**Assignment 2**

Please work in **groups** to complete this assignment. This assignment is worth **30% of the total course grade** and will be evaluated through your written submission. Each day being late will result in 10% mark penalty.

Please submit the following files through Blackboard:

* Assign2.docx

1. Add this declaration on the top of your Assign2.docx file.

We, ------------ (mention your names), declare that the attached assignment is our own work in accordance with the Seneca Academic Policy. No part of this assignment has been copied manually or electronically from any other source (including web sites) **or distributed to other students.**

1. Specify what each member has done towards the completion of this work:

|  |  |  |
| --- | --- | --- |
|  | Name | Task(s) |
| 1 | Trung Kien Phan | Question 01 - 08 |
| 2 | Hiu Fung Chan | Question 09 - 16 |
| 3 | Henry Lau | Question 17 - 24 |

# Importing the Database

You are to create a new mongodb database named “booksdb” by importing books.json by running the following command:

1. Open a command prompt
2. Type cd c:\Program Files\MongoDB\mongodb-database-tools\bin

You may use a different path depending on where you have mongoimport executable files.

1. Type the following command to import the books json file into a mongodb database.
2. mongoimport.exe "c:\Program Files\MongoDB\Mongo\_Import\_Export\books.json" -d booksdb -c books --jsonArray

the following output will display:

2021-06-22T21:20:52.713-0400 connected to: mongodb://localhost/

2021-06-22T21:20:52.744-0400 18 document(s) imported successfully. 0 document(s) failed to import.

1. Make sure the database is created. Go to mongodb terminal, type the following commands:

use booksdb

db.books.findOne()

A document will display in the output.

Instructions

Answer each of the following questions with a MongoDB command, and the corresponding output. Add the screenshot as well below each question.

# marking:

12 questions, 3pts each. Total: 36pts

# Part 1: create and update

1. Use insert() to insert the following book in the books collection.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| \_id | title | isbn | publishedDate | authors | categories | publisher |
| 21 | MongoDB: The Definitive Guide | 9781491954461 | 2019-12-01 | Kristina Chodorow | Next Generation Databases | O'Reilly Media, Inc. |

Add a field named pageCount equal 544 to book id 21. Use the $set modifier.

Show book id 21 after the update.

Command: db.books.insertOne({

"\_id": 21,

"title": "MongoDB: The Definitive Guide",

"isbn": "9781491954461",

"publishedDate": new Date("2019-12-01"),

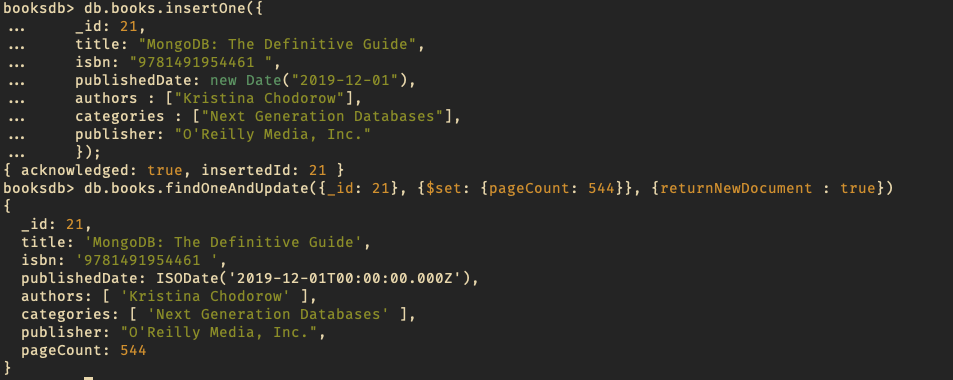
"authors": ["Kristina Chodorow"],

"categories": ["Next Generation Databases"],

"publisher": "O'Reilly Media, Inc."

})

db.books.findOneAndUpdate({\_id: 21}, {$set: {pageCount: 544}}, {returnNewDocument : true})



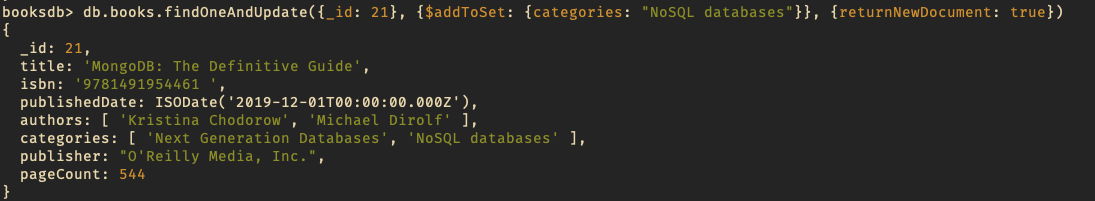
1. Add the author “Michael Dirolf” to book 21. Use the $push modifier.

Show book id 21 after the update.

Command: db.books.findOneAndUpdate({\_id: 21}, {$push: {authors: "Michael Dirolf"}}, {returnNewDocument: true})

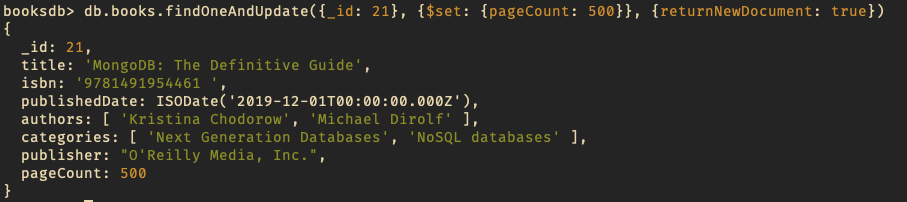
1. Add category category “NoSQL databases” to book id 21. Use $addToSet modifier.

Show book id 21 after the update.

Command: db.books.findOneAndUpdate({\_id: 21}, {$addToSet: {categories: "NoSQL databases"}}, {returnNewDocument: true})  


1. Update the pageCount for bookid 21 to 500.

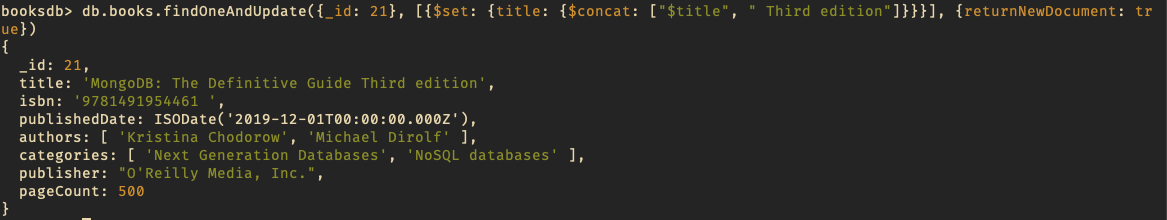
Show book id 21 after the update.

Command: db.books.findOneAndUpdate({\_id: 21}, {$set: {pageCount: 500}}, {returnNewDocument: true})  


1. Update the title for book id 21 by adding the text “Third edition”. Use update() function, the first argument is to identify the document to update, the second argument holds the update. For the second argument, use aggregation pipeline and enclose it with square brackets []. use $set and $concat. $title holds the current value in the key “title”.

Show book id 21 after the update.

reference: <https://stackoverflow.com/questions/23868963/append-a-string-to-the-end-of-an-existing-field-in-mongodb>

Command: db.books.findOneAndUpdate({\_id: 21}, [{$set: {title: {$concat: ["$title", " Third edition"]}}}], {returnNewDocument: true})  


db.books.findOne({ "\_id": 21 })

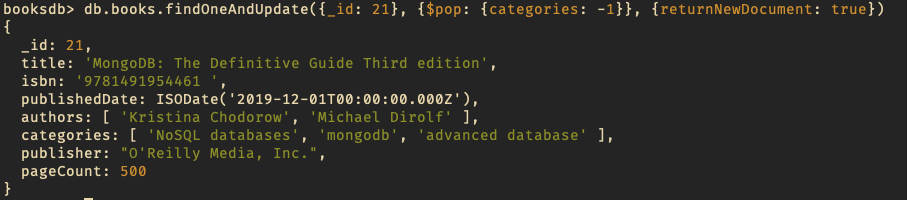
1. Push the following categories “mongodb”, ‘advanced databases” in categories field for book id 21. Use $push and $each modifiers.

Show book id 21 after the update.

Command: db.books.findOneAndUpdate({\_id: 21}, {$push: {categories: {$each: ["mongodb", "advanced database"]}}}, {returnNewDocument: true})  


1. Remove the first category value in the categories array for book id 21. Use $pop modifier.

Show book id 21 after the update.

Command: db.books.findOneAndUpdate({\_id: 21}, {$pop: {categories: -1}}, {returnNewDocument: true})  


1. Pull the element “advanced DB” from the categories array for book id 21.

Show book id 21 after the update.

Command: db.books.findOneAndUpdate({\_id: 21}, {$pull: {categories: "advanced database"}}, {returnNewDocument: true})  


1. Roll a new field named “pubyear” and assign its value by extracting the year from the published date for book id 21. Use update() function, the second argument should be enclosed with square brackets to refer to aggregation pipeline.

Command:

db.books.update( { \_id: 21 }, [ { $set: { pubyear: { $year: "$publishedDate" } } } ] )

Output:

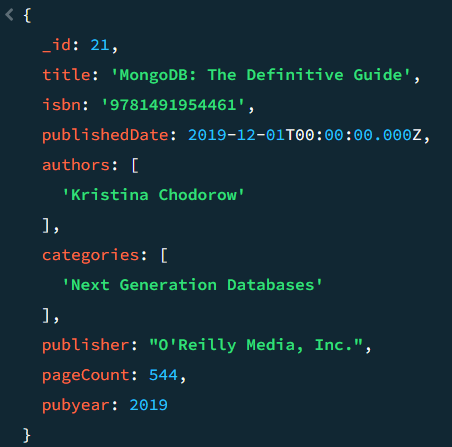
A screen shot of a computer code

Description automatically generated

Command to show the updated document:

db.books.findOne({ \_id: 21 })

Output:



1. Roll a new field named “pubyear” and assign its value by extracting the year from the published date for all books. Use update() function, the second argument should be enclosed with square brackets to refer to aggregation pipeline.

Command:

db.books.update( {}, [ { $set: { pubyear: { $year: "$publishedDate" } } } ], { multi: true } )

Output:

A screen shot of a computer code

Description automatically generated

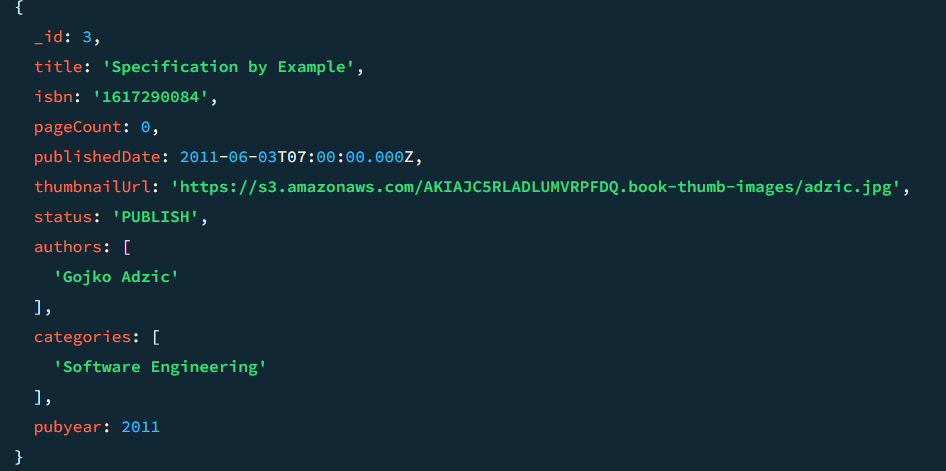
Command to show a few updated documents:

db.books.find().limit(5).pretty()

Output:

A computer screen shot of a computer code

Description automatically generated with medium confidence

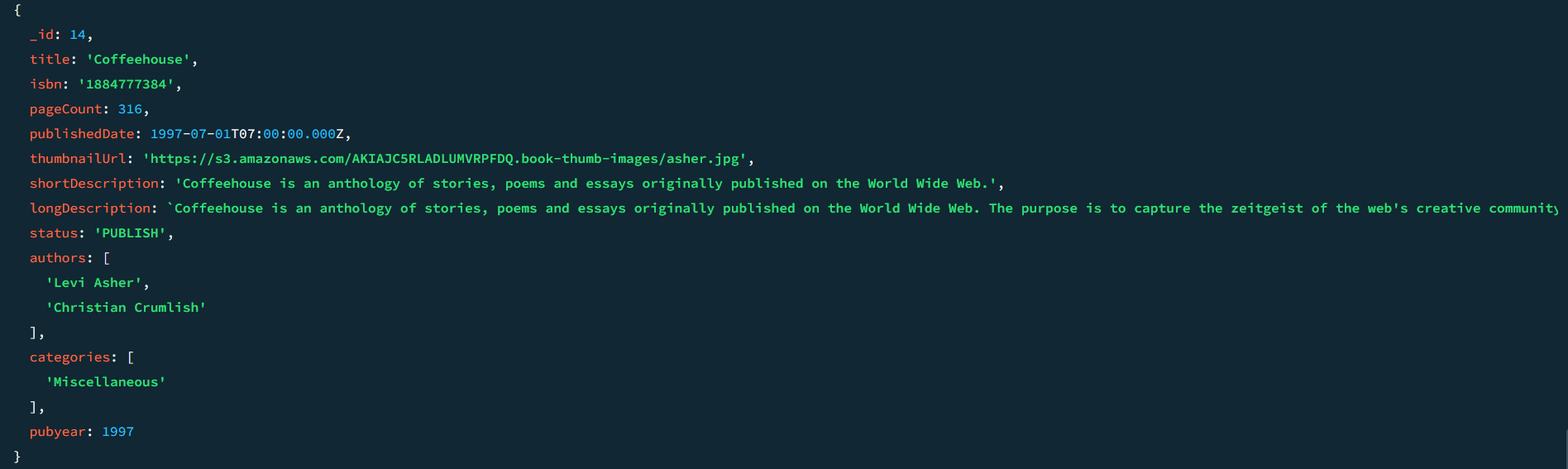


A computer code with green text

Description automatically generated

A screen shot of a computer

Description automatically generated



1. Remove the long description fields from all books.

Command:

db.books.update( {}, { $unset: { longDescription: "" } }, { multi: true } )

Output:

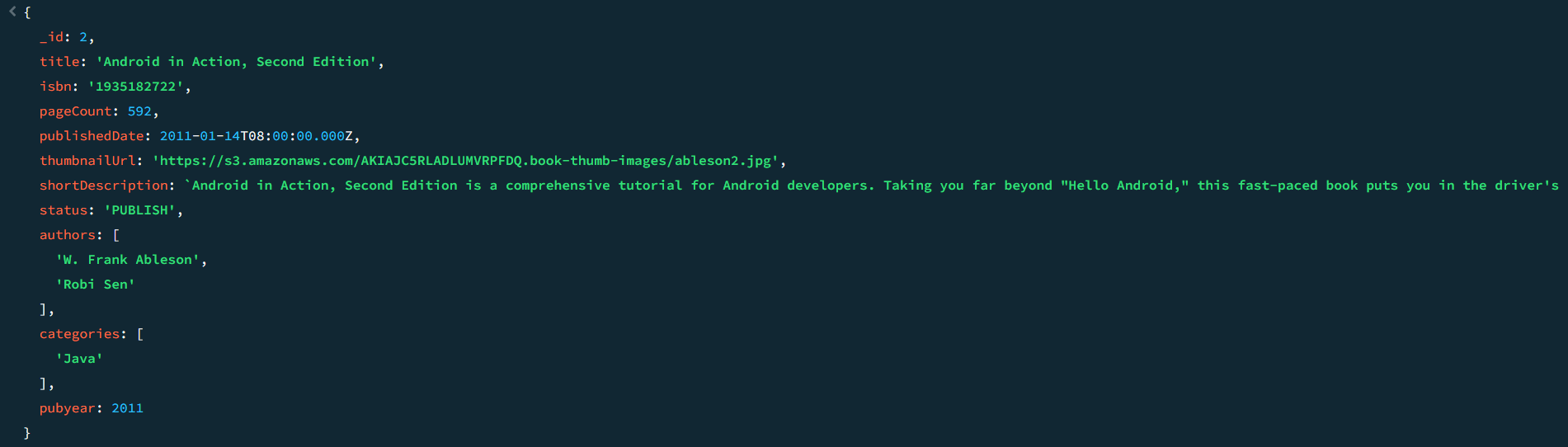
A screen shot of a computer code

Description automatically generated

Command to show a few of updated documents:

db.books.find().limit(5).pretty()

Output:



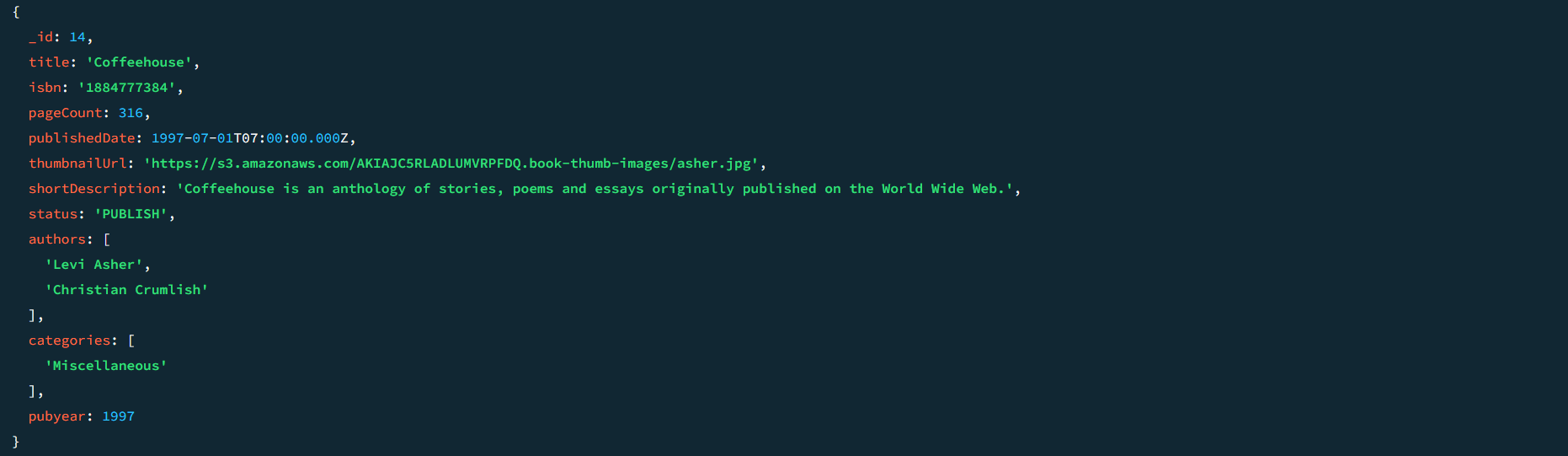
A computer screen shot of a computer screen

Description automatically generated



A screenshot of a computer

Description automatically generated



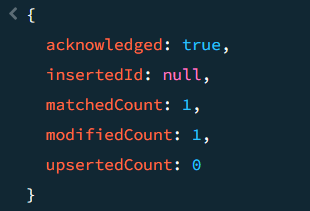
1. Add an array comment to book ids 19, 21. Push the two comments for each book together in one update command. Show documents 19, 21 after the update.

|  |  |
| --- | --- |
| Book id | comments |
| 19 | Name: "Joe"  Content: "nice book. Practice questions are helpful"  rating: 5 |
| Name: "David"  Content: "excellent book. Contains illustratives"  rating: 4 |
| 21 | Name: "Joe"  Content: "nice book."  Rating: 4 |
| Name: "Sam"  Content: "excellent book. Contains illustratives"  rating: 5 |

Command:

db.books.update( { \_id: { $in: [19, 21] } }, { $push: { comments: { $each: [ { name: "Joe", content: "nice book. Practice questions are helpful", rating: 5 }, { name: "David", content: "excellent book. Contains illustratives", rating: 4 } ] } } }, { multi: true } ) db.books.update( { \_id: 21 }, { $push: { comments: { $each: [ { name: "Joe", content: "nice book.", rating: 4 }, { name: "Sam", content: "excellent book. Contains illustratives", rating: 5 } ] } } } )

Output:



Command:

db.books.find({ \_id: { $in: [19, 21] } }).pretty()

Output:

A green lines on a dark background

Description automatically generated

A blue rectangle with white dots

Description automatically generated

# Part 2: Query

# marking:

12 questions, 2pts each. Total: 24pts

1. Find the number of books in the collection.

Command:

db.books.countDocuments()

Output:

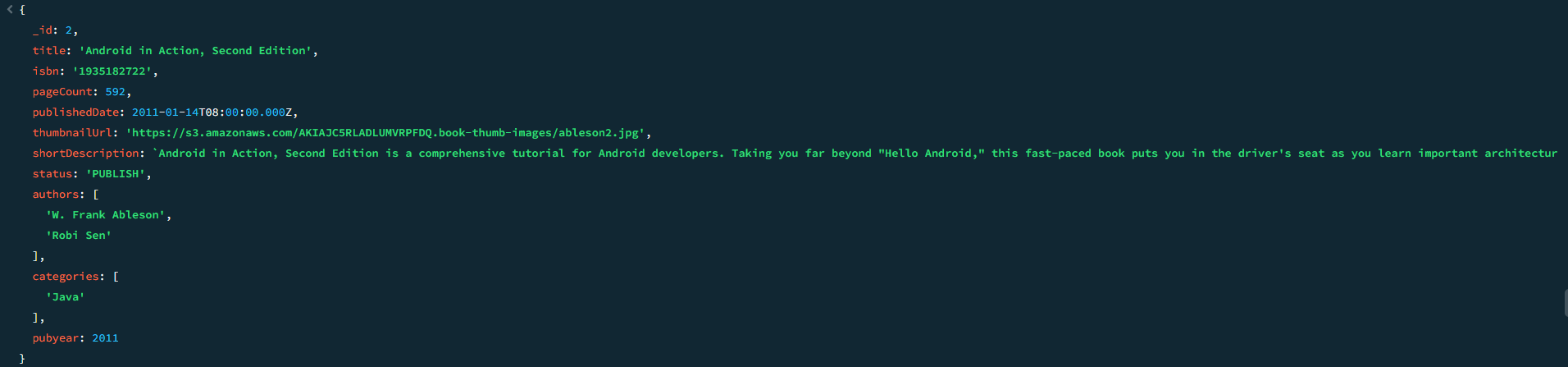


1. Find the books that have pagecount greater than 100.

Command:

db.books.find({ pageCount: { $gt: 100 } }).pretty()

Output:



A green lines on a dark background

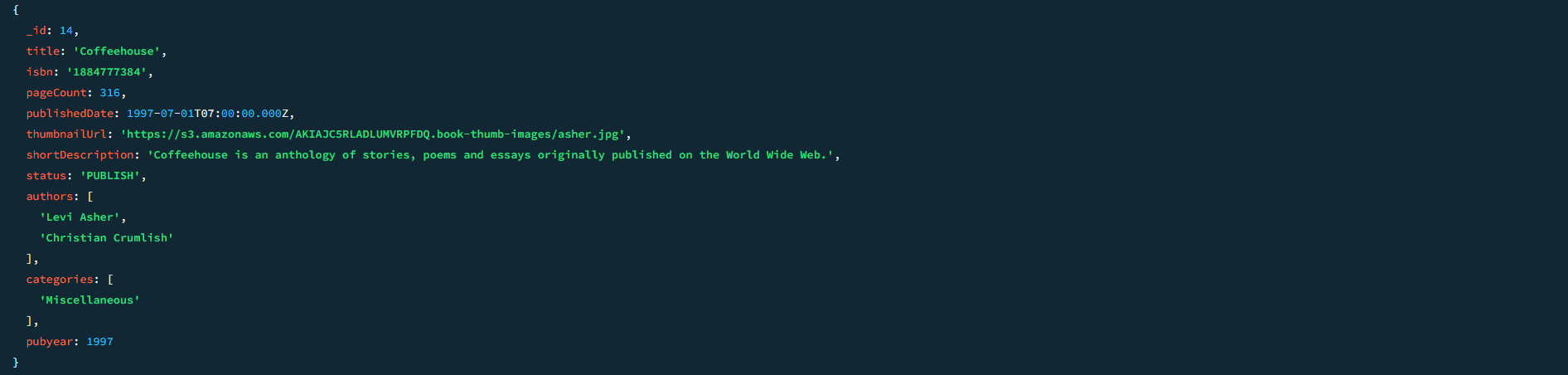
Description automatically generated

A blue and green screen

Description automatically generated with medium confidence

A blue background with green text

Description automatically generated



A green dot in the sky

Description automatically generated with medium confidence

A green and black background

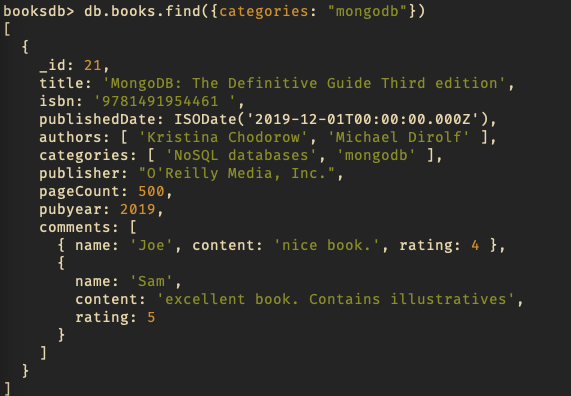
Description automatically generated with medium confidence

1. Find the books that belong to mongodb category.

Command:

db.books.find({ categories: "mongodb" })

Output:



1. Find the books that are published in 2019. Use find() function and pubyear key.

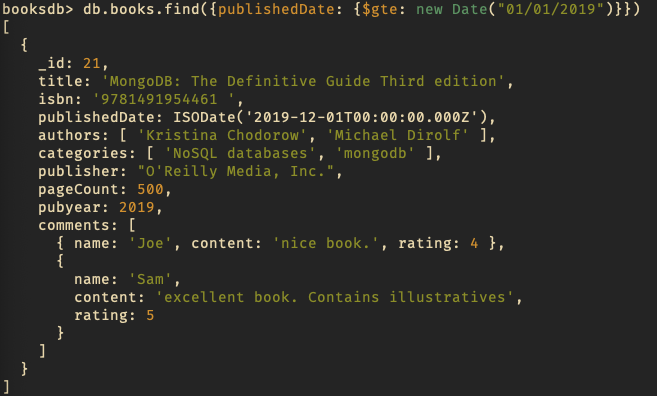
Command:

db.books.find({ pubyear: 2019 })

Output:

A screen shot of a computer program

Description automatically generated

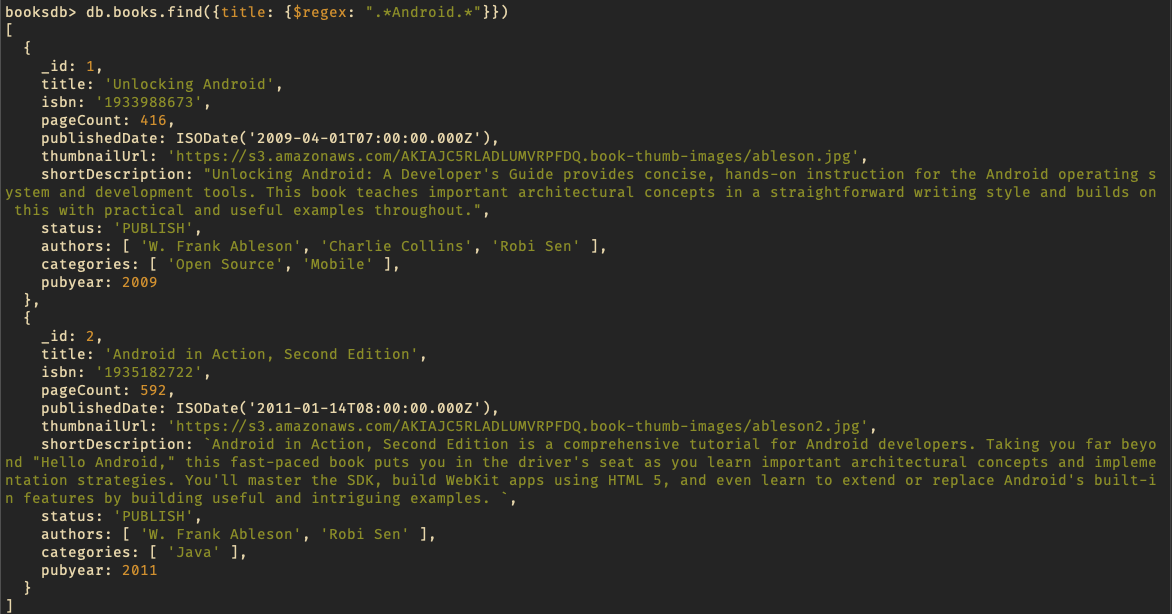
1. Find the books that are published on or after 01/01/2019. A date variable is defined with “01/01/2019” as initial value. Use find() function, publishedDate key and $gte operators to answer the question.   
     
   Command: db.books.find({publishedDate: {$gte: new Date("01/01/2019")}})  
   

**String functions**

1. Find the books that have 3 authors. Use $size operator. How many books are they? Use count() function.   
     
   Command: db.books.find({authors: {$size: 3}}).count()



1. Find the books that have the word ‘Android’ in the title.   
     
   Command: db.books.find({title: {$regex: ".\*Android.\*"}})

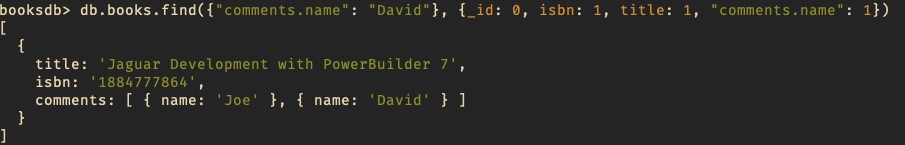
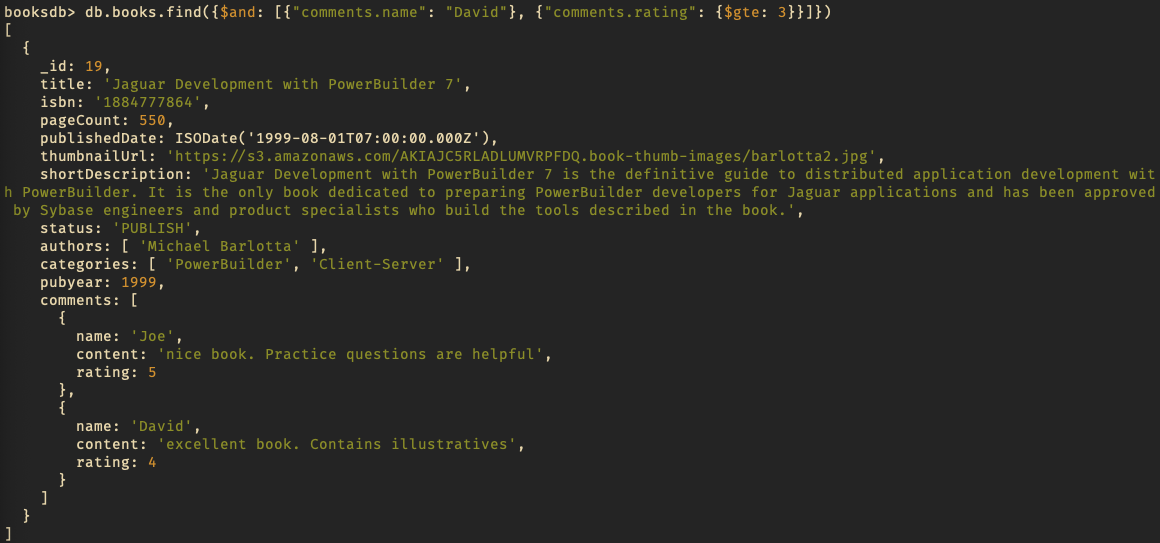


**Reference**:

You can use $regex operator to check if a field contains a string in MongoDB. The syntax is as follows –

db.yourCollectionName.findOne({"yourFieldName":{$regex:".\*yourValue.\*"}});

<https://www.tutorialspoint.com/check-if-a-field-contains-a-string-in-mongodb>

1. Find the books that miss the short description key. Use $exists operator.   
     
   Command: db.books.find({shortDescription: {$exists: false}})  
     
   
2. Find the books that David commented on. Show only the book isbn, title and the Name in the embedded document of the comments array.   
     
   Command: db.books.find({"comments.name": "David"}, {\_id: 0, isbn: 1, title: 1, "comments.name": 1})  
   
3. Find the books that have a rating less than or equal to 3.   
     
   Command: db.books.find({"comments.rating": {$lte: 3}})  
   
4. Find the books that are commented by David and have rating equal 3 or greater than 3.   
     
   Command: db.books.find({$and: [{"comments.name": "David"}, {"comments.rating": {$gte: 3}}]})  
   
5. Find the books that have 2 comments.  
     
   Command: db.books.find({comments: {$size: 2}})