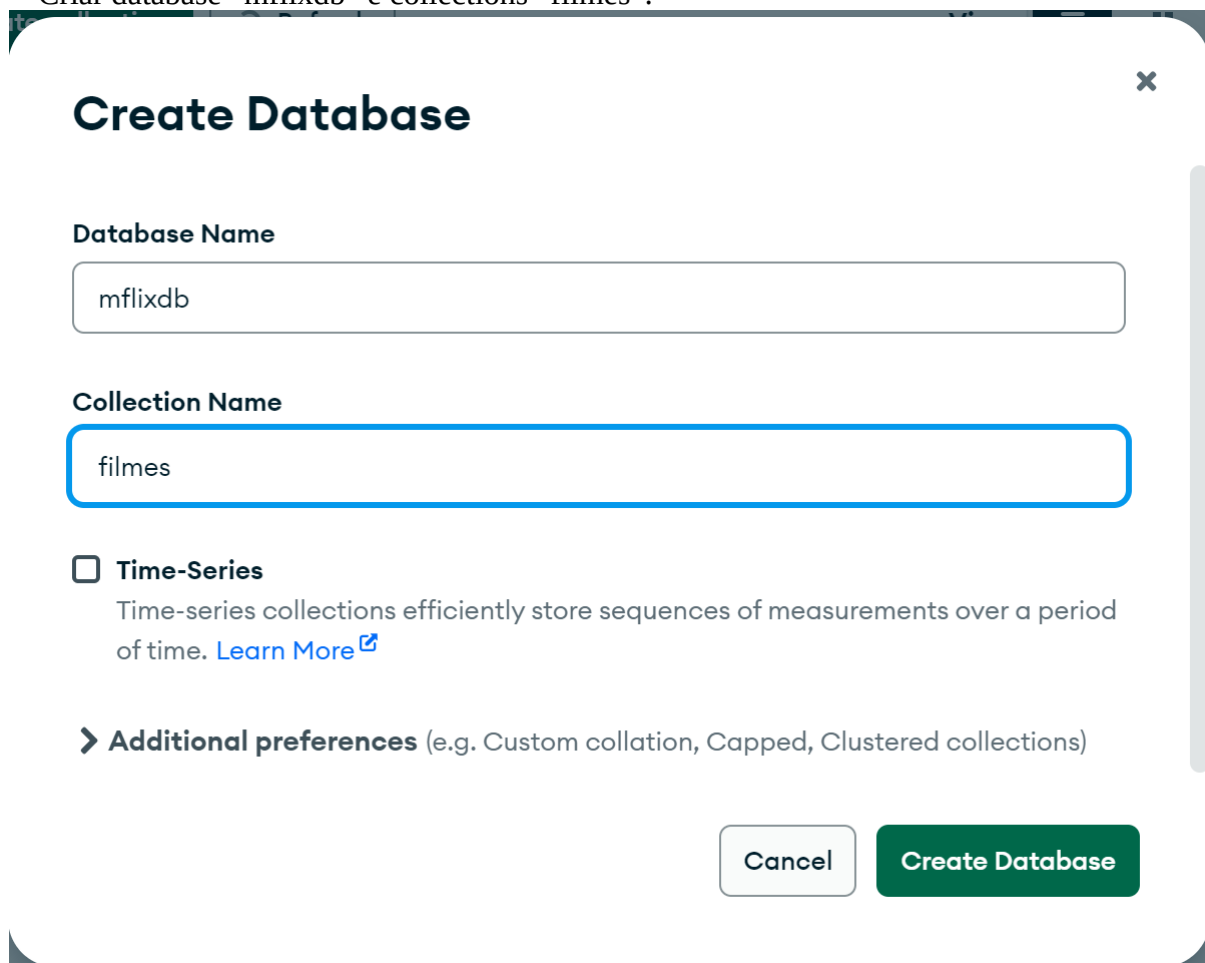


Trabalho Prático

Aula Prática 1.

Check List :

- Importando documentos.
 - Criar database “mflixdb” e collections “filmes”.



Create Database

Database Name

mflixdb

Collection Name

filmes

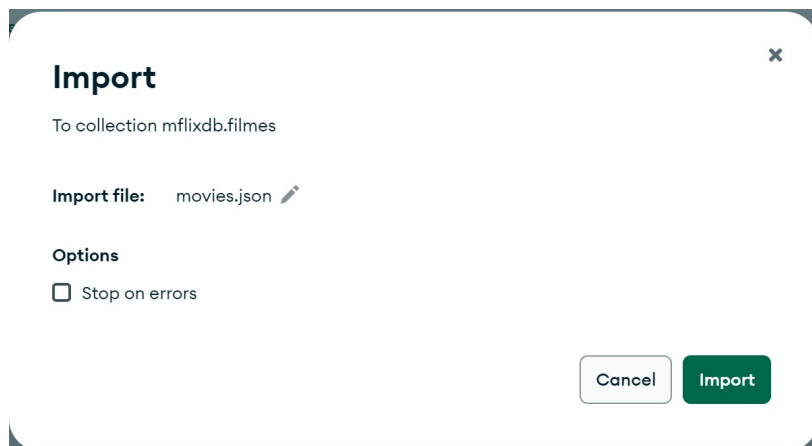
☐ **Time-Series**

Time-series collections efficiently store sequences of measurements over a period of time. [Learn More](#)

➤ **Additional preferences** (e.g. Custom collation, Capped, Clustered collections)

Cancel Create Database

- Importar o arquivo “movies.json”.



Import

To collection mflixdb.filmes

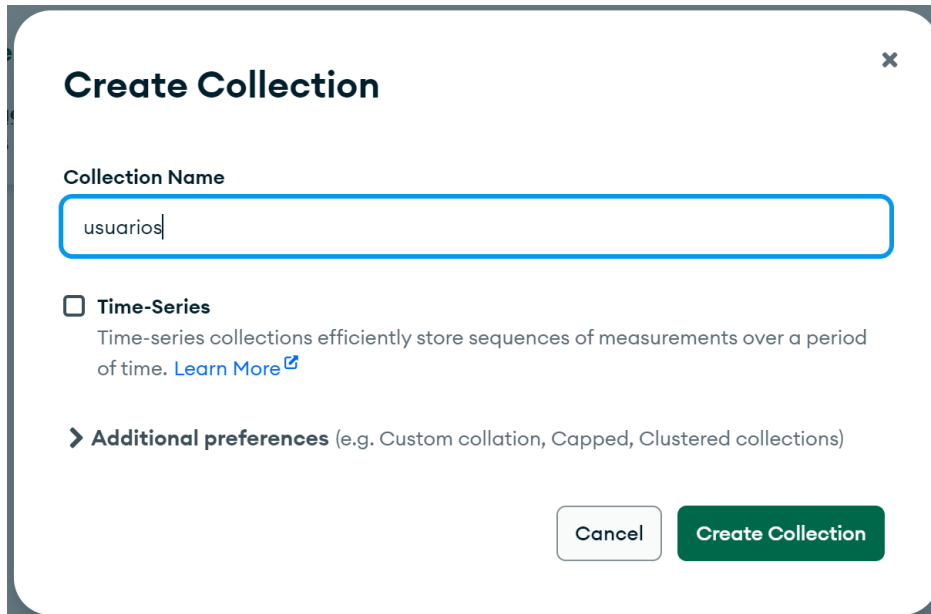
Import file: movies.json

Options

☐ Stop on errors

Cancel Import

- Criar collections “usuarios”.



Create Collection

Collection Name

usuarios

☐ **Time-Series**
Time-series collections efficiently store sequences of measurements over a period of time. [Learn More](#)

➤ **Additional preferences** (e.g. Custom collation, Capped, Clustered collections)

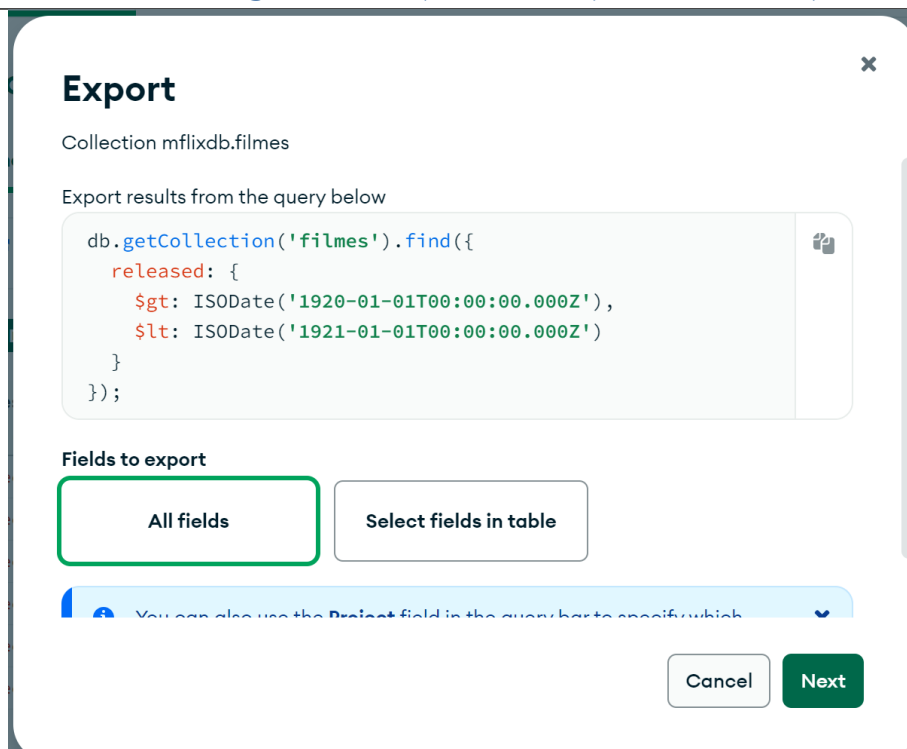
Cancel Create Collection

- Importar o arquivo “users.json”.

- Exportando documentos.

- Na collections “filmes” exportar todos os filmes lançados em 1920.

FILTER : { "released": { \$gt: new Date('1920-01-01') , \$lt: new Date('1921-01-01')} }



Export

Collection mflixdb.filmes

Export results from the query below

```
db.getCollection('filmes').find({
  released: {
    $gt: ISODate('1920-01-01T00:00:00.000Z'),
    $lt: ISODate('1921-01-01T00:00:00.000Z')
  }
});
```

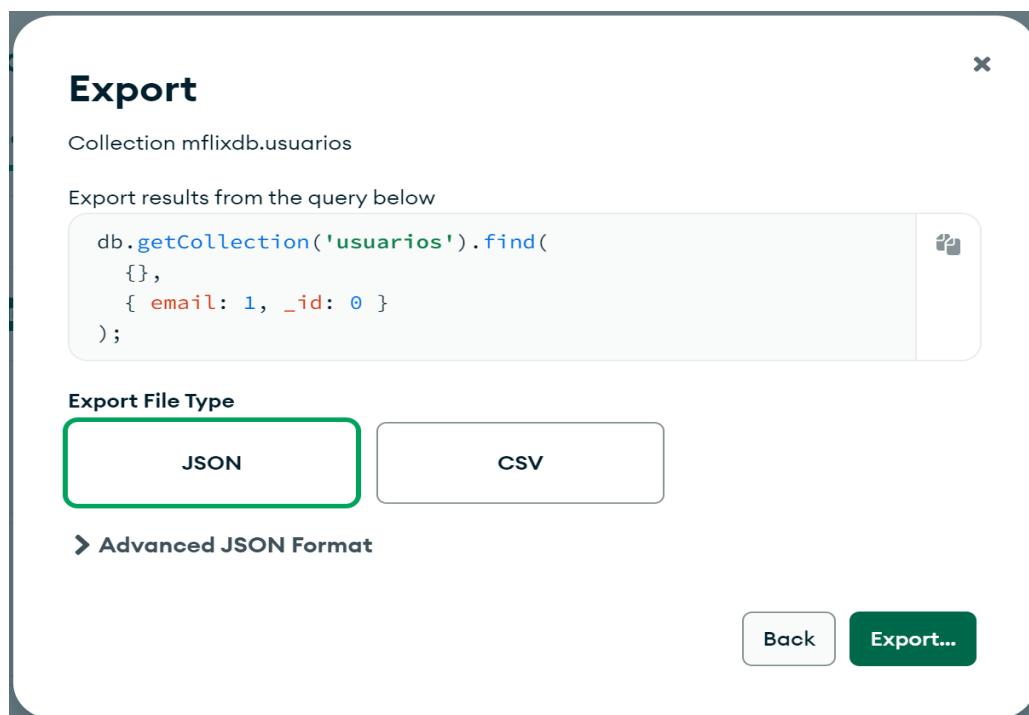
Fields to export

All fields Select fields in table

You can also use the Project field in the query bar to specify which

Cancel Next

- Na collections “usuarios” exportar apenas os e-mails de todos os usuários cadastrados.



Export ✕

Collection mflixdb.usuarios

Export results from the query below

```
db.getCollection('usuarios').find(
  {},
  { email: 1, _id: 0 }
);
```

Export File Type

☒ JSON ☐ CSV

[➤ Advanced JSON Format](#)

Entregar :

- Arquivo em formato JSON contendo os filmes lançados no ano 1920.

<https://github.com/ramiro-ebac-2022/ampli-engenharia-software/blob/main/Engenharia%20de%20Software/5%20semestre/BD%20n%C3%A3o%20relacionais/Aula%20Pratica/1.mflixdb.filmes%5Bfilmes%20lan%C3%A7ados%20em%201920%5D.json>

- Arquivo em formato JSON contendo os e-mails dos usuários cadastrados.

<https://github.com/ramiro-ebac-2022/ampli-engenharia-software/blob/main/Engenharia%20de%20Software/5%20semestre/BD%20n%C3%A3o%20relacionais/Aula%20Pratica/1.mflixdb.usuarios%5Bapenas%20o%20e-mail%5D.json>

Aula Prática 2.**Check List :**

- Criar banco de dados “lojadb”.
- Criar collections “vendas”.

Create Database

Database Name

lojadb

Collection Name

vendas

☐ Time-Series
Time-series collections efficiently store sequences of measurements over a period of time. [Learn More](#)

> Additional preferences (e.g. Custom collation, Capped, Clustered collections)



Cancel

Create Database

- Inserir documentos na collections criada.

Insert Document

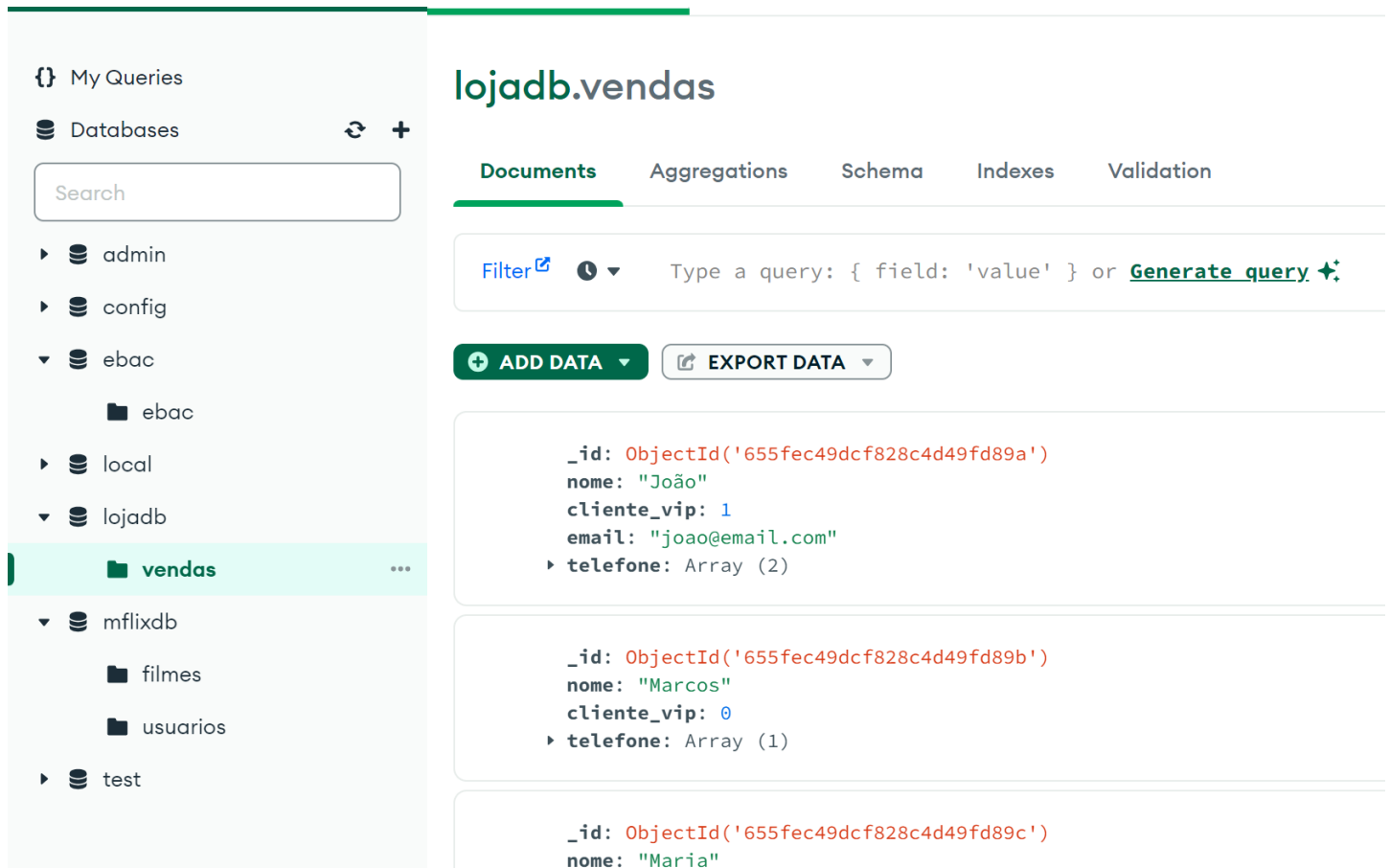
To collection lojadb.vendas

VIEW  

```
1  /**
2  * Dados básicos dos clientes
3  */
4  [
5  {
6    "nome" : "João",
7    "cliente_vip" : 1,
8    "email" : "joao@email.com",
9    "telefone" : ["9999-1111", "8888-1111"]
10  },
11  {
12    "nome" : "Marcos",
13    "cliente_vip" : 0,
14    "telefone" : ["9999-3333"]
15  },
16  {
```

Cancel

Insert



The screenshot shows the MongoDB Ampli web interface. On the left is a sidebar with a 'My Queries' section and a 'Databases' section. Under 'Databases', there is a search bar and a list of databases: 'admin', 'config', 'ebac' (expanded to show a folder 'ebac'), 'local', 'lojadb' (expanded to show a folder 'vendas' which is selected), 'mflixdb' (expanded to show 'filmes' and 'usuarios'), and 'test'. The main area is titled 'lojadb.vendas' and has tabs for 'Documents', 'Aggregations', 'Schema', 'Indexes', and 'Validation'. The 'Documents' tab is active, showing a list of documents. At the top of the document list are buttons for 'Filter', a clock icon, and a dropdown arrow, followed by the text 'Type a query: { field: 'value' } or [Generate query](#)'. Below this are buttons for 'ADD DATA' and 'EXPORT DATA'. The document list contains three entries, each with a JSON object:

```
{
  "_id": ObjectId('655fec49dcf828c4d49fd89a'),
  "nome": "João",
  "cliente_vip": 1,
  "email": "joao@email.com",
  "telefone": Array (2)
}
```

```
{
  "_id": ObjectId('655fec49dcf828c4d49fd89b'),
  "nome": "Marcos",
  "cliente_vip": 0,
  "telefone": Array (1)
}
```

```
{
  "_id": ObjectId('655fec49dcf828c4d49fd89c'),
  "nome": "Maria"
}
```

Entregar :

- Arquivo em formato JSON contendo os 3 documentos criados na collections “vendas”.

<https://github.com/ramiro-ebac-2022/ampli-engenharia-software/blob/main/Engenharia%20de%20Software/5%20semestre/BD%20n%C3%A3o%20relacionais/Aula%20Pratica/2.lojadb.vendas%5Bclientes%5D.json>

Aula Prática 3.**Check List :**

- Atualizar a collections “vendas”
 - Incluir o Endereço de cada cliente, como um campo de tipo “objeto”.
 - Adicionar os dados de compras de cada cliente, como “array de objetos”.

My Queries

Databases

Search

admin

config

ebac

ebac

local

lojadb

vendas

mflixdb

filmes

usuarios

test

lojadb.vendas

DocumentsAggregationsSchemaIndexesValidation

FilterType a query: { field: 'value' } or [Generate query](#)

ADD DATAEXPORT DATA

```
{
  "_id": ObjectId('655fec49dcf828c4d49fd89b'),
  "nome": "Marcos",
  "cliente_vip": 0,
  "telefone": Array (1),
  "Endereço": "Rua Dois, 4000. Campinas/SP",
  "Compras": Array (3)
    0: Object
      Nome do Produto: "Caderno"
      Preço: 20
      Quantidade: 1
    1: Object
      Nome do Produto: "Caneta"
      Preço: 3
      Quantidade: 5
    2: Object
      Nome do Produto: "Borracha"
```

Export

Collection lojadb.vendas

Export results from the query below

db.getCollection('vendas').find({});

Export File Type

JSON

CSV

> Advanced JSON Format

BackExport...

Entregar :

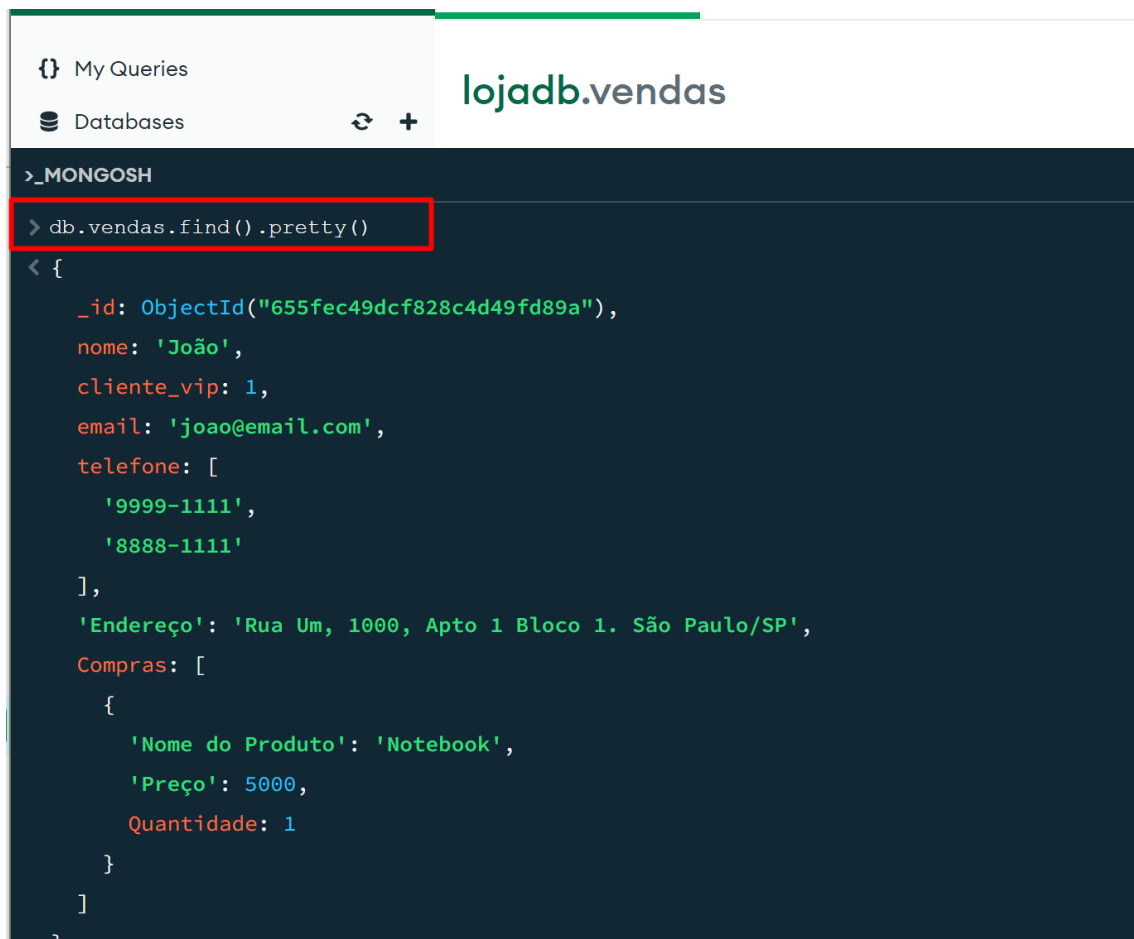
- Arquivo em formato JSON contendo os 3 objetos que foram atualizados na collections “vendas”.

<https://github.com/ramiro-ebac-2022/ampli-engenharia-software/blob/main/Engenharia%20de%20Software/5%20semestre/BD%20n%C3%A3o%20relacionais/Aula%20Pratica/3.lojadb.vendas%5Binclui%20endere%C3%A7o%20e%20compras%5D.json>

Aula Prática 4.**Check List :**

1. Realize uma consulta que retorne todos os documentos da collection.

db.vendas.find().pretty()



The screenshot shows a MongoDB CLI window with the title "lojadb.vendas". The command prompt is ">_MONGOSH". The command entered is "db.vendas.find().pretty()", which is highlighted with a red box. The output is a JSON document for a customer named João, including his ID, name, VIP status, email, phone numbers, address, and a list of purchases (one notebook for 5000).

```
>_MONGOSH
> db.vendas.find().pretty()
< {
  _id: ObjectId("655fec49dcf828c4d49fd89a"),
  nome: 'João',
  cliente_vip: 1,
  email: 'joao@email.com',
  telefone: [
    '9999-1111',
    '8888-1111'
  ],
  'Endereço': 'Rua Um, 1000, Apto 1 Bloco 1. São Paulo/SP',
  Compras: [
    {
      'Nome do Produto': 'Notebook',
      'Preço': 5000,
      Quantidade: 1
    }
  ]
}
```

2. Realize uma consulta que localize as informações da cliente “Maria”.

db.vendas.find(nome:"Maria")

lojadb.vendas

DO

Documents Aggregations Schema Indexes Validation

Filter

`{"nome": "Maria"}`



Generate query

Explain

Reset

Find

ADD DATA

EXPORT DATA

1 - 1 of 1

```
email: "maria@email.com"
  telephone: Array (3)
    0: "9999-2222"
    1: "8888-3333"
    2: "9988-0000"
  Endereço: "Rua Três, 3000. Londrina/PR"
  Compras: Array (2)
    0: Object
      Nome do Produto: "Tablet"
      Preço: 2500
      Quantidade: 1
    1: Object
      Nome do Produto: "Capa para Tablet"
      Preço: 50
      Quantidade: 1
```

3. Realize uma busca que retorna os clientes VIPs da loja (VIP = 1). Retorne apenas o campo “nome” de cada um.

```
db.vendas.find({cliente_vip:1},{nome:1}).pretty()
```

```
> _MONGOSH
<
> db.vendas.find({cliente_vip:1},{nome:1}).pretty()
< {
  _id: ObjectId("655fec49dcf828c4d49fd89a"),
  nome: 'João'
}
{
  _id: ObjectId("655fec49dcf828c4d49fd89c"),
  nome: 'Maria'
}
lojadb> |
```

4. Realize uma consulta que exiba as compras efetuadas por “Marcos”.

```
db.vendas.find({nome:"Marcos"},{Compras:1}).pretty()
```



```
>_MONGOSH
> db.vendas.find({nome:"Marcos"}, {Compras:1}).pretty()
< {
  _id: ObjectId("655fec49dcf828c4d49fd89b"),
  Compras: [
    {
      'Nome do Produto': 'Caderno',
      'Preço': 20,
      Quantidade: 1
    },
    {
      'Nome do Produto': 'Caneta',
      'Preço': 3,
      Quantidade: 5
    },
    {
      'Nome do Produto': 'Borracha',
      'Preço': 2,
      Quantidade: 2
    }
  ]
}
```

5. Realize uma consulta que retorne todos os nomes de produtos comprados por todos os clientes. *Nesta consulta em específico, utilize a linha de comando do MongoDB.

```
db.vendas.distinct("Compras.Nome do Produto")
```

```
> db.vendas.distinct("Compras.Nome do Produto")
< [
  'Borracha',
  'Caderno',
  'Caneta',
  'Capa para Tablet',
  'Notebook',
  'Tablet'
]
lojadb>
```