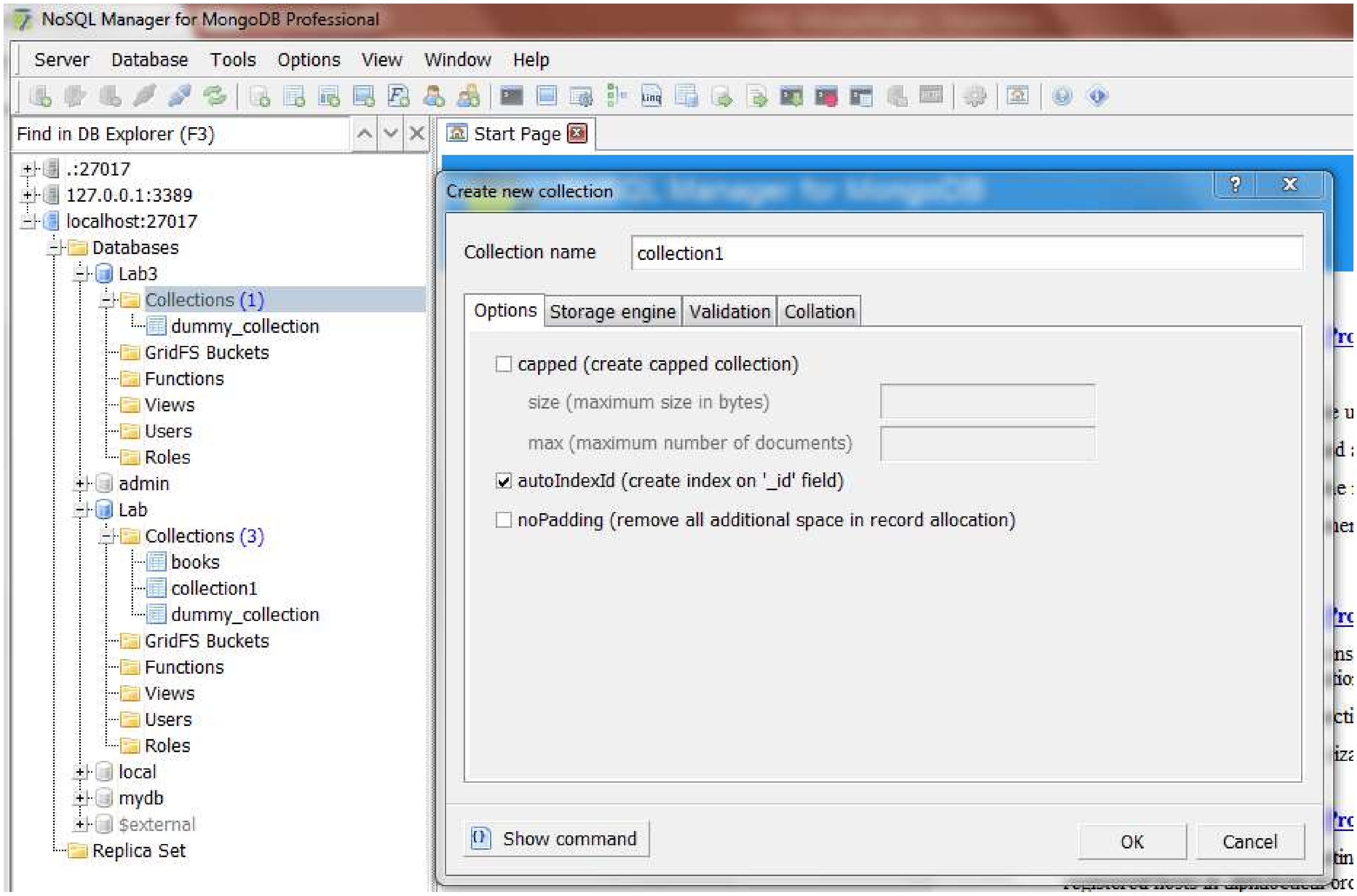
Cairo University
Faculty of Engineering
Computer Engineering Department

Fall 2019
Advanced Database
Lab3: NoSQL MongoDB

Installations

- 1- Install mangodb:
https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/#prerequisites
  - Download the installer (.msi).
  - Install
    - If installed in "D:...." make a folder in the D drive named "data" in it a folder named "db“
- 2-Install NoSQL manager for mongodb Freeware:
https://www.mongodbmanager.com/download
- To ensure that everything is working open "mongod.exe" in the installed folder for mongodb.
Then open the NoSQL manager connecting to the default:"localhost:27017" (keeping mongod.exe opened).

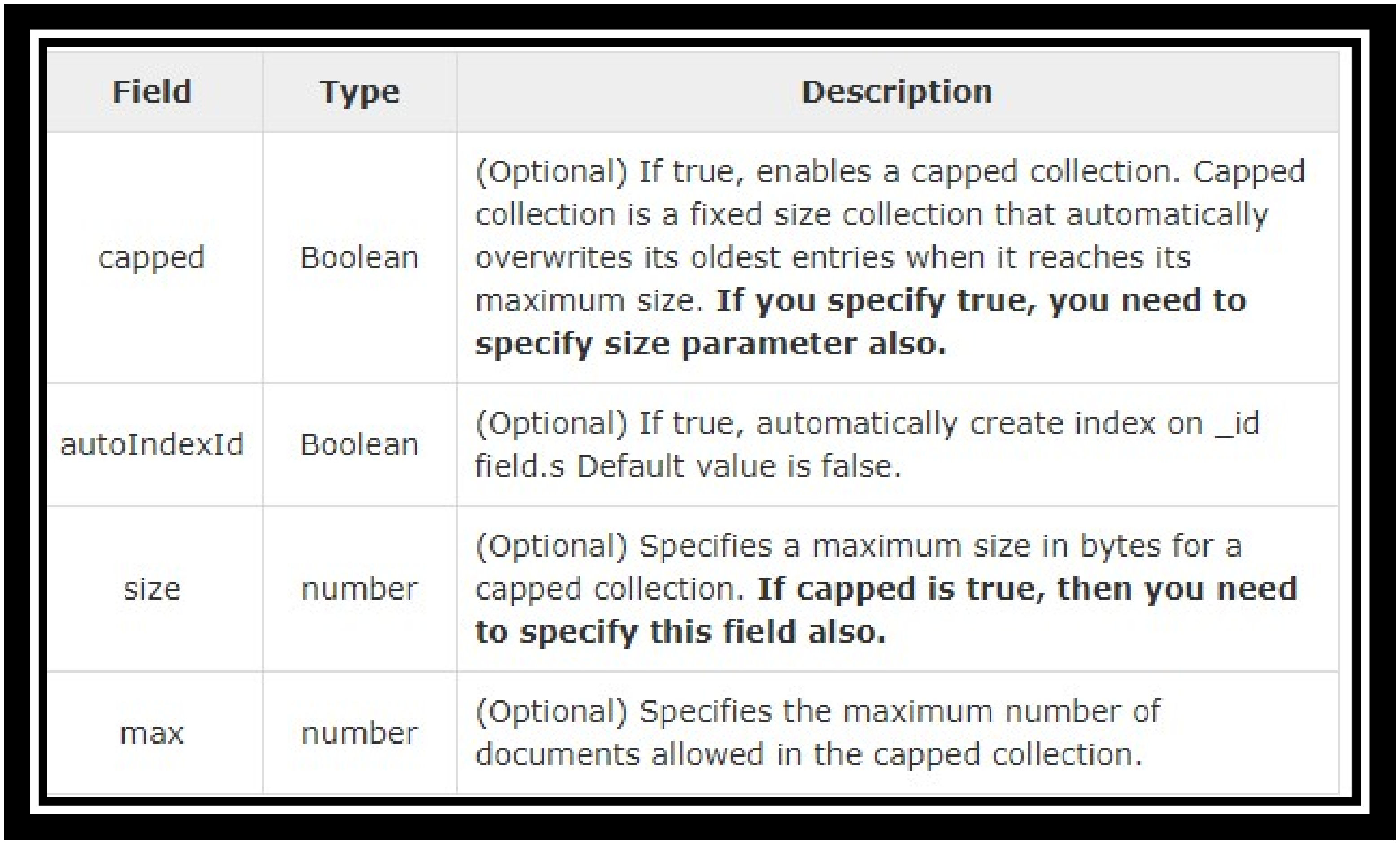
Create Collection



Collection name: collection1

- [ ] capped (create capped collection)
  - size (maximum size in bytes)
  - max (maximum number of documents)
- [✓] autoIndexId (create index on '_id' field)
- [ ] noPadding (remove all additional space in record allocation)

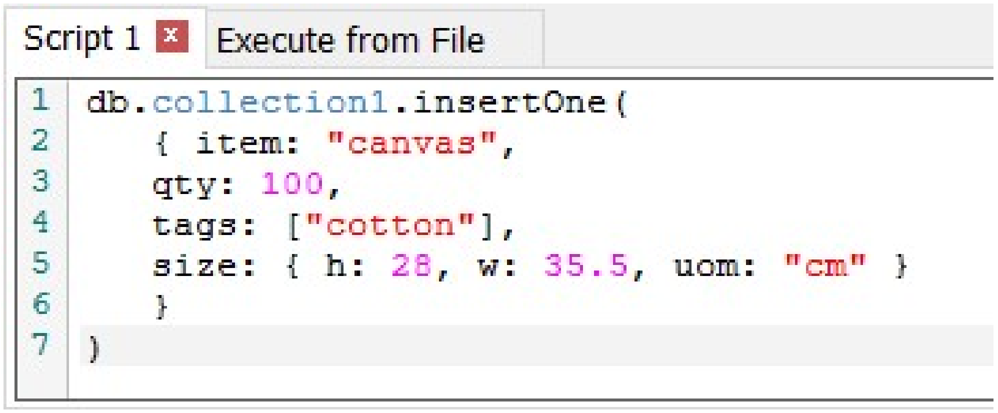
Create Collection



Field	Type	Description
capped	Boolean	(Optional) If true, enables a capped collection. Capped collection is a fixed size collection that automatically overwrites its oldest entries when it reaches its maximum size. **If you specify true, you need to specify size parameter also.**
autoIndexId	Boolean	(Optional) If true, automatically create index on _id field.s Default value is false.
size	number	(Optional) Specifies a maximum size in bytes for a capped collection. **If capped is true, then you need to specify this field also.**
max	number	(Optional) Specifies the maximum number of documents allowed in the capped collection.

Insert Document

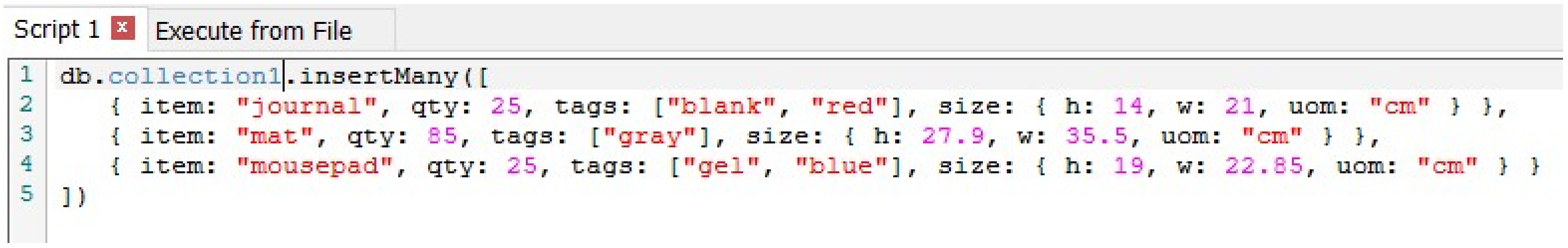
- db.COLLECTION_NAME.insertone(document)



```
db.collection1.insertOne(
   { item: "canvas",
   qty: 100,
   tags: ["cotton"],
   size: { h: 28, w: 35.5, uom: "cm" }
   }
)
```

Insert Document

- To insert multiple documents in a single query, you can pass an array of documents in insertmany() command.



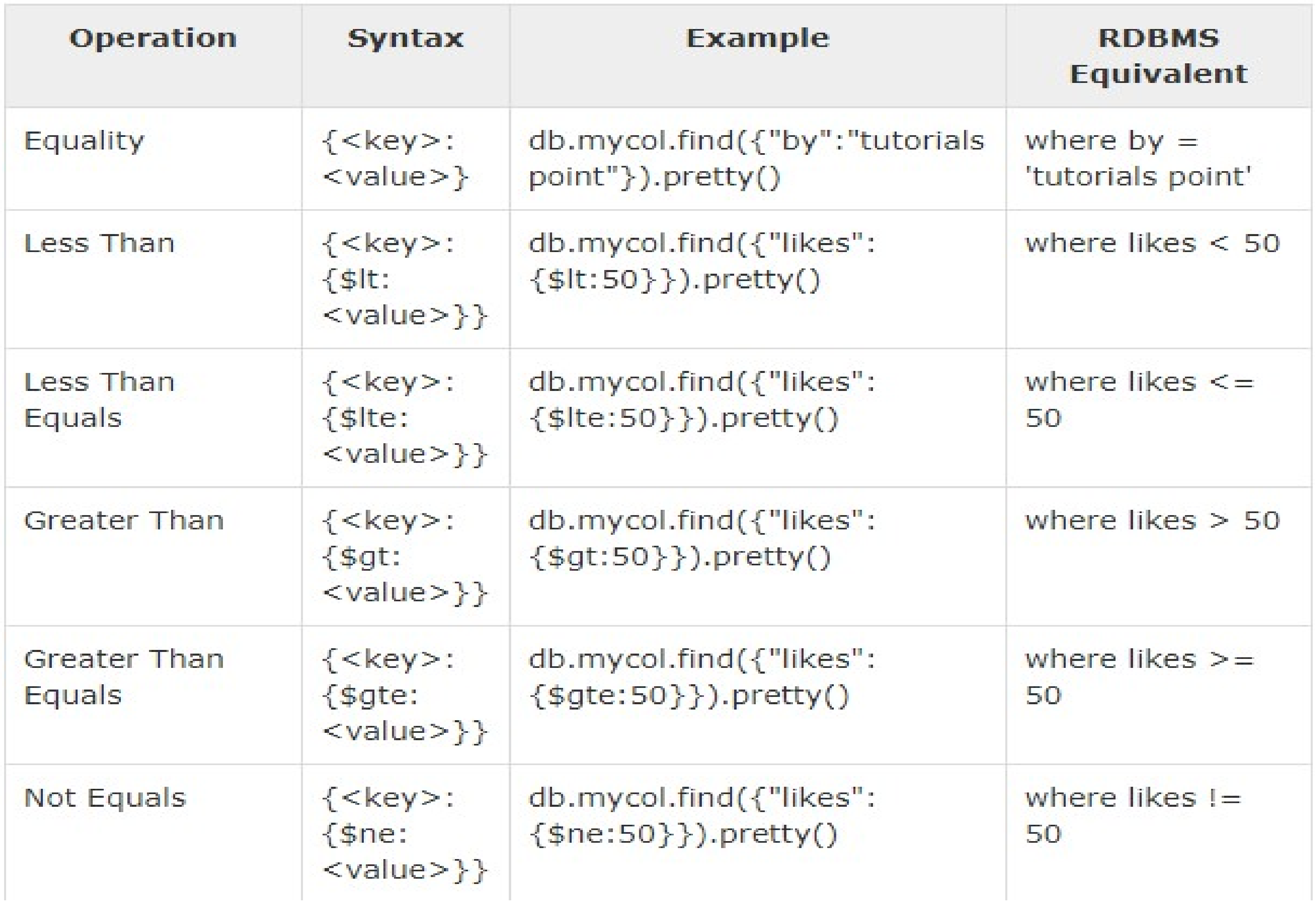
Script 1 Execute from File

```
db.collection1.insertMany([
   { item: "journal", qty: 25, tags: ["blank", "red"], size: { h: 14, w: 21, uom: "cm" } },
   { item: "mat", qty: 85, tags: ["gray"], size: { h: 27.9, w: 35.5, uom: "cm" } },
   { item: "mousepad", qty: 25, tags: ["gel", "blue"], size: { h: 19, w: 22.85, uom: "cm" } }
])
```

Query Document

- db.COLLECTION_NAME.find()
  - **find()** method will display all the documents in a non-structured way.
  - SELECT * FROM COLLECTION_NAME
  - db. COLLECTION_NAME.find({ status: "D" })
  - db. COLLECTION_NAME.find({ status: { $in: ["A", "D"] } })
- db.COLLECTION_NAME.find().pretty()
  - to display the results in a formatted way.
- **findOne()** method, returns only one document.

RDBMS Where Clause Equivalents in MongoDB

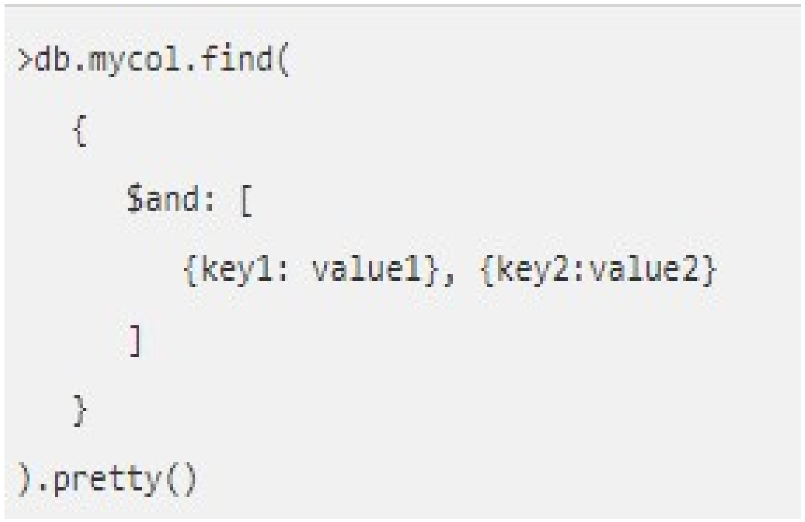


Operation	Syntax	Example	RDBMS Equivalent
Equality	{<key>:<value>}	db.mycol.find({"by":"tutorials point"}).pretty()	where by = 'tutorials point'
Less Than	{<key>:{$lt:<value>}}	db.mycol.find({"likes":{$lt:50}}).pretty()	where likes < 50
Less Than Equals	{<key>:{$lte:<value>}}	db.mycol.find({"likes":{$lte:50}}).pretty()	where likes <= 50
Greater Than	{<key>:{$gt:<value>}}	db.mycol.find({"likes":{$gt:50}}).pretty()	where likes > 50
Greater Than Equals	{<key>:{$gte:<value>}}	db.mycol.find({"likes":{$gte:50}}).pretty()	where likes >= 50
Not Equals	{<key>:{$ne:<value>}}	db.mycol.find({"likes":{$ne:50}}).pretty()	where likes != 50

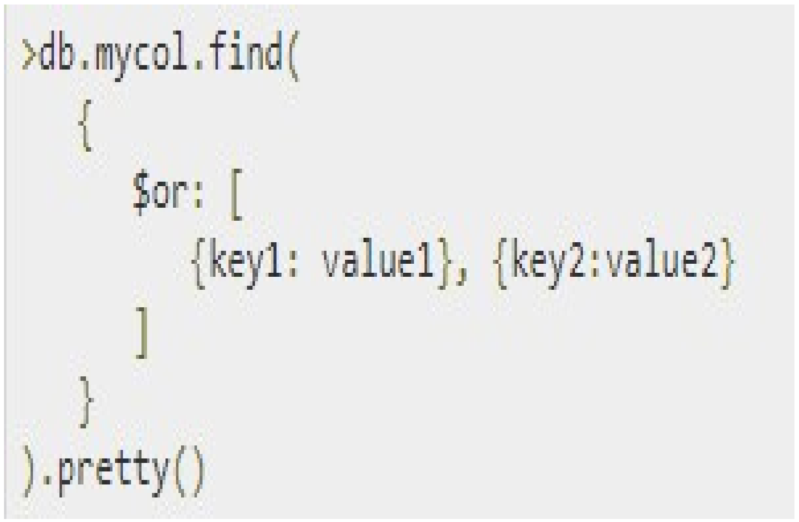
AND/OR in MongoDB

if you pass multiple keys by separating them by ‘,’

- db.inventory.find({ status: "A", qty: { $lt: 30 } })
- db.inventory.find({ $or: [{ status: "A" }, { qty: { $lt: 30 } }] })



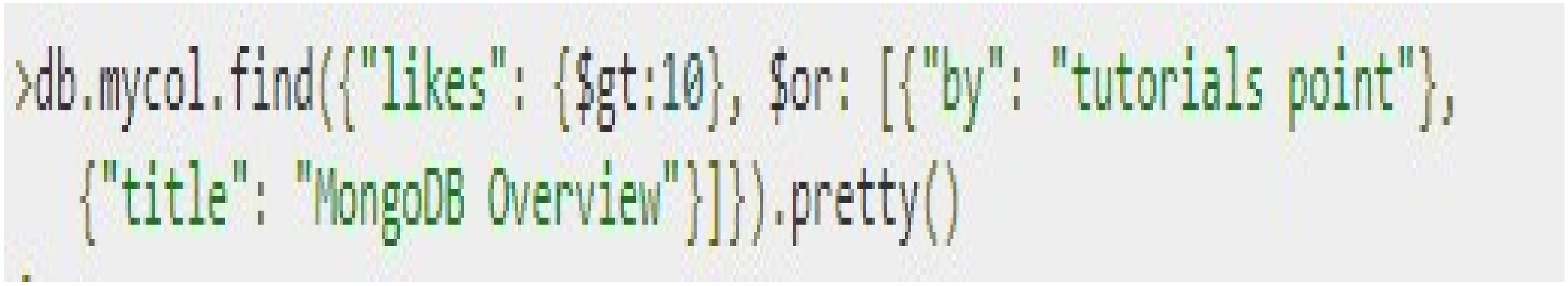
```
>db.mycol.find(
   {
      $and: [
         {key1: value1}, {key2:value2}
      ]
   }
).pretty()
```



```
>db.mycol.find(
   {
      $or: [
         {key1: value1}, {key2:value2}
      ]
   }
).pretty()
```

Example

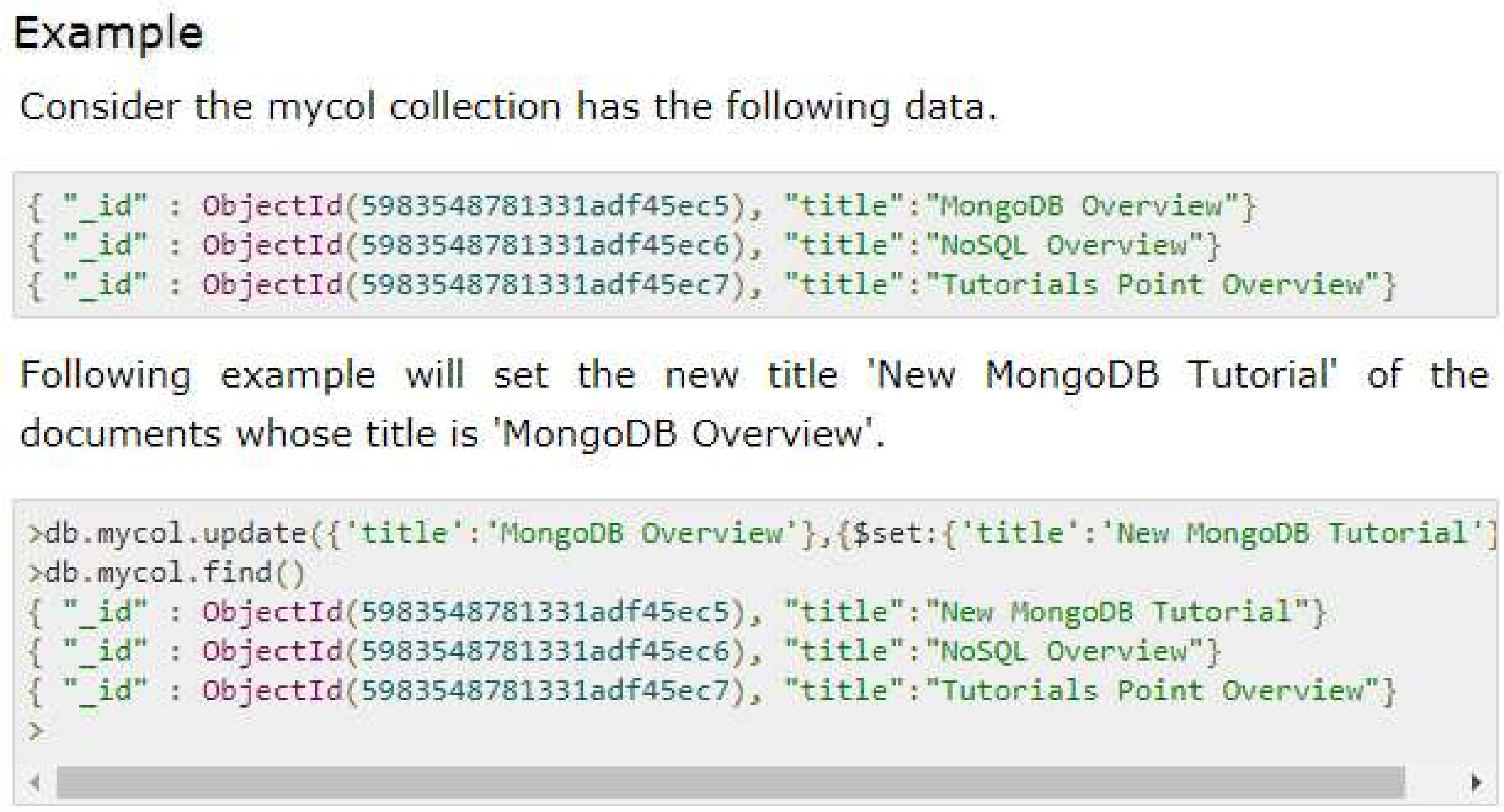
- Equivalent SQL where clause is **'where likes>10 AND (by = 'tutorials point' OR title = 'MongoDB Overview')'**



```
>db.mycol.find({"likes": {$gt:10}, $or: [{"by": "tutorials point"},
   {"title": "MongoDB Overview"}]}).pretty()
```

Update Document

- db.COLLECTION_NAME.update(SELECTION_CRITERIA, UPDATED_DATA)



Example

Consider the mycol collection has the following data.

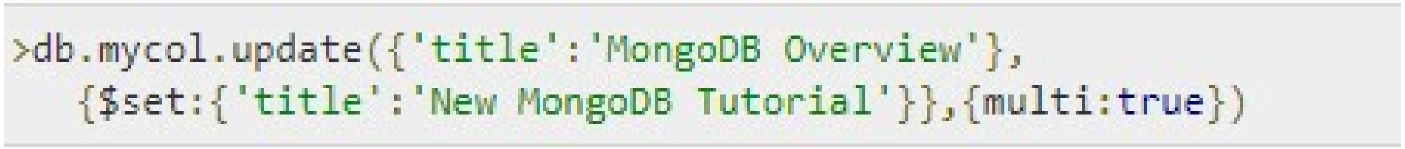
```
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
```

Following example will set the new title 'New MongoDB Tutorial' of the documents whose title is 'MongoDB Overview'.

```
>db.mycol.update({'title':'MongoDB Overview'},{$set:{'title':'New MongoDB Tutorial'}
>db.mycol.find()
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"New MongoDB Tutorial"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
>
```

Update Document

- By default, MongoDB will update only a single document.
  - To update multiple documents, you need to set parameter 'multi' to true.



```
>db.mycol.update({'title':'MongoDB Overview'},
   {$set:{'title':'New MongoDB Tutorial'}},{multi:true})
```

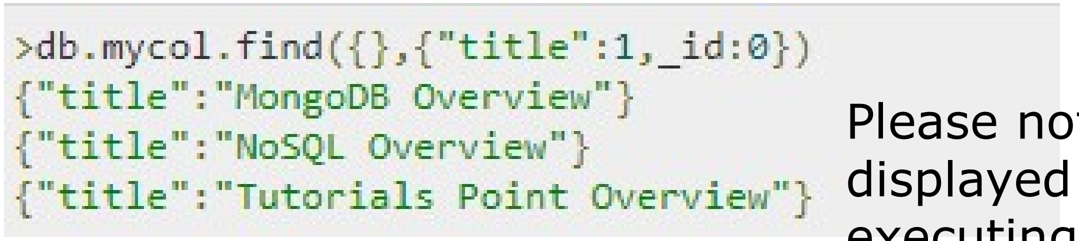
  - db.inventory.updateOne({ item: "paper" }, { $set: { "size.uom": "cm", status: "P" }, $currentDate: { lastModified: true } })
  - db.inventory.updateMany({ "qty": { $lt: 50 } }, { $set: { "size.uom": "in", status: "P" }, $currentDate: { lastModified: true } })

Delete Document

- db.inventory.deleteMany({ status : "A" }).
- db.inventory.deleteOne({ status: "D" })

MongoDB - Projection

- When you execute **find()** method, then it displays all fields of a document. To limit this, you need to set a list of fields with value 1 or 0.
  - 1 is used to show the field while 0 is used to hide the fields.
  - db.COLLECTION_NAME.find({},{KEY:1})

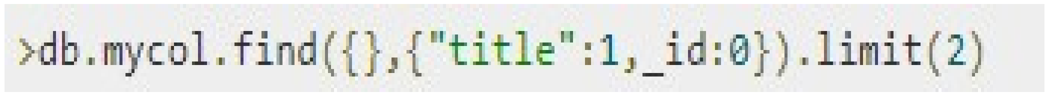


```
>db.mycol.find({},{"title":1,_id:0})
{"title":"MongoDB Overview"}
{"title":"NoSQL Overview"}
{"title":"Tutorials Point Overview"}
```

Please note **_id** field is always displayed while executing **find()** method, if you don't want this field, then you need to set it as 0.

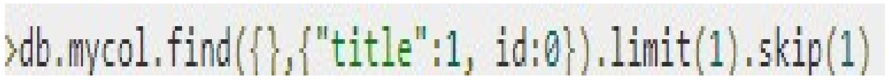
MongoDB - Limit Records

- db.COLLECTION_NAME.find().limit(NUMBER)
  - The method accepts one number type argument, which is the number of documents that you want to be displayed.



```
>db.mycol.find({},{"title":1,_id:0}).limit(2)
```

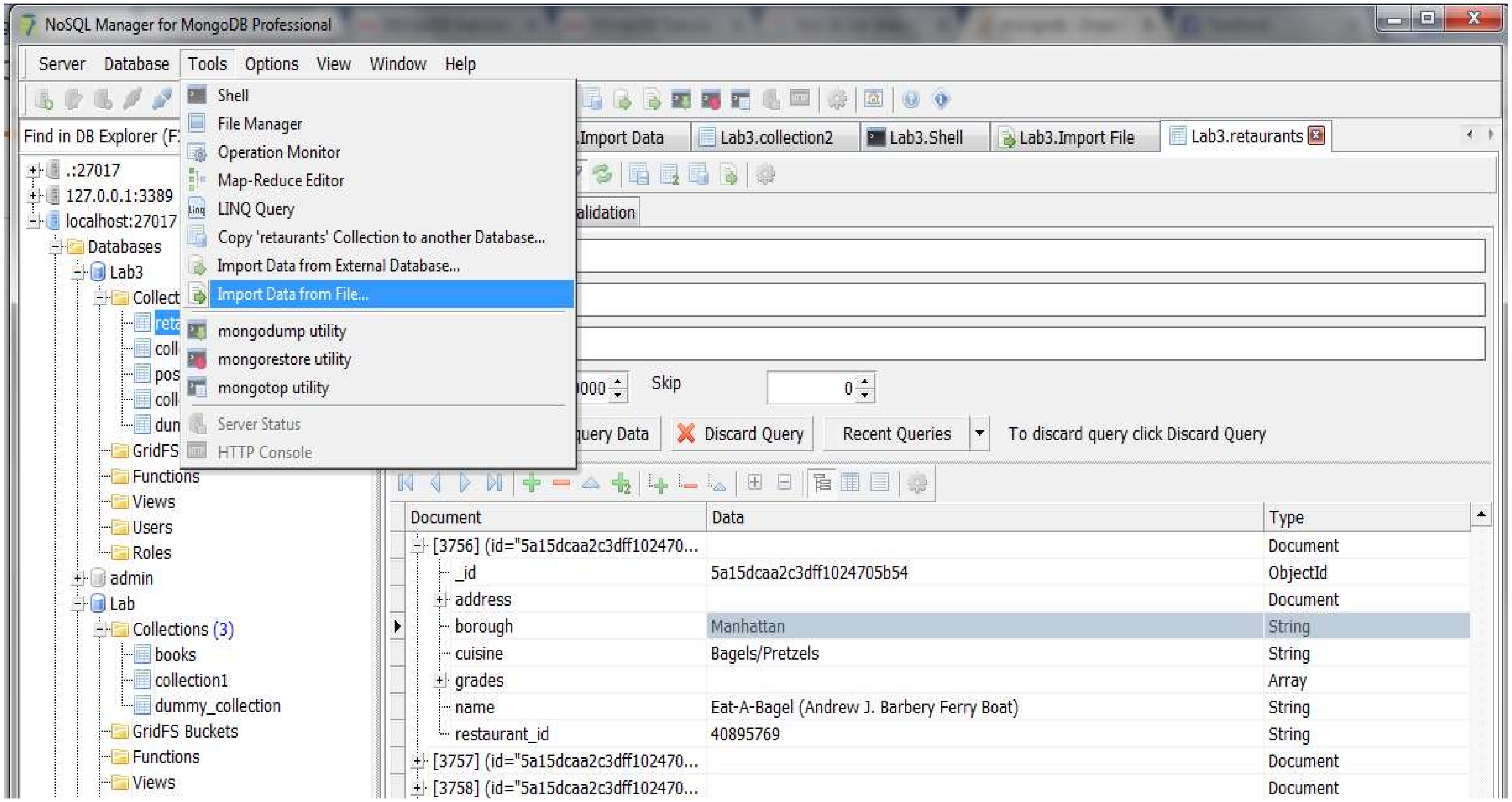
- **skip()** which also accepts number type argument and is used to skip the number of documents.



```
>db.mycol.find({},{"title":1, id:0}).limit(1).skip(1)
```

will display only the second document.

Importing Data from File



Document	Data	Type
[3756] (id="5a15dcaa2c3dff102470...		Document
_id	5a15dcaa2c3dff1024705b54	ObjectId
address		Document
borough	Manhattan	String
cuisine	Bagels/Pretzels	String
grades		Array
name	Eat-A-Bagel (Andrew J. Barbery Ferry Boat)	String
restaurant_id	40895769	String
[3757] (id="5a15dcaa2c3dff102470...		Document
[3758] (id="5a15dcaa2c3dff102470...		Document

ANY QUESTIONS?

THANK YOU