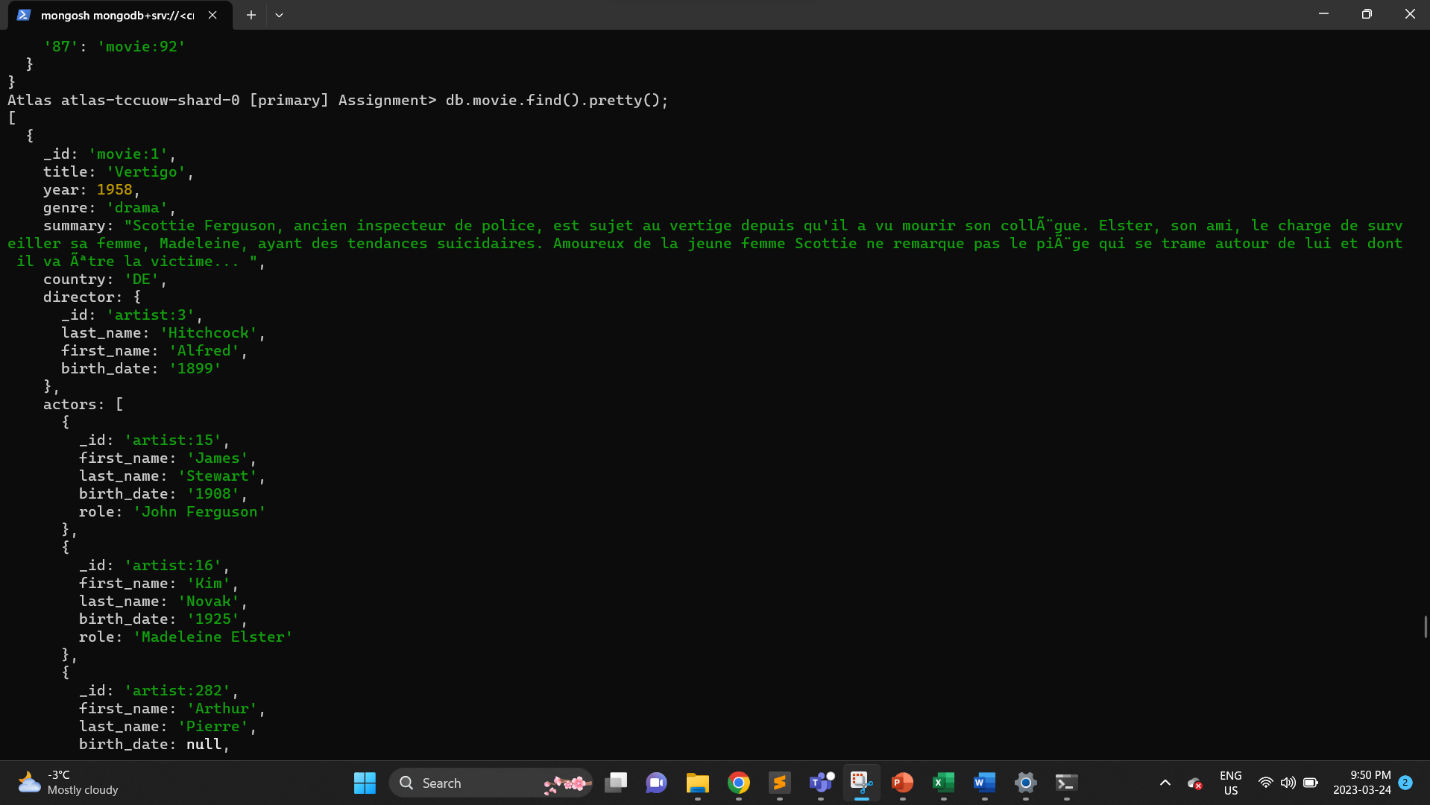
## Write the following script to fetch the document on movie collection.

1. Fetch all the document from movie collection.

db.movie.find.pretty();

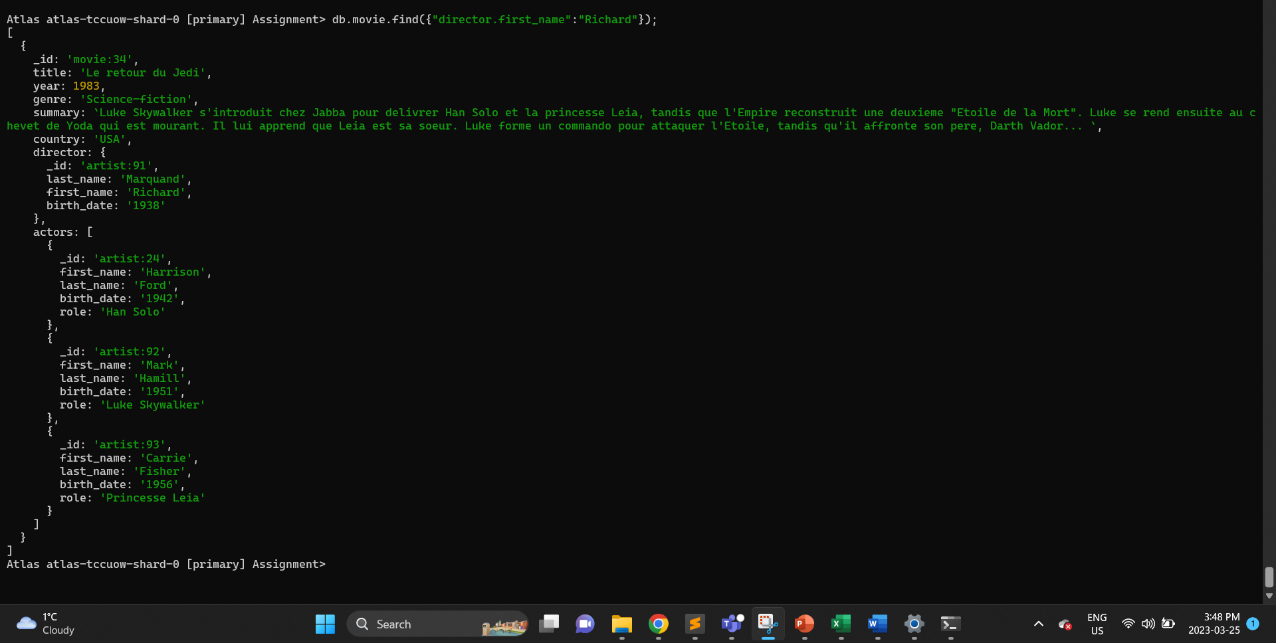


find: Used to query a MongoDB collection and retrieve documents that match the specified query criteria.

.pretty(): Formats the output of a find command to be more readable, by displaying the documents in a formatted JSON structure.

2. Fetch all the document with director firstname Richard

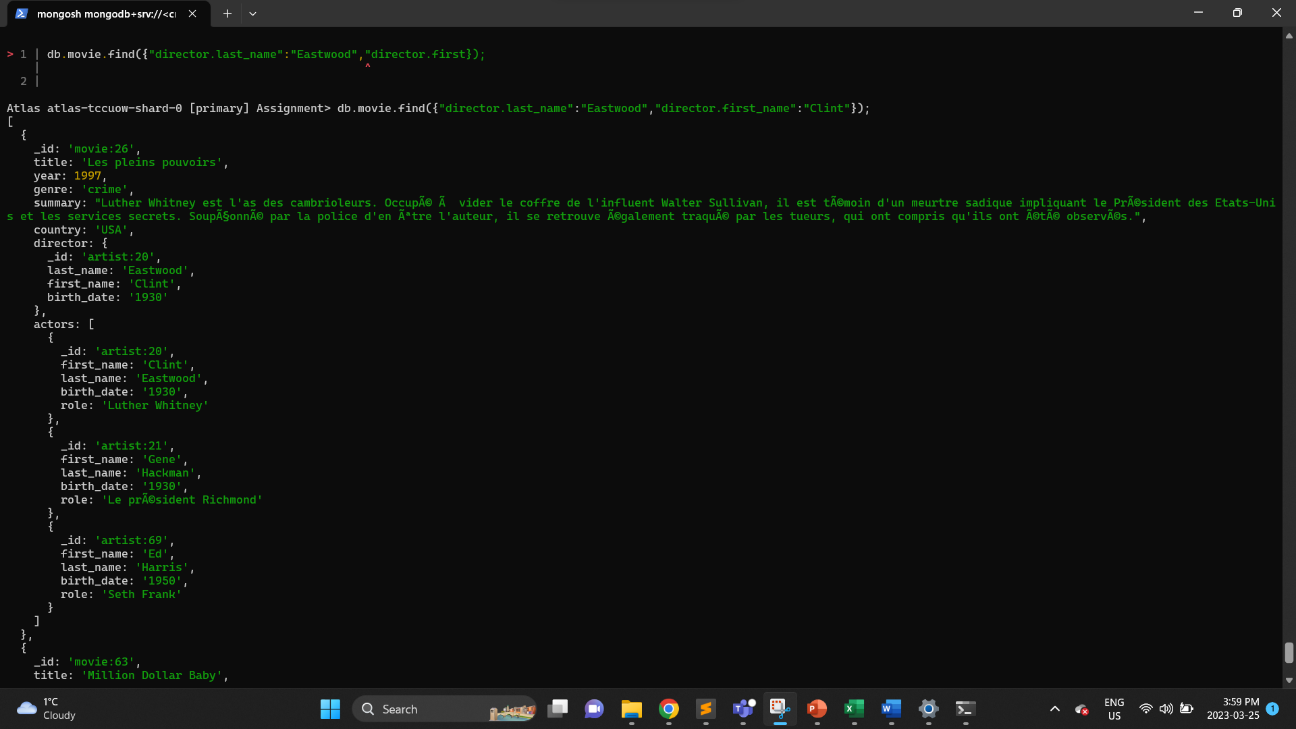
db.movie.find({"director.first\_name":"Richard"});



This is a MongoDB find command that searches the "movie" collection for documents where the "director" subdocument has a field called "first\_name" with a value of "Richard". This query will return all the matching documents in the collection.

3. Fetch all the document with director Eastwood Clint.

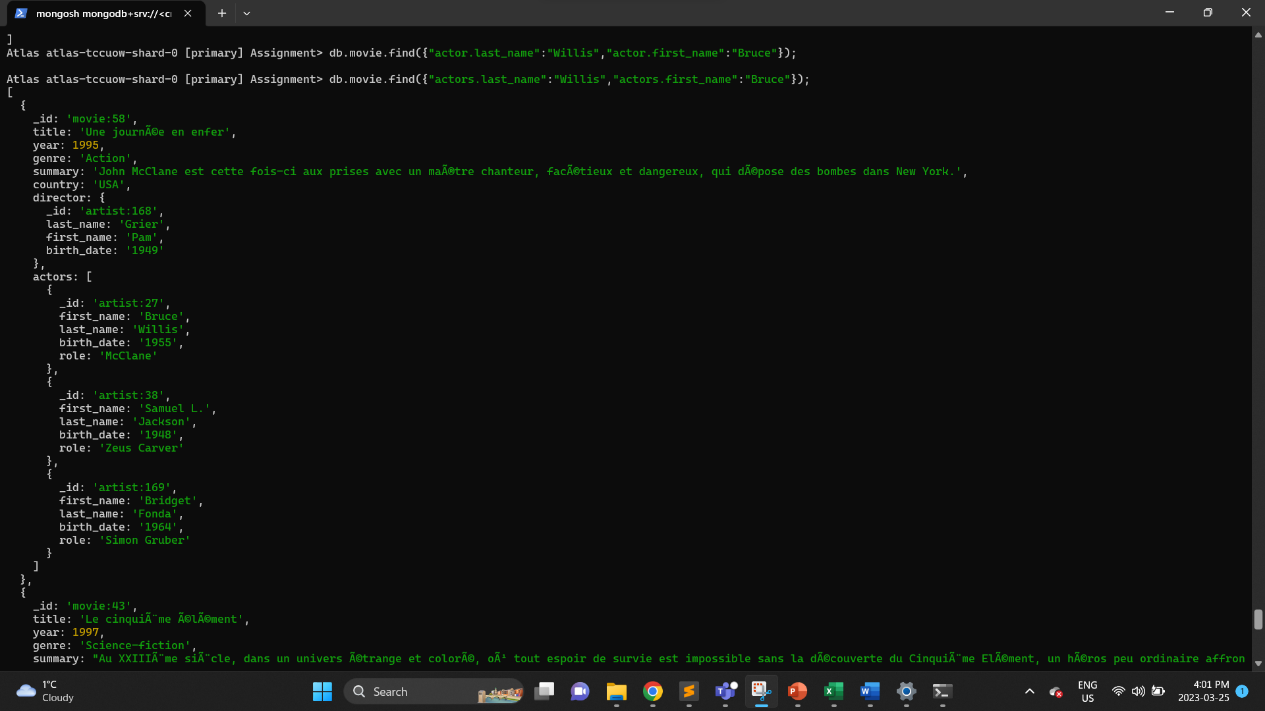
db.movie.find({"director.last\_name":"Eastwood","director.first\_name":"Clint"});



This is a MongoDB find command that searches the "movie" collection for documents where the "director" subdocument has a field called "first\_name" with a value of "Clint" and "last\_name" with value " Eastwood". This query will return all the matching documents in the collection.

4. Fetch all the document with actor â€œBruce Willisâ€.

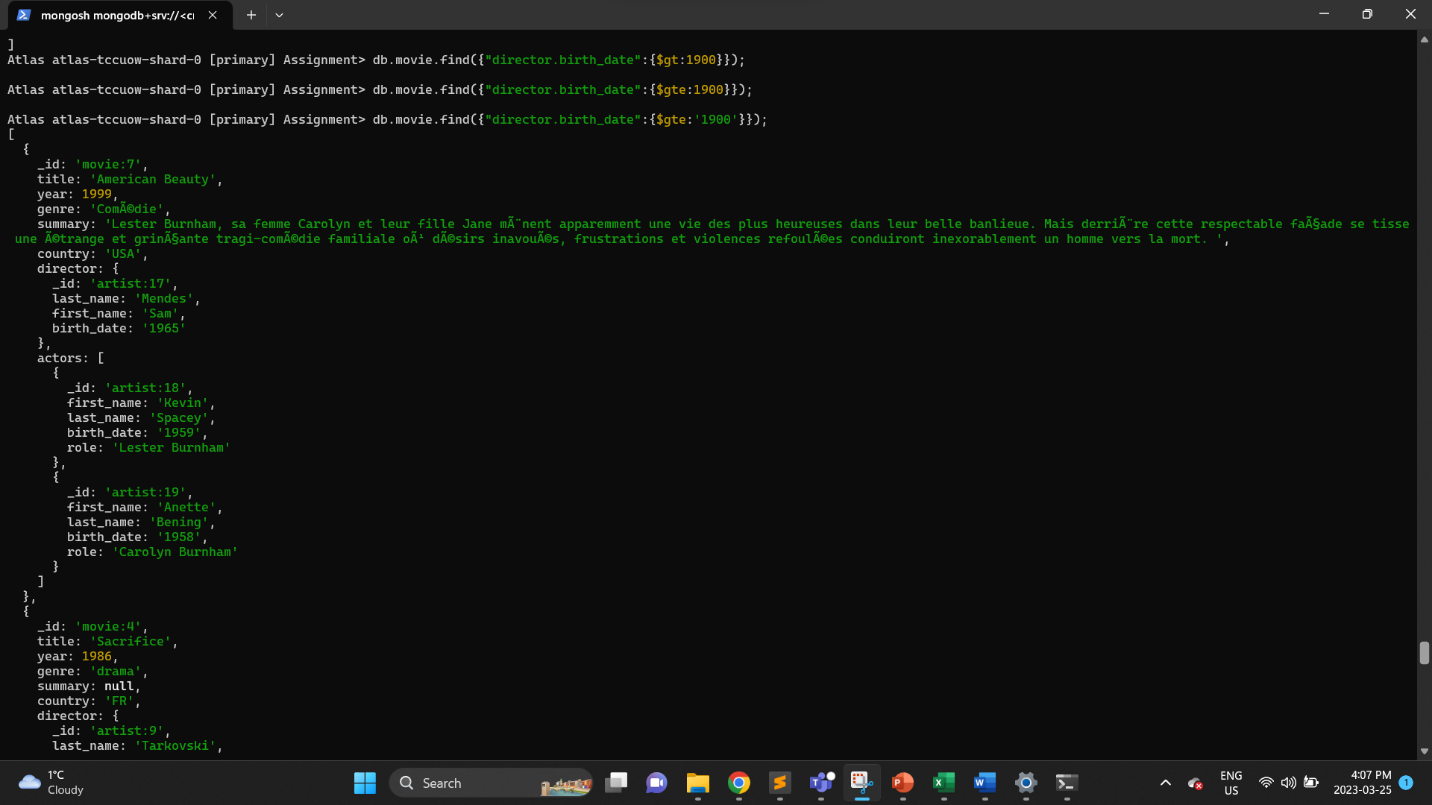
db.movie.find({"actors.last\_name":"Willis","actors.first\_name":"Bruce"});



This is a MongoDB find command that searches the "movie" collection for documents where the "actors" subdocument has a field called "last\_name" with a value of "Willis" and a field called "first\_name" with a value of "Bruce". This query will return all the matching documents in the collection.

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

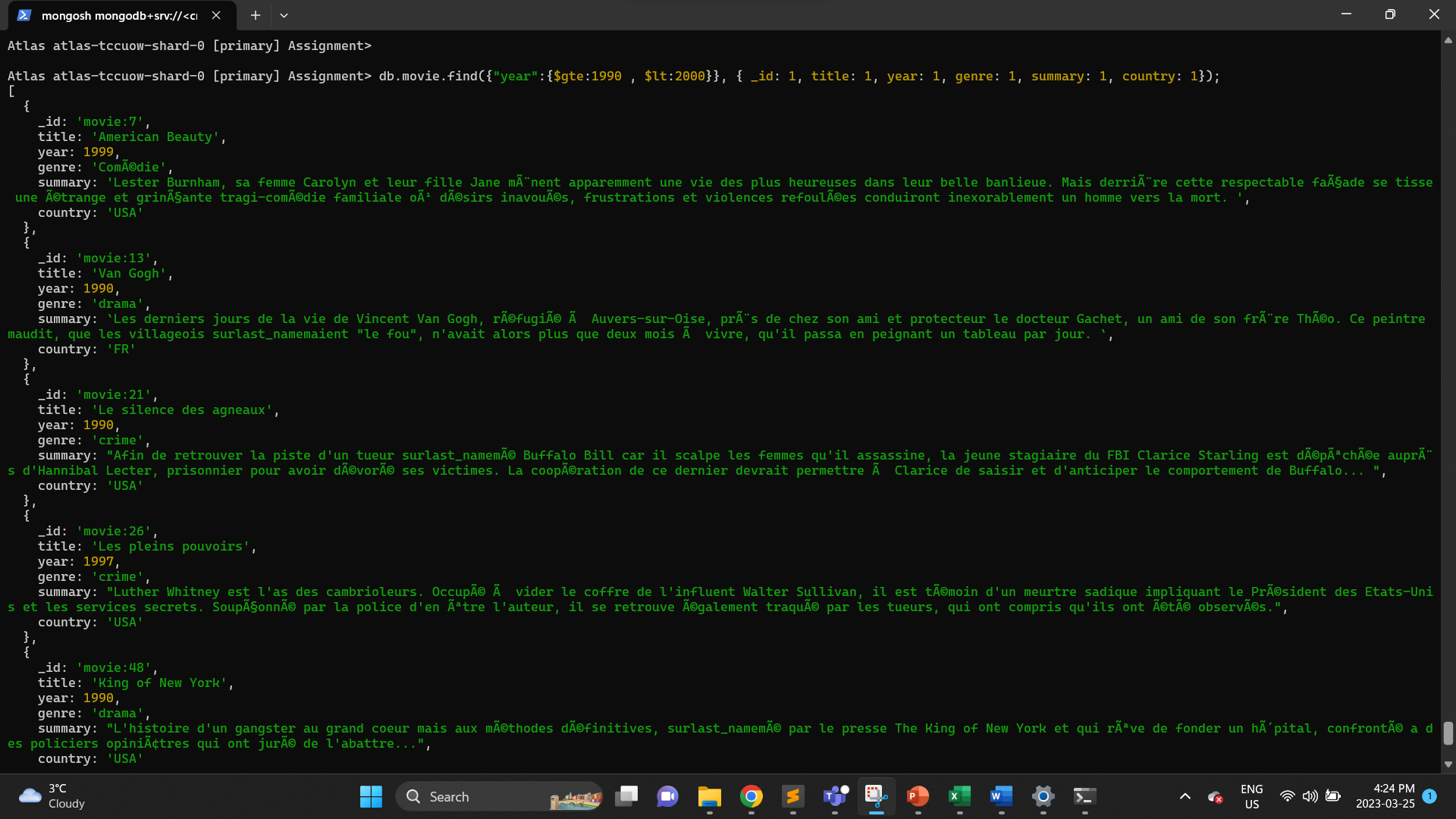
db.movie.find({"director.birth\_date":{$gte:'1900'}});



This is a MongoDB find command that searches the "movie" collection for documents where the "director" subdocument has a field called "birth\_date" with a value greater than or equal to '1900'. This query will return all the matching documents in the collection.

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

db.movie.find({"year":{$gte:1990 , $lt:2000}}, { \_id: 1, title: 1, year: 1, genre: 1, summary: 1, country: 1});

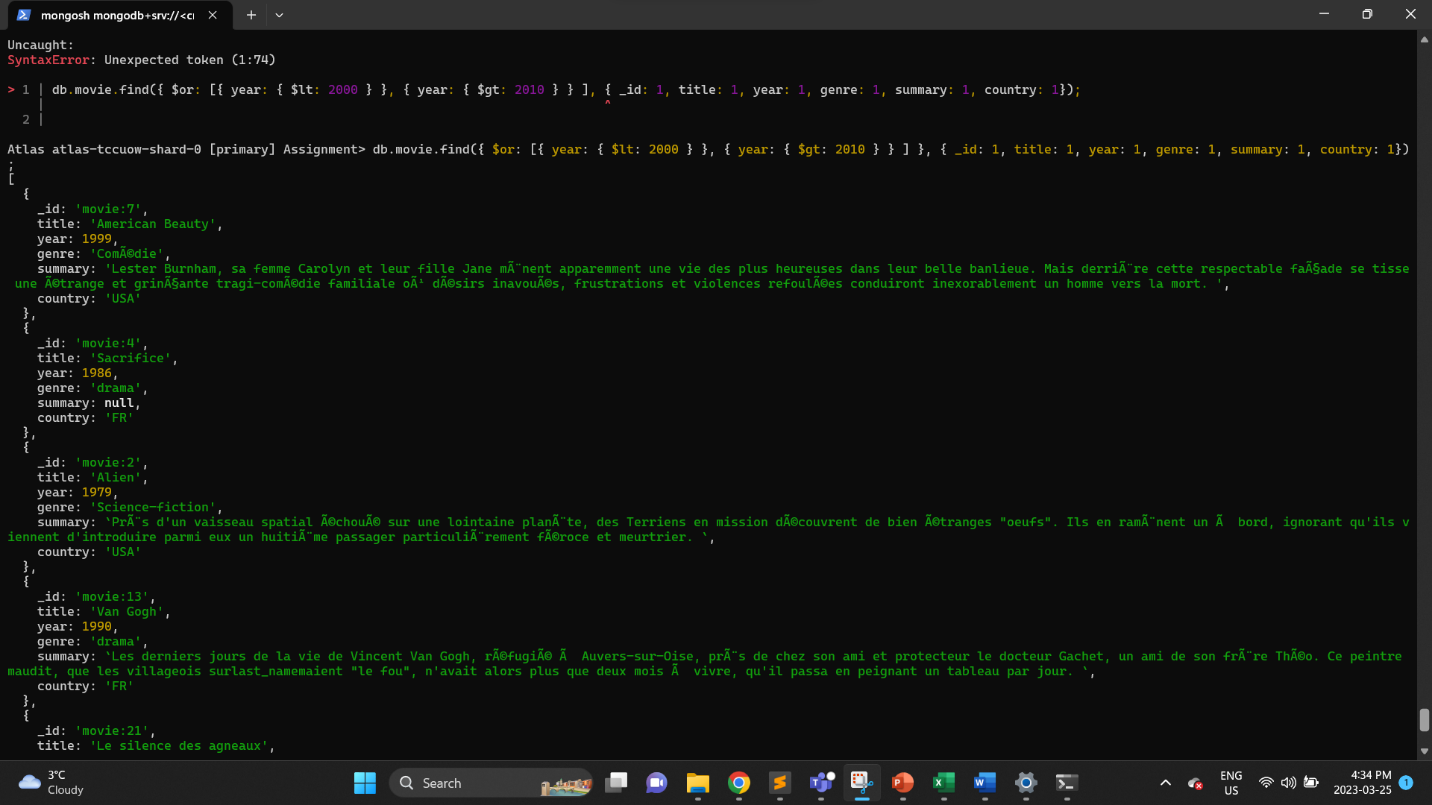


This is a MongoDB find command that searches the "movie" collection for documents where the "year" field has a value greater than or equal to 1990 and less than 2000.

The second parameter of this command { \_id: 1, title: 1, year: 1, genre: 1, summary: 1, country: 1 } specifies which fields to include in the returned documents. This query will return only the specified fields (i.e., \_id, title, year, genre, summary, and country) for all the documents that match the specified criteria.

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

db.movie.find({ $or: [{ year: { $lt: 2000 } }, { year: { $gt: 2010 } } ] }, { \_id: 1, title: 1, year: 1, genre: 1, summary: 1, country: 1});

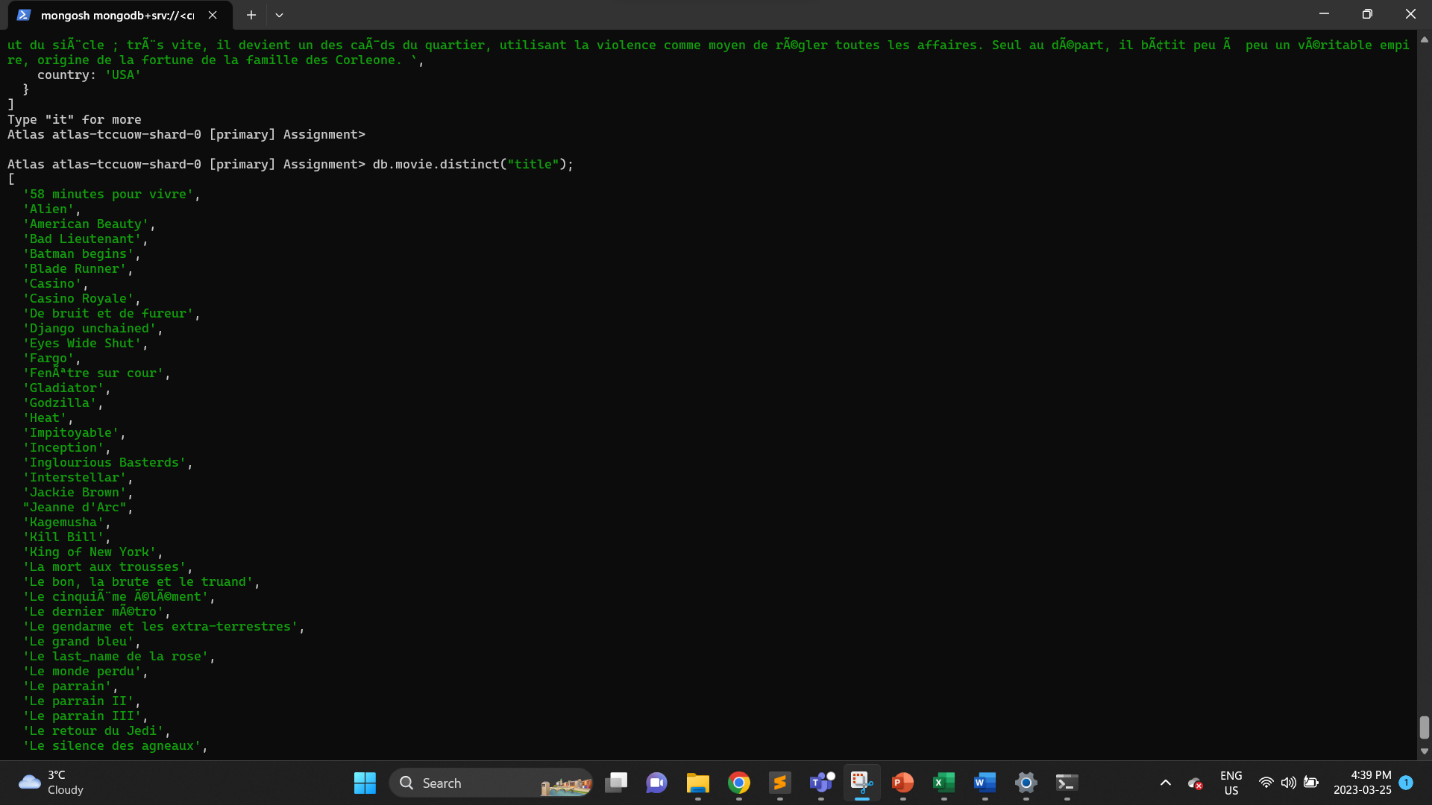


This is a MongoDB find command that searches the "movie" collection for documents where the "year" field has a value less than 2000 or greater than 2010.

The second parameter of this command { \_id: 1, title: 1, year: 1, genre: 1, summary: 1, country: 1 } specifies which fields to include in the returned documents. This query will return only the specified fields (i.e., \_id, title, year, genre, summary, and country) for all the documents that match the specified criteria.

8. Fetch all the movie distinct title.

db.movie.distinct("title");

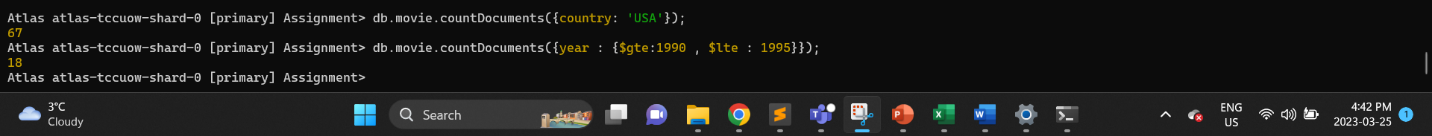


This is a MongoDB distinct command that returns an array of unique values for the "title" field in the "movie" collection.

Each element in the returned array will be a distinct value that appears in the "title" field of at least one document in the collection. This query will not return any duplicate values.

9. Count the number of movies released in USA.

db.movie.countDocuments({country: 'USA'});

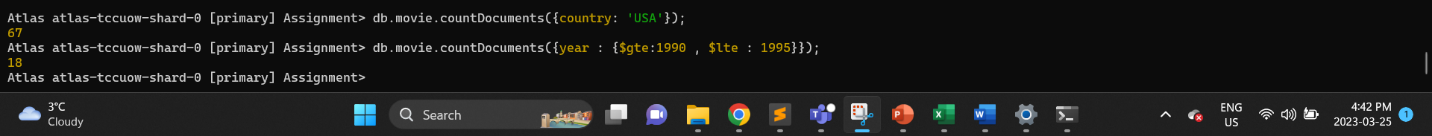


This is a MongoDB countDocuments command that returns the count of documents in the "movie" collection where the "country" field has a value of "USA".

This query will return the number of documents in the collection that match the specified criteria, in this case, the number of movies with the country of production being USA.

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

db.movie.countDocuments({ year : {$gte:1990 , $lte : 1995}});

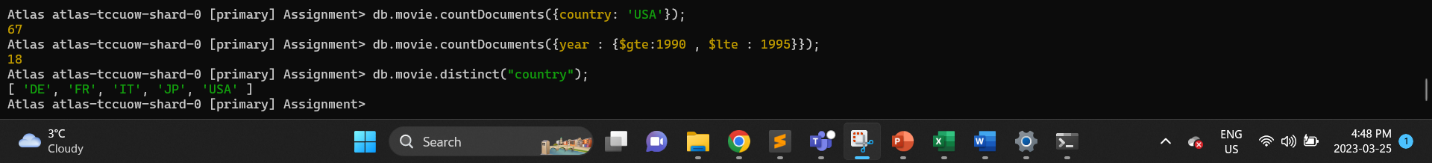


This is a MongoDB countDocuments command that returns the count of documents in the "movie" collection where the "year" field has a value greater than or equal to 1990 and less than or equal to 1995.

This query will return the number of documents in the collection that match the specified criteria, in this case, the number of movies released between 1990 and 1995 (inclusive).

11. Fetch all the country where movies released.

db.movie.distinct("country");

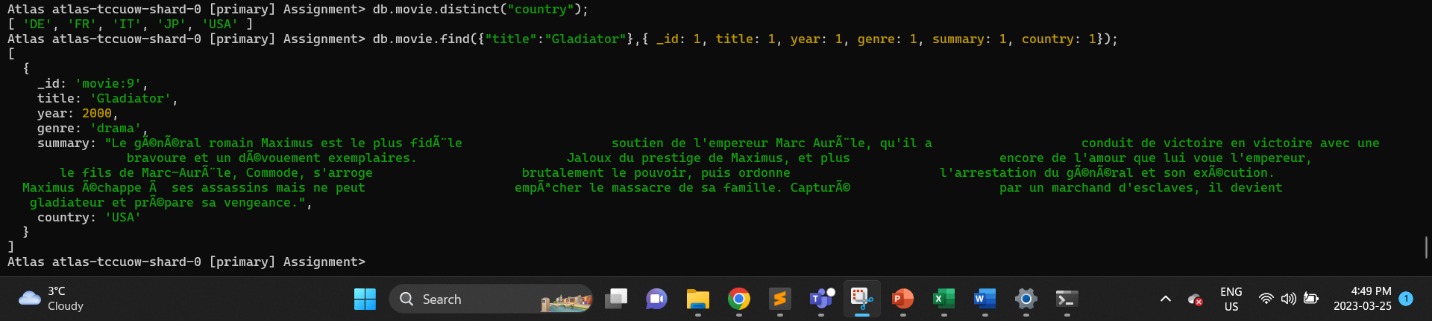


This is a MongoDB distinct command that returns an array of unique values for the "country" field in the "movie" collection.

Each element in the returned array will be a distinct value that appears in the "country" field of at least one document in the collection. This query will not return any duplicate values.

12. Fetch all the document with movies titled as â€œGladiatorâ€.

db.movie.find({"title":"Gladiator"},{ \_id: 1, title: 1, year: 1, genre: 1, summary: 1, country: 1});



This is a MongoDB find command that searches the "movie" collection for documents where the "title" field has a value of "Gladiator".

The second parameter of this command { \_id: 1, title: 1, year: 1, genre: 1, summary: 1, country: 1 } specifies which fields to include in the returned documents. This query will return only the specified fields (i.e., \_id, title, year, genre, summary, and country) for the document that matches the specified criteria, in this case, the movie with the title "Gladiator".

13. Fetch distinct genre values of movies.

db.movie.distinct("genre");

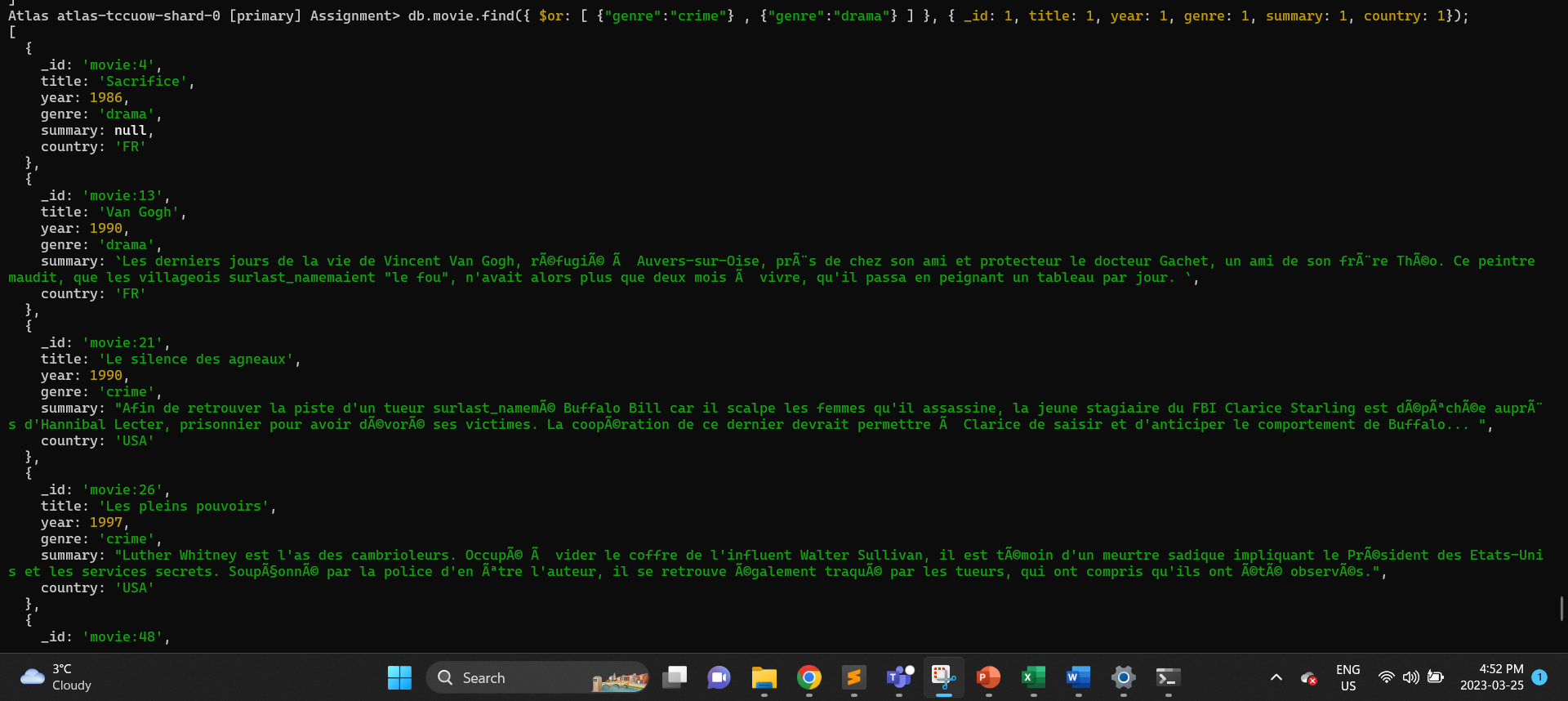


This is a MongoDB distinct command that returns an array of unique values for the "genre" field in the "movie" collection.

Each element in the returned array will be a distinct value that appears in the "genre" field of at least one document in the collection. This query will not return any duplicate values.

14. Fetch all the document with movie â€œcrime" or "drama" genre.

db.movie.find({ $or: [ {"genre":"crime"} , {"genre":"drama"} ] }, { \_id: 1, title: 1, year: 1, genre: 1, summary: 1, country: 1});

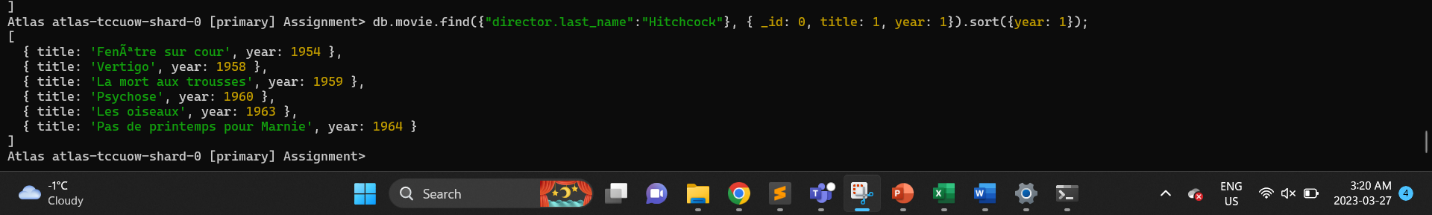


This is a MongoDB find command that searches the "movie" collection for documents where the "genre" field has a value of either "crime" or "drama".

The second parameter of this command { \_id: 1, title: 1, year: 1, genre: 1, summary: 1, country: 1 } specifies which fields to include in the returned documents. This query will return only the specified fields (i.e., \_id, title, year, genre, summary, and country) for all the documents that match the specified criteria, in this case, movies that have either "crime" or "drama" as their genre.

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

db.movie.find({"director.last\_name":"Hitchcock"}, { \_id: 0, title: 1, year: 1}).sort({year: 1});



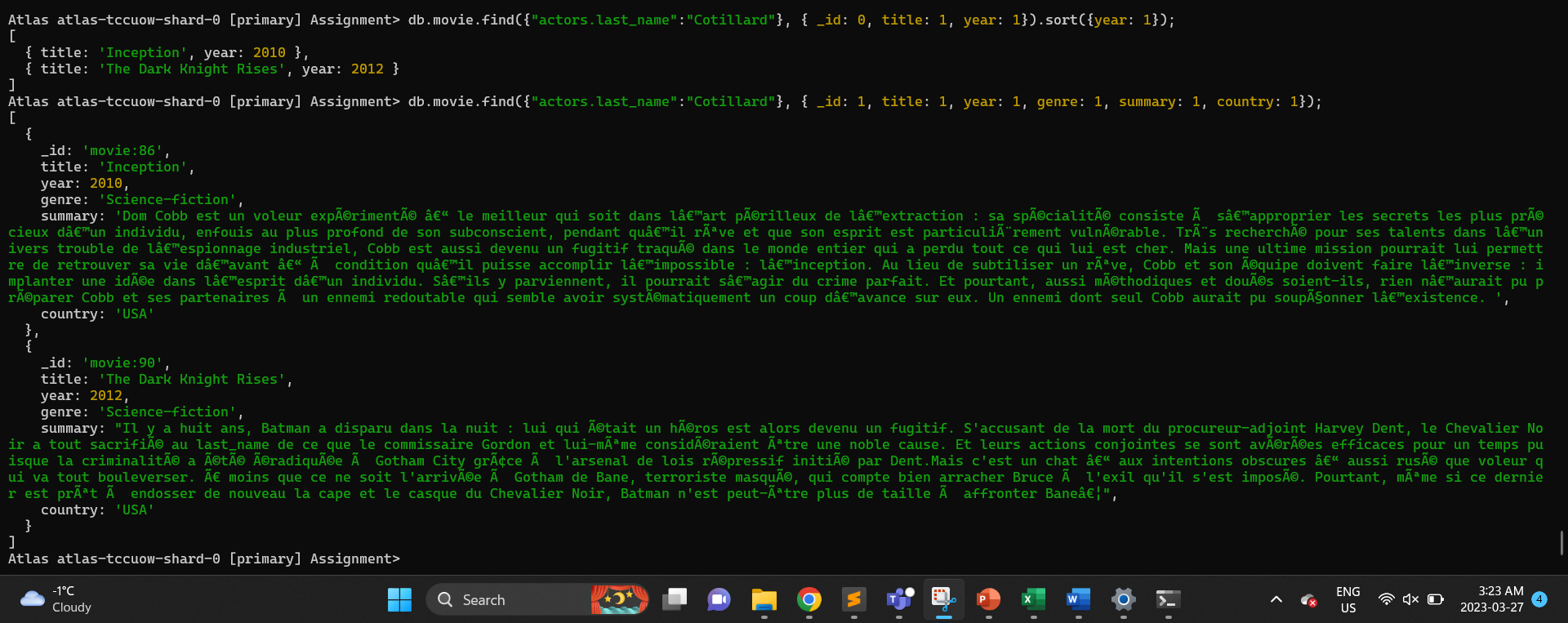
This is a MongoDB find command that searches the "movie" collection for documents where the "last\_name" field of the "director" subdocument has a value of "Hitchcock".

The second parameter of this command { \_id: 0, title: 1, year: 1 } specifies which fields to include in the returned documents. This query will return only the specified fields (i.e., title and year) for all the documents that match the specified criteria, in this case, movies directed by Alfred Hitchcock.

The sort method is used to sort the returned documents in ascending order based on the "year" field. This query will return the titles and years of all movies directed by Alfred Hitchcock, sorted by year in ascending order.

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

db.movie.find({"actors.last\_name":"Cotillard"}, { \_id: 1, title: 1, year: 1, genre: 1, summary: 1, country: 1});

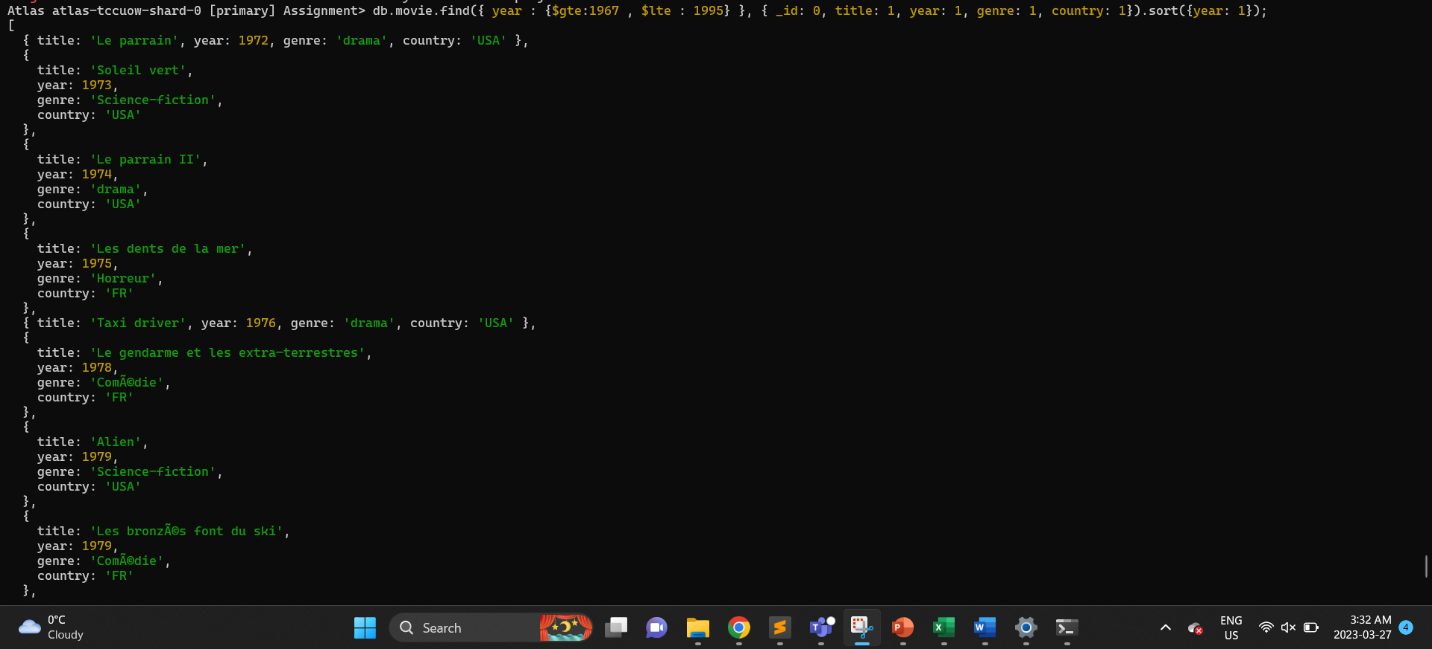


This is a MongoDB find command that searches the "movie" collection for documents where the "last\_name" field of the "actors" subdocument has a value of "Cotillard".

The second parameter of this command { \_id: 1, title: 1, year: 1, genre: 1, summary: 1, country: 1 } specifies which fields to include in the returned documents. This query will return only the specified fields (i.e., \_id, title, year, genre, summary, and country) for all the documents that match the specified criteria, in this case, movies in which Marion Cotillard has acted.

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

db.movie.find({ year : {$gte:1967 , $lte : 1995} }, { \_id: 0, title: 1, year: 1, genre: 1, country: 1}).sort({year: 1});



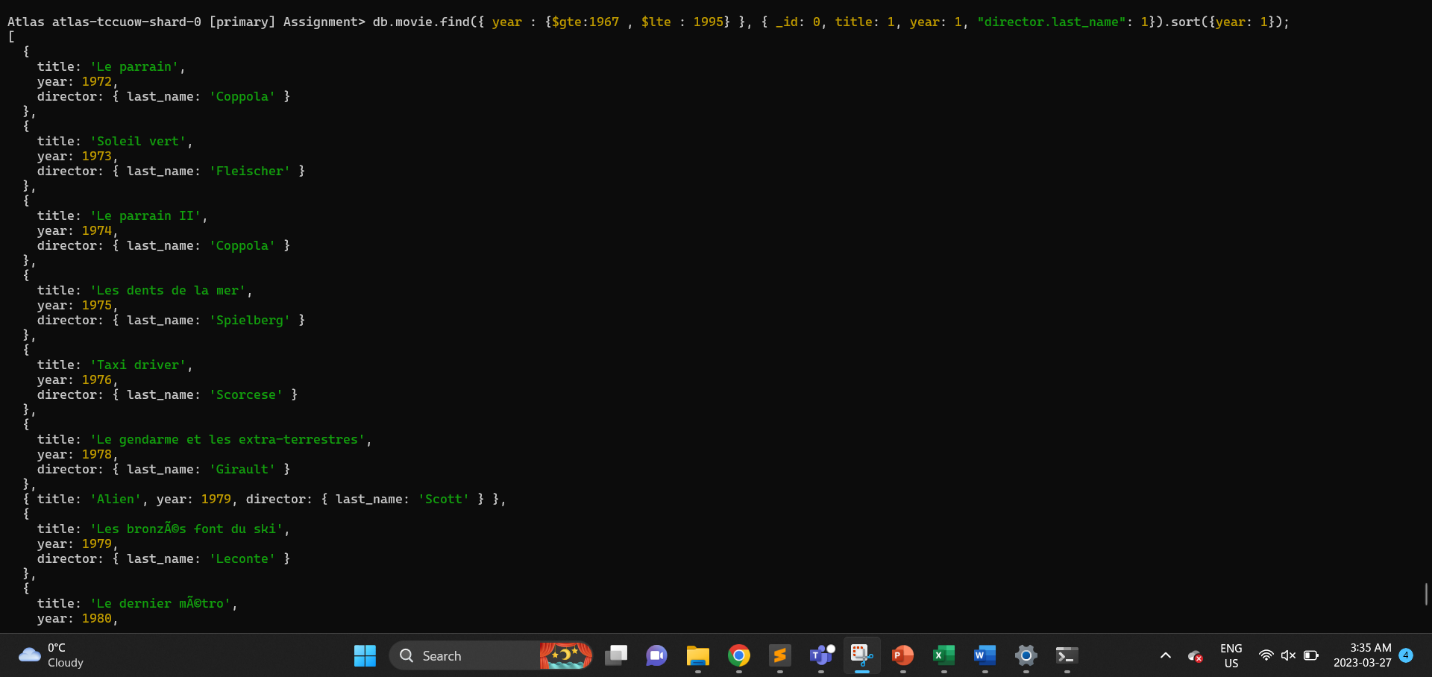
This is a MongoDB find command that searches the "movie" collection for documents where the "year" field has a value greater than or equal to 1967 and less than or equal to 1995.

The second parameter of this command { \_id: 0, title: 1, year: 1, genre: 1, country: 1 } specifies which fields to include in the returned documents. This query will return only the specified fields (i.e., title, year, genre, and country) for all the documents that match the specified criteria.

The sort method is used to sort the returned documents in ascending order based on the "year" field. This query will return the titles, years, genres, and countries of all movies released between 1967 and 1995, sorted by year in ascending order.

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

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



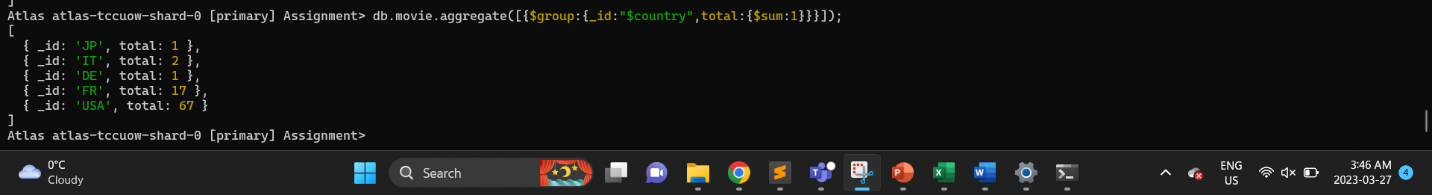
This is a MongoDB find command that searches the "movie" collection for documents where the "year" field has a value greater than or equal to 1967 and less than or equal to 1995.

The second parameter of this command { \_id: 0, title: 1, year: 1, "director.last\_name": 1 } specifies which fields to include in the returned documents. This query will return only the specified fields (i.e., title, year, and the last name of the movie director) for all the documents that match the specified criteria.

The sort method is used to sort the returned documents in ascending order based on the "year" field. This query will return the titles, years, and last names of the directors of all movies released between 1967 and 1995, sorted by year in ascending order.

19. Fetch the number of movies by country.

db.movie.aggregate([{$group:{\_id:"$country",total:{$sum:1}}}]);



This is a MongoDB aggregate command that groups documents in the "movie" collection by their "country" field and calculates the total number of documents in each group using the $sum aggregation operator.

The $group aggregation stage has two fields, \_id and total. The \_id field specifies the grouping criterion, in this case, the "country" field of the documents. The total field uses the $sum operator to calculate the total number of documents in each group.

This query returns an array of documents that contain the \_id field, which represents the country name, and the total field, which represents the total number of documents in the group. The output will show the total number of movies produced in each country in the collection.

20. Fetch the number of movies by country and actor.

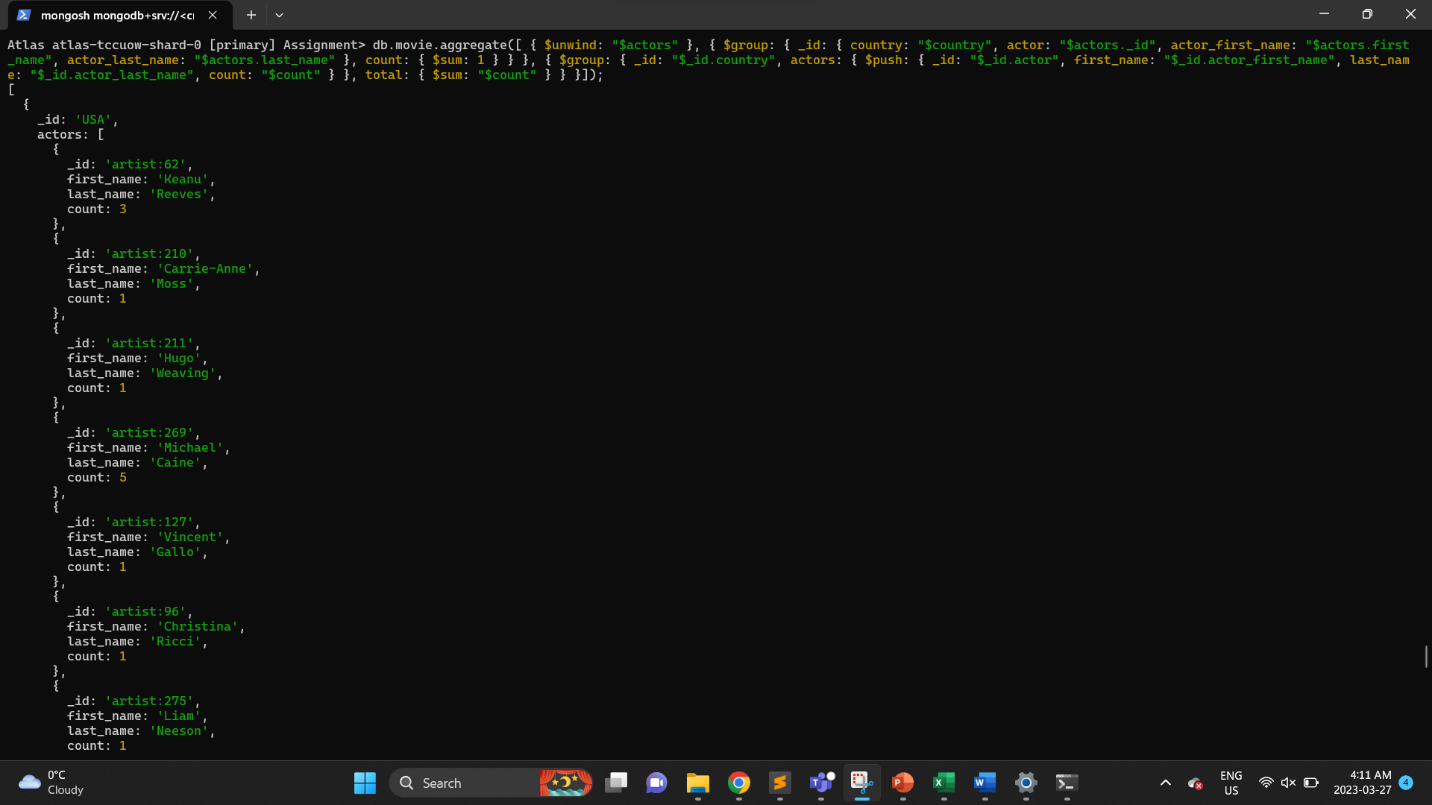
db.movie.aggregate([

{ $unwind: "$actors" },

{ $group: { \_id: { country: "$country", actor: "$actors.\_id", actor\_first\_name: "$actors.first\_name", actor\_last\_name: "$actors.last\_name" }, count: { $sum: 1 } } },

{ $group: { \_id: "$\_id.country", actors: { $push: { \_id: "$\_id.actor", first\_name: "$\_id.actor\_first\_name", last\_name: "$\_id.actor\_last\_name", count: "$count" } }, total: { $sum: "$count" } } }

]);



This is a MongoDB aggregate command that performs a multi-stage aggregation pipeline on the "movie" collection.

The first stage of the pipeline uses the $unwind operator to create a new document for each element of the "actors" array in the original documents.

The second stage uses the $group operator to group the unwound documents by country, actor, actor\_first\_name, and actor\_last\_name, and calculates the count of documents in each group.

The third stage uses another $group operator to group the previous groups by country and creates an array of actors with their count of documents for each country. It also calculates the total count of documents for each country.

The output of this aggregation will show the total count of movies produced in each country, along with an array of actors who appeared in those movies, their first name, last name, and the count of movies in which they appeared.

## Write the script to update document on movie collection.

1. Update the movie with title â€œThe Titanicâ€ from â€œTitanicâ€ where \_id is "movie:3".

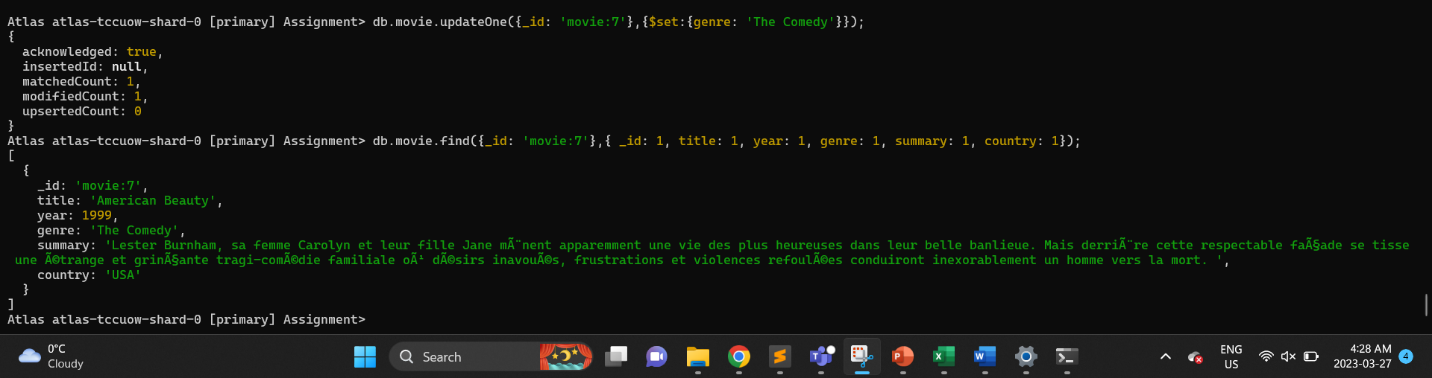
db.movie.updateOne({\_id: 'movie:3'},{$set:{title: 'The Titanic'}});



This MongoDB command updates the title of a document in the "movie" collection where the \_id field is equal to "movie:3". The new title will be "The Titanic".

2. Update the movie with genre â€œThe Comedyâ€ from "ComÃ©die" where \_id is â€œmovie:7â€.

db.movie.updateOne({\_id: 'movie:7'},{$set:{genre: 'The Comedy'}});



This MongoDB command updates the genre field of a document in the "movie" collection where the \_id field is equal to "movie:7". The new genre will be "The Comedy".

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

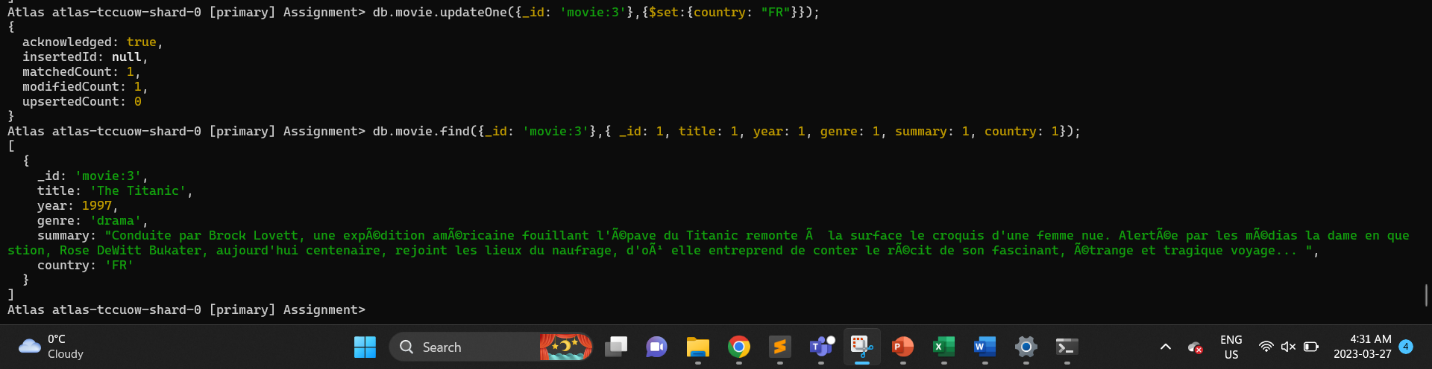
db.movie.updateOne({\_id: 'movie:6'},{$set:{year: 1990}});



The command db.movie.updateOne({\_id: 'movie:6'},{$set:{year: 1990}}) updates the year field of the movie with ID "movie:6" to 1990.

4. Update the movie with country â€œFRâ€ from "USA" where \_id is â€œmovie:3â€.

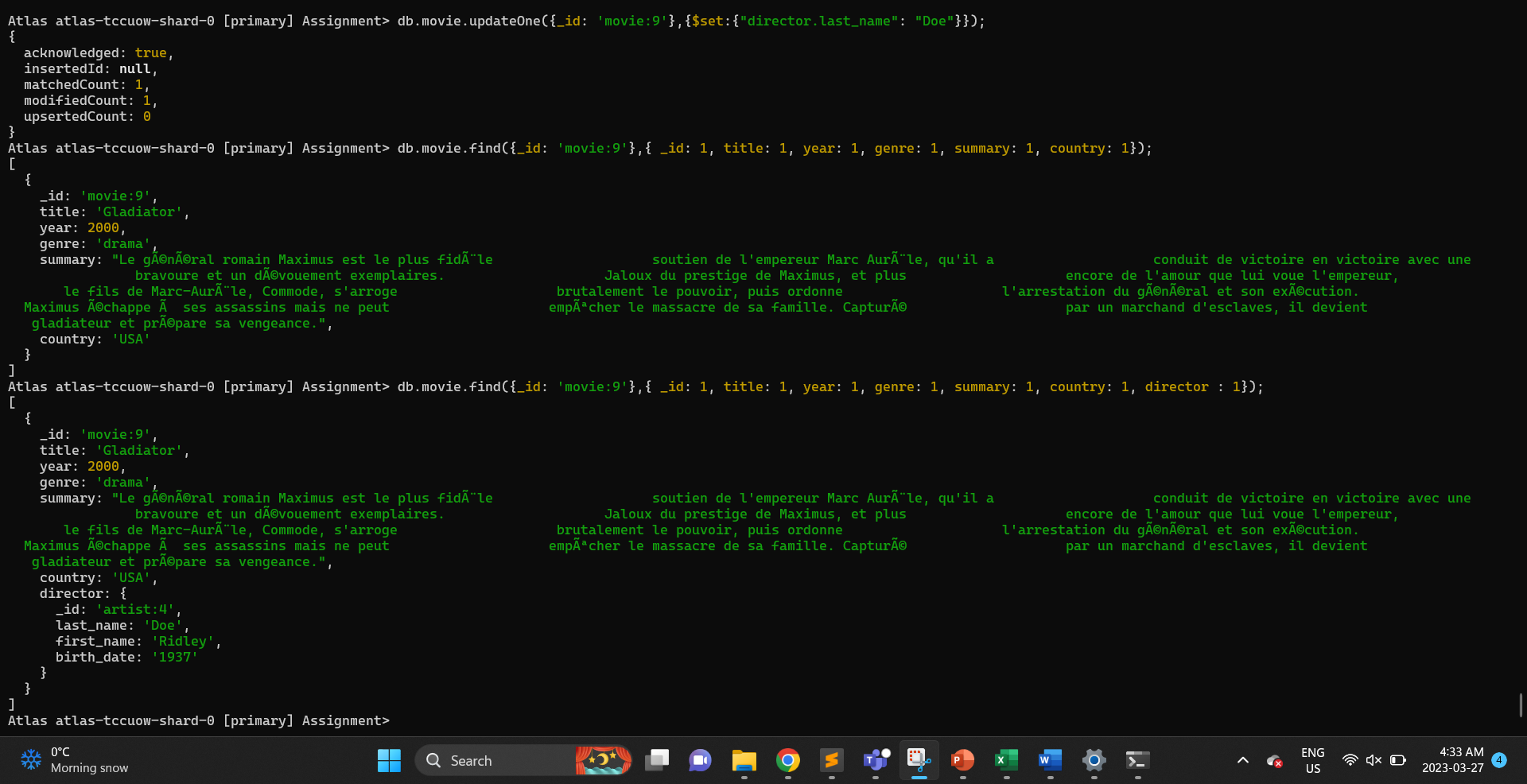
db.movie.updateOne({\_id: 'movie:3'},{$set:{country: "FR"}});



This command will update the country field of the document with \_id equal to movie:3 to "FR".

5. Update the movie with directorâ€™s last\_name is â€œDoeâ€ from "Scott" where \_id is "movie:9".

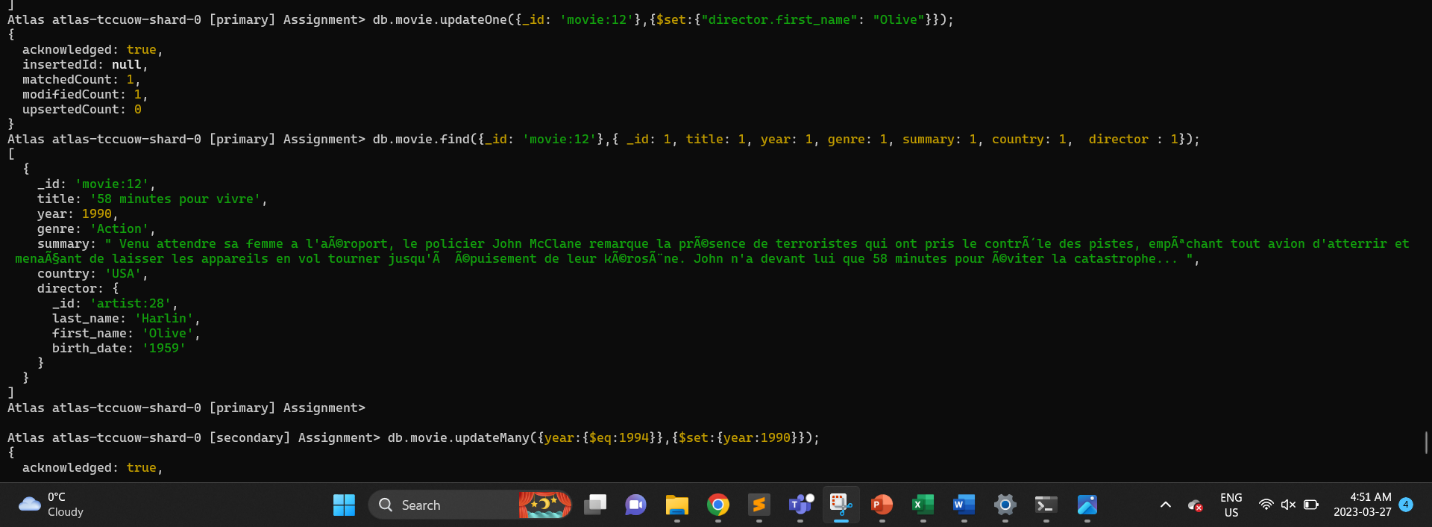
db.movie.updateOne({\_id: 'movie:9'},{$set:{"director.last\_name": "Doe"}});



This command will update the last name of the director in the document with \_id equal to movie:9 to "Doe".

6. Update the movie with directorâ€™s first\_name â€œOliveâ€ from â€œBruceâ€ where \_id is "movie:12". - ERROR in Q.

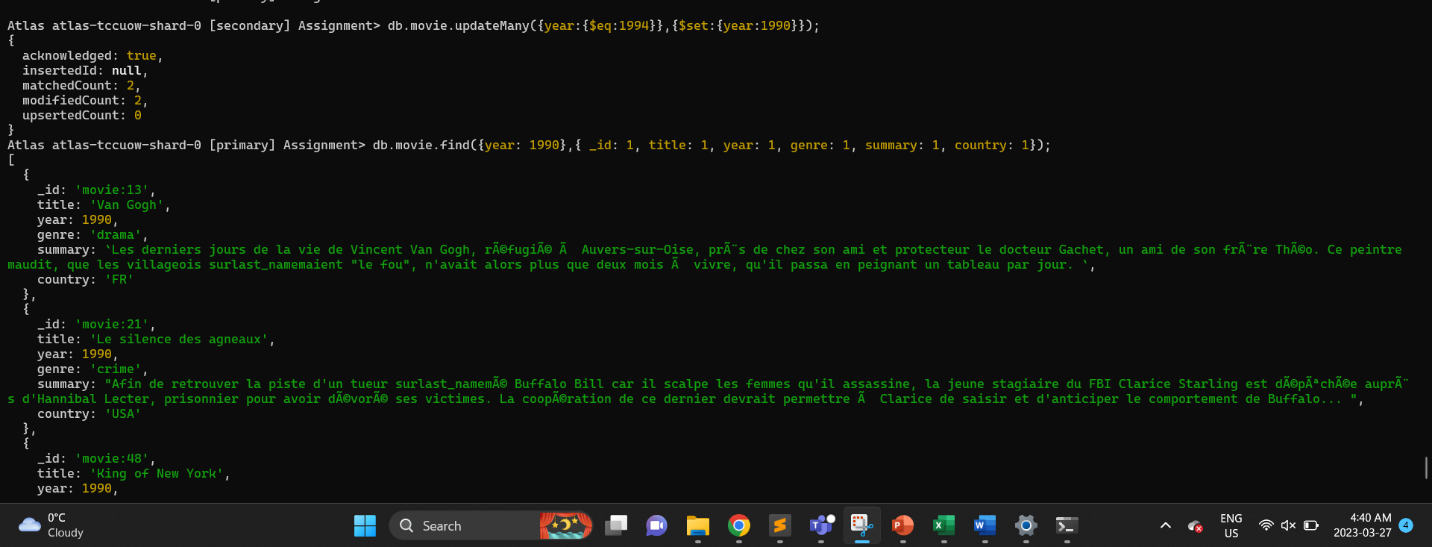
db.movie.updateOne({\_id: 'movie:12'},{$set:{"director.first\_name": "Olive"}});



This command will update the director.first\_name field of the document with \_id equal to movie:12 to "Olive".

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

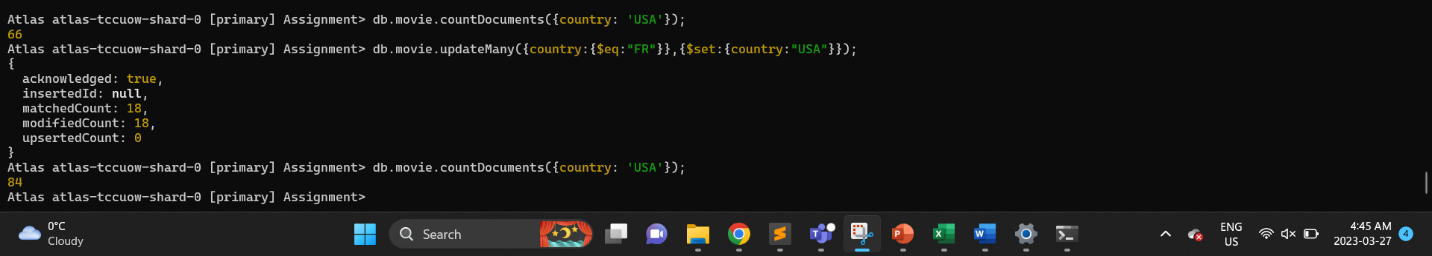
db.movie.updateMany({year:{$eq:1994}},{$set:{year:1990}});



This command will update the year field of all documents with year equal to 1994 to 1990.

8. Update the movies with country â€œUSAâ€ from "FR".

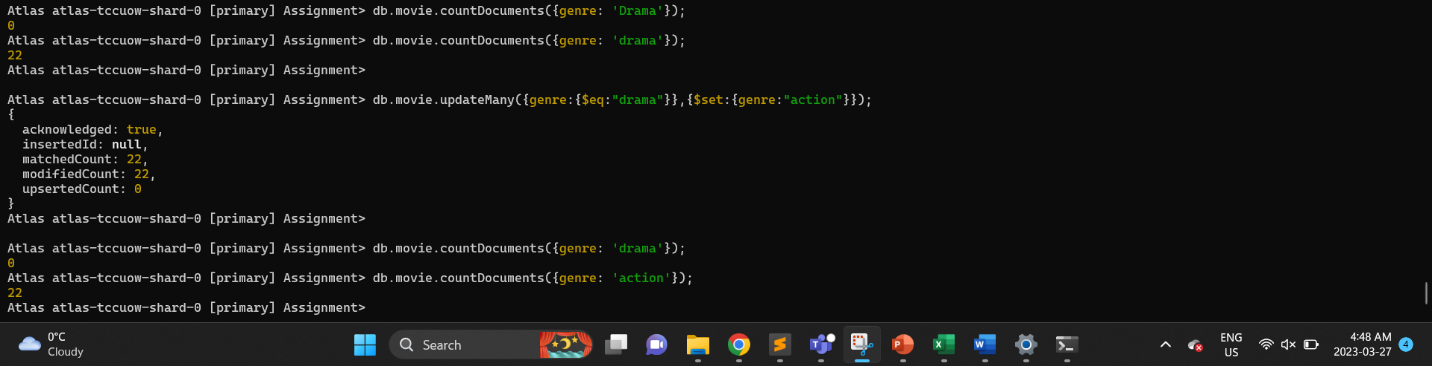
db.movie.updateMany({country:{$eq:"FR"}},{$set:{country:"USA"}});



This command will update the country field of all documents where the country is equal to "FR" to "USA".

9. Update the movies with genre â€œDramaâ€ from â€œActionâ€.

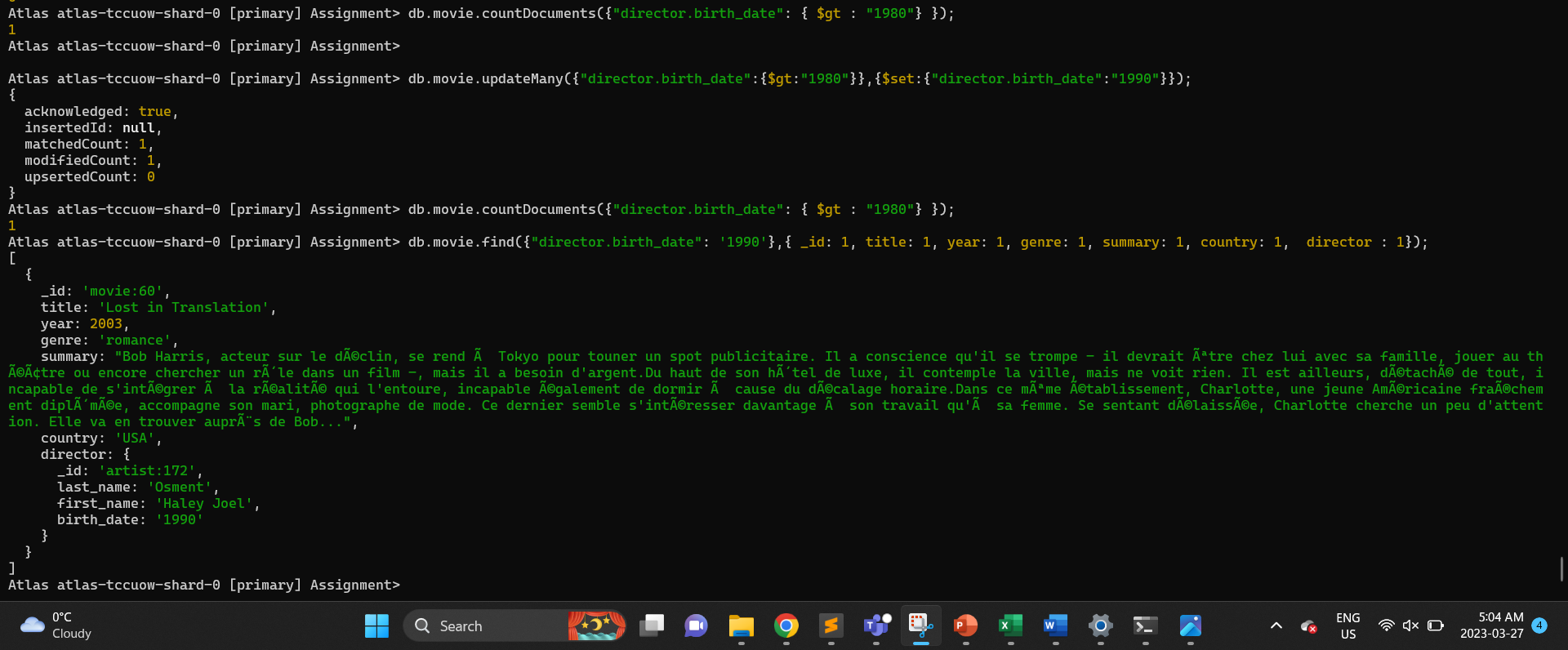
db.movie.updateMany({genre:{$eq:"drama"}},{$set:{genre:"action"}});



This command will update the genre field of all documents with genre equal to "drama" to "action".

10. Update the movies with directorâ€™s birth\_date to 1990 where birth date older than 1980.

db.movie.updateMany({"director.birth\_date":{$gt:"1980"}},{$set:{"director.birth\_date":"1990"}});



This command would update the birth\_date of directors born year after 1980 to 1990.