PUNE INSTITUTE OF COMPUTER TECHNOLOGY

Subject: ADBMS (LP LAB)

Name: Aditya Kangune Roll No.: 33323

Batch: K11 Academic Year: 2021-22

Assignment 1
MONGO CRUD OPERATIONS

Aim:

Create a database with suitable examples using MongoDB and implement:

- Inserting and saving document (batch insert, insert validation).
- Removing document
- Updating document (document replacement, using modifiers, up inserts, updating multiple documents, returning updated documents)
- Execute at least 10 queries on any suitable MongoDB database that demonstrates the following:
    - Find and find One (specific values)
    - Query criteria (Query conditionals, OR queries, $not, Conditional semantics)
    - Type-specific queries (Null, Regular expression, Querying arrays)
    - $ where queries
    - Cursors (Limit, skip, sort, advanced query options)

Objective:

Following are the objectives:

1. To understand MongoDB basic commands.
2. To implement the concept of document-oriented databases.
3. To understand MongoDB retrieval commands.

Theory:

Comparison between MongoDB and SQL concepts:

SQL concepts	MongoDB concepts
Table	Collection
Row	Document
Column	Field
Table Join	Embedded Documents
Primary Key	Primary Key
Specify any Unique column as a Primary key	_Id field is the primary key
Aggregation (Group By)	Aggregation Pipeline

SQL Database	NoSQL Database (MongoDB)
Relational database	Non-relational database
Supports SQL query language	Supports JSON query language
Table based	Collection based and key-value pair
Row based	Document based
Column based	Field based
Support foreign key	No support for foreign key
Support for triggers	No Support for triggers
Contains schema which is predefined	Contains dynamic schema
Not fit for hierarchical data storage	Best fit for hierarchical data storage

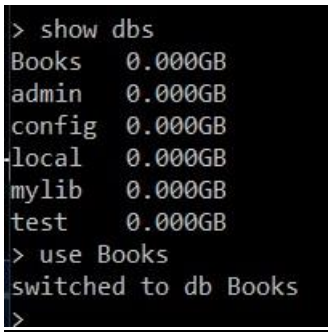
Vertically scalable - increasing RAM	Horizontally scalable - add more servers
Emphasizes on ACID properties (Atomicity, Consistency, Isolation and Durability)	Emphasizes on CAP theorem (Consistency, Availability and Partition tolerance)

Commands:

// Opening MongoDB in windows command shell:

i) Download MongoDB and install it.
ii) Create a new path variable in environment variables to use mongo shell anywhere in the device.
iii) Use commands:
(1) mongod
(2) mongo

// Create commands:



> show collections
Book store
> db.createCollection(Books)

// Shows current database

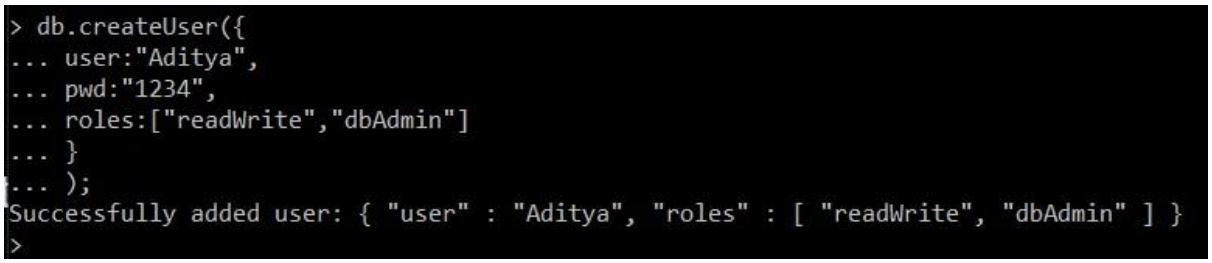
{ "ok" : 1 }
> show collections
Book store
Books
> use Books
switched to db Books
> db.Books.find().pretty()

> db.Books.insertMany([{name:"Harry Potter", author:"JK Rowling", id:123456789, price:490, pages:370}, {name:"The mortal instruments", author:"Cassandra Clare", id:4453356543, price:260, pages:290}, {name:"Who will cry when you die", id:7656445678, price:240, pages:120}, {name:"An Autobiography of a Yogi", id:8768905678, author:"Paramhansa Yogananda", price:390, pages:560}, {name:"The Subtle art of not giving a *uck", author:"Mark Manson", id:2574103698, price:450, pages:230}, {name:"Who Moved My Cheese?", author:"Spencer Johnson", id:9865741520, price:345, pages:180}])

```
{
	"acknowledged" : true,
	"insertedIds" : [
		ObjectId("611fa8bffc6e6eecbe413420"),
		ObjectId("611fa8bffc6e6eecbe413421"),
		ObjectId("611fa8bffc6e6eecbe413422"),
		ObjectId("611fa8bffc6e6eecbe413423"),
		ObjectId("611fa8bffc6e6eecbe413424"),
		ObjectId("611fa8bffc6e6eecbe413425")
	]
}
```

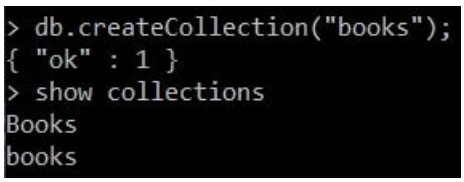
// Adding user

```
> db.createUser({
... user:"Aditya",
... pwd:"1234",
... roles:["readWrite","dbAdmin"]
... }
... );
Successfully added user: { "user" : "Aditya", "roles" : [
"readWrite", "dbAdmin" ] }
```



// Creating new collection(like tables in NoSQL)

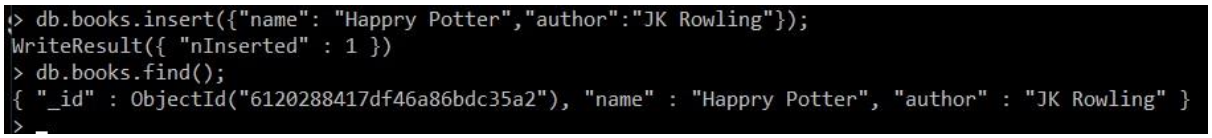
```
> db.createCollection("books");
{ "ok" : 1 }
> show collections
Books
books
```



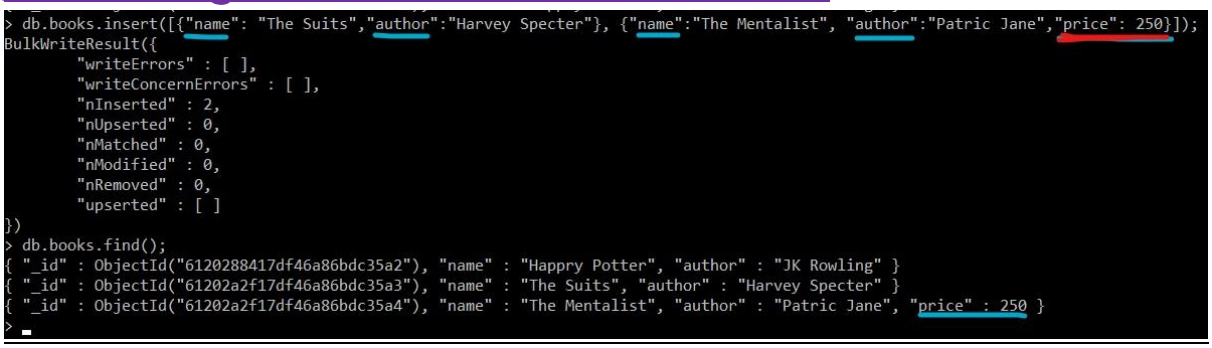
// Inserting one record and viewing it

// It automatiacally adds _id, which is a unique value so we don't have to worry about auto increament etc like we do in SQL

> db.books.insert({"name": "Happry Potter","author":"JK Rowling"});
WriteResult({ "nInserted" : 1 })
> db.books.find();
{ "_id" : ObjectId("6120288417df46a86bdc35a2"), "name" : "Happry Potter", "author" : "JK Rowling" }



// Inserting a record with an extra field:



// Using pretty to get cleaner output

// Output after create commands

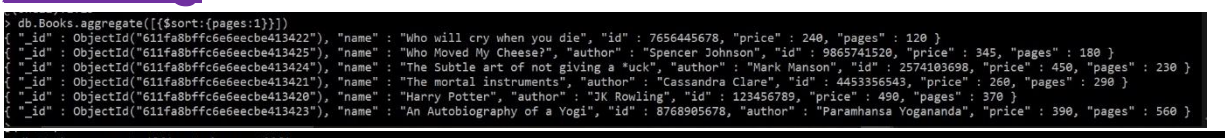
```
> db.Books.find().pretty()
{
        "_id" : ObjectId("611fa8bffc6e6eecbe413420"),
        "name" : "Harry Potter",
        "author" : "JK Rowling",
        "id" : 123456789,
        "price" : 490,
        "pages" : 370
}
{
        "_id" : ObjectId("611fa8bffc6e6eecbe413421"),
        "name" : "The mortal instruments",
        "author" : "Cassandra Clare",
        "id" : 4453356543,
        "price" : 260,
        "pages" : 290
}
{
        "_id" : ObjectId("611fa8bffc6e6eecbe413422"),
        "name" : "Who will cry when you die",
        "id" : 7656445678,
        "price" : 240,
        "pages" : 120
}
{
        "_id" : ObjectId("611fa8bffc6e6eecbe413423"),
        "name" : "An Autobiography of a Yogi",
        "id" : 8768905678,
        "author" : "Paramhansa Yogananda",
        "price" : 390,
        "pages" : 560
}
{
        "_id" : ObjectId("611fa8bffc6e6eecbe413424"),
        "name" : "The Subtle art of not giving a *uck",
        "author" : "Mark Manson",
        "id" : 2574103698,
        "price" : 450,
        "pages" : 230
}
{
        "_id" : ObjectId("611fa8bffc6e6eecbe413425"),
        "name" : "Who Moved My Cheese?",
        "author" : "Spencer Johnson",
        "id" : 9865741520,
        "price" : 345,
        "pages" : 180
}
```

```
>  db.Books.find({},{_id:0});
{ "name" : "Harry Potter", "author" : "JK Rowling", "id" : 123456789, "price" : 490, "pages" : 370 }
{ "name" : "The mortal instruments", "author" : "Cassandra Clare", "id" : 4453356543, "price" : 260, "pages" : 290 }
{ "name" : "Who will cry when you die", "id" : 7656445678, "price" : 240, "pages" : 120 }
{ "name" : "An Autobiography of a Yogi", "id" : 8768905678, "author" : "Paramhansa Yogananda", "price" : 390, "pages" : 560 }
{ "name" : "The Subtle art of not giving a *uck", "author" : "Mark Manson", "id" : 2574103698, "price" : 450, "pages" : 230 }
{ "name" : "Who Moved My Cheese?", "author" : "Spencer Johnson", "id" : 9865741520, "price" : 345, "pages" : 180 }
>
```

```
> db.Books.find({pages:{$lt:200}},{_id:0}).pretty();
{
        "name" : "Who will cry when you die",
        "id" : 7656445678,
        "price" : 240,
        "pages" : 120
}
{
        "name" : "Who Moved My Cheese?",
        "author" : "Spencer Johnson",
        "id" : 9865741520,
        "price" : 345,
        "pages" : 180
```

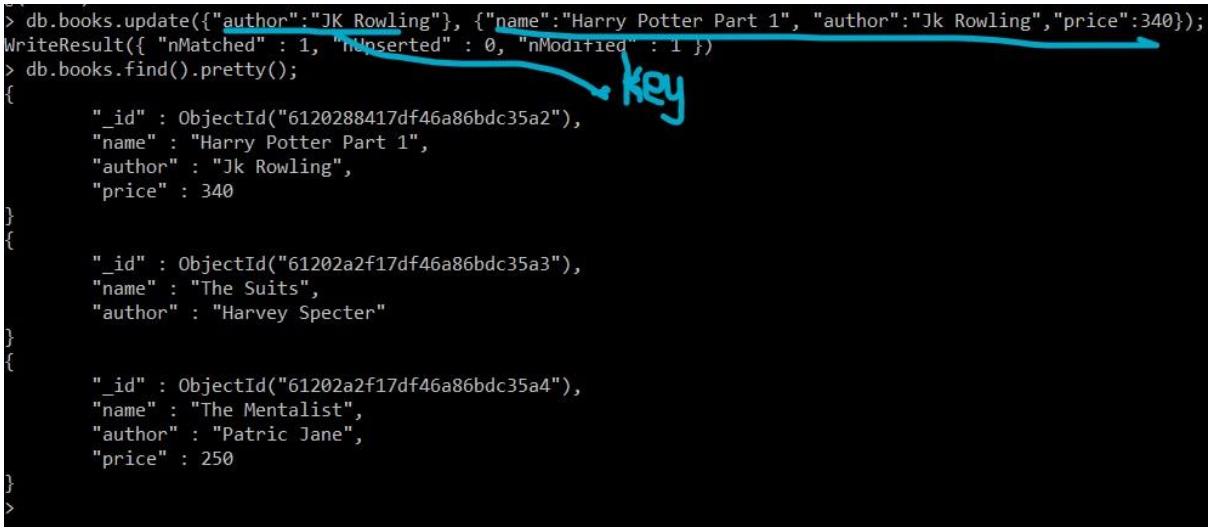
```
> db.Books.find({pages:{$gt:200}},{_id:0}).pretty();
{
        "name" : "Harry Potter",
        "author" : "JK Rowling",
        "id" : 123456789,
        "price" : 490,
        "pages" : 370
}
{
        "name" : "The mortal instruments",
        "author" : "Cassandra Clare",
        "id" : 4453356543,
        "price" : 260,
        "pages" : 290
}
{
        "name" : "An Autobiography of a Yogi",
        "id" : 8768905678,
        "author" : "Paramhansa Yogananda",
        "price" : 390,
        "pages" : 560
}
{
        "name" : "The Subtle art of not giving a *uck",
        "author" : "Mark Manson",
        "id" : 2574103698,
        "price" : 450,
        "pages" : 230
```

// Sorting




```
> db.Books.aggregate([{$sort:{pages:1}}])
{ "_id" : ObjectId("611fa8bffc6e6eecbe413422"), "name" : "Who will cry when you die", "id" : 7656445678, "price" : 240, "pages" : 120 }
{ "_id" : ObjectId("611fa8bffc6e6eecbe413425"), "name" : "Who Moved My Cheese?", "author" : "Spencer Johnson", "id" : 9865741520, "price" : 345, "pages" : 180 }
{ "_id" : ObjectId("611fa8bffc6e6eecbe413424"), "name" : "The Subtle art of not giving a *uck", "author" : "Mark Manson", "id" : 2574103698, "price" : 450, "pages" : 230 }
{ "_id" : ObjectId("611fa8bffc6e6eecbe413421"), "name" : "The mortal instruments", "author" : "Cassandra Clare", "id" : 4453356543, "price" : 260, "pages" : 290 }
{ "_id" : ObjectId("611fa8bffc6e6eecbe413420"), "name" : "Harry Potter", "author" : "JK Rowling", "id" : 123456789, "price" : 490, "pages" : 370 }
{ "_id" : ObjectId("611fa8bffc6e6eecbe413423"), "name" : "An Autobiography of a Yogi", "id" : 8768905678, "author" : "Paramhansa Yogananda", "price" : 390, "pages" : 560 }
>
```

// Updating by using “author” as key, and changing the name as well as adding a new field of “price” to it.




```
> db.books.update({"author":"JK Rowling"}, {"name":"Harry Potter Part 1", "author":"Jk Rowling","price":340});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.books.find().pretty();
{
        "_id" : ObjectId("6120288417df46a86bdc35a2"),
        "name" : "Harry Potter Part 1",
        "author" : "Jk Rowling",
        "price" : 340
}
{
        "_id" : ObjectId("61202a2f17df46a86bdc35a3"),
        "name" : "The Suits",
        "author" : "Harvey Specter"
}
{
        "_id" : ObjectId("61202a2f17df46a86bdc35a4"),
        "name" : "The Mentalist",
        "author" : "Patric Jane",
        "price" : 250
}
>
```

// Using $set to keep whatever was there previously and adding new field

```
> db.books.update({"name":"The Suits"}, {$set:{"price": 120}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.books.find().pretty();
{
        "_id" : ObjectId("6120288417df46a86bdc35a2"),
        "name" : "Harry Potter Part 1",
        "author" : "Jk Rowling",
        "price" : 340
}
{
        "_id" : ObjectId("61202a2f17df46a86bdc35a3"),
        "name" : "The Suits",
        "author" : "Harvey Specter",
        "price" : 120
}
{
        "_id" : ObjectId("61202a2f17df46a86bdc35a4"),
        "name" : "The Mentalist",
        "author" : "Patric Jane",
        "price" : 250
}
```

// Increameting price by 500

```
> db.books.update({"name":"The Suits"}, {$inc:{price: 500}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.books.find().pretty();
{
        "_id" : ObjectId("6120288417df46a86bdc35a2"),
        "name" : "Harry Potter Part 1",
        "author" : "Jk Rowling",
        "price" : 340
}
{
        "_id" : ObjectId("61202a2f17df46a86bdc35a3"),
        "name" : "The Suits",
        "author" : "Harvey Specter",
        "price" : 620
}
{
        "_id" : ObjectId("61202a2f17df46a86bdc35a4"),
        "name" : "The Mentalist",
        "author" : "Patric Jane",
        "price" : 250
}
```

// Using unset to remove a field

```
> db.books.update({"name":"The Suits"}, {$unset:{price:1}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.books.find().pretty();
{
        "_id" : ObjectId("6120288417df46a86bdc35a2"),
        "name" : "Harry Potter Part 1",
        "author" : "Jk Rowling",
        "price" : 340
}
{
        "_id" : ObjectId("61202a2f17df46a86bdc35a3"),
        "name" : "The Suits",
        "author" : "Harvey Specter"
}
{
        "_id" : ObjectId("61202a2f17df46a86bdc35a4"),
        "name" : "The Mentalist",
        "author" : "Patric Jane",
        "price" : 250
}
```

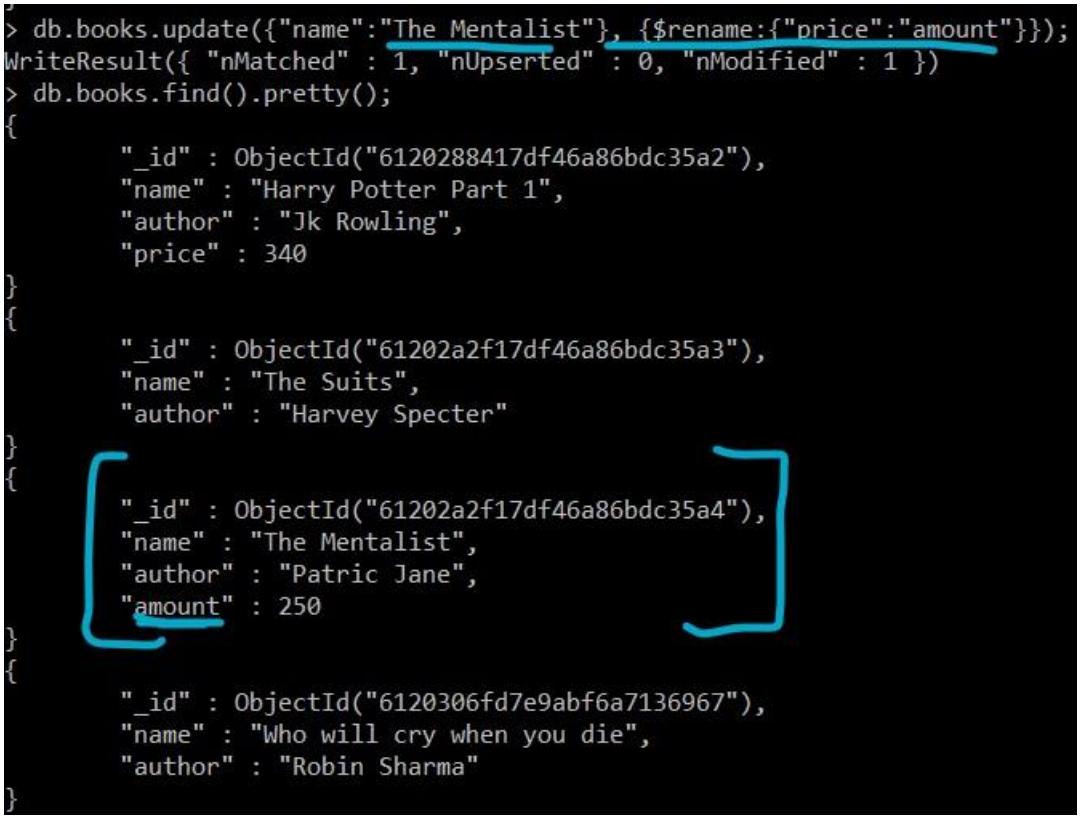
// Searching for an entry that is not there – We don't ger any error nor is the record created

```
> db.books.update({name:"Who will cry when you die"}, {name:"Who will cry when you die","author":"Robin Sharma"});
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
> db.books.find().pretty();
{
        "_id" : ObjectId("6120288417df46a86bdc35a2"),
        "name" : "Harry Potter Part 1",
        "author" : "Jk Rowling",
        "price" : 340
}
{
        "_id" : ObjectId("61202a2f17df46a86bdc35a3"),
        "name" : "The Suits",
        "author" : "Harvey Specter"
}
{
        "_id" : ObjectId("61202a2f17df46a86bdc35a4"),
        "name" : "The Mentalist",
        "author" : "Patric Jane",
        "price" : 250
}
>
```

// Creating a new entry if not found in the collection by setting upsert to true

```
> db.books.update({name:"Who will cry when you die"}, {name:"Who will cry when you die","author":"Robin Sharma"}, {upsert:true});
WriteResult({
        "nMatched" : 0,
        "nUpserted" : 1,
        "nModified" : 0,
        "_id" : ObjectId("6120306fd7e9abf6a7136967")
})
> db.books.find().pretty();
{
        "_id" : ObjectId("6120288417df46a86bdc35a2"),
        "name" : "Harry Potter Part 1",
        "author" : "Jk Rowling",
        "price" : 340
}
{
        "_id" : ObjectId("61202a2f17df46a86bdc35a3"),
        "name" : "The Suits",
        "author" : "Harvey Specter"
}
{
        "_id" : ObjectId("61202a2f17df46a86bdc35a4"),
        "name" : "The Mentalist",
        "author" : "Patric Jane",
        "price" : 250
}
{
        "_id" : ObjectId("6120306fd7e9abf6a7136967"),
        "name" : "Who will cry when you die",
        "author" : "Robin Sharma"
}
```

// Changing “price” to “amount” using rename



```
> db.books.update({"name":"The Mentalist"}, {$rename:{"price":"amount"}});
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.books.find().pretty();
{
        "_id" : ObjectId("6120288417df46a86bdc35a2"),
        "name" : "Harry Potter Part 1",
        "author" : "Jk Rowling",
        "price" : 340
}
{
        "_id" : ObjectId("61202a2f17df46a86bdc35a3"),
        "name" : "The Suits",
        "author" : "Harvey Specter"
}
{
        "_id" : ObjectId("61202a2f17df46a86bdc35a4"),
        "name" : "The Mentalist",
        "author" : "Patric Jane",
        "amount" : 250
}
{
        "_id" : ObjectId("6120306fd7e9abf6a7136967"),
        "name" : "Who will cry when you die",
        "author" : "Robin Sharma"
}
```

// Removing a particular record

```
> db.books.remove({"name":"The Suits"});
WriteResult({ "nRemoved" : 1 })
> db.books.find().pretty();
{
        "_id" : ObjectId("6120288417df46a86bdc35a2"),
        "name" : "Harry Potter Part 1",
        "author" : "Jk Rowling",
        "price" : 340
}
{
        "_id" : ObjectId("61202a2f17df46a86bdc35a4"),
        "name" : "The Mentalist",
        "author" : "Patric Jane",
        "amount" : 250
}
{
        "_id" : ObjectId("6120306fd7e9abf6a7136967"),
        "name" : "Who will cry when you die",
        "author" : "Robin Sharma"
}
>
```

//Removing only one record if there are multiple records with same name

```
> db.books.remove({"name":"The Suits"},{justOne:true});
WriteResult({ "nRemoved" : 0 })
```

// Adding nested records using arrays

```
> db.inventory.insertOne(
...     { item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } }
... )
{
        "acknowledged" : true,
        "insertedId" : ObjectId("61203c3617df46a86bdc35a6")
}
> db.inventory.find().pretty();
{
        "_id" : ObjectId("6120393c17df46a86bdc35a5"),
        "item" : "canvas",
        "qty" : 100,
        "tags" : [
                "cotton"
        ],
        "size" : {
                "h" : 28,
                "w" : 35.5,
                "uom" : "cm"
        }
}
{
        "_id" : ObjectId("61203c3617df46a86bdc35a6"),
        "item" : "canvas",
        "qty" : 100,
        "tags" : [
                "cotton"
        ],
        "size" : {
                "h" : 28,
                "w" : 35.5,
                "uom" : "cm"
        }
}
>
```

// Adding many records with nested arrays

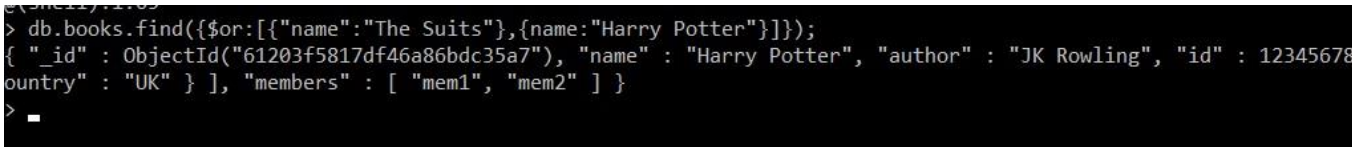
```
{
        "_id" : ObjectId("61203f5817df46a86bdc35a7"),
        "name" : "Harry Potter",
        "author" : "JK Rowling",
        "id" : 123456789,
        "price" : 490,
        "pages" : 370,
        "address" : [
                {
                        "street" : "Bakers Street",
                        "city" : "London",
                        "country" : "UK"
                }
        ],
        "members" : [
                "mem1",
                "mem2"
        ]
}
```

```
> db.books.find().pretty();
{
        "_id" : ObjectId("6120288417df46a86bdc35a2"),
        "name" : "Harry Potter Part 1",
        "author" : "Jk Rowling",
        "price" : 340
}
{
        "_id" : ObjectId("61202a2f17df46a86bdc35a4"),
        "name" : "The Mentalist",
        "author" : "Patric Jane",
        "amount" : 250
}
{
        "_id" : ObjectId("6120306fd7e9abf6a7136967"),
        "name" : "Who will cry when you die",
        "author" : "Robin Sharma"
}
{
        "_id" : ObjectId("61203f5817df46a86bdc35a7"),
        "name" : "Harry Potter",
        "author" : "JK Rowling",
        "id" : 123456789,
        "price" : 490,
        "pages" : 370,
        "address" : [
                {
                        "street" : "Bakers Street",
                        "city" : "London",
                        "country" : "UK"
                }
        ],
        "members" : [
                "mem1",
                "mem2"
```

```
        ]
}
{
        "_id" : ObjectId("61203f5817df46a86bdc35a8"),
        "name" : "The mortal instruments",
        "author" : "Cassandra Clare",
        "id" : 4453356543,
        "price" : 260,
        "pages" : 290,
        "address" : [
                {
                        "street" : "Times Square",
                        "city" : "New York",
                        "country" : "USA"
                }
        ],
        "members" : [
                "mem1",
                "mem2",
                "mem3"
        ]
}
{
        "_id" : ObjectId("61203f5817df46a86bdc35a9"),
        "name" : "Who will cry when you die",
        "id" : 7656445678,
        "price" : 240,
        "pages" : 120
}
{
        "_id" : ObjectId("61203f5817df46a86bdc35aa"),
        "name" : "An Autobiography of a Yogi",
        "id" : 8768905678,
        "author" : "Paramhansa Yogananda",
        "price" : 390,
        "pages" : 560
}
```

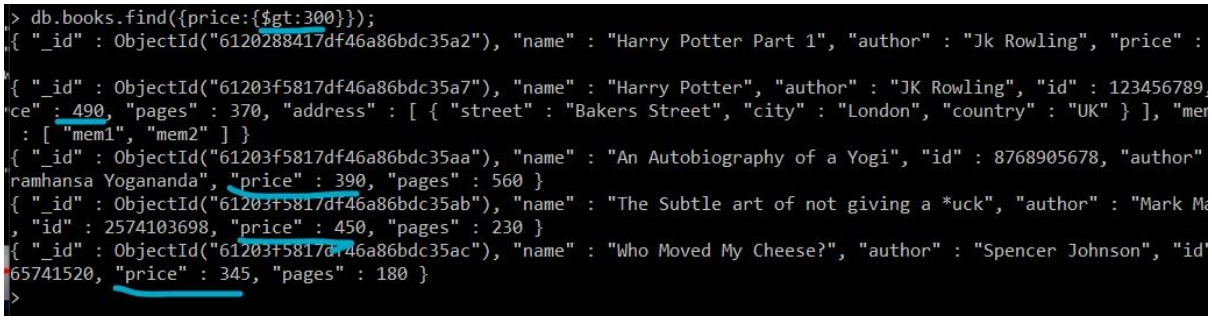
```
{
        "_id" : ObjectId("61203f5817df46a86bdc35ab"),
        "name" : "The Subtle art of not giving a *uck",
        "author" : "Mark Manson",
        "id" : 2574103698,
        "price" : 450,
        "pages" : 230
}
{
        "_id" : ObjectId("61203f5817df46a86bdc35ac"),
        "name" : "Who Moved My Cheese?",
        "author" : "Spencer Johnson",
        "id" : 9865741520,
        "price" : 345,
        "pages" : 180
}
>
```

// $or operator

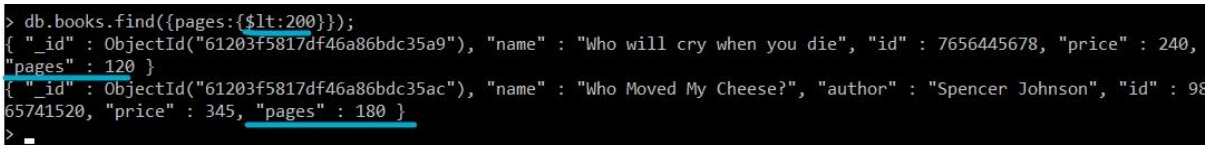



```
> db.books.find({$or:[{"name":"The Suits"},{name:"Harry Potter"}]});
{ "_id" : ObjectId("61203f5817df46a86bdc35a7"), "name" : "Harry Potter", "author" : "JK Rowling", "id" : 123456789, "price" : 490, "pages" : 370, "address" : [ { "street" : "Bakers Street", "city" : "London", "country" : "UK" } ], "members" : [ "mem1", "mem2" ] }
>
```

// Getting book with price greater than 300

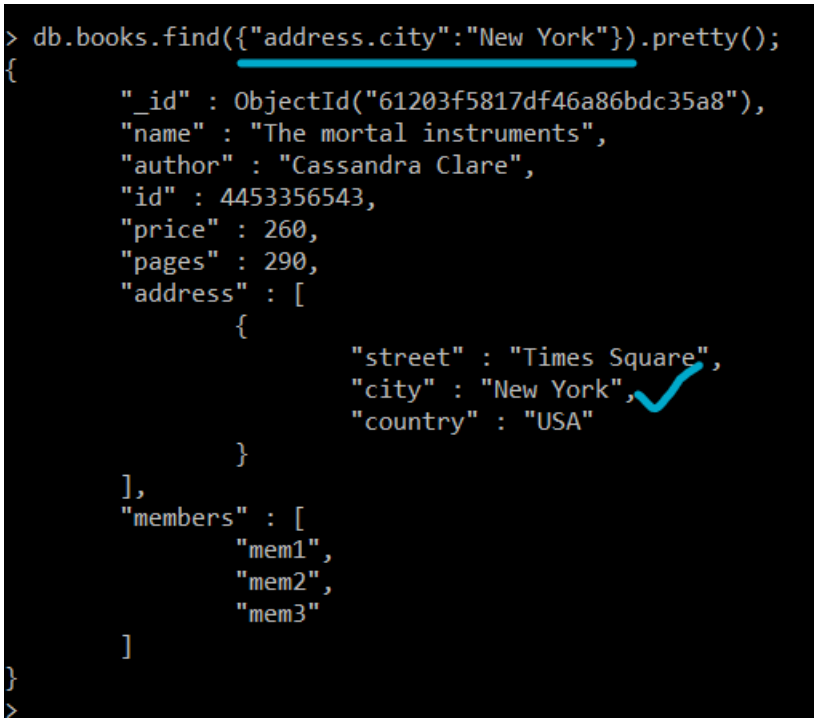


// Getting book with less than 200 pages

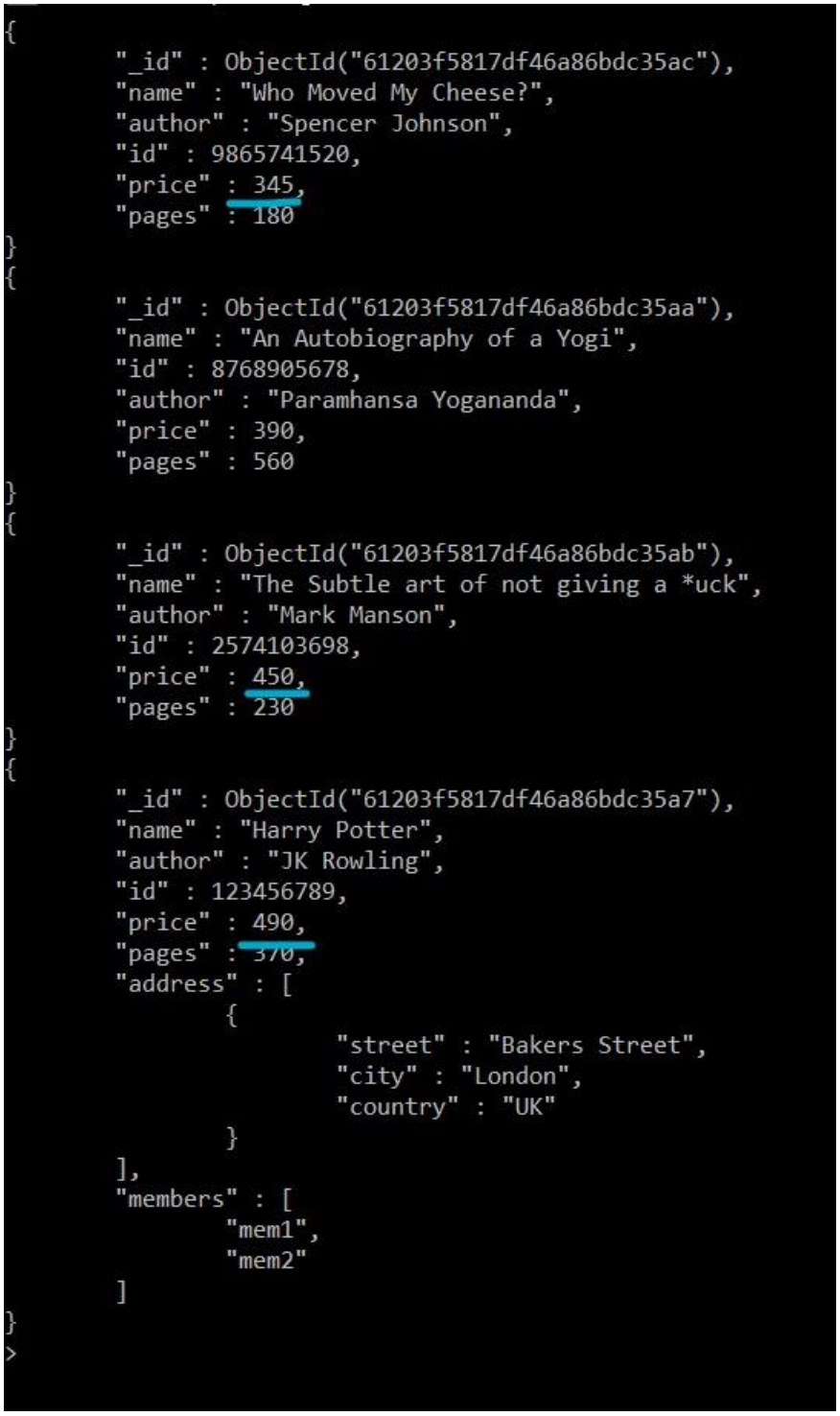


> db.books.find({pages:{$lt:200}});
{ "_id" : ObjectId("61203f5817df46a86bdc35a9"), "name" : "Who will cry when you die", "id" : 7656445678, "price" : 240, "pages" : 120 }
{ "_id" : ObjectId("61203f5817df46a86bdc35ac"), "name" : "Who Moved My Cheese?", "author" : "Spencer Johnson", "id" : 9865741520, "price" : 345, "pages" : 180 }
>

// Finding records inside a nested record(row)


> db.books.find({"address.city":"New York"}).pretty();
{
"_id" : ObjectId("61203f5817df46a86bdc35a8"),
"name" : "The mortal instruments",
"author" : "Cassandra Clare",
"id" : 4453356543,
"price" : 260,
"pages" : 290,
"address" : [
{
"street" : "Times Square",
"city" : "New York",
"country" : "USA"
}
],
"members" : [
"mem1",
"mem2",
"mem3"
]
}
>

// Sorting all the books according to price(ascending)



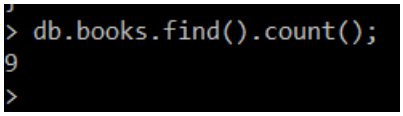

```
> db.books.find().sort({price:1}).pretty();
{
	"_id" : ObjectId("61202a2f17df46a86bdc35a4"),
	"name" : "The Mentalist",
	"author" : "Patric Jane",
	"amount" : 250
}
{
```

```
        "_id" : ObjectId("6120306fd7e9abf6a7136967"),
        "name" : "Who will cry when you die",
        "author" : "Robin Sharma"
}
{
        "_id" : ObjectId("61203f5817df46a86bdc35a9"),
        "name" : "Who will cry when you die",
        "id" : 7656445678,
        "price" : 240,
        "pages" : 120
}
{
        "_id" : ObjectId("61203f5817df46a86bdc35a8"),
        "name" : "The mortal instruments",
        "author" : "Cassandra Clare",
        "id" : 4453356543,
        "price" : 260,
        "pages" : 290,
        "address" : [
                {
                        "street" : "Times Square",
                        "city" : "New York",
                        "country" : "USA"
                }
        ],
        "members" : [
                "mem1",
                "mem2",
                "mem3"
        ]
}
{
        "_id" : ObjectId("6120288417df46a86bdc35a2"),
        "name" : "Harry Potter Part 1",
        "author" : "Jk Rowling",
        "price" : 340
}
```

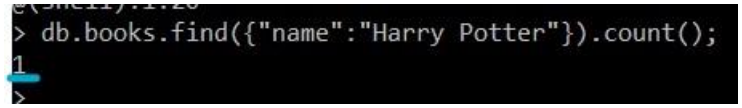
```
{
	"_id" : ObjectId("61203f5817df46a86bdc35ac"),
	"name" : "Who Moved My Cheese?",
	"author" : "Spencer Johnson",
	"id" : 9865741520,
	"price" : 345,
	"pages" : 180
}
{
	"_id" : ObjectId("61203f5817df46a86bdc35aa"),
	"name" : "An Autobiography of a Yogi",
	"id" : 8768905678,
	"author" : "Paramhansa Yogananda",
	"price" : 390,
	"pages" : 560
}
{
	"_id" : ObjectId("61203f5817df46a86bdc35ab"),
	"name" : "The Subtle art of not giving a *uck",
	"author" : "Mark Manson",
	"id" : 2574103698,
	"price" : 450,
	"pages" : 230
}
{
	"_id" : ObjectId("61203f5817df46a86bdc35a7"),
	"name" : "Harry Potter",
	"author" : "JK Rowling",
	"id" : 123456789,
	"price" : 490,
	"pages" : 370,
	"address" : [
		{
			"street" : "Bakers Street",
			"city" : "London",
			"country" : "UK"
		}
```

```
        ],
        "members" : [
                "mem1",
                "mem2"
        ]
}
>
```

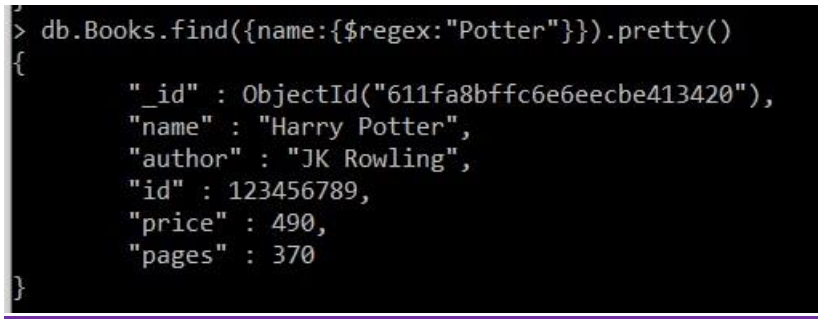
// Using count() function



// Counting occurrence of a specific record



// Using regex to find the word “Potter”



// Using validators

```
> db.createCollection("Books1", {
...    validator: {
...       $jsonSchema: {
...          bsonType: "object",
...          required: [ "name", "author", "pages", "price" ],
...          properties: {
...             name: {
...                bsonType: "string",
...                description: "must be a string and is required"
...             },
...             pages: {
...                bsonType: "int",
...                minimum: 50,
...                maximum: 1000,
...                description: "must be an integer in [ 50, 1000 ] and is required"
...             },
...             price: {
...               bsonType: "int",
...               minimum: 0,
...               description: "Required and should be greater than 0"
...             },
...          }
...       }
...    }
... });
{ "ok" : 1 }
```

```
> db.createCollection("Books1", {
...   validator: {
...     $jsonSchema: {
...       bsonType: "object",
...       required: [ "name", "author", "pages", "price" ],
...       properties: {
...         name: {
...           bsonType: "string",
...           description: "must be a string and is required"
...         },
...         pages: {
...           bsonType: "int",
...           minimum: 50,
...           maximum: 1000,
...           description: "must be an integer in [ 50, 1000 ] and is required"
...         },
...         price: {
...           bsonType: "int",
...           minimum: 0,
...           description: "Required and should be greater than 0"
...         },
...       }
...     }
...   }
```

... });
{ "ok" : 1 }

// Using push, pop for arrays:

BEFORE push:

AFTER push:

Conclusion:

- Commands for insert, saving removing document, updating document were executed successfully.
- Queries were executed on a “book” database such as find, find one, conditions, OR, $ not, conditional semantics, Type-specific queries (Null, Regular expression, Querying arrays), $ where, cursors (Limit, skip, sort, advanced query options), etc.