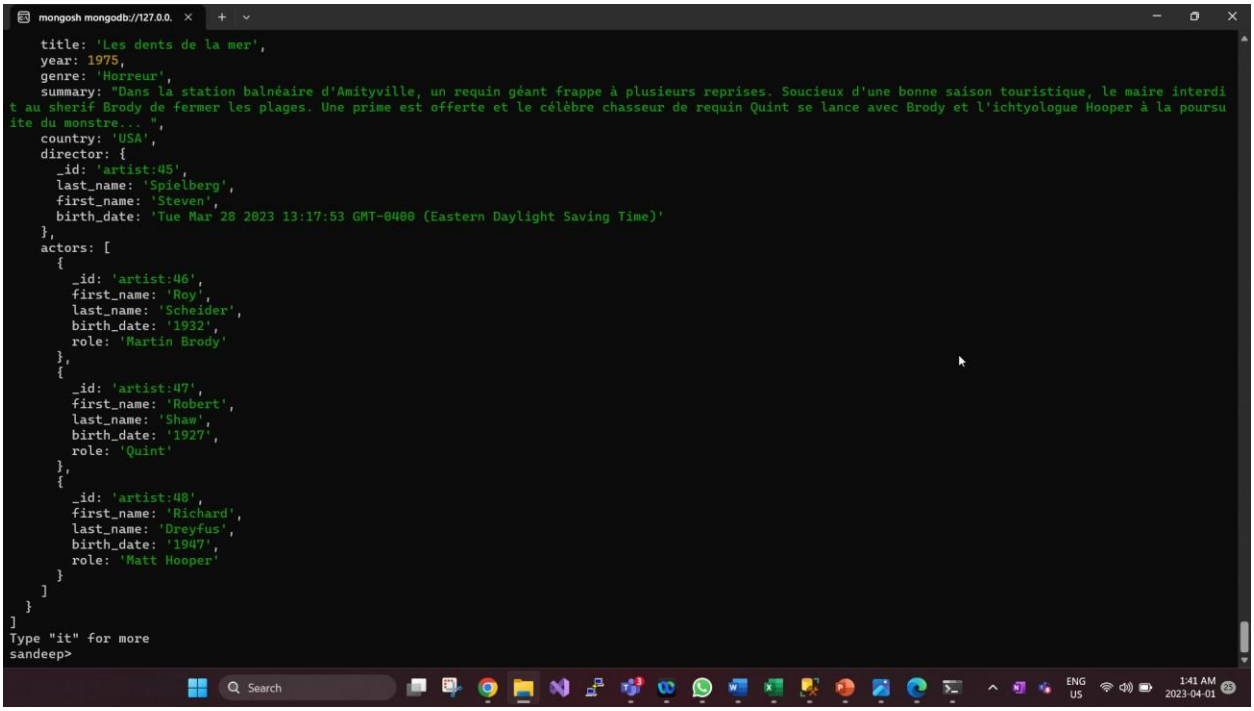
Contents

Midterm project - Mongo DB

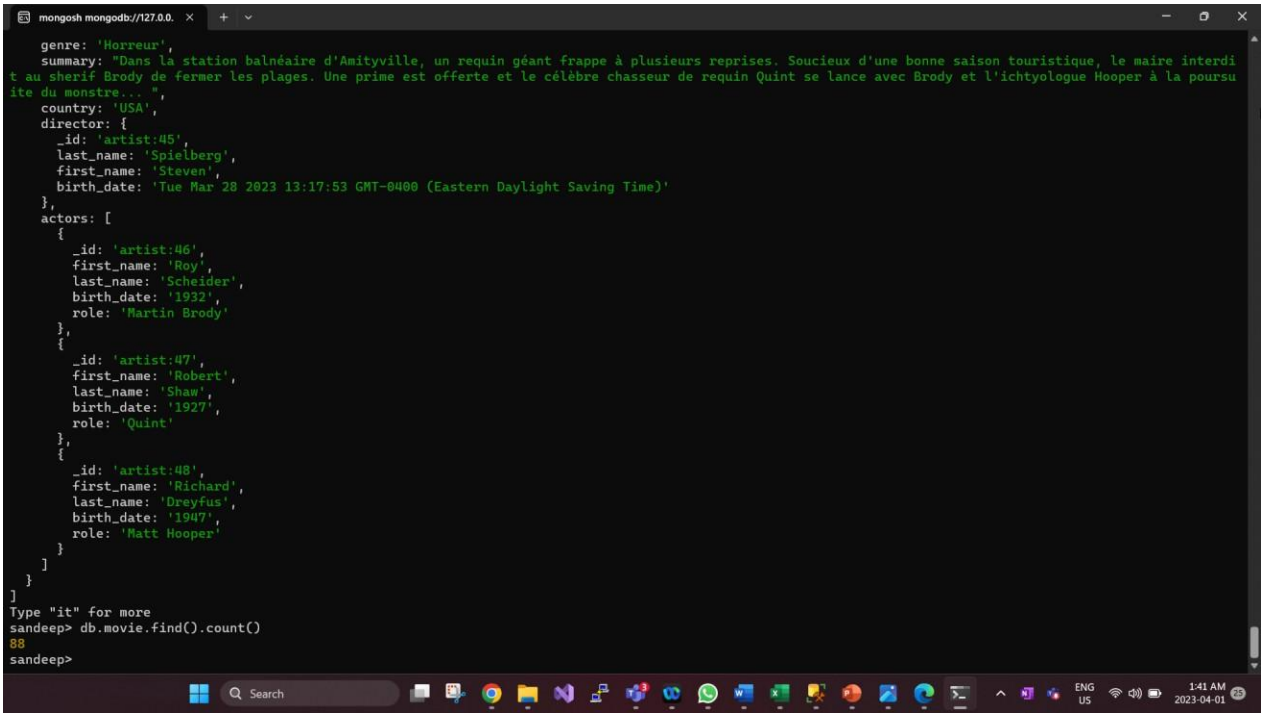
A Write the following script to fetch the document on movie collection.

1. **Fetch all the document from movie collection.**

   To fetch all documents from the "movie" collection in MongoDB, I used the find() method without any parameters. Here's an example command **db.movie.find()**

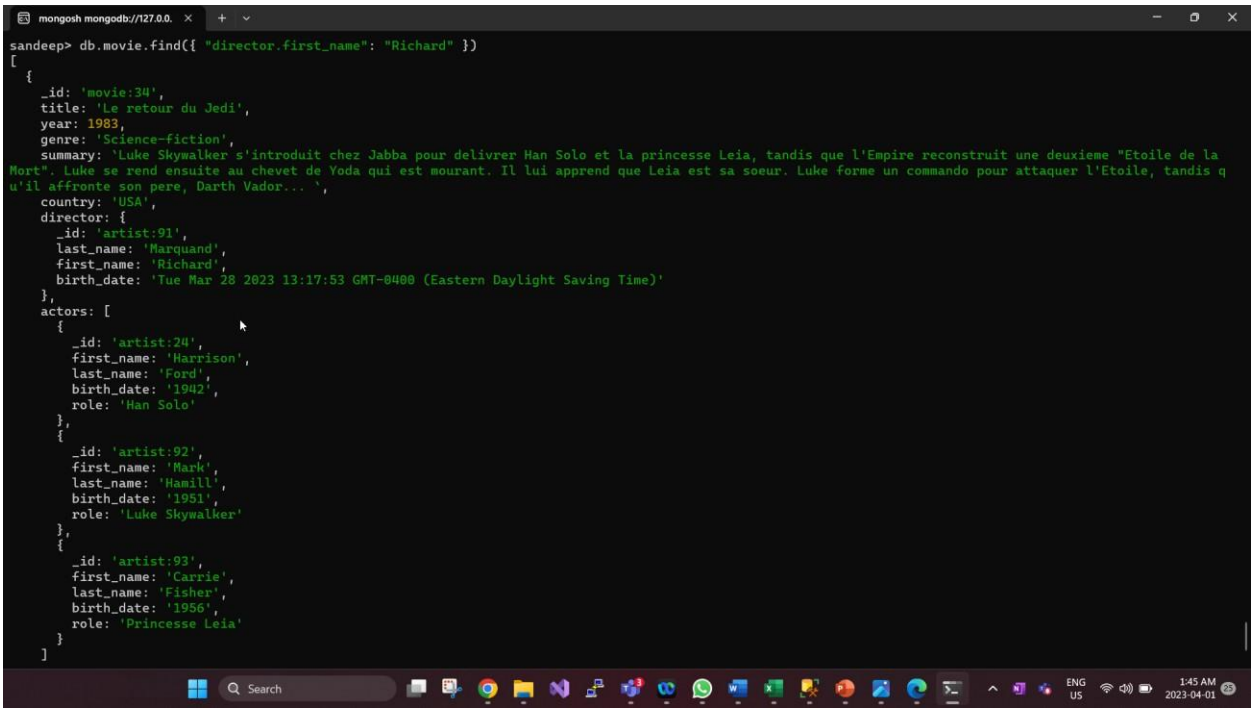


I used count command to check the number of movie collection

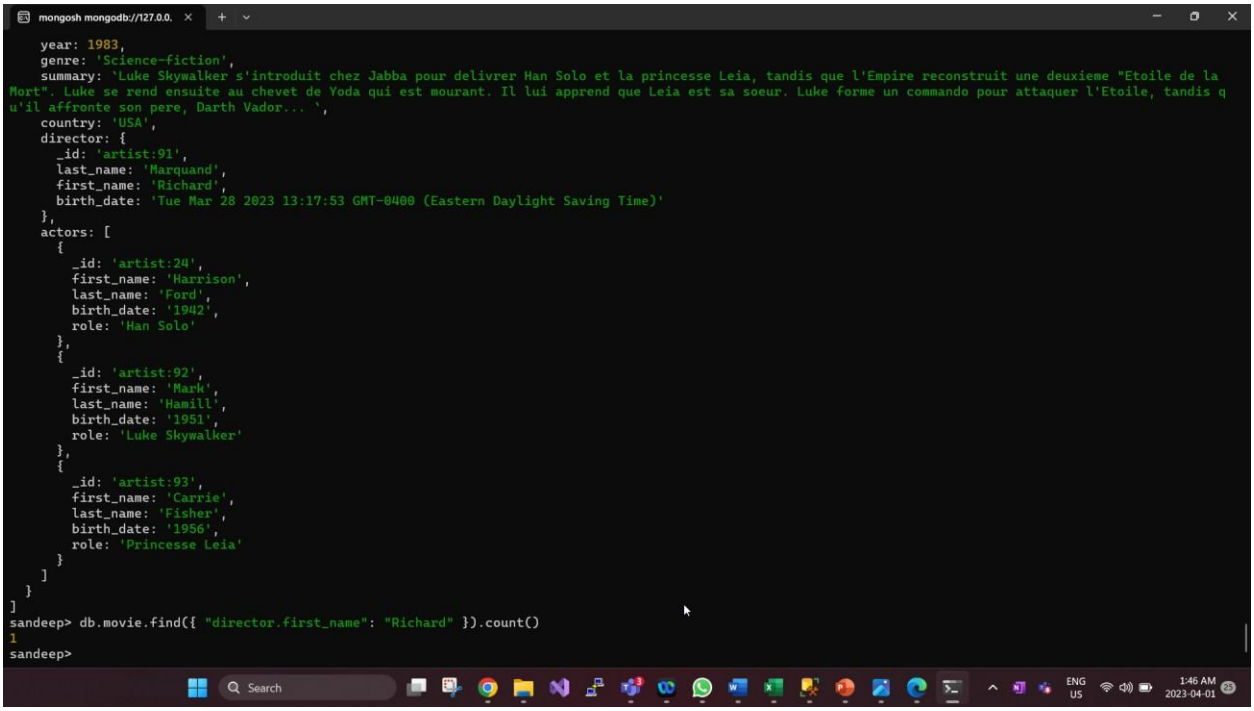


2 Fetch all the document with director firstname “Richard”.

I have first name of director was Richard so I had to fetch Richard which is first name of director while using **db.movie.find({ "director.first_name": "Richard" })**

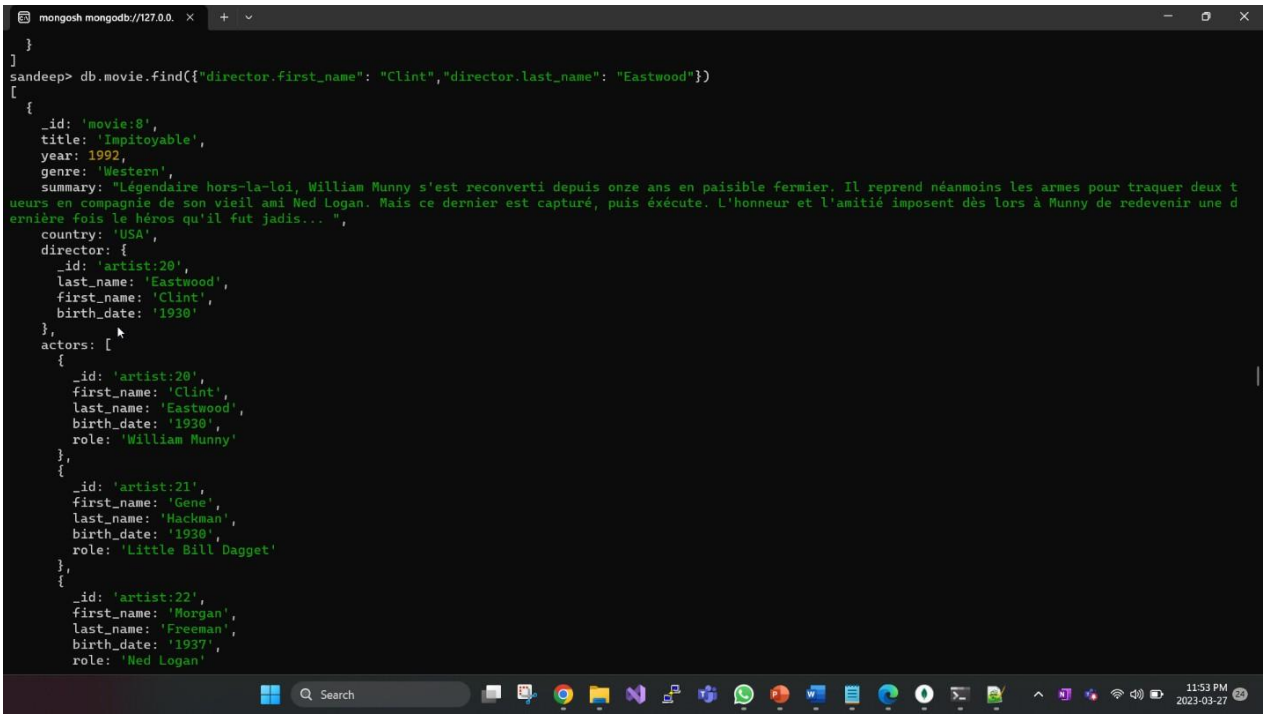


I wanted to know that how many titles where I can see directors name Richard then I used count to know how many there are where I got Richard as shown in Figure



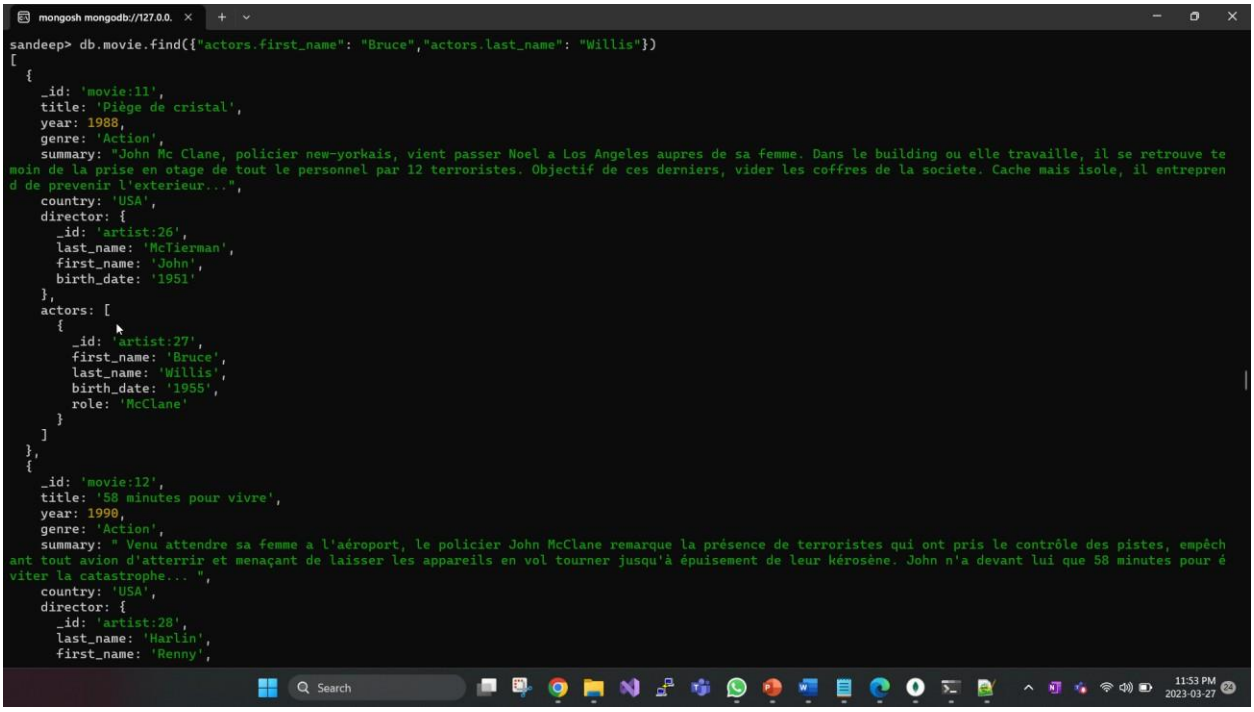
3 Fetch all the document with director “Eastwood Clint”

To fetch all the documents from a MongoDB collection called "movie" that have a director field matching "Eastwood Clint", Here I used find command to fetch the data like db.movie.find({“director.first_name)



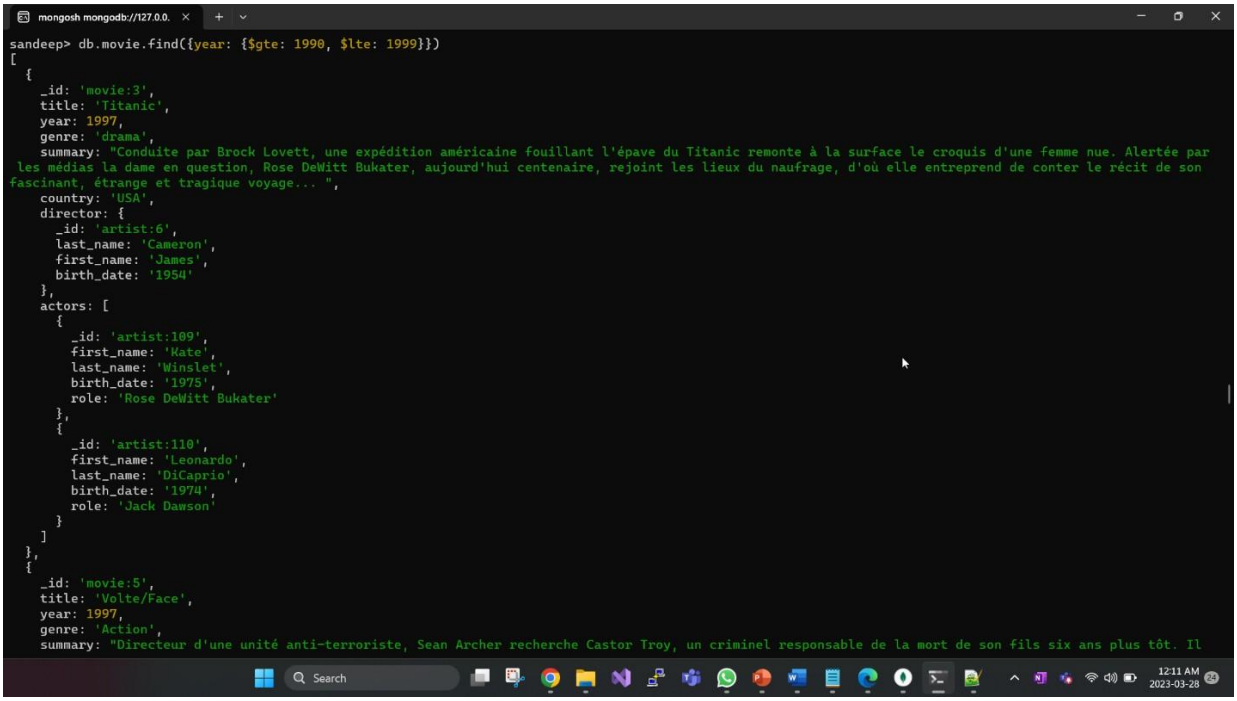
4 Fetch all the document with actor “Bruce Willis”.

To fetch all the documents from a MongoDB collection called "movie" that have an actor field containing "Bruce Willis", I used **db.movie.find({“actors.first_name”: “Bruce”, “actors.last_name”: “Willis”})**



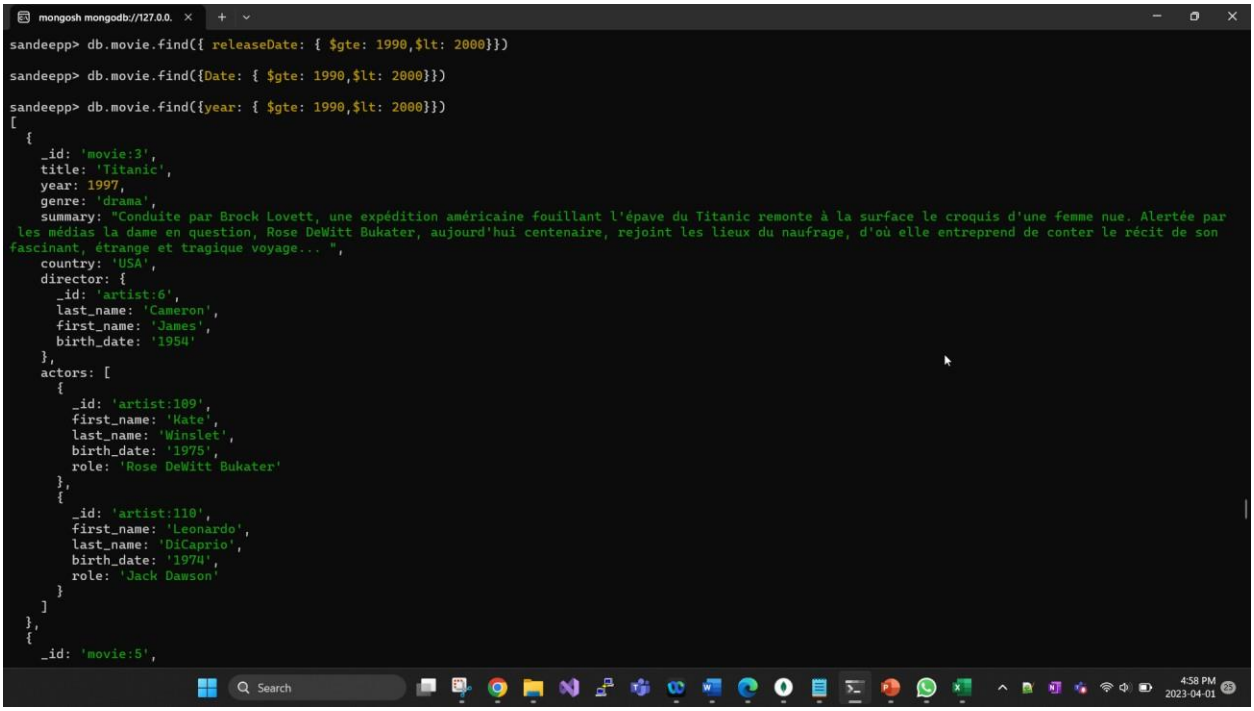
5 Fetch all the document with director birthdate is older than 1900.

To fetch all the documents from a MongoDB collection called "movie" that have a director with a birthdate older than 1900, I used **db.movie.find({year: {$gte: 1990, $lte: 1999}})**

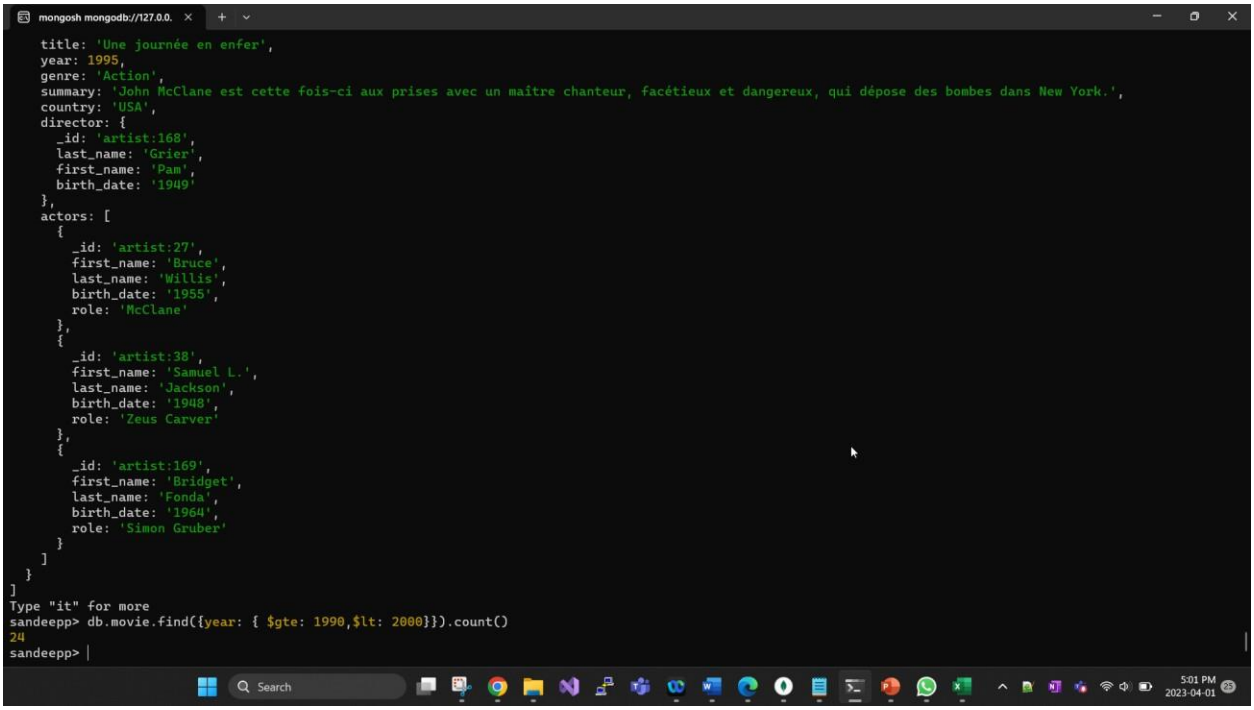


6 Fetch all the document with movies released in the 90s.

I had to fetch all those movie which have been released in 90s so I used find command to find it **db.movie.find({ year : {$gte: 1990. $lt: 2000})**



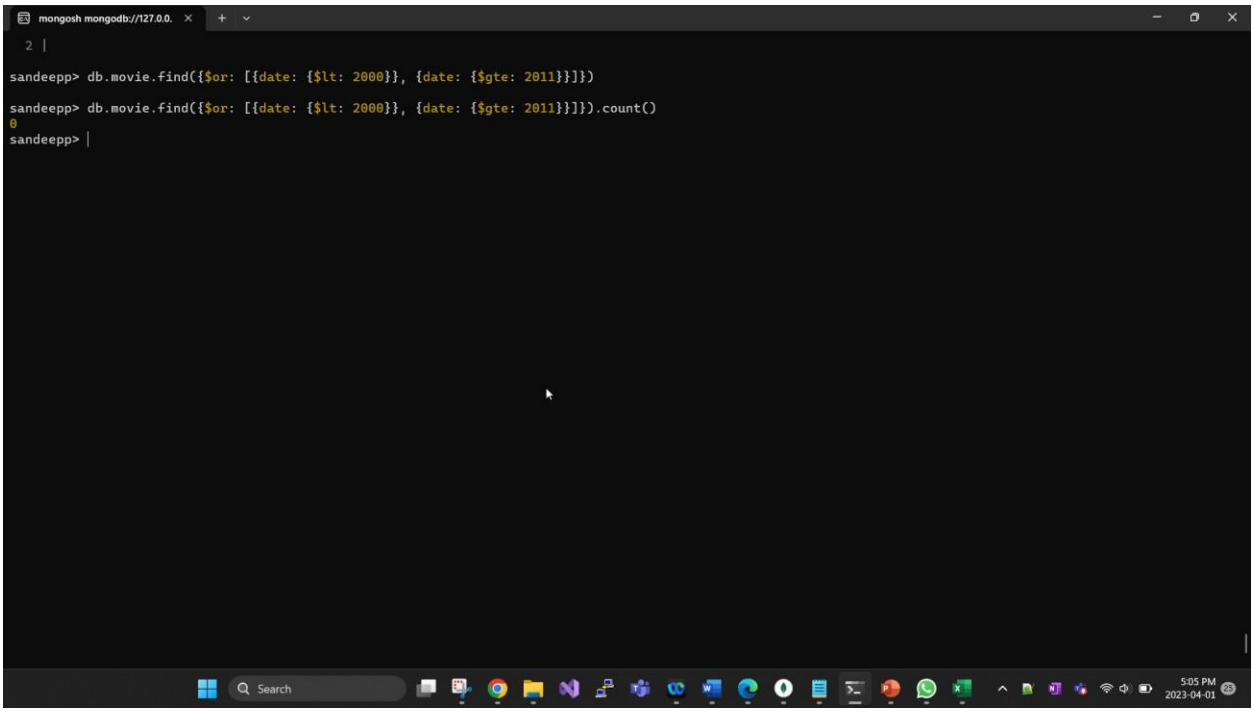
I used count command to know that how many movies which released in 90s so at the end of previous command I used. count() where I got that there are 24 movies as shown in figure.



7 Fetch all the document with movies released before the year 2000 or after 2010.

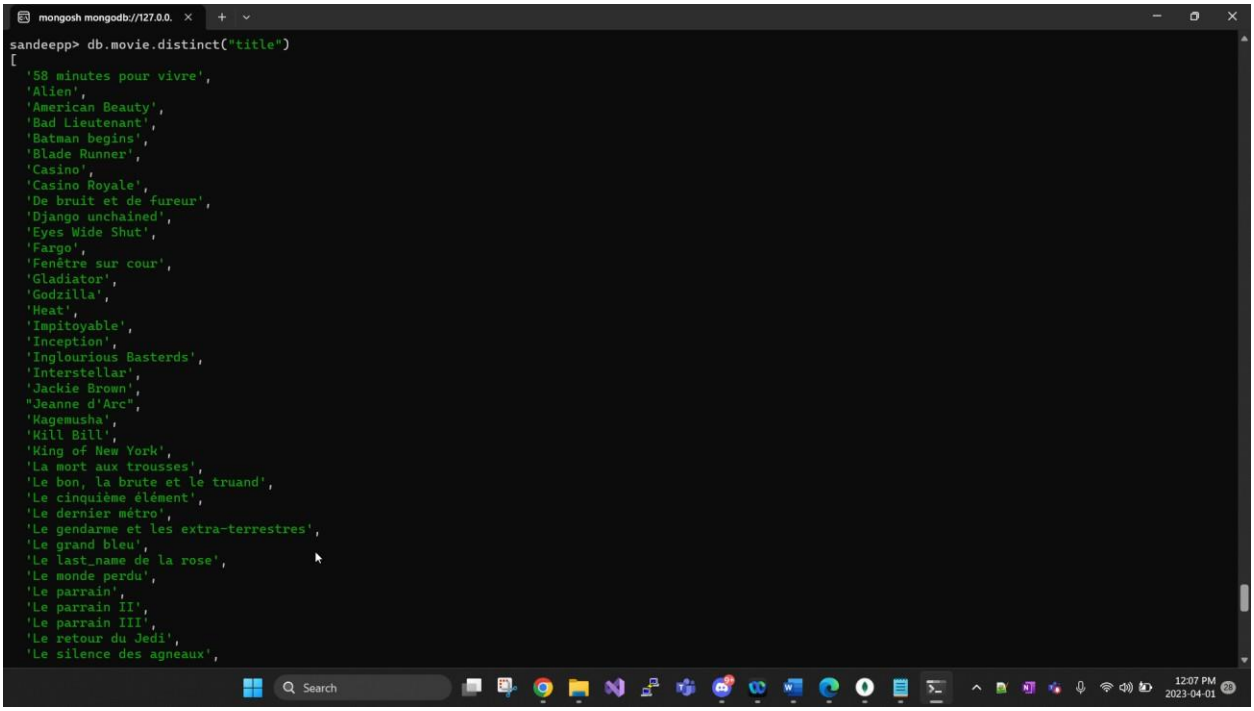
I used find command with set **db.movie.find($or: [{date: {$lt: 2000)}, {date: {$gte: 2011}}]})**

Along with it I used count to check that how many movies are there.



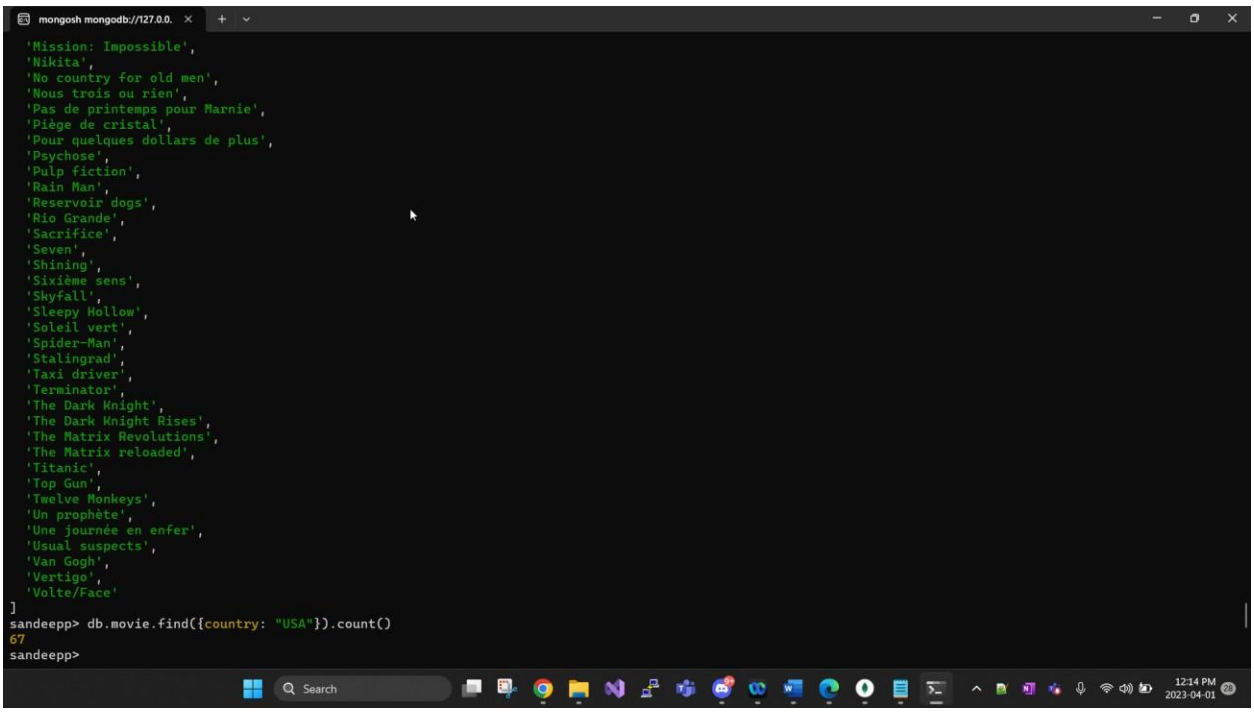
8 Fetch all the movie distinct title.

I had to fetch all movie with distinct title. I had to use **db.movie.distinct("title")** where db is my database and movie is a collection to find distinct title I used filter as shown in figure.



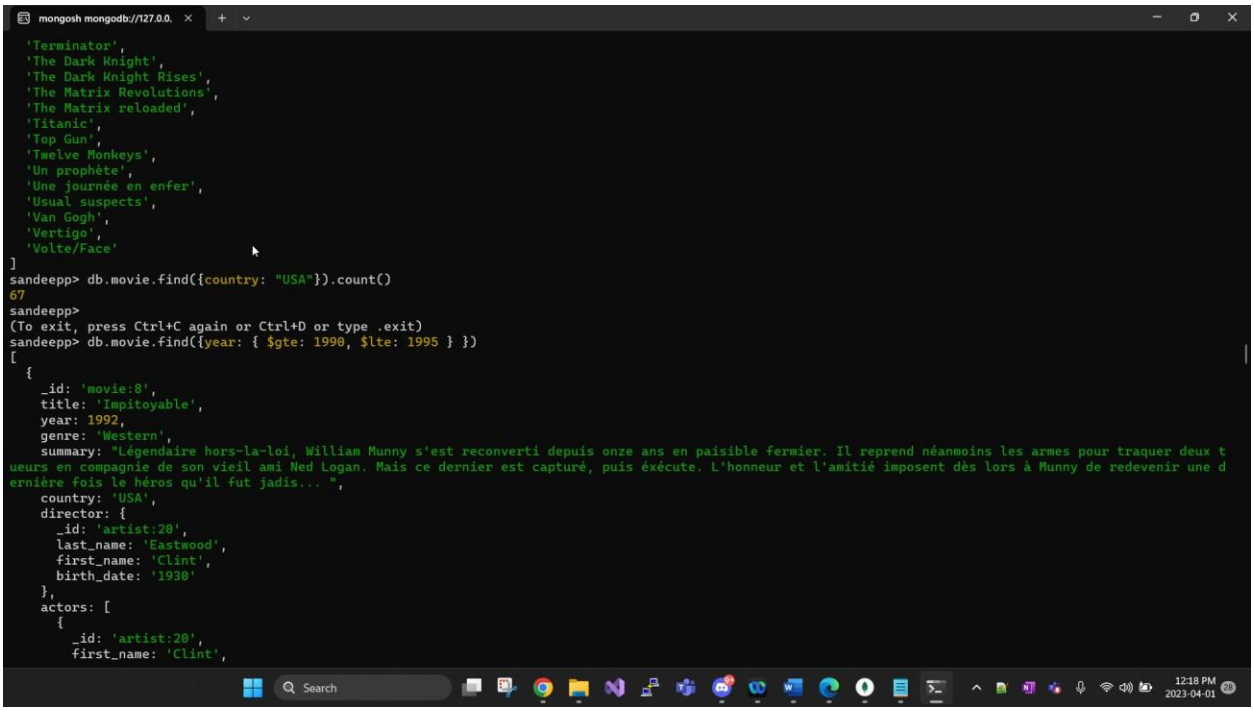
9. Count the number of movies released in USA.

In 9th question I had to count there how many movies which are released in USA then I used a command to fetch the number of movies released in USA like **db.movie.find({country: “USA”})** as shown in figure.

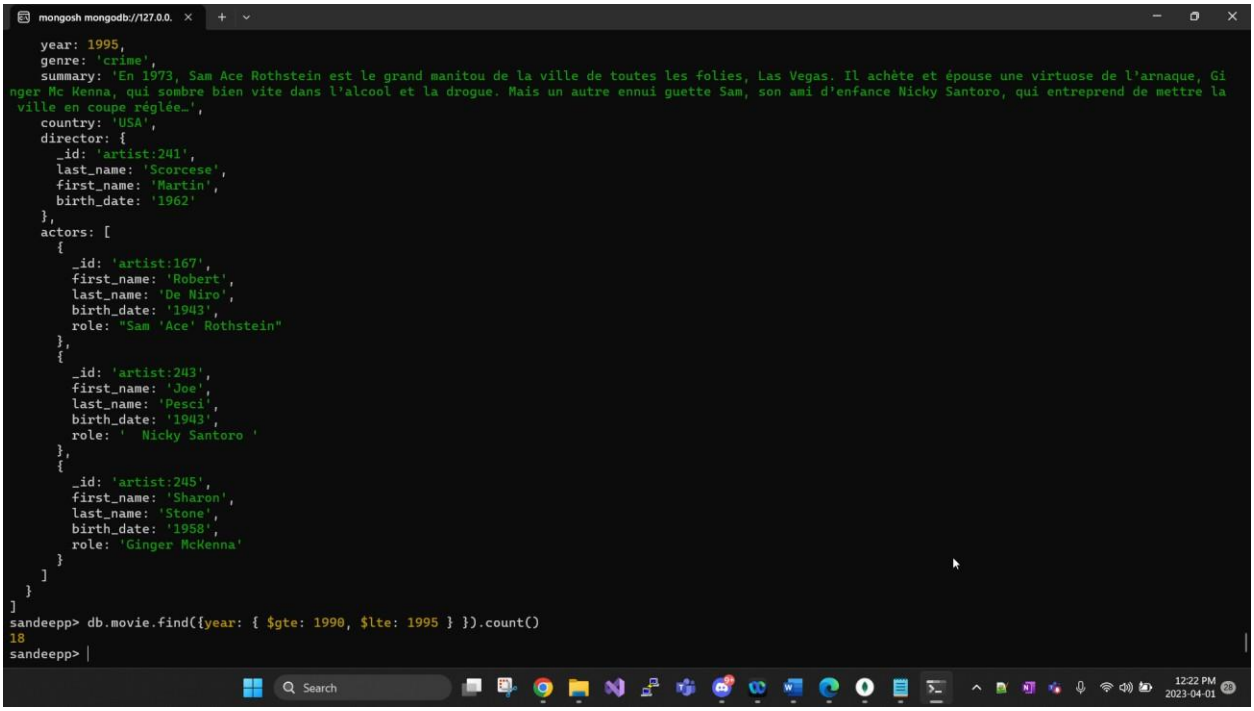


10. Count the movie released between year 1990 to 1995.

I had to fetch year which are released in 1990 to 1995 then I used this command **db.movie.find({year: { $gte: 1990, $lte: 1995 } }).** gte stands for greater than equal and less then equal.

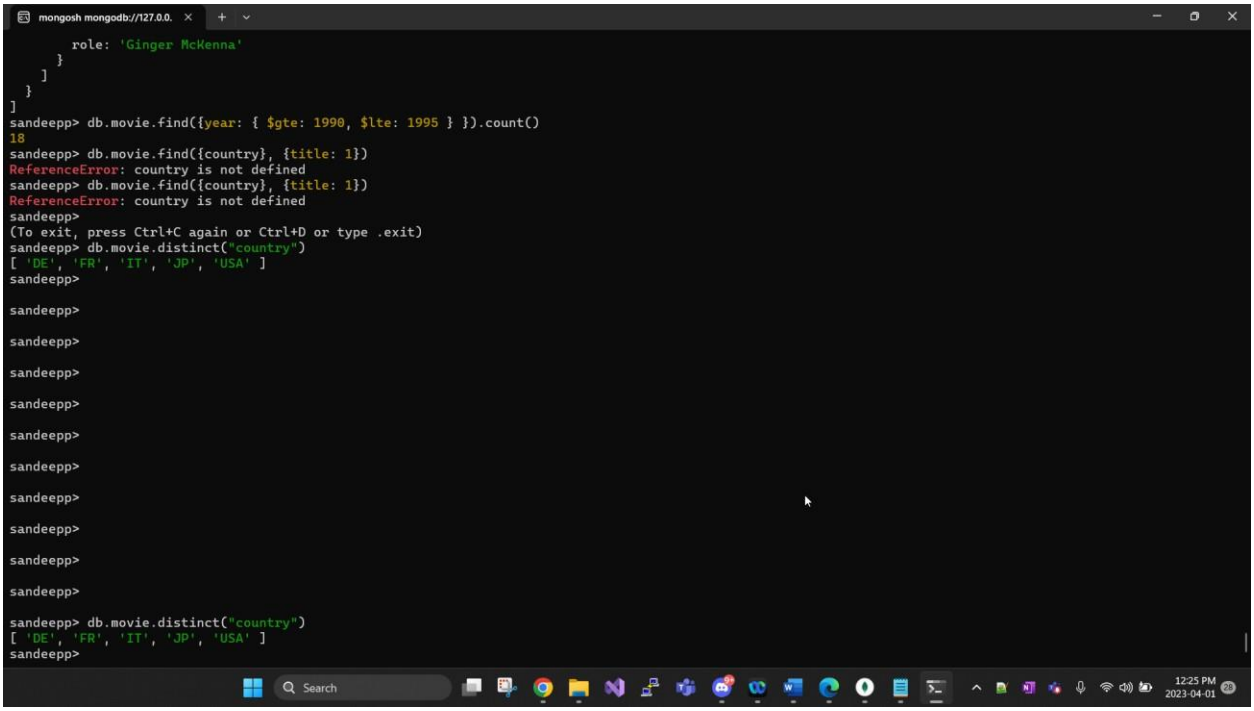


db.movie.find({year: { $gte: 1990, $lte: 1995 } }).count() I used same command



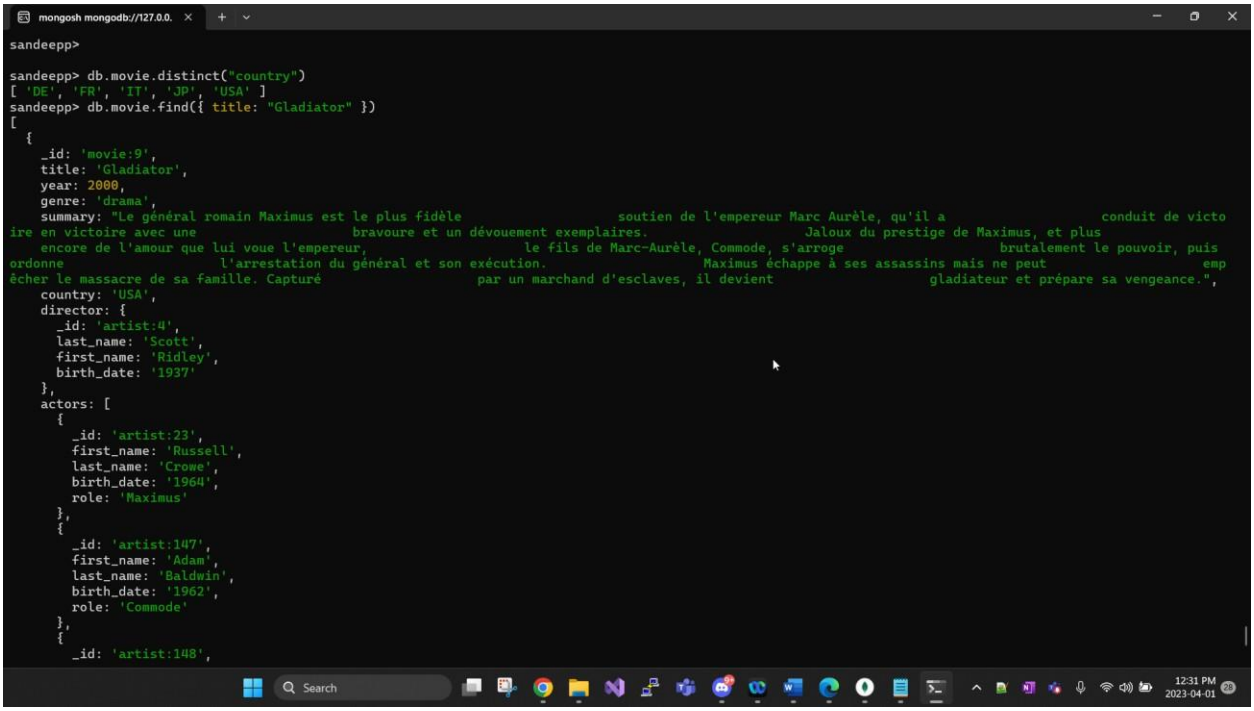
11. Fetch all the country where movies released.

I had to fetch all country where movies have been released then we can use distinct command which I have used and shown in figure ##########
db.movie.distinct(“country”)



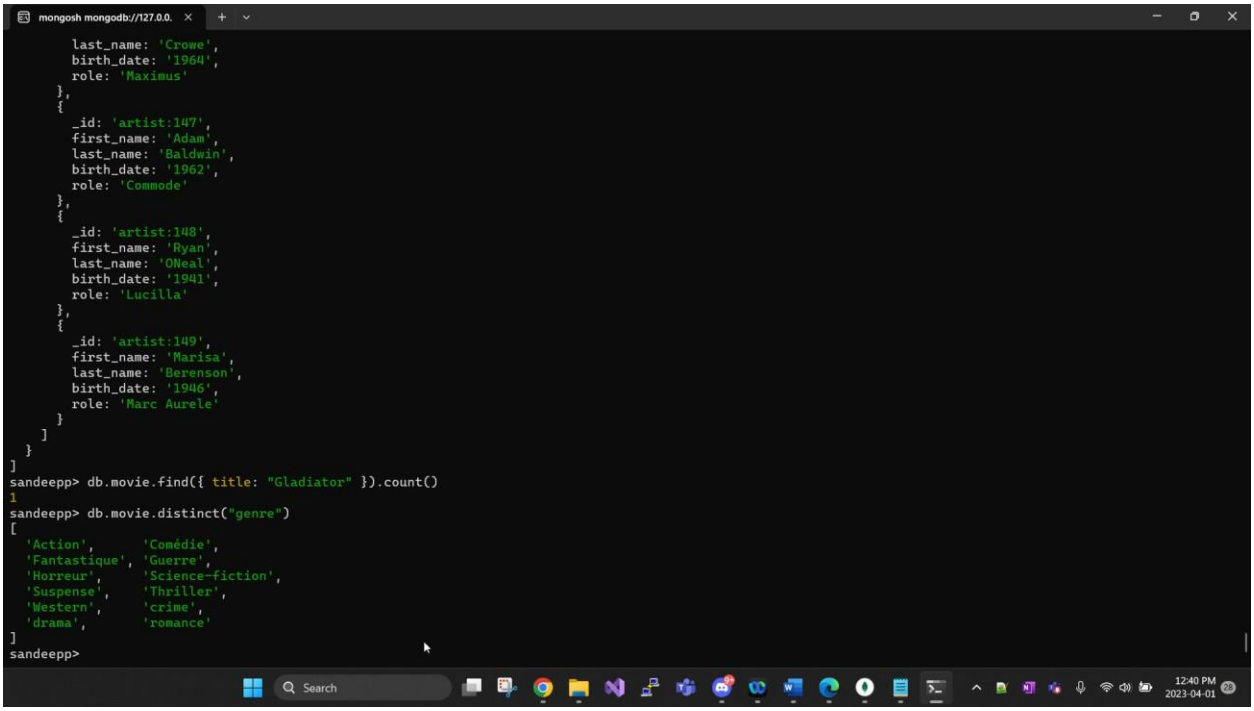
12. Fetch all the document with movies titled as “Gladiator”.

I used find command to **db.movie.find({ title: "Gladiator" })** I had to fetch movie title Gladiator. There is only one movie which has title Gladiator.



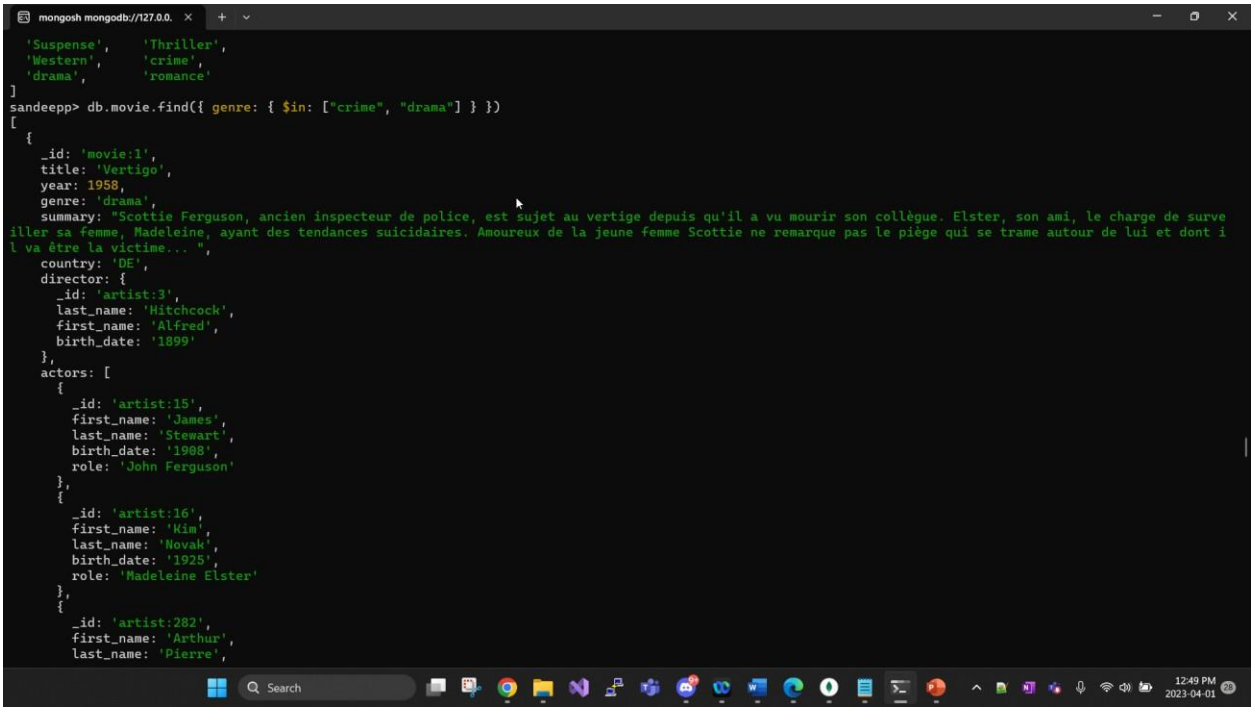
13. Fetch distinct genre values of movies.

I had to fetch only genre where I used ############**db.movie.distinct("genre")** where I found all distinct genre.

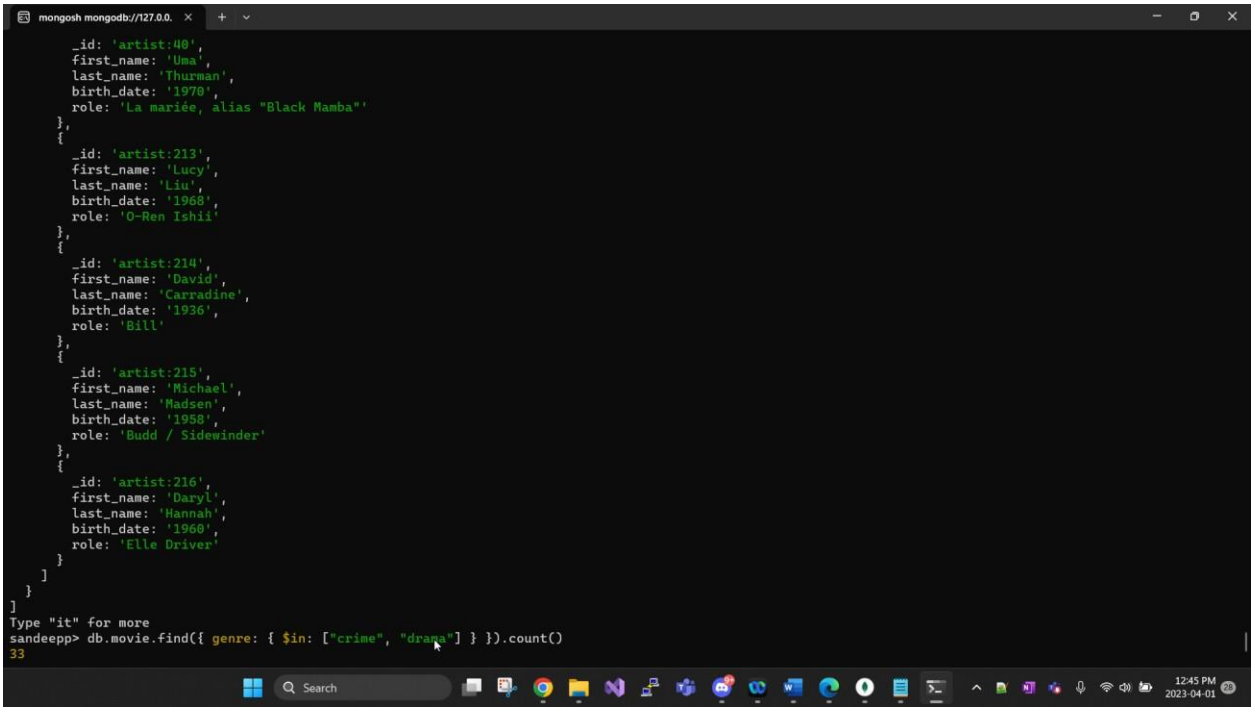


14. Fetch all the document with movie “crime" or "drama" genre.

To fetch all the documents with movies having the "crime" or "drama" genre from the movies collection, we can use the find () method in MongoDB with the $in operator to match documents where the genre field is either "crime" or "drama".
db.movies.find({ genre: { $in: ["crime", "drama"] } })

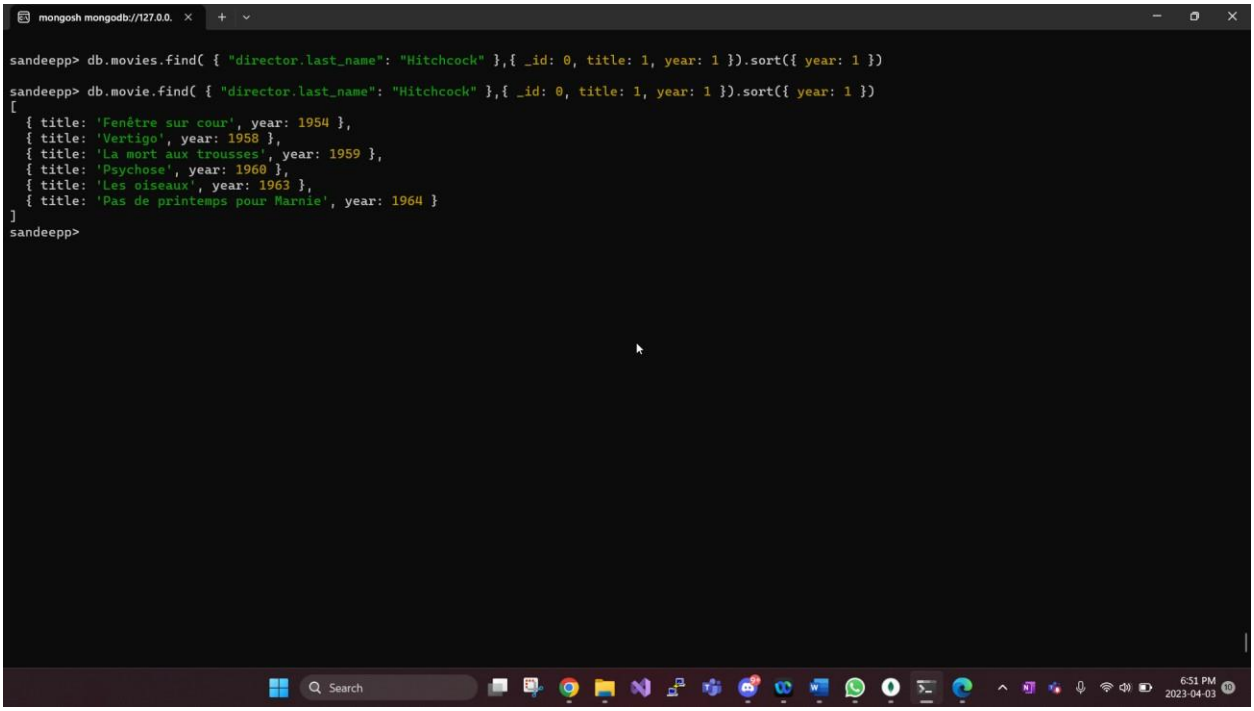


I just used count command to get that how many movies are with these genres.



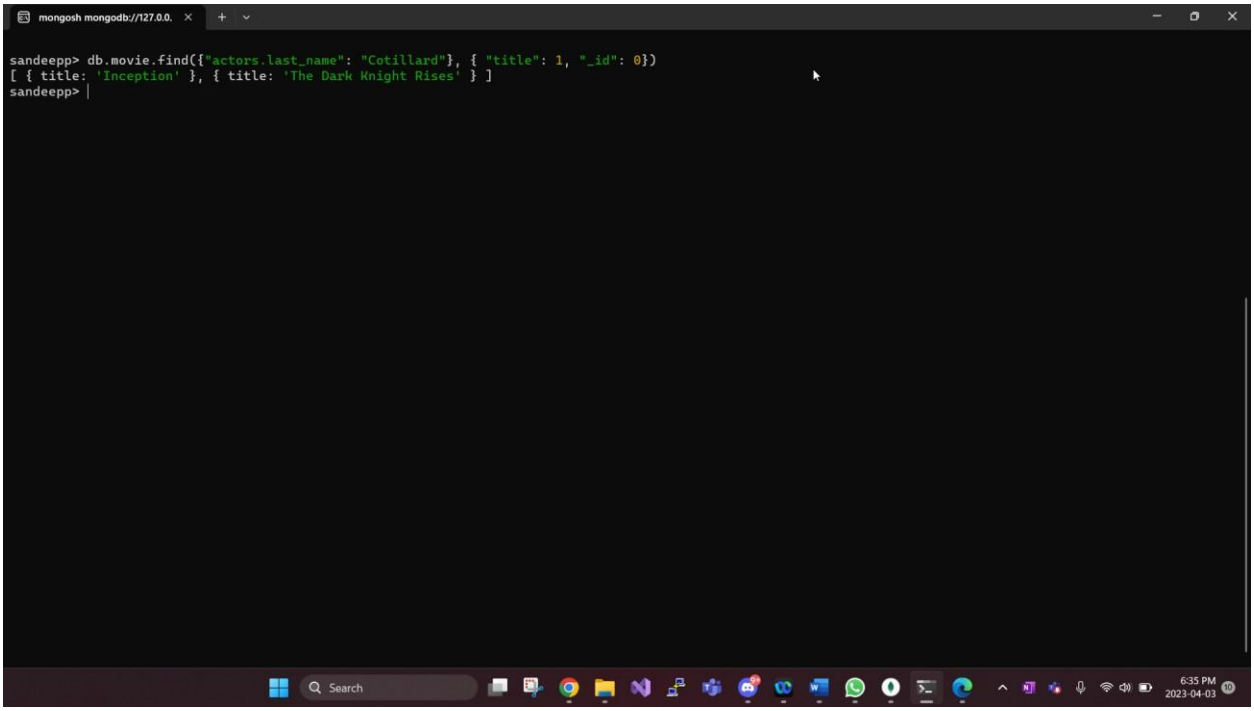
15. Fetch all the movies directed by "Hitchcock", display only title and year and sort them by year.

I used the find() method to query the movies collection for movies directed by "Hitchcock". I specified the filter { "director.last_name": "Hitchcock" } to match movies directed by "Hitchcock". **db.movies.find({ "director.last_name": "Hitchcock" },{ _id: 0, title: 1, year: 1 }).sort({ year: 1 })**



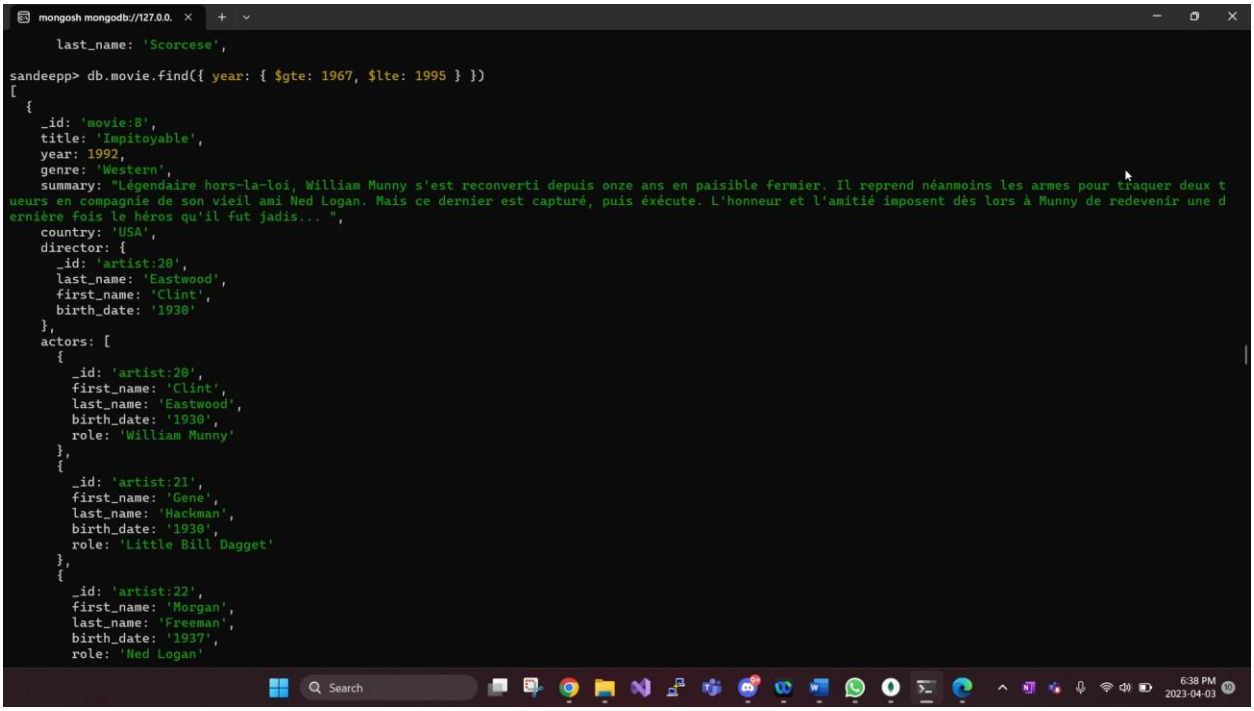
16. Fetch the list of movies where "Cotillard" played.

Here I have used find command to fetch the movie where I needed to find and I used **db.movie.find({“actors.last_name”: “Cotillard”}, {“title”: 1, “_id”: 0})**

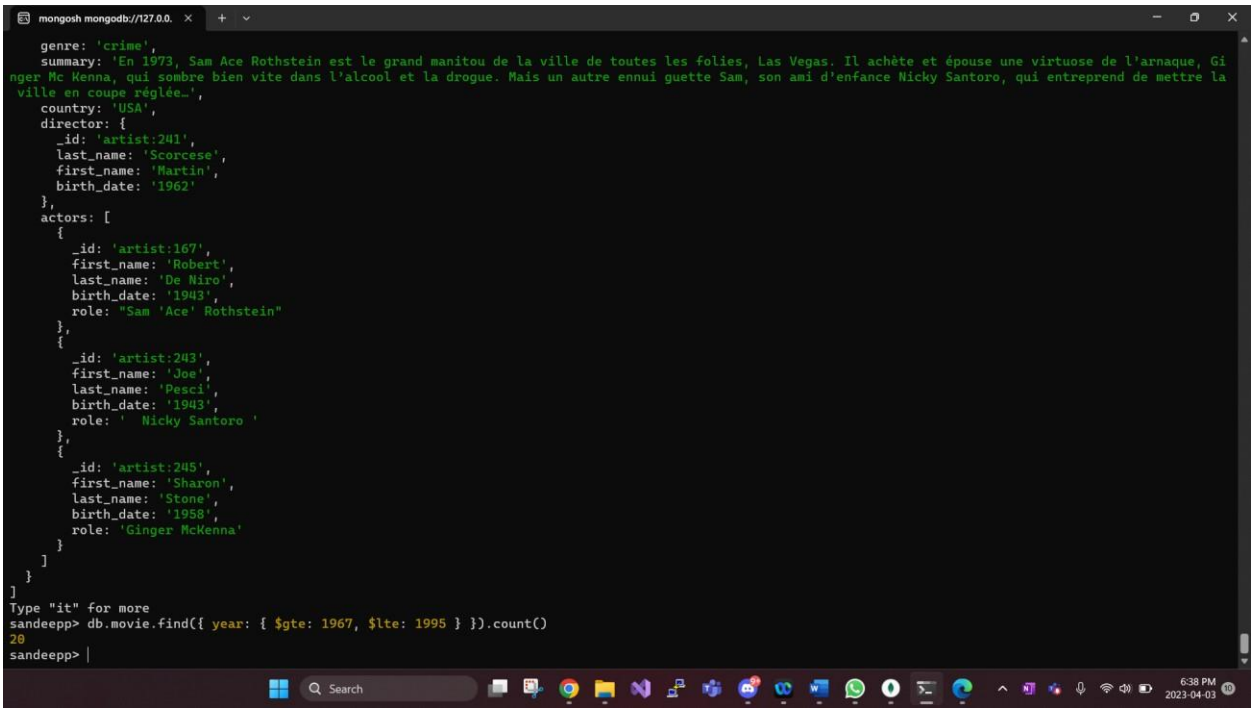


17. Fetch the list of movies released between 1967 and 1995.

In this to fetch the year when movie released between 1967 and 1995 and used find command with this function **db.movie.find({year : { $gte: 1967, $lte: 1995} })**



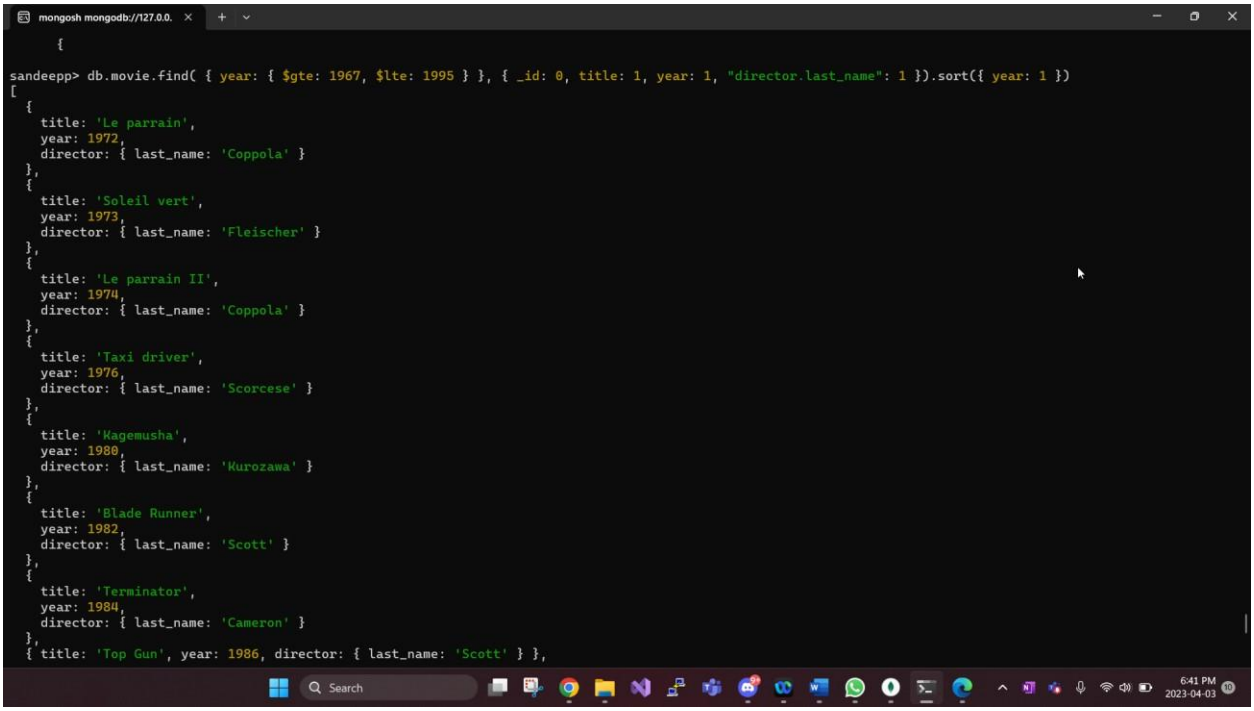
Here I used count to check the number .



18. Fetch the list of movies released between 1967 and 1995, by displaying only title, year, director's last name sorted by year.

I used the find() method to query the movies collection for movies with a year between 1967 and 1995 and used the $gte and $lte operators to specify the range of years.

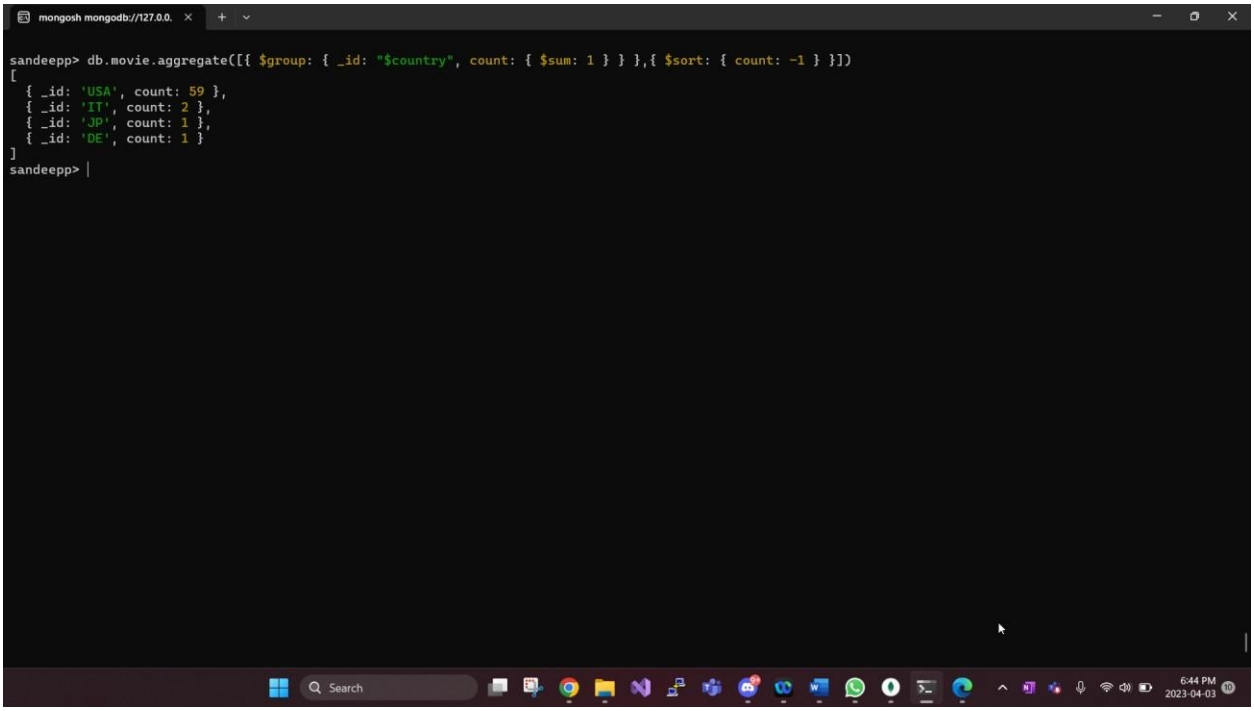
db.movie.find({ year: { $gte: 1967, $lte: 1995 } }, { _id: 0, title: 1, year: 1, "director.last_name": 1 }).sort({ year: 1 })



19. Fetch the number of movies by country.

In this we find number of movies by country using find command and used

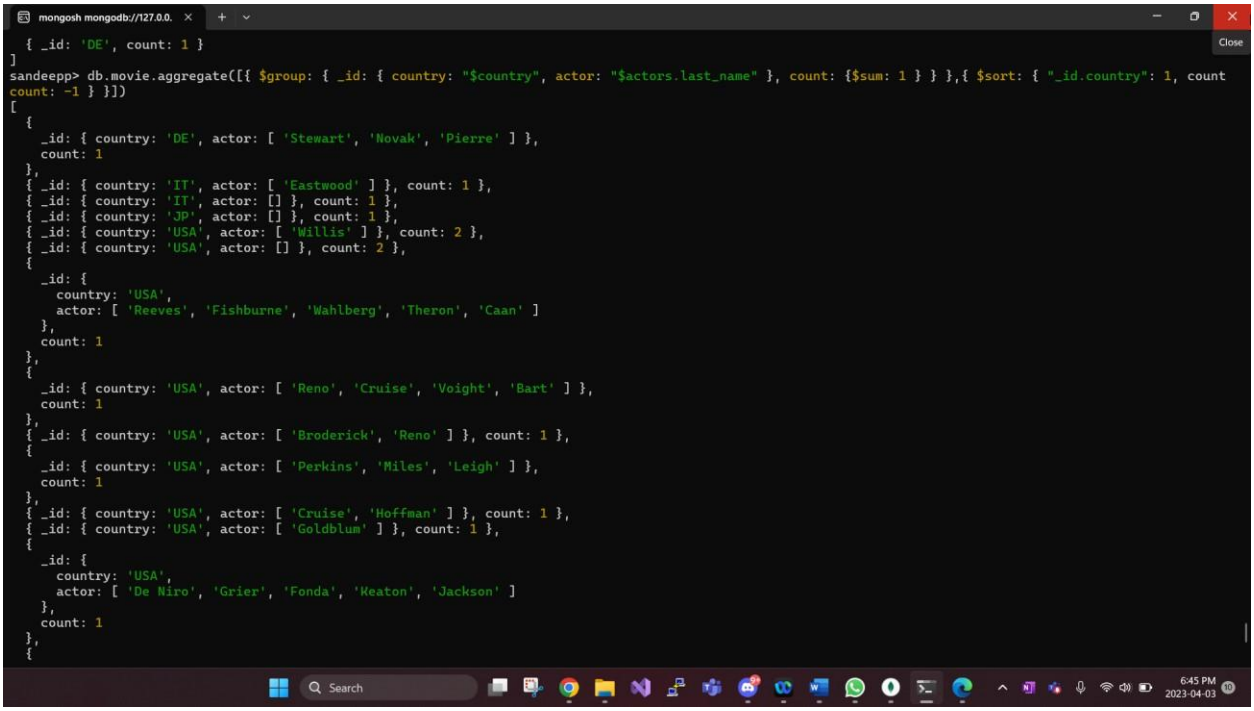
db.movie.aggregate([{ $group: { _id: "$country", count: { $sum: 1 } } },{ $sort: { count: -1 } }])



20. Fetch the number of movies by country and actor

As in previous question we used find command to fetch now we have to fetch number of movies by country and actor

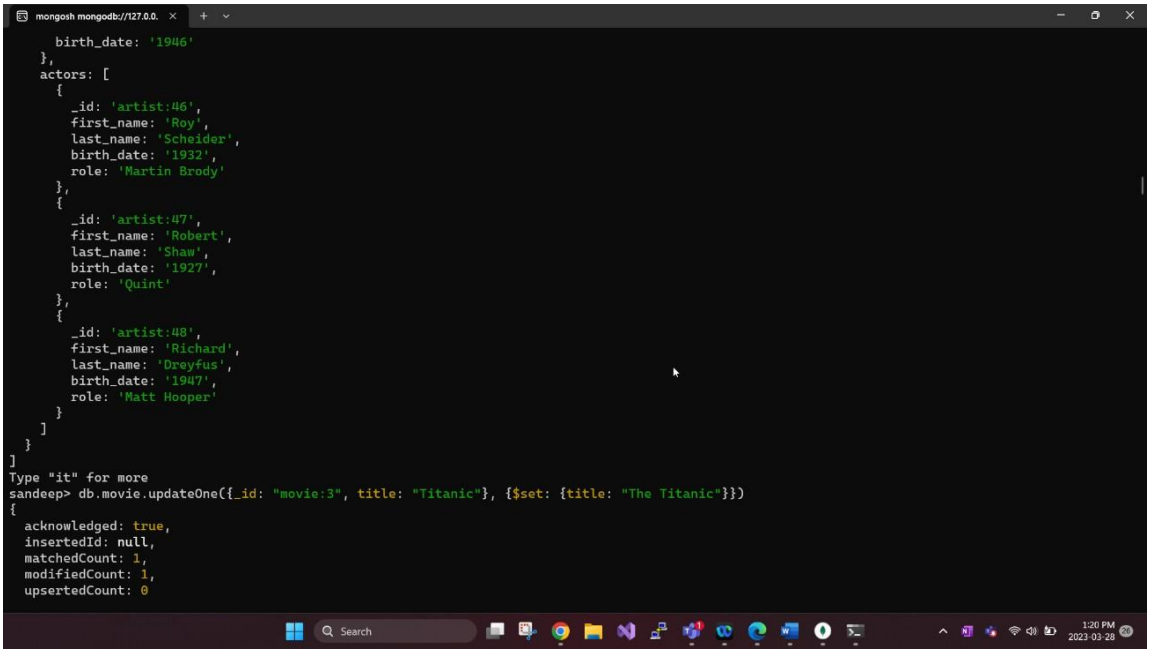
db.movie.aggregate([{ $group: { _id: { country: "$country", actor: "$actors.last_name" }, count: {$sum: 1 } } },{ $sort: { "_id.country": 1, count: -1 } }])

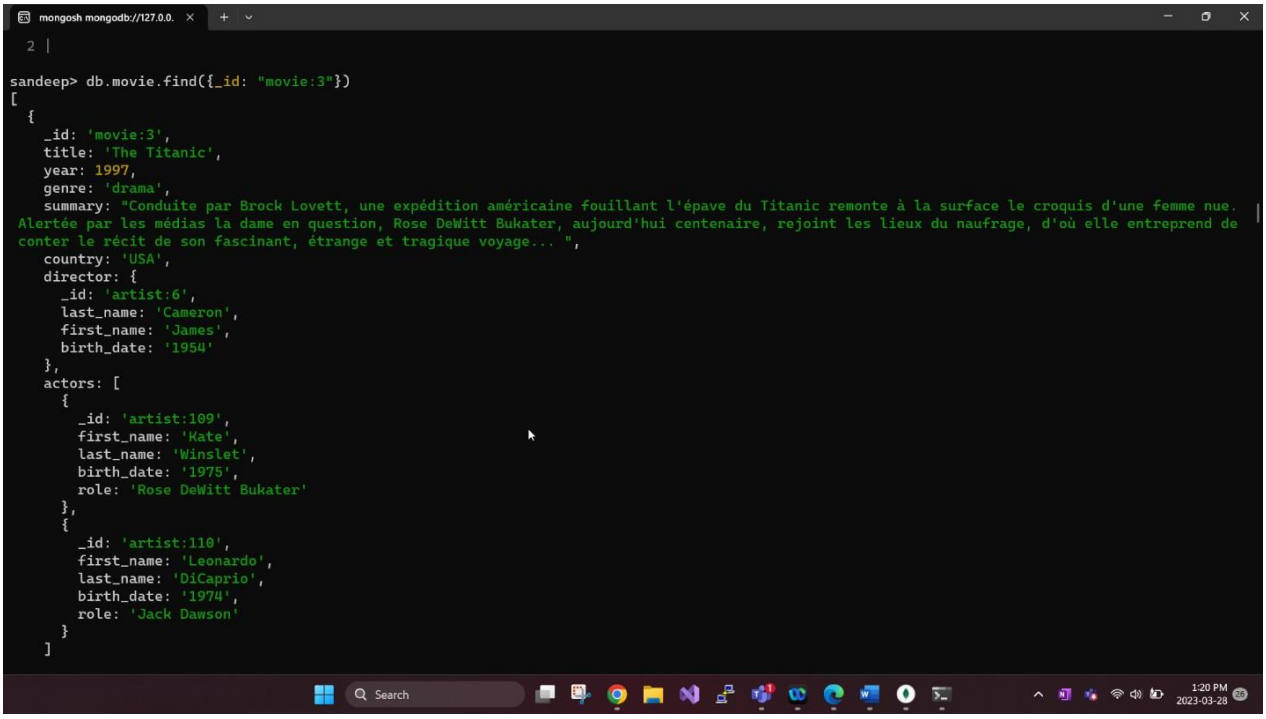


B Write the Script to update document on movie collection.

1. **Update the movie with title “The Titanic” from “Titanic” where _id is "movie:3".**

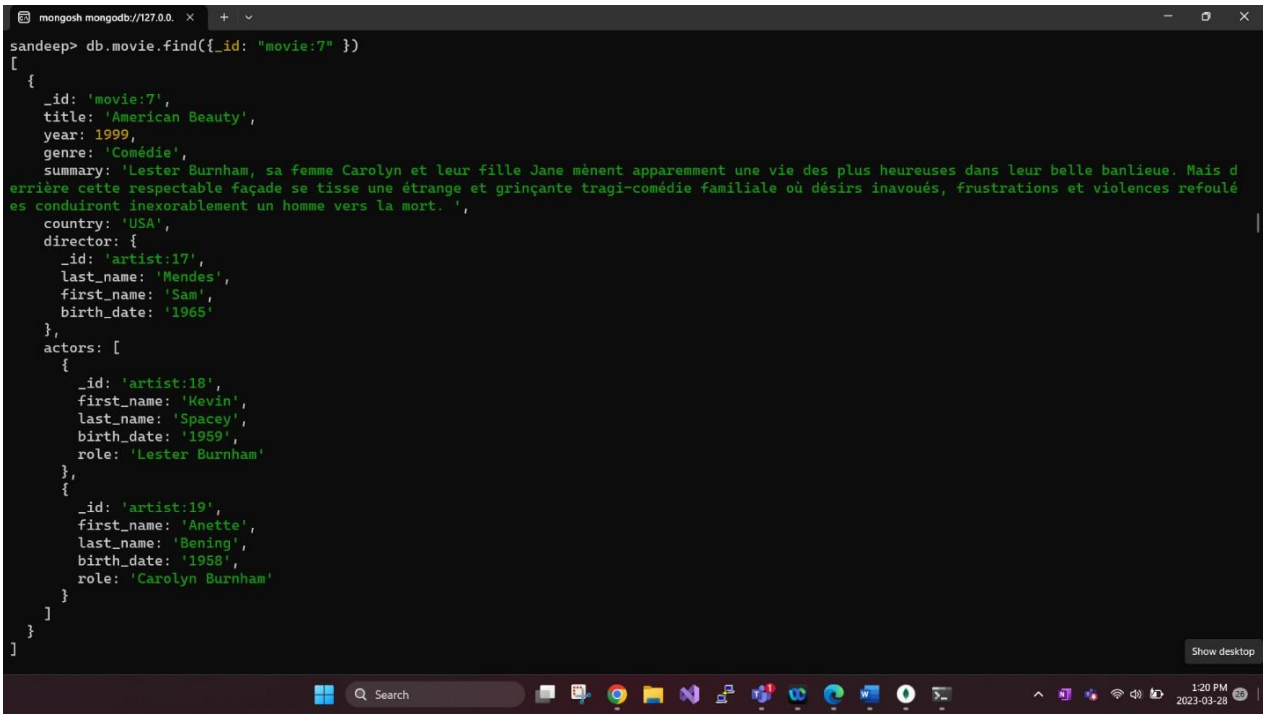
   I have a movie script which is a collection of Sandeep database and I used update command and filtered it which is **db.movie.updateOne({_id: “movie:3” , title: “Titanic”}, {$set: {title: “The Titanic”}}).** For more explanation I just wanted to update one id that’s why I used updateOne where I told the title of movie and used operation to update it as shown in figure.



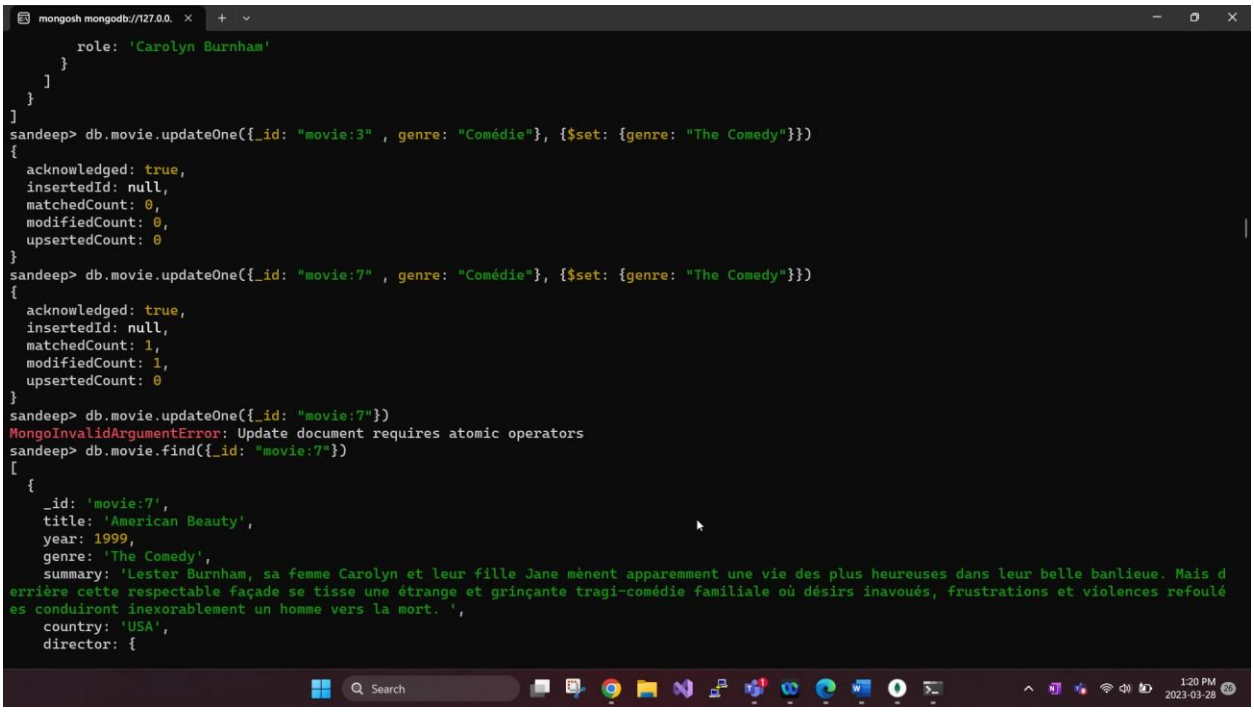


2. **Update the movie with genre “The Comedy” from "Comédie" where _id is “movie:7”.**

   I had to update Comédie to The comedy in movie: 7 for that

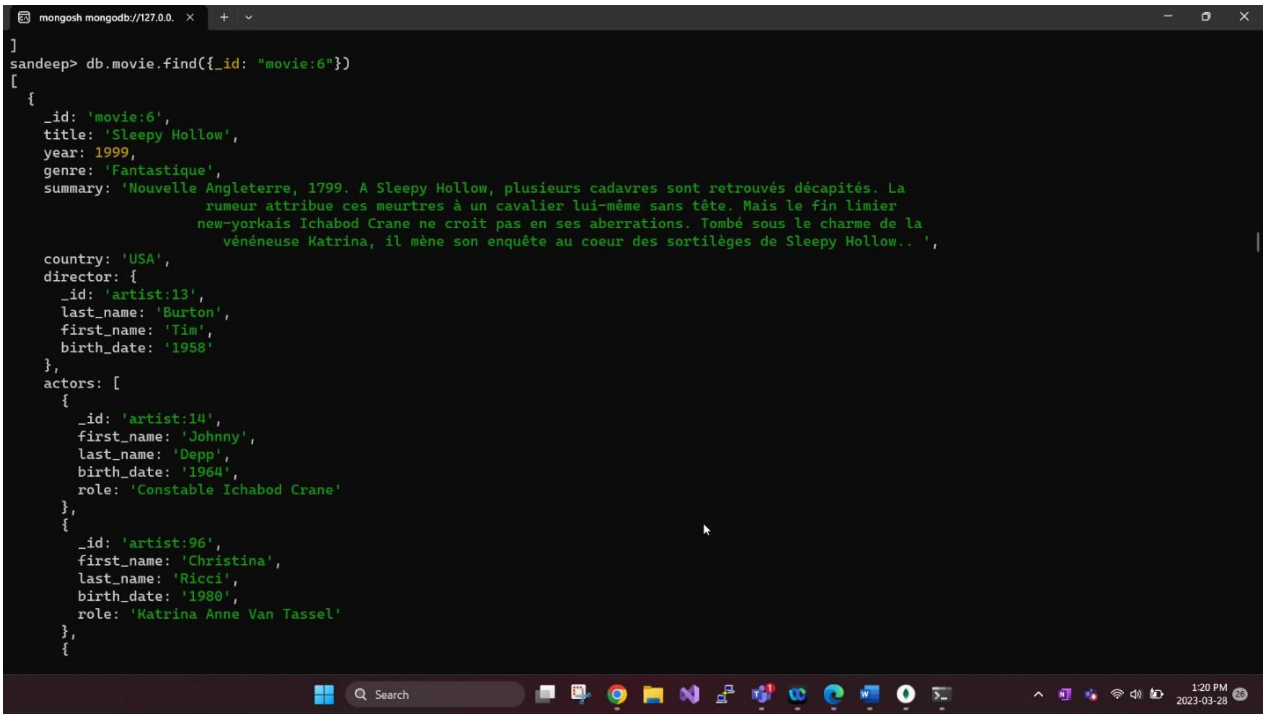


I used update command **db.movie.updateOne({_id: “movie:7”, genre: “Comédie”} {$set:{genre: “The comedy}})** as I shown in figure.

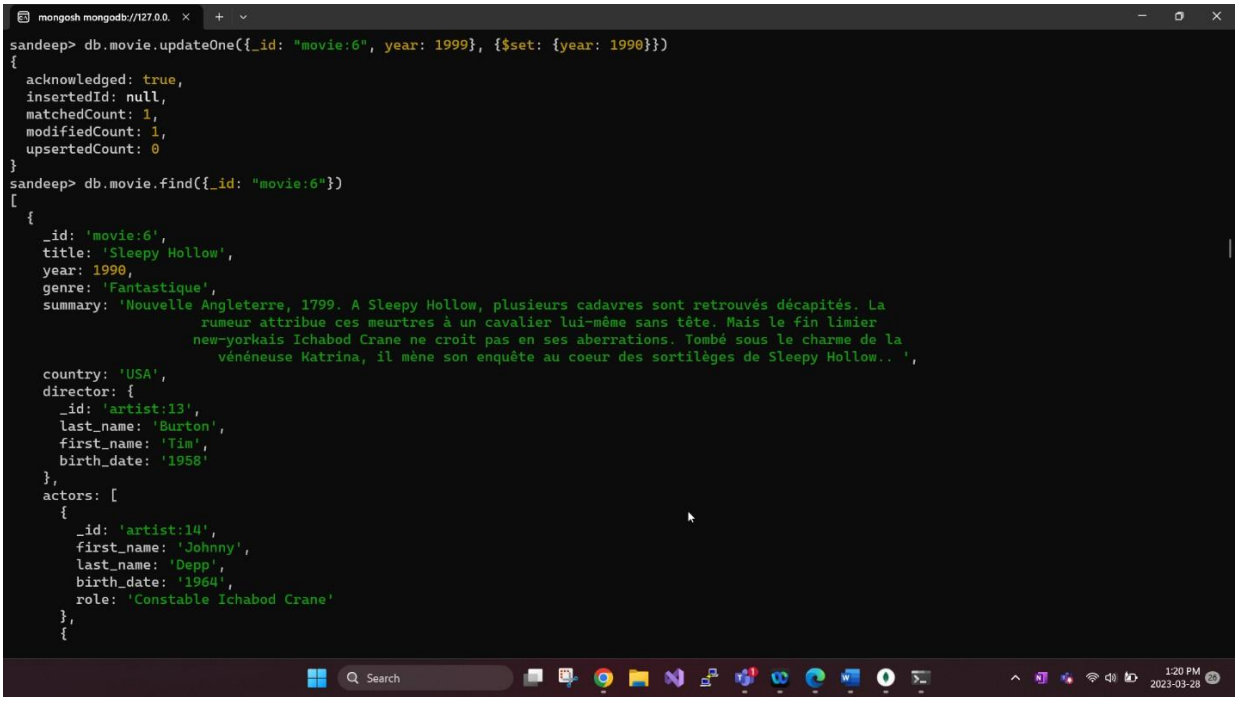


3 Update the movie with year 1990 from 1999 where _id is "movie:6"

In question 3rd I have to update the year of the movie id 6.

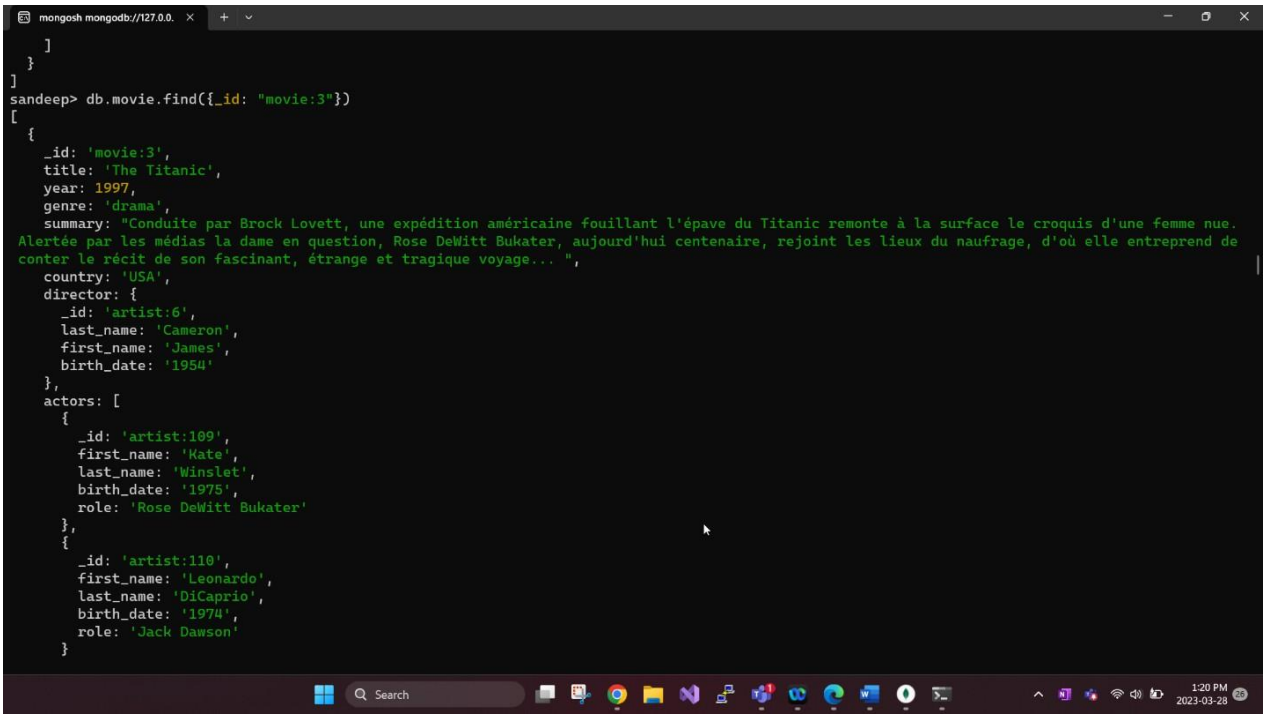


I used command **db.movie.updateOne({_id: “movie:6” , year: 1999} {$set:{year: 1990}})** as shown in below figure. Db is a database and movie is a collection.

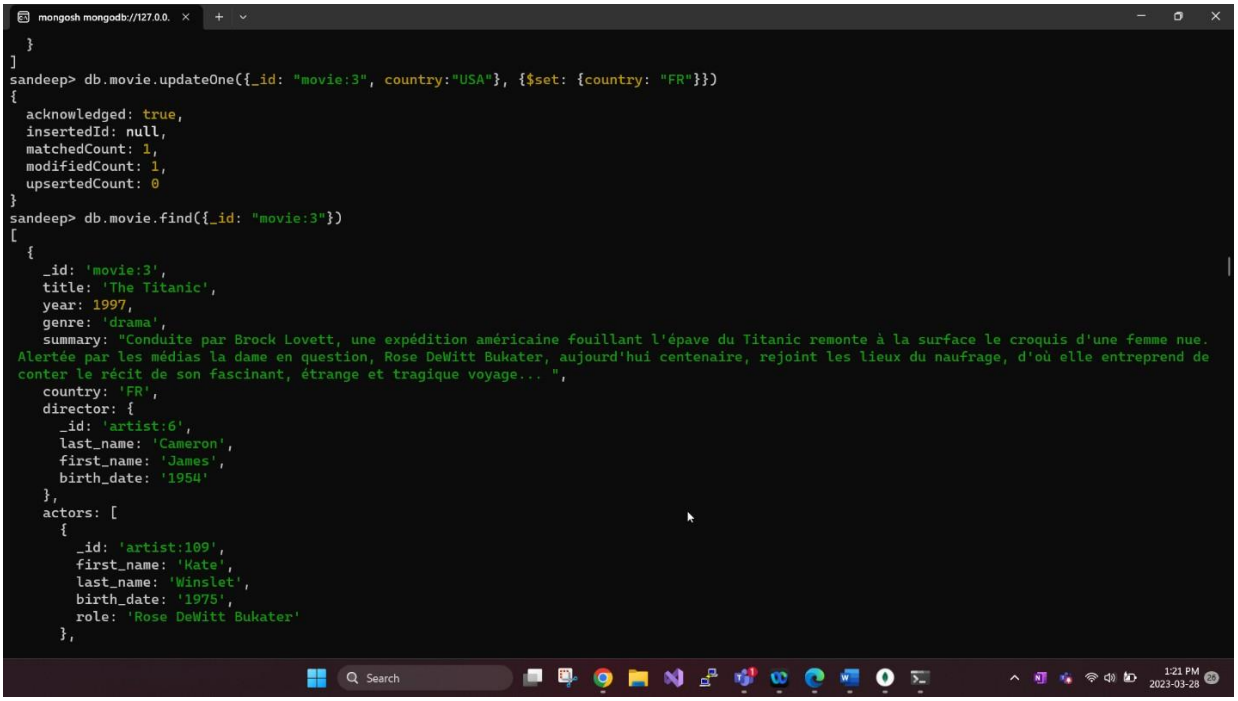


4. Update the movie with country “FR” from "USA" where _id is “movie:3”.

In this we must update only one id of movie in which we have mentioned USA and update with FR as shown in figure.

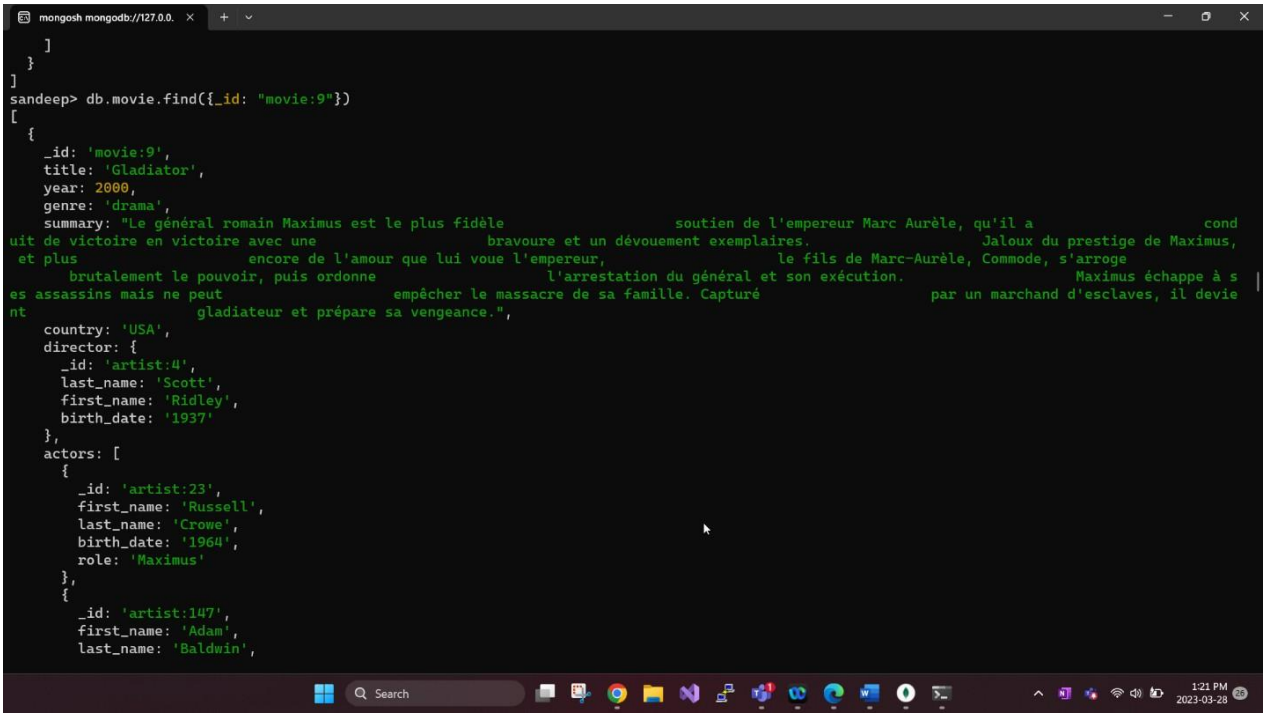


I used the command **db.movie.updateOne({_id: “movie:3”, country: “USA” } {$set:{country: “FR”}})** as I shown in the below figure.

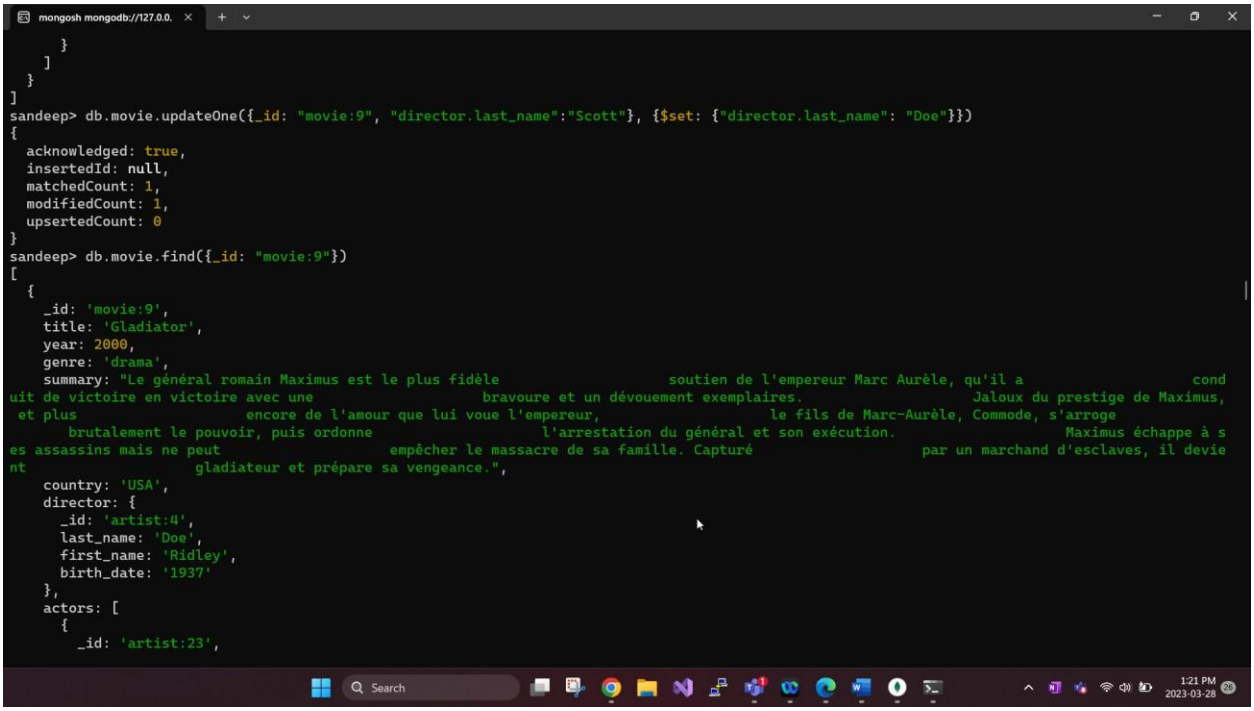


5 Update the movie with director’s last_name is “Doe” from "Scott" where _id is "movie:9".

I have a id which has named _id: “movie:9” there was only one updation which I have to do. I had directors’s last name Scott which I have updated with Doe.

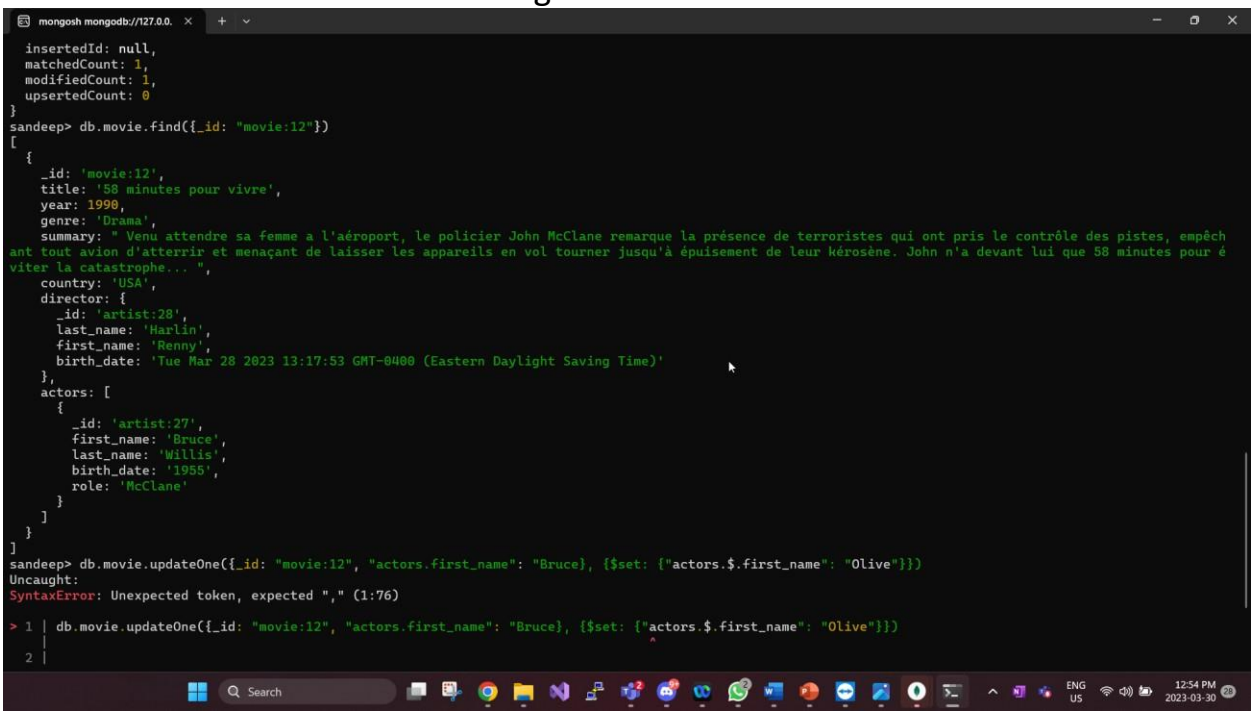


In the above figure I showed the actual result but In upcoming figure I updated with **db.movie.updateOne({_id: “movie:9” , “director.last_name”: “Scott” } {$set: {“ director.last_name”: “Doe”}})**

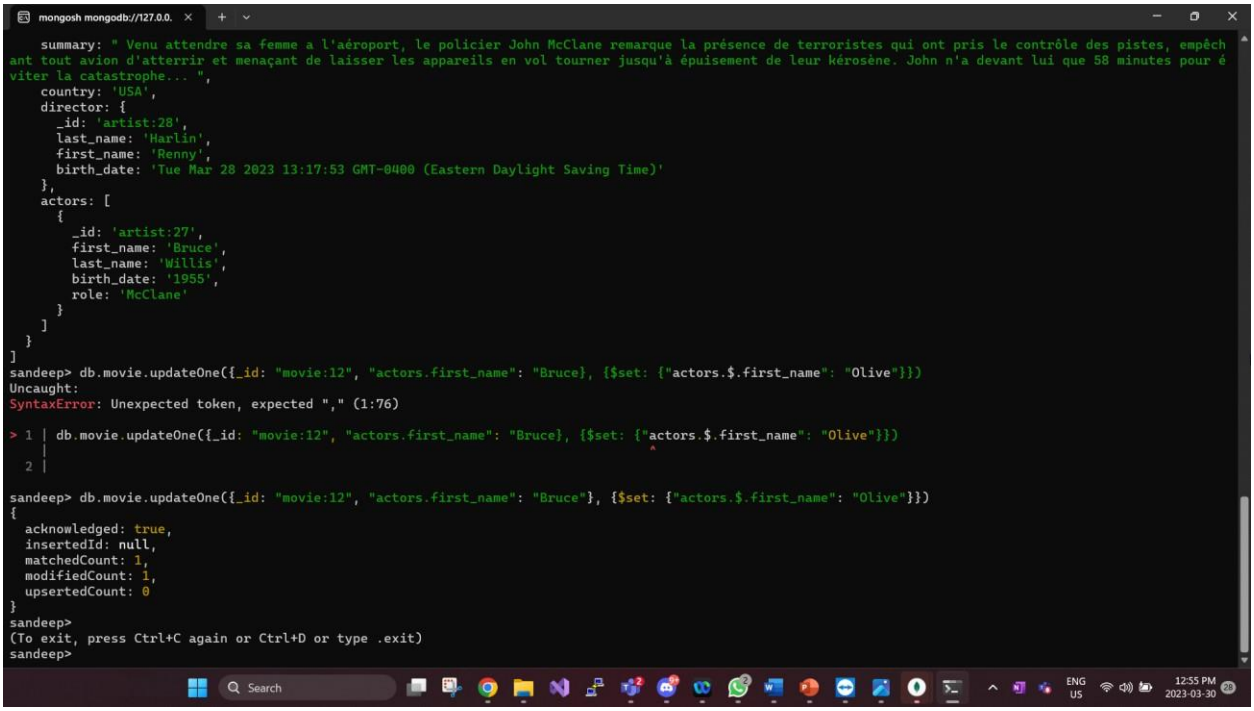


6 Update the movie with director's first_name "Olive" from "Bruce" where _id is "movie:12".

There was no name Bruce of director in movie: 12 what as I know there was a name bruce of actor which I changed from Bruce to Doe.

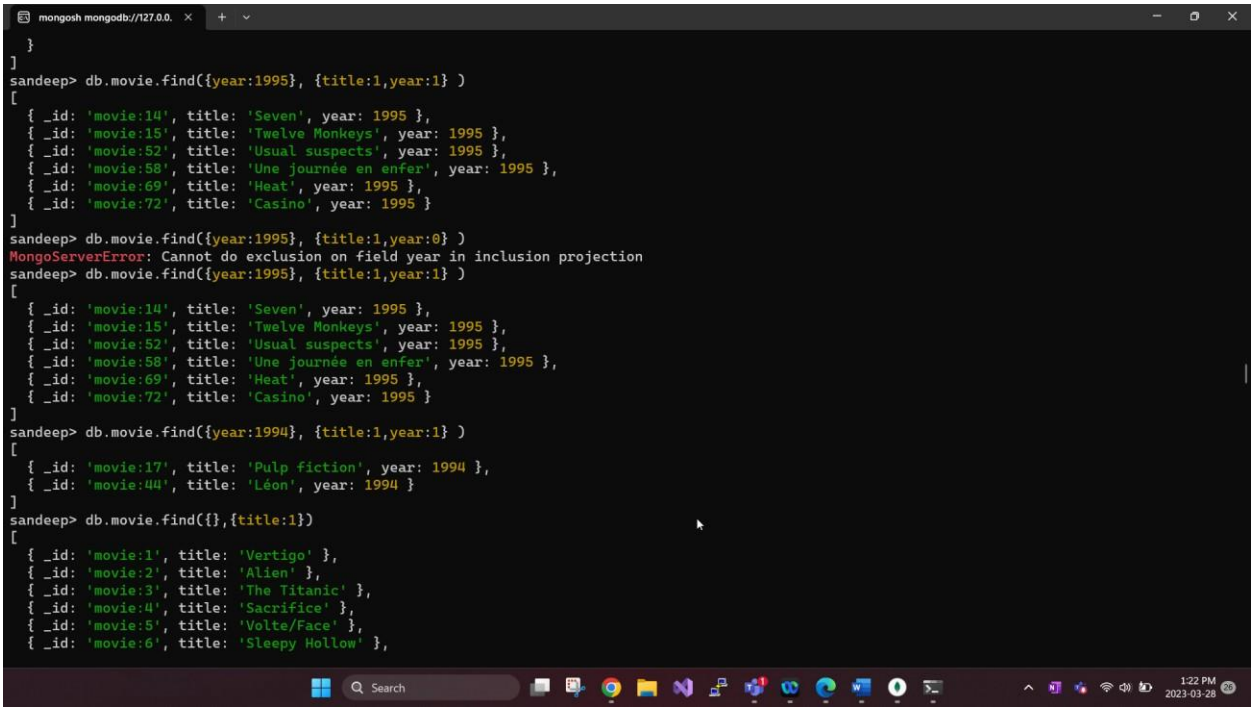


I changed the actors name from Bruce to Olive. I used command **db.movie.updateOne({_id: “movie:12”, “actors.first_name”: “Bruce”}, {$set: {“actors.$.first_name”: “Olive”}})**

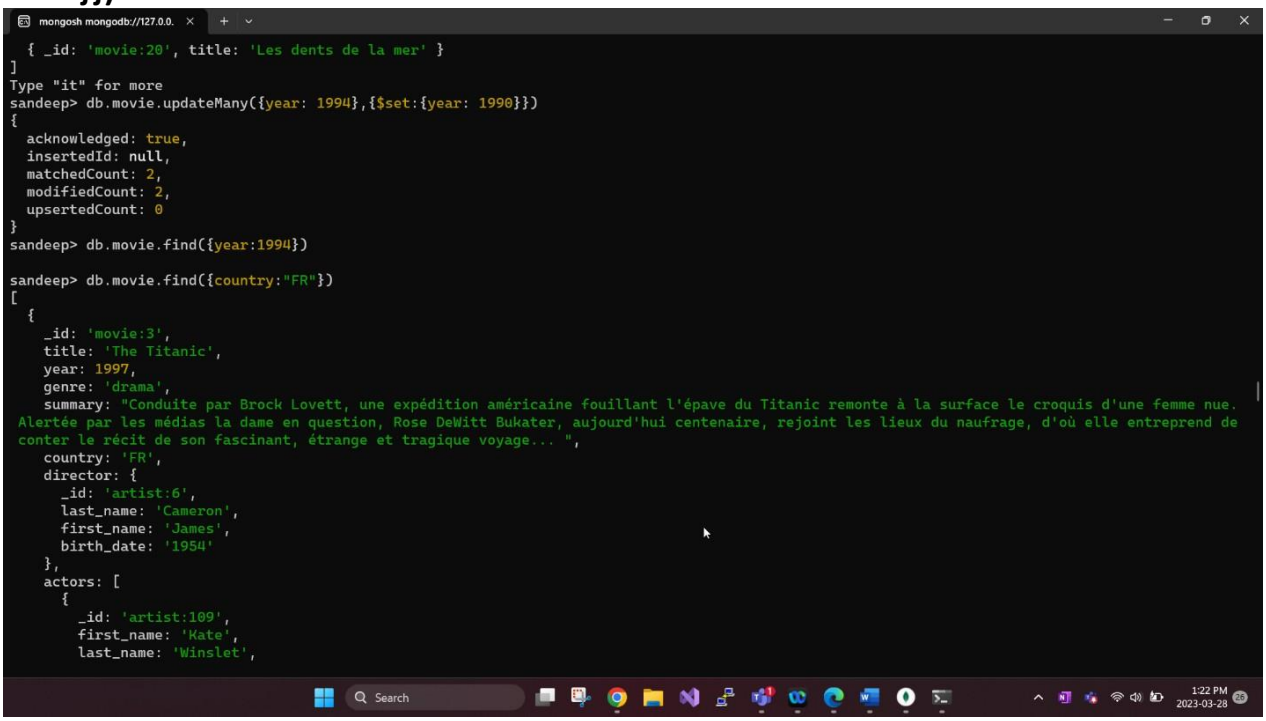


7 Update the movies with year 1990 where year is 1994.

I have had a movie script where I got 1994 and I had to change to 1990 as I shown in figure where I had only 2 id where I had 1994.

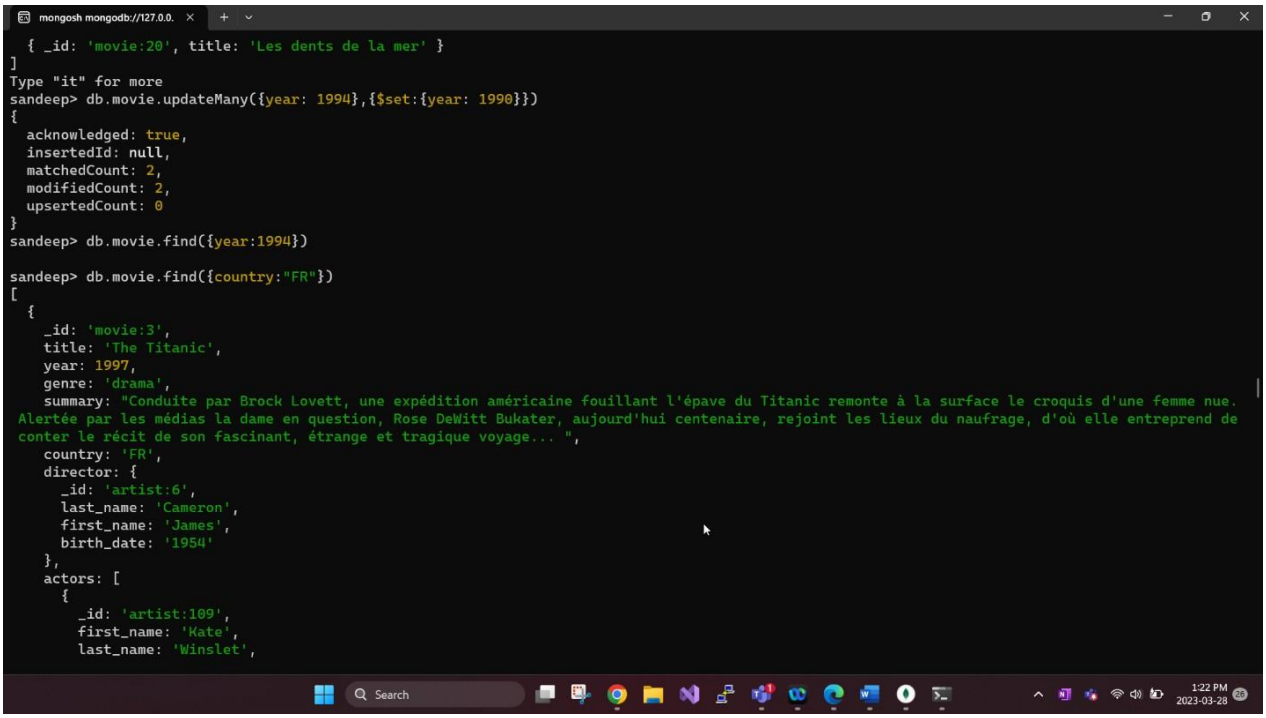


I had used update command with many because I had more than one and the command was **db.movie.updateMany({year: 1994},{$set:{year: 1990}})**

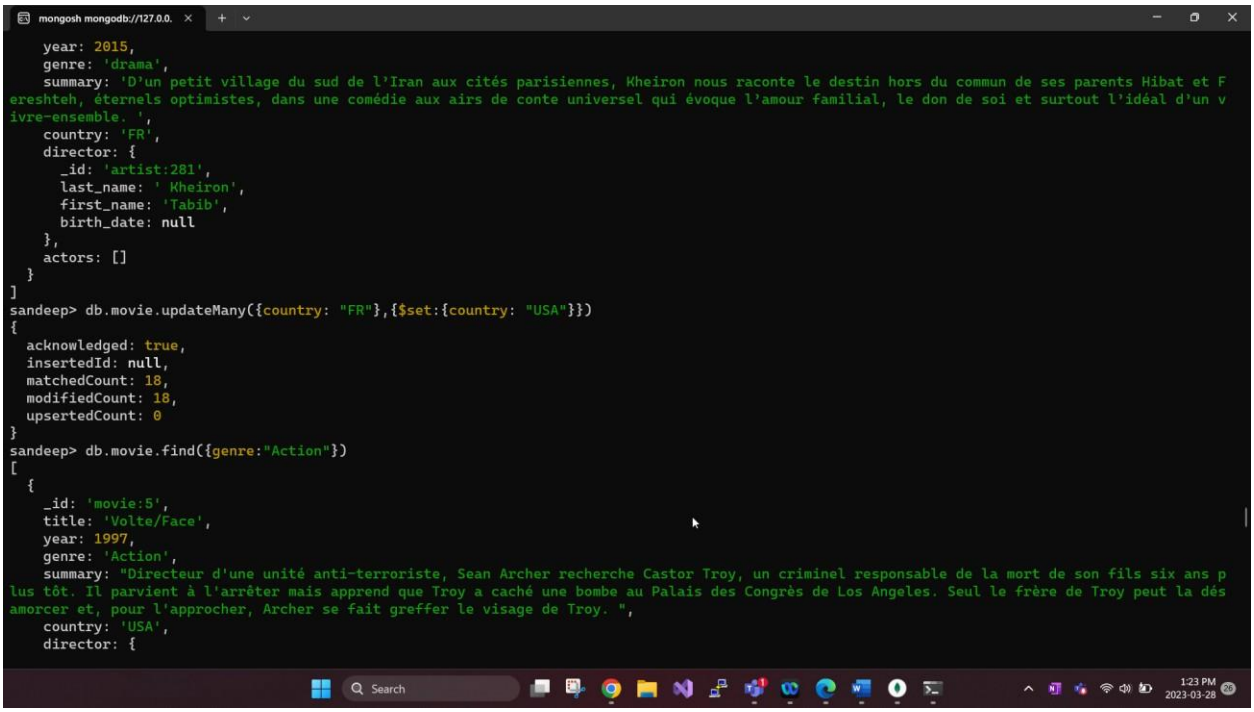


8 Update the movies with country “USA” from "FR".

I have had a movie script where I got FR country and I had to change to USA as I shown in figure where I had 18 id where I had FR.

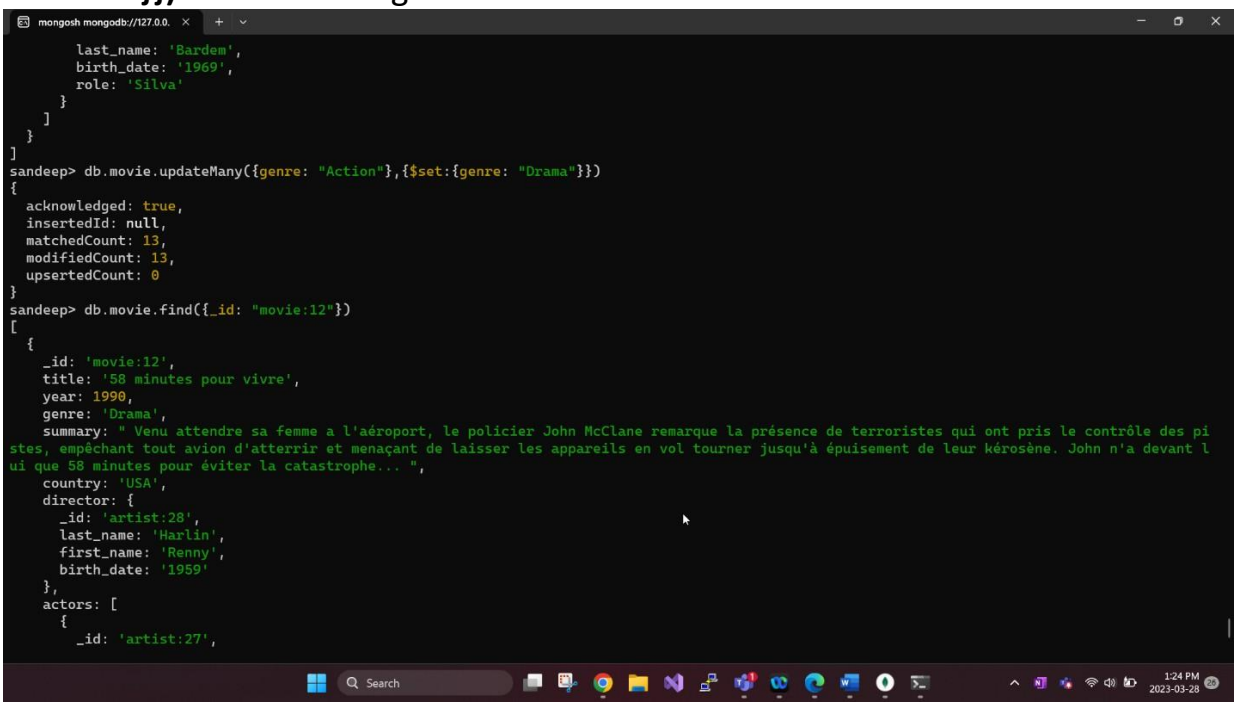


I had used update command with many because I had more than one and the command was **db.movie.updateMany({country: “FR”}, {$set: {country: “USA”}})**



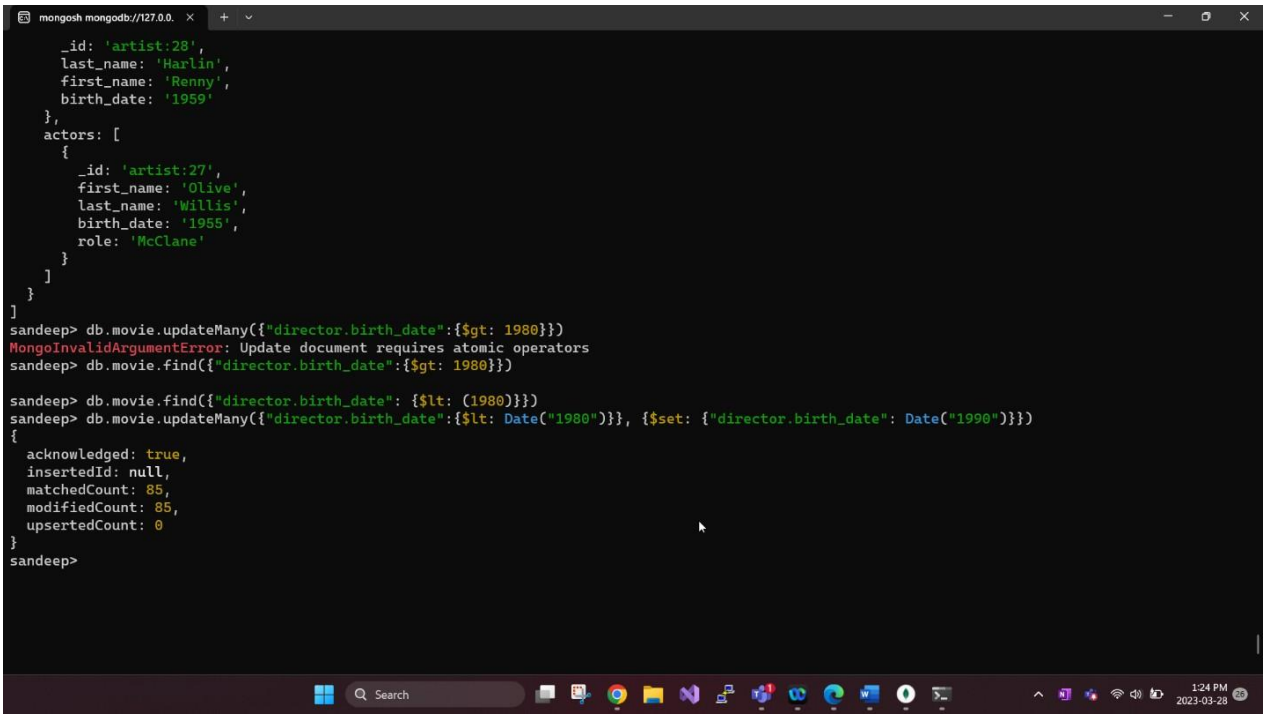
9 Update the movies with genre “Drama” from “Action”.

I have had a movie script where I got Action genre and I had to change to Drama as I shown in figure where I had 13 id where I had Action.
I had used update command with many because I had more than one and the command was **db.movie.updateMany({genre: “Action”}, {$set: {genre: “Drama”}})** as shown in figure.



10 Update the movies with director’s birth_date to 1990 where birth date older than 1980.

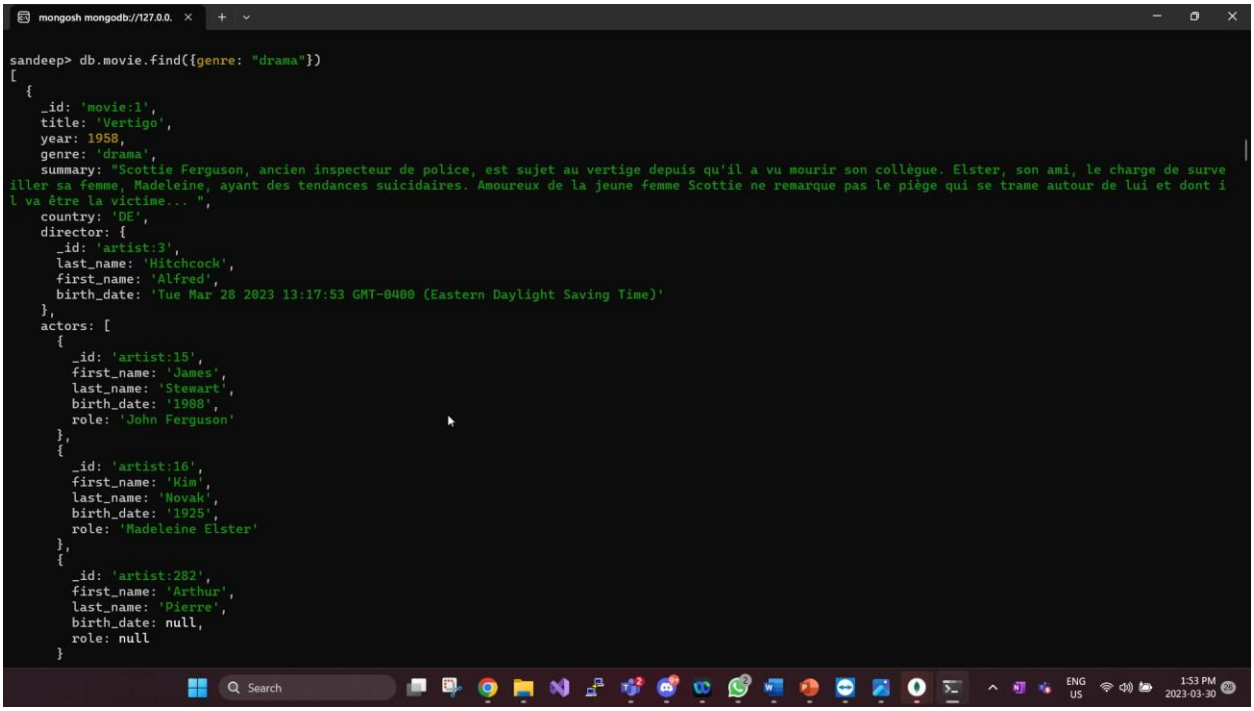
I had directors birth_date with 1980 which I had to change to 1990 there was 85 ids where I got 1990 as I shown in figure while using **db.movie.updateMany({“director.birth_date”: {$lt: Date(“190”)}}, {$set: {“directors.birth_date”: Date(“1990”)}})**

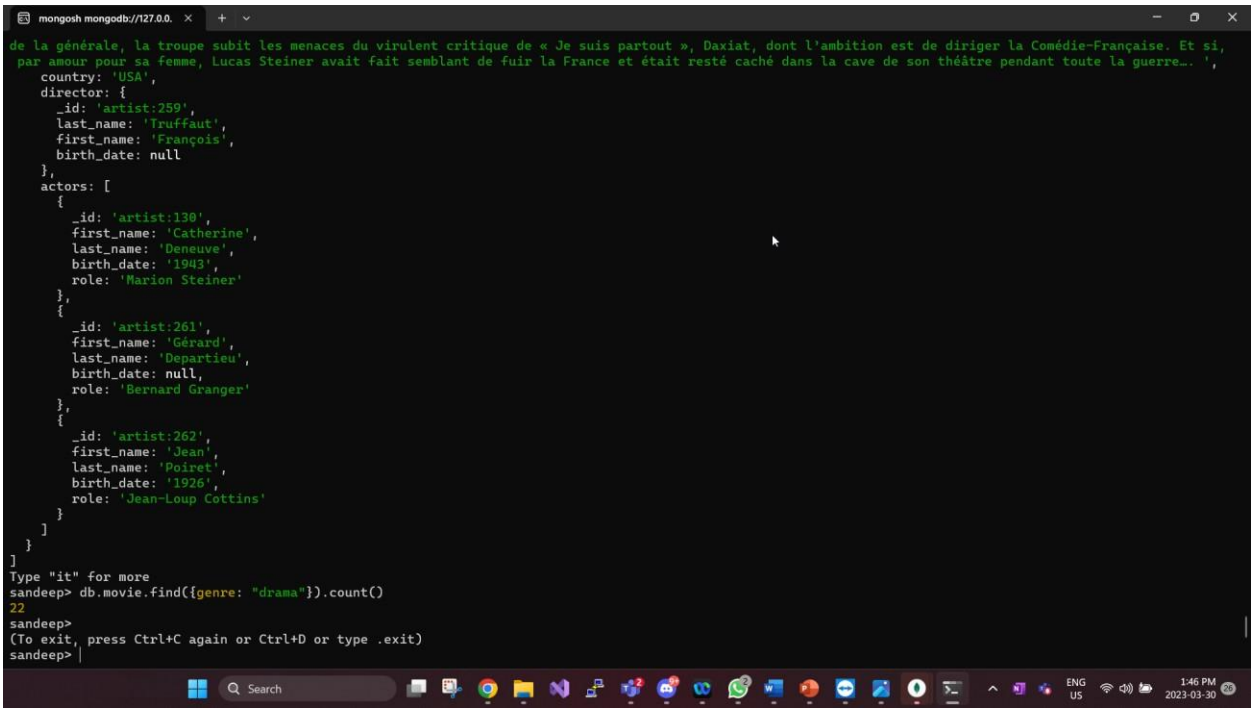


C. Write the script to Search a string in the document on movie collection.

1. **Fetch all the movies genre as “drama”.**

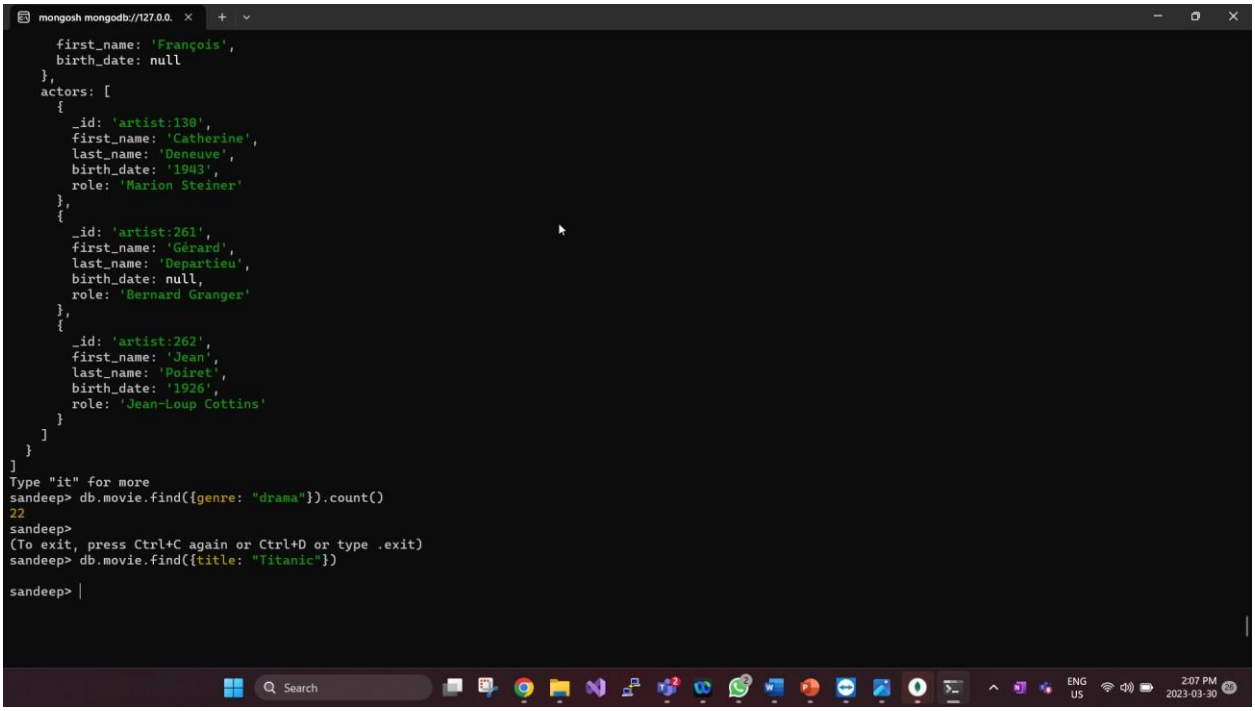
   I had to fetch drama genre from all the movies and for that I used command like **db.movie.find({genre: “drama”}).** There are 22 movie where I had genre with drama as shown in two figure.





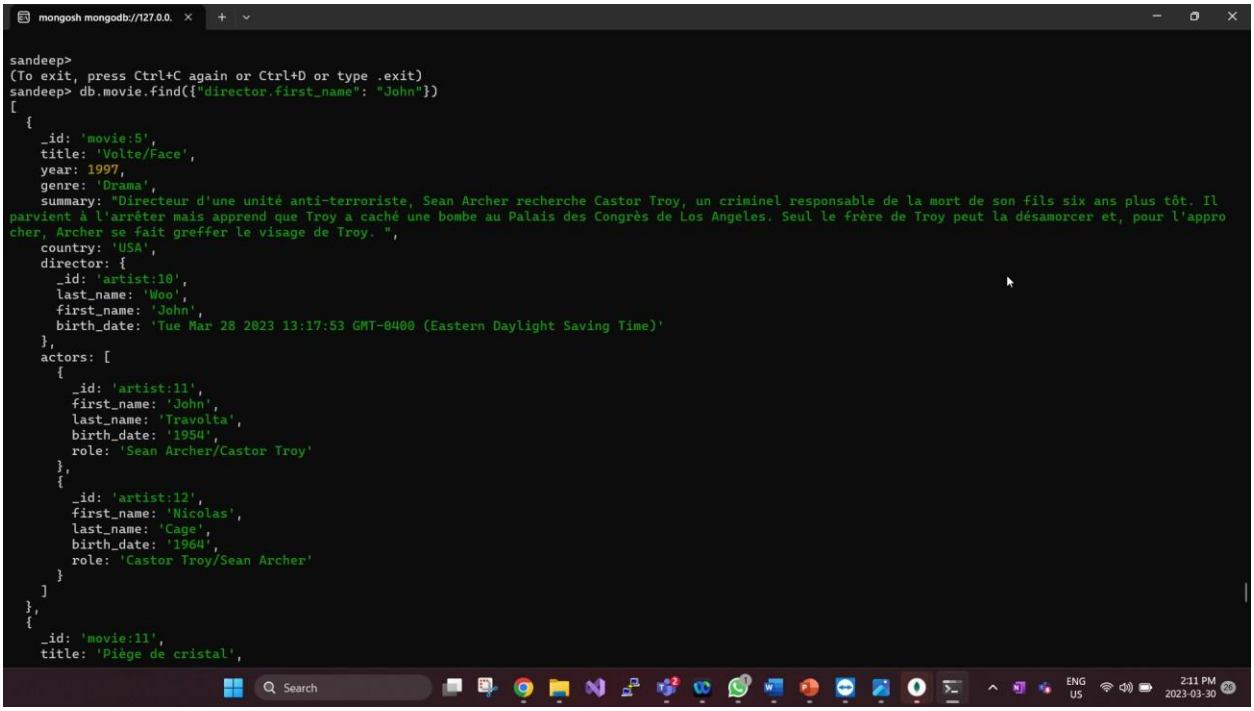
2. **Fetch all the movies title with “Titanic”.**

As in previous part I updated Titanic to The Titanic so it will not show anything there will no movie with Titanic title as shown in figure. The command is **db.movie.find({title: “Titanic”})**

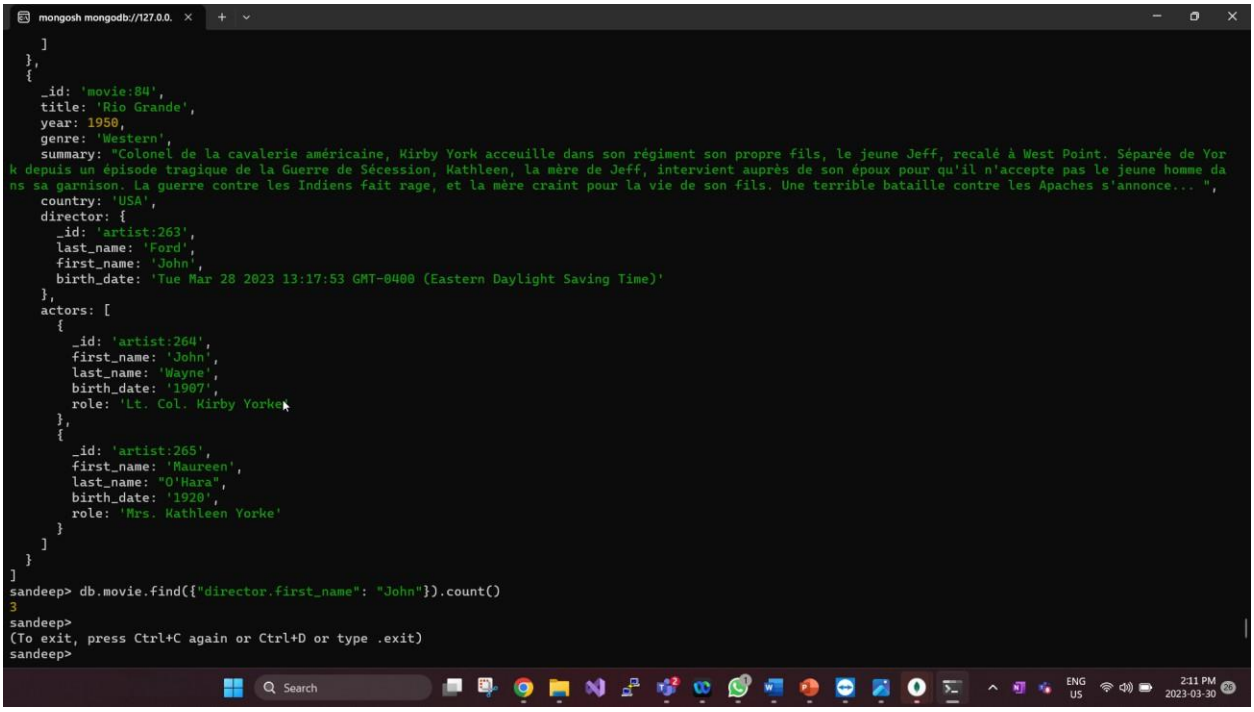


3 Fetch all the movies director’s first_name as “John”.

Well, I had to fetch John which is director name from all movies and it’s a first_name of director as I shown in figure the command is **db.movie.find({“director.first_name”: “john”})**



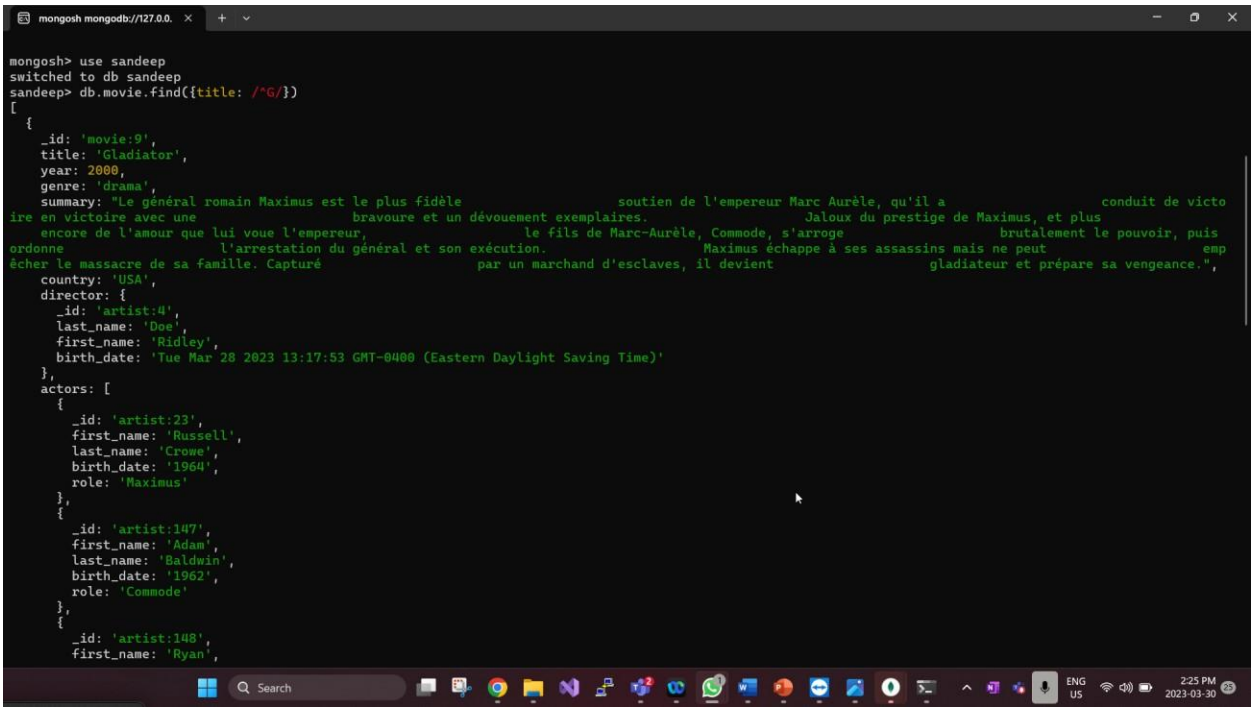
As I shown in figure there was 3 movies where I found John which is the directors first_name.

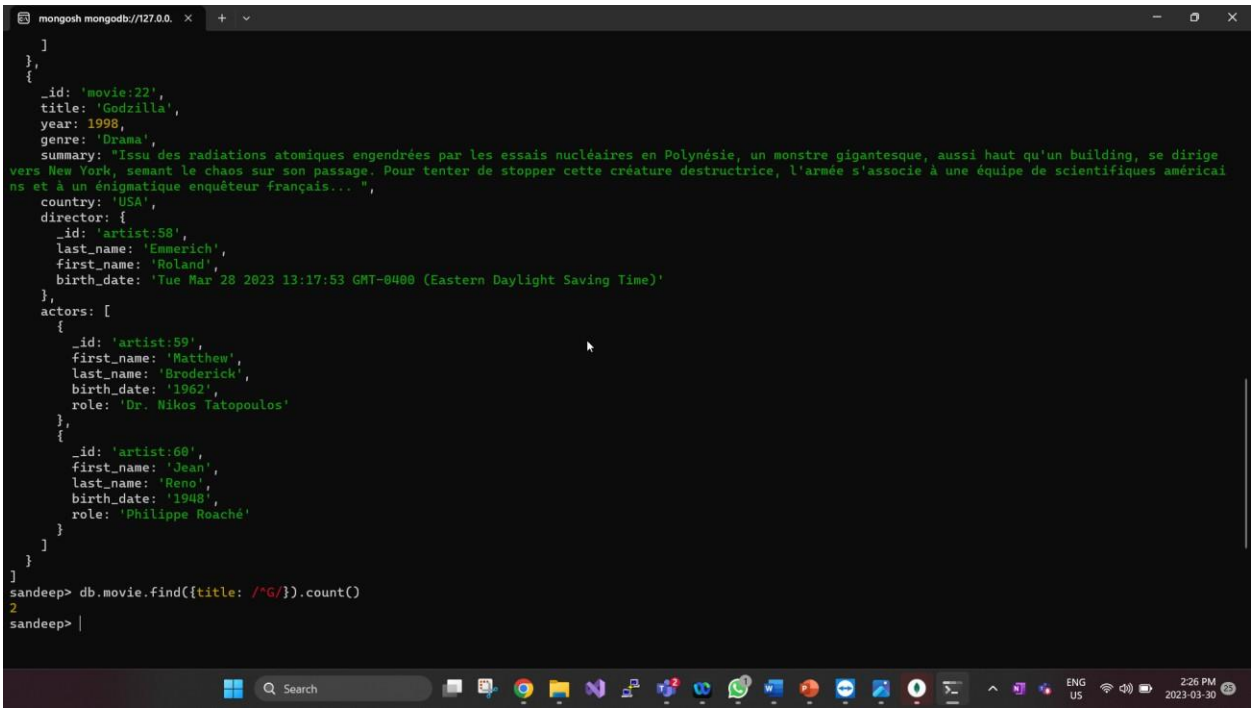


4 Fetch all the movies title start with "G".

To fetch all the documents from a MongoDB collection called "movie" that have movie titles starting with "G", we can use the $regex operator with a regular expression pattern that matches the desired title format.

db.movie.find({title: /^G/})

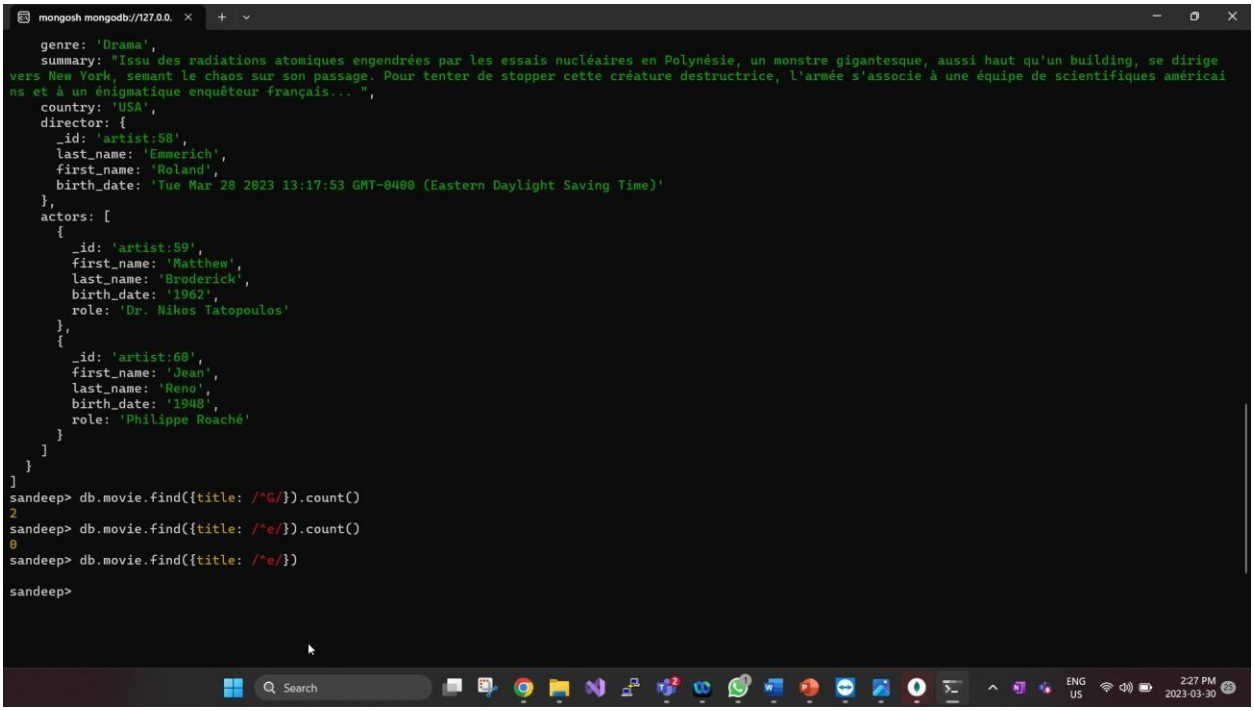




5 Fetch all the movies title end with "e".

To fetch all the documents from a MongoDB collection called "movie" that have movie titles ending with "e", we can use the $regex operator with a regular expression pattern that matches the desired title format.

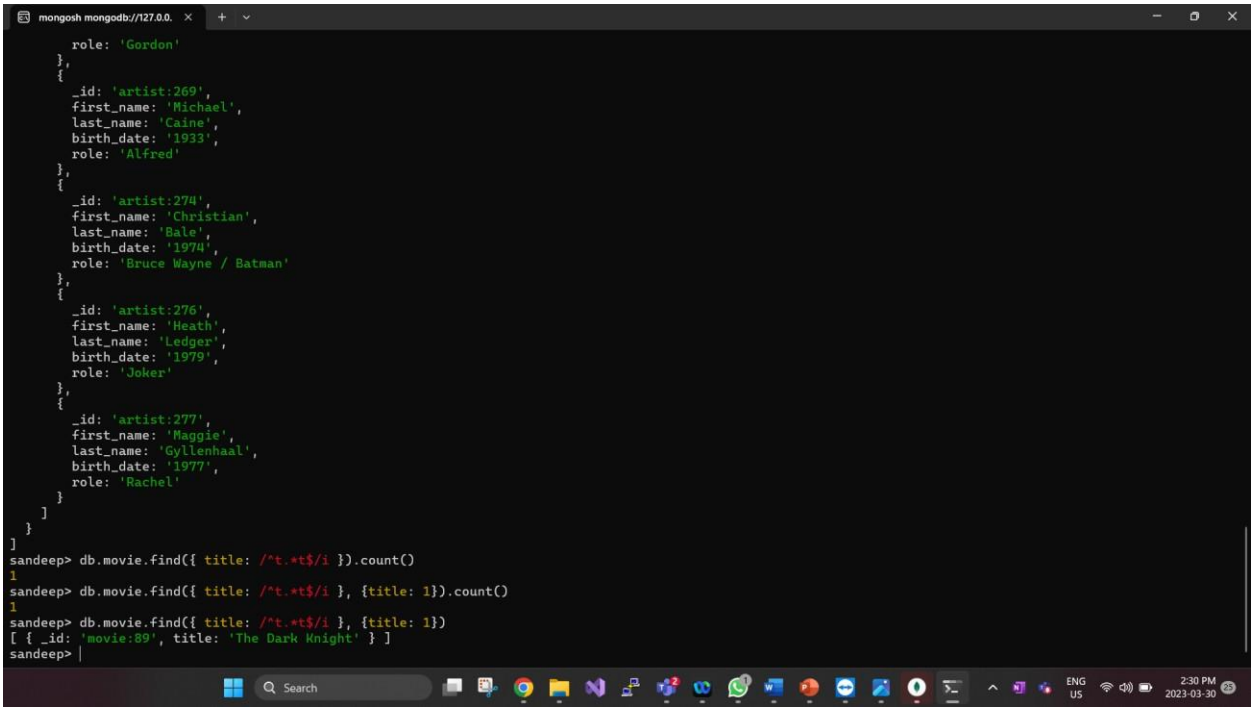
db.movie.find({title: /^e/})



6 Fetch all the movies title start and end with "t".

To fetch all the documents from a MongoDB collection called "movie" that have movie titles starting and ending with "t", you can use the $regex operator with a regular expression pattern that matches the desired title format.

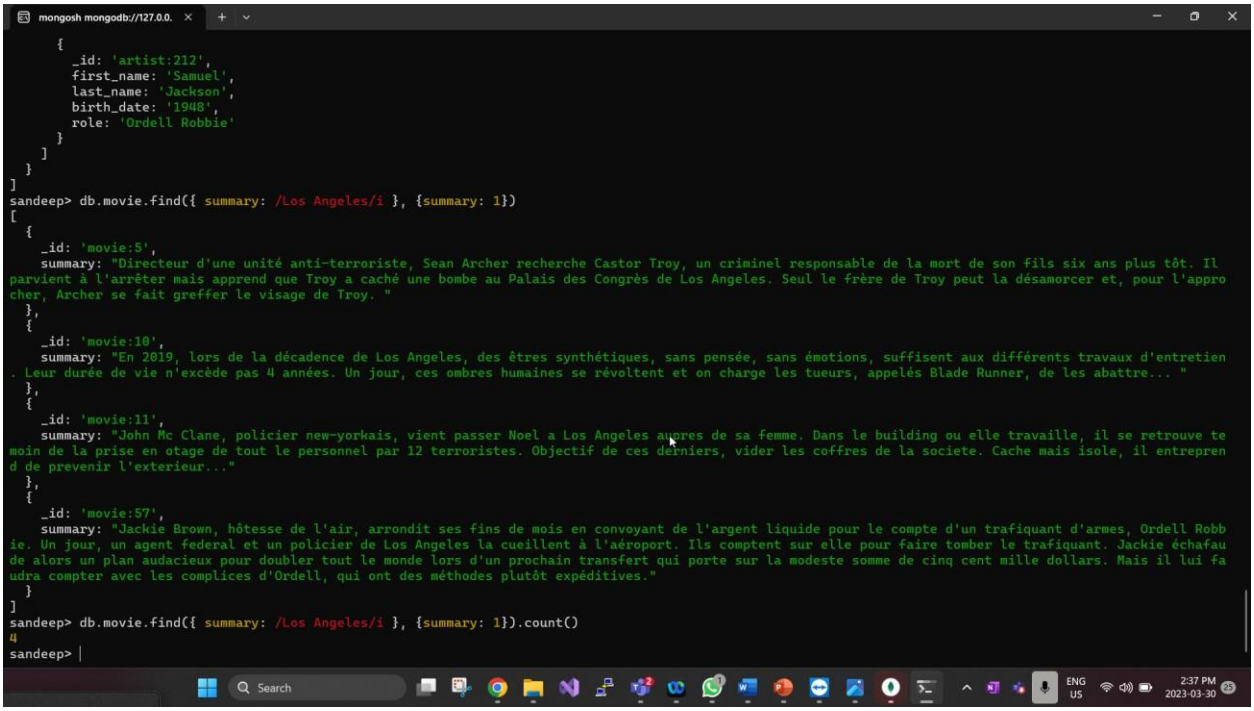
db.movie.find({ title: /^t.*t$/i }, {title: 1})



7 find all movies that have a summary that contains the word "Los Angeles".

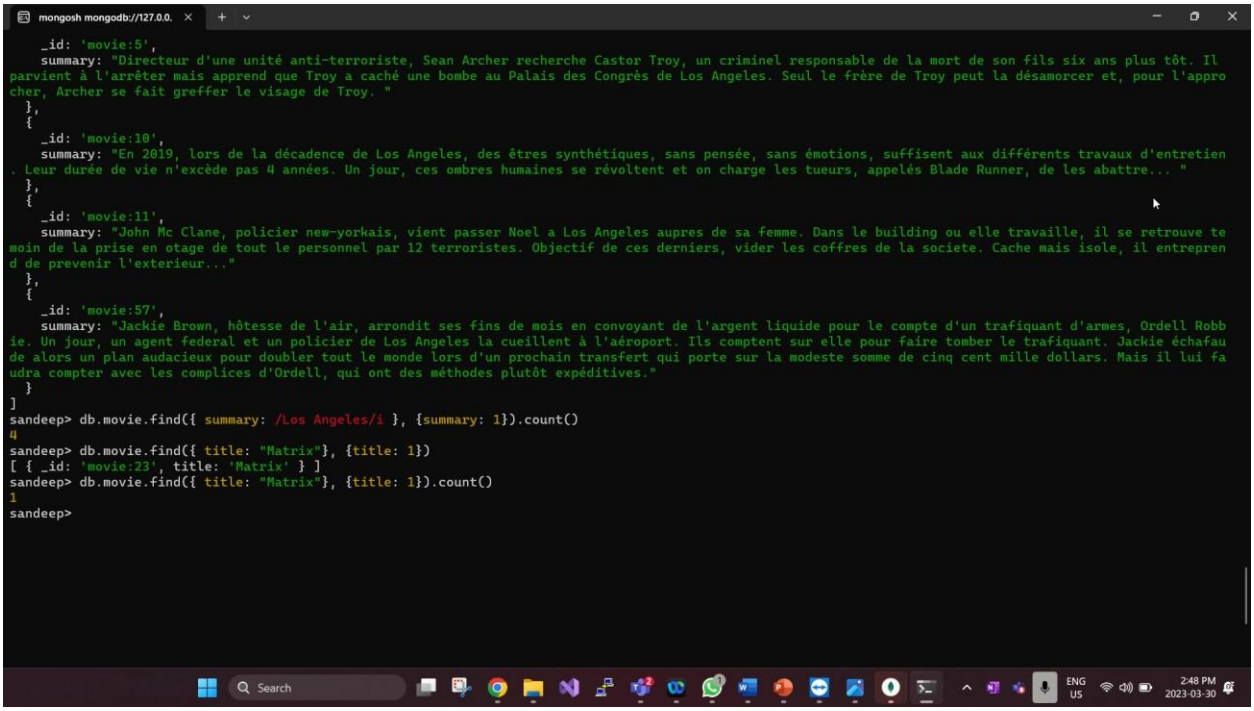
To fetch all the documents from a MongoDB collection called "movie" that have a summary containing the word "Los Angeles", we can use the $regex operator with a regular expression pattern that matches the desired summary format.

db.movie.find({ summary: /Los Angeles/i }, {summary: 1})



8 find all movies that have a title that contains the word "Matrix".

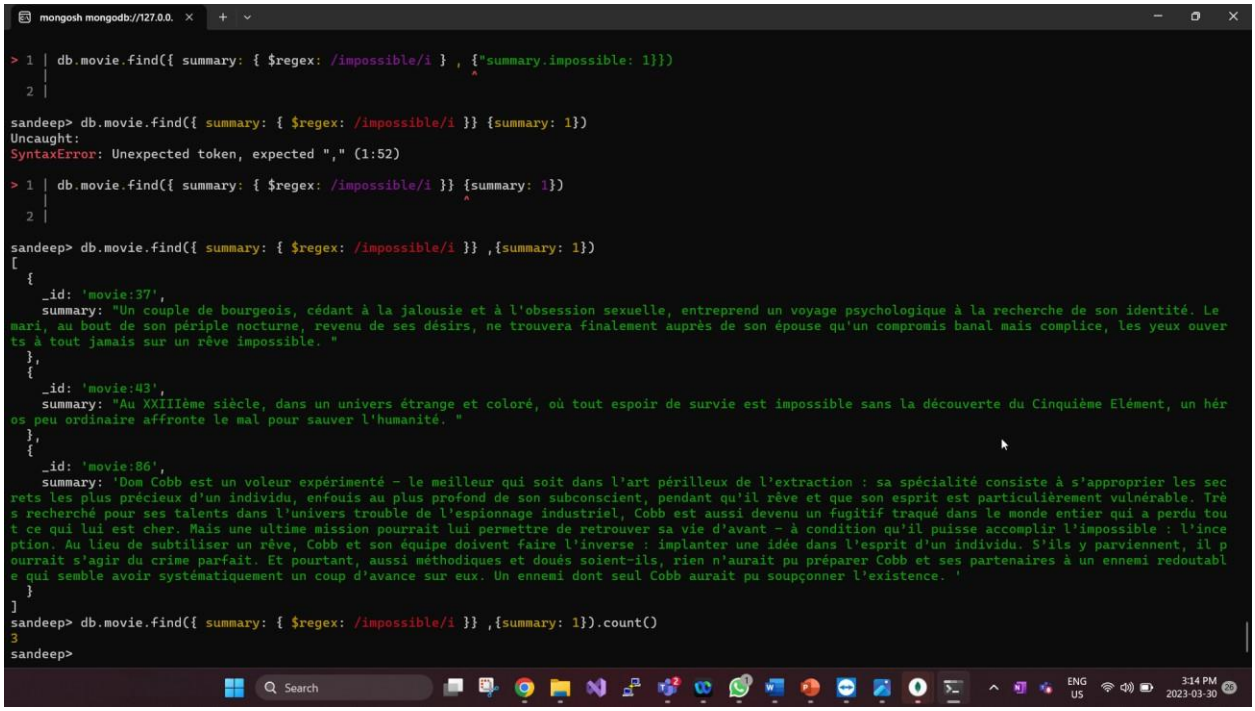
To fetch all the documents from a MongoDB collection called "movie" that have a title containing the word "Matrix", you can use the $regex operator with a regular expression pattern that matches the desired title format. **db.movie.find({ title: "Matrix"}, {title: 1})**



9 find all movies that have a summary that contains the word "impossible".

To find all movies that have a summary that contains the word "impossible" in MongoDB, you can use the $regex operator to perform a regular expression search.

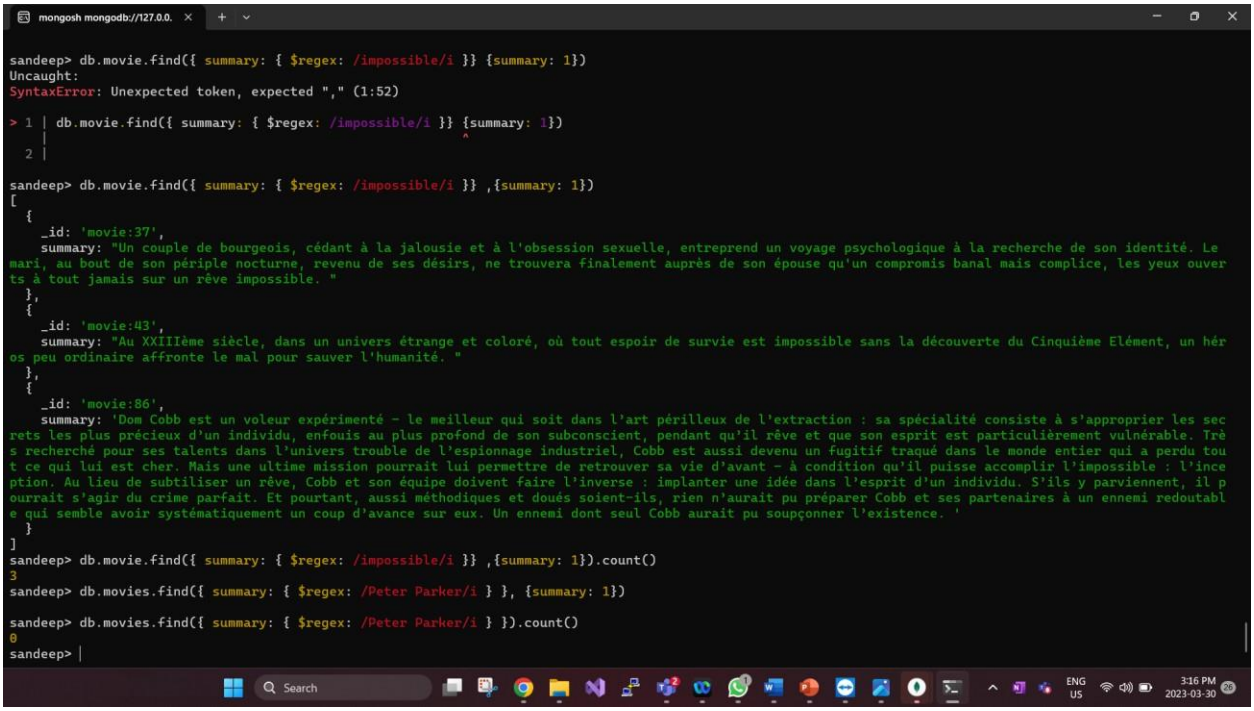
db.movie.find({ summary: { $regex: /impossible/i }} ,{summary: 1})



10 find all movies that have a summary that contains the word "Peter Parker".

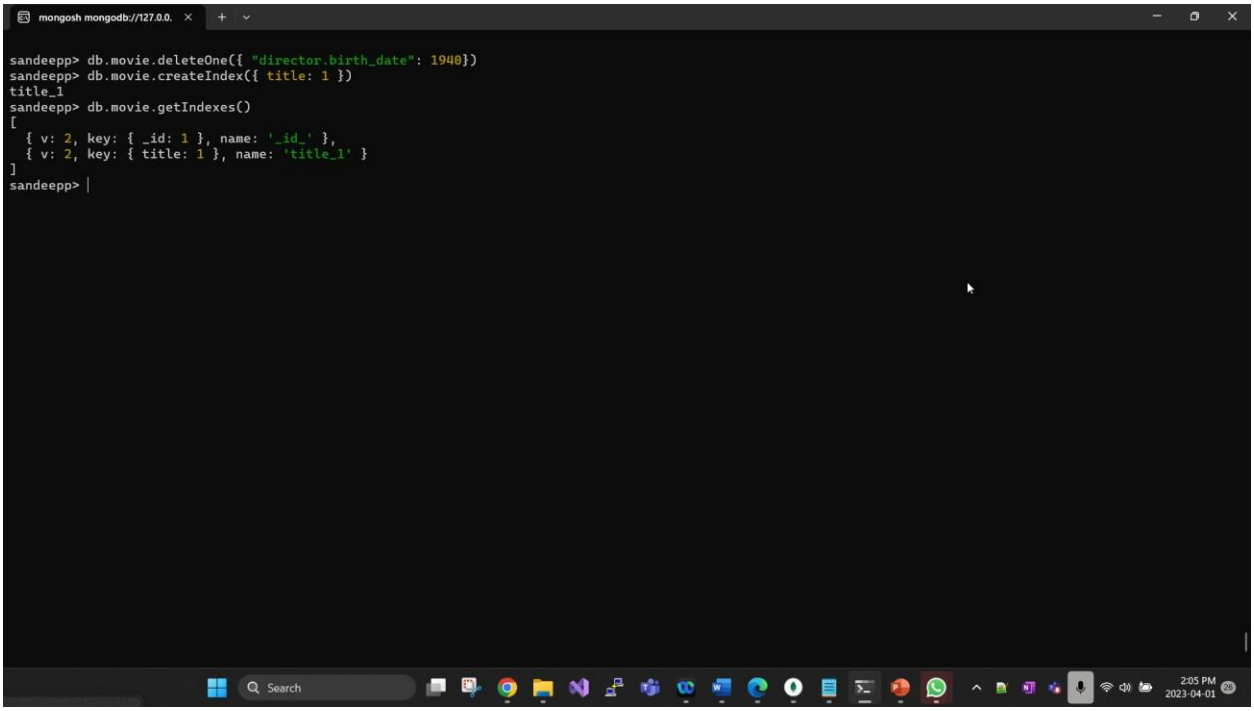
To find all movies that have a summary that contains the phrase "Peter Parker" in MongoDB, you can use the $regex operator to perform a regular expression search.

db.movies.find({ summary: { $regex: /Peter Parker/i } }, {summary: 1})



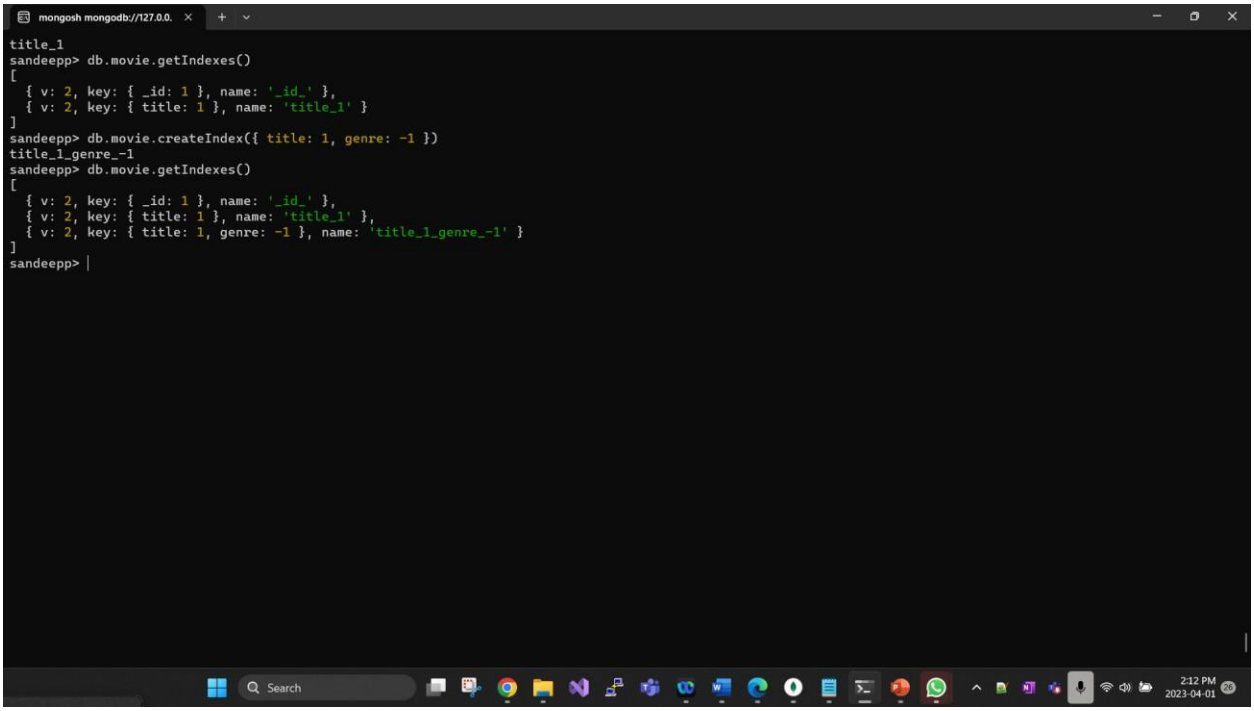
D Write the script to create the indexes on movie collection.

1. **Create an index on title attribute in ascending order on movie collection.**
   I used this command **db.movie.createIndex({ title: 1 }).** This command creates a new index on the title attribute of the movie collection with an ascending order. The value 1 specifies that the index should be created in ascending order. db.movie.getIndexes()



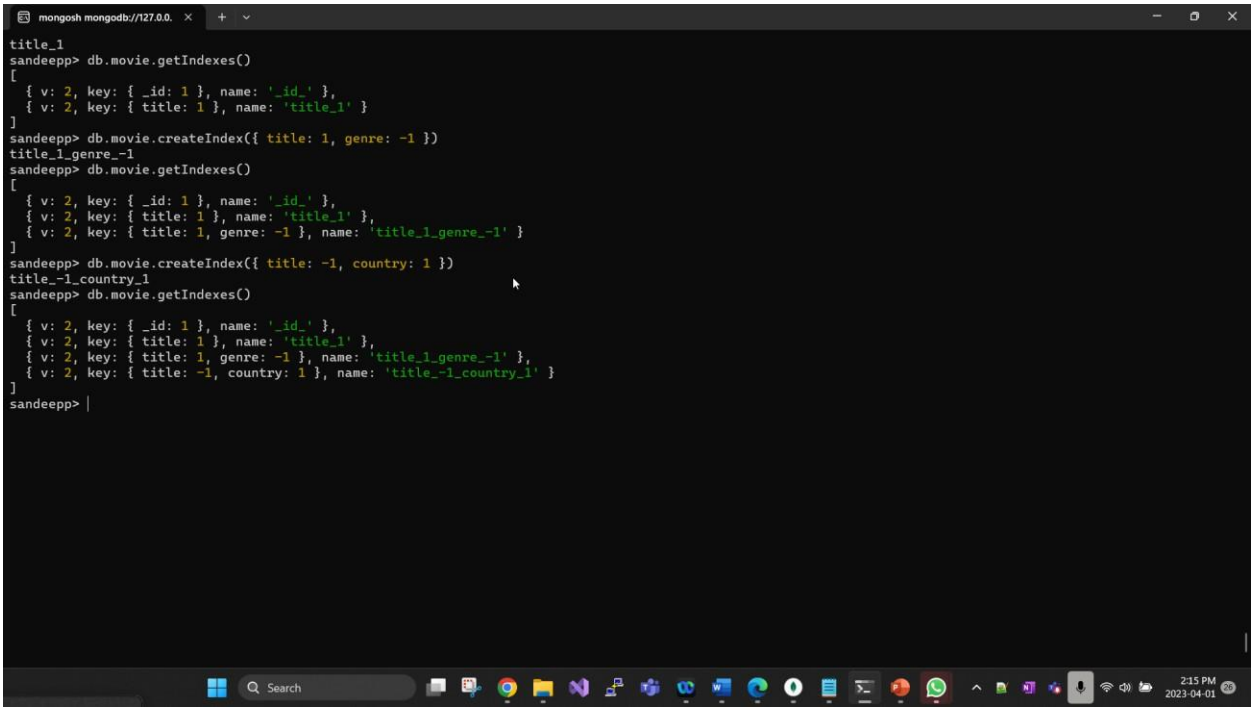
2 **Create a single index with multiple attributes (title by ascending order and genre by descending order) on movie.**

I used this command. **db.movie.createIndex({ title: 1, genre: -1 }).** This command creates a new index on the title and genre attributes of the movie collection with title in ascending order and genre in descending order. The value 1 specifies that the title index should be created in ascending order, and -1 specifies that the genre index should be created in descending order.



3 Create a single index with multiple attributes (title by descending order and country by ascending order) on movie.

Well I used this command **db.movie.createIndex({ title: -1, country: 1 }).** This command creates a new index on the title and country attributes of the movie collection with title in descending order and country in ascending order. The value -1 specifies that the title index should be created in descending order, and 1 specifies that the country index should be created in ascending order.db.movie.getIndexes()



4. Create multiple indexes based on the multiple attributes including composite index.

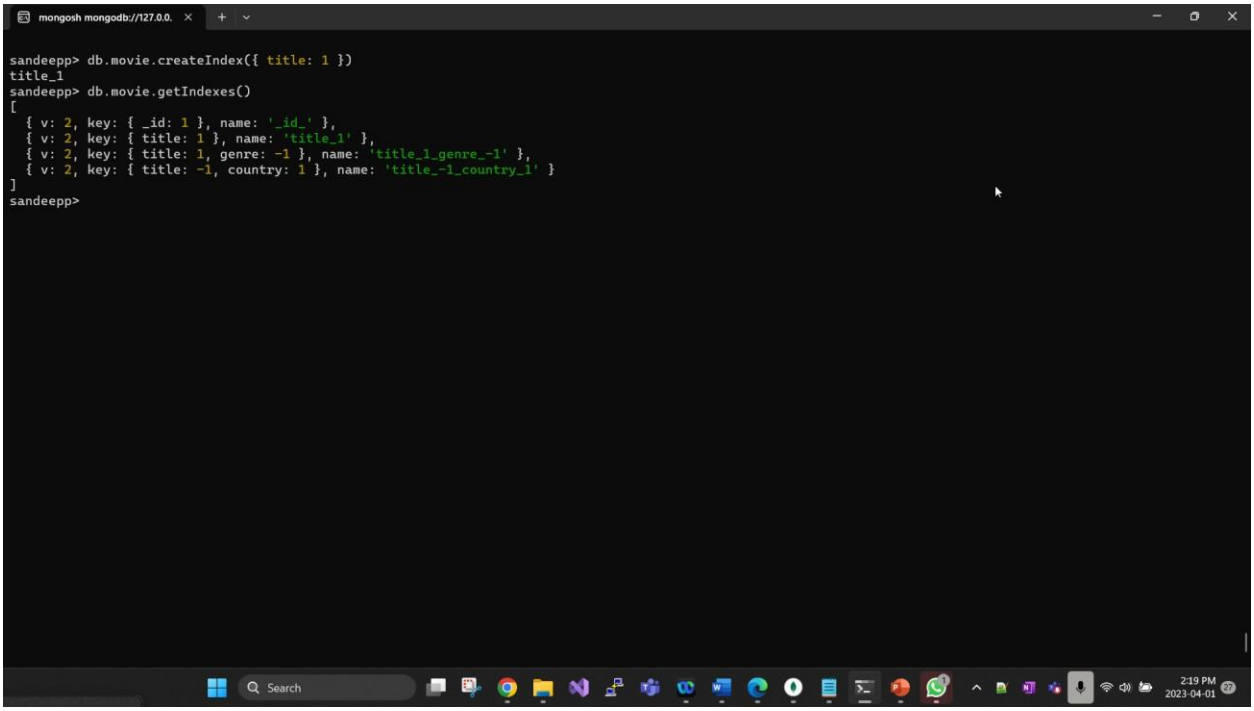
title: single index by ascending order

title and genre: composite index by descending order

year: single index by descending order

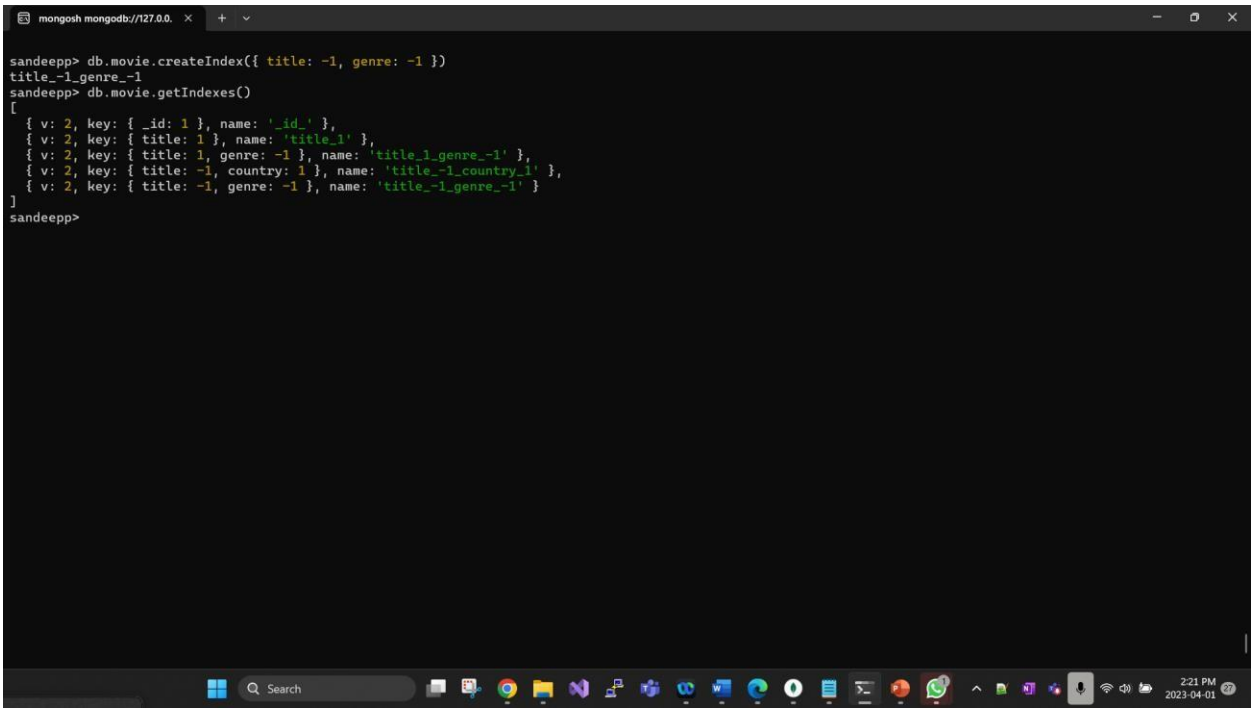
To create a single index on the title field in ascending order:

db.movie.createIndex({ title: 1 })



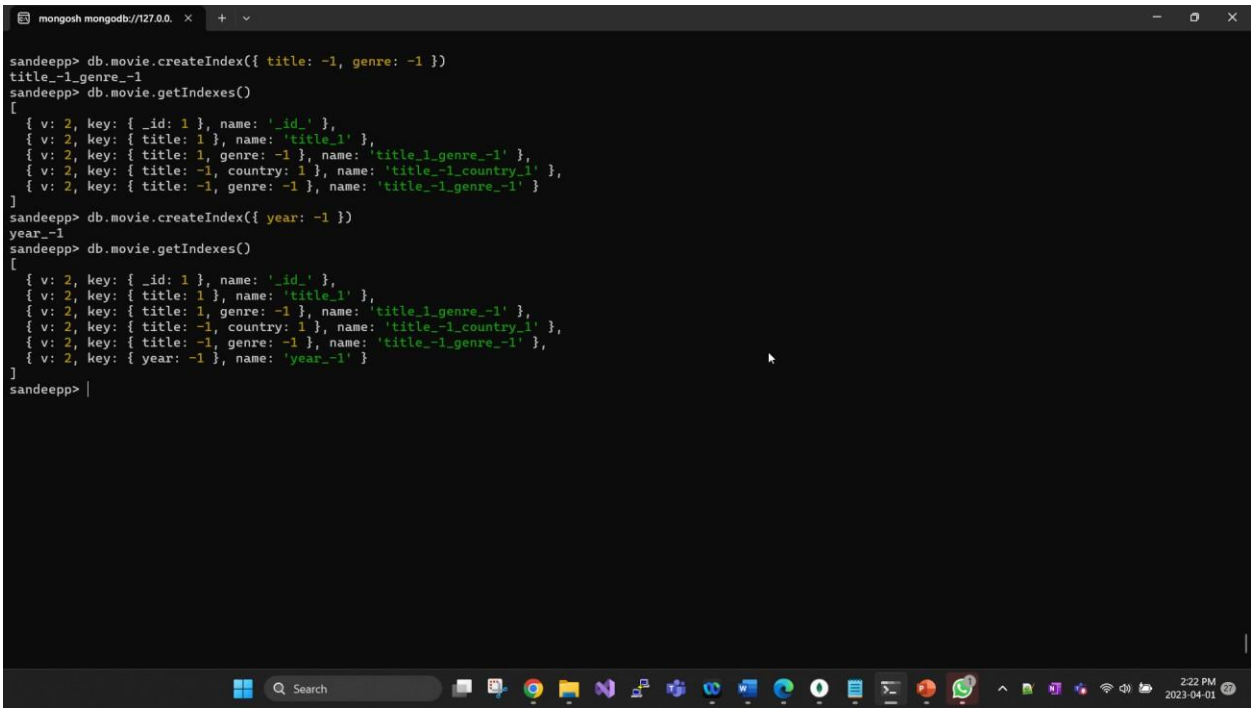
To create a composite index on the title and genre fields in descending order:

db.movie.createIndex({ title: -1, genre: -1 })



To create a single index on the year field in descending order:

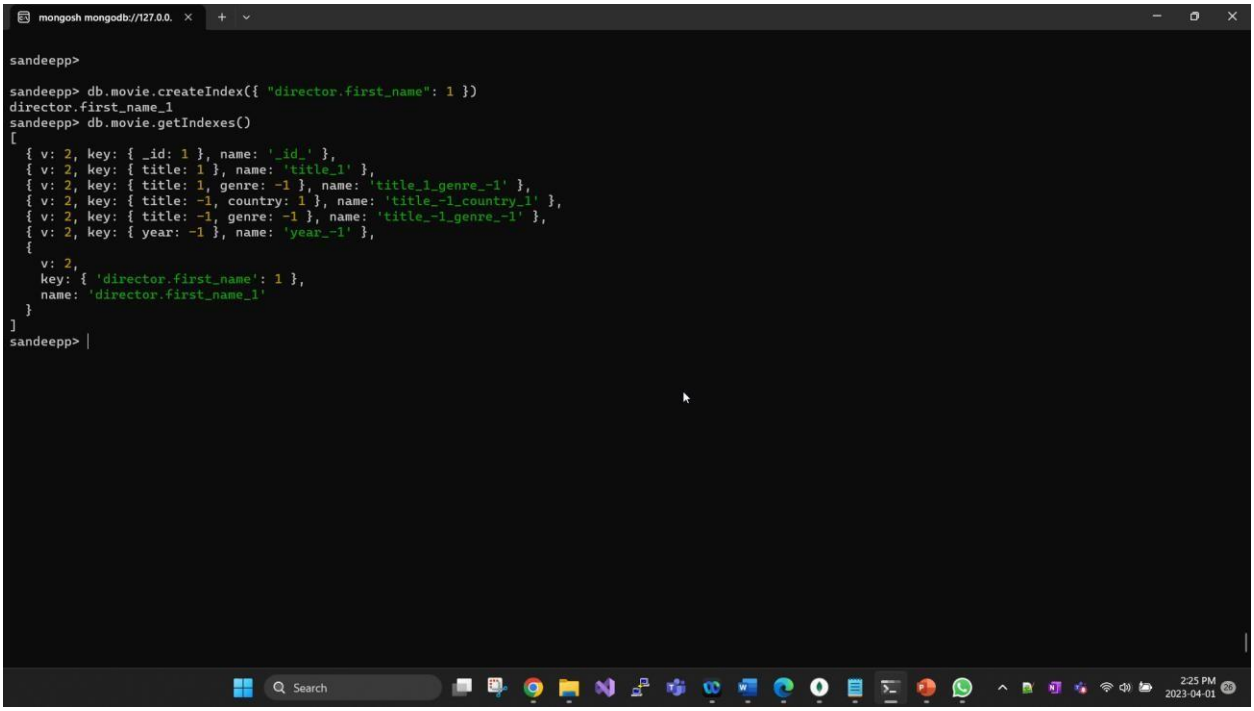
db.movie.createIndex({ year: -1 })



5 **Create a wildcard Index on director first_name order by ascending order on movie collection.**

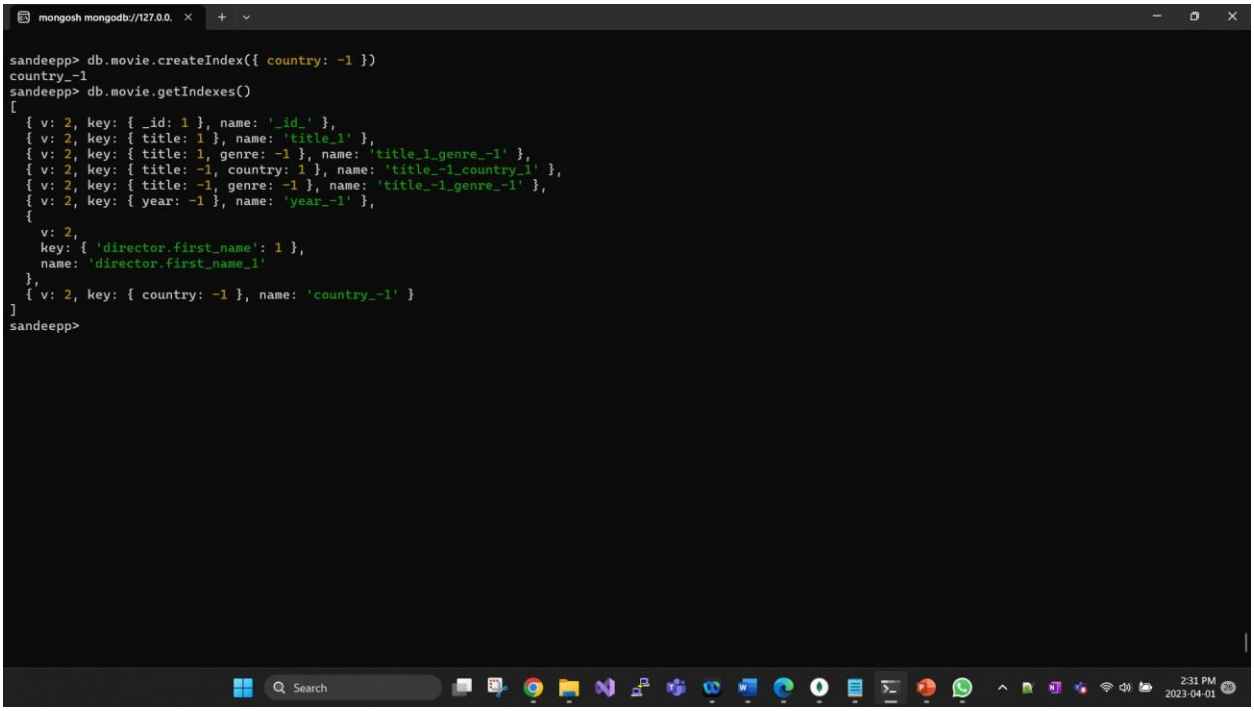
This command creates a new wildcard index on the director.first_name field of the movie collection, which indexes all of the words in the director.first_name field in ascending order. This index can be used to search for any words or phrases in the director.first_name field using the $text operator.

db.movie.createIndex({ "director.first_name": 1 })



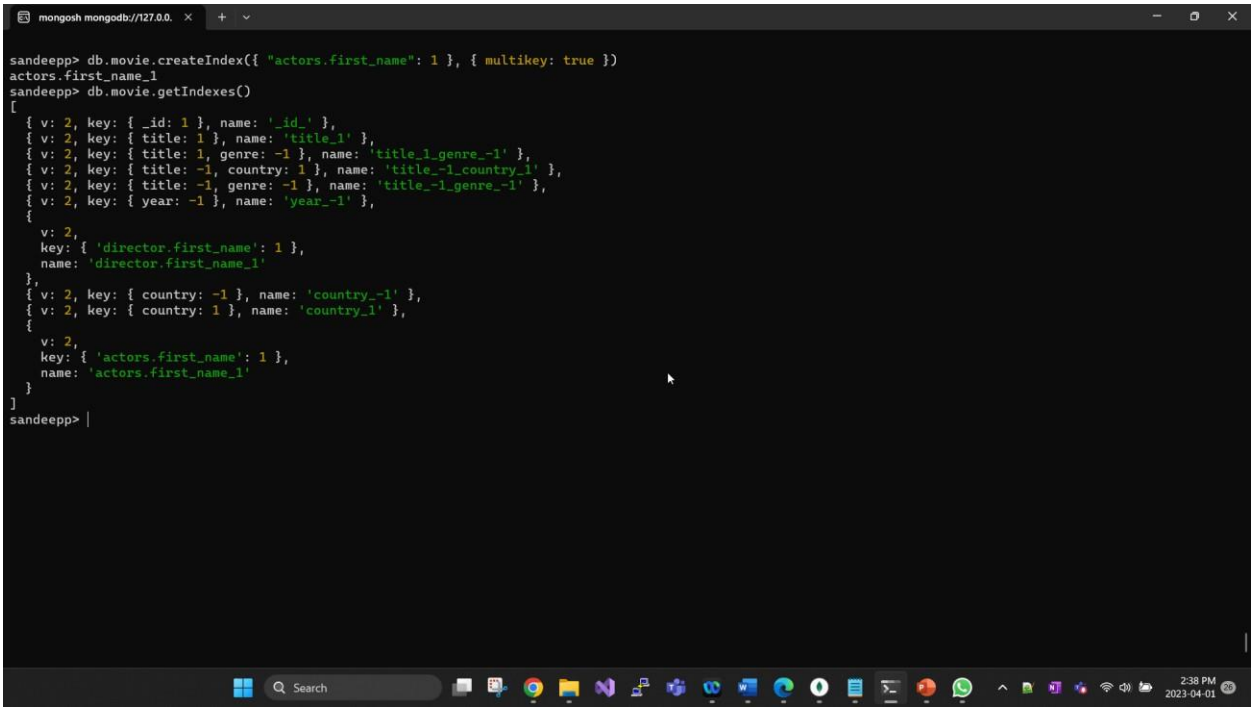
6 **Create a unique index on country attribute by ascending order on movie collection.**

db.movie.createIndex({ country: 1 }). This command creates a new unique index on the country attribute of the movie collection, which indexes the country attribute in ascending order and enforces uniqueness on the values of the country attribute. This ensures that there can be no two documents in the movie collection with the same country value.



7 Create a multikey Index on actors first_name order by ascending order on movie collection.

db.movie.createIndex({ "actors.first_name": 1 }, { multikey: true }). This command creates a new multikey index on the actors.first_name field of the movie collection, which indexes all of the values of the actors.first_name array in ascending order. This index can be used to search for any values in the actors.first_name array using the $in operator.

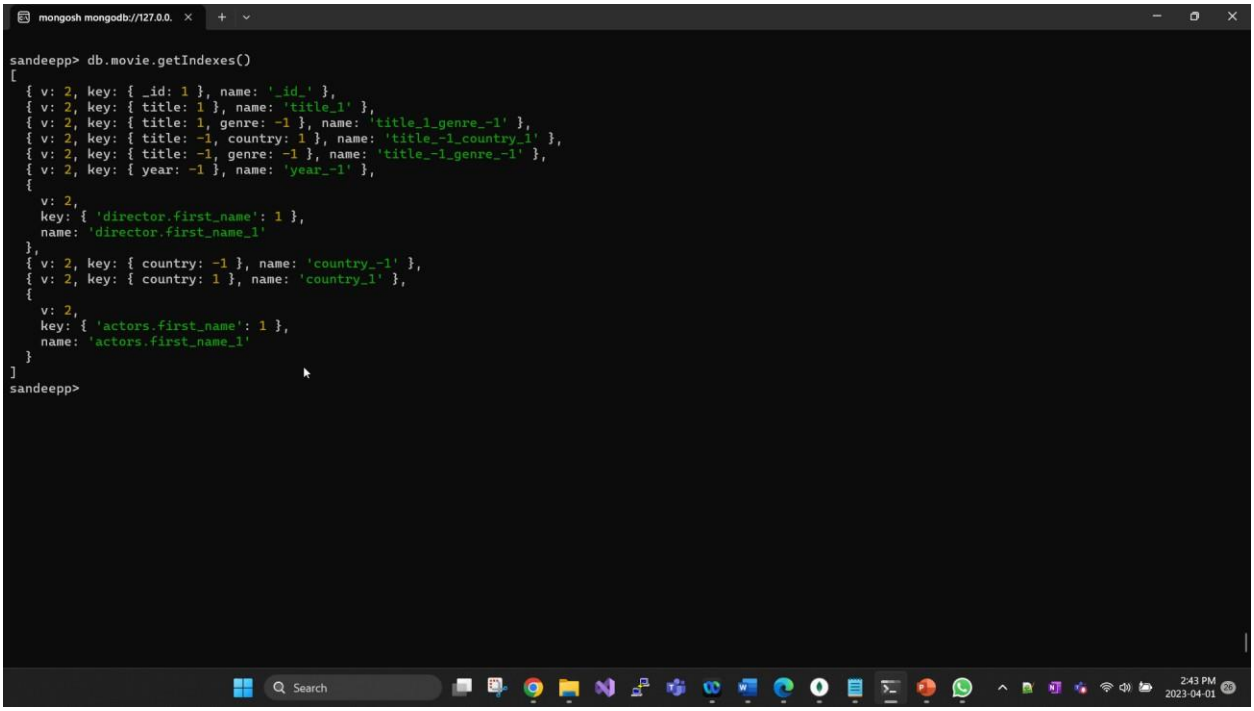


8 Get all the Indexes from movie collection.

I used **db.movie.getIndexes().** This command returns an array of documents, with each document describing an index on the movie collection. The documents include information such as the index name, the indexed fields, and any additional options set for the index.

db.movie.showIndexes()

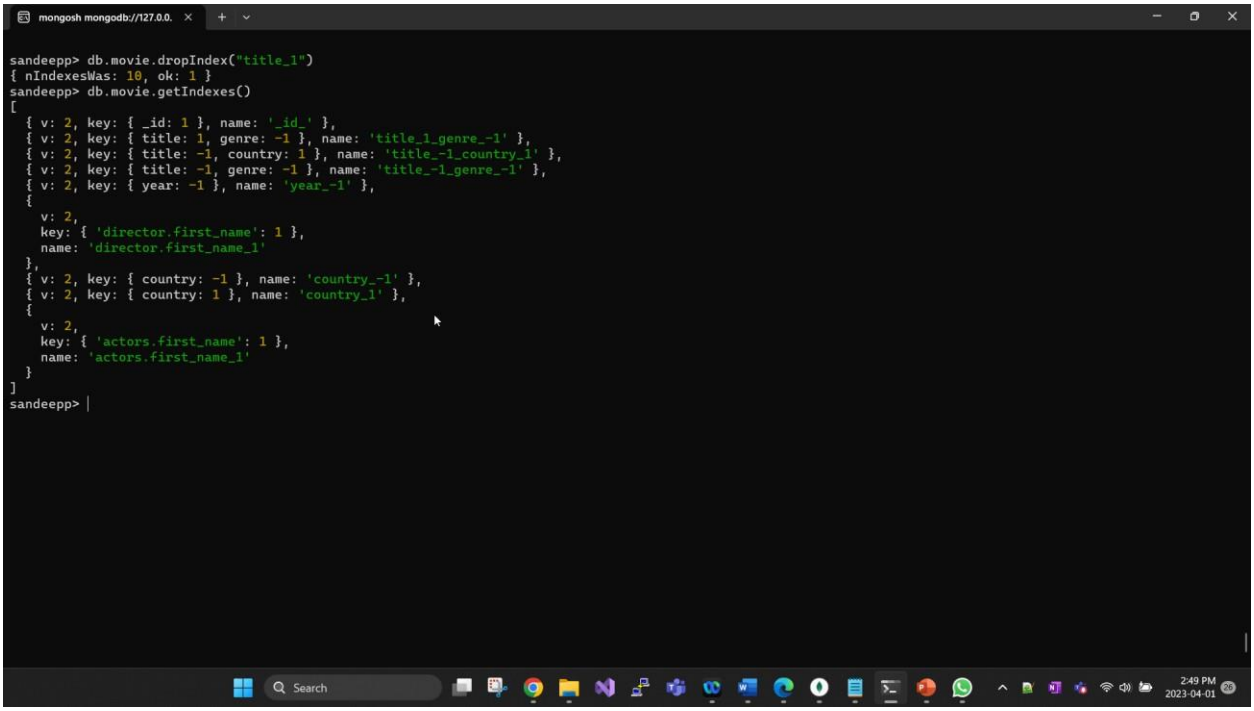
This command returns the same array of documents as getIndexes(), but with a more concise output format.



9 Remove “title” index from movie collection.

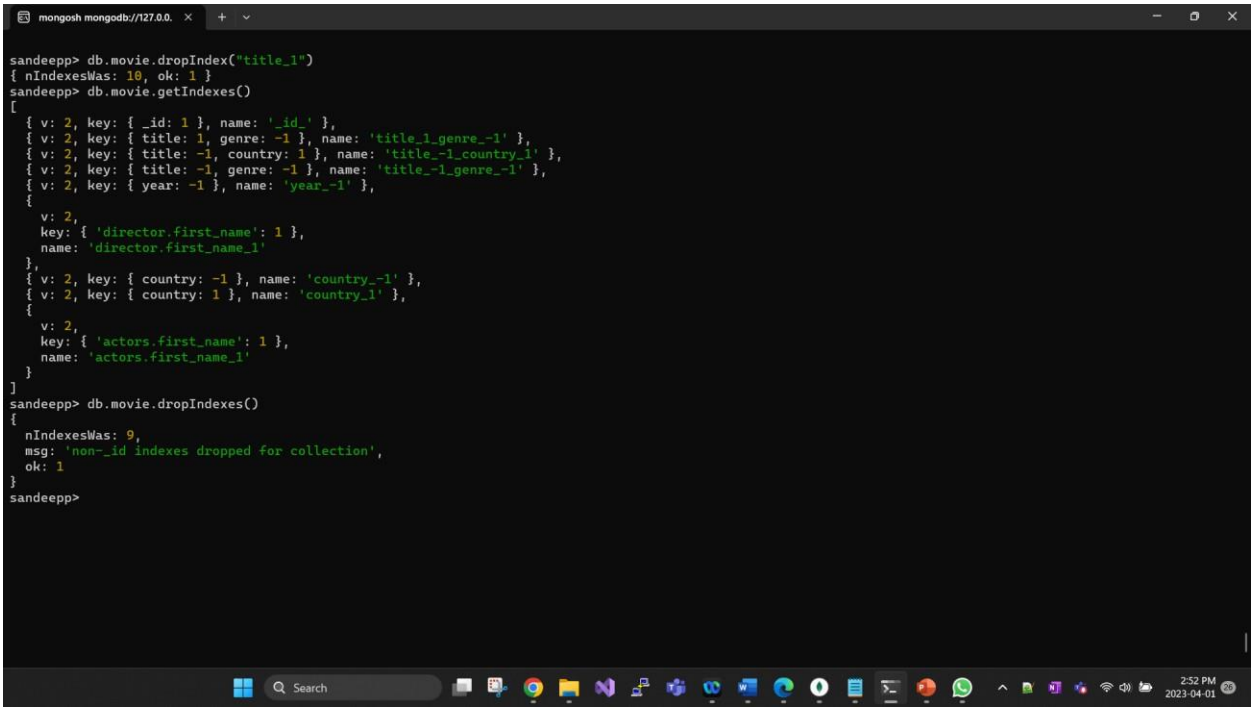
movie.dropIndex("title_1")

This command removes the index with the name "title_1" from the movie collection. The name of the index is a combination of the field name ("title") and the sort order (1 for ascending order). If the index was created with a different name, you will need to replace "title_1" with the correct name of the index.



10. Drop all the indexes from movie collection.

db.movie.dropIndexes(). This command removes all indexes from the movie collection, including the default index on the _id field. After executing this command, you will need to recreate any indexes that are necessary for your application to function properly.

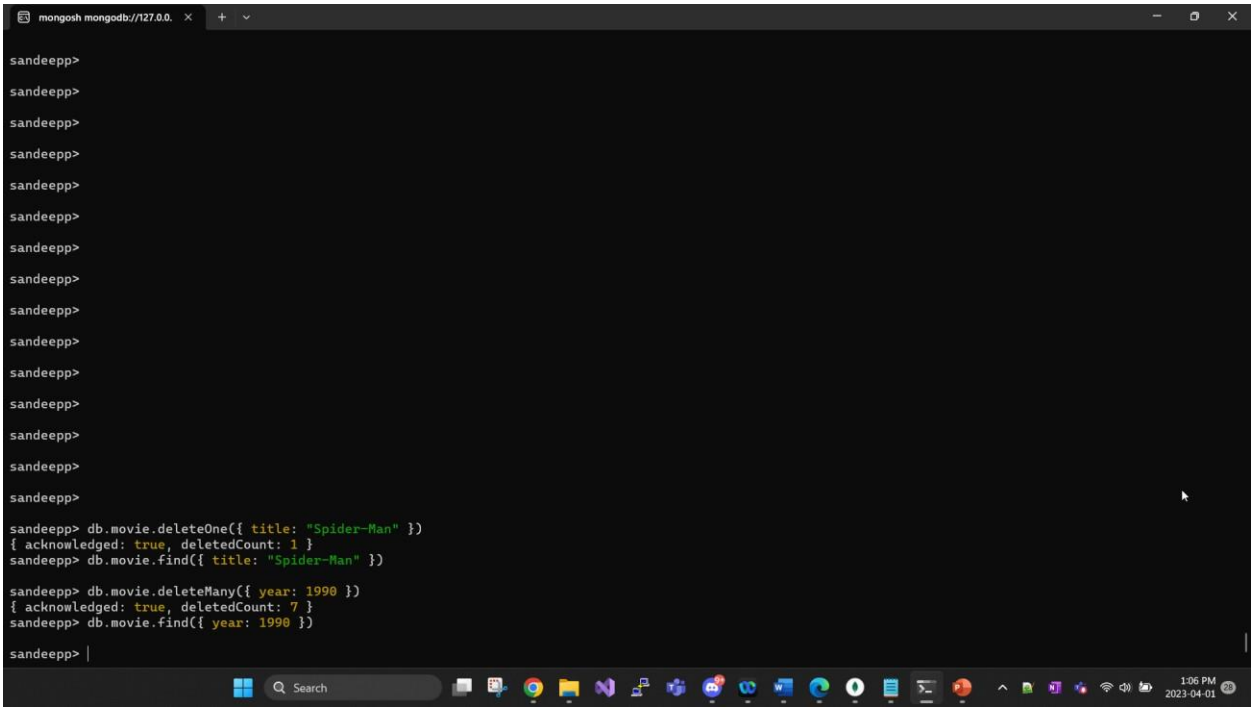


E. Write the script to delete document on movie collection.

1. Delete the movie with title "Spider-Man".

db.movie.deleteMany({ title: "Spider-Man" })

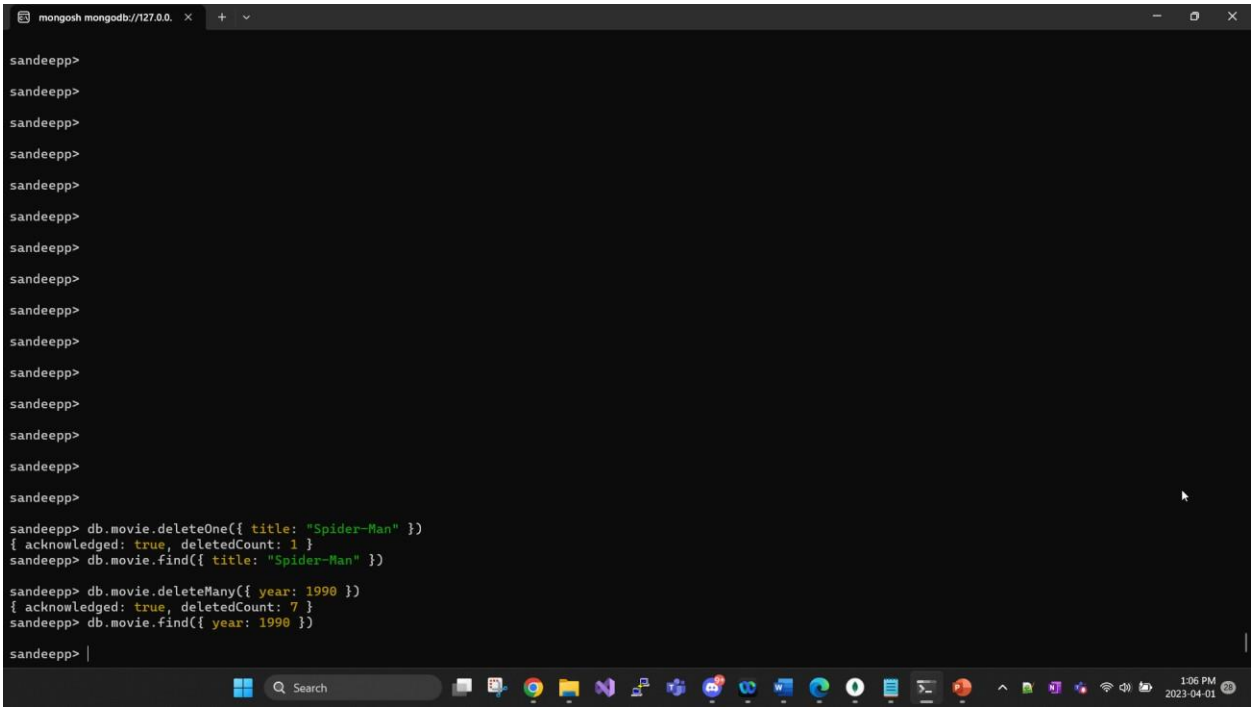
This command removes all documents that match the { title: "Spider-Man" } filter from the movie collection. Make sure to use the correct filter to ensure that you are deleting the correct documents.



2 **Delete the movies which is released in 1990.**

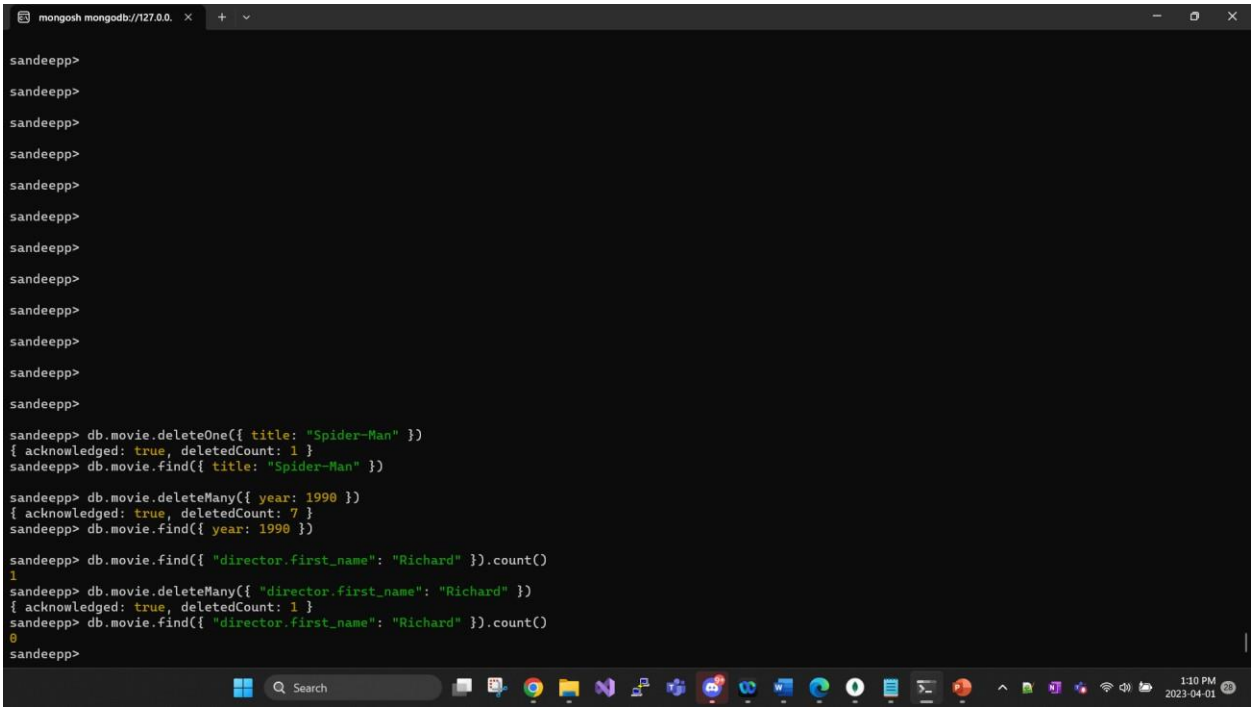
db.movie.deleteMany({ release_year: 1990 })

This command removes all documents that match the { release_year: 1990 } filter from the movie collection. Make sure to use the correct filter to ensure that you are deleting the correct documents.



3. **Delete the movies with director first_name is “Richard”.**

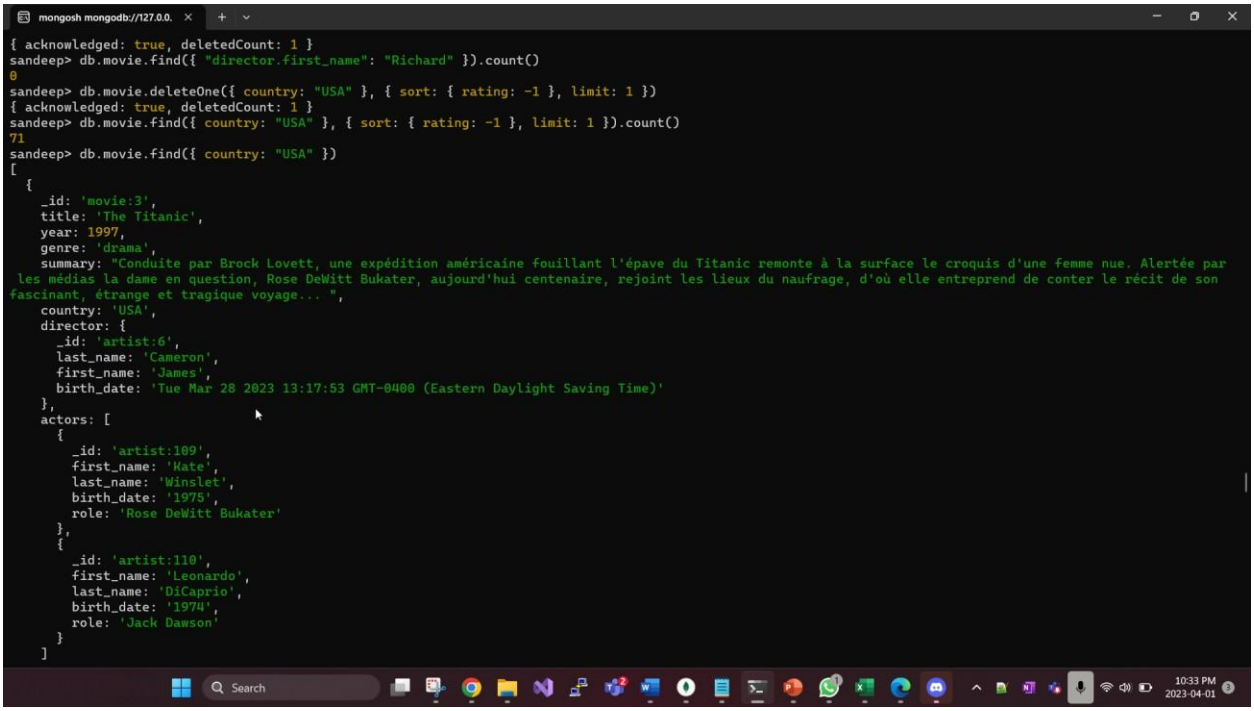
   This command removes all documents that match the { "director.first_name": "Richard" } filter from the movie collection. Make sure to use the correct filter to ensure that you are deleting the correct documents. **db.movie.deleteMany({ "director.first_name": "Richard" })**

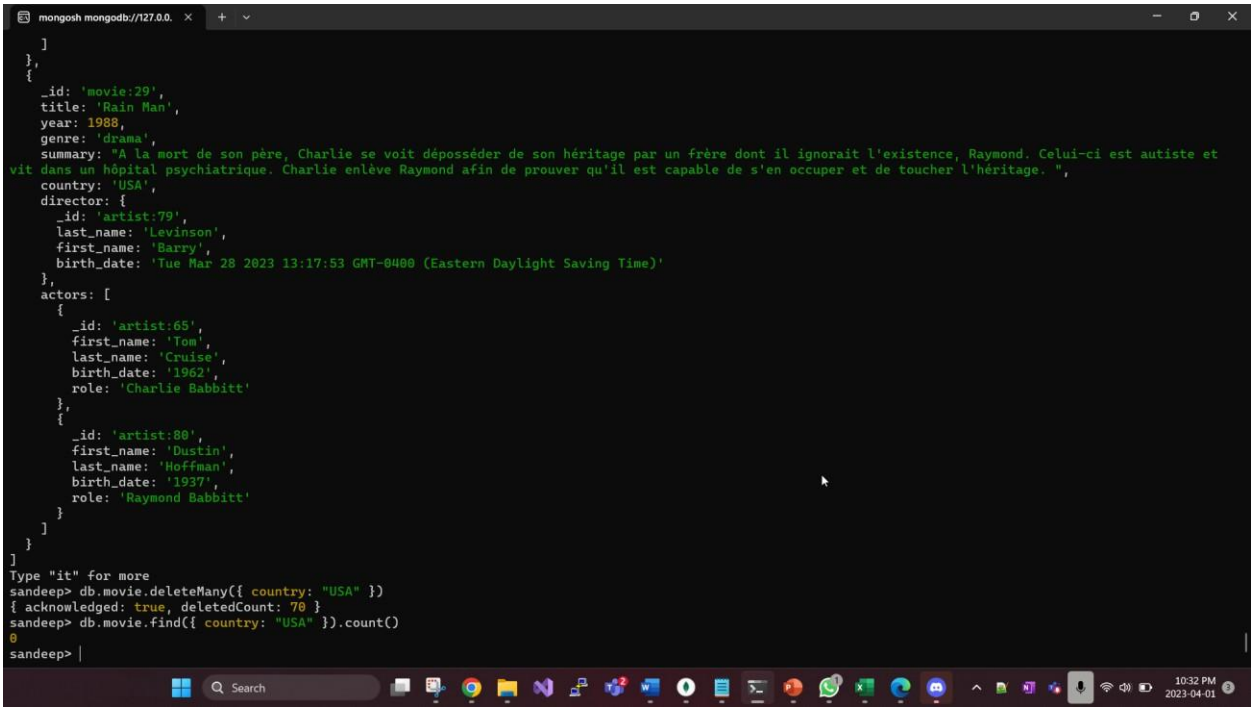


4 Delete a top movie released in USA.

db.movie.deleteOne({ country: "USA" }, { sort: { rating: -1 }, limit: 1 })

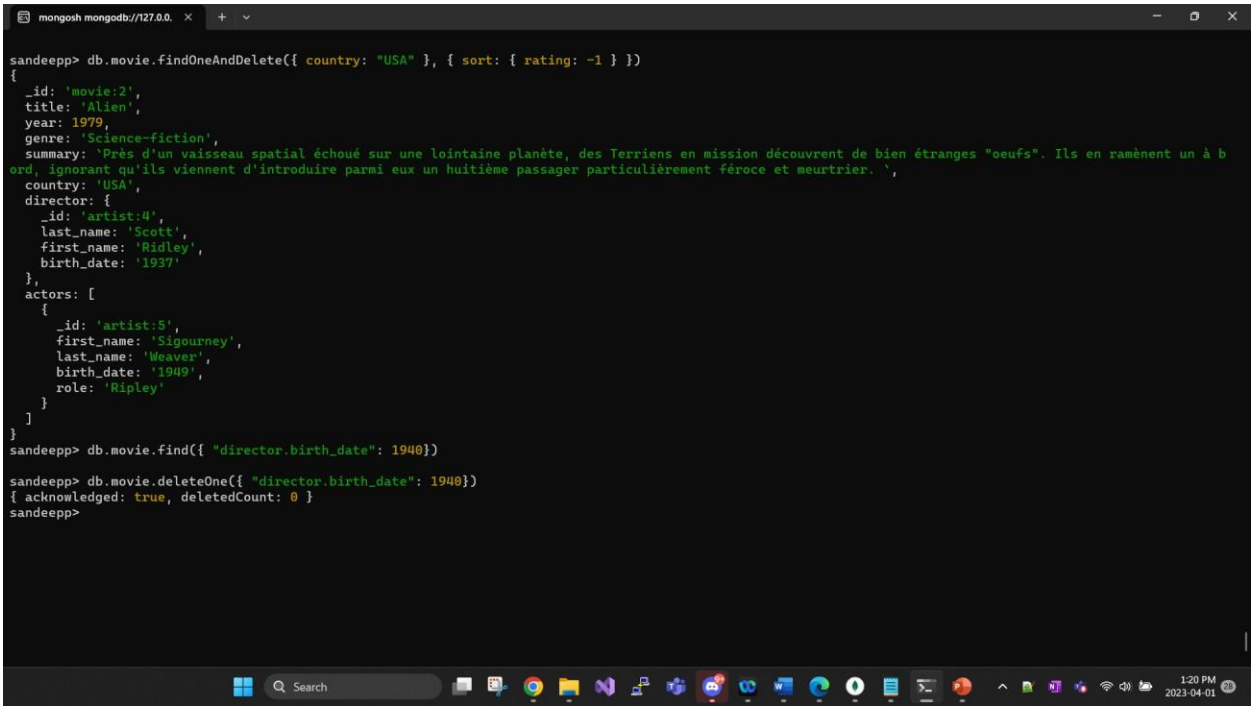
This command finds all documents in the movie collection that have a country value of "USA", sorts them in descending order by rating, and deletes the top document (i.e., the one with the highest rating). If there are multiple movies with the same highest rating, the command will delete only the first one it encounters.



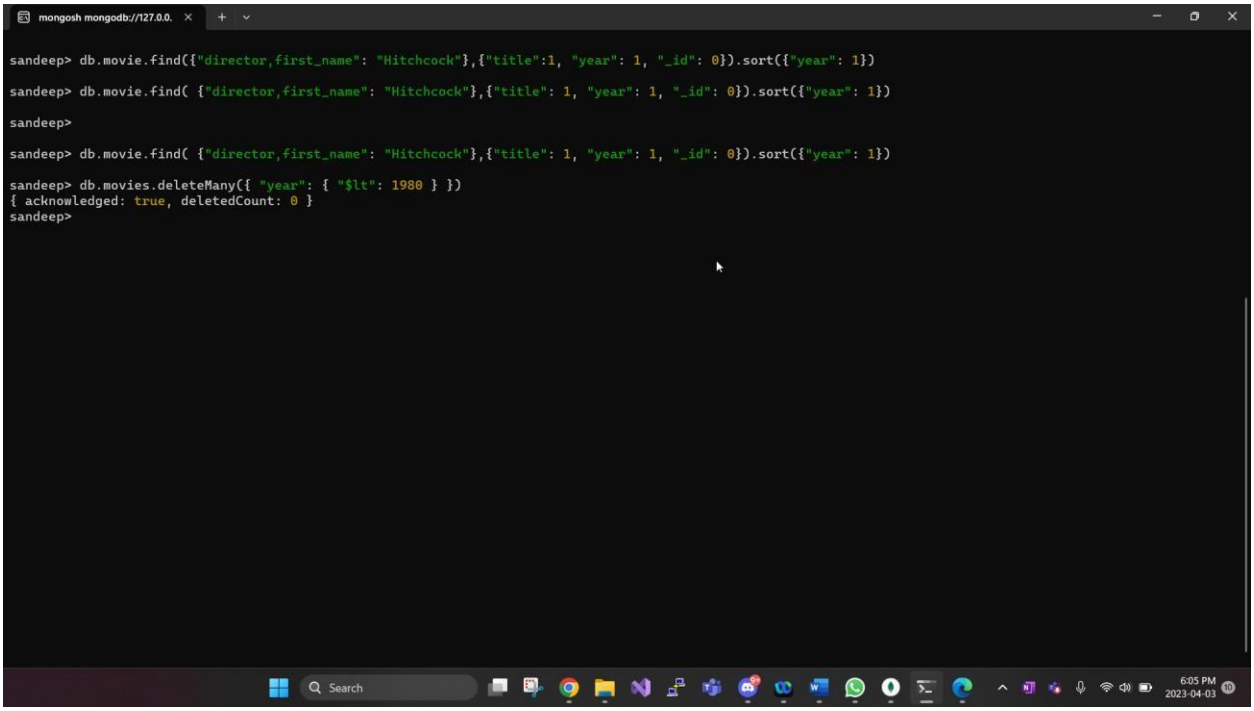


5 Delete a movie with director birth_date is 1940.

There was no birth_date of any director so I got 0. I used deleteOne command.
db.movie.deleteOne({ "director.birth_date": 1940 })



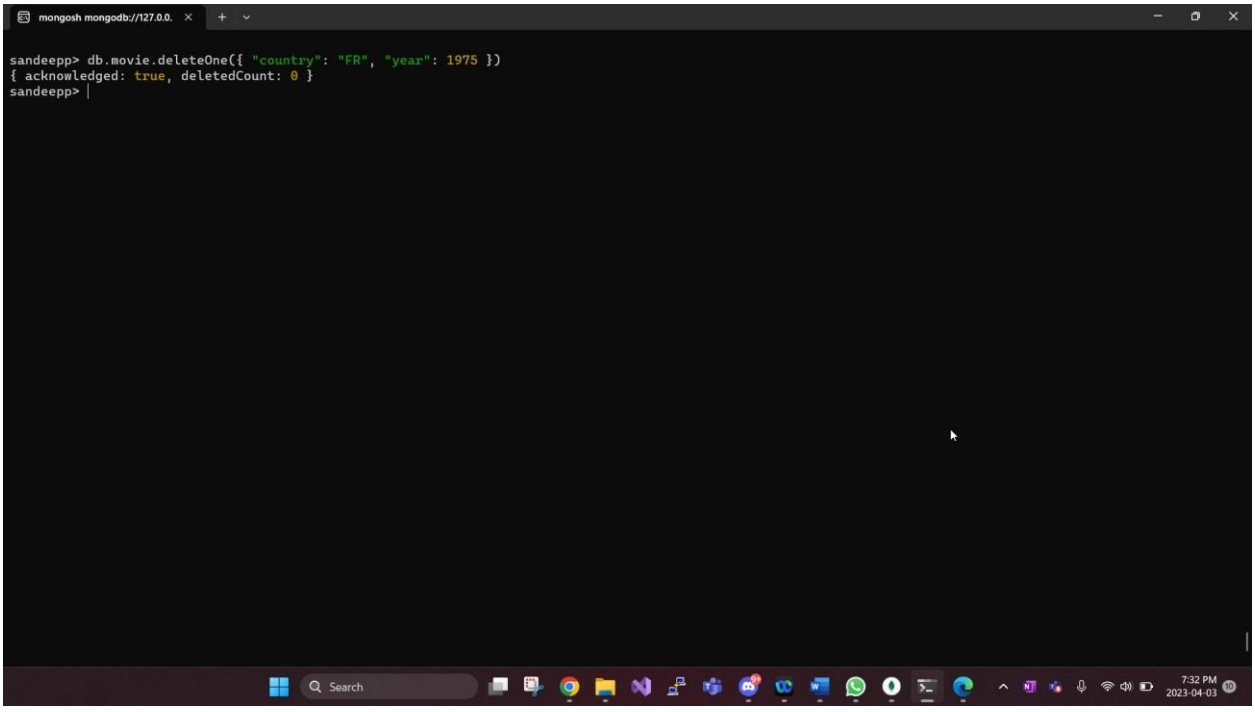
6 **Delete the movies which is older than 1980.**
Next, we use the deleteMany() method to delete the movies which are older than 1980. The query object { "year": { "$lt": 1980 } } specifies the condition for selecting the documents to delete.
db.movie.deleteMany({“year”: { “$lt”: 1980}})



7 Delete the movie which is from country FR and year 1975.

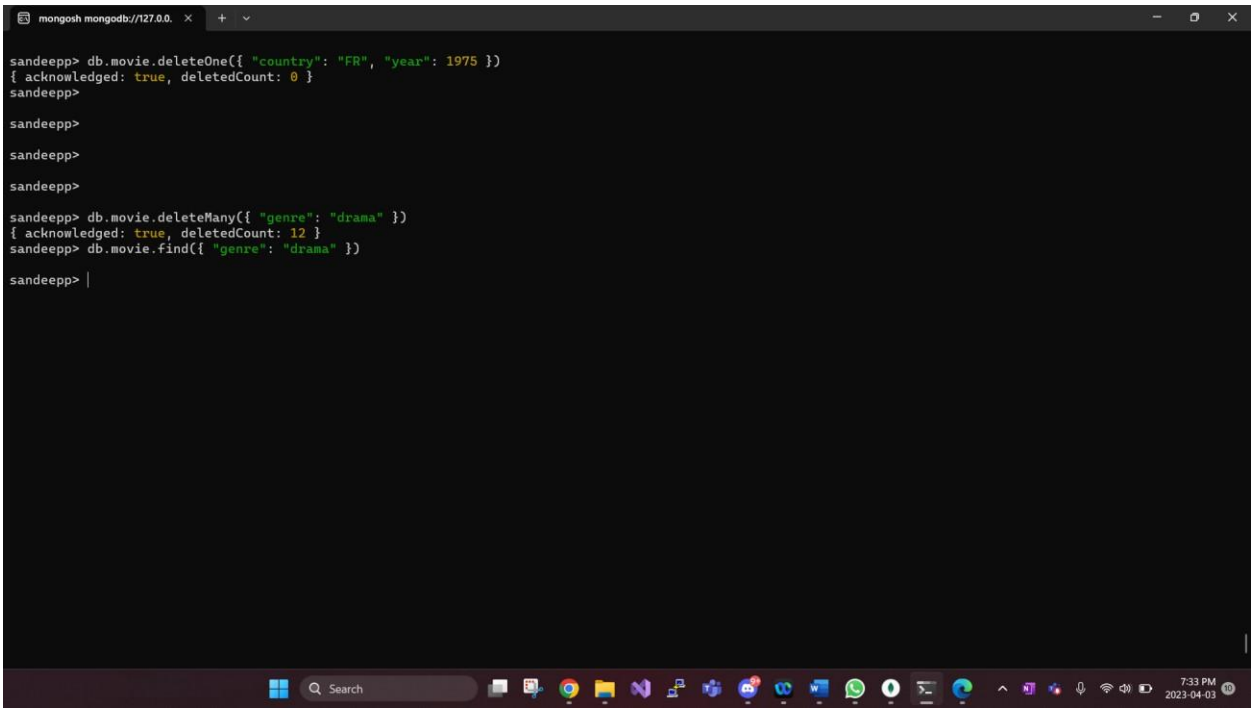
we use the deleteOne() method to delete the movie which is from country "FR" and year 1975. The query object { "country": "FR", "year": 1975 } specifies the condition for selecting the document to delete.

db.movie.deleteOne({“country”: “FR”, “year”: 1975})



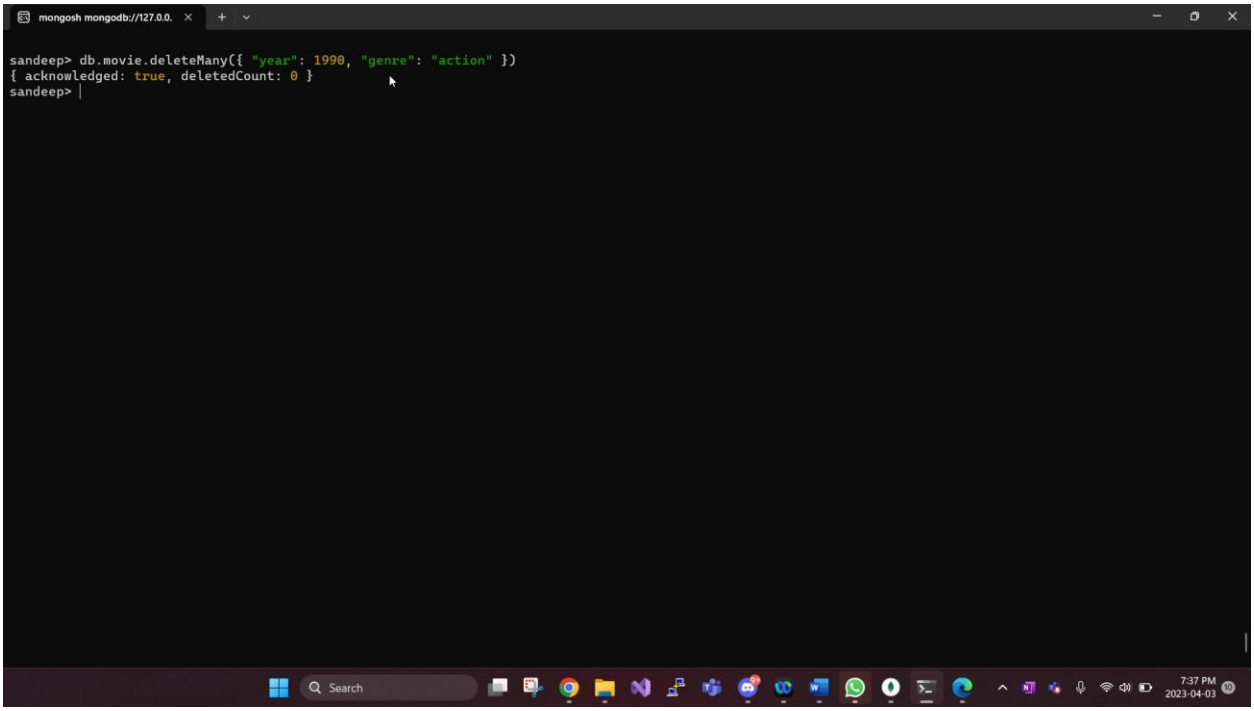
8 Delete the movies with genre is drama.

we use the deleteMany() method to delete the movies with the genre "drama". The query object { "genre": "drama" } specifies the condition for selecting the documents to delete. **db.movie.deleteMany({“genre”: “drama”})**



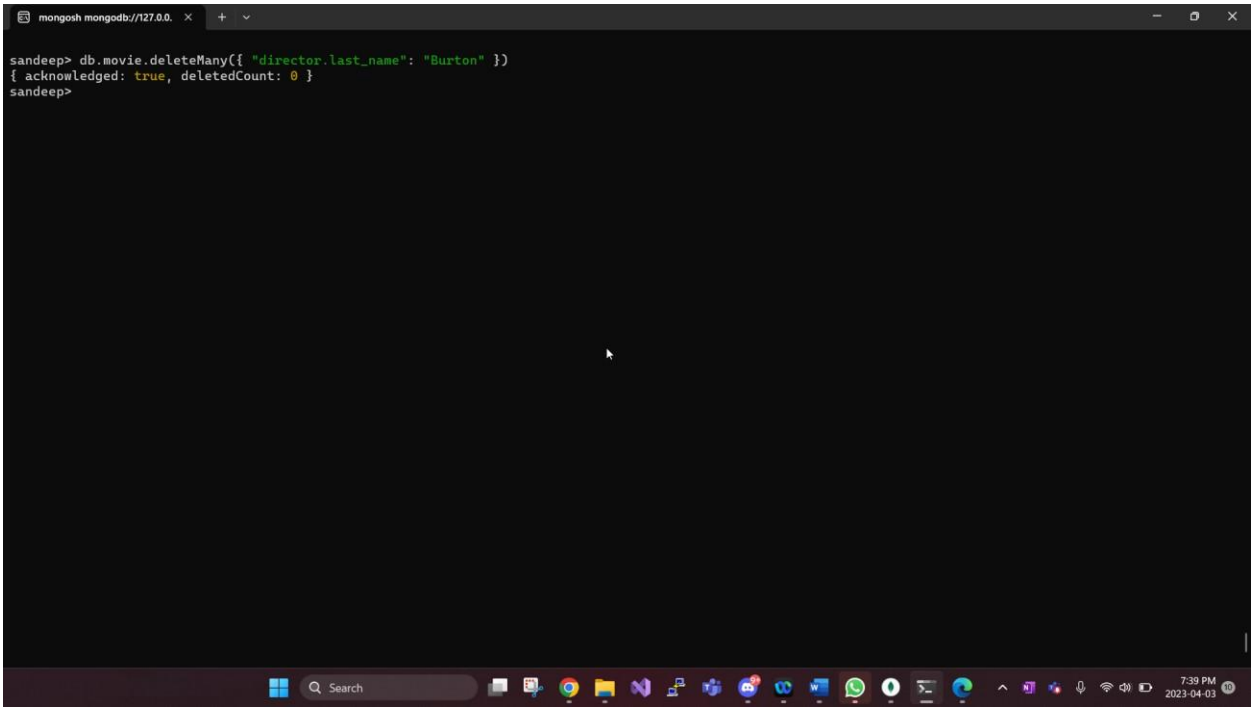
9 Delete the movies with year is 1990 and genre is Action.

we use the deleteMany() method to delete the movies with year 1990 and genre "Action". The query object { "year": 1990, "genre": "Action" } specifies the condition for selecting the documents to delete. in this I have already deleted so it is showing 0 the command is **db.movie.deleteMany({“year”: 1990, “genre”: “action”})**



10 Delete the movies with last_name “Burton”.

we use the deleteMany() method to delete the movies with last name "Burton". The query object { "last_name": "Burton" } specifies the condition for selecting the documents to delete. **db.movie.deleteMany({“director.last_name”: “Burton”})**


mongosh mongodb://127.0.0.
sandeep> db.movie.deleteMany({ "director.last_name": "Burton" })
{ acknowledged: true, deletedCount: 0 }
sandeep>
Search
7:39 PM
2023-04-03