11

Faça como eu fiz

Chegou a hora de você pôr em prática o que foi visto na aula. Para isso, execute os passos listados abaixo.

1) Aqui iremos ver comandos de consultas a base **MongoDB**.

2) Primeiro o comando para exibir todos os dados de uma coleção:

```
db.funcionario.find({})
```

3) Podemos escolher os valores a serem exibidos na saída do comando. Igual a 0 significa excluir da saída e igual a 1 manter na saída.

```
db.funcionario.find({},{"_id": 0, "Primeiro_Nome": 1, "Ultimo_Nome": 1, "Sexo": 1})
```

O comando acima mostra os elementos **Primeiro_Nome**, **Ultimo_Nome** e **Sexo** sendo exibidos. Já **_id** é excluído da saída dos dados.

4) Podemos limitar a saída de dados:

```
db.funcionario.find({},{"Primeiro_Nome": 1, "Ultimo_Nome": 1, "Sexo": 1}).limit(5)
```

5) Ou ordenar a saída de dados de forma ascendente:

```
db.funcionario.find({},{"Primeiro_Nome": 1, "Ultimo_Nome": 1, "Sexo": 1}).sort({"Primeiro_Nome"
```

6) Ou de forma descendente:

```
db.funcionario.find({},{"Primeiro_Nome": 1, "Ultimo_Nome": 1, "Sexo": 1}).sort({"Primeiro_Nome"
```

7) Os dois comandos podem ser executados em conjunto:

```
db.funcionario.find({},{"_id": 0,"Primeiro_Nome": 1, "Ultimo_Nome": 1, "Sexo": 1}).sort({"Prime
```

8) Podemos filtrar os dados a serem exibidos. O comando abaixo exibe o **Primeiro_Nome**, **Ultimo_Nome** e o **Sexo** apenas para funcionários do sexo masculino.

```
db.funcionario.find({"Sexo": "M"},{"_id": 0,"Primeiro_Nome": 1, "Ultimo_Nome": 1, "Sexo": 1}).s
```

9) O filtro pode ser aplicado a campos de valor, como mostrado abaixo:

```
db.funcionario.find({"Salario": 31000},{"_id": 0,"Primeiro_Nome": 1, "Salario": 1})
```

10) Também podemos aplica consultas envolvendo condições do tipo maior, menor, maior ou igual ou menor ou igual. Abaixo temos o comando para exibir maiores que 31000.

```
db.funcionario.find({"Salario": {$gte: 31000}},{"_id": 0,"Primeiro_Nome": 1, "Salario": 1})
```

11) Se vamos aplicar condições sobre as datas temos que usar a função **New Date**.

```
db.funcionario.find({"Data_Nascimento": {$gte: new Date("1990-01-01")}},{"_id": 0,"Primeiro_Nome
```

12) Para usar a condição **in** procedemos como mostrado abaixo:

```
db.funcionario.find({"Primeiro_Nome": {$in: ["Fatima","Tonico"] }},{"_id": 0,"Primeiro_Nome": 1
```

13) Quando, nas opções de filtro, colocamos mais de uma condição, ela funciona como se fosse um **AND**. Veja abaixo:

```
db.funcionario.find({"Numero_Departamento": "5"},{"_id": 0,"Primeiro_Nome": 1, "Salario": 1, "Nu
```

14) Caso queira usar o **OR** tenho que representa-lo explicitamente:

```
db.funcionario.find({$or: [{"Numero_Departamento": "5"}, {"Sexo": "M"}]},{"_id": 0,"Primeiro_Nor
```

15) Podemos mesclar o **AND** ou **OR** com condições de filtros.

```
db.funcionario.find({$and: [{"Salario": {$gt:30000}}, {"Sexo": "M"}]},{"_id": 0,"Primeiro_Nome"
```

16) Estamos aptos a executar o desafio de listar os funcionários que não possuem dependentes. O resultado seria o mostrado abaixo:

```
db.funcionario.find({Dependentes: {$exists: false}},{"_id": 0, "Primeiro_Nome": 1, "Nome_Meio":
```