Utilizandos conceitos aprendidos na aula anterior crie um novo database de cliente com uma collection chamada: clientes e insira pelo menos 20 documentos.

Abaixo um exemplo de insertMany, mas pode inserir outros valores caso queira.

db.clientes.insertMany([

{ nome: "Ana Lima", idade: 22, cidade: "Salvador", interesses: ["viagem", "música"], pontosFidelidade: 120 },

{ nome: "Bruno Souza", idade: 28, cidade: "Salvador", interesses: ["esportes", "tecnologia"], pontosFidelidade: 80 },

{ nome: "Carla Nunes", idade: 35, cidade: "Feira", interesses: ["culinária", "leitura"], pontosFidelidade: 200 },

{ nome: "Diego Alves", idade: 41, cidade: "Lauro", interesses: ["corrida", "viagem"], pontosFidelidade: 45 },

{ nome: "Eduarda Reis", idade: 30, cidade: "Camaçari", interesses: ["cinema", "música"], pontosFidelidade: 150 },

{ nome: "Felipe Araújo", idade: 19, cidade: "Salvador", interesses: ["games", "tecnologia"], pontosFidelidade: 15 },

{ nome: "Gabriela Mota", idade: 27, cidade: "Feira", interesses: ["fotografia", "viagem"], pontosFidelidade: 95 },

{ nome: "Heitor Prado", idade: 33, cidade: "Lauro", interesses: ["esportes", "corrida"], pontosFidelidade: 60 },

{ nome: "Isabela Dias", idade: 30, cidade: "Salvador", interesses: ["yoga", "leitura"], pontosFidelidade: 220 },

{ nome: "João Pedro", idade: 45, cidade: "Camaçari", interesses: ["tecnologia", "música"], pontosFidelidade: 300 },

{ nome: "Karla Menezes", idade: 24, cidade: "Feira", interesses: ["cinema", "fotografia"], pontosFidelidade: 55 },

{ nome: "Lucas Rocha", idade: 38, cidade: "Lauro", interesses: ["viagem", "culinária"], pontosFidelidade: 130 },

{ nome: "Mariana Silva", idade: 26, cidade: "Salvador",interesses: ["corrida", "esportes"], pontosFidelidade: 75 },

{ nome: "Nathan Costa", idade: 32, cidade: "Salvador", interesses: ["leitura", "música"], pontosFidelidade: 110 },

{ nome: "Olívia Pires", idade: 42, cidade: "Feira", interesses: ["viagem", "yoga"], pontosFidelidade: 260 },

{ nome: "Paulo Gomes", idade: 25, cidade: "Camaçari", interesses: ["tecnologia", "games"], pontosFidelidade: 40 },

{ nome: "Quezia Freitas", idade: 29, cidade: "Lauro", interesses: ["fotografia", "viagem"], pontosFidelidade: 140 },

{ nome: "Rafael Brito", idade: 31, cidade: "Salvador", interesses: ["esportes", "cinema"], pontosFidelidade: 85 },

{ nome: "Sofia Matos", idade: 37, cidade: "Feira", interesses: ["leitura", "culinária"], pontosFidelidade: 170 },

{ nome: "Tiago Neri", idade: 21, cidade: "Salvador", interesses: ["yoga", "música"], pontosFidelidade: 35 },

{ nome: "Ursula Prado", idade: 34, cidade: "Camaçari", interesses: ["corrida", "viagem"], pontosFidelidade: 190 },

{ nome: "Vitor Azevedo", idade: 30, cidade: "Lauro", interesses: ["tecnologia", "fotografia"], pontosFidelidade: 210 },

{ nome: "Willian Santos", idade: 46, cidade: "Feira", interesses: ["esportes", "viagem"], pontosFidelidade: 320 },

{ nome: "Yasmin Queiroz", idade: 23, cidade: "Camaçari",interesses: ["cinema", "games"], pontosFidelidade: 65 },

{ nome: "Zeca Carvalho", idade: 39, cidade: "Salvador",interesses: ["culinária", "leitura"], pontosFidelidade: 155 }

]);

**1) $eq (igual) — explique e pratique**

**O que faz:** retorna documentos cujo campo é **exatamente igual** ao valor informado.

**Tarefas**

1. Encontre clientes com **idade = 30**.

db.clientes.find({ idade: { $eq: 30 } });

1. Liste clientes cuja **cidade = "Salvador"**.

db.clientes.find({ cidade: { $eq: “Salvador” } });

Ana Lima, Bruno Souza, Felipe Araújo, Isabela Dias, Mariana Silva, Nathan Costa, Rafael Brito, Tiago Neri, Zeca Carvalho.

1. Busque quem tem **nome = "João Pedro"**.

db.clientes.find({ nome: { $eq: "João Pedro" } });

**2) $ne (diferente)**

**O que faz:** retorna documentos cujo campo é **diferente** do valor informado.

**Tarefas**

1. Clientes com **idade ≠ 25**.

db.clientes.find({ idade: { $ne: 25 } });

1. Clientes cuja **cidade ≠ "Feira"**.

db.clientes.find({ cidade: { $ne: "Feira" } });

1. Clientes cujo **nome ≠ "Ana Lima"**.

db.clientes.find({ nome: { $ne: "Ana Lima" } });

1. Clientes cuja **idade ≠ 30** **e** **cidade = "Salvador"** (combinações ainda usando apenas os operadores listados).

db.clientes.find({ nome: { $ne: 30 }, cidade: { $eq: "Salvador" } });

**3) $gt (maior que)**

**O que faz:** retorna documentos cujo valor do campo é **maior** que o informado.

**Tarefas**

1. Idade **> 40**.

db.clientes.find({ idade: { $gt: 40 } });

1. Pontos de fidelidade **> 200**.

db.clientes.find({ pontosFidelidade: { $gt: 200 } });

1. Idade **> 30** e **cidade = "Lauro"**.

db.clientes.find({ idade: { $gt: 30 }, cidade: { $eq: "Lauro" } });

1. Idade **> 25** para clientes **de Salvador**.

db.clientes.find({ idade: { $gt: 25 }, cidade: { $eq: "Salvador" } });

**4) $lt (menor que)**

**O que faz:** retorna documentos cujo valor do campo é **menor** que o informado.

**Tarefas**

1. Idade **< 30**.

db.clientes.find({ idade: { $lt: 30 } });

1. Pontos de fidelidade **< 100**.

db.clientes.find({ pontosFidelidade: { $lt: 100 } });

1. Idade **< 30** e **cidade = "Feira"**.

db.clientes.find({ idade: { $lt: 30 }, cidade: { $eq: "Feira" } });

1. Idade **< 25** **em Salvador**.

db.clientes.find({ idade: { $lt: 25 }, cidade: { $eq: "Salvador" } });

**5) $gte (maior ou igual)**

**O que faz:** retorna documentos com valor **≥** ao informado.

**Tarefas**

1. Idade **≥ 35**.

db.clientes.find({ idade: { $gte: 35 } });

1. Pontos de fidelidade **≥ 150**.

db.clientes.find({ pontosFidelidade: { $gte: 150 } });

1. Idade **≥ 30** **em "Camaçari"**.

db.clientes.find({ idade: { $gte: 30 }, cidade: { $eq: "Camaçari" } });

1. Idade **≥ 40** **ou** cidade **"Feira"** (faça as duas consultas separadas usando apenas os operadores listados).

db.clientes.find({ idade: { $gte: 40 }});

db.clientes.find({ cidade: { $eq: "Feira" } });

**6) $lte (menor ou igual)**

**O que faz:** retorna documentos com valor **≤** ao informado.

**Tarefas**

1. Idade **≤ 30**.

db.clientes.find({ idade: { $lte: 30 } });

1. Pontos de fidelidade **≤ 80**.

db.clientes.find({ pontosFidelidade: { $lte: 80 } });

1. Idade **≤ 28** **em "Feira"**.

db.clientes.find({ idade: { $lte: 28 }, cidade: { $eq: "Feira" } });

1. Idade **≤ 25** **em "Camaçari"**.

db.clientes.find({ idade: { $lte: 25 }, cidade: { $eq: "Camaçari" } });

**7) $in (pertence à lista)**

**O que faz:** retorna documentos cujo campo **contém algum valor** presente na lista informada (para arrays, verifica a presença).

**Tarefas**

1. Interesses em **"viagem"** ou **"esportes"**.

db.clientes.find({ interesses: { $in: ["viagem", "esportes"] } });

1. Clientes de **cidades** dentro de ["Salvador", "Feira"].

db.clientes.find({ cidade: { $in: ["Salvador", "Feira"] } });

1. Idades dentro de **[19, 22, 30, 45]**.

db.clientes.find({ idade: { $in: [19, 22, 30, 45] } });

**8) $set (atualizar campos)**

**O que faz:** atualiza o valor de um campo existente ou cria o campo se não existir.

**Tarefas**

Substitua ObjectId("...") por um \_id real encontrado via find().

1. Atualize o **nome** para "Carlos Souza" de um cliente específico.

db.clientes.updateOne({ \_id: ObjectId("...") }, { $set: { nome: "Carlos Souza" } });

1. Defina **cidade = "Salvador"** para um cliente de outra cidade.

db.clientes.updateOne({ \_id: ObjectId("68d72e4bfdcad3f757d2783a") }, { $set: { cidade: "Salvador" } });

1. Marque um cliente como **vip = true** (novo campo).

db.clientes.updateOne({ \_id: ObjectId("68d72e4bfdcad3f757d2783a") }, { $set: { vip: true } });

1. db.clientes.updateOne({ \_id: ObjectId("...") }, { $set: { vip: true } });

Feito acima.

1. Padronize um registro para ter **pontosFidelidade = 100**.

db.clientes.updateOne({ \_id: ObjectId("68d72e4bfdcad3f757d2783a") }, { $set: { pontosFidelidade: 100 } });

**9) $inc (incrementar/decrementar)**

**O que faz:** incrementa (ou decrementa, se negativo) um campo numérico.

**Tarefas**

1. Aumente a **idade** em **+5**.

db.clientes.updateOne({ \_id: ObjectId("...") }, { $inc: { idade: 5 } });

1. Some **+20** em **pontosFidelidade**.

db.clientes.updateOne({ \_id: ObjectId("68d72e4bfdcad3f757d2783a") }, { $inc: { pontosFidelidade: 20 } });

1. Desconte **-10** em **pontosFidelidade** (ex.: ajuste por uso de benefício).

db.clientes.updateOne({ \_id: ObjectId("68d72e4bfdcad3f757d2783a") }, { $inc: { pontosFidelidade: -10 } }); Obs: Não entendi o parêntese

1. Para um cliente de **idade 30**, incremente em **+1** apenas esse registro (combine com $eq na consulta).

db.clientes.find({ idade: { $eq: 30 } });

db.clientes.updateOne({ \_id: ObjectId("68d72e4bfdcad3f757d2783a") }, { $inc: { idade: 1 } });

OU:

db.clientes.updateOne({ idade: { $eq: 30 } }, { $inc: { pontosFidelidade: 1 } });

**10) $group (agregação)**

**O que faz:** agrupa documentos para cálculos, como média, soma, contagem etc. (exemplo de média no seu material).

**Tarefas**

1. **Média de idade** geral.

db.clientes.aggregate([{ $group: { \_id: null, mediaIdade: { $avg: "$idade" } } }]);

1. **Média de idade por cidade**.

db.clientes.aggregate([{ $group: { \_id: "$cidade", mediaIdade: { $avg: "$idade" } } }]);

1. **Contagem por cidade** (usando $sum: 1 dentro de $group).

db.clientes.aggregate([{ $group: { \_id: "$cidade", contagem: { $sum: 1 } } }]);

1. **Média de pontosFidelidade** por cidade.

db.clientes.aggregate([{ $group: { \_id: "$cidade", mediaFidelidadePorCidade: { $avg: "$pontosFidelidade" } } }]);

**11) $project (projeção)**

**O que faz:** escolhe quais campos entram no resultado (ex.: mostrar só nome e idade sem \_id).

**Tarefas**

1. Mostre apenas **nome** e **idade** (oculte \_id).

db.clientes.aggregate([{ $project: { \_id: 0, nome: 1, idade: 1 } }]);

1. Mostre **nome**, **cidade** e **pontosFidelidade** (sem \_id).

db.clientes.aggregate([{ $project: { \_id: 0, nome: 1, cidade: 1, pontosFidelidade: 1 } }]);

1. Liste apenas **nome** (sem \_id).

db.clientes.aggregate([{ $project: { \_id: 0, nome: 1, } }]);

1. Mostre **nome** e **interesses** (sem \_id).

db.clientes.aggregate([{ $project: { \_id: 0, nome: 1, interesses: 1} }]);