Bootcamp: Arquiteto(a) de Big Data

Desafio

Módulo 3	Persistência em Banco de Dados NoSQL

Objetivos

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 e 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. Disponível em: 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, ajuste no campo data e campo 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:

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"
      }
    }
  }
 }
})
```

d) 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"
      }
    }
  }
 }
})
```

e) 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"
        }
      }
    }
  }
})
```

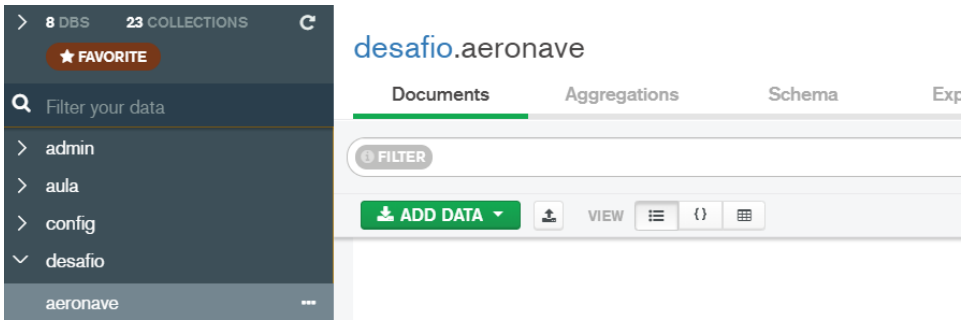
f) Abrir o MongoDB Compass para fazer as importações dos dados:

ATENÇÃO: se você editar e salvar o arquivo csv no excel ele cria uma última linha em branco no arquivo o 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.

g) 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 collecion.



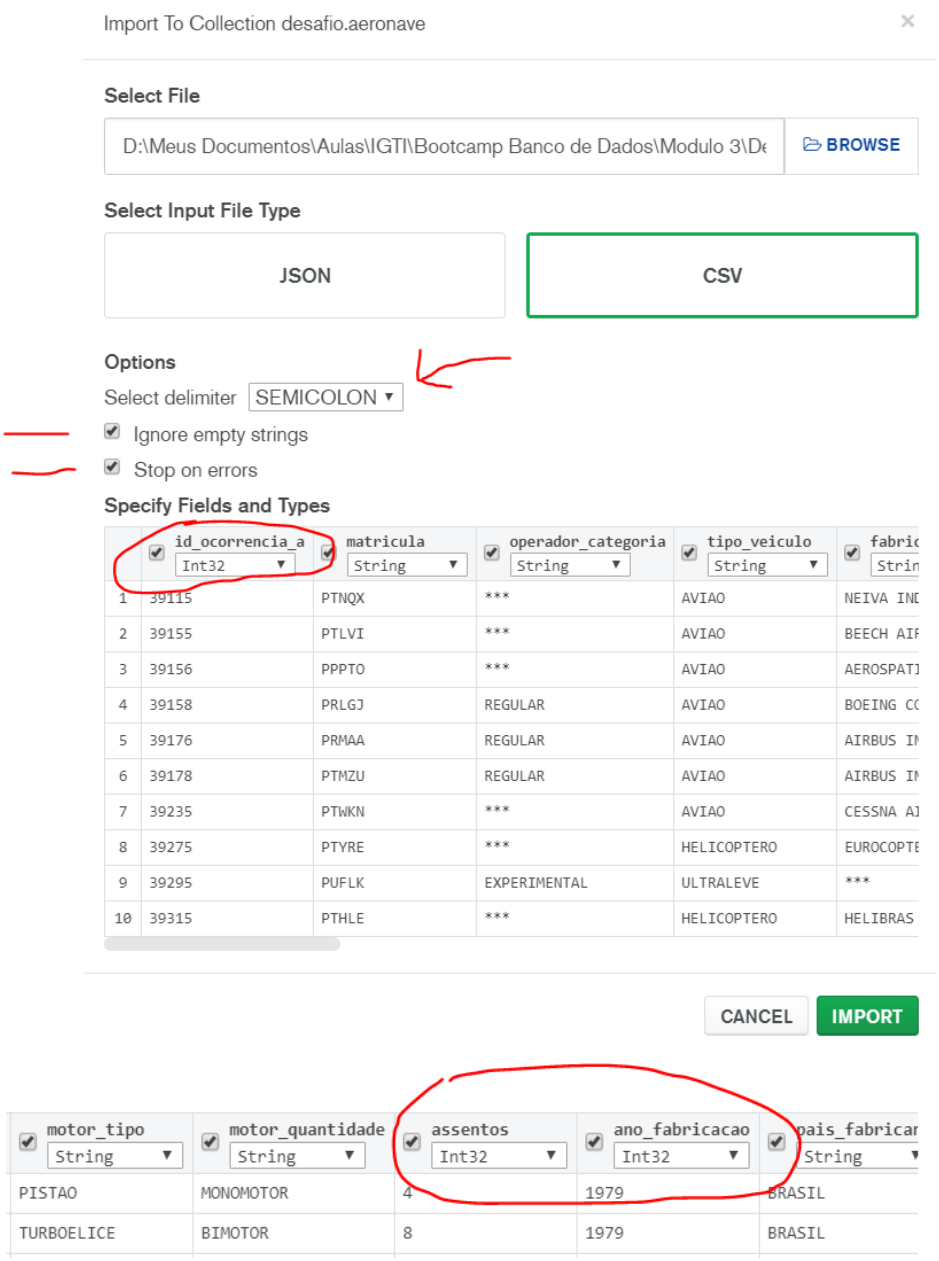
Selecione o arquivo "aeronave.csv". Repare que para o arquivo csv aparecer na seleção é necessário você mudar o tipo de arquivo.

Selecione a opção de tipo "CSV".

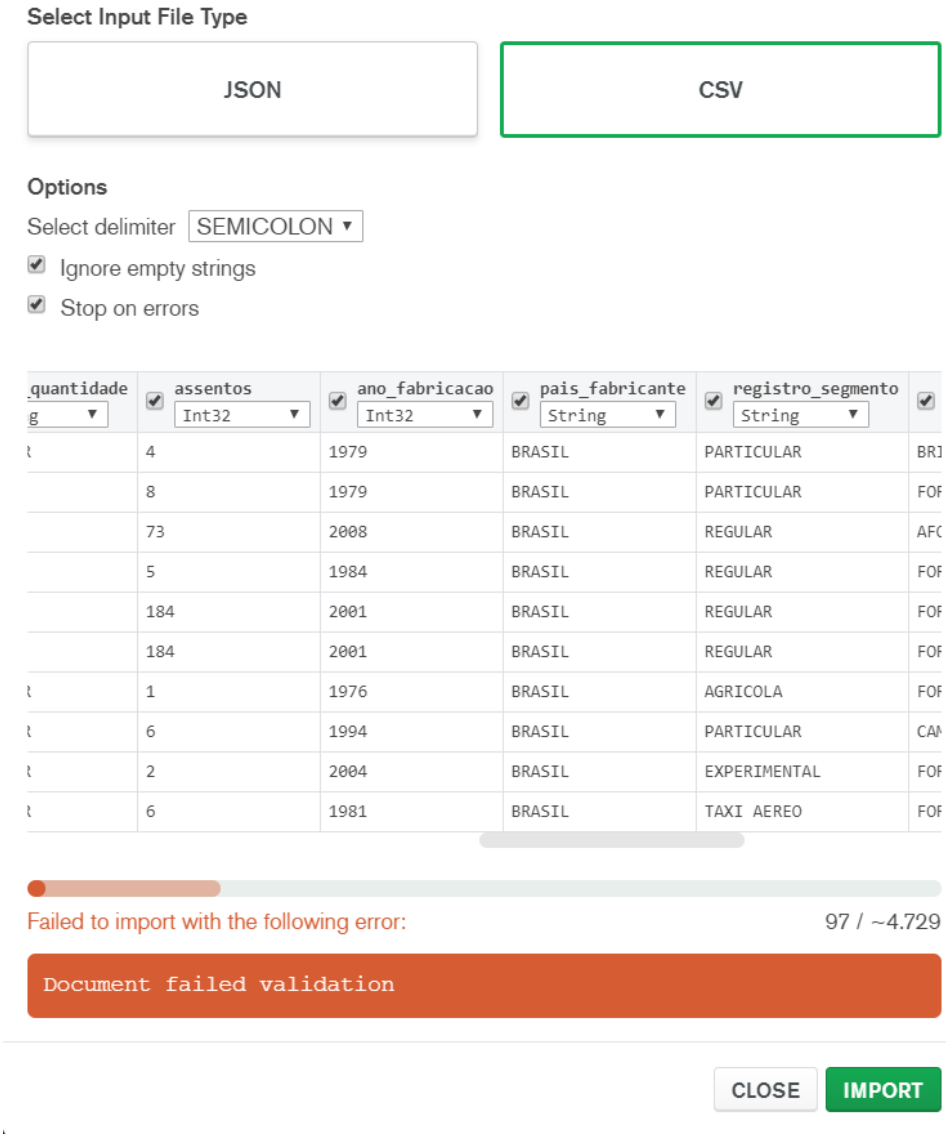
Selecione o delimitador "ponto e vírgula" (SEMICOLON).

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**.

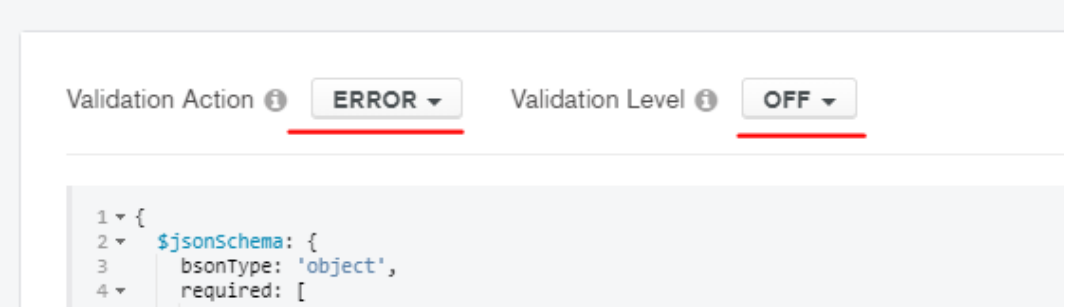


Ao executar a importação, você terá uma tela como a abaixo indicando erros na importação. A importação termina com a mensagem “Document failed validation”. Observe que marcamos o flag: “Stop on erros”.

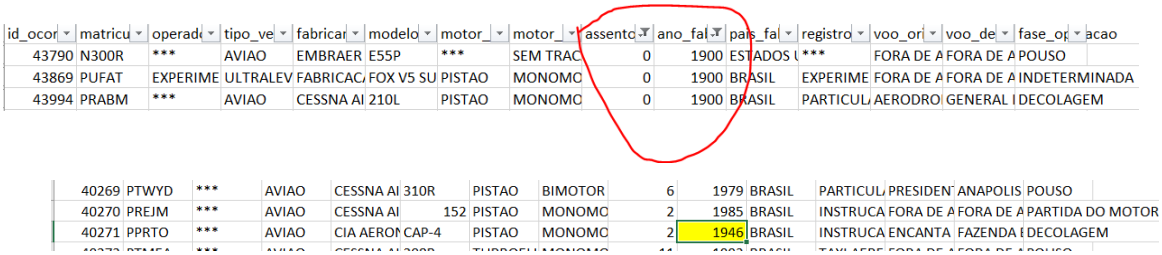


Por que isso acontece?

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.

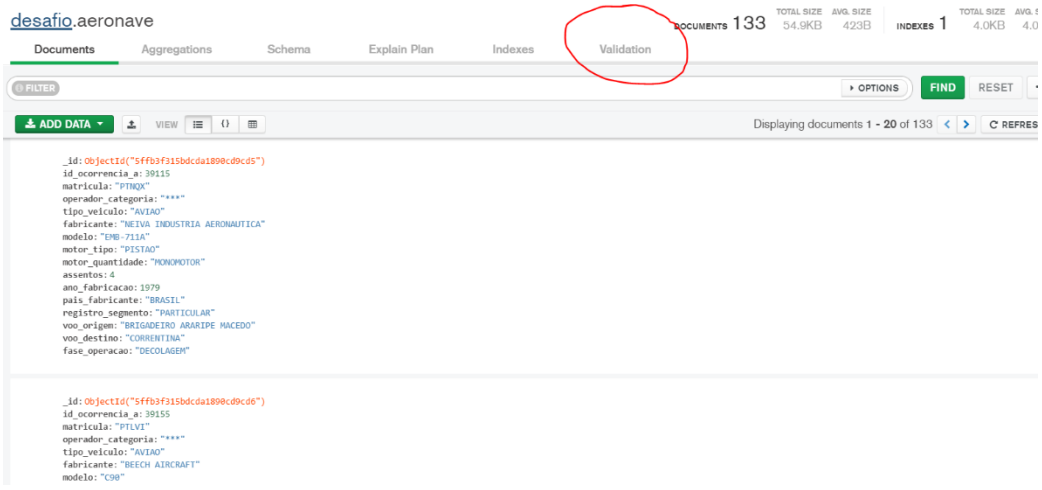


Se você abrir o arquivo com os dados “aeronave.csv” verá que algumas informações estão sendo barradas pelo validator dos fields “assentos” e “ano_fabricacao”.

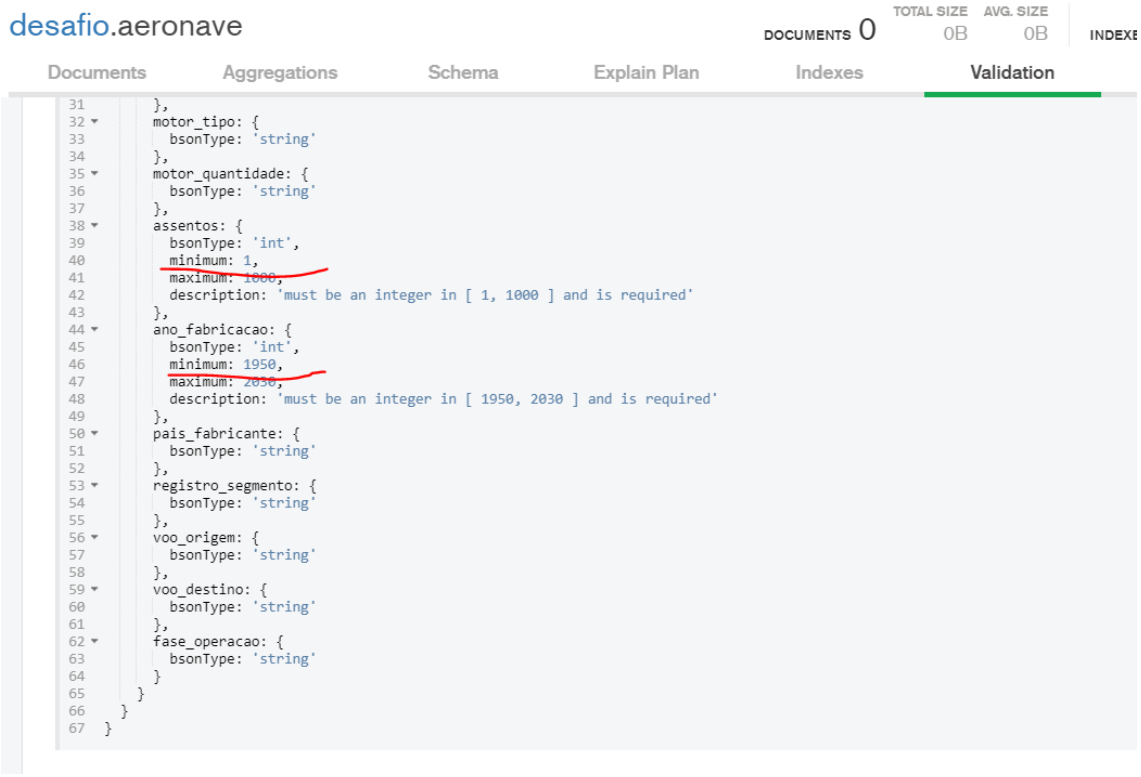


id_ocor	matricu	operad	tipo_ve	fabricar	modelo	motor_	motor_	assento	ano_fal	pais_fa	registro	voo_or	voo_de	fase_o	acao
43790	N300R	***	AVIAO	EMBRAER	E55P	***	SEM TRAC	0	1900	ESTADOS	***	FORA DE A	FORA DE A	POUSO	
43869	PUFAT	EXPERIME	ULTRALEV	FABRICAC	FOX V5 SU	PISTAO	MONOMO	0	1900	BRASIL	EXPERIME	FORA DE A	FORA DE A	INDETERMINADA	
43994	PRABM	***	AVIAO	CESSNA AI	210L	PISTAO	MONOMO	0	1900	BRASIL	PARTICUL	AERODRO	GENERAL	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	DECOLAGEM	

Para verificar isso, ainda no MongoDB Compass 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 no terminal (prompt de comando) o comando de criação da collection Aeronave com as validações corrigidas 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"
},
```

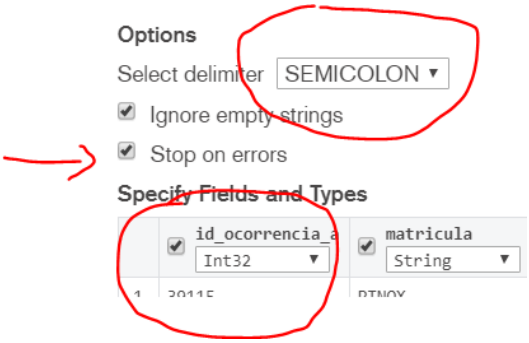
Ou aba Validation no MongoDB Compass:

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

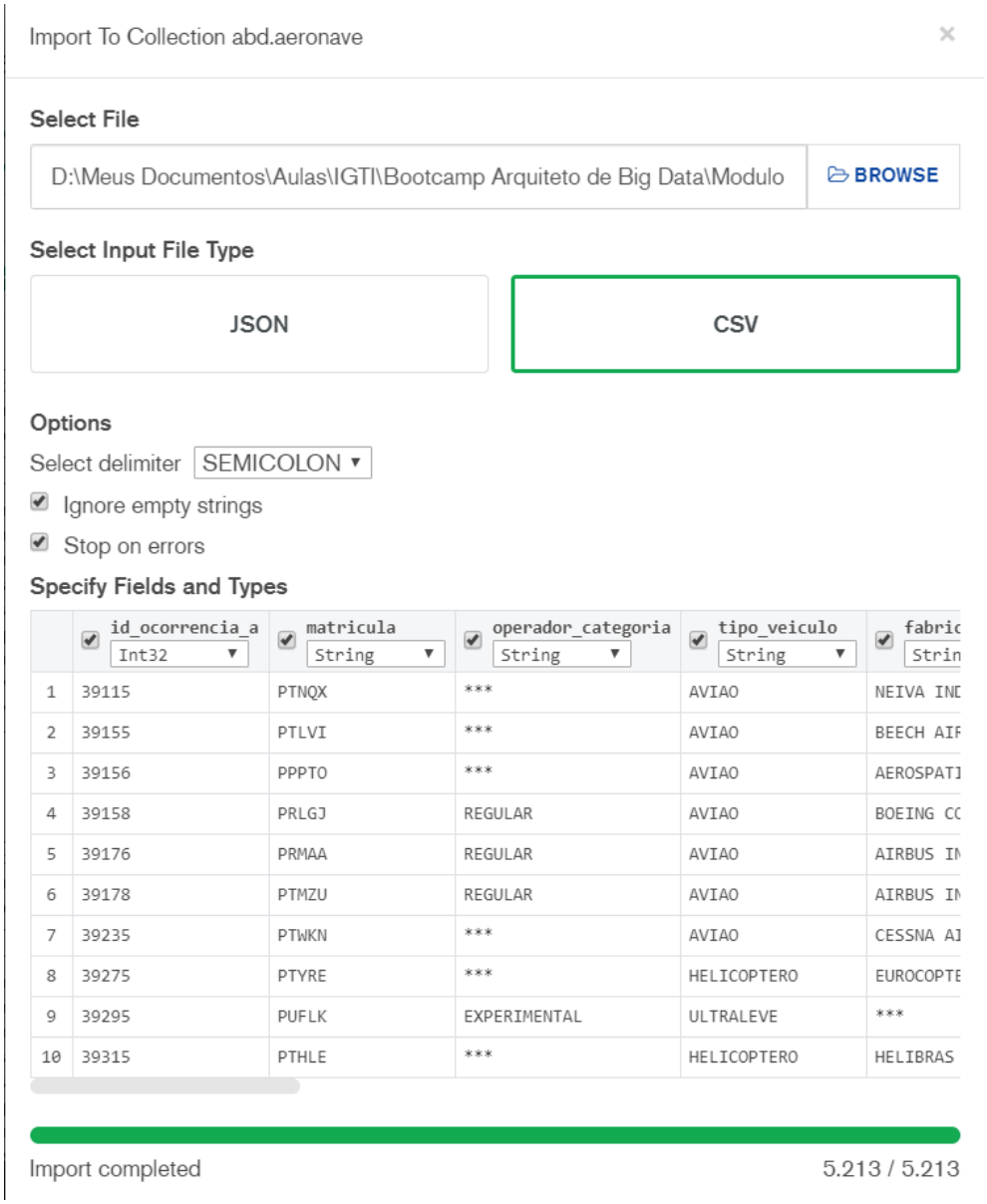
db.aeronave.remove({})

Atenção, anote a quantidade de documentos removidos, pois fará parte da nossa prática.

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



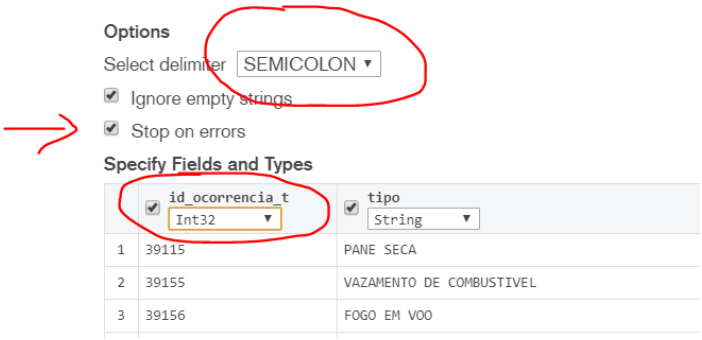
A mensagem será **Import completed** em verde.



h) Carregar a collection “ocorrencia_tipo”

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

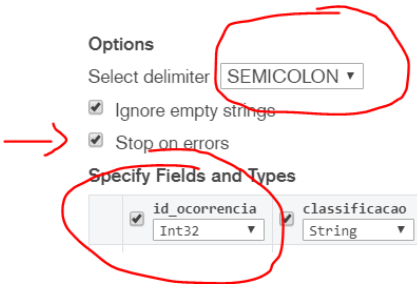
- Id_ocorrencia_t é do tipo **Int32**.
- Os demais são do tipo S**tring**.

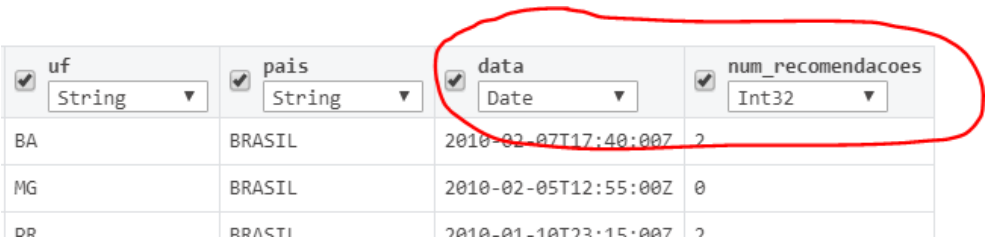


i) Carregar a collection "ocorrencia"

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**.





Práticas: **execute os comandos das práticas abaixo e anote os resultados, pois você vai precisar deles ao responder às questões objetivas.**

1) Antes de fazer a importação novamente na collection Aeronave, você precisou limpar os dados que forma importados anteriormente. Você executou o comando ***db.aeronave.remove({}).***

   Quantos documentos foram removidos?

2) **Verifique o número de documentos carregados na collection "ocorrencia".**

Você pode usar a função count() ou db.collection.aggregate com {$sum:1}.

3) **Verifique o número de documentos carregados na collection "ocorrencia_tipo".**

Você pode usar a função count() ou db.collection.aggregate com {$sum:1}.

4) **Verifique o número de documentos carregados na collection "aeronave".**

Você pode usar a função count() ou db.collection.aggregate com {$sum:1}.

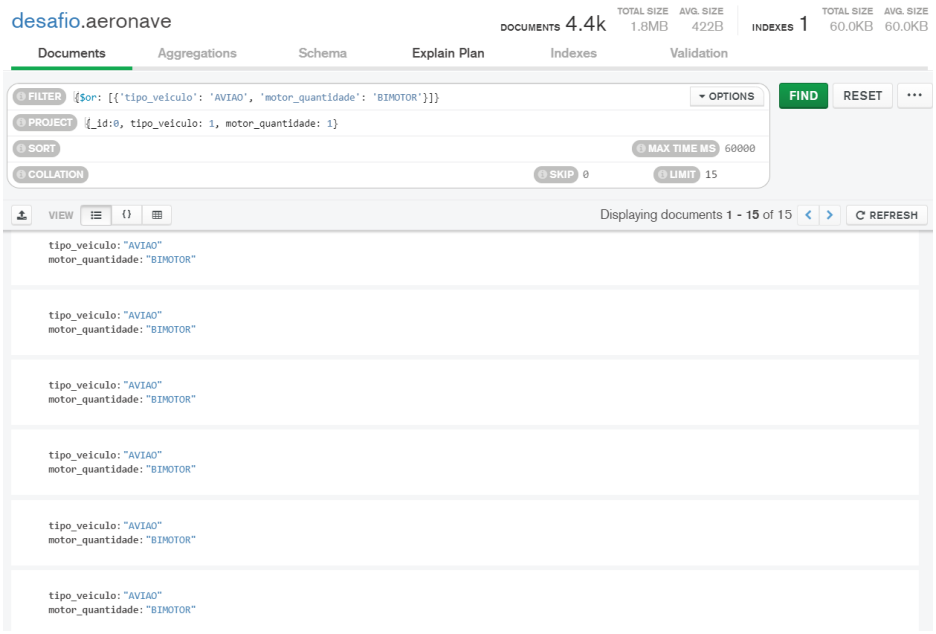
5) **Execute um comando find() na collection aeronave com modelo= 'AB-115' OU tipo_veiculo = 'AVIAO'.**

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

Limite a consulta para trazer apenas os 5 primeiros documentos.

{_id:0, matricula: 1}).limit(5).pretty()

Exemplo dos comandos no MongoDB Compass



6) **Execute um comando find() na collection aeronave onde o tipo_veiculo não são os seguintes tipos:**

['AVIAO', 'HELICOPTERO', 'HIDROAVIAO', 'PLANADOR', 'ULTRALEVE']

Limite o resultado da consulta para trazer apenas os 8 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.

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

8) **Execute um comando find() na collection ocorrencia para buscar os documentos com o field num_recomendacoes menor ou igual ($lte) a 5 (<=5). Limite o resultado da sua consulta em 10 documentos. Anote o maior valor para a recomendação encontrada.**

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

- Field num_recomendacoes menor ou igual ($lte) a 5 (<=5).
- Agrupar pelo field uf ($uf).
- Fazer a contagem ($sum:1).
- Ordenar de forma descendente

Dica:

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

Observe que, no retorno da cosulta, ela pode vir com paginação. Você precisa digitar "it" para ver mais resultados.

```
Type "it" for more
MongoDB Enterprise > it
```

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

11) **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()
```

12) 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.

13) Vamos analisar a criação de índices na collection ocorrencia_tipo.

Execute os comandos abaixos e analise os resultados.

db.aeronave.find({voo_destino: "SERRA DO SOL"}).explain();

db.aeronave.find({voo_destino: "SERRA DO SOL"}).explain(true).executionStats;

Em ambos, observe o resultado para o parâmetro “stage”, que será COLLSCAN e o número de documentos examinados ("totalDocsExamined") foi 5213.
Observe também o parâmetro "executionTimeMillis", que no meu caso foi 5 milisegundos.

Agora vamos criar um índice para voo_destino, ascendente.

Db.aeronave.createIndex({voo_destino: 1})

Execute novamente os comandos abaixo e analise os resultados.

db.aeronave.find({voo_destino: "SERRA DO SOL"}).explain();

db.aeronave.find({voo_destino: "SERRA DO SOL"}).explain(true).executionStats;

Observe o resultado para o parâmetro “stage”, que agora será será FETCH e o número de documentos examinados ("totalDocsExamined") foi apenas 1.
O parâmetro "executionTimeMillis", no meu caso retornou agora 1 milisegundo.

Se quiser remover o índice é só executar o comando abaixo.

db.aeronave.dropIndex({voo_destino: 1})

Chegamos ao final da prática do nosso Desafio.

De acordo com os resultados encontrados, responda às questões objetivas.

Respostas Finais

1.	Pelo uso correto da função aggregate, podemos agrupar os documentos por um determinado campo (field). Como fica a visualização do agrupamento se utilizarmos {$sum:1}?
x	Os documentos serão agrupados pelo campo informado e será exibida a CONTAGEM dos documentos.
	Os documentos serão agrupados pelo campo informado e será exibida a MÉDIA (AVG) dos documentos.
	Os documentos serão apenas listados e não serão agrupados por causa do uso do {$sum:1}.
	Os documentos serão agrupados pelo campo informado e será exibido o somatório das datas de criação dos documentos.

2.	Pela PRÁTICA 1, antes de fazer a importação novamente na collection aeronave, você precisou limpar essa collection. Quantos documentos foram removidos?
x	97.
	86.
	99.
	54.

3.	Pelas PRÁTICAS 2 a 4, quais foram os valores para os totais de documentos carregados (importados dos arquivos csv) respectivamente nas collections “ocorrencia” e “aeronave”?
x	5155, 5213.
	5355 e 6250.
	4250, 9355.
	5337 e 5397.

4. Pela PRÁTICA 5, executamos o comando find() na collection aeronave com a restrição modelo= “AB-115” OU tipo_veiculo = “AVIAO”. Quais foram as matrículas dos dois primeiros documentos listados?	
x	PTNQX e PTLVI.
	PPPTO e PPVOB.
	PPTLS e PRLGJ.
	PPTLS e PPTMA.

5. Pela PRÁTICA 6 executamos o comando find() na collection aeronave observando o campo tipo_veiculo. **Verificando não os 8, mas os 20 primeiros documentos** retornados, qual foi o “tipo_veiculo” mais listado? **QUESTÃO CORRIGIDA.**	
x	ANFIBIO.
	DIRIGIVEL.
	ULTRALEVE.
	HIDROPLANO.

6. Sobre manipulação de datas no MongoDB, É CORRETO AFIRMAR que:	
x	O construtor ISODate () retorna um objeto Date.
	O construtor ISOTIMESTAMP () retorna um objeto Date.
	O construtor ISONOWSQL () retorna um objeto Date.
	Não existe o tipo Date no MongoDB.

7. Pela PRÁTICA 7, executamos o comando aggregate() na collection aeronave para agrupar os documentos pelo campo “tipo_veiculo” fazendo uma contagem ($sum:1) para cada “tipo_veiculo”. Qual foi a contagem de documentos (número retornado) para os tipos de veículos DIRIGIVEL E ULTRALEVE, respectivamente?	
x	2 e 307.
	4 e 307.

	4 e 600.
	76 e 1.

8.	Pela PRÁTICA 8, executamos o comando find() na collection ocorrencia para buscar os documentos com o campo num_recomendacoes menor ou igual ($lte) a 5 (<=5). Considerando os 10 primeiros documentos retornados, qual foi o valor MÁXIMO que apareceu para os números de recomendações (num_recomendacoes)?
x	3.
	5.
	2.
	4.

9.	Pela PRÁTICA 9, executamos o comando aggregate() na collection ocorrencia. Qual o total da contagem que apareceu para o estado “RN”?
x	14.
	9.
	27.
	67.

10.	Pela PRÁTICA 9, executamos o comando aggregate() na collection ocorrencia. Qual o estado (campo uf) que obteve uma maior contagem, ou seja, tem mais documentos?
x	“SP” com 1211 documentos.
	“RJ” com 1465 documentos.
	“RS” com 860 documentos.
	“ES” com 1123 documentos.

11. Pela PRÁTICA 10, executamos o comando aggregate() na collection ocorrencia para buscar a média dos números de recomendações ($num_recomendacoes). Qual foi o valor aproximado obtido como média?	
x	0,3259.
	1,2103.
	0,5541.
	0,6187.

12. Pela PRÁTICA 11, onde executamos o comando lookup aggregate() na collection aeronave fazendo uma junção com a collection ocorrência, é CORRETO AFIRMAR:	
x	Foram listados os documentos das collections de aeronave e ocorrencia, sendo que os documentos da collection ocorrencia apareceram de forma “embedded”, ou seja, documento dentro de documento.
	Foram listados os documentos das collections de ocorrencia_tipo e ocorrencia, sendo que os documentos da collection ocorrencia_tipo apareceram de forma “embedded”, ou seja, documento dentro de documento.
	Foram listados os documentos apenas da collection ocorrencia_tipo.
	Foram listados os documentos das collections de ocorrencia_tipo e ocorrencia de forma alternada.

13. Pela PRÁTICA 11, onde executamos o comando lookup aggregate() na collection aeronave fazendo uma junção com a collection ocorrencia é CORRETO AFIRMAR:	
x	A junção se deu pelos campos (fields) "id_ocorrencia_a" e "id_ocorrencia".
	O comando não faz junção entre collections.
	A junção se deu pelos campos (fields) "ocorrencia_aeronave " e "id_ocorrencia".
	A junção se deu pelos campos (fields) "voo_origem" e "voo_destino".

	14. Pela PRÁTICA 12, onde executamos um comando lookup aggregate(), é CORRETO AFIRMAR que no primeiro documento retornado o valor do campo (field) “cidade” foi:
x	“CORRENTINA”.
	“SAO ROQUE”.
	“PATO BRANCO”.
	“SAO JOSE DOS CAMPOS”.

	15. NÃO É CORRETO afirmar que pelo MongoDB Compass:
x	É possível executar comandos SQL em tabelas no MongoDB.
	É possível executar consultas em collections.
	É possível criar databases e collections.
	É possível alterar as regras de validação de uma collection.