MongoDB

https://ibm-learning.udemy.com/course/mongodb-the-complete-developers-guide/learn/practice/1060372/instructor-solution#overview

- **OBS:** Palavras em destaque com cor devem ser substituídas corretamente, de acordo com o que queira realizar.

cls	limpar o cmd
show dbs	visualizar bancos de dados já criados
use NOMEBD	criar banco de dados
db.dropDatabase()	excluir um banco de dados (primeiro devemos acessar o banco de dados desejado)
db.nomecolecao.insertOne([{}]) ou **db.nomecolecao.insertMany([{}])**	incluir uma coleção no banco de dados desejado (primeiro devemos acessar o banco de dados desejado)
db.nomecolecao.find().pretty()	visualizar dados dentro de uma coleção
$set	serve pra modificar algum dado dentro da coleção, e caso o dado á ser modificado não exista, ele será criado
$gt	seleciona documentos maior que (>)
db.nomecolecao.deleteMany({"": ""})	deleta um dado dentro de uma coleção
db.nomecolecao.drop()	deleta a coleção inteira
db.stats()	método utilitário fornecido pelo shell que gera algumas estatísticas sobre esse banco de dados
db.runCommand()	para executar alguns comandos administrativos
collMod	modificador de coleção
$eq	condição de igualdade (é = a)

$	alterar/modificar os dados
$inc	atualização
$ne	valores que não são iguais a um valor especificado.
$lt	valores inferiores a um valor especificado
$lte	valores menores ou iguais a um valor especificado
$in	valores especificados em uma matriz
$nin	não corresponde a nenhum dos valores especificados em uma matriz (ou seja, quero encontrar todos os valores que não são TAL valor)
.count()	identifica a quantidade de itens que vc deseja e optou por ver nas listas
$or	mostra apenas opções que você deseja visualizar, precisa colocar a condição antes e depois definir dois ou mais parametros que vc deseja separados por vírgula - ***ex**: db.movies.find({$or: [{"rating.average": {$lt: 5}}, {"rating.average": {$gt: 9.3}}]}).pretty()* - e aqui só serão exibidos os valores menores que 5 ou valores maiores que 9.3
$nor	ao contrário da condição $or - ***ex**: db.movies.find({$nor: [{"rating.average": {$lt: 5}}, {"rating.average": {$gt: 9.3}}]}).pretty()* - aqui será exibido valores MAIORES que 5 (mesmo que a condição seja $lt, definimos o $nor no inicio) ou mostrará valores menores que 9.3
$and	visualiza valores combinados - ***ex**: db.movies.find({$and: [{"rating.average": {$gt: 9}}, {genres: "Drama"}]}).pretty()* - aqui só mostrará os documentos que são Drama e que possuem nota maior que 9 - ESCREVER ASSIM também da na mesma: *db.movies.find({"rating.average": {$gt: 9}, genres: "Drama"}).pretty()* - o $and só é mais útil quando queremos valores iguais, ex: *db.movies.find({$and: [{genres: "Horror"}, {genres: "Drama"}]}).pretty()*
$not	inverte o efeito de uma expressão de consulta - ex procurar algo que não exista. - ex *db.movies.find({runtime: {**$not: {$eq:** 60}}}).count()* - aqui está buscando o tempo de execução que **NÃO é igual** á 60. / ou podemos usar diretamente somente uma condição: ex - *db.movies.find({runtime: {$ne: 60}}).count()*

$exists	apresenta apenas o que existe - *db.users.find({age: {$exists: false}}).pretty()* - nesse exemplo só irá mostrar os usuários que não possuem o campo idade em seu documento, pois definimos como falso. Ao definir como verdadeiro, mostrará todos os documentos que possuem idade "age".
$type	mostrará os documentos de acordo com o tipo que vc definir - https://docs.mongodb.com/manual/reference/operator/query/type/ - db.users.find({phone: {$type: "number"}}).pretty()
$regex	maneira de procurar no texto certos padrões - *db.movies.find({summary: {$regex: /musical/}}).pretty()*
$expr	operador de expressão, ele pega um documento que descreve a expressão - *db.sales.find({$expr: {$gt: ["$volume", "$target"]}}).pretty()* - comparando se o volume é maior que o destino. **ou podemos usar condições:** *db.sales.find({$expr: {$gt: [{$cond: {if: {$gte: ["$volume", 190]}, then: {$subtract: ["$volume", 30]}, else: "$volume" }}, "$target"]}}).pretty()*
$size	buscar pelo tamanho uma determinada informação no documento
$all	pesquisa gênero por essas palavras-chaves e garantirá que esses itens existam no gênero e que podem ser números, documentos incorporados, outras matrizes... - ***ex***: *db.moviestarts.find({genre: {$all: ["action", "thriller"]}}).pretty()*

Exemplos:

- **INSERINDO COLEÇÕES**

Inserindo <u>somente UMA coleção</u> dentro de um banco de dados:

1. use hospital
2. db.patients.insert([{name: "Amanda", age: 17, hight: 151, problem: [{appointment: "yes", return: "no", emergency: "no"}]}])

Inserindo uma <u>coleção múltipla</u> dentro de um banco de dados:

1. use hospital
2. db.patients.insertMany([{name: "Clara", age: 27, problem: [{appointment: "no", return: "yes", emergency: "no"}]}, {name: "Roberto", age: 23, problem: [{appointment: "yes", return: "no", emergency: "no"}]}])

- **USANDO $SET**

Modificando a idade e o problema do Roberto.

1. use hospital
2. db.patients.find().pretty()
3. db.patients.updateOne({name: "Roberto"}, {$set: {age: 22, problem: [{appointment: "no", return: "yes", emergency: "no"}]}})

Inserindo um novo dado dentro da coleção.

1. use hospital
2. db.patients.find().pretty()
3. db.patients.updateOne({name: "Roberto"}, {$set: {doctor: "William Yeszk"}})

- **USANDO $GT**

Esse exemplo mostrará todos os pacientes que possuem idade maior que 25.

1. use hospital
2. db.patients.find({age: {$gt: 25}}).pretty()

- **DELETANDO DADOS**

Deletando dados de uma coleção.

OBS: Foi selecionado apenas um dado dentro da coleção ROBERTO e ela foi excluída completamente.

1. use hospital
2. db.patients.deleteMany({"doctor": "William Yeszk"})

NumberInt	cria um valor int32 => NumberInt (55)
NumberLong	cria um valor int64 => NumberLong (7489729384792)
NumberDecimal	cria um valor duplo de alta precisão => NumberDecimal ("12,99") => Isso pode ser útil para casos em que você precisa de (muitas) casas decimais exatas para cálculos.

Se você usar apenas um número (por exemplo, insertOne ({a: 1}), ele será adicionado como um duplo normal no banco de dados. A razão para isso é que o shell é baseado em JS, que só conhece valores flutuantes / duplos e não não difere entre inteiros e flutuantes.

- **Importar arquivos json no mongodb:**

1. Acessar a pasta em que o arquivo esteja salvo com (cd)
2. Digitar **mongoimport -c** nomedacoleção **--jsonArray --file** nomedoarquivo.json

Diário - é como uma única linha que descreve a operação de gravação. A gravação nos arquivos dos bancos de dados é obviamente uma tarefa mais complexa, pois é preciso encontrar uma posição correta para inseri-lo, se você tiver índices.
Atomicidade - garante que uma transação seja “tudo ou nada”, ou dá certo ou não altera nada, mas não fará pela metade.

Inserções Ordenadas e Não-ordenadas:

Ordenadas - Quando utiliza-se várias inserções, e que se vier algum erro (ex: id duplicado) a inserção não será gravada no banco de dados e é basicamente interrompida.

Não-ordenadas -

Operadores = começam com $ no início