San José State University Date: December 2, 2022

Project Report

CS 157C - NoSQL Database Systems Team: nokimchi

Index

1. Team Members

- Deep Shah
- Yunseo Han
- Dylan Lehuan Ha

2. Project Name

- ALotOfAuthors

3. Project Description

- ALotOfAuthors (ALOA) is a simple database application that lets you find, update, delete, and insert authors. ALOA could be used by libraries to manage books, keep track of authors and their ratings, and let people search for specific authors. The database stores documents of each author which consists of information such as author name, ratings, profile picture, about, gender, and number of fans. The dataset is relatively large with the total size being 1.8 gigabytes. The database was deployed in an AWS cluster of 7 nodes. There is one mongos server node, three config servers in a replica set, and three shard nodes, each of which have a replica set of 3 members. ALOA is written in Python, and has an end-to-end connection to the MongoDB database using pymongo. While using the application, the user can select from 15 different functions. After selection, the application gives instructions on making necessary inputs, which changes depending on the chosen function. After entering input, the ALOA will let the user know if their command was entered correctly and reject the input if any exceptions occur. The result from the command can be confirmed by checking the MongoDB database directly, which can be easily achieved by using a database GUI software like MongoDB Compass.

4. Project Successes

- While implementing ALotOfAuthors, we successfully deployed a working MongoDB Database on AWS Elastic Compute Cloud (EC2). We then imported a 1.8 gigabyte dataset of authors and 230 megabytes of series into the cloud using MongoDB Compass, and connected it to a python application using pymongo. We then utilized MongoDB CRUD operations to create 15 different dynamic queries and commands that would be intuitive and practical in a library setting. Finally, we created a simple text GUI where users can interact with the database.
- This Project helped us understand the practical implications of NoSQL databases, their merits, and limitations. We also learned to effectively work as a team in a remote setting. It enabled us to work with the cloud platform AWS and learn more about cloud storage.
- We devised an efficient sharding setup that can ensure availability in case of node failure. To achieve this, we created 3 shards in one node with their replicas in the other 2 nodes. We then assigned higher priority to node 1 for shard 1, node 2 for shard 2, and node 3 for shard 3, so that all 3 shards have their primary replica in a different node. This will ensure efficient division of load and protection against failure.

5. Unexpected Events

- We initially had difficulty in establishing an end-to-end connection from the server to our application. It was a critical step and after several efforts, we realized that there are some changes required in the security group configuration. After changing the inbound connection rules, we were able to use the public DNS to connect the database with our application.

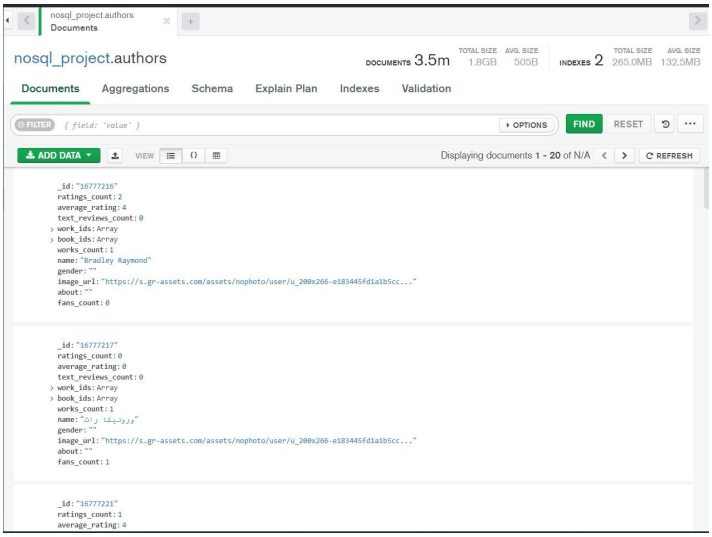
- While testing some queries for our application, we got errors for certain inputs. We realized that some documents were not in the correct format and were causing errors. We hence introduced error handling mechanisms to our application and made it dynamic to be able to handle variations in format.

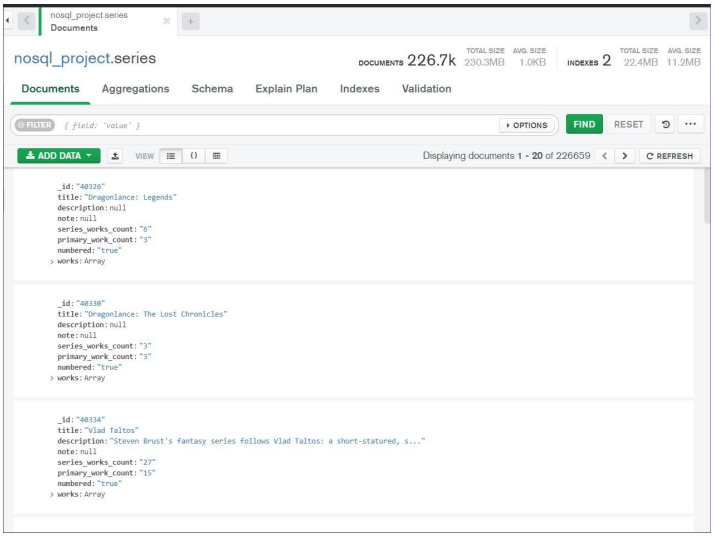
6. Lessons Learned

- Important lessons learned were to create a complete application with efficient cloud storage. We learned the importance of horizontal scaling and also implemented it using AWS.
- Following improvements can be made:
    - improvements at the application level such as a better GUI
    - allowing several concurrent users
    - optimizing the server to be able to handle large number of requests from multiple users

7. DataSet

- https://www.kaggle.com/datasets/opalskies/large-books-metadata-dataset-50-mill-entries
- We used “authors.json” of size 1.8 gigabytes (original 3.35 gigabytes) and “series.json” of size 230 megabytes which are subsets of the “large-books-meta-dataset” and the authors.json has fields like name, books, gender, profile image, description, rating, about, and more required for our application.
- We first converted the files downloaded from Kaggle into an appropriate json format using sed commands to add commas and square brackets. We then used a json editor application to get only the required fields from authors.json and exported a part of it as a new authors.json file. We also changed the “id” field to “_id” in both the files.





8. Database Schema

- An “authors” collection is created. This collection contains authors along with all the information that describes the author. We also created a collection called “series” which contains book series as well as information such as series title, description, fans, and books within the series.
- Both “authors” and “series” collections have an “_id” field which serves as the default index created by MongoDB. This _id is unique and everytime a new author is populated, a unique _id is added. This means

that every single document has a unique _id which allows it to be an index. It was also appropriate to have an index at _id since it is used to search authors many times in update statements.

- We also added hashed indexes on each collection. A hashed index was added to the “name” field in the collection “authors”, and another was added to the “title” field in the “series” collection. We added these indexes so that we can search and access those fields quicker than the default which would look through every value in the collection.

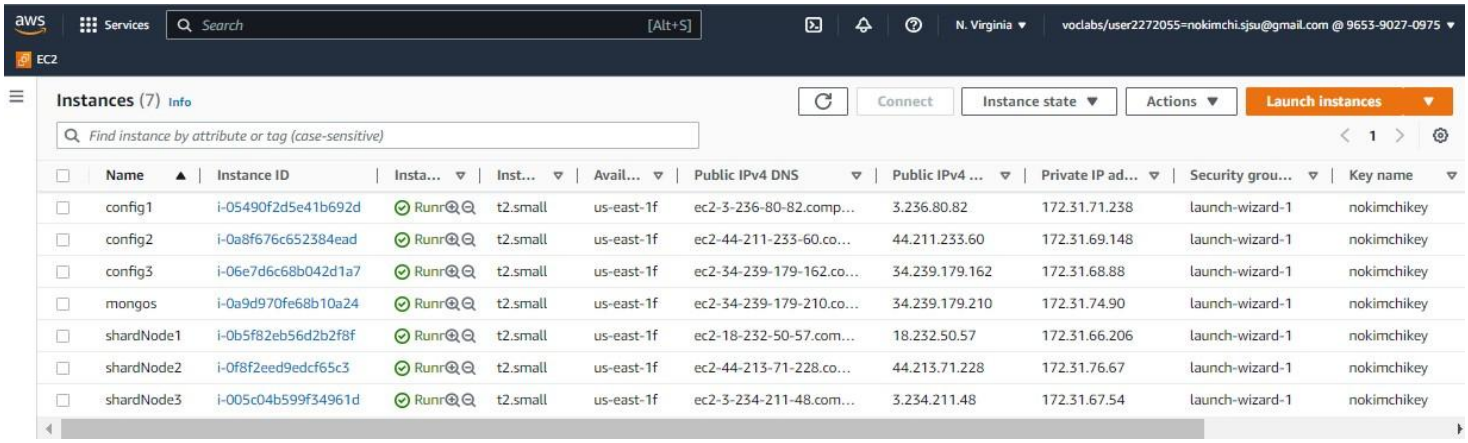
9. NoSQL configuration

- To deploy ALotOfAuthors, we used Amazon Elastic Compute Cloud to create virtual machine Instances. With a total of 7 different instances, there is one mongos server, three config servers in a replica set, and three shard servers, each of which have a replica set of 3 members.
- To allow lower load per shard and protection against node failure, we created three shards in one node with their replicas in the other two nodes. We then assigned higher priority to node 1 for shard 1, node 2 for shard 2, and node 3 for shard 3, so that all three shards have their primary replica in a different node.

Deployment Summary

Service	Instance/ Node Name	IP Address (Private)	Port	Replica Set Name	Core Process	DB Path	Replica Priority
Shard Controller	mongos	172.31.80.79	27020	-	mongos	-	-
Config Server Primary	config1	172.31.71.238	27019	crs	mongod	/data/db	1
Config Server Secondary 1	config2	172.31.69.148	27019	crs	mongod	/data/db	1
Config Server Secondary 2	config3	172.31.68.88	27019	crs	mongod	/data/db	1
Shard a Primary	shardNode1	172.31.66.206	27021	a	mongod	/data/a	4
Shard b Secondary 1		172.31.66.206	27022	b	mongod	/data/b	1
Shard c Secondary 1		172.31.66.206	27023	c	mongod	/data/c	1
Shard a Secondary 1	shardNode2	172.31.76.67	27021	a	mongod	/data/a	1
Shard b Primary		172.31.76.67	27022	b	mongod	/data/b	4
Shard c Secondary 2		172.31.76.67	27023	c	mongod	/data/c	1
Shard a Secondary 2	shardNode3	172.31.67.54	27021	a	mongod	/data/a	1
Shard b Secondary 2		172.31.67.54	27022	b	mongod	/data/b	1
Shard c Primary		172.31.67.54	27023	c	mongod	/data/c	4

Instances on AWS:



Running config servers:

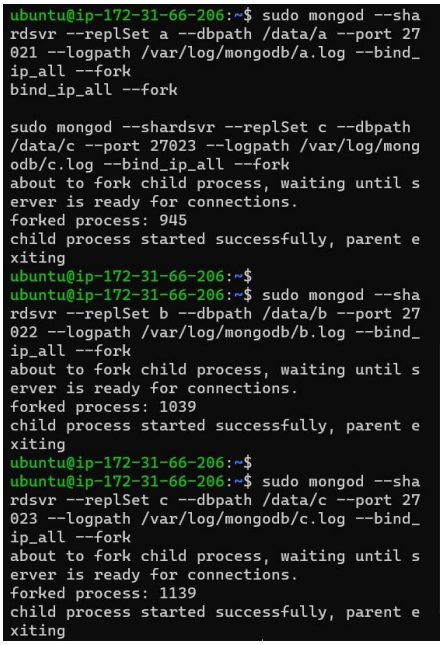
```
ubuntu@ip-172-31-71-238:~$ sudo mongod --configsvr --replSet crs --dbpath /data/db --port 27019 --logpath /var/log/mongodb/mongod.log --bind_ip_all --fork
about to fork child process, waiting until server is ready for connections.
forked process: 950
child process started successfully, parent exiting
```

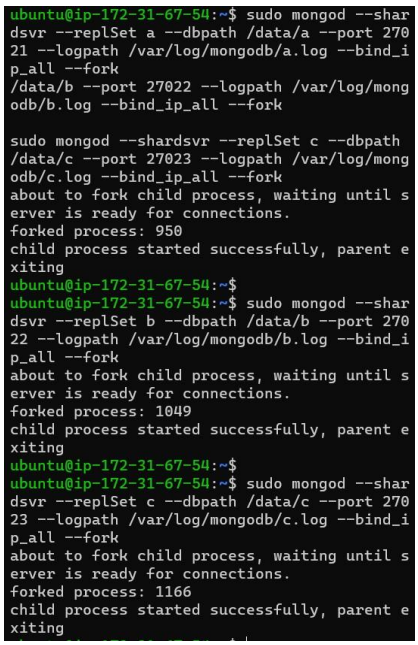
```
ubuntu@ip-172-31-69-148:~$ sudo mongod --configsvr --replSet crs --dbpath /data/db --port 27019 --logpath /var/log/mongodb/mongod.log --bind_ip_all --fork
about to fork child process, waiting until server is ready for connections.
forked process: 941
child process started successfully, parent exiting
```

```
ubuntu@ip-172-31-68-88:~$ sudo mongod --configsvr --replSet crs --dbpath /data/db --port 27019 --logpath /var/log/mongodb/mongod.log --bind_ip_all --fork
about to fork child process, waiting until server is ready for connections.
forked process: 943
child process started successfully, parent exiting
```

rs.status() for Config servers:

Running Shard servers:



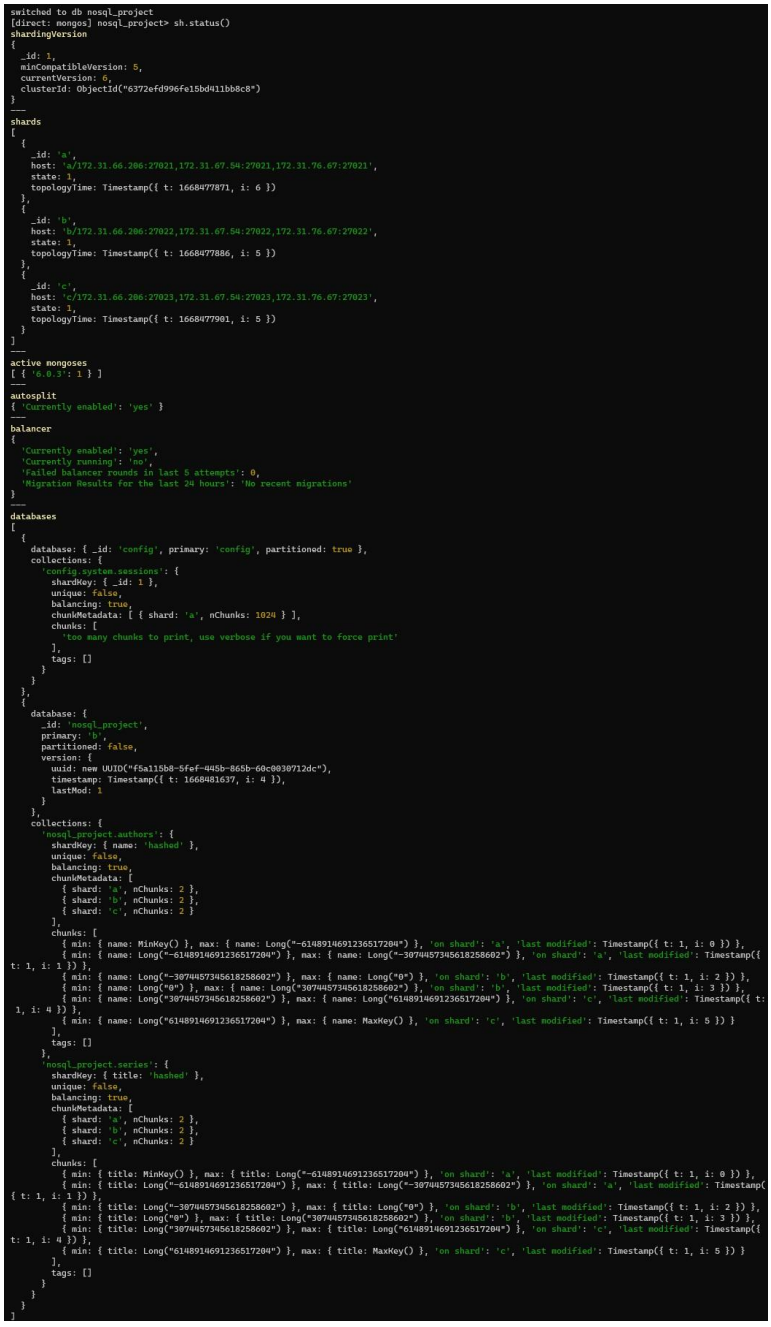


Shard replicas:

10. Shard Key & Sharding Strategy

- We have used hash based sharding on the field “name” for the Authors collection and on the field “title” for “Series” collection. These fields are the same as the one we previously added hashed indexes on. We chose to use “name” and “title” as shard keys because it can quickly fetch a document by searching for these fields, and using a monotonically increasing field like_id would be a bad idea. However, names and titles can also have frequency issues with common names. For example, there may be a lot of authors with the name “John” which may lead to uneven chunks of data. That is why we chose to use hash based sharding that will make it random. “Name” and “title” are fitting to be a hash based shard key since it has high cardinality, meaning there are a large number of different values.

Shard status:



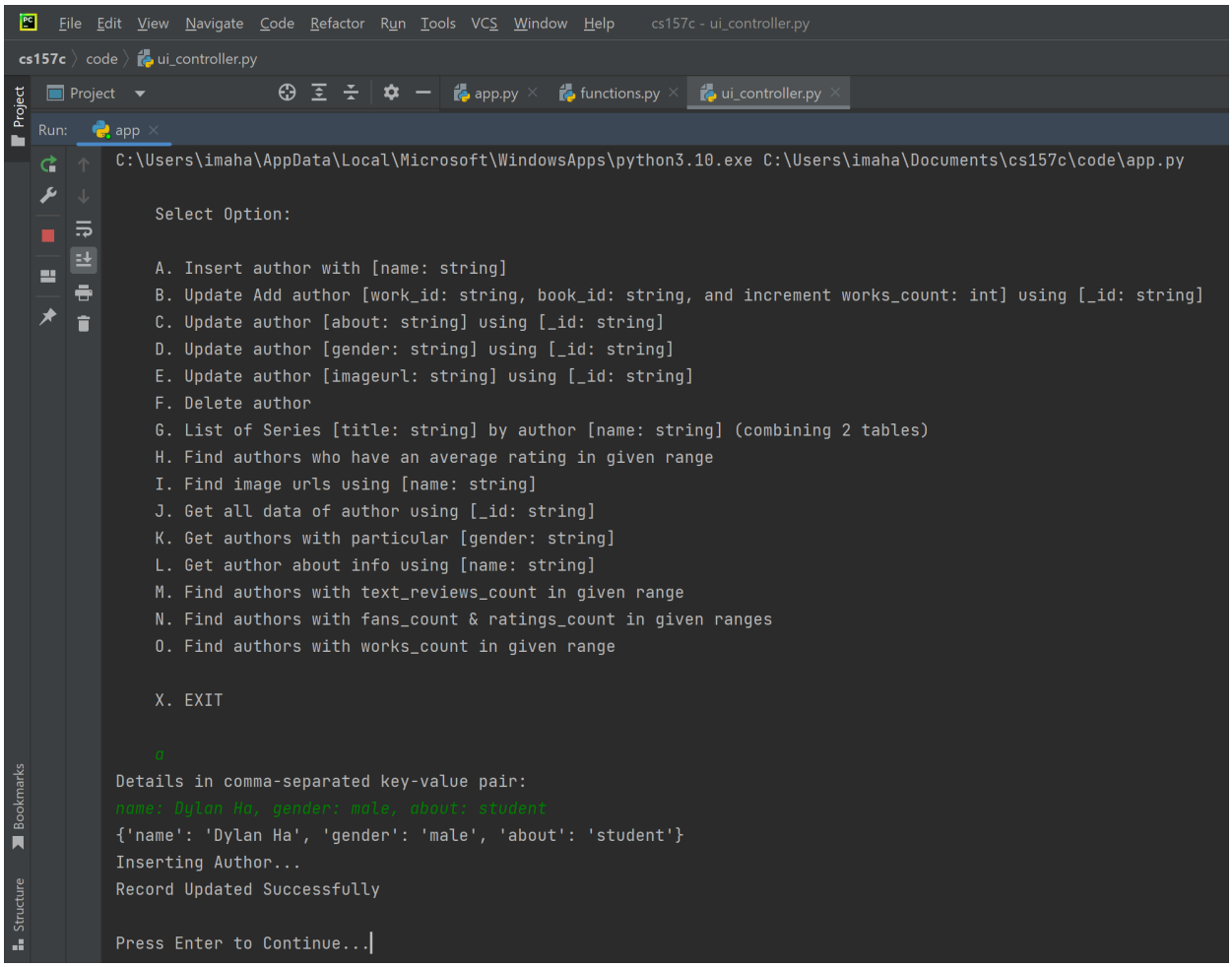
11. Connecting to the Application

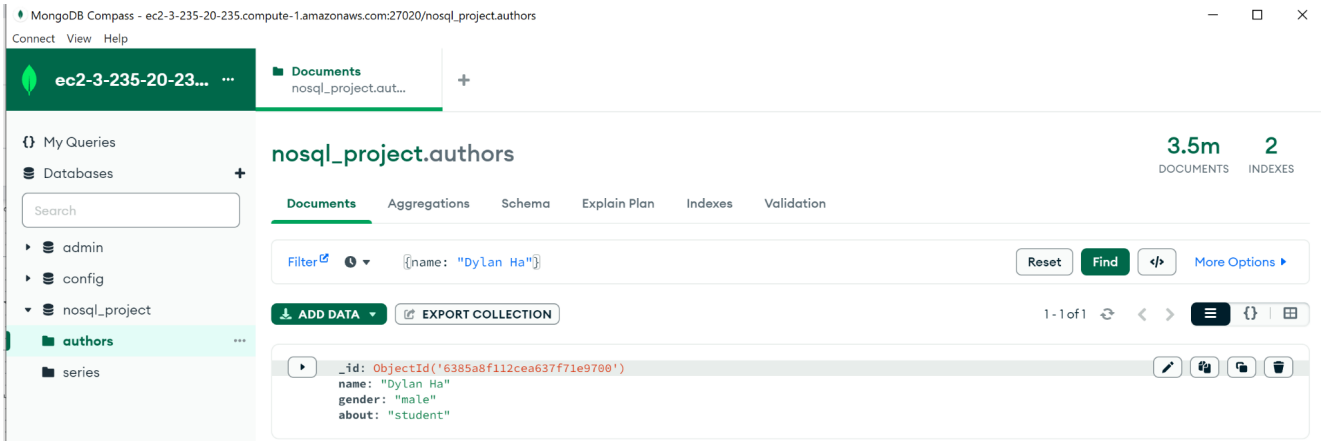
- After setting up the mongos server, we changed the security group settings and added inbound network rules to allow external connections on port 27020 where the mongos server is running.
- We then used the public IPv4 DNS of the mongos node to create an end to end connection between the application and the database.
- We used pymongo as our NoSQL driver and created a connection using the command below: myclient=pymongo.MongoClient("mongodb://ec2-3-235-20-235.compute-1.amazonaws.com:27020/test")
- Once the connection is established, myclient variable can be used to select the database and access the required collections. Command to access DB: db=myclient["nosql_project"]
- Now using mongodb queries our application can send requests to the mongos server to update the documents in a database collection.
- Our application can also request to fetch data, in which case the mongos server with help of the config server will search through the shards, get the results and send it back to our application.

12. Running Queries on Data

For each function, we have attached the output of the python application and screenshots of MongoDB compass to clearly show that the function works.

A. Insert author with [name: string]
   Description: Can be used to insert a new document to the authors table.
   Command: db.authors.insert_one(indata)





B. Update Add author [work_id: string, book_id: string, and increment works_count: int] using [_id: string]

Description: Update an author's data to add details about new work they published.

Command:
db.authors.update_one({"_id": id}, {"$inc": {"works_count":1}, "$push": {"work_ids": workdata, "book_ids": bookdata} })

(Before)

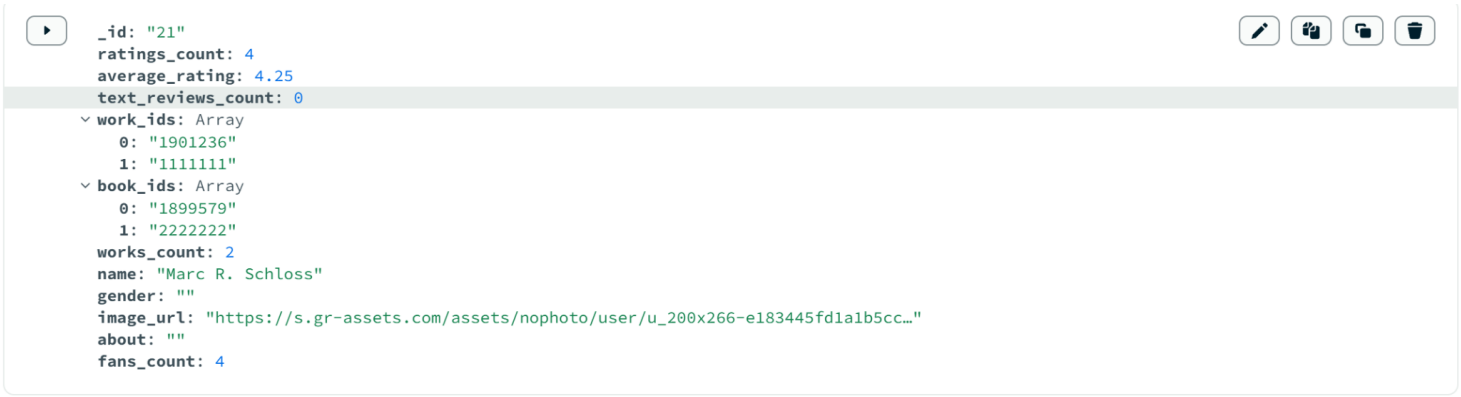
```
    A. Insert author with [name: string]
    B. Update Add author [work_id: string, book_id: string, and increment works_count: int] using [_id: string]
    C. Update author [about: string] using [_id: string]
    D. Update author [gender: string] using [_id: string]
    E. Update author [imageurl: string] using [_id: string]
    F. Delete author
    G. List of Series [title: string] by author [name: string] (combining 2 tables)
    H. Find authors who have an average rating in given range
    I. Find image urls using [name: string]
    J. Get all data of author using [_id: string]
    K. Get authors with particular [gender: string]
    L. Get author about info using [name: string]
    M. Find authors with text_reviews_count in given range
    N. Find authors with fans_count & ratings_count in given ranges
    O. Find authors with works_count in given range

    X. EXIT

    b
author_id, new work_id, new book_id :
21, 1111111, 2222222
21 1111111 2222222
Updating Author Data...
Record Updated Successfully
```

(After)

C. Update author [about: string] using [_id: string]

Description: Update about info of an author. _id needed as an update is a sensitive operation and needs to be unique in this case.
Command: db.authors.update_one({"_id": id}, {"$set": {"about":newValue}})

(Before)




```
c
Ø
    Select Option:

    A. Insert author with [name: string]
    B. Update Add author [work_id: string, book_id: string, and increment works_count: int] using [_id: string]
    C. Update author [about: string] using [_id: string]
    D. Update author [gender: string] using [_id: string]
    E. Update author [imageurl: string] using [_id: string]
    F. Delete author
    G. List of Series [title: string] by author [name: string] (combining 2 tables)
    H. Find authors who have an average rating in given range
    I. Find image urls using [name: string]
    J. Get all data of author using [_id: string]
    K. Get authors with particular [gender: string]
    L. Get author about info using [name: string]
    M. Find authors with text_reviews_count in given range
    N. Find authors with fans_count & ratings_count in given ranges
    O. Find authors with works_count in given range

    X. EXIT

    author_id, new about:
21, This is a description about the author (test).
Updating Author Data...
Record Updated Successfully

Press Enter to Continue...
```

(After)

D. Update author [gender: string] using [_id: string]

Description: Update or add gender info for an author.
Command: db.authors.update_one({"_id": id}, {"$set": {"gender":newValue}})
(Before)


```
Press Enter to Continue...

    Select Option:

    A. Insert author with [name: string]
    B. Update Add author [work_id: string, book_id: string, and increment works_count: int] using [_id: string]
    C. Update author [about: string] using [_id: string]
    D. Update author [gender: string] using [_id: string]
    E. Update author [imageurl: string] using [_id: string]
    F. Delete author
    G. List of Series [title: string] by author [name: string] (combining 2 tables)
    H. Find authors who have an average rating in given range
    I. Find image urls using [name: string]
    J. Get all data of author using [_id: string]
    K. Get authors with particular [gender: string]
    L. Get author about info using [name: string]
    M. Find authors with text_reviews_count in given range
    N. Find authors with fans_count & ratings_count in given ranges
    O. Find authors with works_count in given range

    X. EXIT

    d
author_id, Gender:
21, male
Updating Author Data...
Record Updated Successfully

Press Enter to Continue...
```

(After)

E. Update author [imageurl: string] using [_id: string]

Description: Change the image of an author by updating the url

Command: db.authors.update_one({"_id": id}, {"$set": {"image_url":newValue}})

(Before)

https://s.gr-assets.com/assets/nophoto/user/u_200x266-e183445fd1a1b5cc7075bb1cf7043306.png

```
    Select Option:

    A. Insert author with [name: string]
    B. Update Add author [work_id: string, book_id: string, and increment works_count: int] using [_id: string]
    C. Update author [about: string] using [_id: string]
    D. Update author [gender: string] using [_id: string]
    E. Update author [imageurl: string] using [_id: string]
    F. Delete author
    G. List of Series [title: string] by author [name: string] (combining 2 tables)
    H. Find authors who have an average rating in given range
    I. Find image urls using [name: string]
    J. Get all data of author using [_id: string]
    K. Get authors with particular [gender: string]
    L. Get author about info using [name: string]
    M. Find authors with text_reviews_count in given range
    N. Find authors with fans_count & ratings_count in given ranges
    O. Find authors with works_count in given range

    X. EXIT

    e
author_id, image_url:
21, https://upload.wikimedia.org/wikipedia/commons/f/fa/Apple_logo_black.svg
Updating Author Data...
Record Updated Successfully
```

(After)

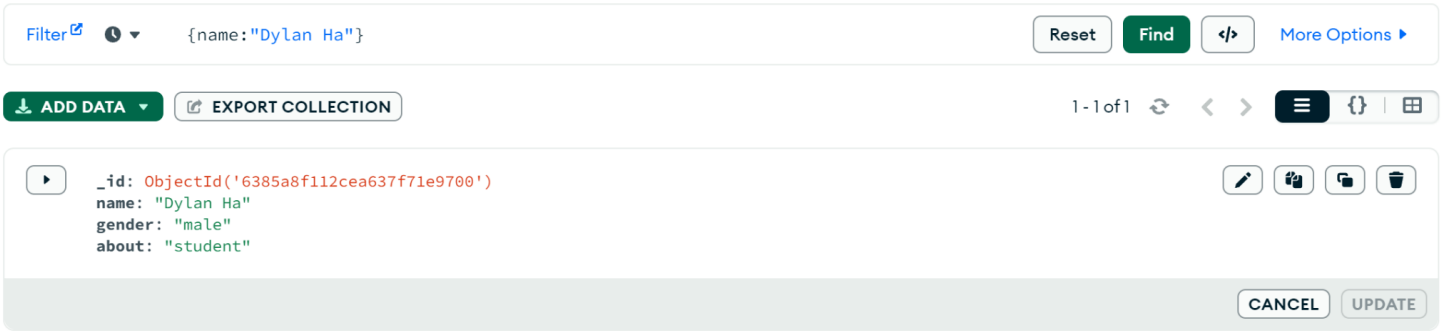
https://upload.wikimedia.org/wikipedia/commons/f/fa/Apple_logo_black.svg

F. Delete Authors

Description: Delete all authors that match the filter criteria

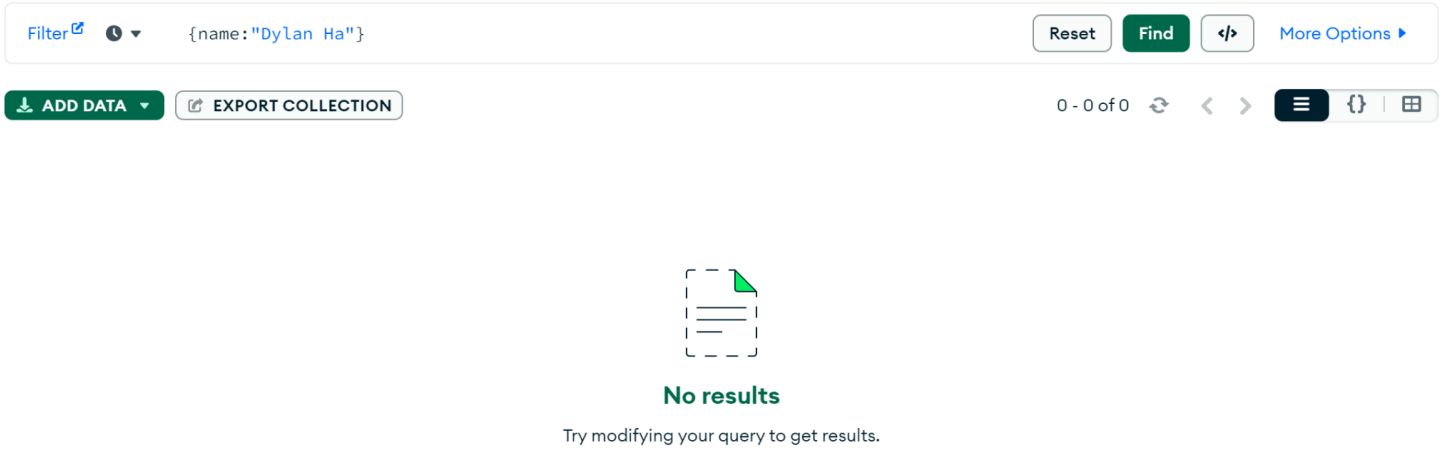
Command: db.authors.delete_many(filter)

(Before)




```
    Select Option:

    A. Insert author with [name: string]
    B. Update Add author [work_id: string, book_id: string, and increment works_count: int] using [_id: string]
    C. Update author [about: string] using [_id: string]
    D. Update author [gender: string] using [_id: string]
    E. Update author [imageurl: string] using [_id: string]
    F. Delete author
    G. List of Series [title: string] by author [name: string] (combining 2 tables)
    H. Find authors who have an average rating in given range
    I. Find image urls using [name: string]
    J. Get all data of author using [_id: string]
    K. Get authors with particular [gender: string]
    L. Get author about info using [name: string]
    M. Find authors with text_reviews_count in given range
    N. Find authors with fans_count & ratings_count in given ranges
    O. Find authors with works_count in given range

    X. EXIT

    f
Enter record details in comma-separated key-value pair to select for deletion:
name: Dylan Ha, gender: male
1 documents deleted.
Records Deleted Successfully
```

(After)



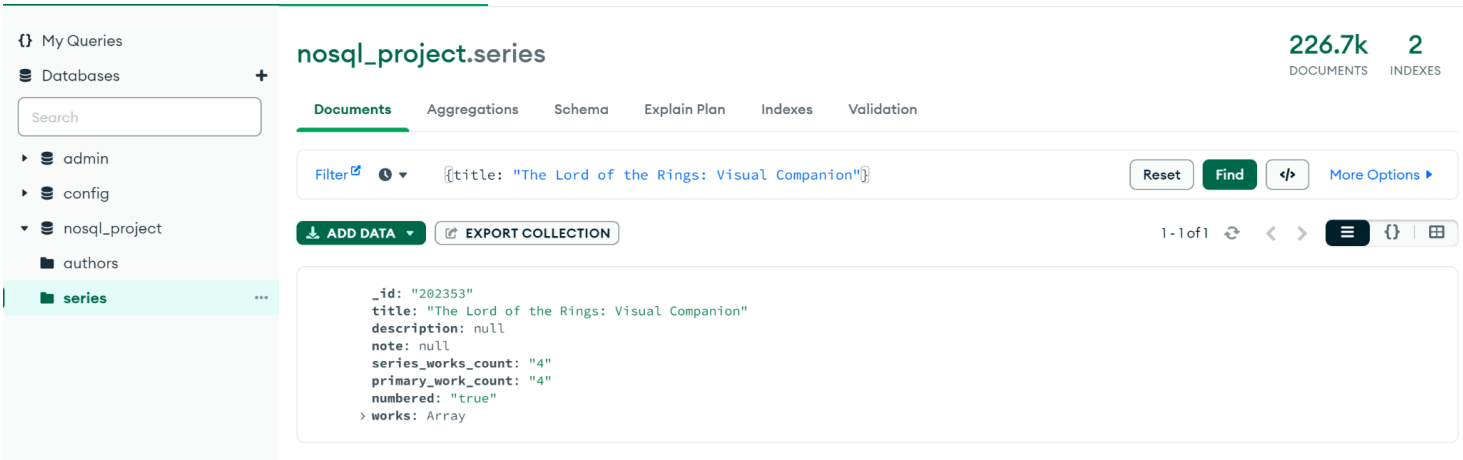
G. List of Series [title: string] by author [name: string] (combining 2 tables)

Description: Get all series written by an author. To do so, first get work_ids for that author. Then get all titles from series table that have these work_ids.

Commands:
db.authors.find({"name":aname}, {'work_ids':1})
db.series.find({"works.work_id": {"$in": wids}}, {"title":1})

```
Press Enter to Continue...
g

    Select Option:

    A. Insert author with [name: string]
    B. Update Add author [work_id: string, book_id: string, and increment works_count: int] using [_id: string]
    C. Update author [about: string] using [_id: string]
    D. Update author [gender: string] using [_id: string]
    E. Update author [imageurl: string] using [_id: string]
    F. Delete author
    G. List of Series [title: string] by author [name: string] (combining 2 tables)
    H. Find authors who have an average rating in given range
    I. Find image urls using [name: string]
    J. Get all data of author using [_id: string]
    K. Get authors with particular [gender: string]
    L. Get author about info using [name: string]
    M. Find authors with text_reviews_count in given range
    N. Find authors with fans_count & ratings_count in given ranges
    O. Find authors with works_count in given range

    X. EXIT

    Author Name: Jude Fisher
["Fool's Gold", 'The Lord of the Rings: Visual Companion']
```

(Example of series existing)



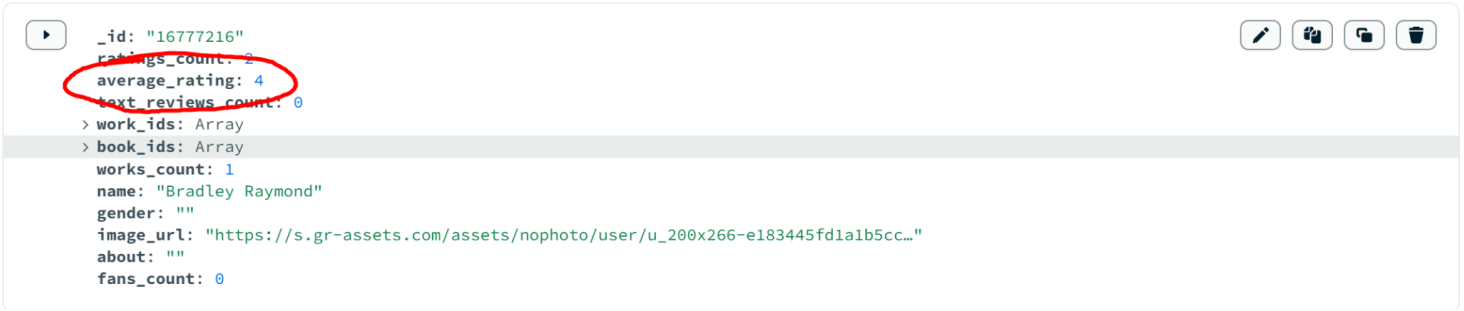
H. Find authors who have an average rating in given range

Description: To filter authors in a particular range of average ratings and get a certain limit of them.

Command: db.authors.find({"average_rating": {"$gte": minr, "$lte": maxr}}, {"_id":0, "name": 1, "average_rating": 1}).limit(limit)

```
    F. Delete author
    G. List of Series [title: string] by author [name: string] (combining 2 tables)
    H. Find authors who have an average rating in given range
    I. Find image urls using [name: string]
    J. Get all data of author using [_id: string]
    K. Get authors with particular [gender: string]
    L. Get author about info using [name: string]
    M. Find authors with text_reviews_count in given range
    N. Find authors with fans_count & ratings_count in given ranges
    O. Find authors with works_count in given range

    X. EXIT

    h
Range: min, max, limit
4, 5, 10
['Name: Bradley Raymond, Rating: 4',
 'Name: Bennie Elgin Calloway III, Rating: 4',
 'Name: Jude Fisher, Rating: 4.53',
 'Name: Dave Thomas, Rating: 4.22',
 'Name: Marc R. Schloss, Rating: 4.25',
 'Name: Sjúrður Skaale, Rating: 4.25',
 'Name: David Lewman and Various, Rating: 4.83',
 'Name: John McPhee, Rating: 4.16',
 'Name: Marie Florence van Es, Rating: 4',
 'Name: Gary Russell, Rating: 4.45']
```

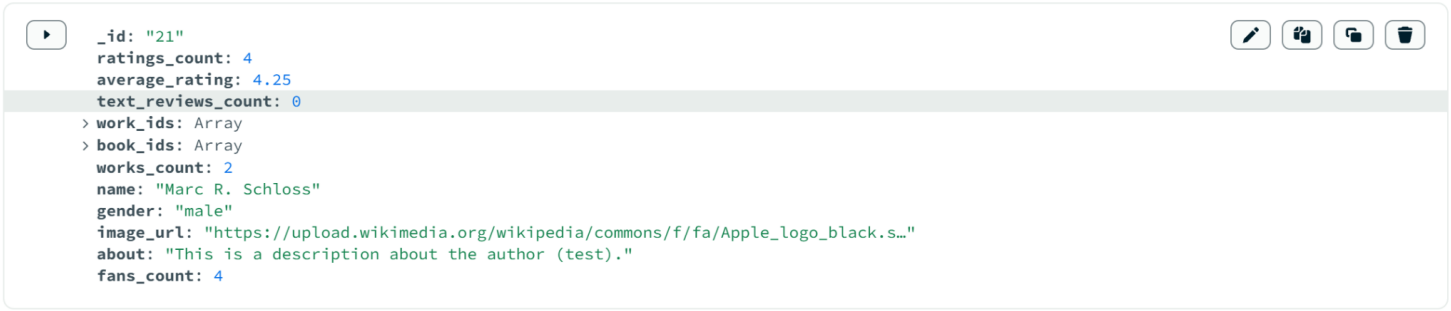
```
    h
Range: min, max, limit
3, 4, 10
['Name: Bradley Raymond, Rating: 4',
 'Name: Bennie Elgin Calloway III, Rating: 4',
 'Name: James Hamilton-Paterson, Rating: 3.79',
 'Name: 成甲, Rating: 3.79',
 'Name: Zilpha Keatley Snyder, Rating: 3.83',
 'Name: Ian Lemke, Rating: 3.7',
 'Name: Elaine Cunningham, Rating: 3.81',
 'Name: Philippa Carr, Rating: 3.88',
 'Name: Heidi Fleiss, Rating: 3.64',
 'Name: Heidi Pitlor, Rating: 3.66']
```



I. Find image urls using [name: string]

Description: Used to get a photo of the author given their name.
Command: db.authors.find({"name":name}, {'image_url':1})

```
    Select Option:

    A. Insert author with [name: string]
    B. Update Add author [work_id: string, book_id: string, and increment works_count: int] using [_id: string]
    C. Update author [about: string] using [_id: string]
    D. Update author [gender: string] using [_id: string]
    E. Update author [imageurl: string] using [_id: string]
    F. Delete author
    G. List of Series [title: string] by author [name: string] (combining 2 tables)
    H. Find authors who have an average rating in given range
    I. Find image urls using [name: string]
    J. Get all data of author using [_id: string]
    K. Get authors with particular [gender: string]
    L. Get author about info using [name: string]
    M. Find authors with text_reviews_count in given range
    N. Find authors with fans_count & ratings_count in given ranges
    O. Find authors with works_count in given range

    X. EXIT

    i
Author Name: Marc R. Schloss
['https://upload.wikimedia.org/wikipedia/commons/f/fa/Apple_logo_black.svg']
```



https://upload.wikimedia.org/wikipedia/commons/f/fa/Apple_logo_black.svg
(which is the image url we updated earlier)

J. Get all data of author using [_id: string]

Description: Used to fetch all data of a given author with their ID

Command: db.authors.find({"_id":id})

```
    G. List of Series [title: string] by author [name: string] (combining 2 tables)
    H. Find authors who have an average rating in given range
    I. Find image urls using [name: string]
    J. Get all data of author using [_id: string]
    K. Get authors with particular [gender: string]
    L. Get author about info using [name: string]
    M. Find authors with text_reviews_count in given range
    N. Find authors with fans_count & ratings_count in given ranges
    O. Find authors with works_count in given range

    X. EXIT

    j
Author ID: 21
{'_id': '21',
 'about': 'This is a description about the author (test).',
 'average_rating': 4.25,
 'book_ids': ['1899579', '2222222'],
 'fans_count': 4,
 'gender': 'male',
 'image_url': 'https://upload.wikimedia.org/wikipedia/commons/f/fa/Apple_logo_black.svg',
 'name': 'Marc R. Schloss',
 'ratings_count': 4,
 'text_reviews_count': 0,
 'work_ids': ['1901236', '1111111'],
 'works_count': 2}
```

K. Get authors with particular [gender: string]

Description: Get list of authors who have a particular gender.

Command: db.authors.find({"gender":g}, {'name':1}).limit(l)

```
    F. Delete author
    G. List of Series [title: string] by author [name: string] (combining 2 tables)
    H. Find authors who have an average rating in given range
    I. Find image urls using [name: string]
    J. Get all data of author using [_id: string]
    K. Get authors with particular [gender: string]
    L. Get author about info using [name: string]
    M. Find authors with text_reviews_count in given range
    N. Find authors with fans_count & ratings_count in given ranges
    O. Find authors with works_count in given range

    X. EXIT

    k
Gender, Limit :
female, 10
['Edith Wharton',
 'Nicole Rae Painter',
 'Loretta Krupinski',
 'Heidi Klum',
 'Heidi Swapp',
 'Heidi Hayes Jacobs',
 'Heidi Betts',
 'Ruby Banks-Payne',
 'Ruby Le Bois',
 'Diane E. Wirth']
```

```
_id: "16"
ratings_count: 436635
average_rating: 3.78
text_reviews_count: 26193
> work_ids: Array
> book_ids: Array
works_count: 1001
name: "Edith Wharton"
gender: "female"
image_url: "https://images.gr-assets.com/authors/1484512230p5/16.jpg"
about: "Edith Newbold Jones was born into such wealth and privilege that her f…"
fans_count: 3179
```

```
_id: "35"
ratings_count: 4
average_rating: 4.25
text_reviews_count: 0
> work_ids: Array
> book_ids: Array
works_count: 1
name: "Nicole Rae Painter"
gender: "female"
image_url: "https://s.gr-assets.com/assets/nophoto/user/f_200x266-3061b784cc8e7f02…"
about: ""
fans_count: 3
```


L. Get author about info using [name: string]

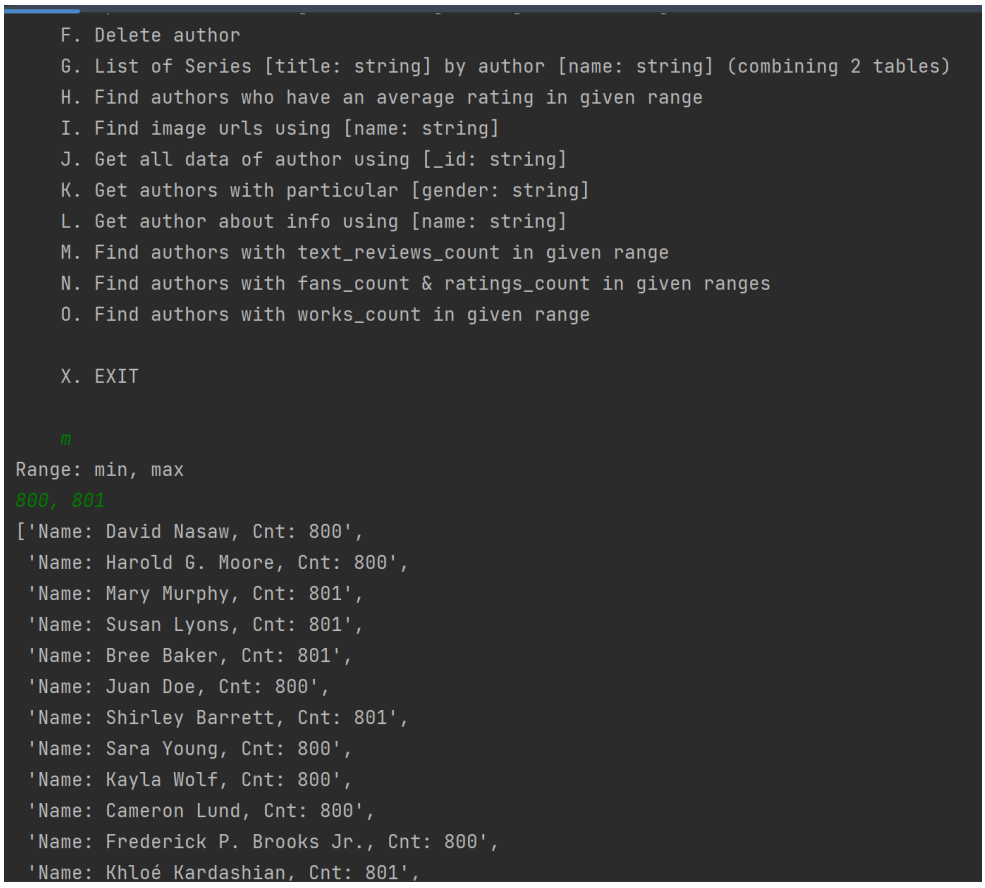
Description: Get the author description by fetching its about info using the author name.
Command: db.authors.find({"name":name}, {'about':1})

```
Press Enter to Continue...
    Select Option:

    A. Insert author with [name: string]
    B. Update Add author [work_id: string, book_id: string, and increment works_count: int] using [_id: string]
    C. Update author [about: string] using [_id: string]
    D. Update author [gender: string] using [_id: string]
    E. Update author [imageurl: string] using [_id: string]
    F. Delete author
    G. List of Series [title: string] by author [name: string] (combining 2 tables)
    H. Find authors who have an average rating in given range
    I. Find image urls using [name: string]
    J. Get all data of author using [_id: string]
    K. Get authors with particular [gender: string]
    L. Get author about info using [name: string]
    M. Find authors with text_reviews_count in given range
    N. Find authors with fans_count & ratings_count in given ranges
    O. Find authors with works_count in given range

    X. EXIT

    L
Author Name: Marc R. Schloss
[['21', 'This is a description about the author (test).']]
```

M. Find authors with text_reviews_count in given range

Description: Find the authors who have number of text reviews in a given range

Command: db.authors.find({"text_reviews_count": {"$gte": minr, "$lte": maxr}}, {"_id":0, "name": 1, "text_reviews_count": 1})



```
'Name: Judith Thurman, Cnt: 801',
'Name: Roger Lea MacBride, Cnt: 800',
'Name: Steven Herrick, Cnt: 801',
'Name: L.A. Kirk, Cnt: 800',
'Name: Justin Gustainis, Cnt: 800',
'Name: Harry Sidebottom, Cnt: 800',
'Name: David Yoo, Cnt: 801',
'Name: Danielle Ganek, Cnt: 800',
'Name: Rick    Wilson, Cnt: 801',
'Name: Katharina Diestelmeier, Cnt: 801',
'Name: Craig Clevenger, Cnt: 800',
'Name: David Servan-Schreiber, Cnt: 801',
'Name: Leigh Kelsey, Cnt: 801',
'Name: Carly Bloom, Cnt: 800',
'Name: S.J. Day, Cnt: 801',
'Name: Jane Lovering, Cnt: 800',
'Name: Warren Fahy, Cnt: 801',
'Name: A.J. Gregory, Cnt: 801']
```


```
_id: "3716"
ratings_count: 10226
average_rating: 3.88
text_reviews_count: 800
work_ids: Array
book_ids: Array
works_count: 12
name: "David Nasaw"
gender: "male"
image_url: "https://images.gr-assets.com/authors/1455755170p5/3716.jpg"
about: "David Nasaw is an American author, biographer and historian who specia…"
fans_count: 80
```

N. Find authors with fans_count & ratings_count in given ranges

Description: Filter the authors by range of their fans count & ratings count.
Command:

db.authors.find({"fans_count": {"$gte": fans[0], "$lte": fans[1]}, "ratings_count": {"$gte": ratin[0], "$lte": ratin[1]}}, {"_id":0, "name": 1, "fans_count": 1, "ratings_count": 1})

```
    L. Get author about info using [name: string]
    M. Find authors with text_reviews_count in given range
    N. Find authors with fans_count & ratings_count in given ranges
    O. Find authors with works_count in given range

    X. EXIT

    n
fans_count min, fans_count max : 100, 200
ratings_count min, ratings_count max : 4, 5
['Name: Paul Brad Logan, fans_count: 119, ratings_count: 4',
 'Name: J.A. Kellman, fans_count: 139, ratings_count: 4',
 'Name: Bilal Alaji, fans_count: 165, ratings_count: 4',
 'Name: J.A. Boulet, fans_count: 148, ratings_count: 4',
 'Name: Karen Leigh Gruber, fans_count: 163, ratings_count: 5',
 'Name: Chantelle Aimée Osman, fans_count: 135, ratings_count: 4',
 'Name: The Birch Twins, fans_count: 109, ratings_count: 5',
 'Name: Linda Rainier, fans_count: 110, ratings_count: 4',
 'Name: Laila Doncaster, fans_count: 109, ratings_count: 5',
 'Name: Jacob S Paulsen, fans_count: 129, ratings_count: 4',
 'Name: Karen Chaboyer, fans_count: 104, ratings_count: 5',
 'Name: Diana Dolea, fans_count: 146, ratings_count: 4',
 'Name: Halle J Ladd, fans_count: 131, ratings_count: 5',
 'Name: William C. Dell, fans_count: 153, ratings_count: 5',
 'Name: Michael C. Cordell, fans_count: 110, ratings_count: 4',
 'Name: Utanu Maa, fans_count: 126, ratings_count: 5']

Press Enter to Continue...
```

O. Find authors with works_count in given range

Description: Filter authors having works count within a certain range

Command:
db.authors.find({"works_count": {"$gte": minr, "$lte": maxr}}, {"_id":0, "name": 1, "works_count": 1})

```
    C. Update author [about: string] using [_id: string]
    D. Update author [gender: string] using [_id: string]
    E. Update author [imageurl: string] using [_id: string]
    F. Delete author
    G. List of Series [title: string] by author [name: string] (combining 2 tables)
    H. Find authors who have an average rating in given range
    I. Find image urls using [name: string]
    J. Get all data of author using [_id: string]
    K. Get authors with particular [gender: string]
    L. Get author about info using [name: string]
    M. Find authors with text_reviews_count in given range
    N. Find authors with fans_count & ratings_count in given ranges
    O. Find authors with works_count in given range

    X. EXIT

    o
Range: min, max
400, 401
['Name: Alma Flor Ada, Cnt: 400',
 'Name: Masashi Kishimoto, Cnt: 400',
 'Name: William Chambers, Cnt: 401',
 'Name: Gabriele Haefs, Cnt: 400',
 'Name: A.E. van Vogt, Cnt: 400',
 'Name: Leonid Andreyev, Cnt: 401',
 'Name: Mary Wollstonecraft, Cnt: 400']

Press Enter to Continue...
```


```
_id: "3430"
ratings_count: 5673
average_rating: 3.9
text_reviews_count: 1305
› work_ids: Array
› book_ids: Array
works_count: 400
name: "Alma Flor Ada"
gender: "female"
image_url: "https://images.gr-assets.com/authors/1258421287p5/3430.jpg"
about: "(<a target="_blank" rel="noopener nofollow" href="http://facebook.com/…"
fans_count: 137
```