Bootcamp: Analista de Banco de Dados

Desafio Prático

Módulo 3: Banco de Dados NoSQL

Objetivos de Ensino

Exercitar os conceitos vistos em aulas em relação ao banco de dados NoSQL MongoDB. A partir de dados da Força Aérea Brasileira sobre a aviação civil Brasileira (CENIPA - Ocorrências Aeronáuticas na Aviação Civil Brasileira), vamos importar algumas informações no MongoDB para executar análises.

Enunciado

A base de dados de ocorrências aeronáuticas é gerenciada pelo Centro de Investigação e Prevenção de Acidentes Aeronáuticos (CENIPA). Constam nesta base de dados as ocorrências aeronáuticas notificadas ao CENIPA nos últimos 10 anos que ocorreram em solo brasileiro.

Dentre as informações disponíveis estão os dados sobre as aeronaves envolvidas, fatalidades, local, data, horário dos eventos e informações taxonômicas típicas das investigações de acidentes (AIG).

Arquivos com os quais trabalharemos:

- Ocorrencia.csv - Informações sobre as ocorrências.
- Ocorrencia_tipo.csv - Informações sobre o tipo de ocorrência.
- Aeronave.csv - Informações sobre as aeronaves envolvidas nas ocorrências.

Fonte: Sistema DÉDALO.

https://dados.gov.br/dataset/ocorrencias-aeronauticas-da-aviacao-civil-

brasileira

Alguns ajustes foram executados para facilitar nosso estudo, tais como eliminação de caracteres especiais, acentos e ajustes nos campos data e hora para facilitar a importação no MongoDB.

Etapas do trabalho:

a) Abrir o prompt de comando do MongoDB. Vamos criar o database e as collections por lá.

b) Criar o Database chamado “desafio”.

c) Criar as collections com validator:

d) Criar a collection “ocorrencia”:

db.ocorrencia.drop() – para o caso de necessitar rodar a criação novamente

```
db.createCollection("ocorrencia", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      properties: {
            id_ocorrencia: {
        bsonType: "int",
        description: "is not required"
      },
            classificacao: {
        bsonType: "string",
        description: "is not required"
      },
            cidade: {
        bsonType: "string",
        description: "is not required"
      },
            uf: {
        bsonType: "string",
        description: "is not required"
      },
            pais: {
        bsonType: "string",
        description: "is not required"
      },
            data: {
        bsonType: "date",
        description: "is not required"
      },
```

```
            num_recomendacoes: {
            bsonType: "int",
            description: "is not required"
          }
        }
      }
    }
})
```

e) Criar a collection “ocorrencia_tipo”

db.ocorrencia_tipo.drop() – para o caso de necessitar rodar a criação novamente

```
db.createCollection("ocorrencia_tipo", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      properties: {
            id_ocorrencia_t: {
          bsonType: "int",
          description: "is not required"
        },
            tipo: {
          bsonType: "string",
          description: "is not required"
        }
      }
    }
  }
})
```

f) Criar a collection “aeronave”

```
  db.aeronave.drop()

db.createCollection("aeronave", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["assentos", "ano_fabricacao"],
      properties: {
            id_ocorrencia_a: {
          bsonType: "int",
          description: "is not required"
        },
            matricula: {
          bsonType: "string",
          description: "is not required"
        },
            operador_categoria: {
          bsonType: "string"
        },
            tipo_veiculo: {
```

```
        bsonType: "string",
        description: "is not required"
      },
        fabricante: {
        bsonType: "string",
        description: "is not required"
      },
        modelo: {
        bsonType: "string",
        description: "is not required"
      },
        motor_tipo: {
        bsonType: "string"
      },
        motor_quantidade: {
        bsonType: "string"
      },
        assentos: {
        bsonType: "int",
        minimum: 1,
        maximum: 1000,
        description: "must be an integer in [ 1, 1000 ] and is required"
      },
        ano_fabricacao: {
        bsonType: "int",
        minimum: 1950,
        maximum: 2030,
        description: "must be an integer in [ 1950, 2030 ] and is required"
      },
        pais_fabricante: {
        bsonType: "string"
      },
        registro_segmento: {
        bsonType: "string"
      },
        voo_origem: {
        bsonType: "string"
      },
        voo_destino: {
        bsonType: "string"
      },
        fase_operacao: {
        bsonType: "string"
      }
    }
  }
})
```

g) Abrir o MongoDB Compass para fazer as importações dos dados.

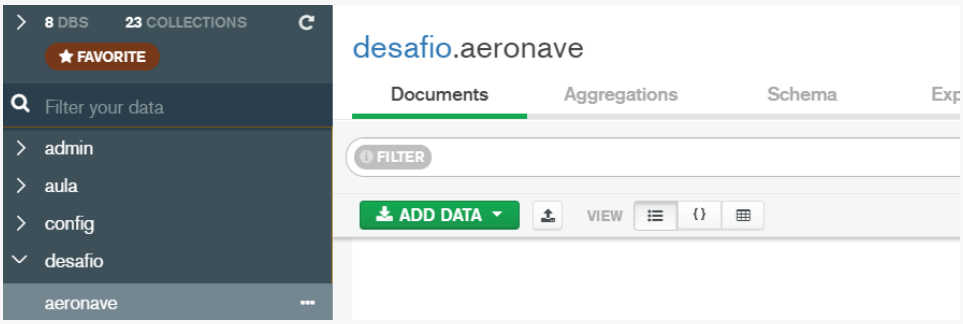
ATENÇÃO: se você editar e salvar o arquivo csv no excel, ele cria uma última linha em branco no arquivo que vai gerar uma indicação de erro na importação. Abra

os arquivos csv com o notepad (bloco de notas), por exemplo, e verifique a última linha. Se não estiver preenchida com dados, você deve apagá-la.

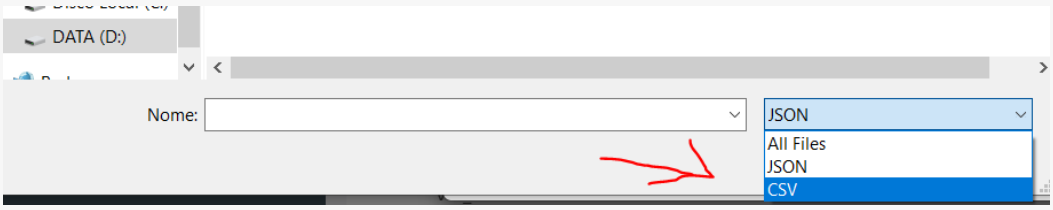
h) Carregar a collection “aeronave”:

Pelo MongoDB Compass, escolha o database Desafio e a collection “aeronave”.

Clique no botão “ADD DATA” para inserir os dados na collection.



Selecione o arquivo “aeronave.csv”. Repare que para o arquivo csv aparecer na seleção é necessário que você mude o tipo de arquivo.

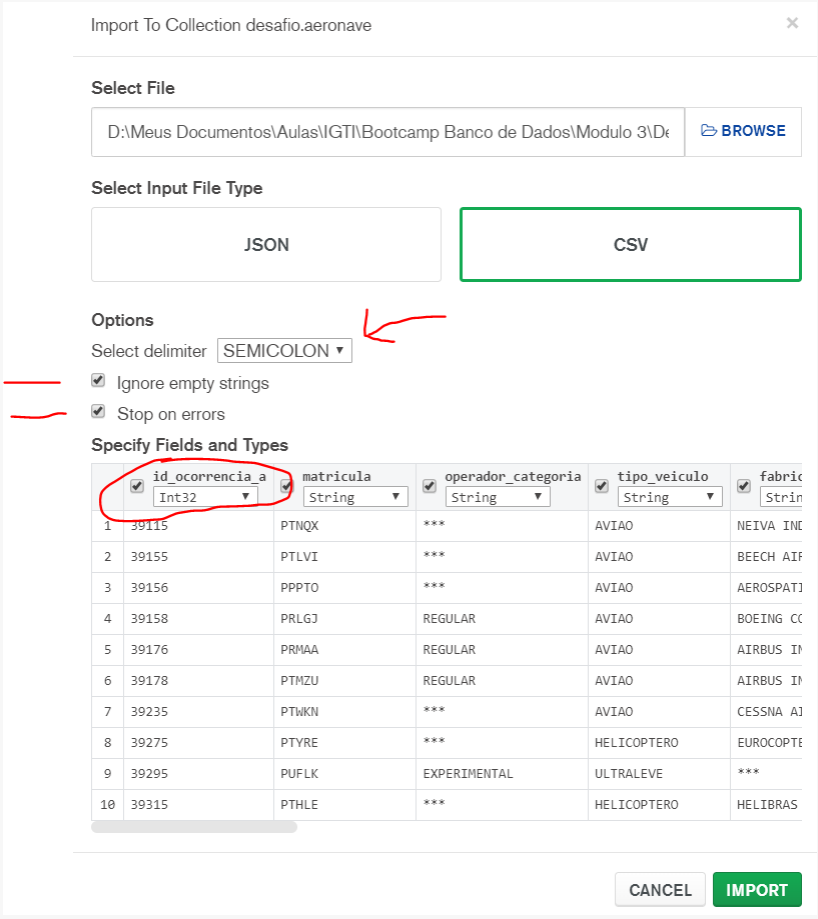


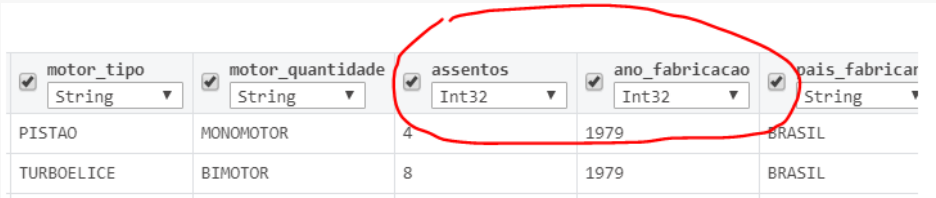
Selecione a opção de tipo “CSV”.

Selecione o delimitador “ponto e vírgula” (SEMICOLON).

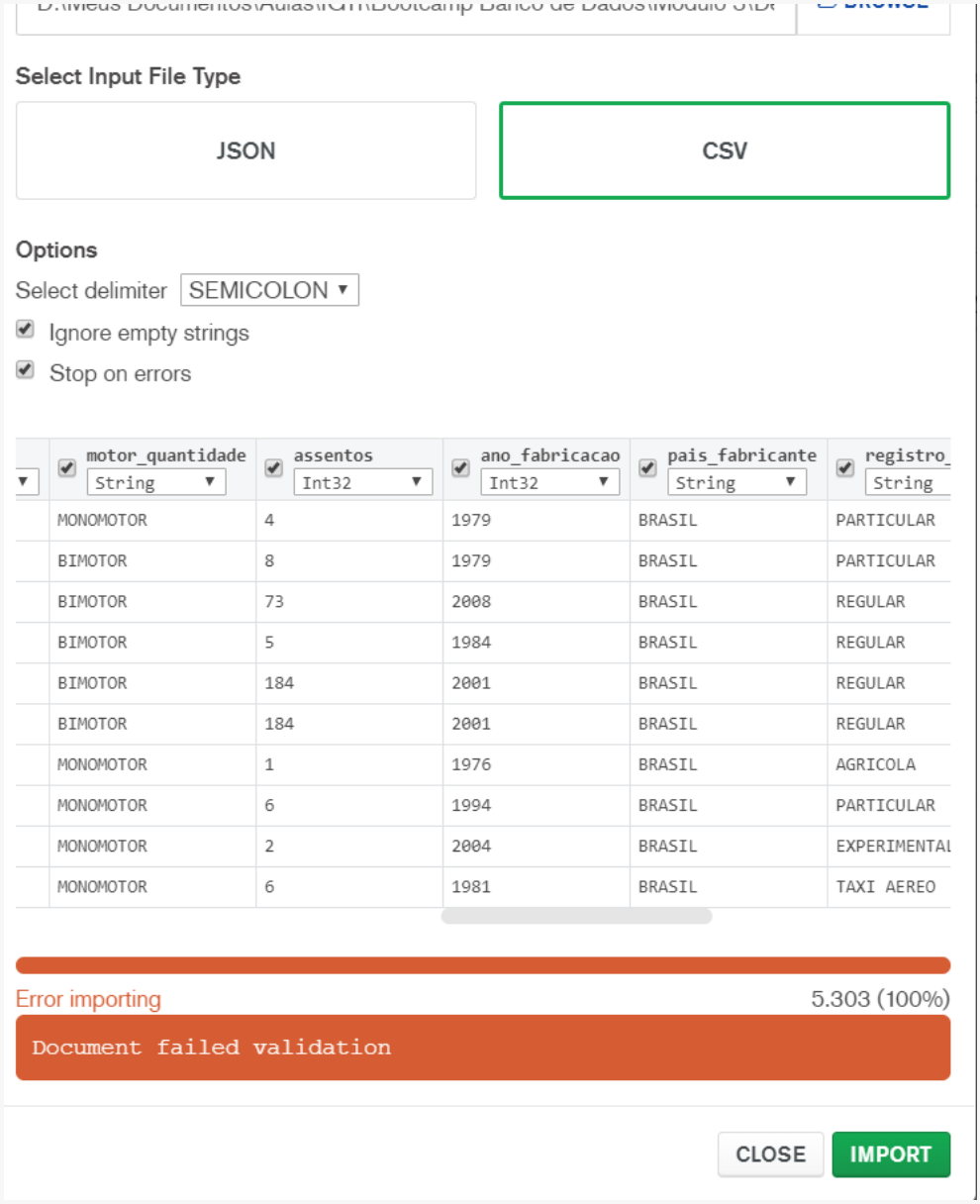
Não se esqueça de informar corretamente os tipos dos fields da collection.

- Id_ocorrencia_a, assentos, ano_fabricacao são fields do tipo Int32.
- Os demais são do tipo String.



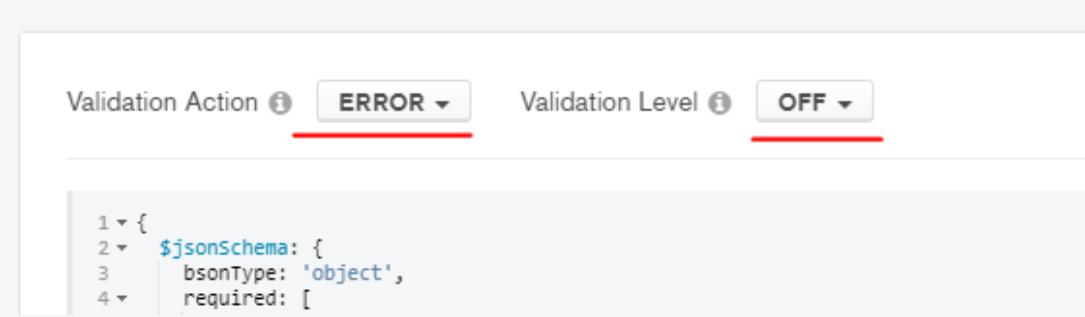


Ao executar a importação, você terá uma tela como a abaixo indicando erros na importação. Alguns documentos serão importados, outros não.

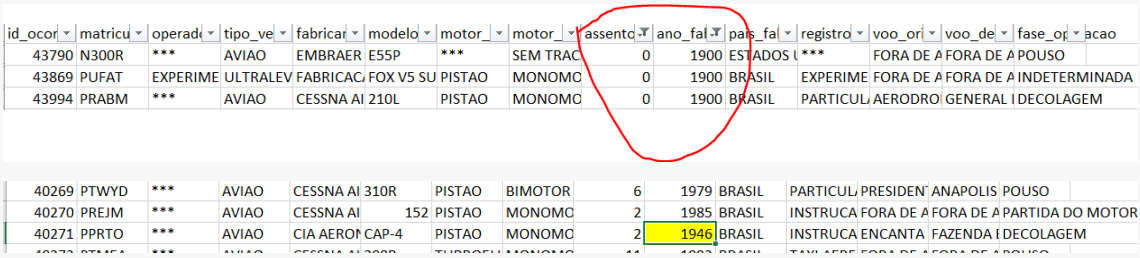


Por que isso acontece?

Verifique que por default as ações na validação e o nível de validação estão conforme imagem abaixo, portanto, se o flag “Stop on errors” estiver marcado, vai parar a carga dos documentos em caso de linhas que não se enquadrem na validação. Se o flag “Stop on errors” não estiver marcado, vai carregar apenas os documentos que passarem na validação.

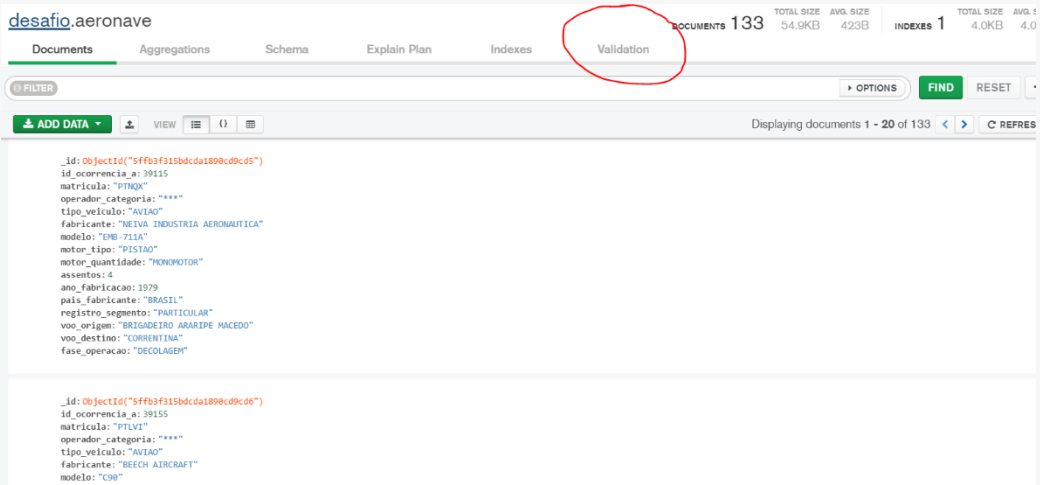


Se você abrir o arquivo com os dados “aeronave.csv”, verá que algumas informações estão sendo barradas pelo validator dos fields “assentos” e “ano_fabricacao”.

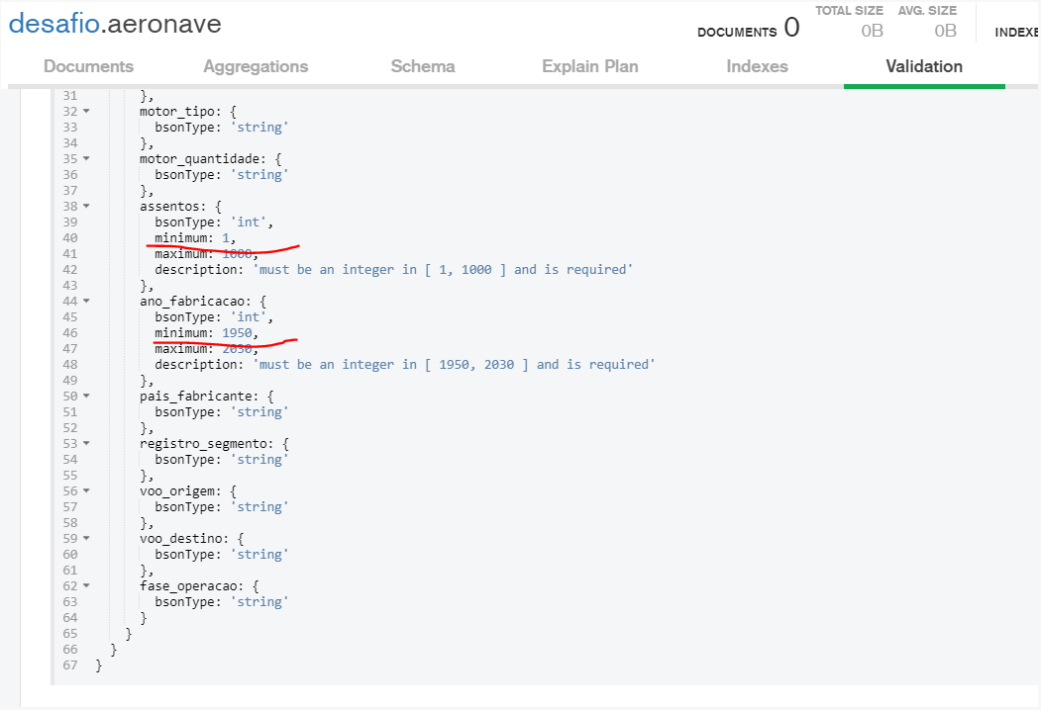


id_ocor	matricu	operad	tipo_ve	fabricar	modelo	motor_	motor_	assento	ano_fal	país_fa	registro	voo_or	voo_de	fase_o	acao
43790	N300R	***	AVIAO	EMBRAER	E55P	***	SEM TRAC	0	1900	ESTADOS U	***	FORA DE A	FORA DE A	POUSO	
43869	PUFAT	EXPERIME	ULTRALEV	FABRICAC	FOX V5 SU	PISTAO	MONOMO	0	1900	BRASIL	EXPERIME	FORA DE A	FORA DE A	INDETERMINADA	
43994	PRABM	***	AVIAO	CESSNA AI	210L	PISTAO	MONOMO	0	1900	BRASIL	PARTICUL	AERODRO	GENERAL I	DECOLAGEM	
40269	PTWYD	***	AVIAO	CESSNA AI	310R	PISTAO	BIMOTOR	6	1979	BRASIL	PARTICUL	PRESIDEN	ANAPOLIS	POUSO	
40270	PREJM	***	AVIAO	CESSNA AI	152	PISTAO	MONOMO	2	1985	BRASIL	INSTRUCA	FORA DE A	FORA DE A	PARTIDA DO MOTOR	
40271	PPRTO	***	AVIAO	CIA AERON	CAP-4	PISTAO	MONOMO	2	1946	BRASIL	INSTRUCA	ENCANTA	FAZENDA I	DECOLAGEM	

Para verificar isso, você deve clicar na aba validation.



Os limites da validação estão barrando a importação de alguns documentos.



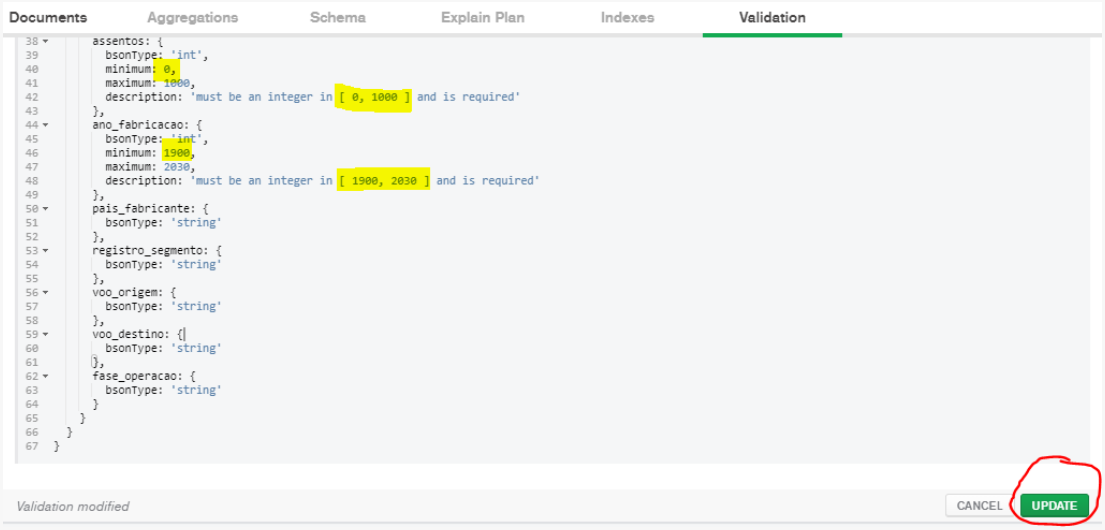
Para acertar isso, você precisa alterar os limites mínimos de validação dos fields assentos e ano_fabricacao.

Rode novamente o comando de criação com as validações corrigidas pelo prompt de comando ou faça isso pela própria aba Validation no MongoDB Compass.

Prompt de comando:

```
    assentos: {
    bsonType: "int",
    minimum: 0,
    maximum: 1000,
    description: "must be an integer in [ 0, 1000 ] and is required"
},
    ano_fabricacao: {
    bsonType: "int",
    minimum: 1900,
    maximum: 2030,
    description: "must be an integer in [ 1900, 2030 ] and is required"
},
```

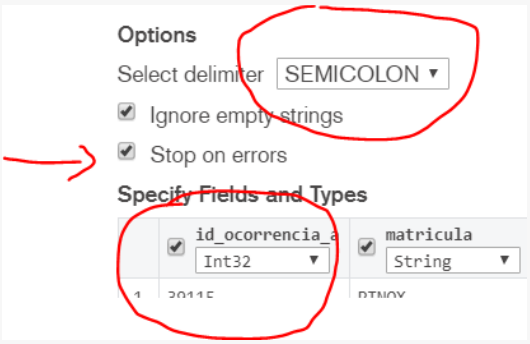
Ou aba Validation no MongoDB Compass:



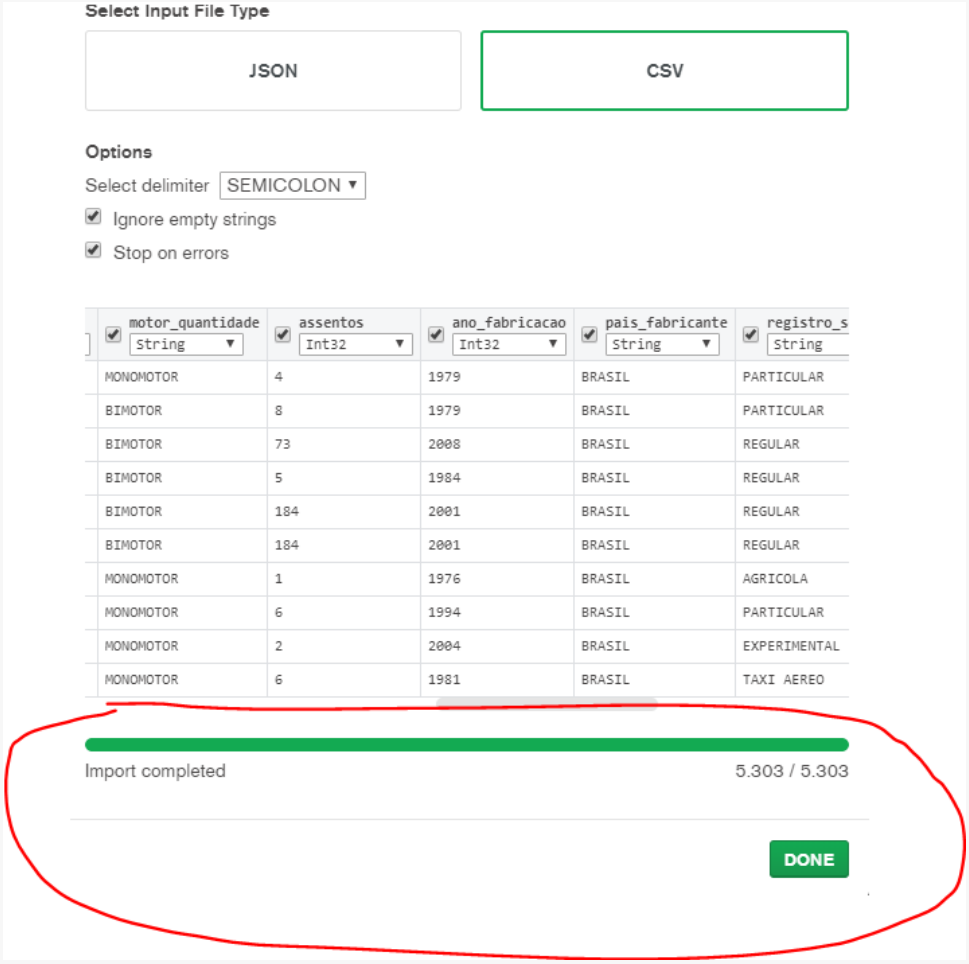
Faça a importação novamente, mas antes limpe a tabela, porque alguns documentos foram importados na última tentativa.

db.aeronave.remove({})

Ao executar a nova importação, lembre-se de alterar os tipos dos campos.



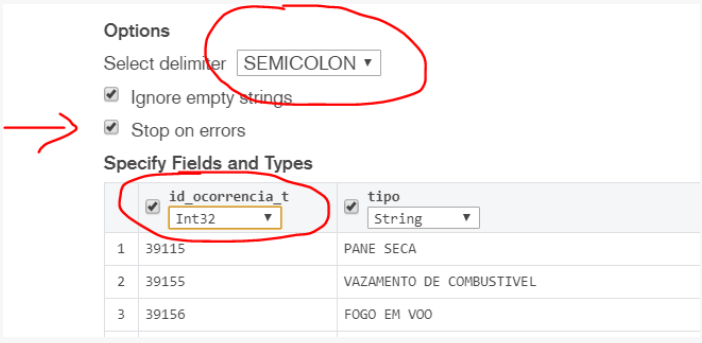
A mensagem será Import completed em verde.



i) Carregar a collection “ocorrencia_tipo”.

Não se esqueça de informar corretamente os tipos dos fields da collection.

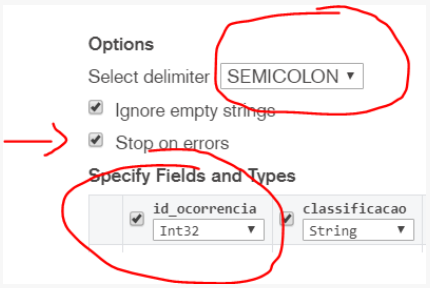
- Id_ocorrencia_t é do tipo Int32.
- Os demais são do tipo String.

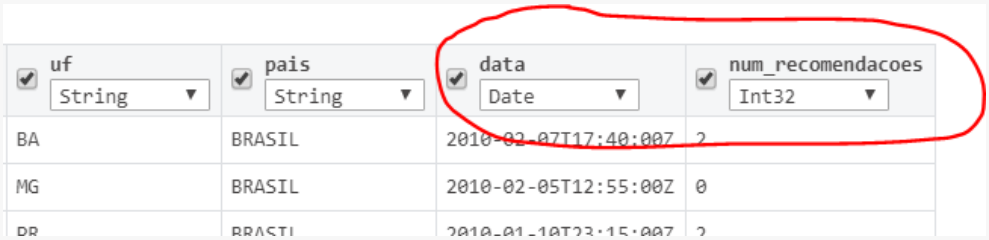


j) Carregar a collection “ocorrencia”.

Não se esqueça de informar corretamente os tipos dos fields da collection.

- Id_ocorrencia e num_recomendacoes são do tipo Int32.
- Data é do tipo date.
- Os demais são do tipo String.





Atividades

Execute os comandos das práticas abaixo e anote os resultados.

1. Verifique o número de documentos carregados na collection “ocorrencia”. Você pode usar a função count() ou db.collection.aggregate com {$sum:1}.
2. Verifique o número de documentos carregados na collection “ocorrencia_tipo”. Você pode usar a função count() ou db.collection.aggregate com {$sum:1}.
3. Verifique o número de documentos carregados na collection “aeronave”. Você pode usar a função count() ou db.collection.aggregate com {$sum:1}.
4. Execute um comando find() na collection aeronave com modelo= 'AB-115' OU tipo_veiculo = 'AVIAO'.

db.collection.find({ [$or: [{ , }]})

Limite a consulta para trazer apenas os 5 primeiros documentos.

5. Execute um comando find() na collection aeronave onde o tipo_veiculo não são os seguintes tipos: ['AVIAO', 'HELICOPTERO', 'HIDROAVIAO', 'PLANADOR', 'ANFIBIO']

Limite o resultado da consulta para trazer apenas os 10 primeiros documentos.

A dica é que podemos usar a condição IN para retornar apenas valores que estão dentro de uma lista e o NOT IN para retornar os valores que não estão dentro de uma lista.

6. Execute um comando aggregate() na collection aeronave para agrupar os documentos pelo campo tipo_veiculo fazendo uma contagem ($sum:1) para cada tipo_veiculo.

7. Execute um comando find() na collection ocorrencia para buscar os documentos com o field num_recomendacoes menor ou igual ($lte) a 2 (<=2). Limite o resultado da sua consulta para 10 documentos.

8. Execute um comando aggregate() na collection ocorrencia para buscar os documentos conforme abaixo.

    - Field num_recomendacoes menor ou igual ($lte) a 2 (<=2).
    - Agrupar pelo field uf ($uf)
    - Fazer a contagem ($sum:1)
    - Ordenar de forma descendente

Dica:

```
db.ocorrencia.aggregate([
    { $match: { },
    { $group: { },
    { $sort:   { }
  ])
```

9. Execute um comando aggregate() na collection ocorrencia para buscar a média dos números de recomendações ($num_recomendacoes).

10. Execute um comando lookup aggregate() na collection aeronave fazendo uma junção com a collection ocorrencia. Limite o resultado do lookup aggregate() em dois documentos para facilitar a visão do que acontece.

Dica:

```
db.aeronave.aggregate([
 {
   $lookup:
    {
      from: "ocorrencia",
      localField:   (" "),
      foreignField: (" "),
```

```
        as: "ocorrencia_aeronave"
      }
  },
  {
    $limit: 2
  }
]).pretty()
```

11. Execute um comando lookup aggregate() na collection ocorrencia fazendo uma junção com a collection ocorrencia_tipo. Limite o resultado do lookup aggregate() em dois documentos para facilitar a visão do que acontece.

Chegamos ao final da prática do nosso desafio.