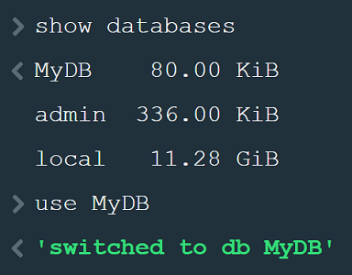
1. To Show the Available Databases:

show dbs OR show databases

1. To Use or Switch to required DB:

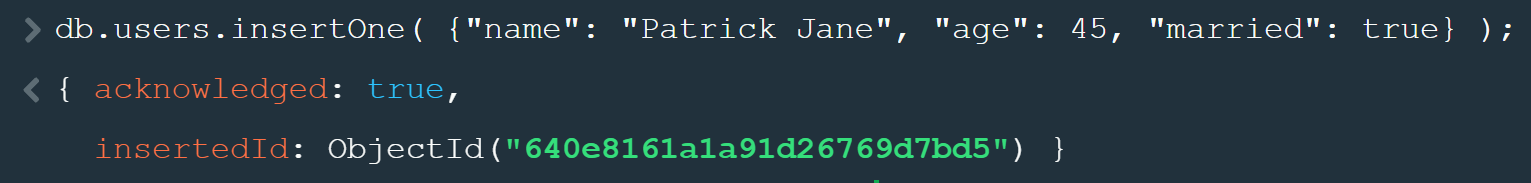
Use dbName

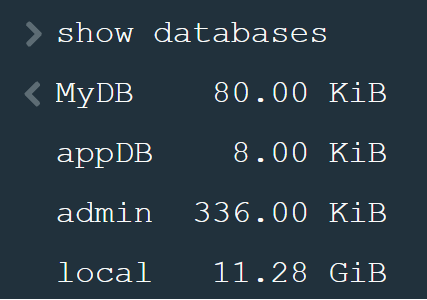
If the DB exists it will Switch to that otherwise MongoDB will create it for you & then switch to it



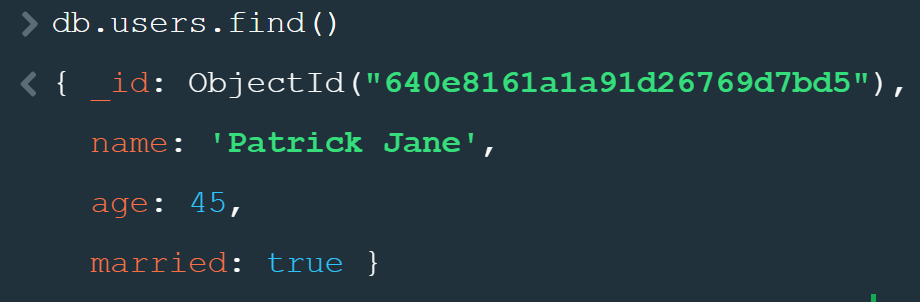
Once data is present only then we can see it in show databases:

Insert Data Using insertOne()



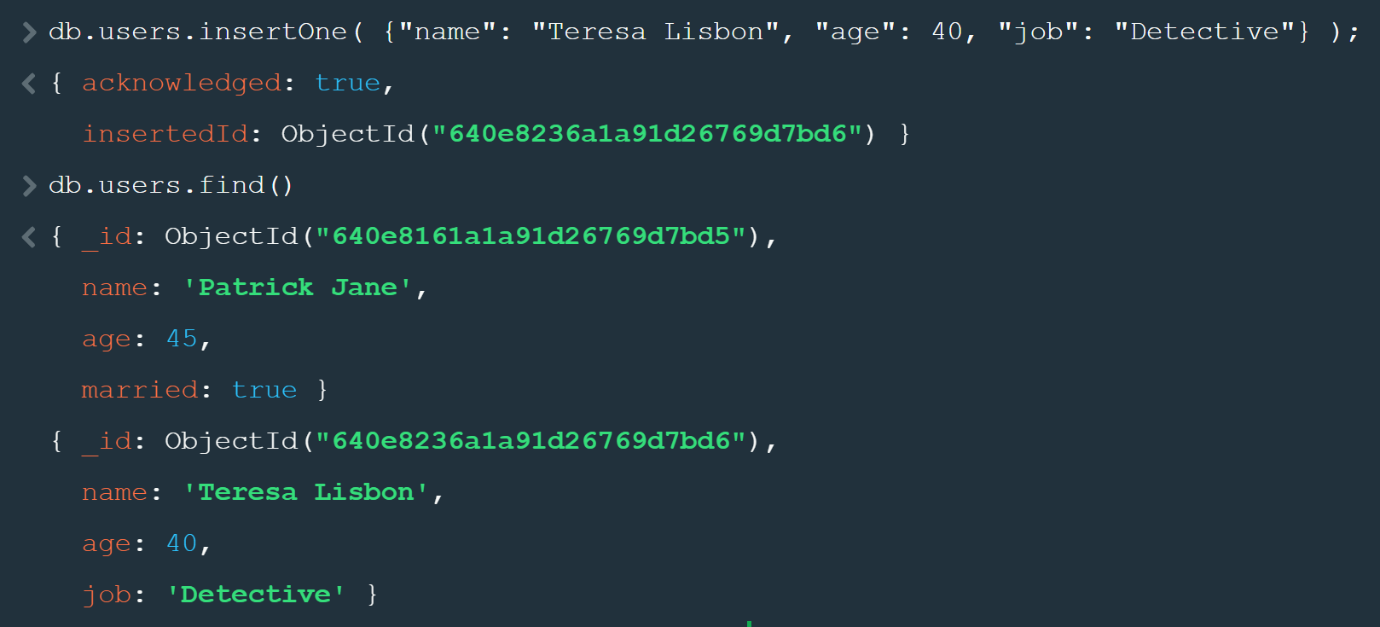


To get all values from a collection use find(): (Like SELECT \* FROM users)

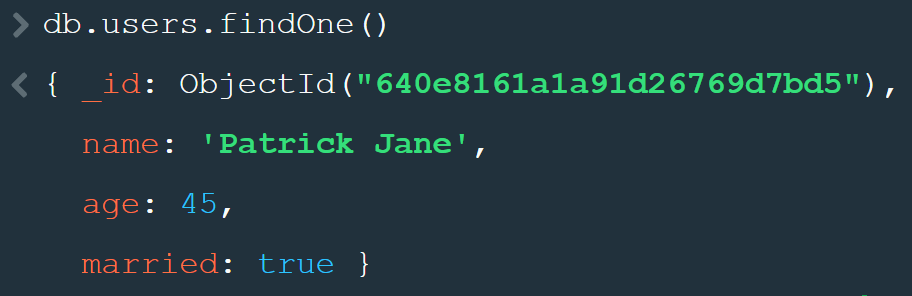


No need to define any Field or Schema’s it’s created based on what you enter.

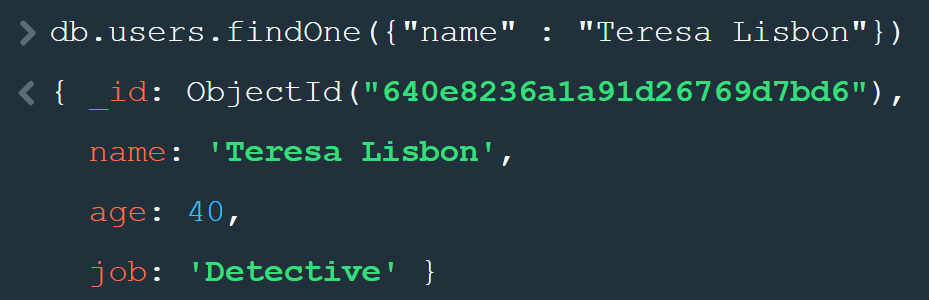
Thus, we can define data with different structure.



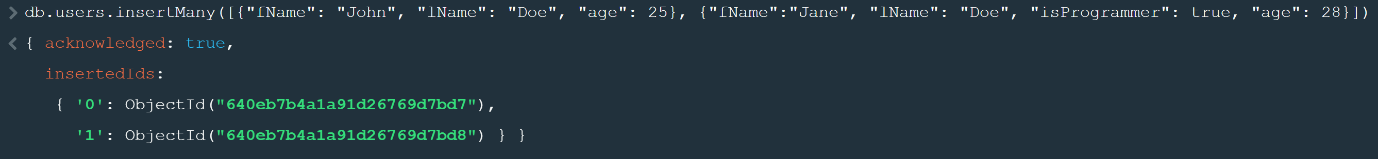
To get the first document:



To Filter Data like WHERE name = “Teresa Lisbon”

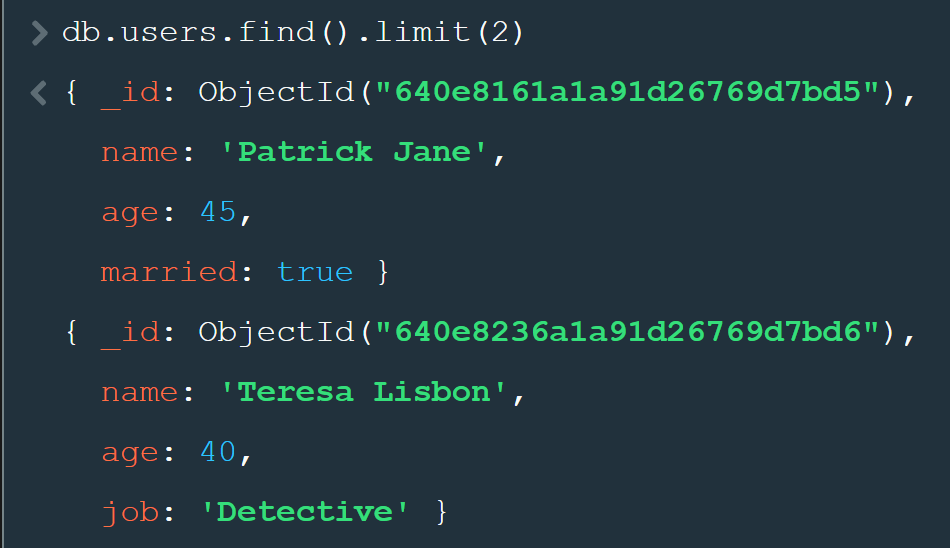


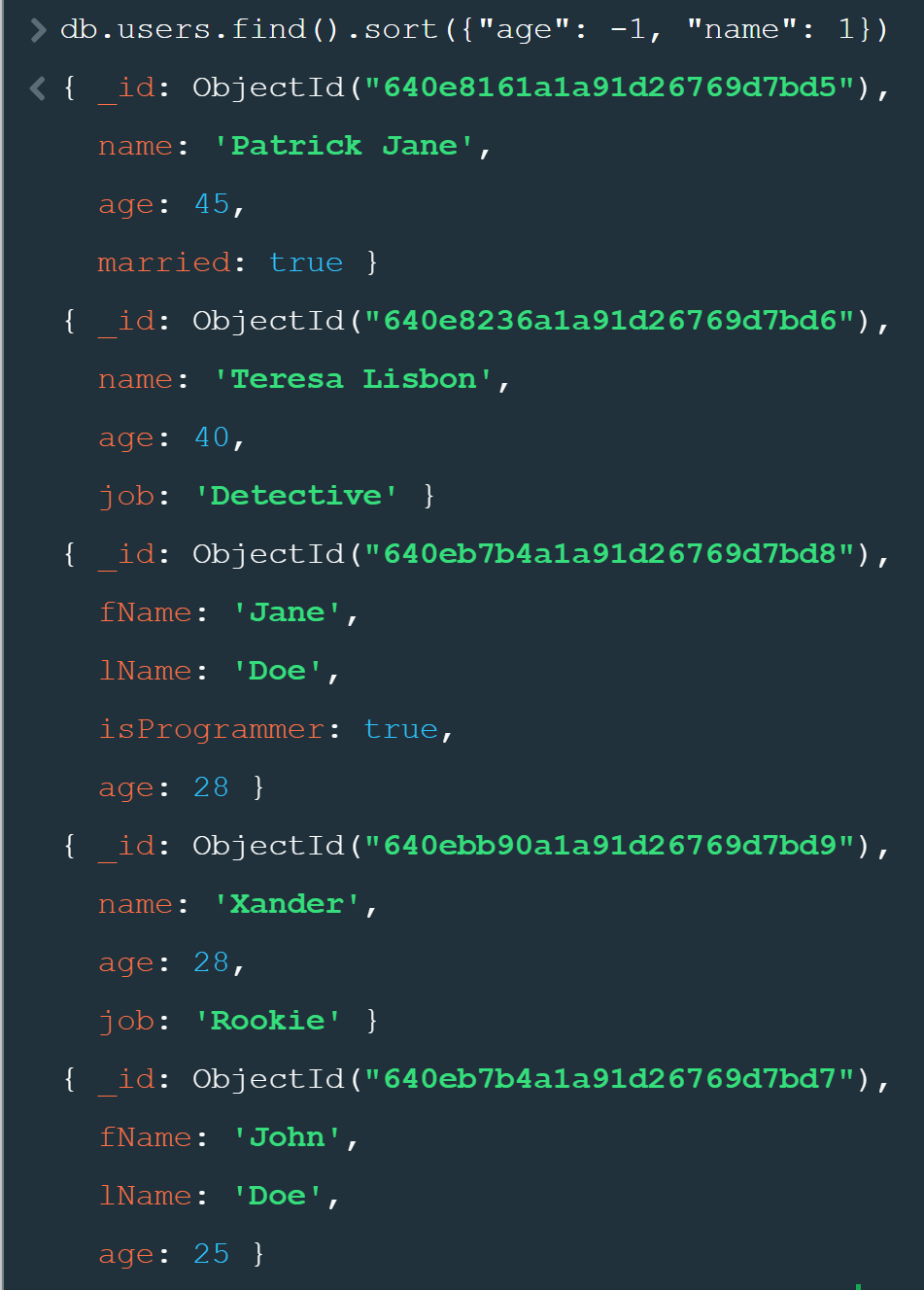
To Insert Multiple documents: JSON Array



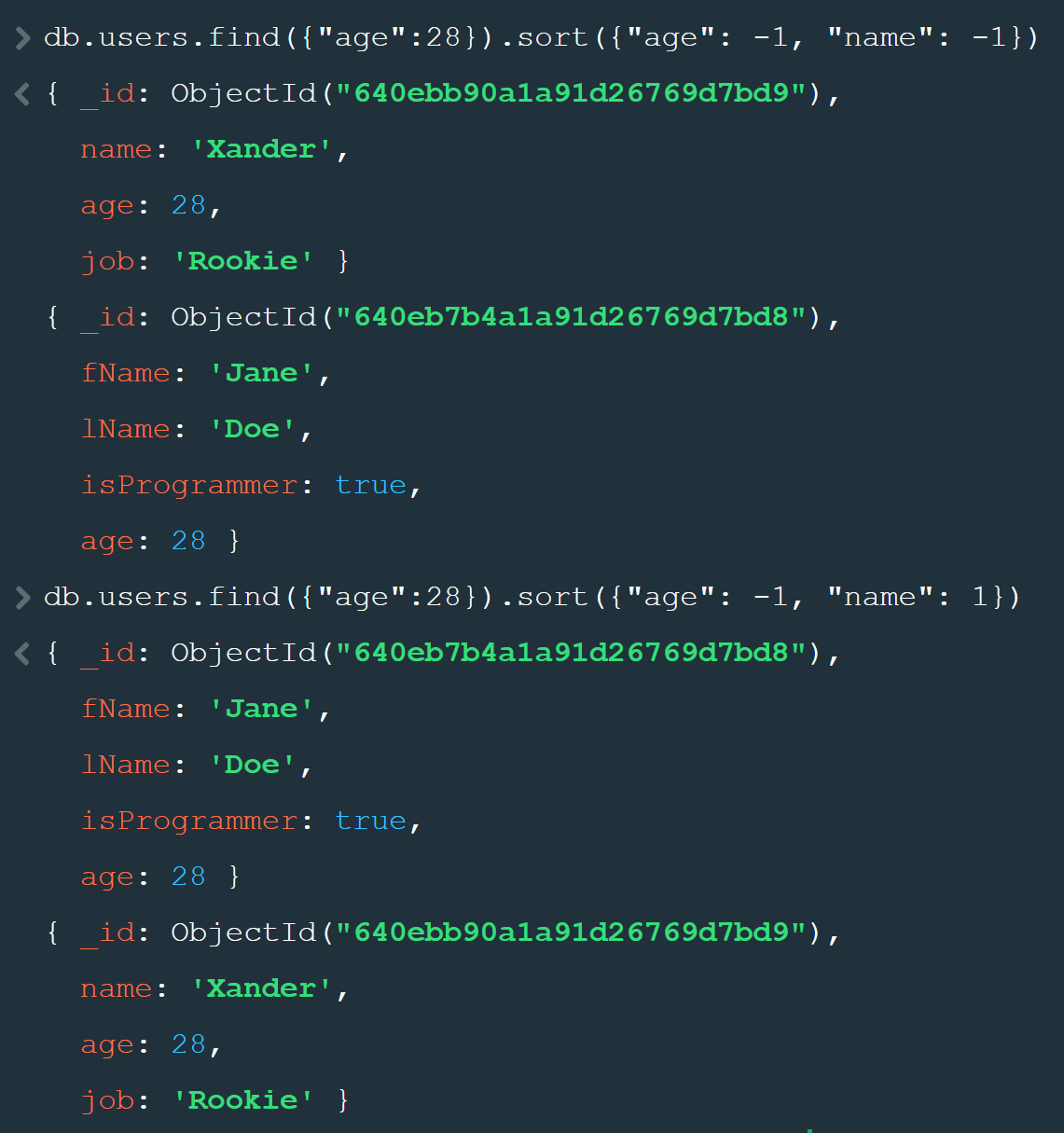
To Limit your Records add limit(numOfRecords):

To sort your data, provide the value and then 1 for Ascending & -1 for Descending

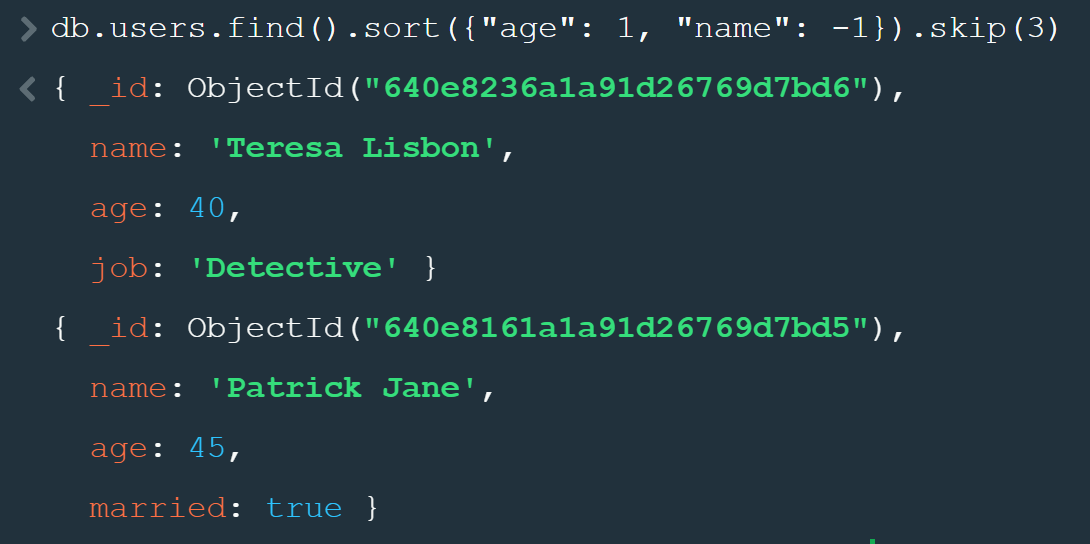




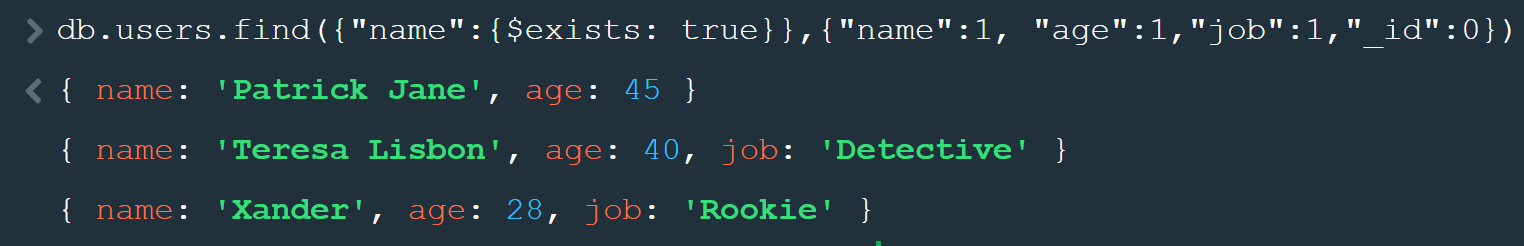
While sorting, if we are sorting based on two values, then first it will be sorted by the first property (age) and if two are same then on the second property (name):



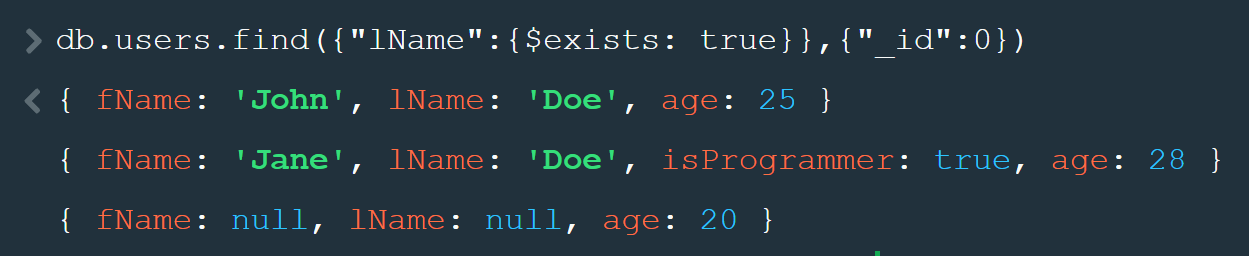
To skip documents we use the skip() method :



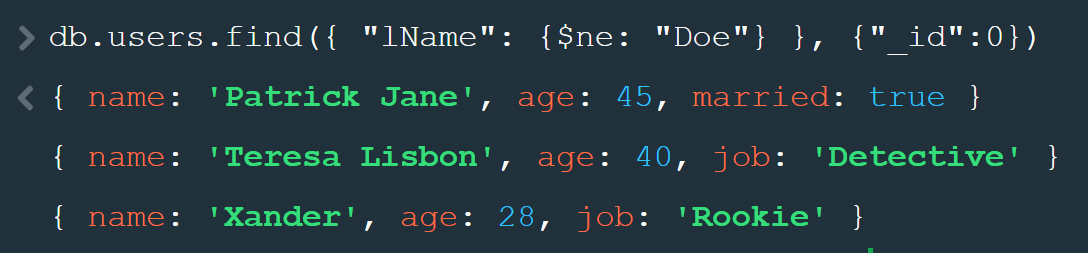
Print Data if Field Exists ($exists: true) and then Print Only Selected properties : 1 to Show and 0 to Hide the properties:



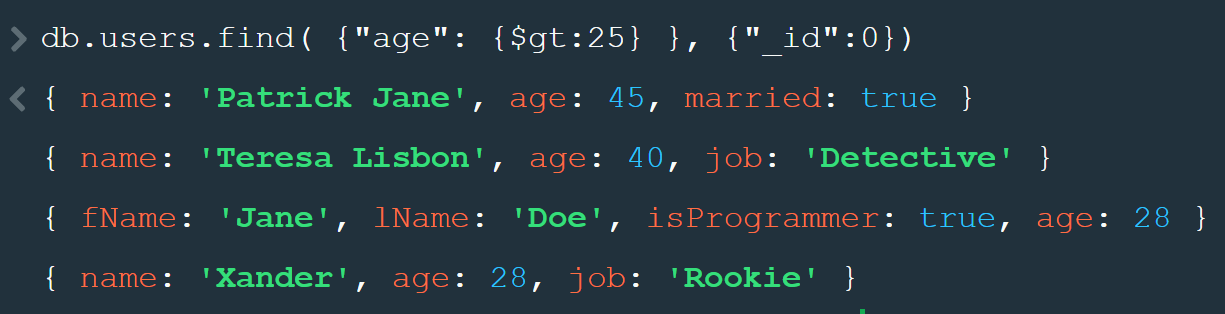
$exists only checks if the key is present even if it’s Null. Prints everything except ID:

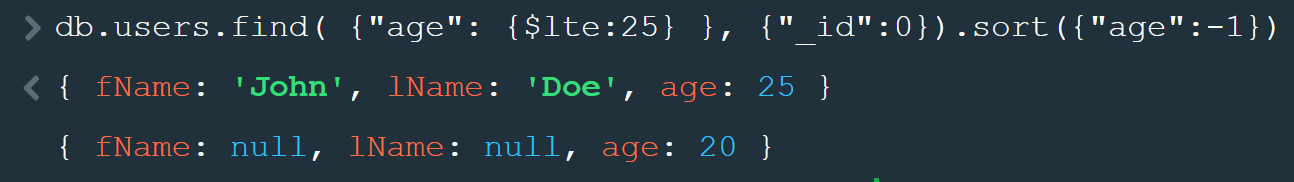


To Check if something is not equal (Find the Records where lName not equal to Doe):

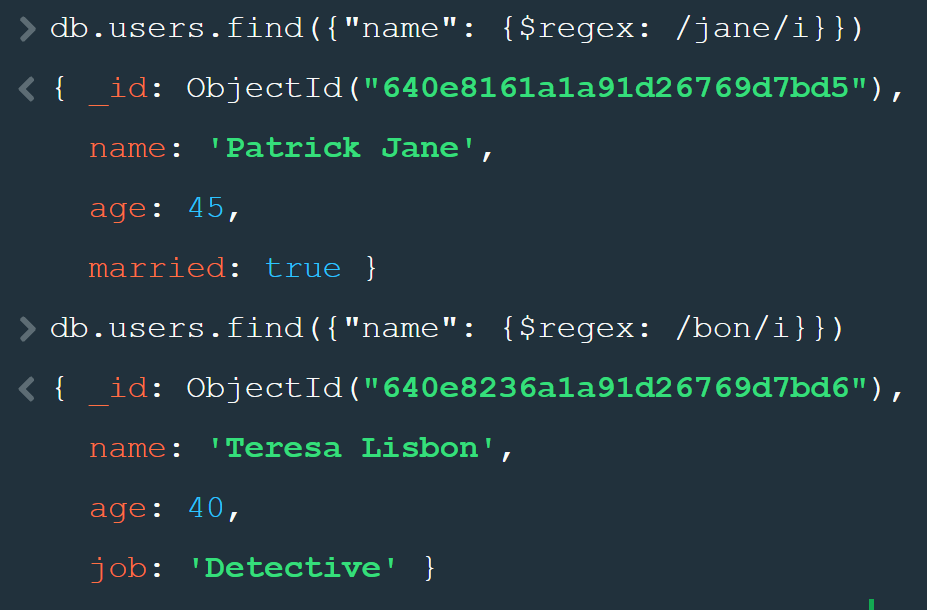


Find Documents with Age > 25 **(> : $gt, < : $lt, >= $gte, <= : $lte, == : $eq, != : $ne )**

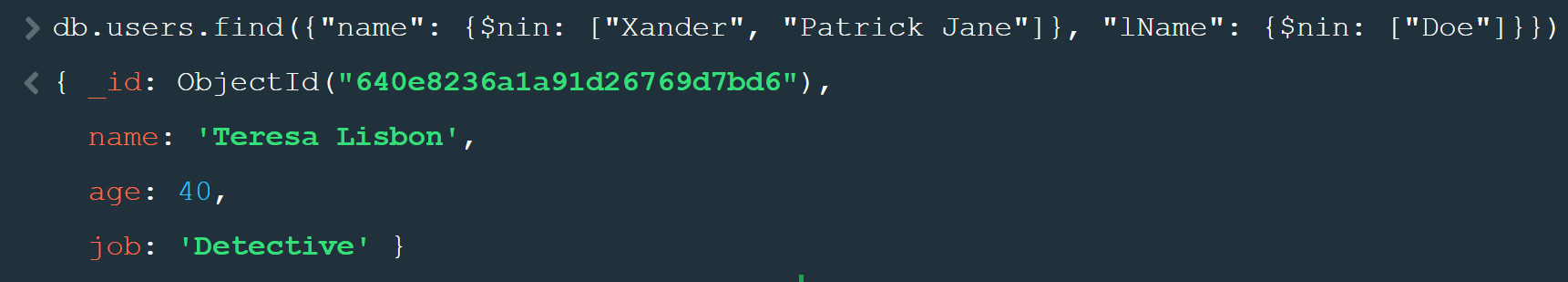


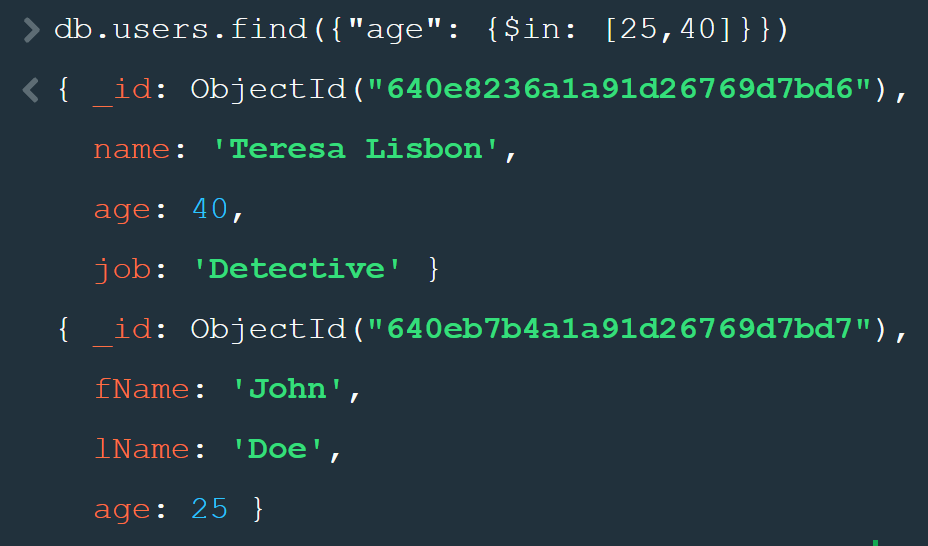


Find Based upon some Value $regex:



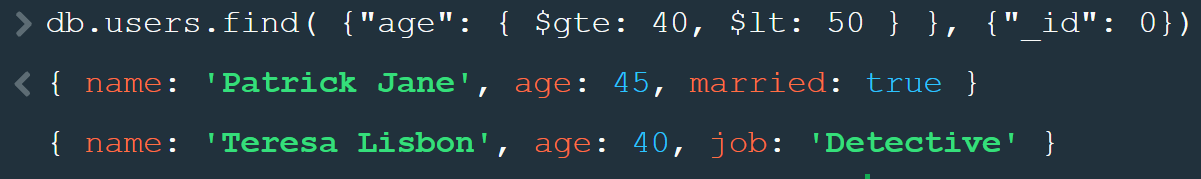
IN is $in or NOT-IN $nin – Find the documents where name NOT IN Xander, Patrick Jane or lName not in Doe so only Teresa:



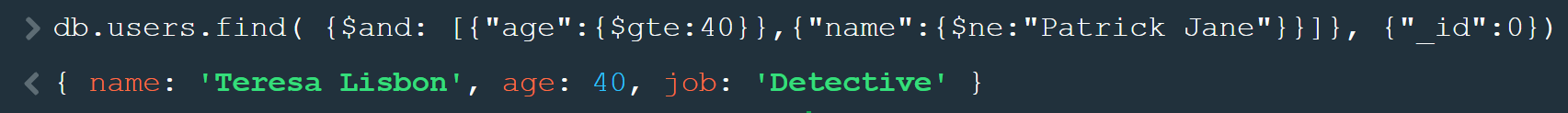


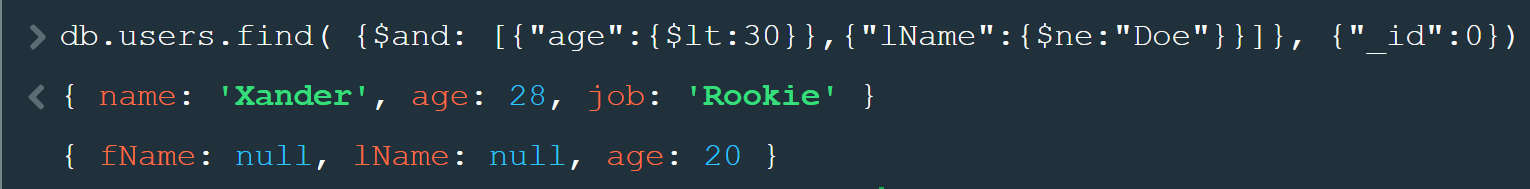
Find Values between a Range:

One way of getting AND values:

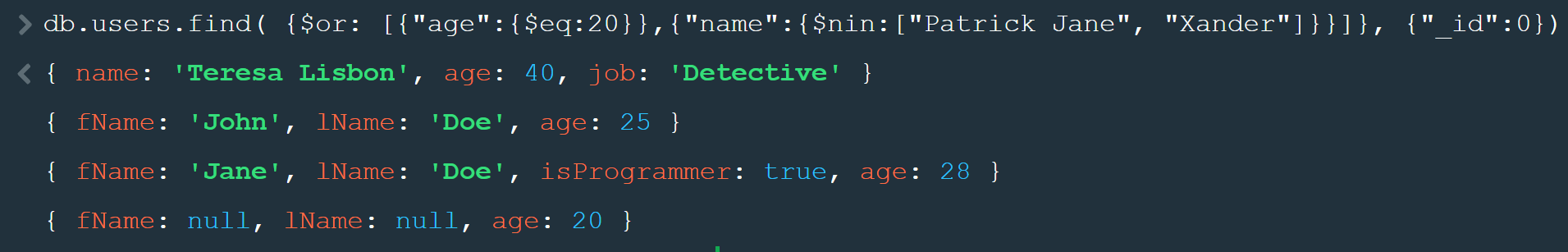


Other way Directly $and:

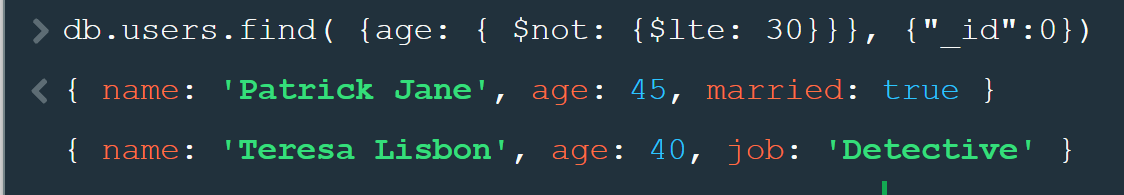




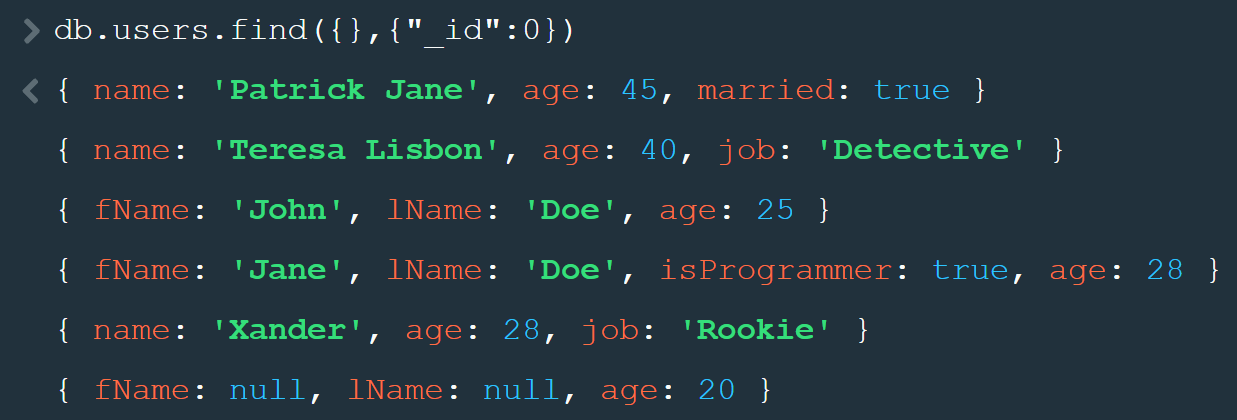
To get the OR Condition:



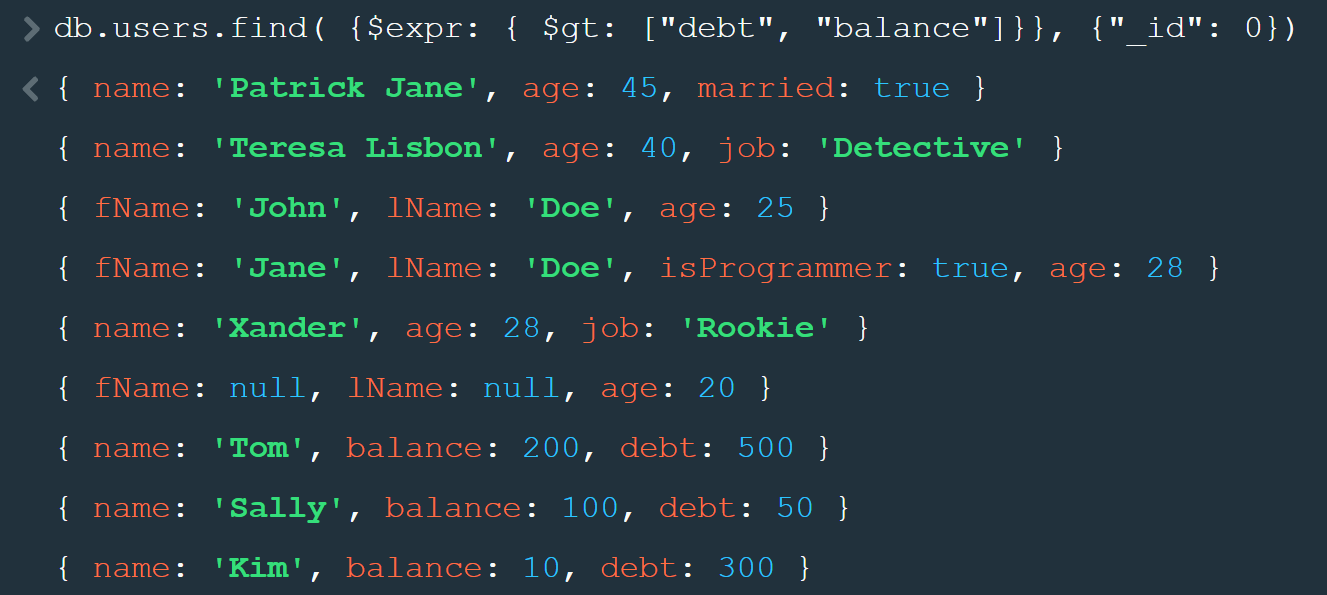
$not Negates the whole condition:



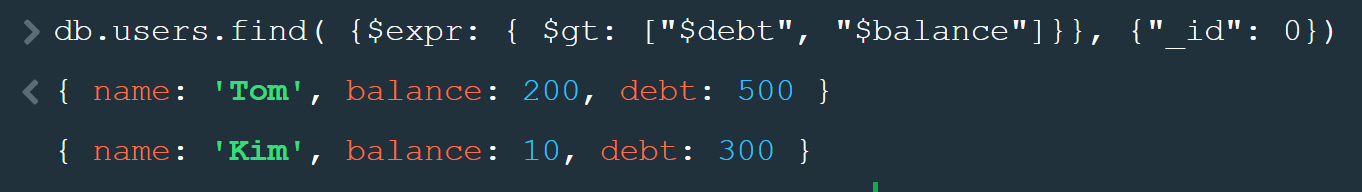
Print Everything without the ID Field:



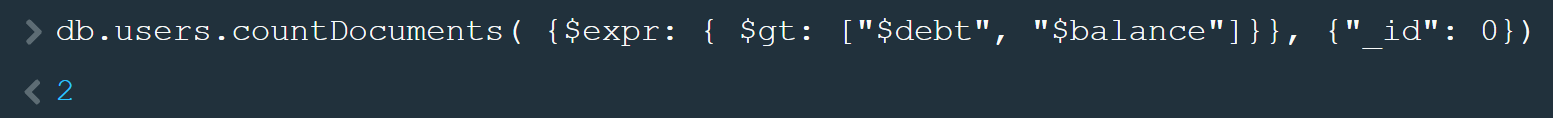
Find People where their Debt > Balance

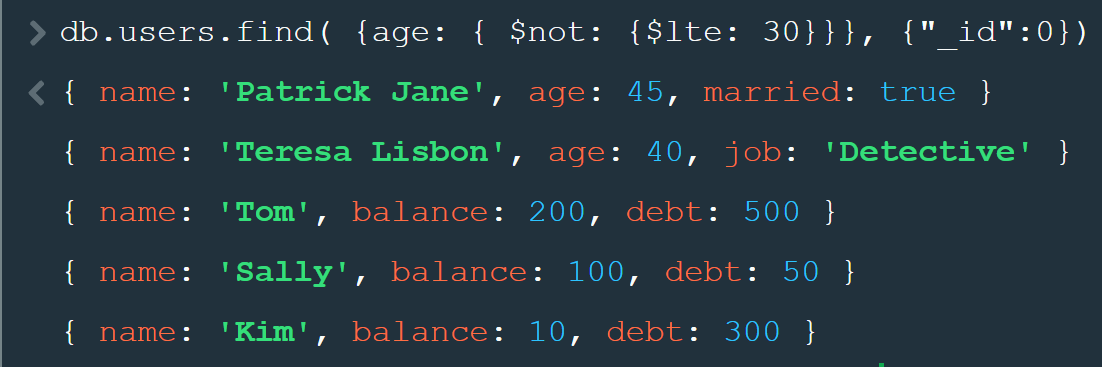


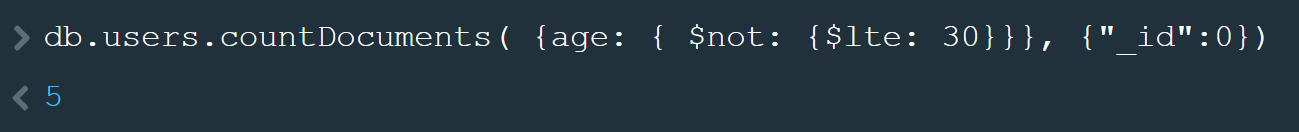
But we get all Documents so we add $ to the Field Names:



To get a count of the documents we replace find with countDocuments:

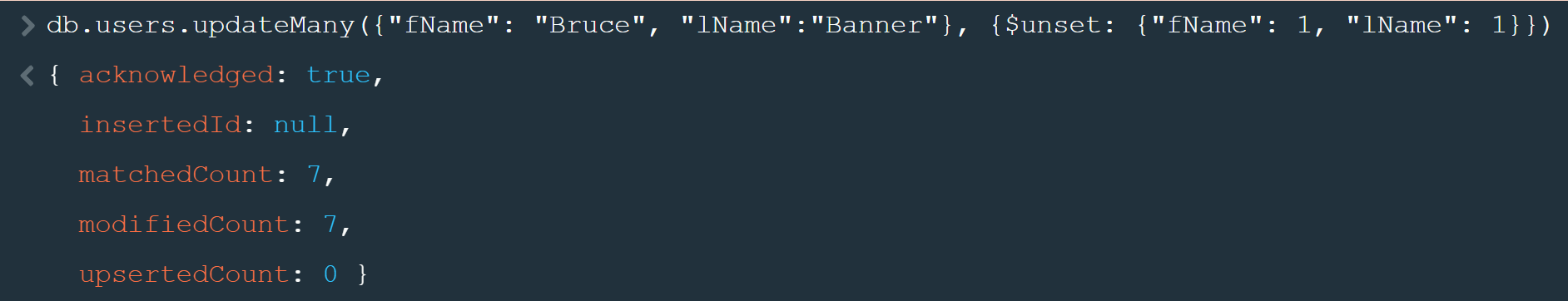




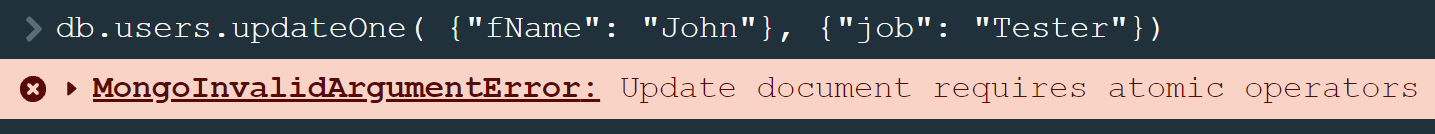


Update the Document based upon Wrong Fields:

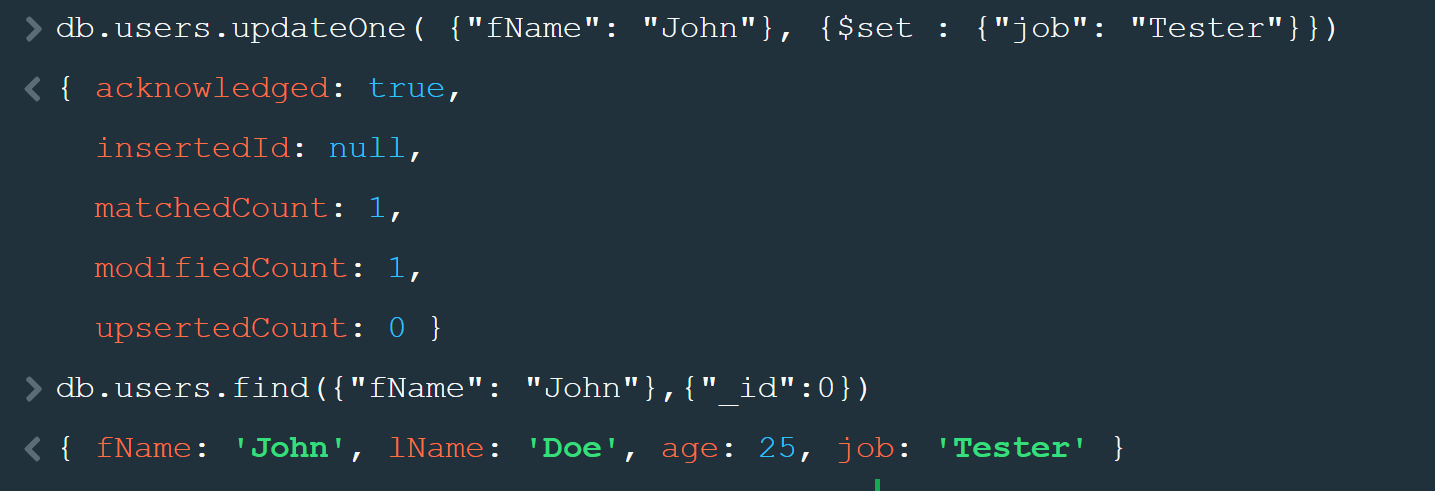
Delete from the documents wherever the fName = Bruce and lName = Banner



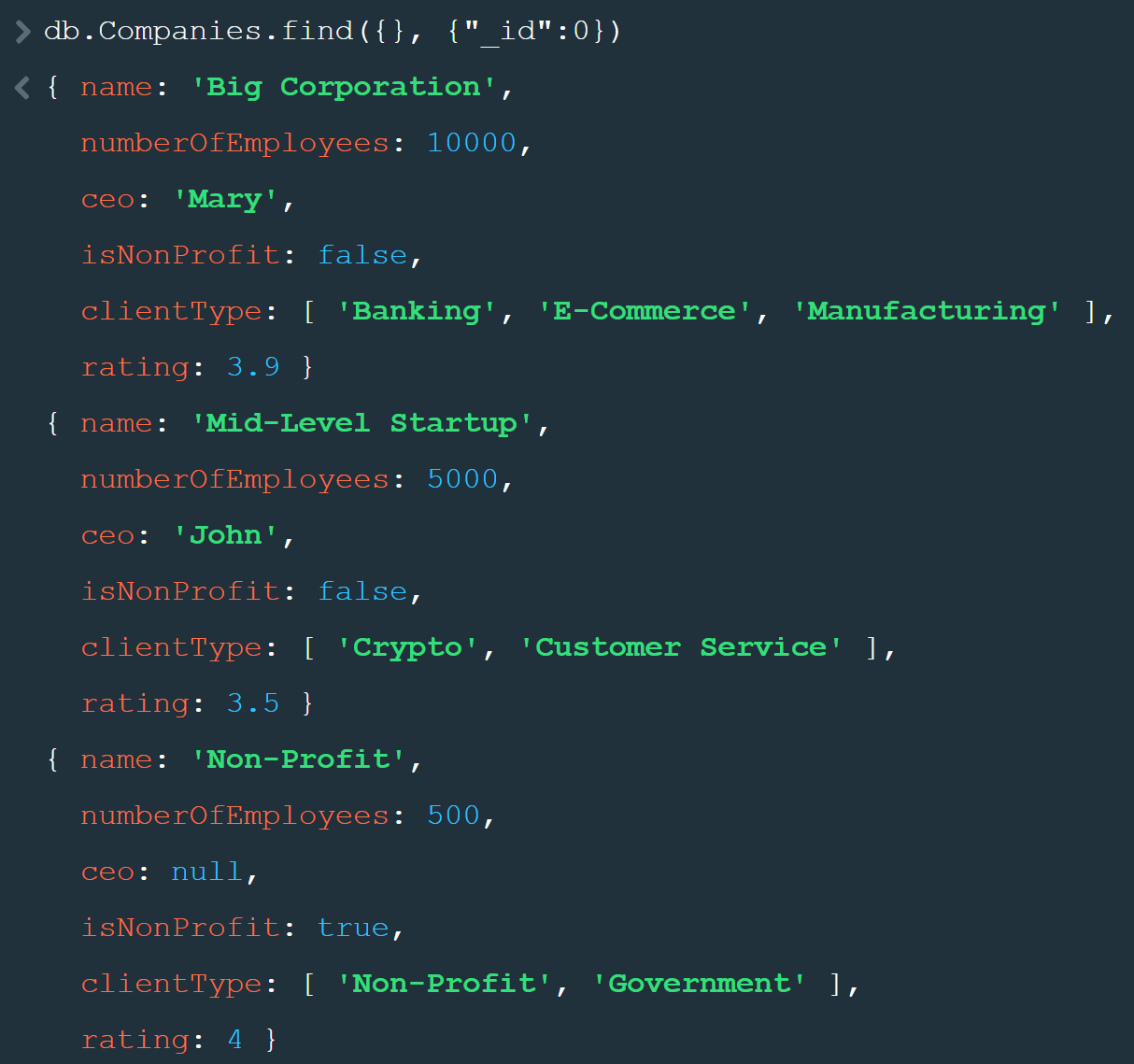
Update Values Documents, just setting values is not allowed

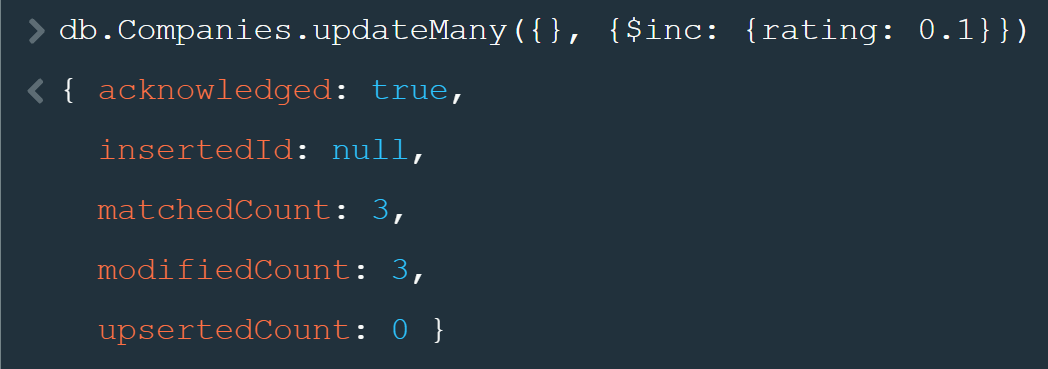


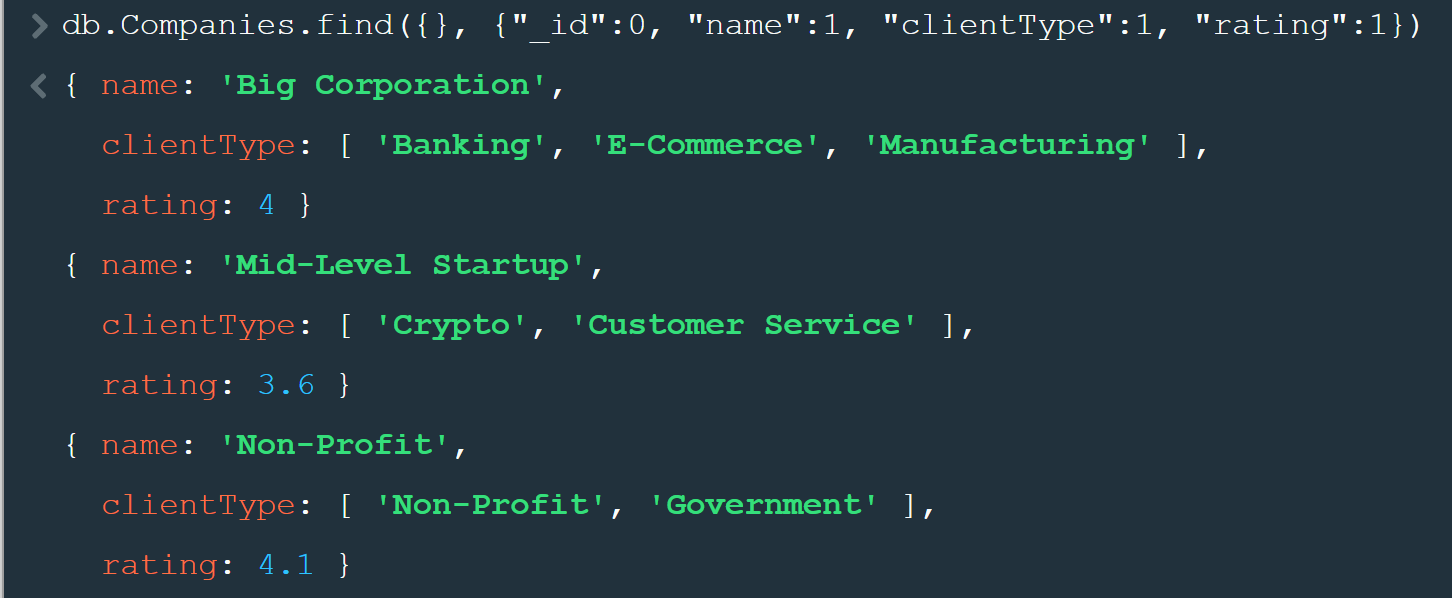
We have to use the $SET operation



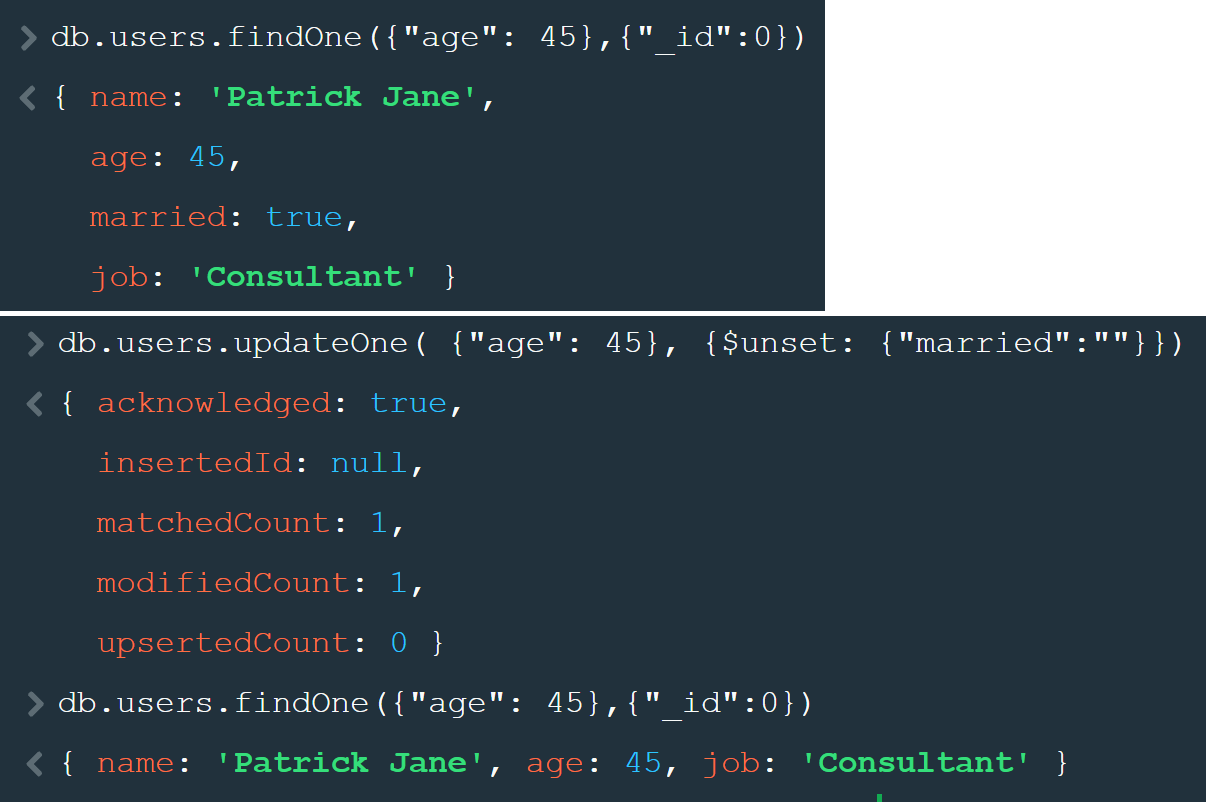
To Increment Certain Values, use $inc :







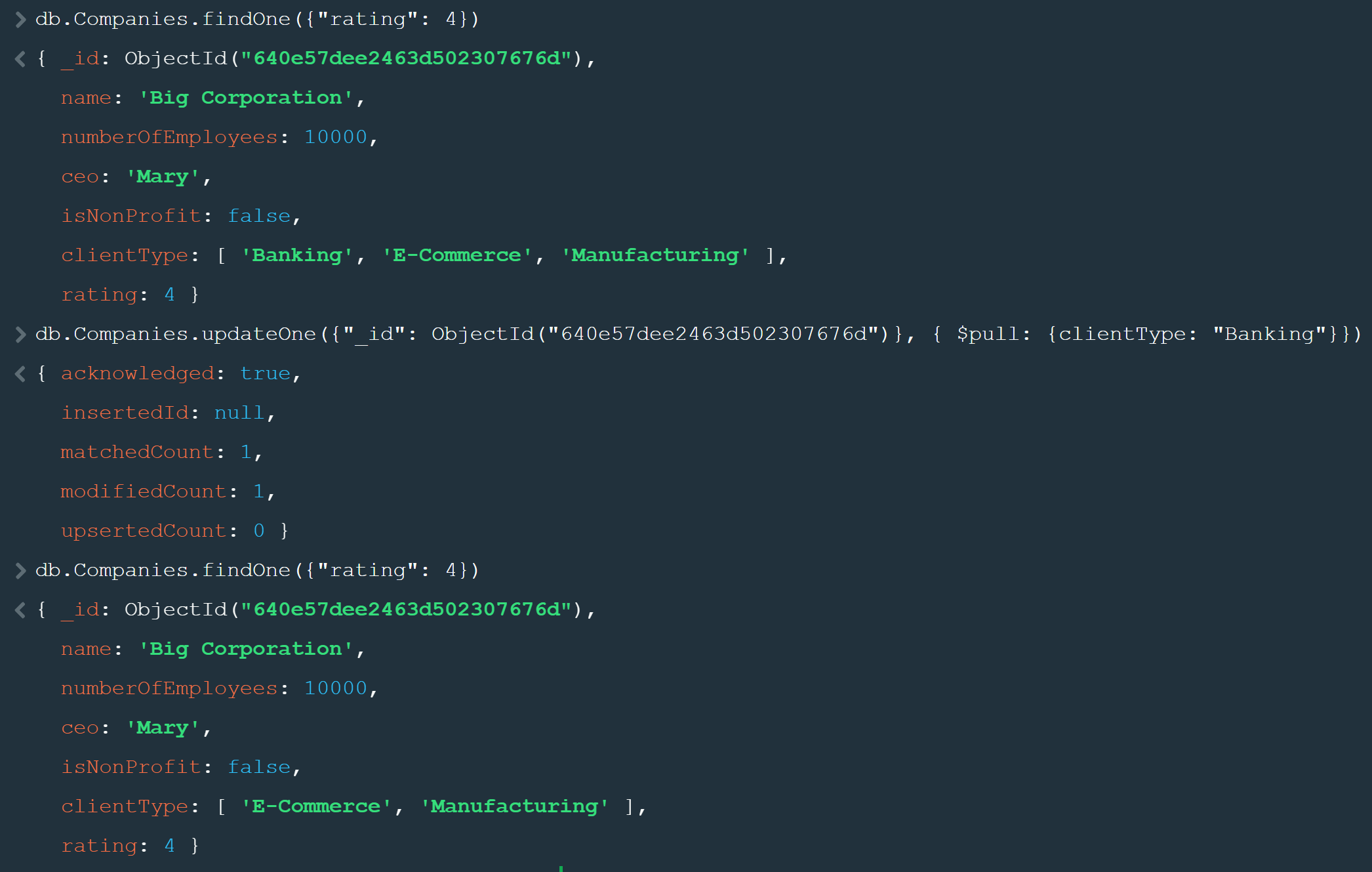
To Remove a Property Use $Unset



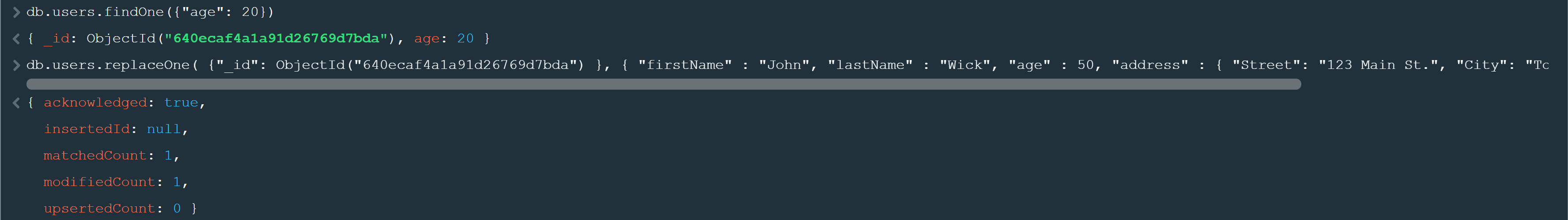
To Update the Array values we can Push data (ADD) to it:

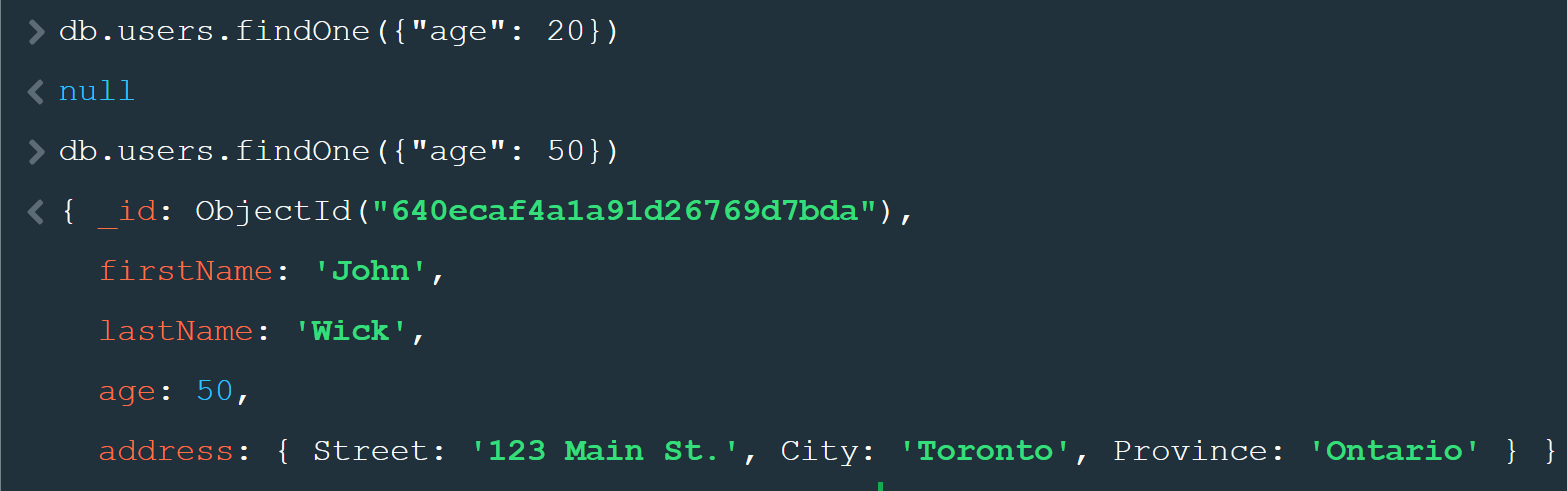


Tp Remove the Data we can pull from it



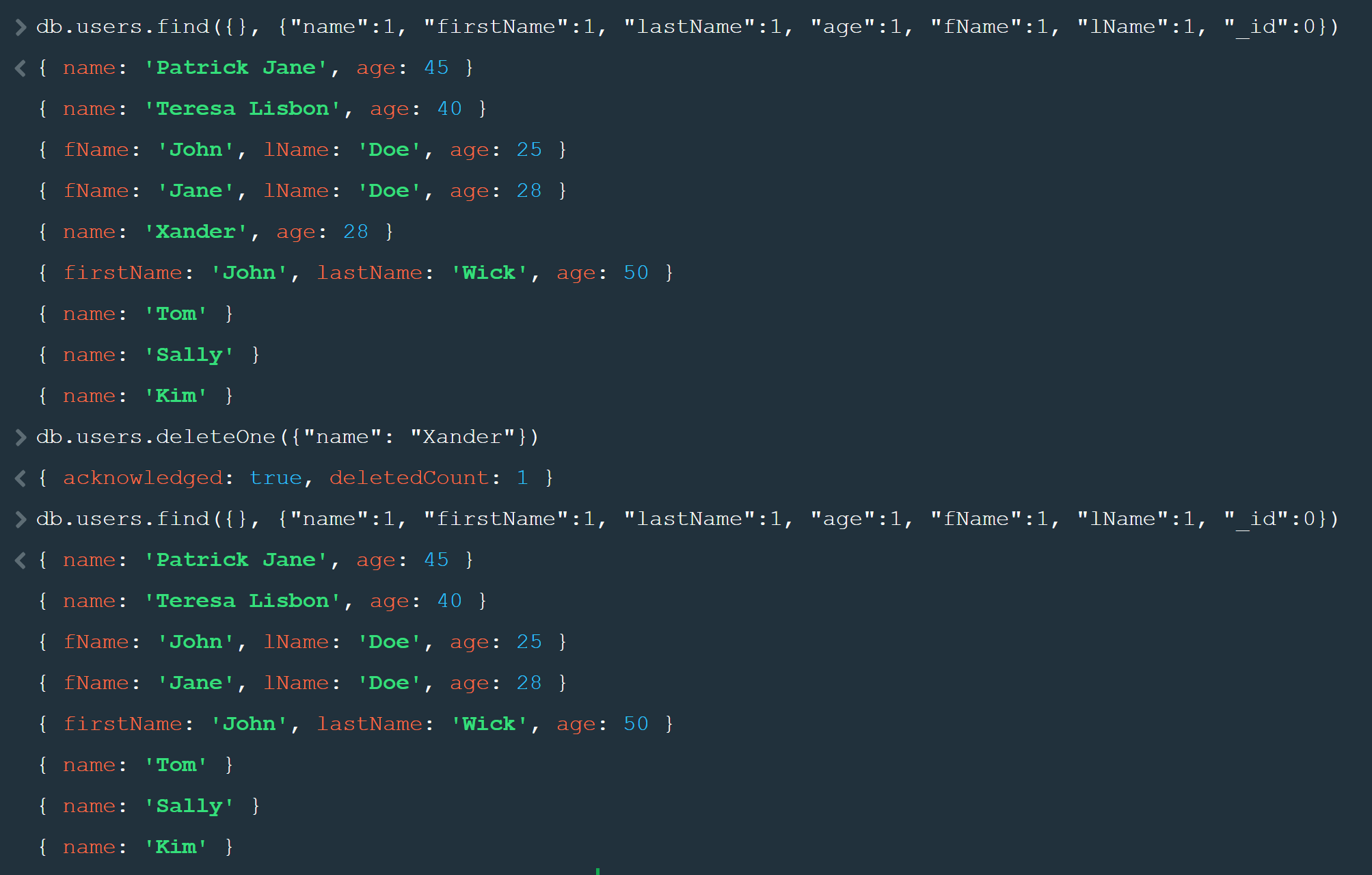
Replace an entire document with the specified JSON Object

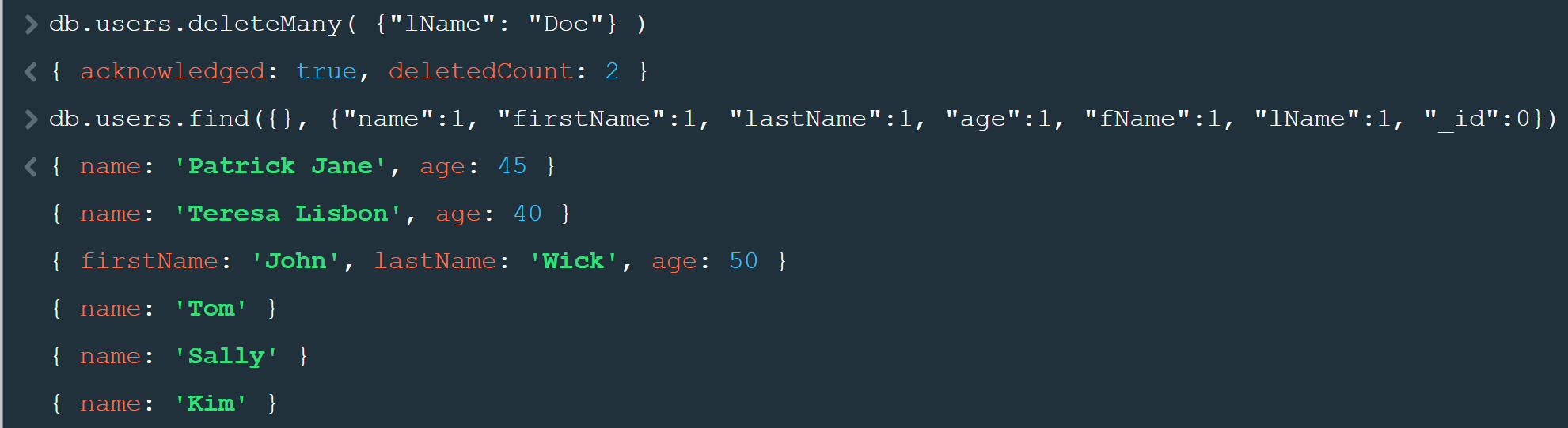




Generally, We use Update not replace

To Delete Record we use deleteOne() or deleteMany() Methods





Aggregation Pipelines:

Follow along the example on below Page :

[Aggregation Pipeline — MongoDB Manual](https://www.mongodb.com/docs/manual/core/aggregation-pipeline/)