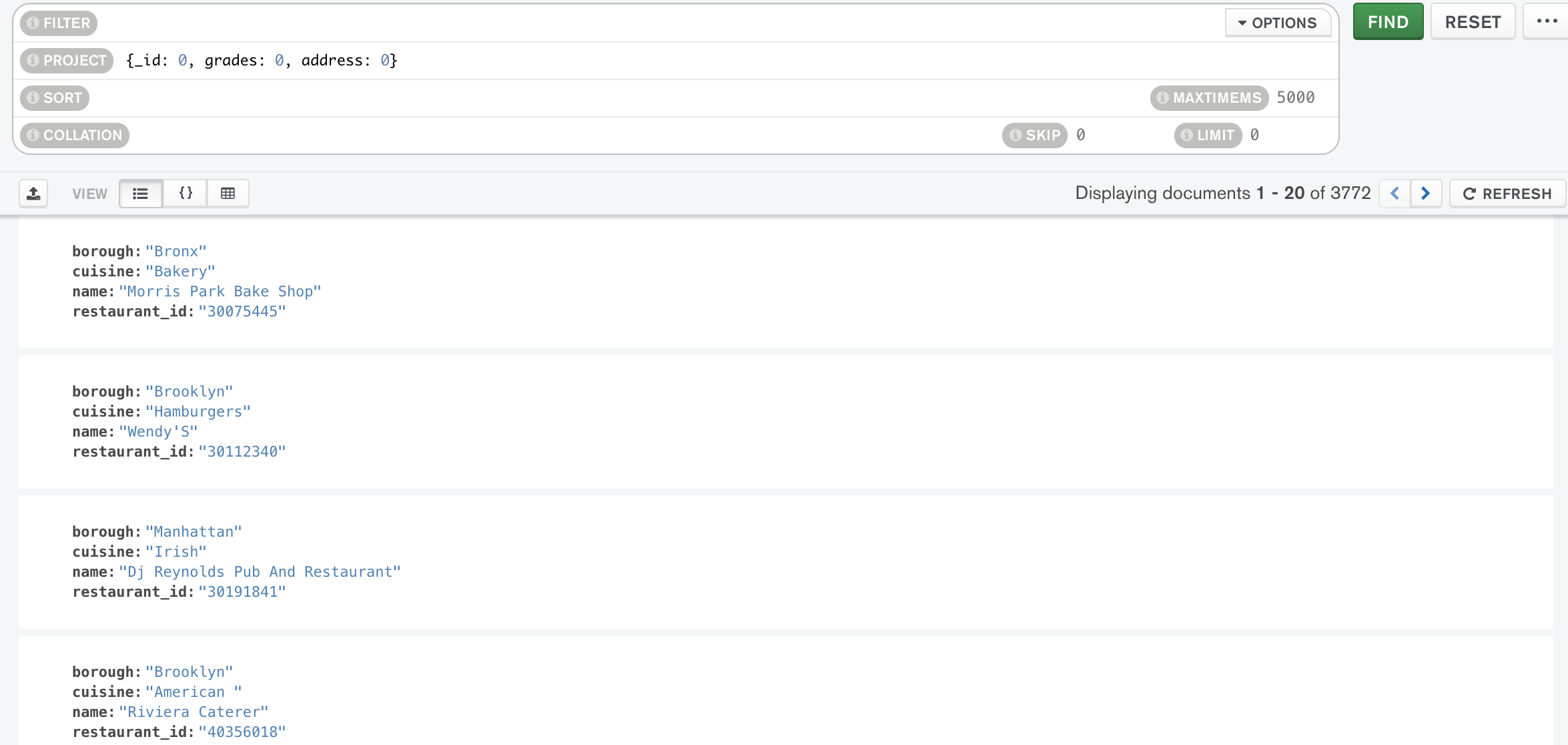
$ db.restaurants.find()

1. **Liste apenas os campos restaurante\_id, name, borough e cuisine para todos os documentos na coleção.**

****

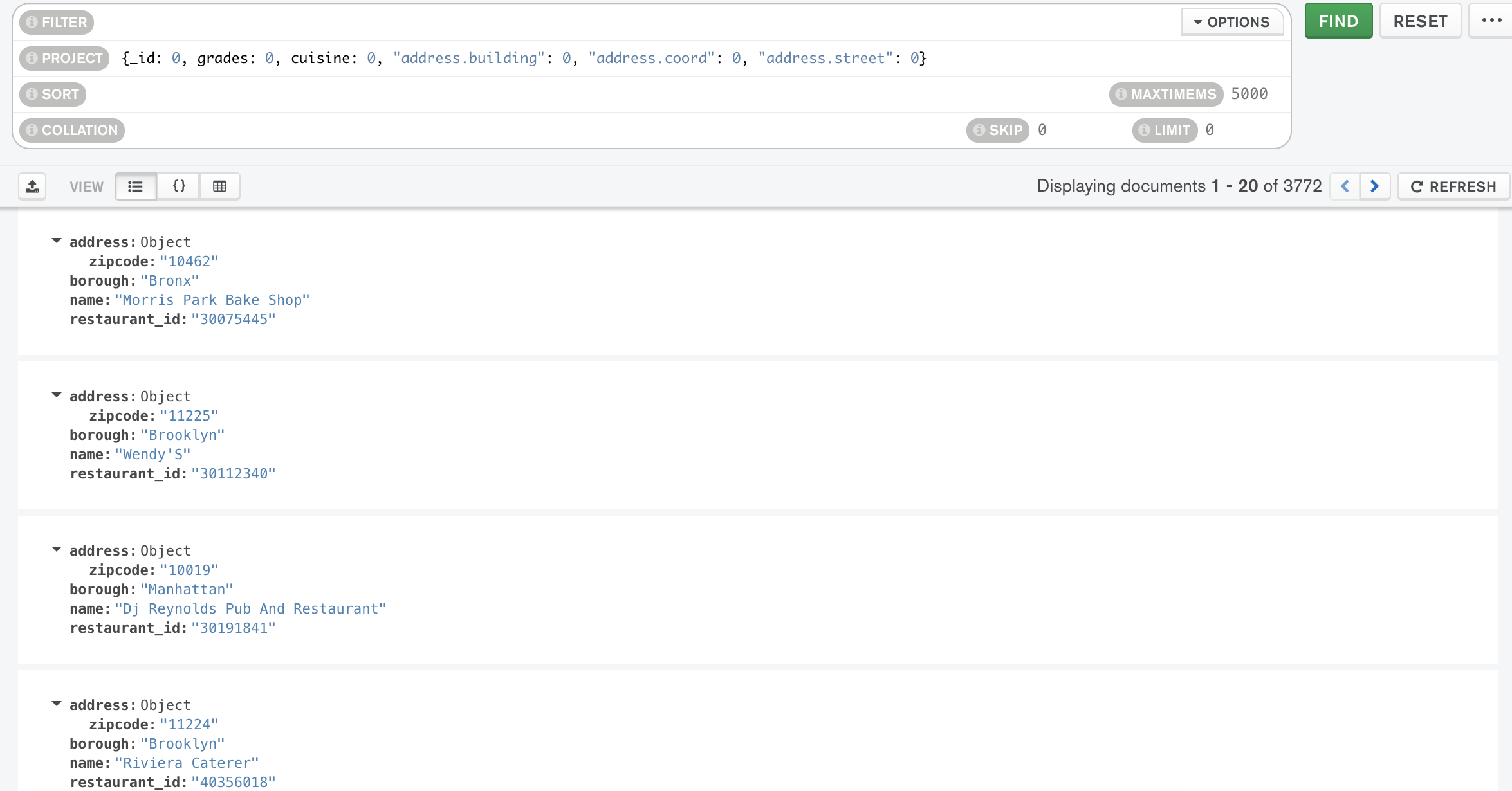
$ db.restaurants.find({},{address:0, grades: 0})

1. **Liste os campos restaurante\_id, name, borough e cuisine para todos os documentos na coleção, mas que exclua o campo \_id.**

****

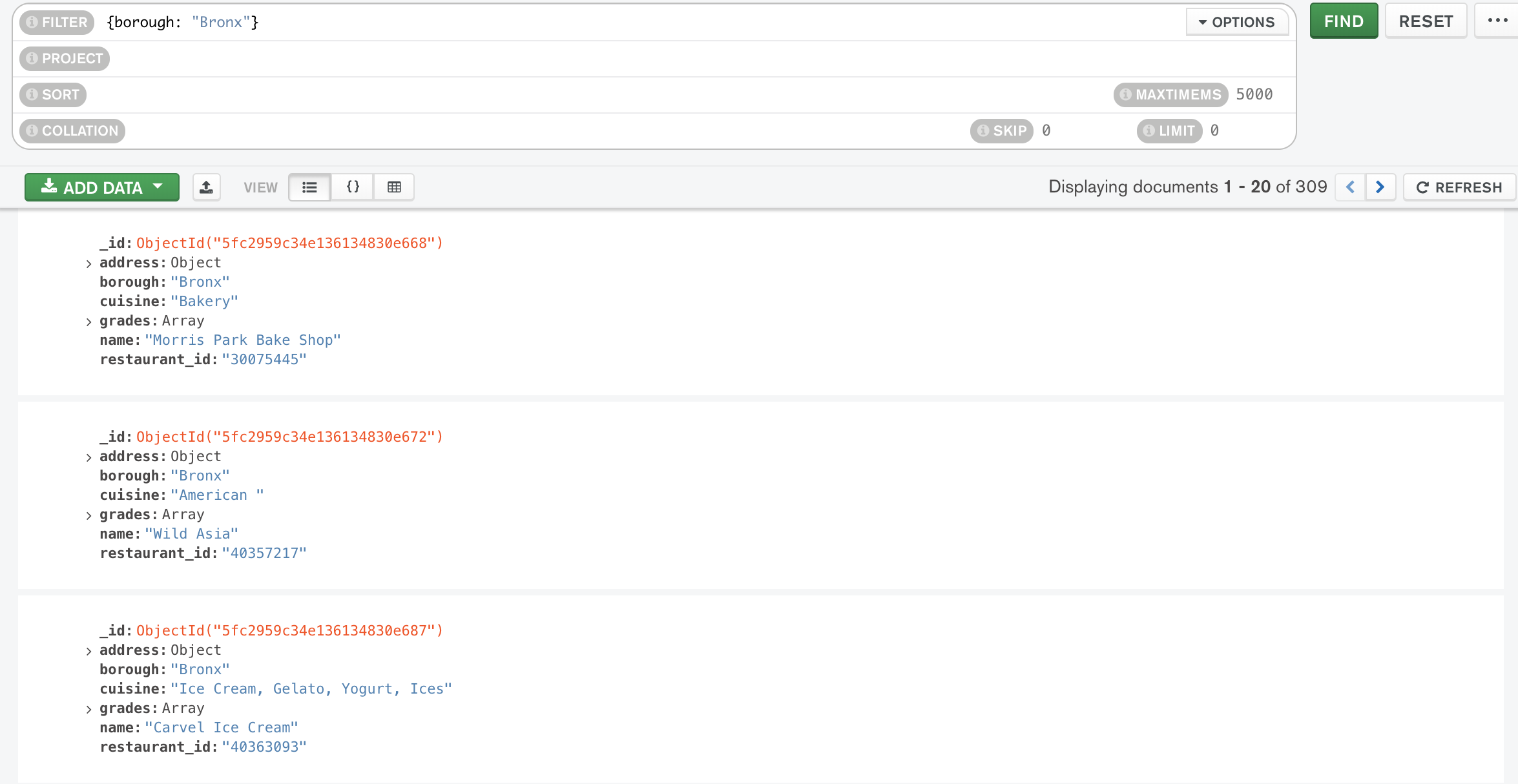
$ db.restaurants.find({},{address:0, grades: 0})

1. **Liste os campos restaurante\_id, name, borough e zipcode para todos os documentos na coleção, mas que exclua o campo \_id.**

****

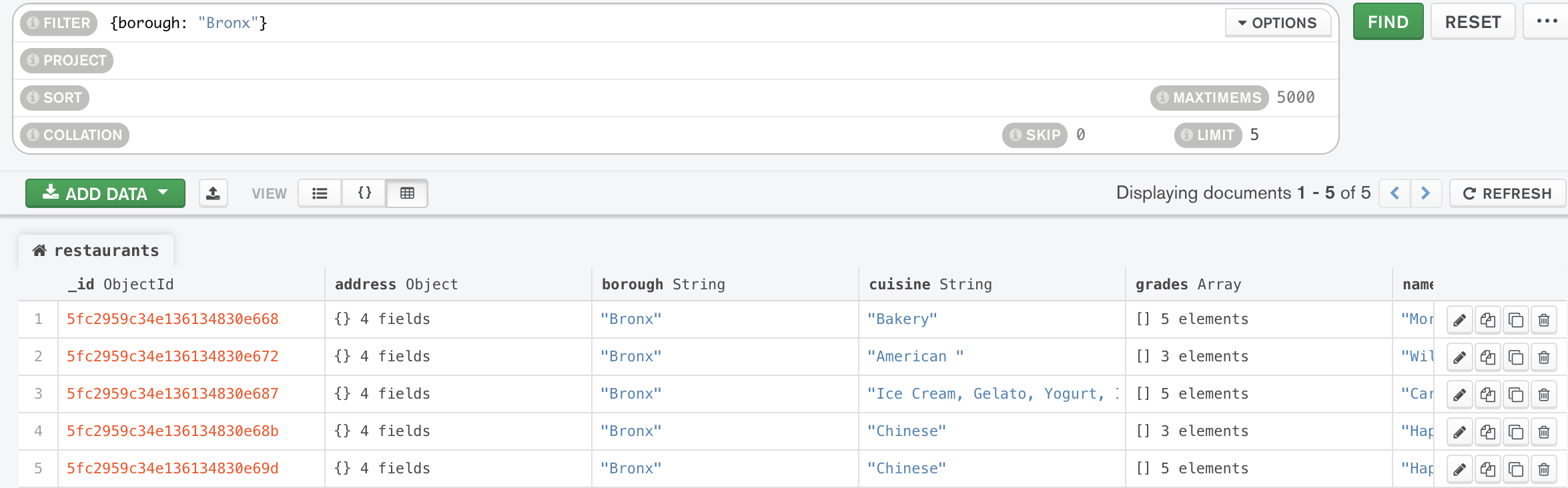
$ db.restaurants.find({},{\_id: 0, grades: 0, "address.building": 0, "address.coord": 0, "address.street": 0})

1. **Liste os restaurantes que estão localizados no bairro (borough) "Bronx".**

****

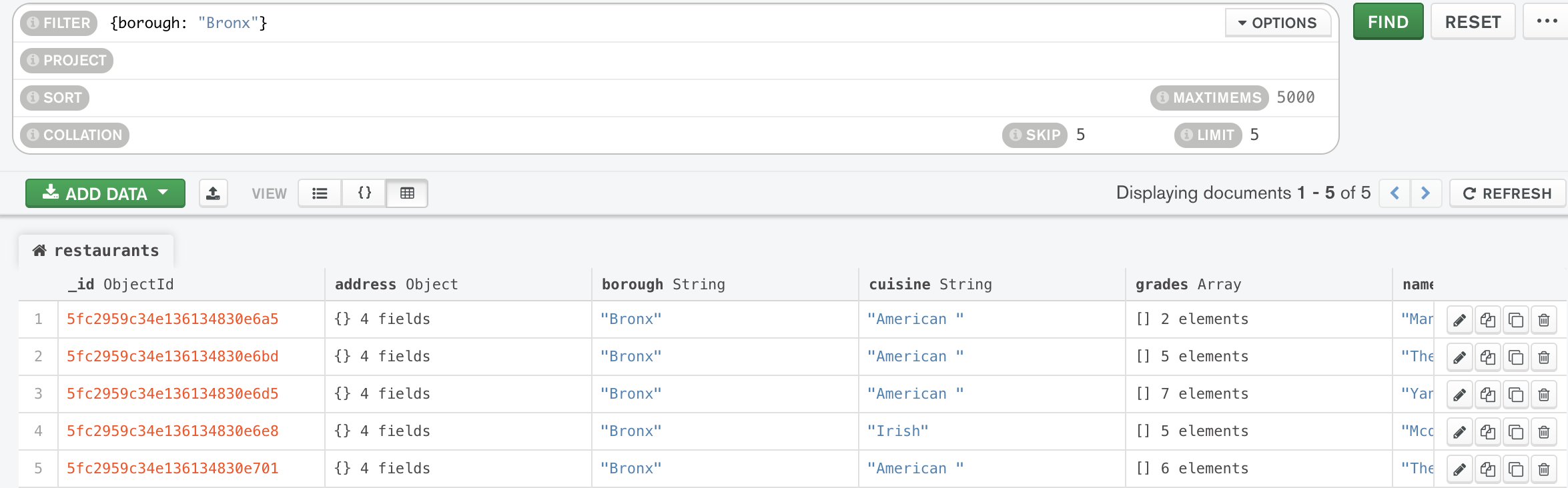
$ db.restaurants.find({borough: "Bronx"}).pretty()

1. **Liste os primeiros 5 restaurantes que estão localizados no bairro (borough) "Bronx".**

****

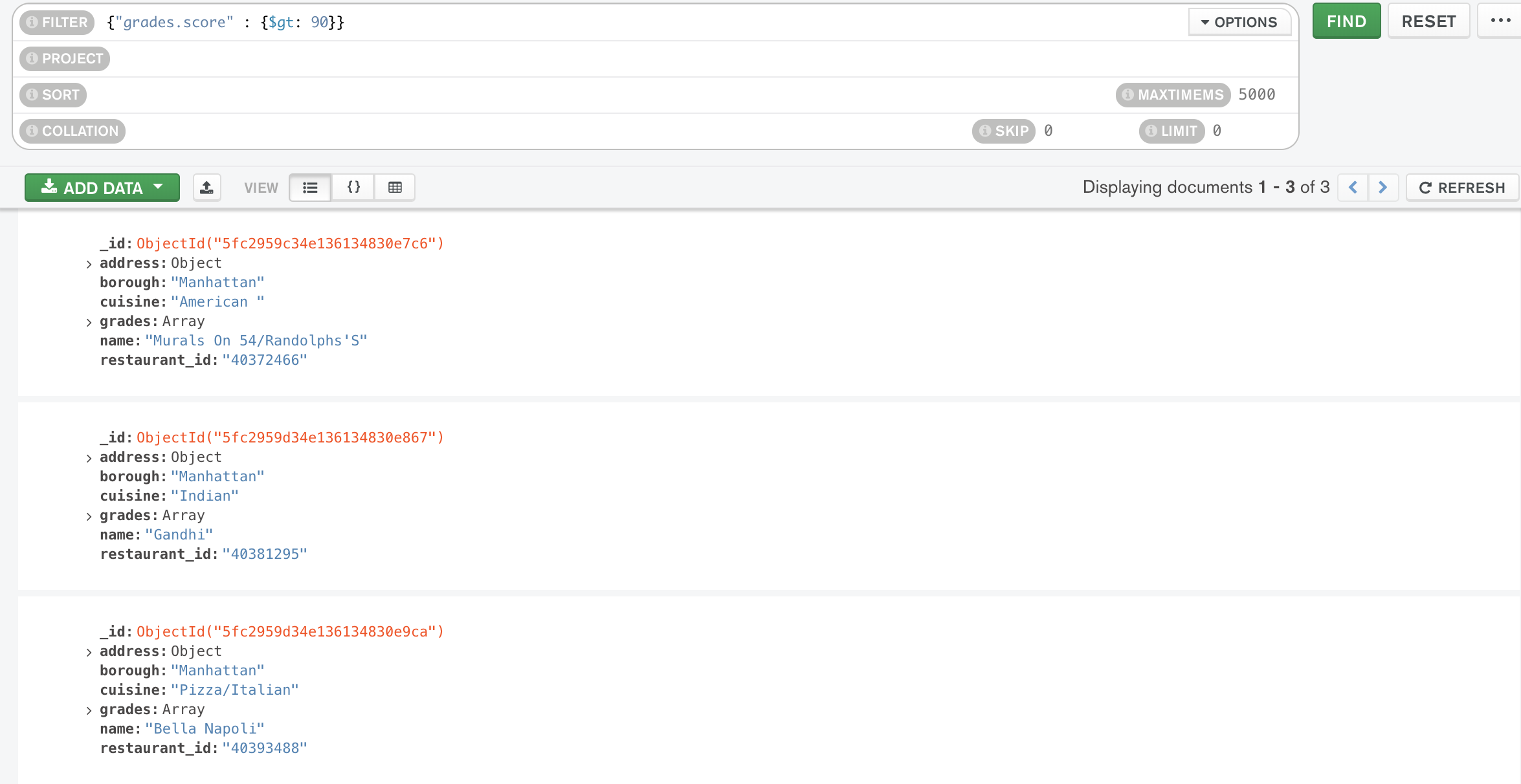
$ db.restaurants.find({borough: "Bronx"}).limit(5).pretty()

1. **Liste os 5 restaurantes após dos primeiros 5 (do 6º ao 10º) que estão localizados no bairro (borough) "Bronx".**

****

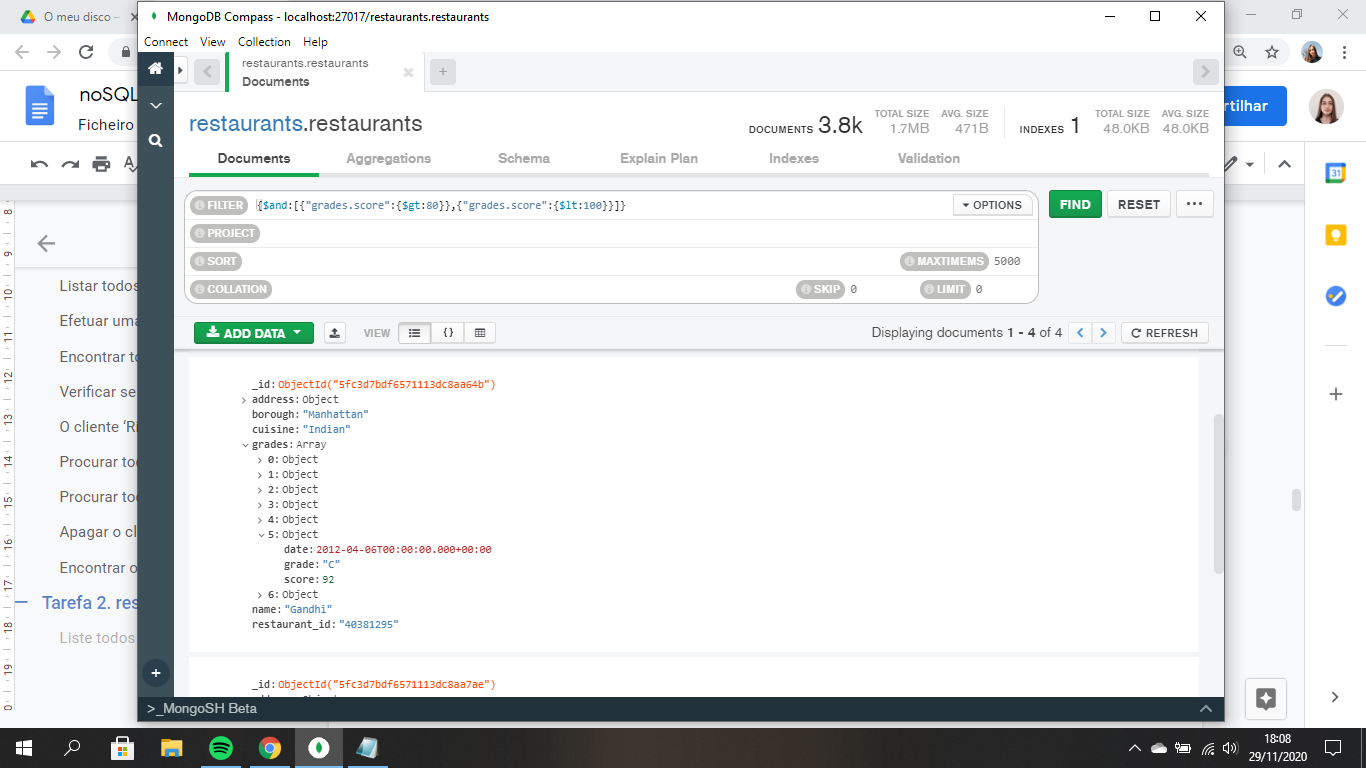
$ db.restaurants.find({borough: "Bronx"}).skip(5).limit(5).pretty()

1. **Liste todos os restaurantes que têm pelo menos uma pontuação (score) maior que 90.**

****

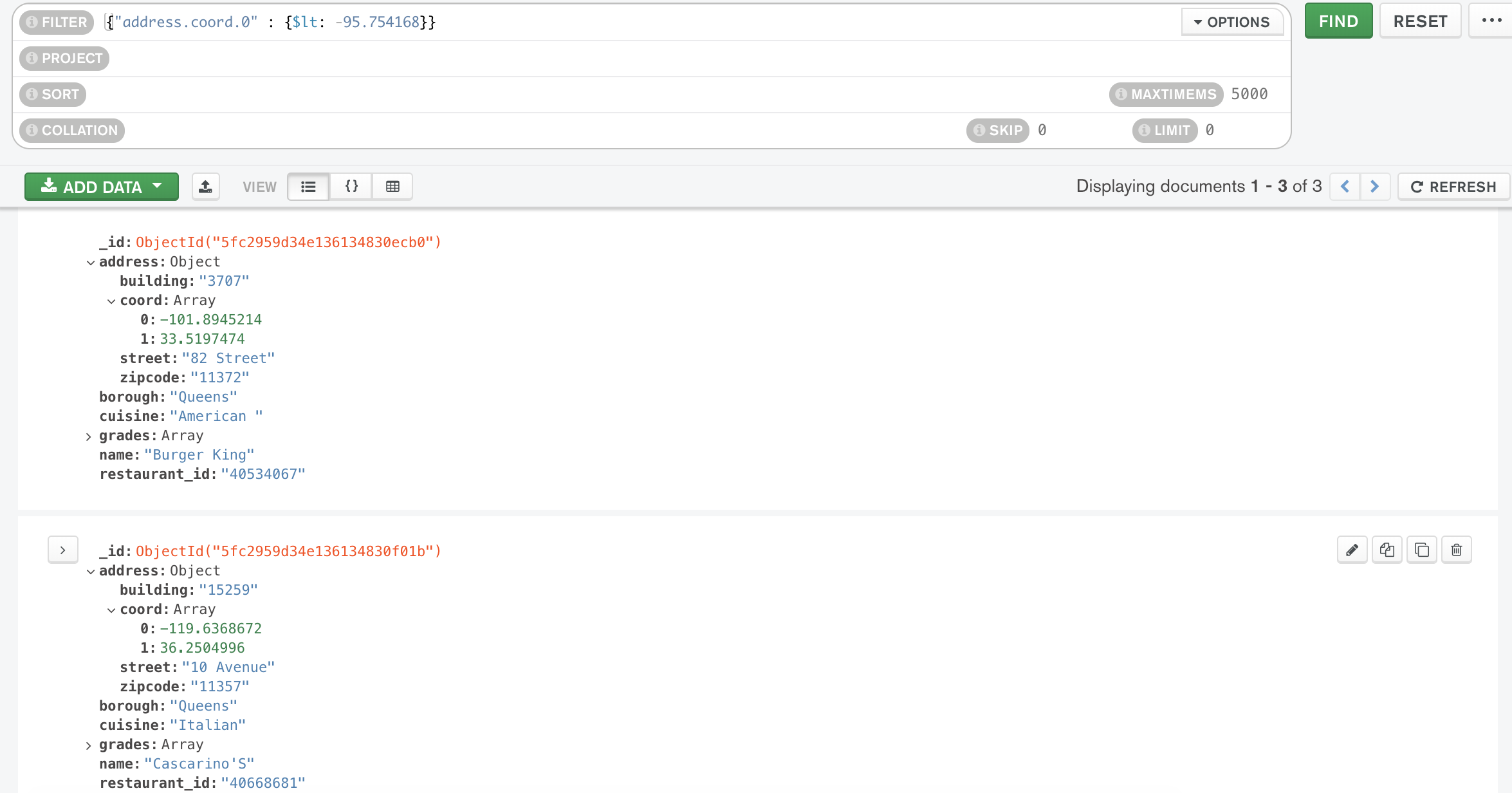
$ db.restaurants.find({"grades.score":{$gt:90}}).pretty()

1. **Liste todos os restaurantes que têm uma pontuação (score) maior que 80 mas menor que 100.**

****

$ db.restaurants.find({$and:[{"grades.score":{$gt:80}}, {"grades.score":{$lt:100}}]}).pretty()

1. **Liste todos os restaurantes que estão localizados numa latitude (coord.0) menor que -95.754168.**

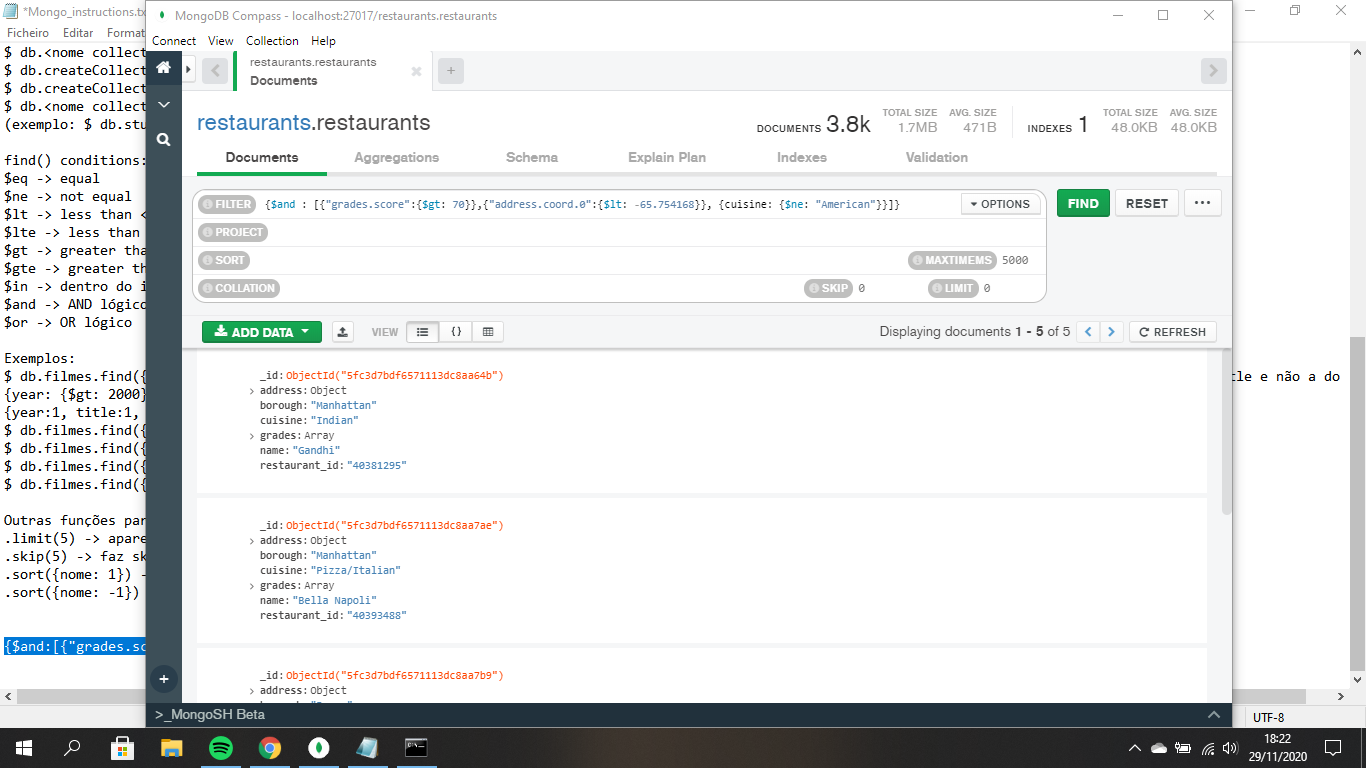
****

$ db.restaurants.find({"address.coord.0":{$lt:-95.754168}}).pretty()

1. **Recorrendo à mongo shell atualize todos os restaurantes que possuam a cozinha "American " para "American".**

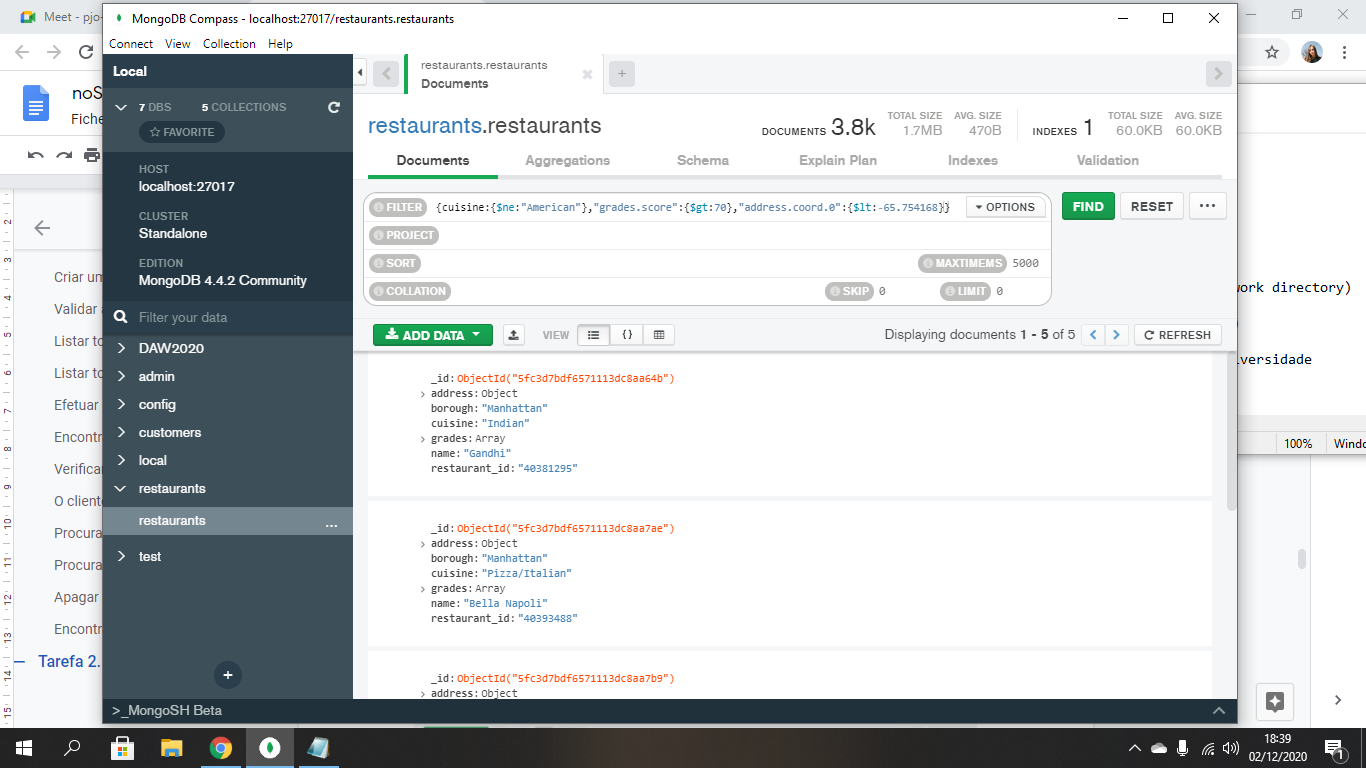
$ db.restaurants.update({cuisine: "American "}, {$set: {cuisine: "American"}}, {multi: true})

1. **Liste todos os restaurantes cujo tipo de cozinha (​cuisine​) não seja "American", que a sua pontuação (​score)​ seja maior que 70 e a latitude (​address .coord.0​) menor que -65.754168, utilizando o operador $and.**



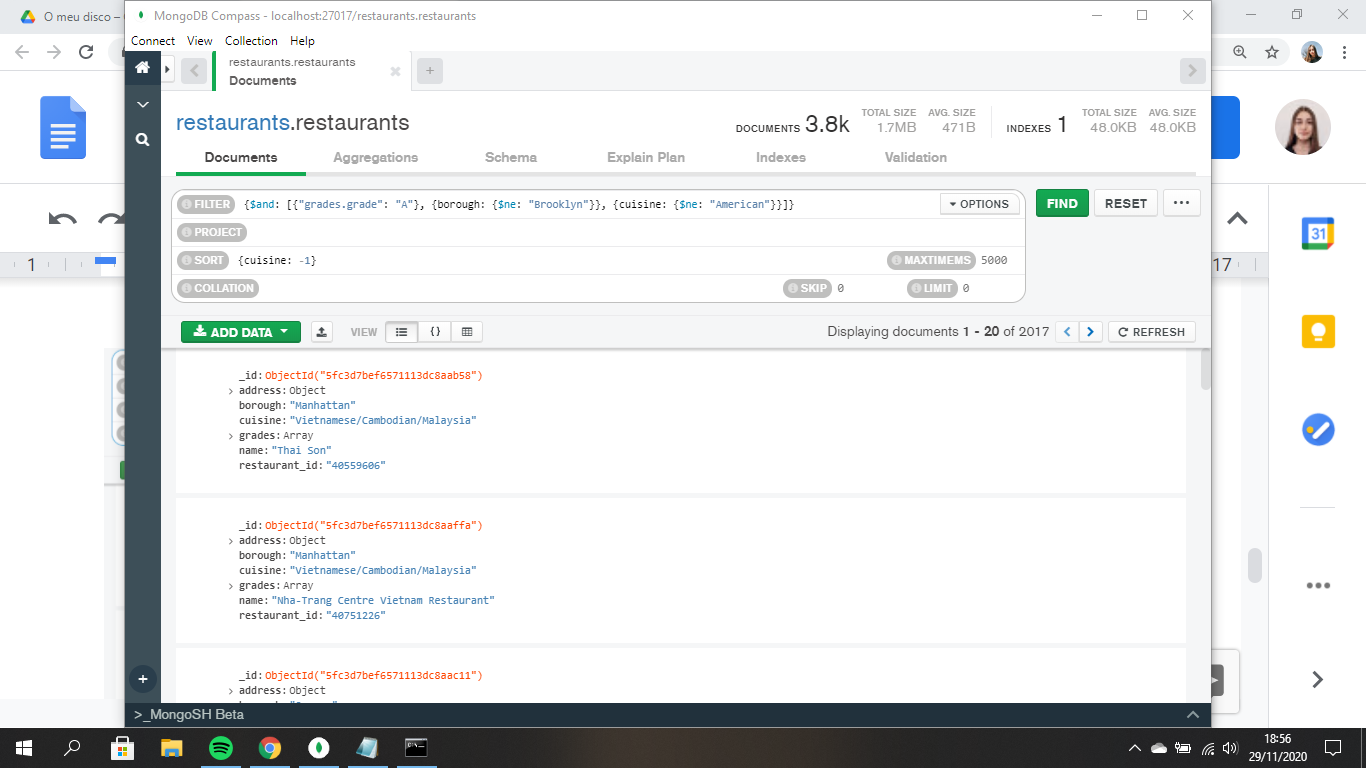
$db.restaurants.find({$and:[{cuisine:{$ne:"American"}},{"grades.score":{$gt:70}},{"address.coord.0":{$lt:-65.754168}}]}).pretty()

1. **Liste todos os restaurantes cujo tipo de cozinha (cuisine) não seja "American", que a sua pontuação (score) seja maior que 70 e a latitude (address .coord.0) menor que -65.754168.**



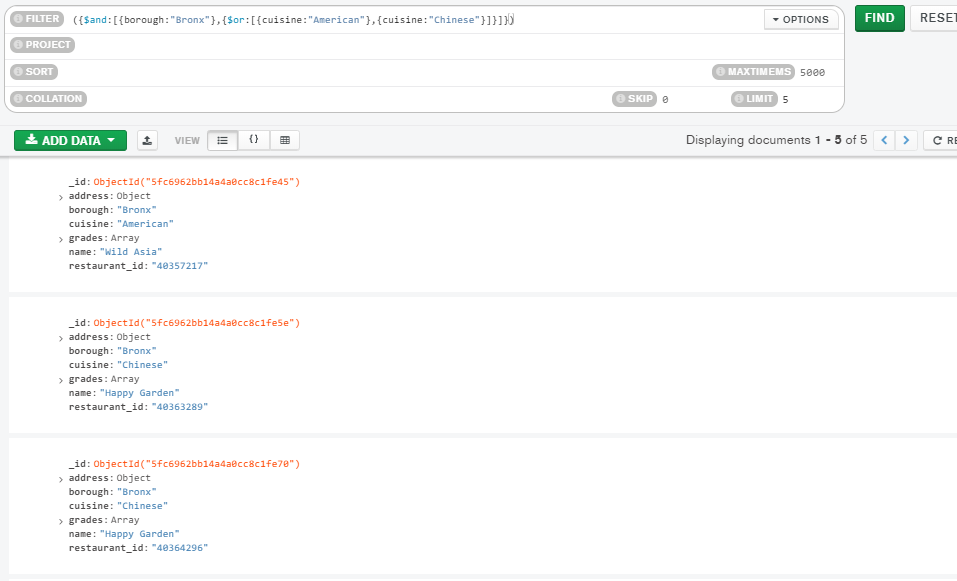
$db.restaurants.find({cuisine:{$ne:"American"},"grades.score":{$gt:70},"address.coord.0":{$lt:-65.754168}}).pretty()

1. **Liste todos os restaurantes cujo tipo de cozinha (cuisine) não seja do tipo "American" e que tenham atingido uma classificação (grade) de "A" mas que não pertençam ao bairro (borough) de "Brooklyn". Deverá ser apresentada de acordo com o tipo de cozinha (cuisine) em ordem descendente.**

****

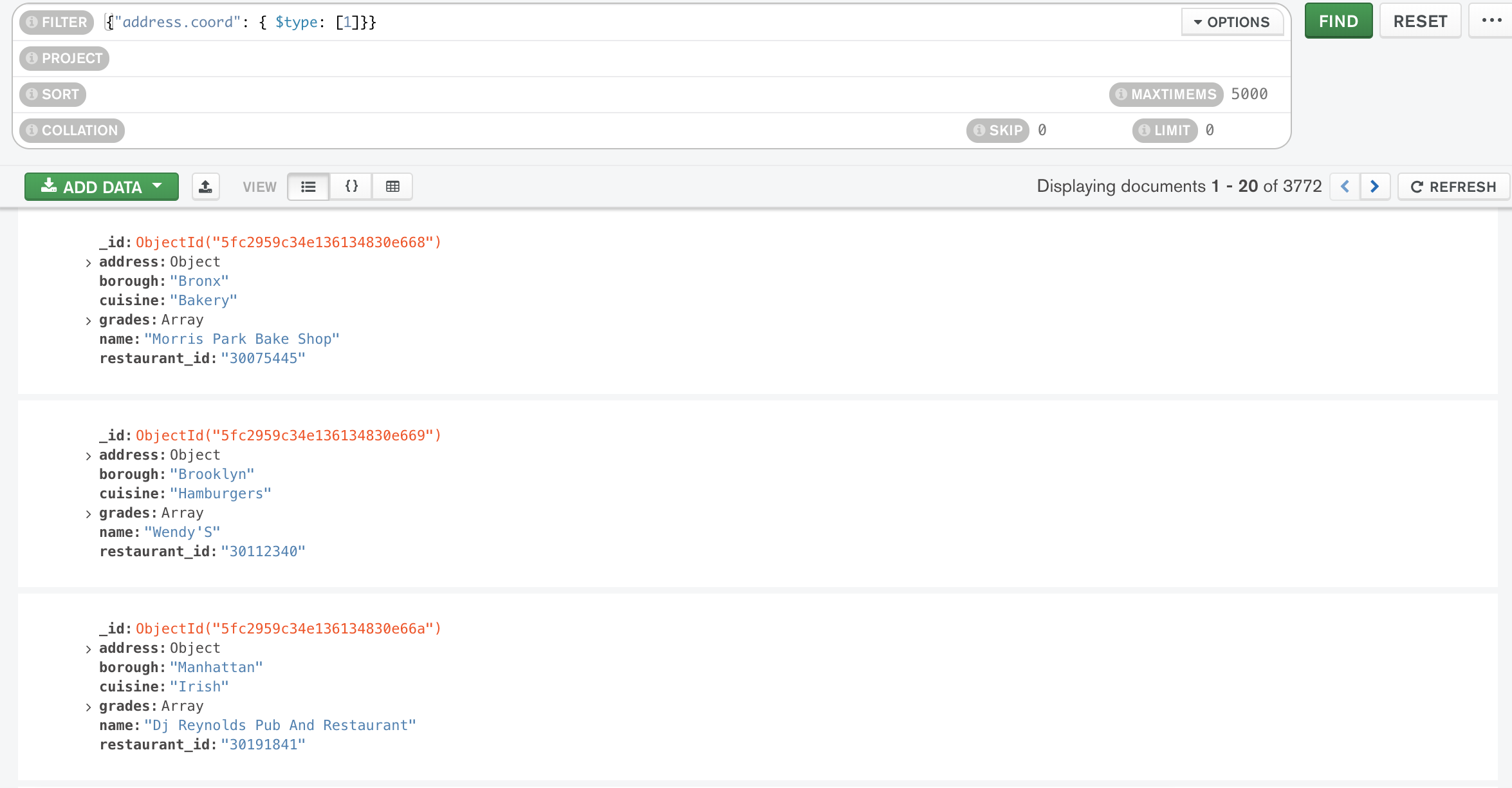
$db.restaurants.find({$and:[{cuisine:{$ne:"American"}},{"grades.grade":"A"},{borough:{$ne:"Brookly"}}]}).sort({cuisine:-1}).

1. **Liste todos os restaurantes que pertençam ao bairro (borough) "Bronx" e cujo tipo de cozinha (cuisine) seja quer "American" quer "Chinese".**

****

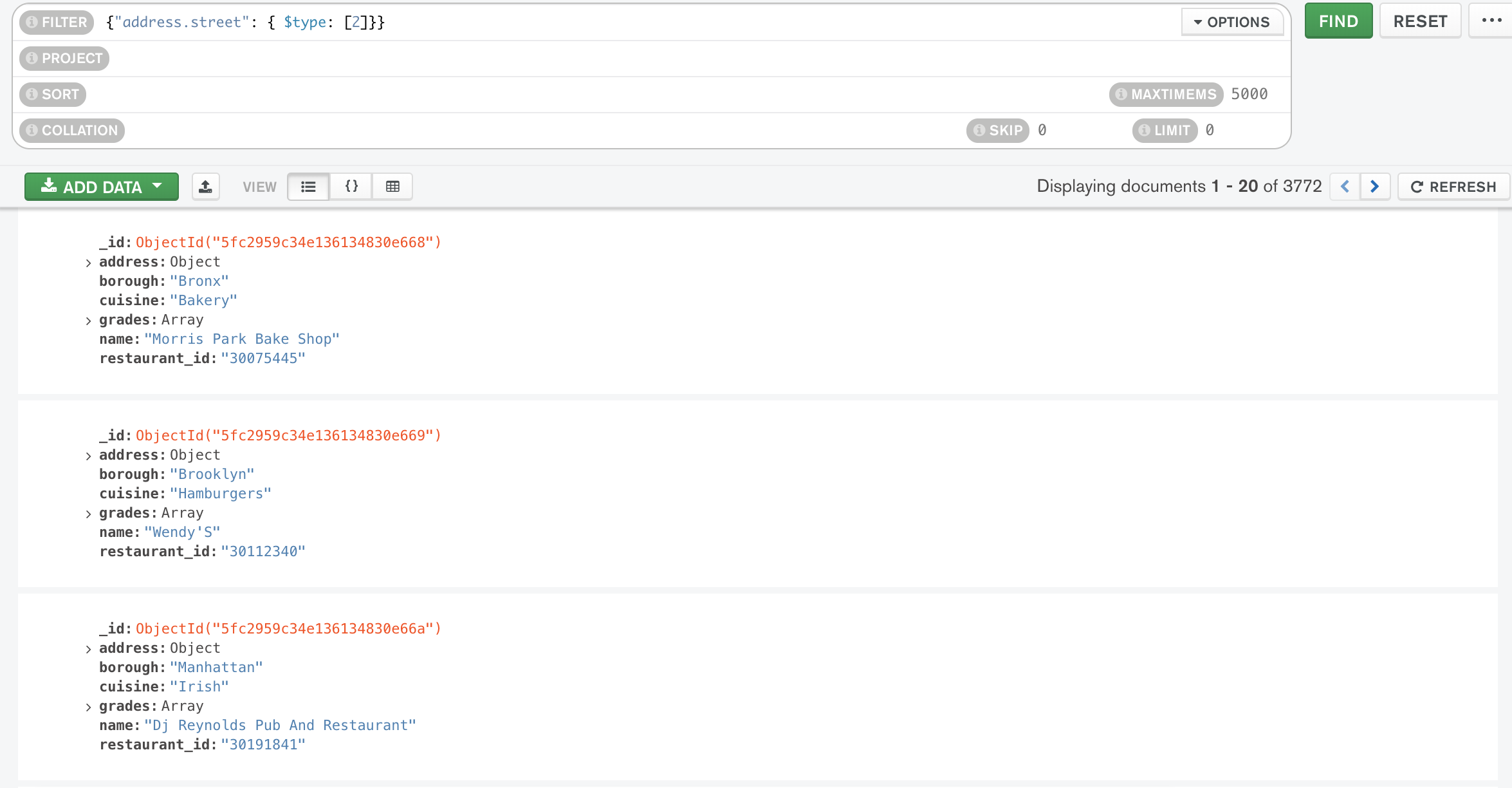
$db.restaurants.find({$and:[{borough:"Bronx"},{$or:[{cuisine:"American"},{cuisine:"Chinese"}]}]})

1. **Liste todos os restaurantes cujas coordenadas (address.coord) sejam do tipo double (type: 1).**

****

$ db.restaurants.find({"address.coord":{$type: [1]}})

1. **Liste todos os restaurantes que contenham informação da rua (address.street).**

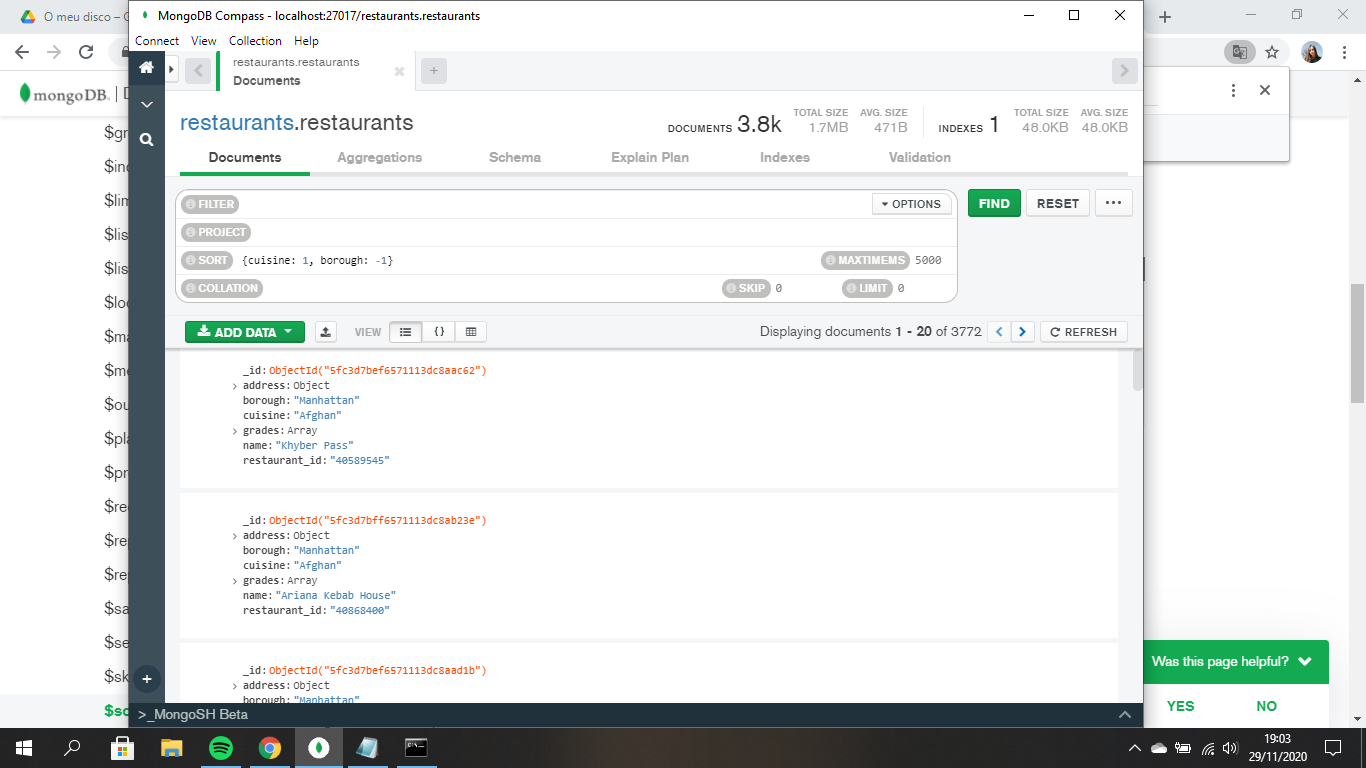
****

$ db.restaurants.find({"address.street":{$ne:null}})

OU

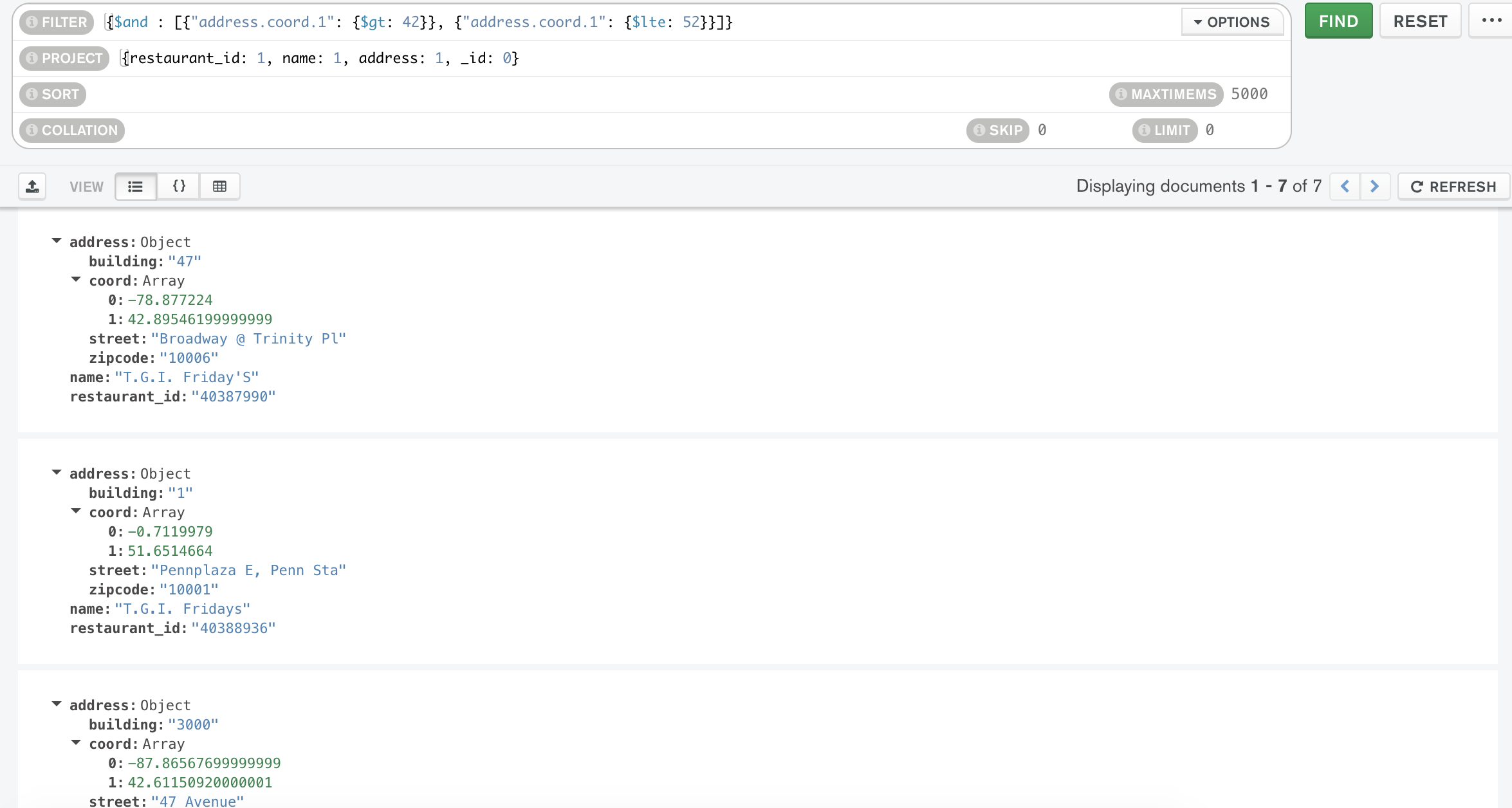
$ db.restaurants.find({"address.street":{$type:[2]}})

1. **Liste todos os restaurantes de forma ascendente pelo tipo de cozinha (cuisine) e descendente pelo bairro (borough).**

****

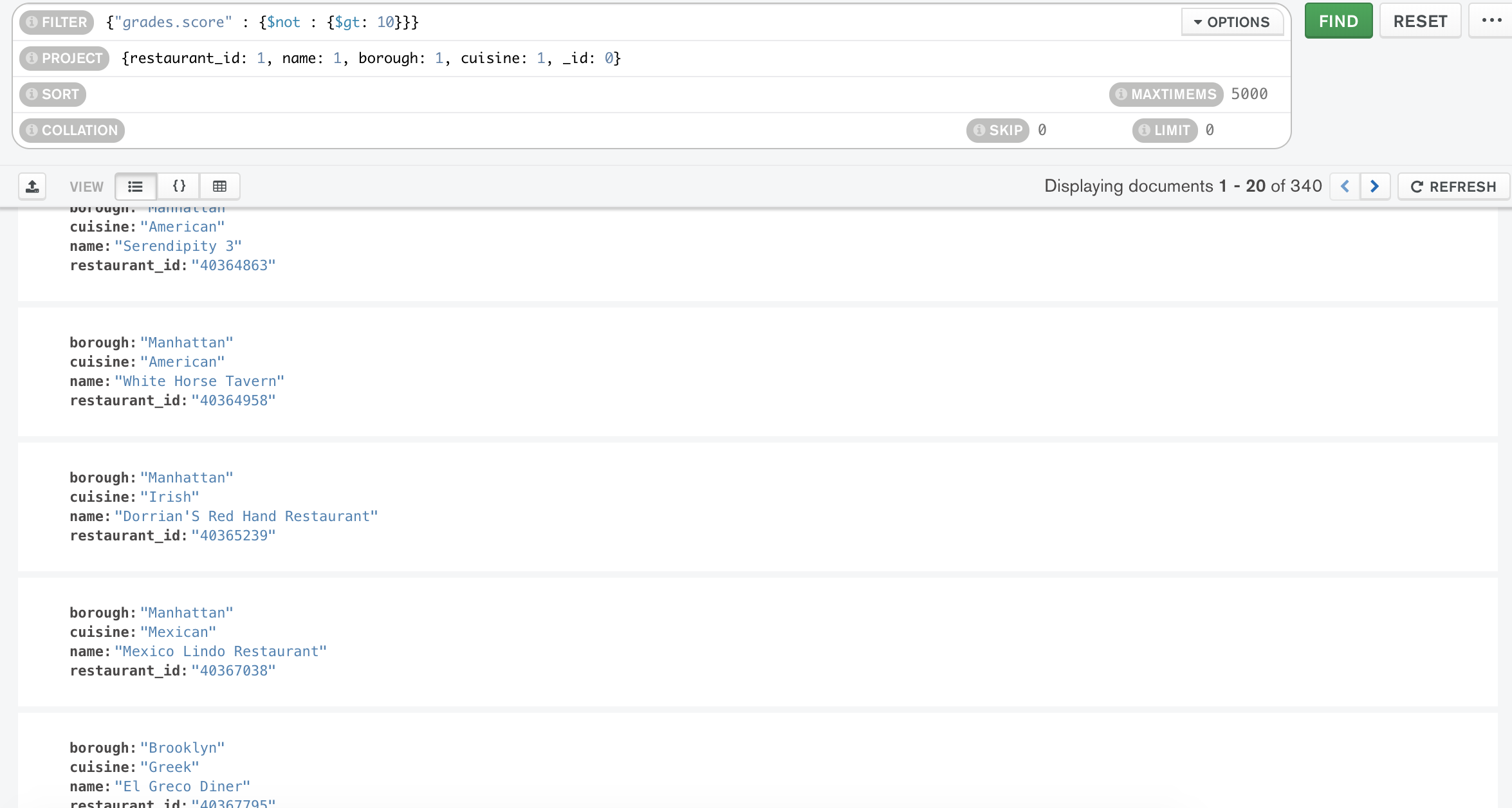
$ db.restaurants.find({}).sort({cuisine:1,borough:-1})

1. **Liste o restaurant\_id, name, address e localização geográfica (coord) para os restaurantes cujo segundo elemento do array da localização geográfica (coord) seja maior que 42 e até 52.**

****

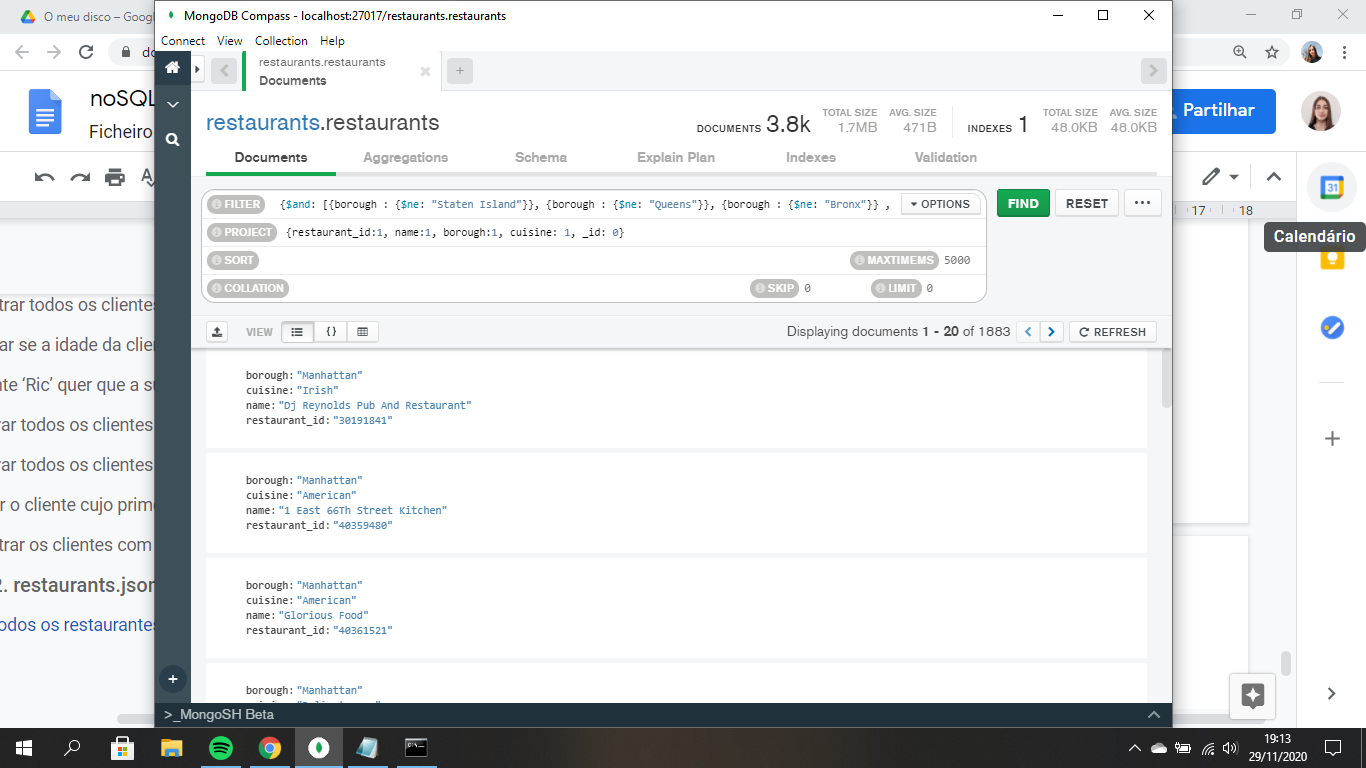
$db.restaurants.find({$and:[{"address.coord.1":{$gt:42}},{"address.coord.1":{$lt:52}}]},{restaurant\_id:1,name:1,address:1,\_id:0})

1. **Liste os restaurantes (restaurante\_id, name, borough, cuisine) que não conseguiram uma pontuação (score) maior que 10.**

****

$db.restaurants.find({"grades.score":{$lte:10}},{restaurants\_id:1,name:1,cuisine:1,\_id:0})

1. **Liste todos os restaurantes (restaurante\_id, name, borough e cuisine) que não pertencem ao bairro (borough) de "Staten Island", ou "Queens" ou "Bronx" ou "Brooklin".**

****

$ db.restaurants.find({$and: [{borough : {$ne: "Staten Island"}}, {borough : {$ne: "Queens"}}, {borough : {$ne: "Bronx"}} , {borough : {$ne: "Brooklyn"}}]},{restaurants\_id:1,borough:1,cuisine:1,\_id:0})