MongoDB – Querying Movies Database

“Querying Movies Database” using MongoDB is an assignment I did for my Data Collection and Curation course in the Big Data Analytics program at Georgian College. The aim of this project was to showcase my understanding of MongoDB and querying data using a non-relational database. I was provided a txt file with a movie database to import into MongoDB and was expected to perform a set of tasks assigned by my professor to show my competency using MongoDB.

Table of Contents

Perform the following Mongo DB Queries 3

A. Write the following script to fetch the document on movie collection. 5

1. Fetch all the document from movie collection. 5

2. Fetch all the document with movies released in the 90s 7

3. Fetch all the document with movies released before the year 2000 or after 2010. 8

4. Fetch all the document with movie “crime" or "drama" genre. 9

B. Write the script to update document on movie collection. 10

1. Update the movie with title “The Titanic” from “Titanic” where _id is "movie:3". 10

2. Update the movie with country “FR” from "USA" where _id is “movie:3”. 11

3. Update the movie with director’s last_name is “Doe” from "Scott" where _id is "movie:9". 12

4. Update the movies with year 1990 where year is 1994. 13

C. Write the script to Search a string in the document on movie collection. 14

1. Fetch all the movies genre as “drama”. 14

2. Fetch all the movies title with “Titanic” 15

3. Fetch all the movies director’s first_name as “John”. 16

4. Fetch all the movies title start and end with "t". 17

D. Write the script to create the indexes on movie collection. 18

1. Create a single index with multiple attributes (title by ascending order and genre by descending order) on movie. 18

2. Create a wildcard Index on director first_name order by ascending order on movie collection. 20

3. Create a unique index on country attribute by ascending order on movie collection. 21

4. Create a multikey Index on actors first_name order by ascending order on movie collection.... 22

E. Write the script to delete document on movie collection. 23

1. Delete the movie with title "Spider-Man". 23

2. Delete the movies which is released in 1990. 24

3. Delete the movie which is from country FR and year 1975. 25

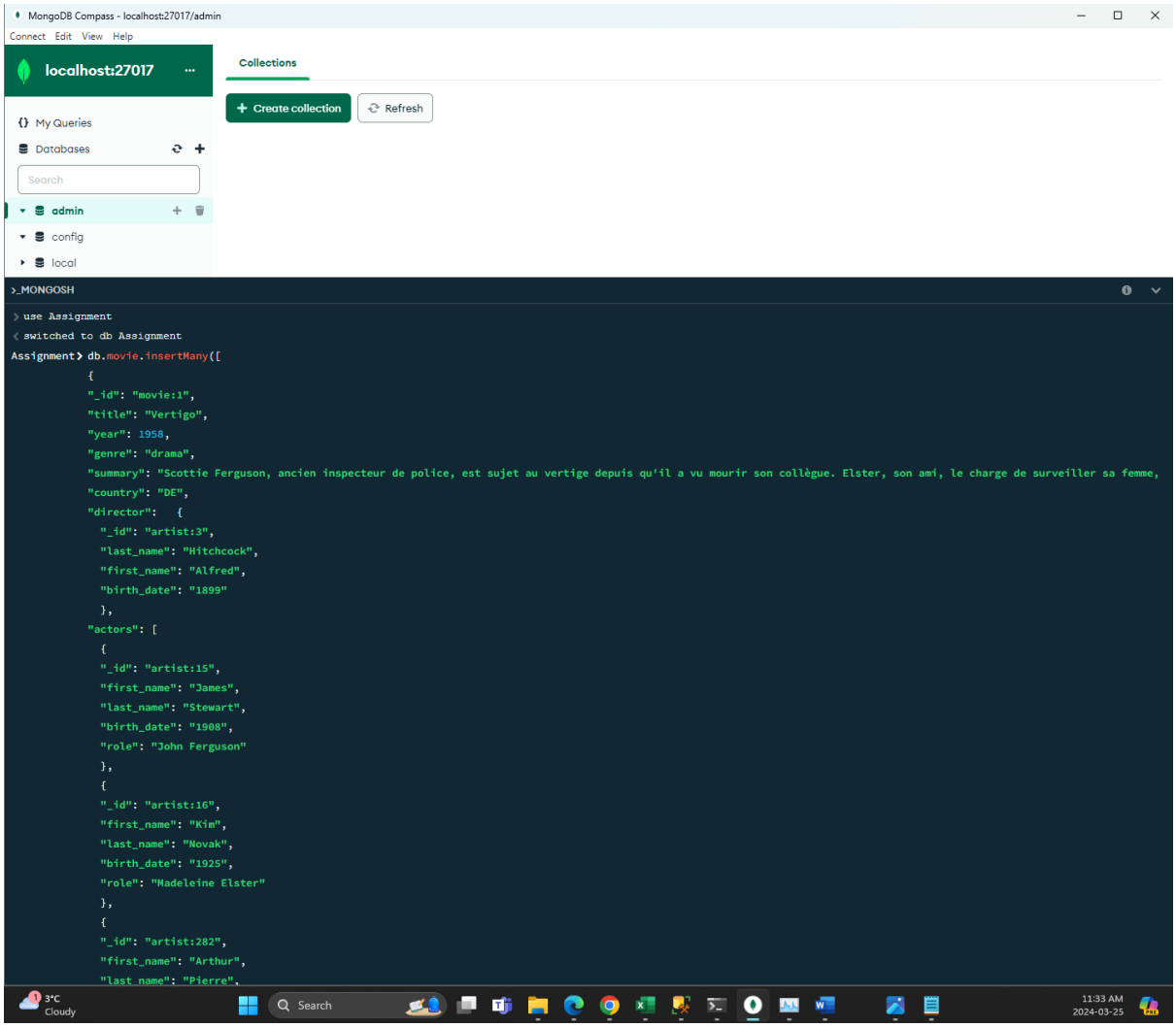
4. Delete the movies with genre is drama. 26

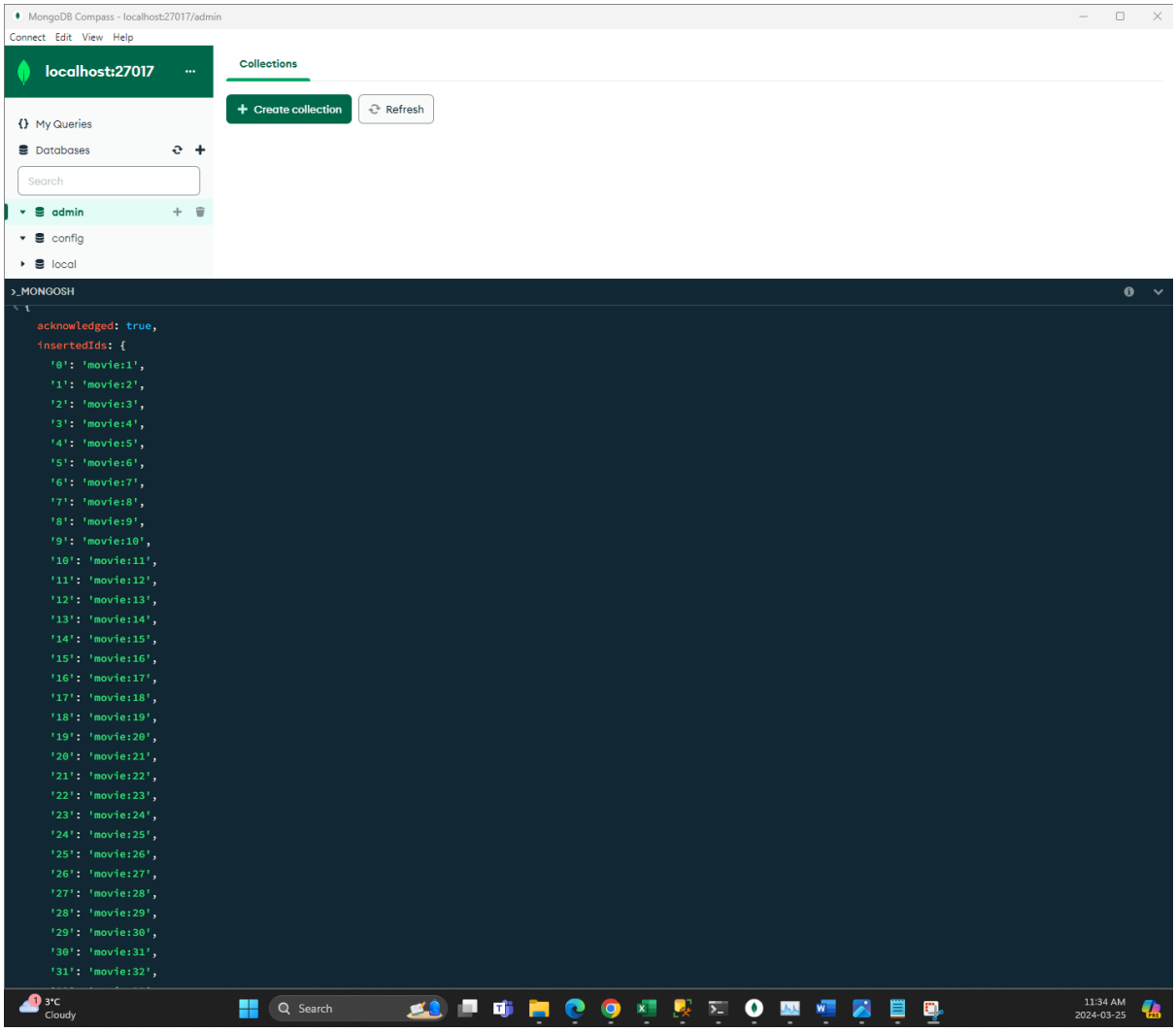
Perform the following Mongo DB Queries

Create an Assignment database.

Copy all the script from movie_Script.txt file and execute on the Mongo DB server under Assignment database.

Solution:

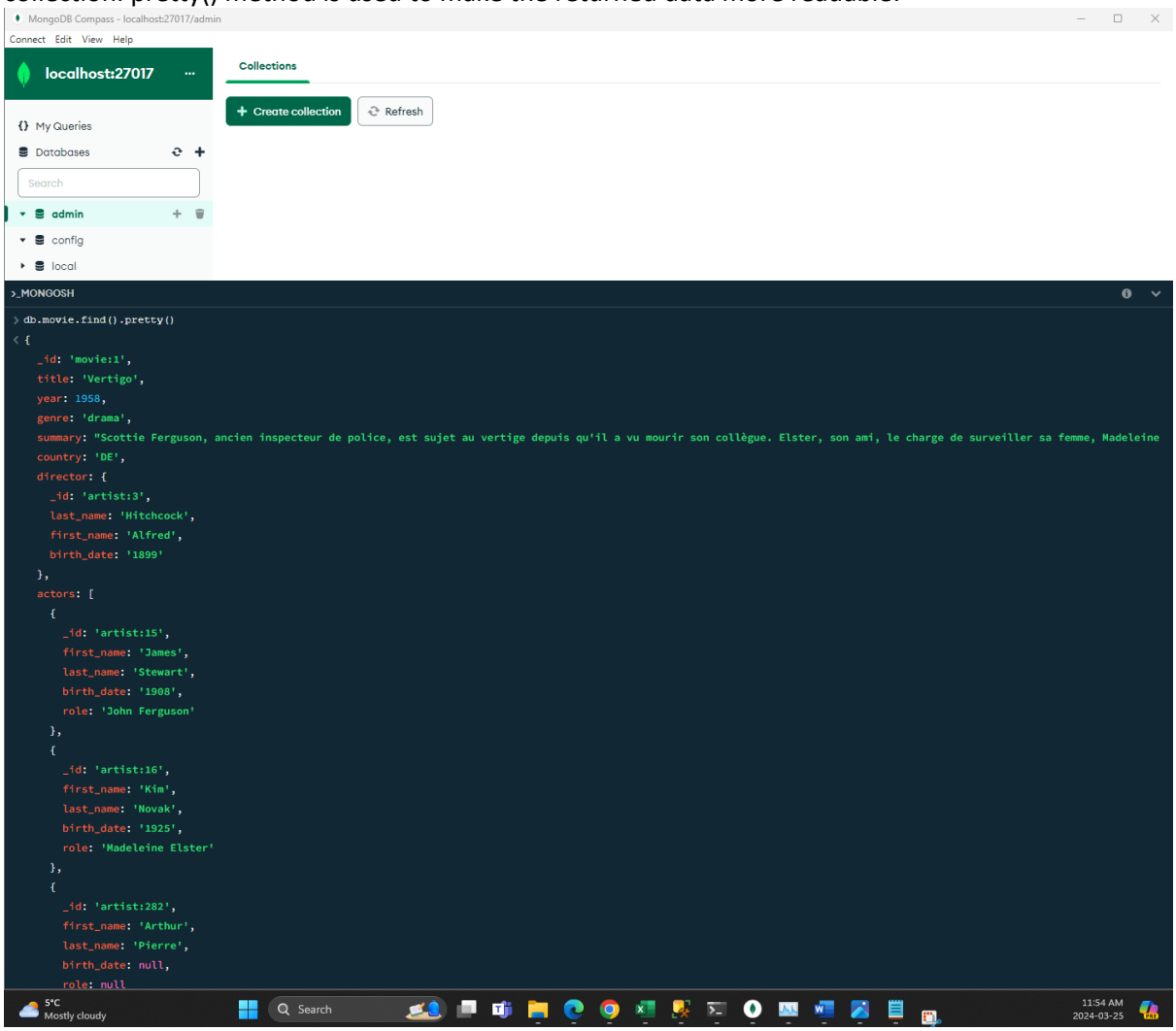


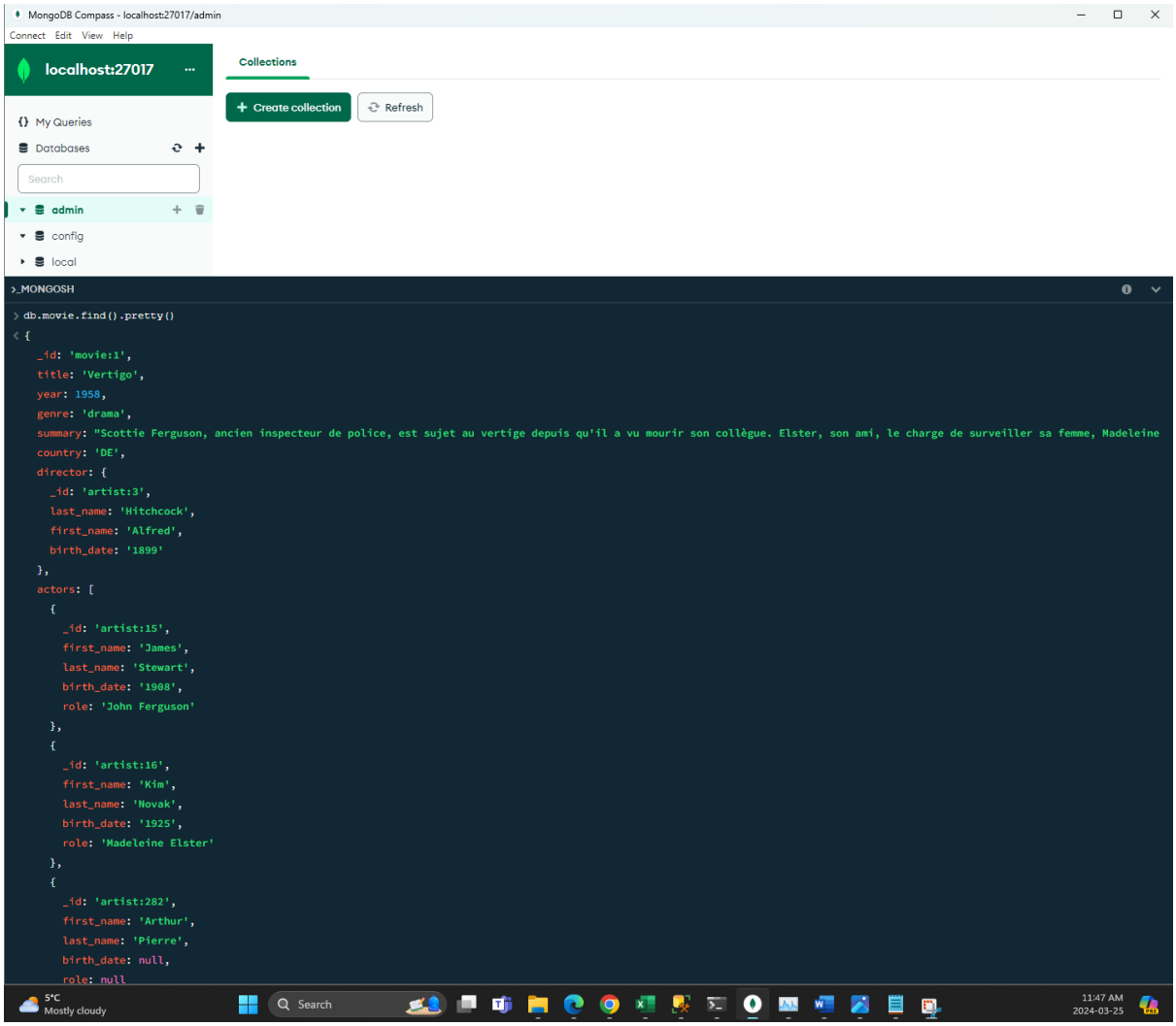

MongoDB Compass - localhost:27017/admin
Connect Edit View Help
localhost:27017
My Queries
Databases
Search
admin
config
local
Collections
Create collection
Refresh
>_MONGOSH
{
acknowledged: true,
insertedIds: {
'0': 'movie:1',
'1': 'movie:2',
'2': 'movie:3',
'3': 'movie:4',
'4': 'movie:5',
'5': 'movie:6',
'6': 'movie:7',
'7': 'movie:8',
'8': 'movie:9',
'9': 'movie:10',
'10': 'movie:11',
'11': 'movie:12',
'12': 'movie:13',
'13': 'movie:14',
'14': 'movie:15',
'15': 'movie:16',
'16': 'movie:17',
'17': 'movie:18',
'18': 'movie:19',
'19': 'movie:20',
'20': 'movie:21',
'21': 'movie:22',
'22': 'movie:23',
'23': 'movie:24',
'24': 'movie:25',
'25': 'movie:26',
'26': 'movie:27',
'27': 'movie:28',
'28': 'movie:29',
'29': 'movie:30',
'30': 'movie:31',
'31': 'movie:32',
3°C
Cloudy
Search
11:34 AM
2024-03-25

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

1. Fetch all the document from movie collection.

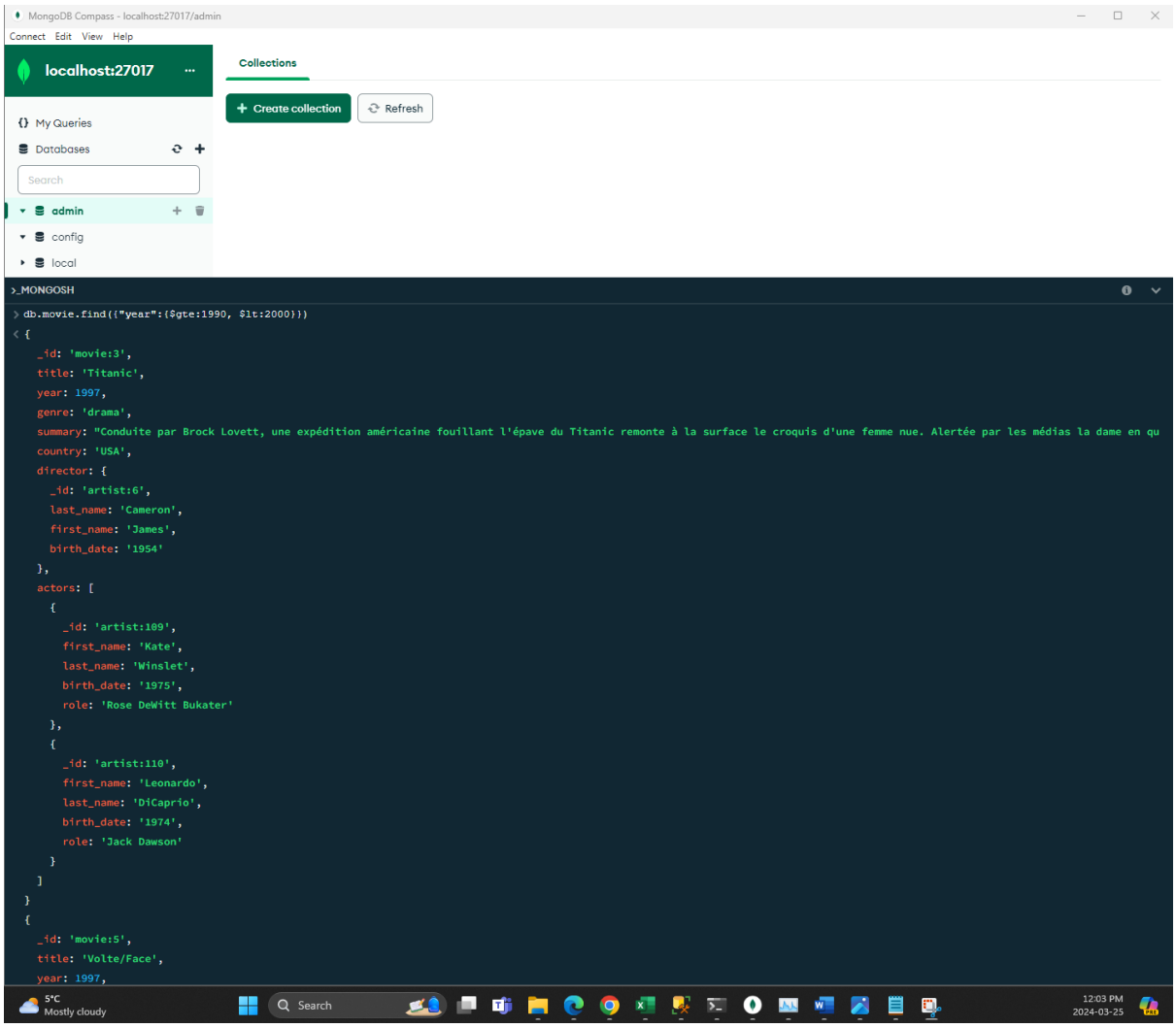
Solution: We use command db.movie.find().pretty() – here movie represents the collection we created to insert the movie data and we use the find() method to return all documents in this collection. pretty() method is used to make the returned data more readable.




MongoDB Compass - localhost:27017/admin
Connect Edit View Help
localhost:27017
My Queries
Databases
Search
admin
config
local
Collections
Create collection
Refresh
>_MONGOSH
db.movie.find().pretty()
{
_id: 'movie:1',
title: 'Vertigo',
year: 1958,
genre: 'drama',
summary: "Scottie Ferguson, ancien inspecteur de police, est sujet au vertige depuis qu'il a vu mourir son collègue. Elster, son ami, le charge de surveiller sa femme, Madeleine
country: 'DE',
director: {
_id: 'artist:3',
last_name: 'Hitchcock',
first_name: 'Alfred',
birth_date: '1899'
},
actors: [
{
_id: 'artist:15',
first_name: 'James',
last_name: 'Stewart',
birth_date: '1908',
role: 'John Ferguson'
},
{
_id: 'artist:16',
first_name: 'Kim',
last_name: 'Novak',
birth_date: '1925',
role: 'Madeleine Elster'
},
{
_id: 'artist:282',
first_name: 'Arthur',
last_name: 'Pierre',
birth_date: null,
role: null
5°C
Mostly cloudy
Search
11:47 AM
2024-03-25

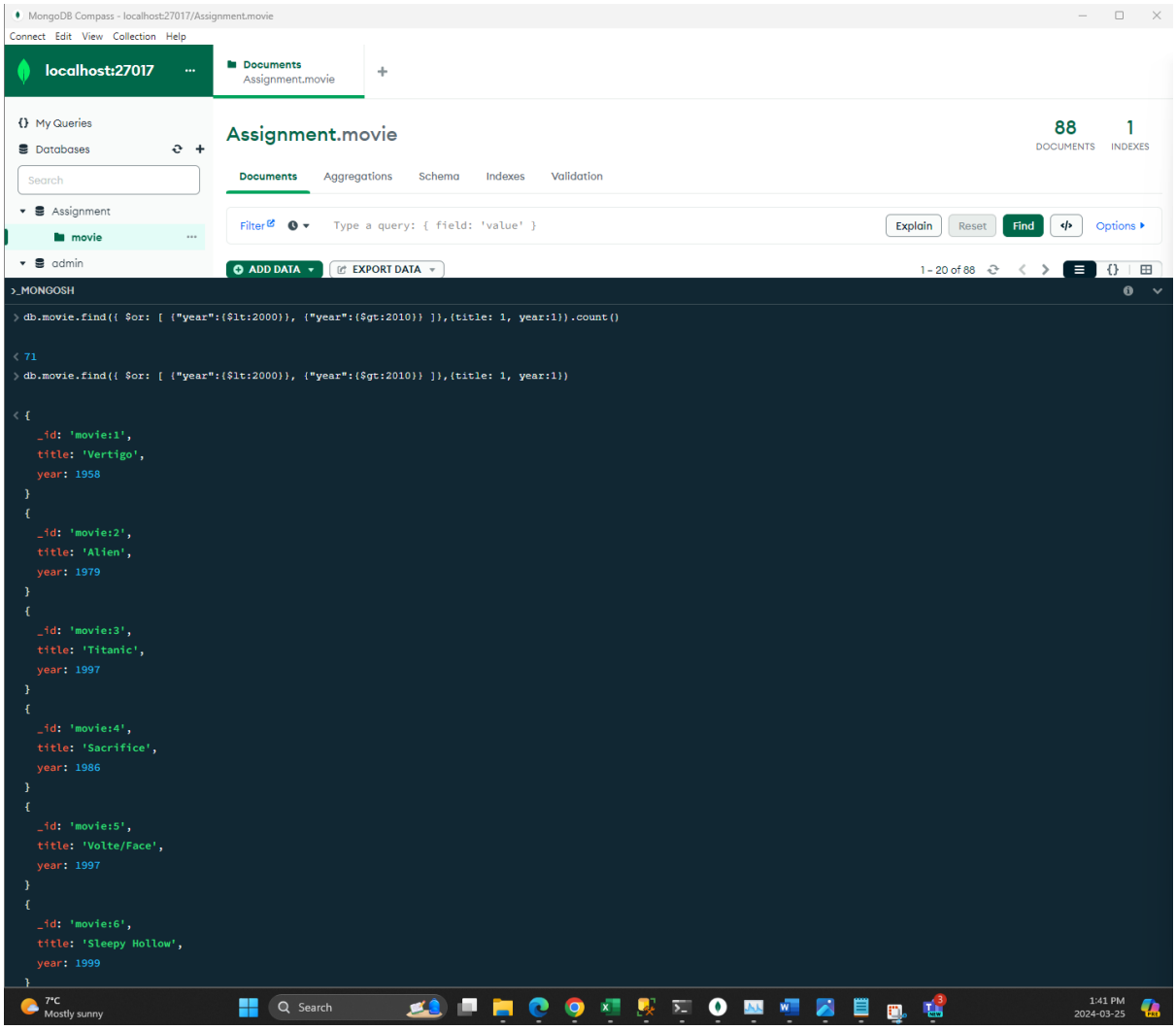
2. Fetch all the document with movies released in the 90s.

Solution: We use command db.movie.find({"year":{$gte:1990, $lt:2000}}) – To get the documents that follow our criteria, we pass arguments through the find() query method such as $gte, which means greater than or equal, and $lt, which means less than. We specify the filed – “year” and use $gte and $lt to get the movies released in 1990s.



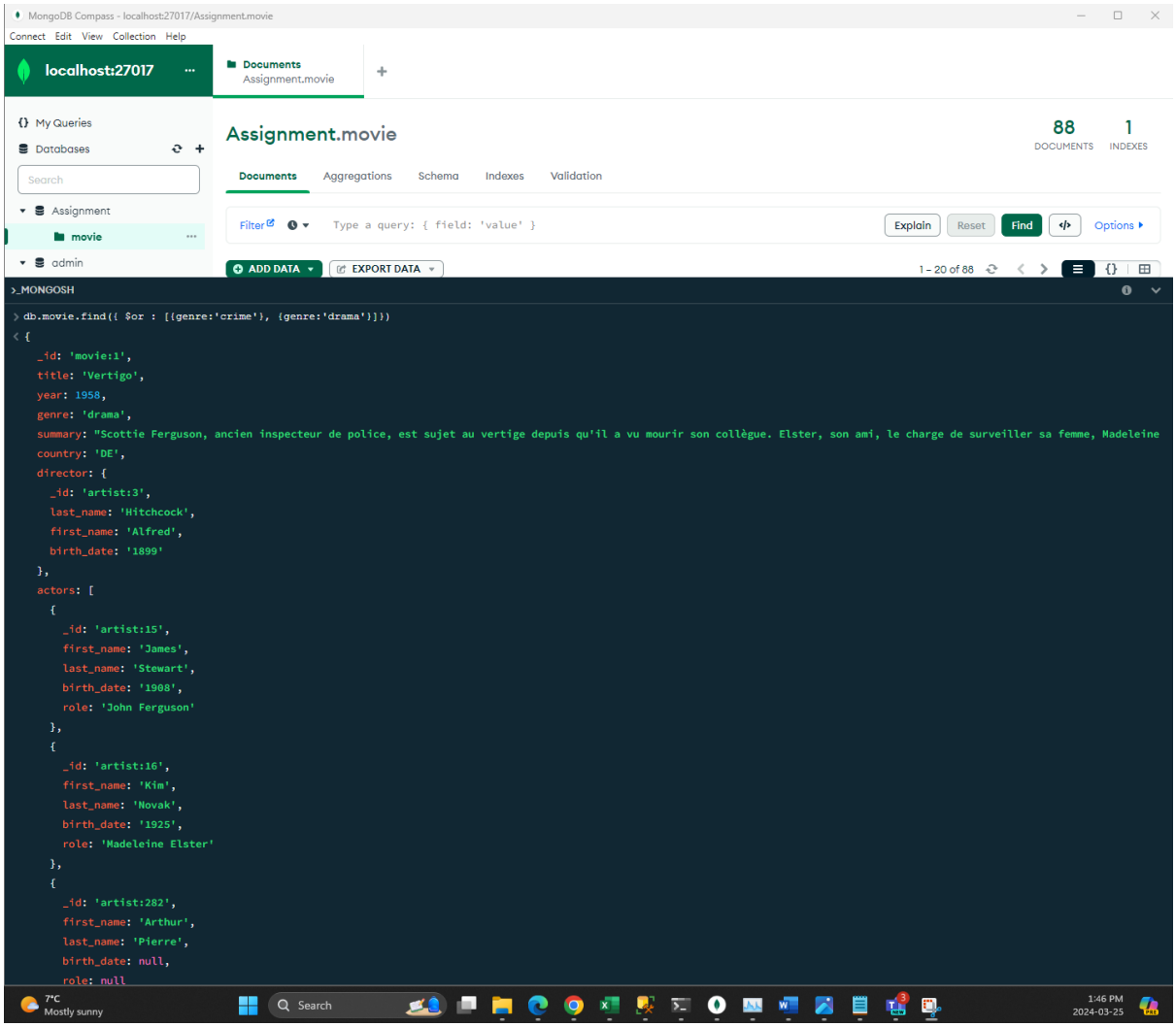
3. Fetch all the document with movies released before the year 2000 or after 2010.

Solution: We use command db.movie.find({ $or: [{"year":{$lt:2000}}, {"year":{$gt:2010}}]},{title: 1, year:1})– It is similar to what we use for movies in the 1990s but we use $or to match documents that are either $lt:2000 or $gt:2010. We could also use Projection Query to return only the Title and the year in this one to only get the title and year as shown here, if needed.



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

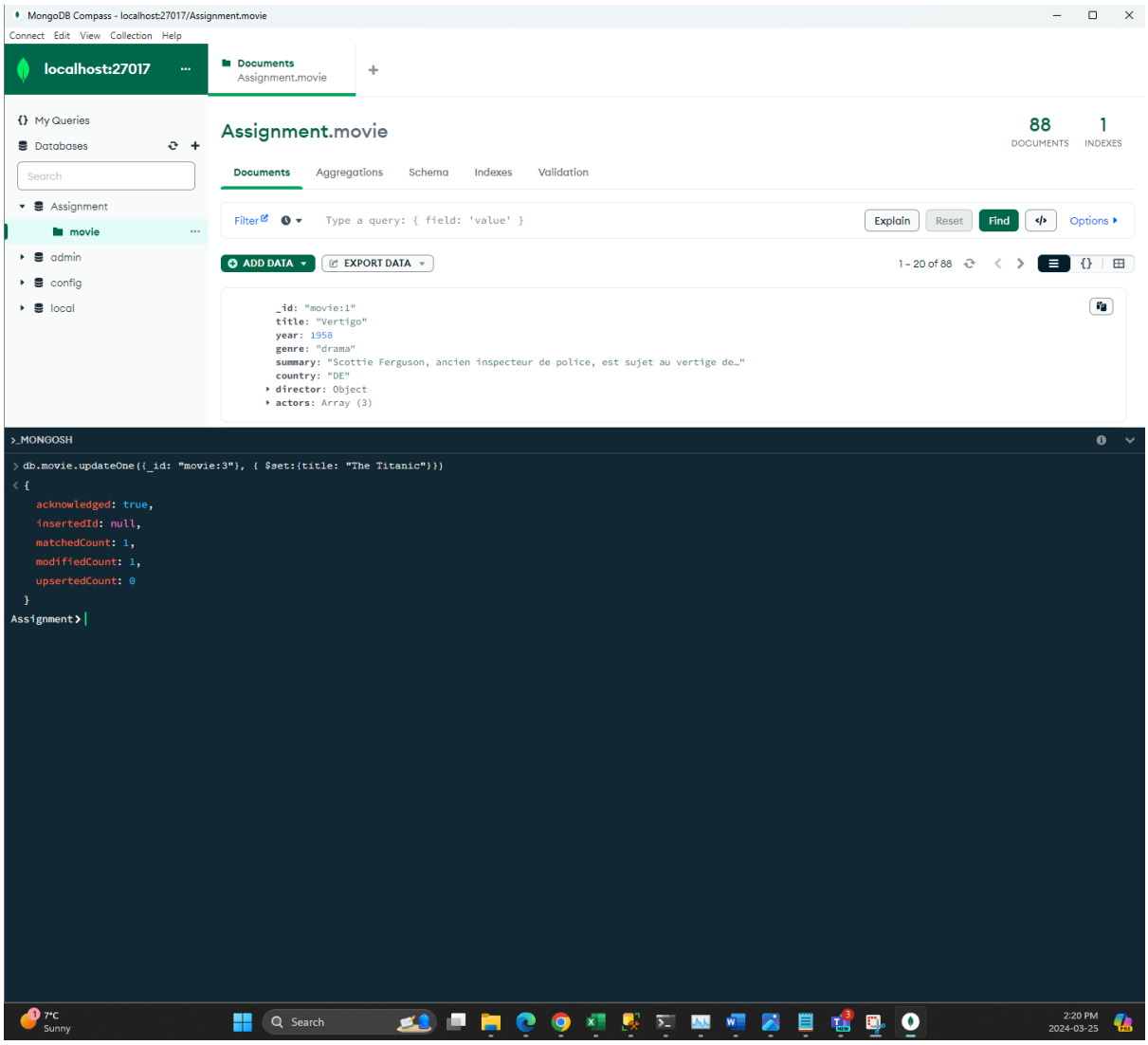
Solution: we use command db.movie.find({ $or : [{genre:'crime'}, {genre:'drama'}]}) – Its similar to our previous solution where we use $or but here we switch the ‘year’ with ‘genre’ and specify crime or drama per our requirement.



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

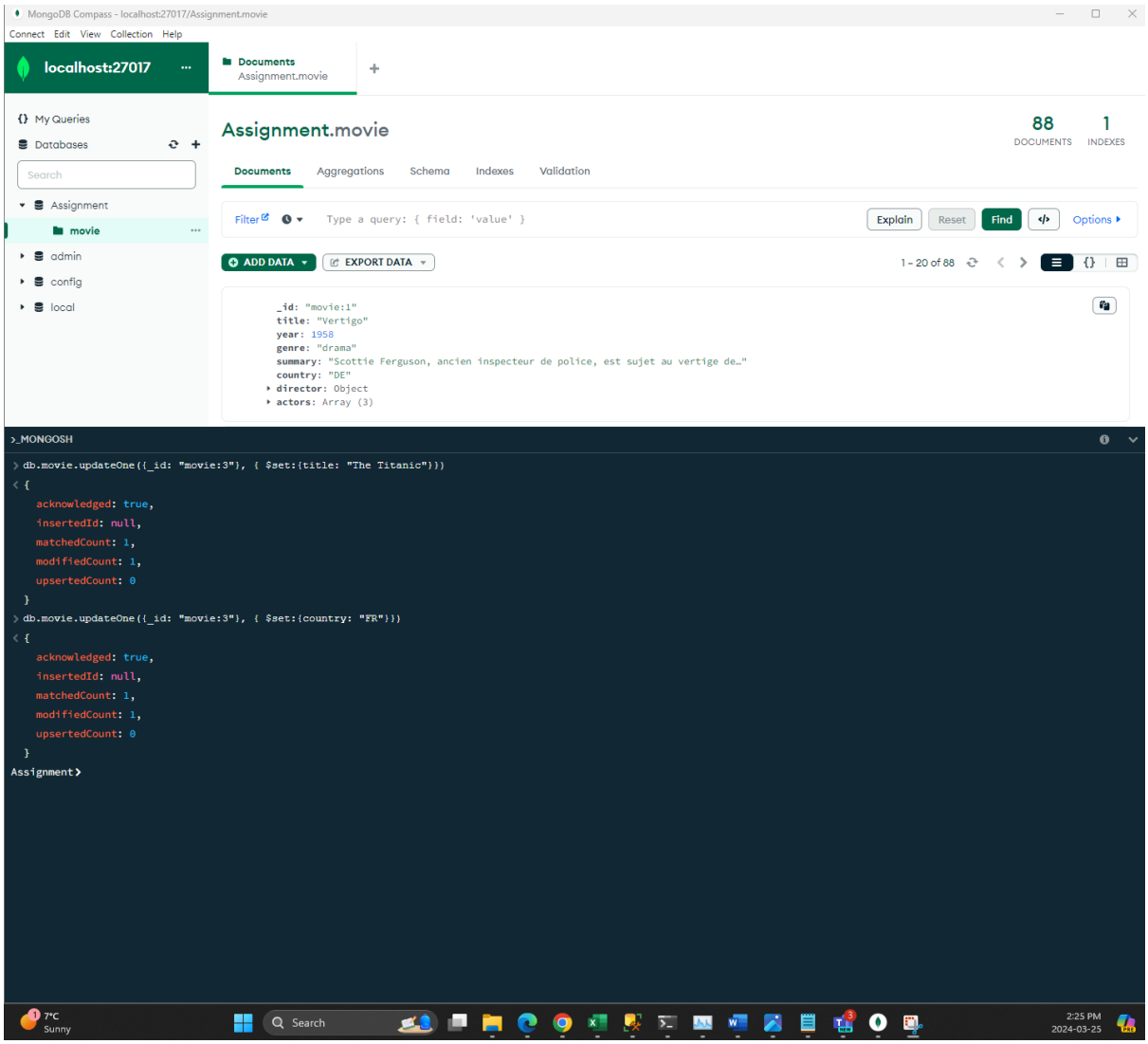
1. Update the movie with title “The Titanic” from “Titanic” where _id is "movie:3".

Solution: we use command db.movie.updateOne({_id: “movie:3”}, { $set:{ title: “The Titanic”}}) – we pass our selection criteria that is _id: “movie:3” and the we pass our new value that is the title.



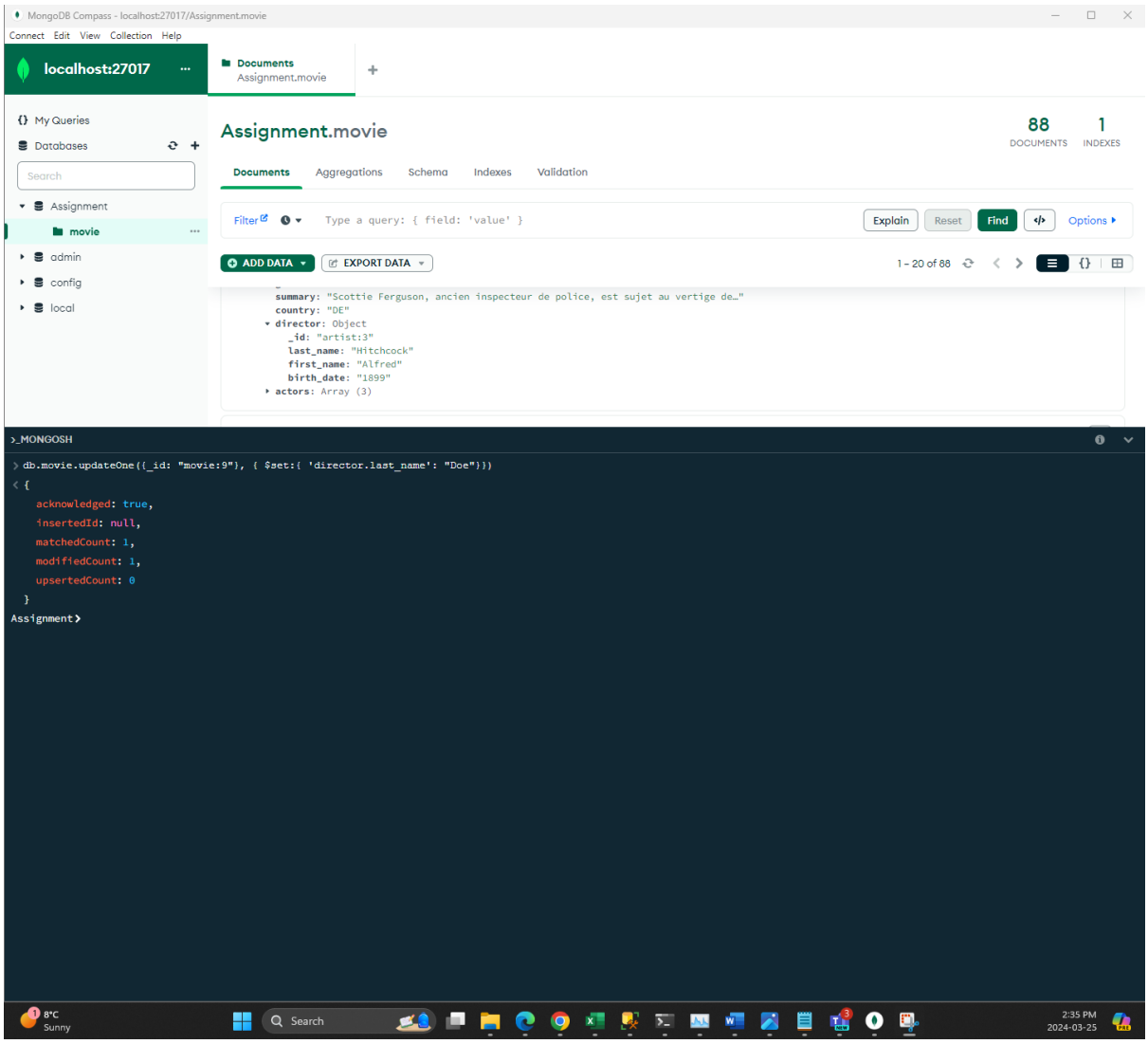
2. Update the movie with country “FR” from "USA" where _id is “movie:3”.

Solution: we use command db.movie.updateOne({_id: “movie:3”}, { $set:{ country: “FR”}}) – It is similar to our last query but in this one we update the country name.



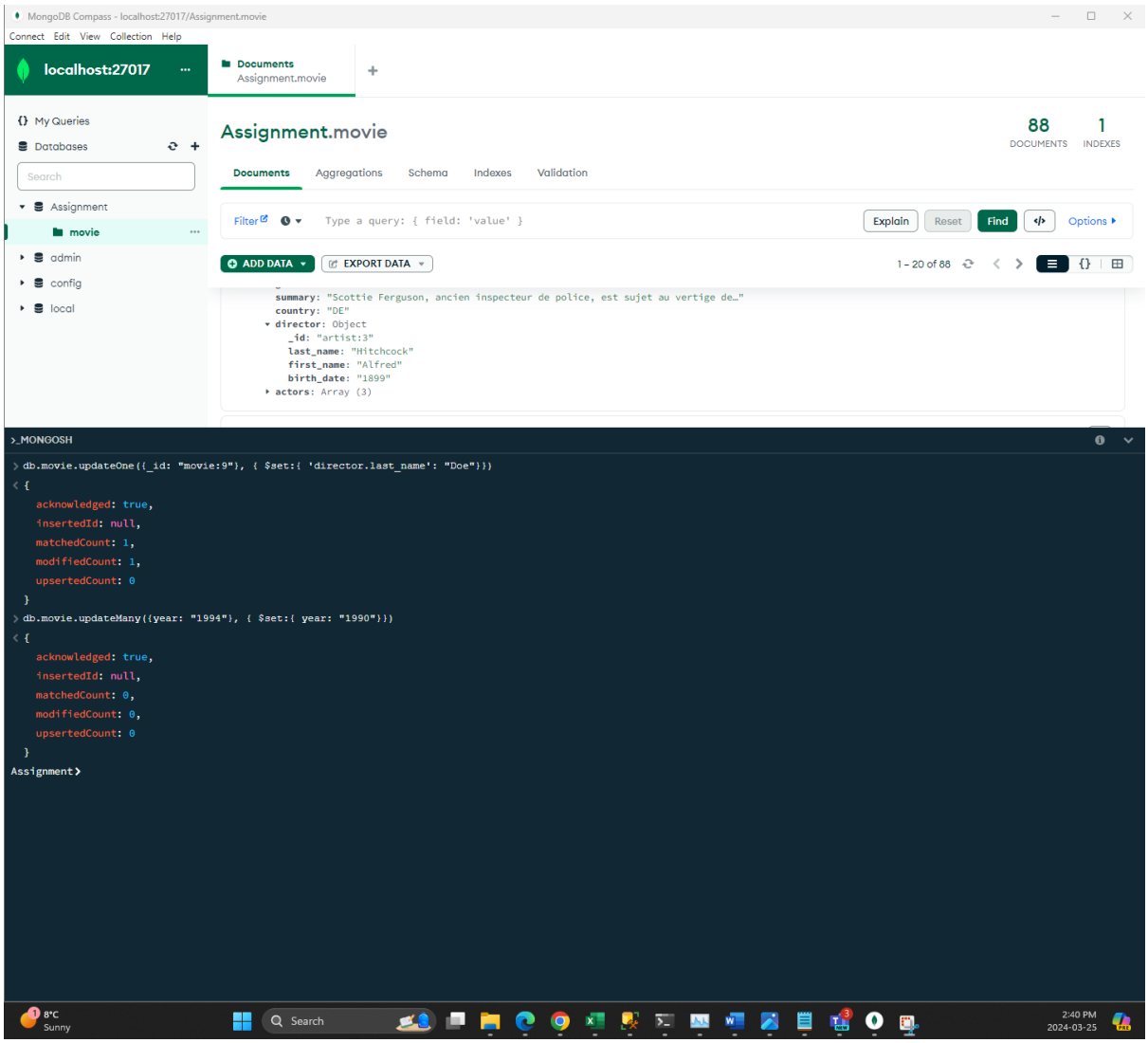
3. Update the movie with director's last_name is "Doe" from "Scott" where _id is "movie:9".

Solution: we use command db.movie.updateOne({_id: "movie:9"}, { $set:{ 'director.last_name': "Doe"}}) – It is similar to our last query but in this one we update the director name.



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

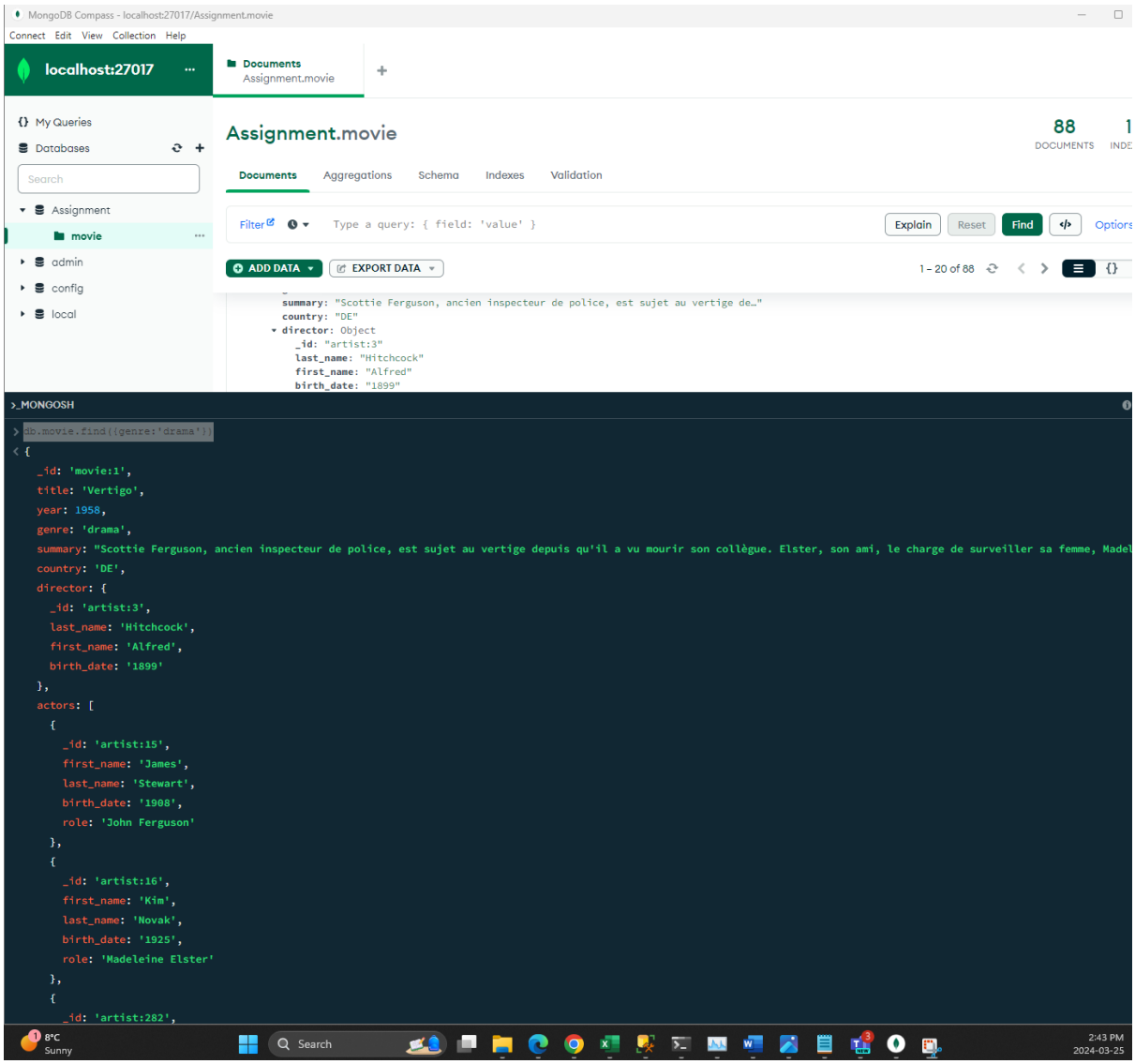
Solution: db.movie.updateMany({year: "1994"}, { $set:{ year: "1990"}}) – we use the updateMany() method and pass our selection criteria {year: "1994"} and updates i.e. { year: "1990"}.



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

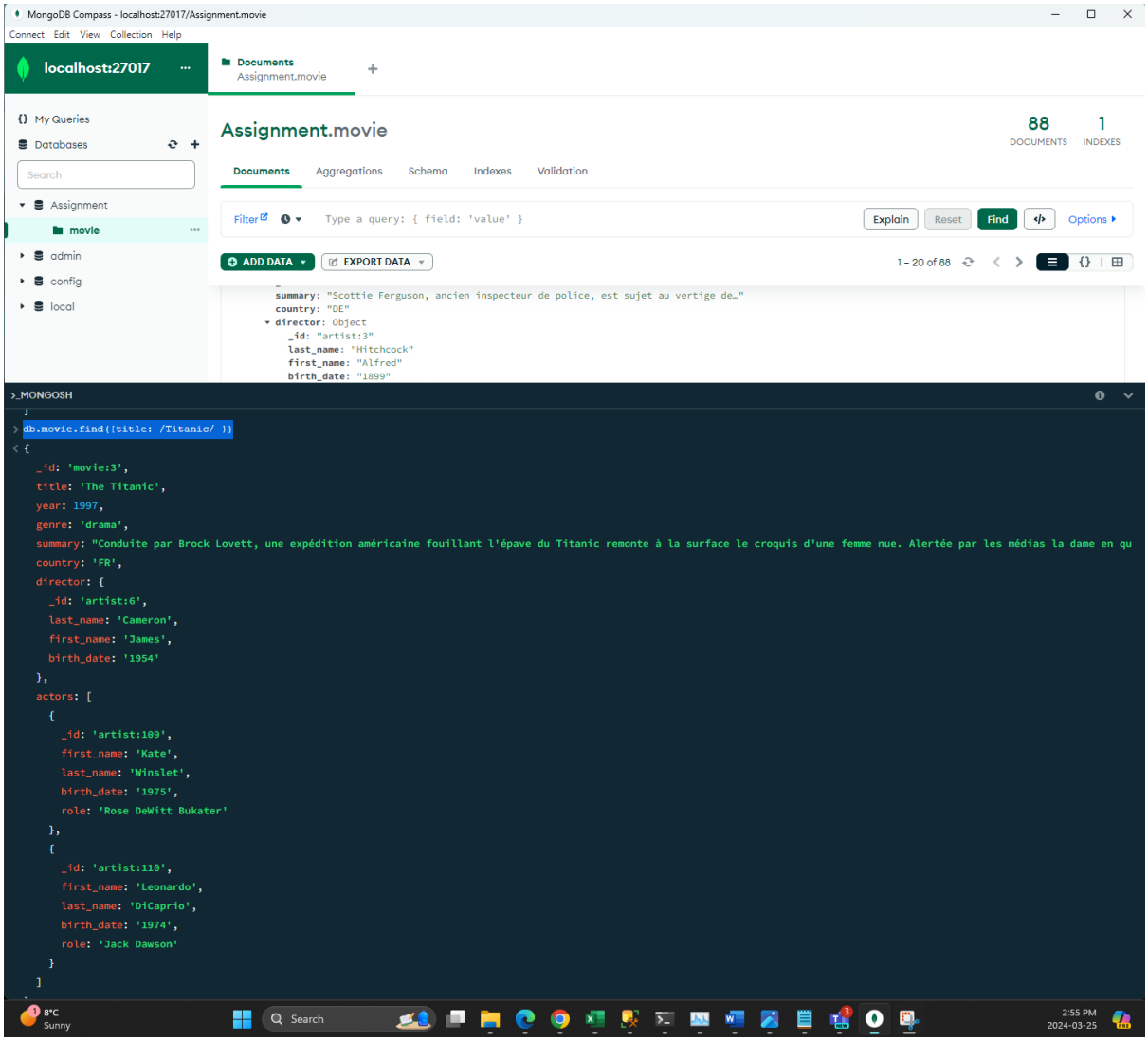
1. Fetch all the movies genre as “drama”.

Solution: db.movie.find({genre:'drama'}) – we use the find method and pass the genre criteria through it to get all drama movies.



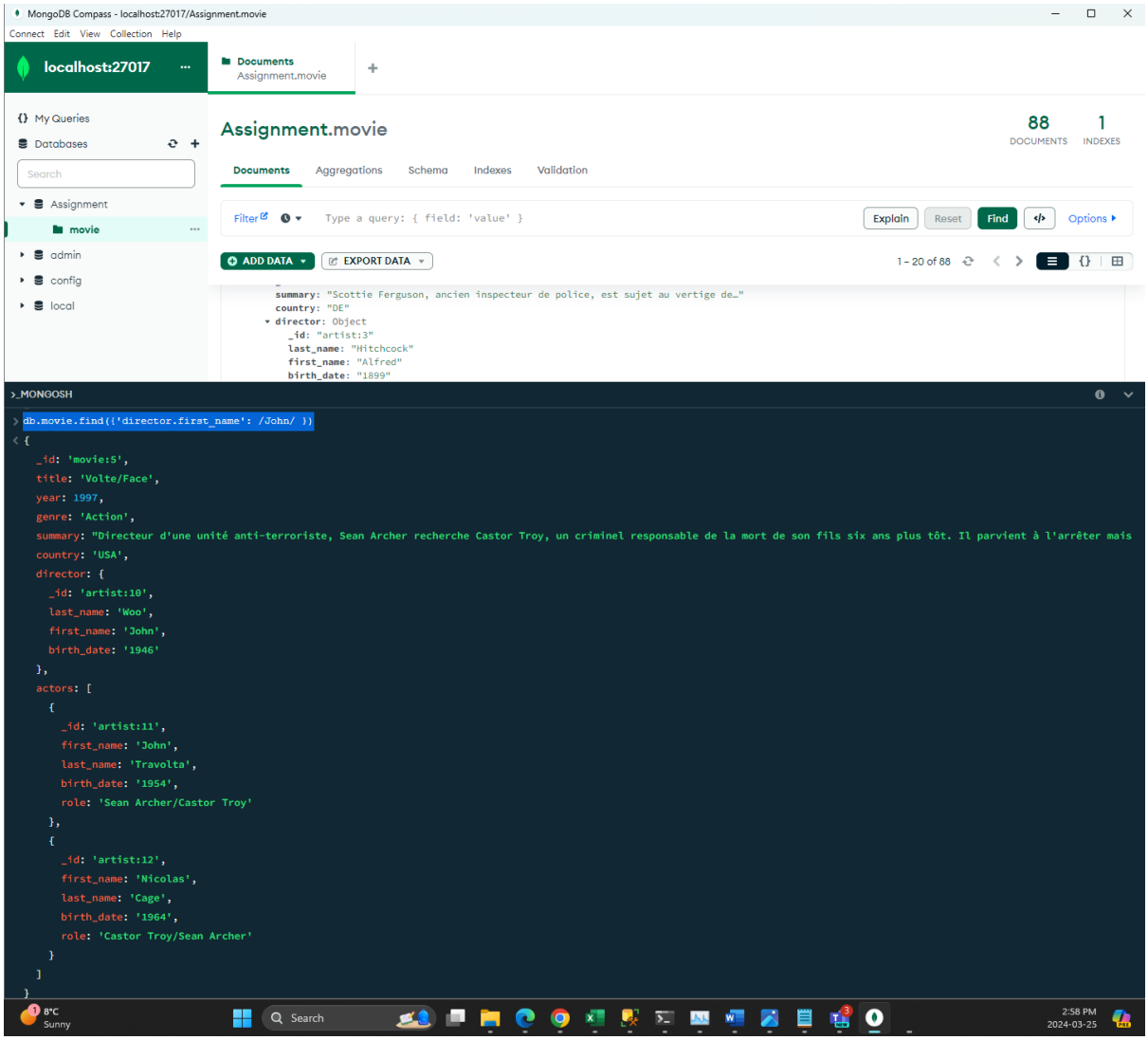
2. Fetch all the movies title with “Titanic”.

Solution: db.movie.find({title: /Titanic/ }) – we use this command to search for movie with title ‘Titanic’ as we previously renamed it to ‘The Titanic’ so {title: ‘Titanic’} would not work. We can also use $regex to find any words that contain the word ‘Titanic’ in the title – {title: {$regex: ‘Titanic’}}



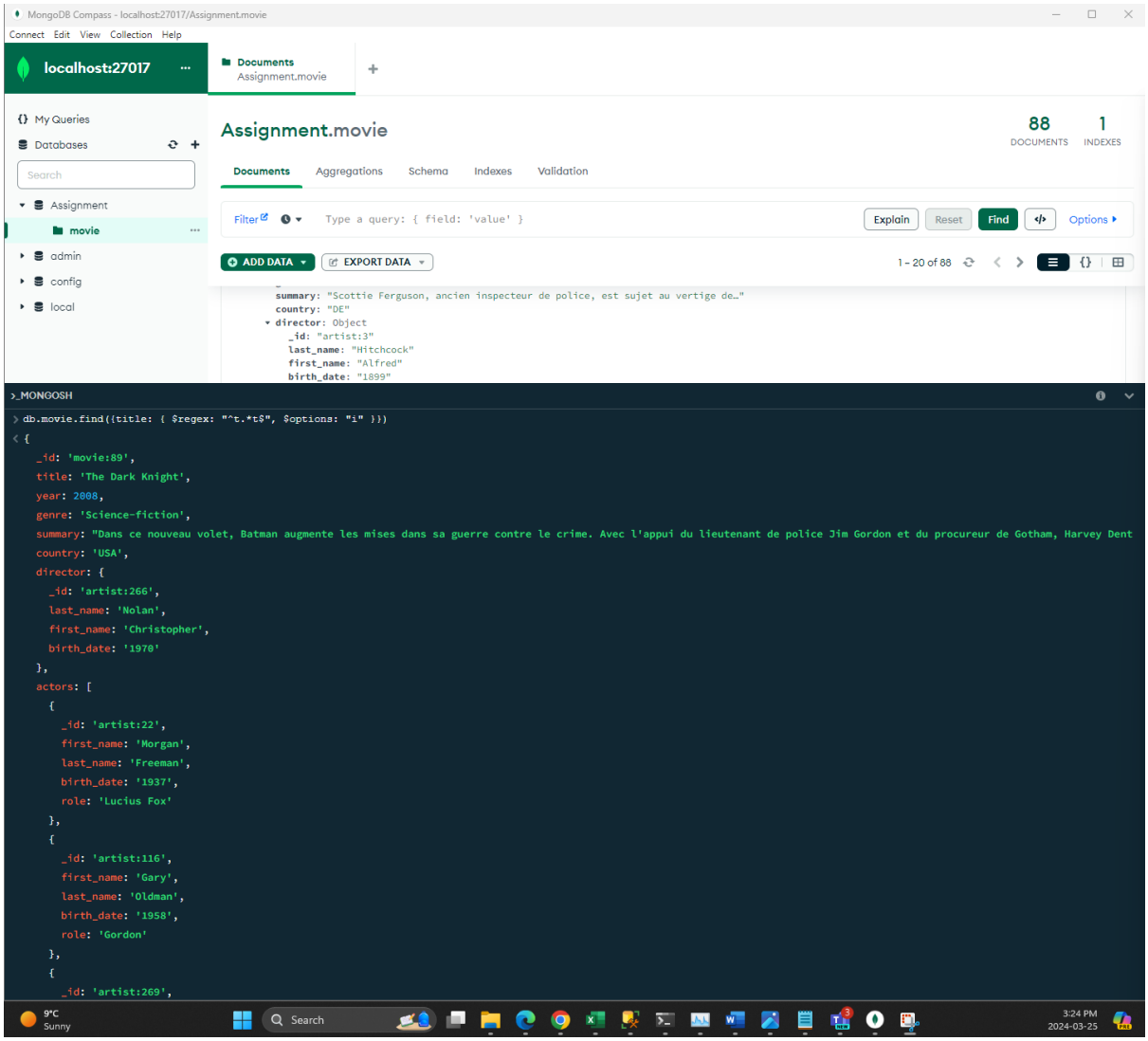
3. Fetch all the movies director's first_name as "John".

Solution: db.movie.find({'director.first_name': /John/ }) – similar to our previous solution, we only replace our criteria from title to 'director.first_name'.



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

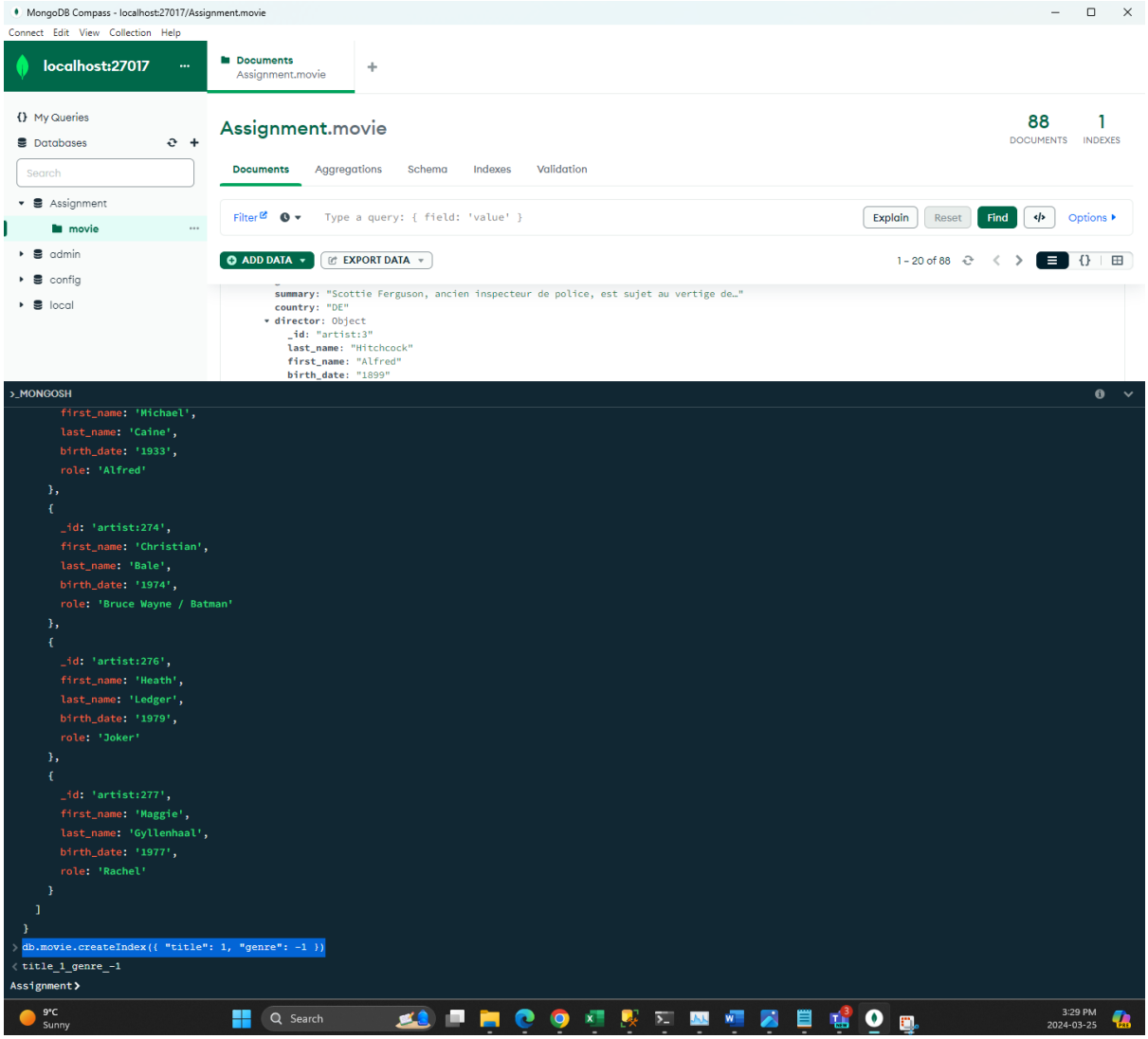
Solution: db.movie.find({title: { $regex: "^t.*t$", $options: "i" }}) – we use $regex to perform this task, '^t means the string starts with t, '.*' allows any characters in the middle and 't$' means it ends with a t. we use $options to ignore case.



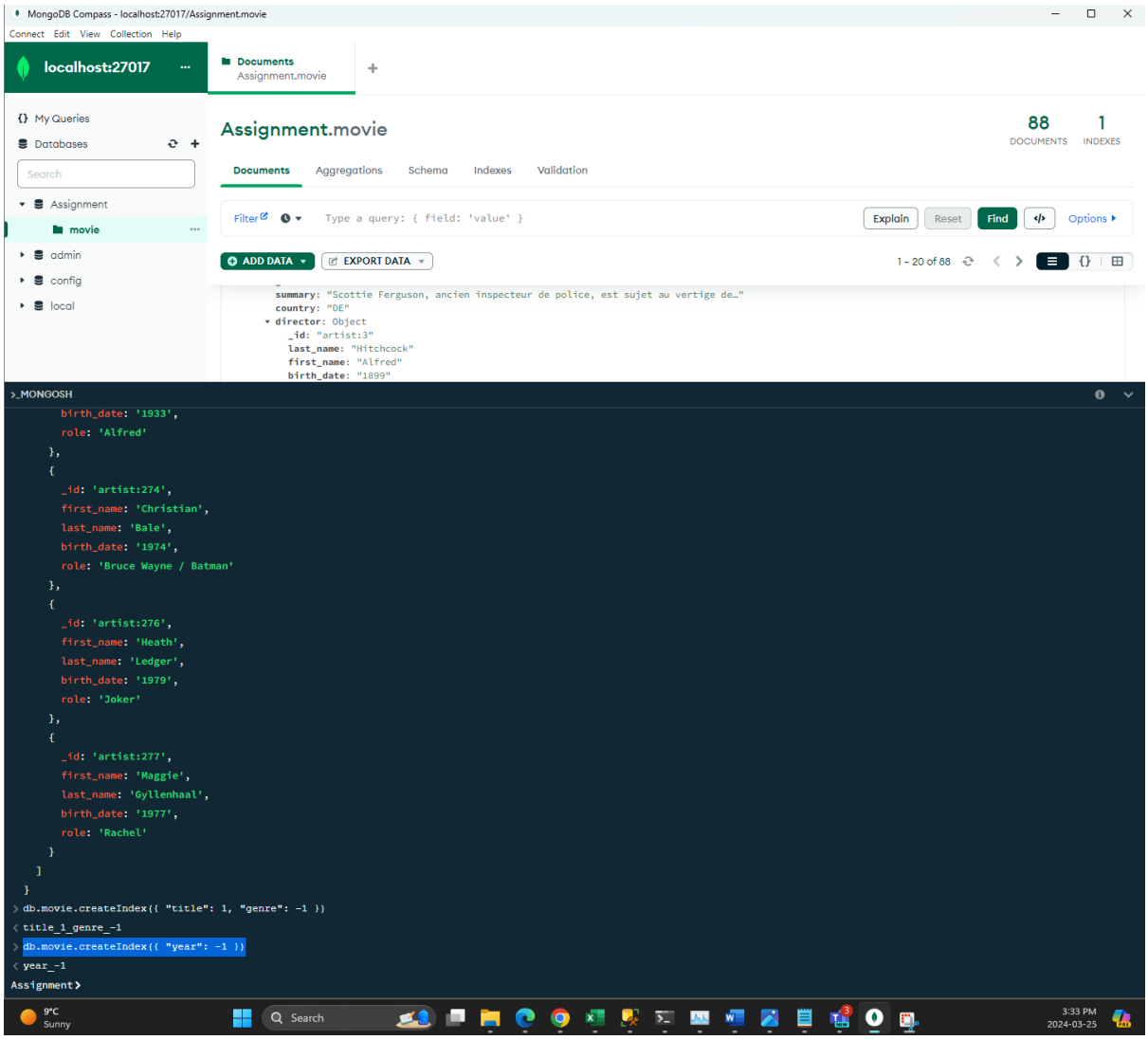
D. Write the script to create the indexes on movie collection.

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

Solution: db.movie.createIndex({ "title": 1, "genre": -1 }) – we use createIndex() method and specify the order of attributes for the index. ‘title’: 1 implies an ascending order and ‘genre’: -1 implies a descending order.

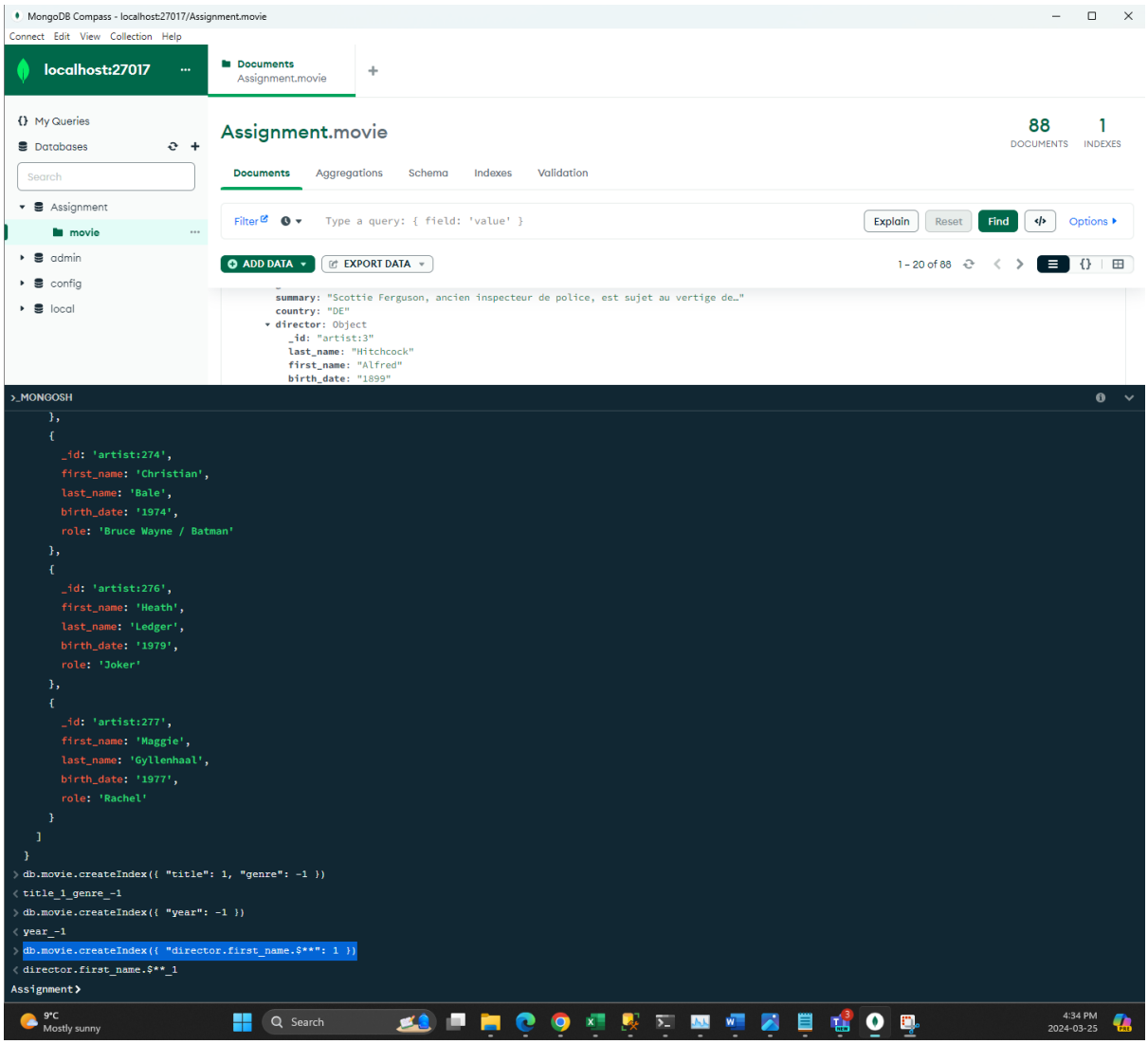


To add year as a single index by descending order we use db.movie.createIndex({ "year": -1 })



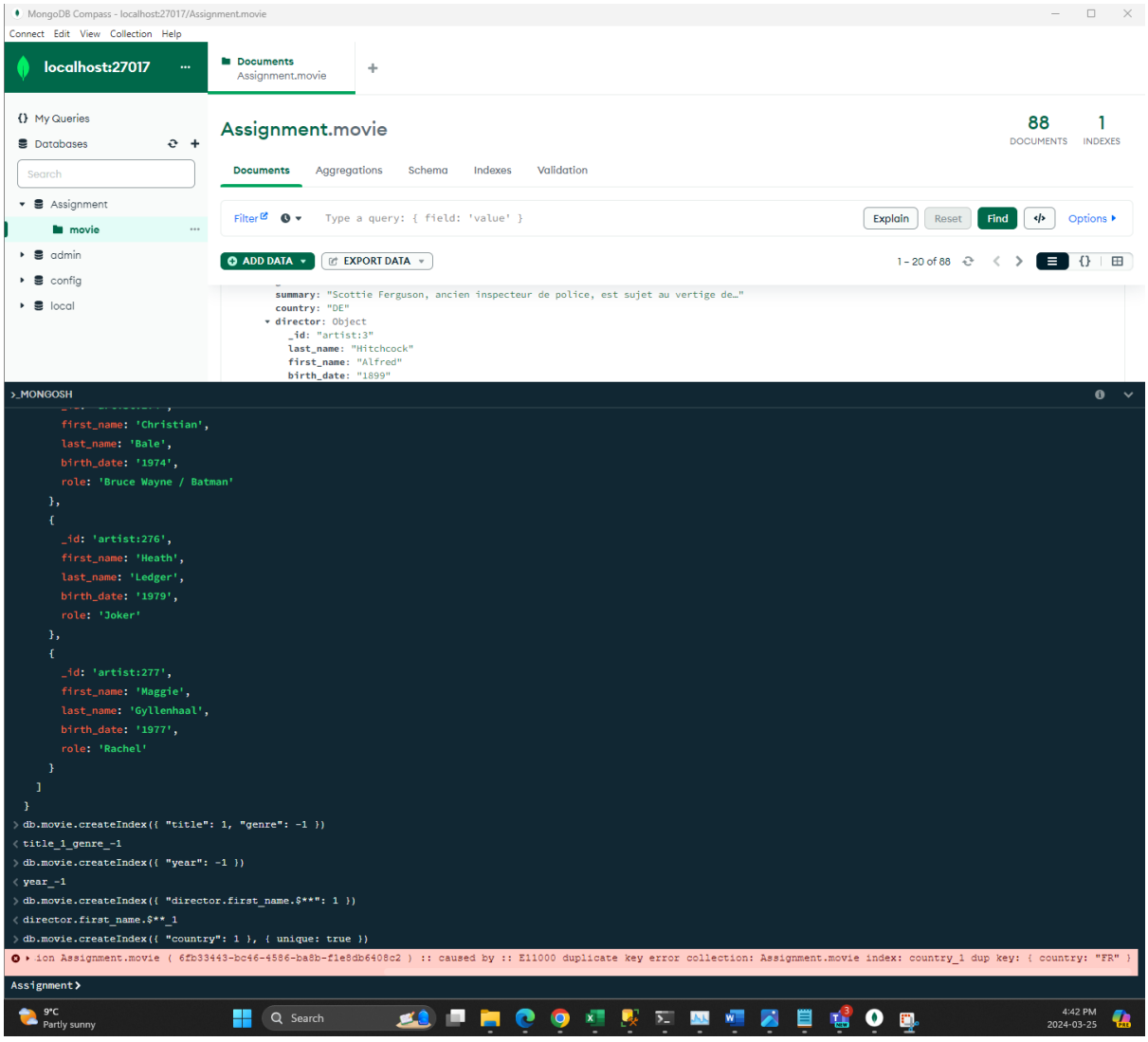
2. Create a wildcard Index on director first_name order by ascending order on movie collection.

Solution: db.movie.createIndex({ "director.first_name.$**": 1 }) – we use the director.first_name and $** to create a wildcard index.



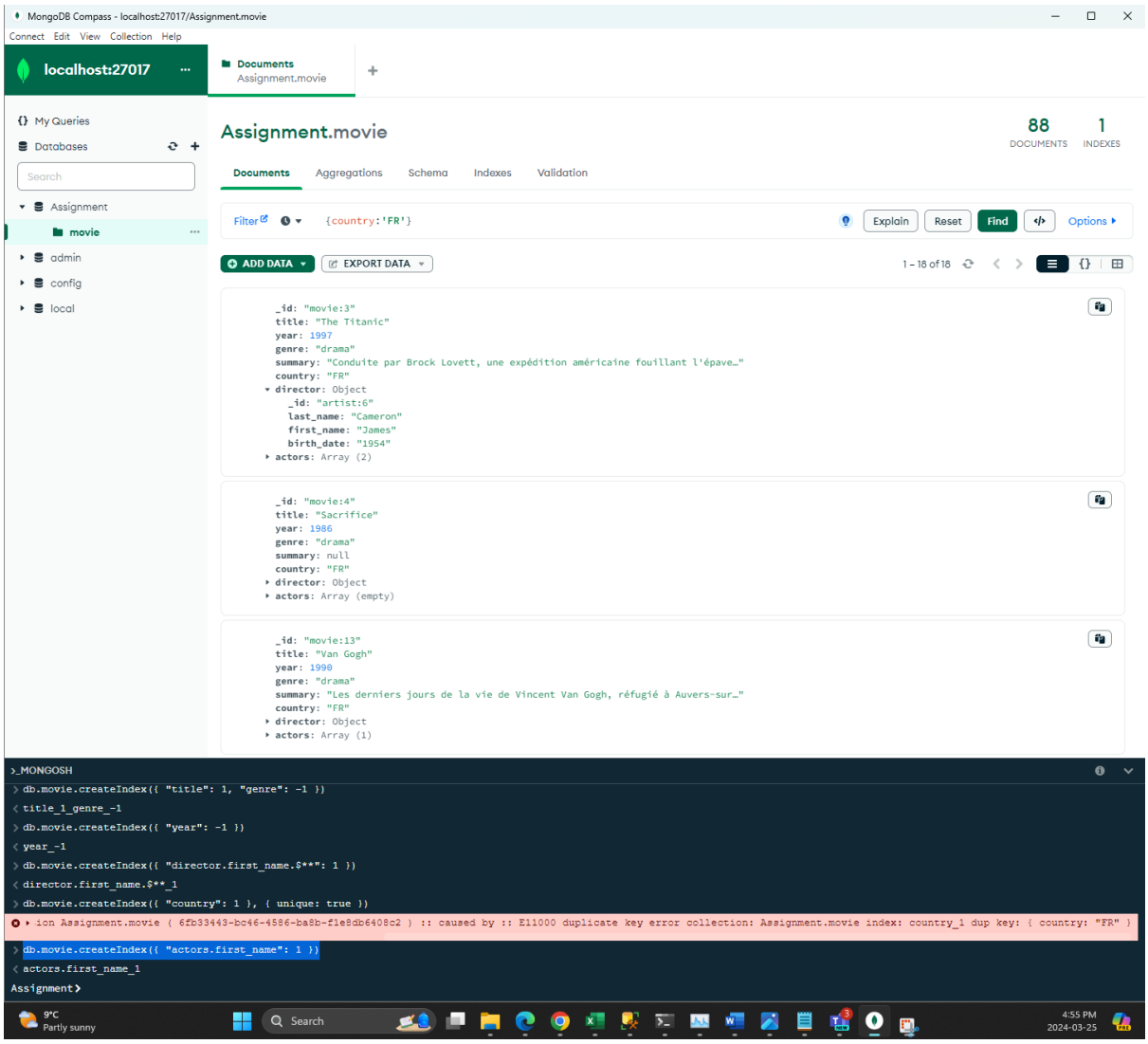
3. Create a unique index on country attribute by ascending order on movie collection.

Solution: db.movie.createIndex({ "country": 1 }, { unique: true }) – we use the createIndex method and specify “country”, its order and unique: true to create a unique index. Since the fields in ‘country’ are common between many movie documents, it does not generate a unique index due to duplicates.



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

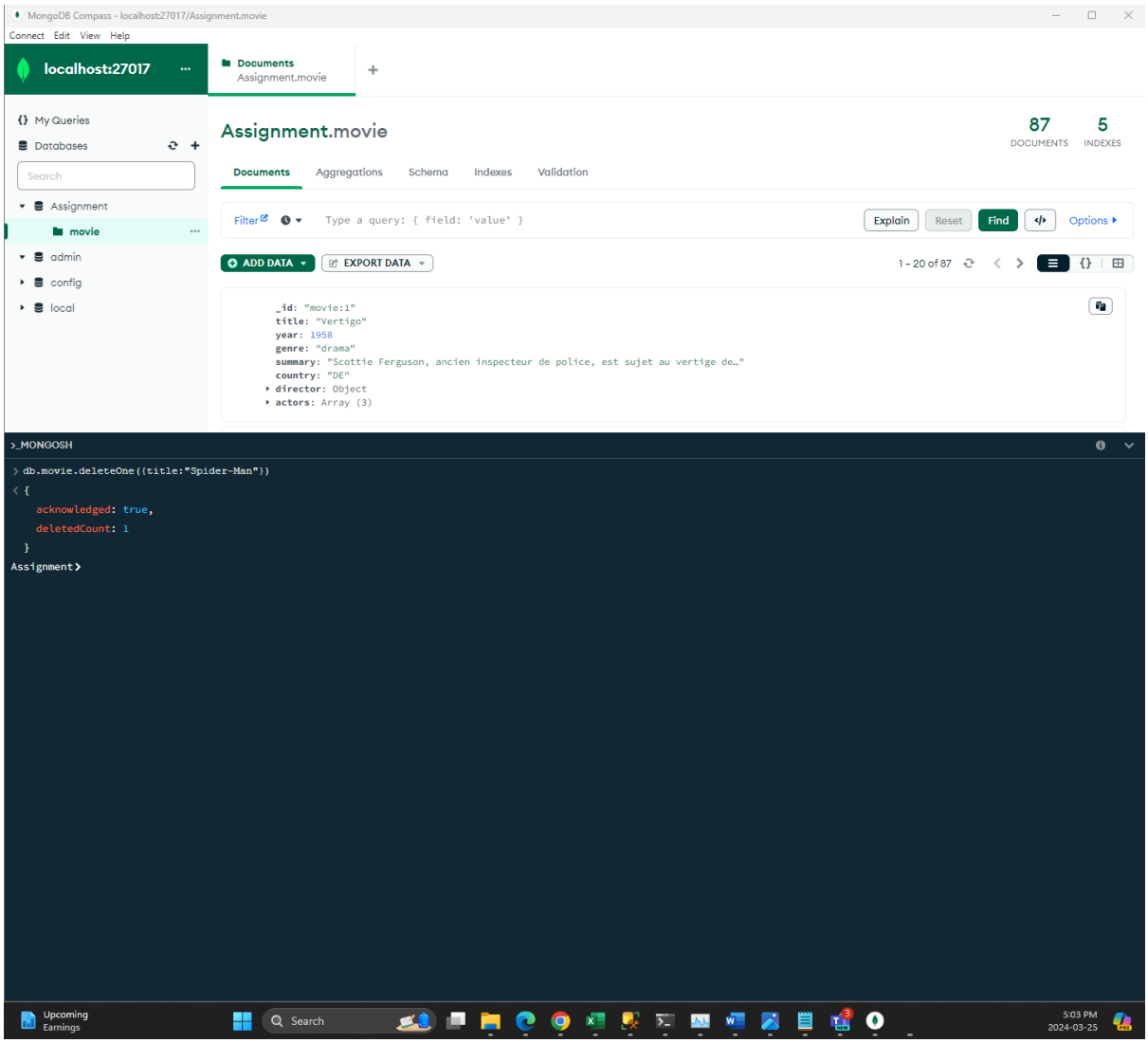
Solution: db.movie.createIndex({ "actors.first_name": 1 }) – we use createIndex() method but since we are indexing a subdocument or element of an array, it creates a multikey index.



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

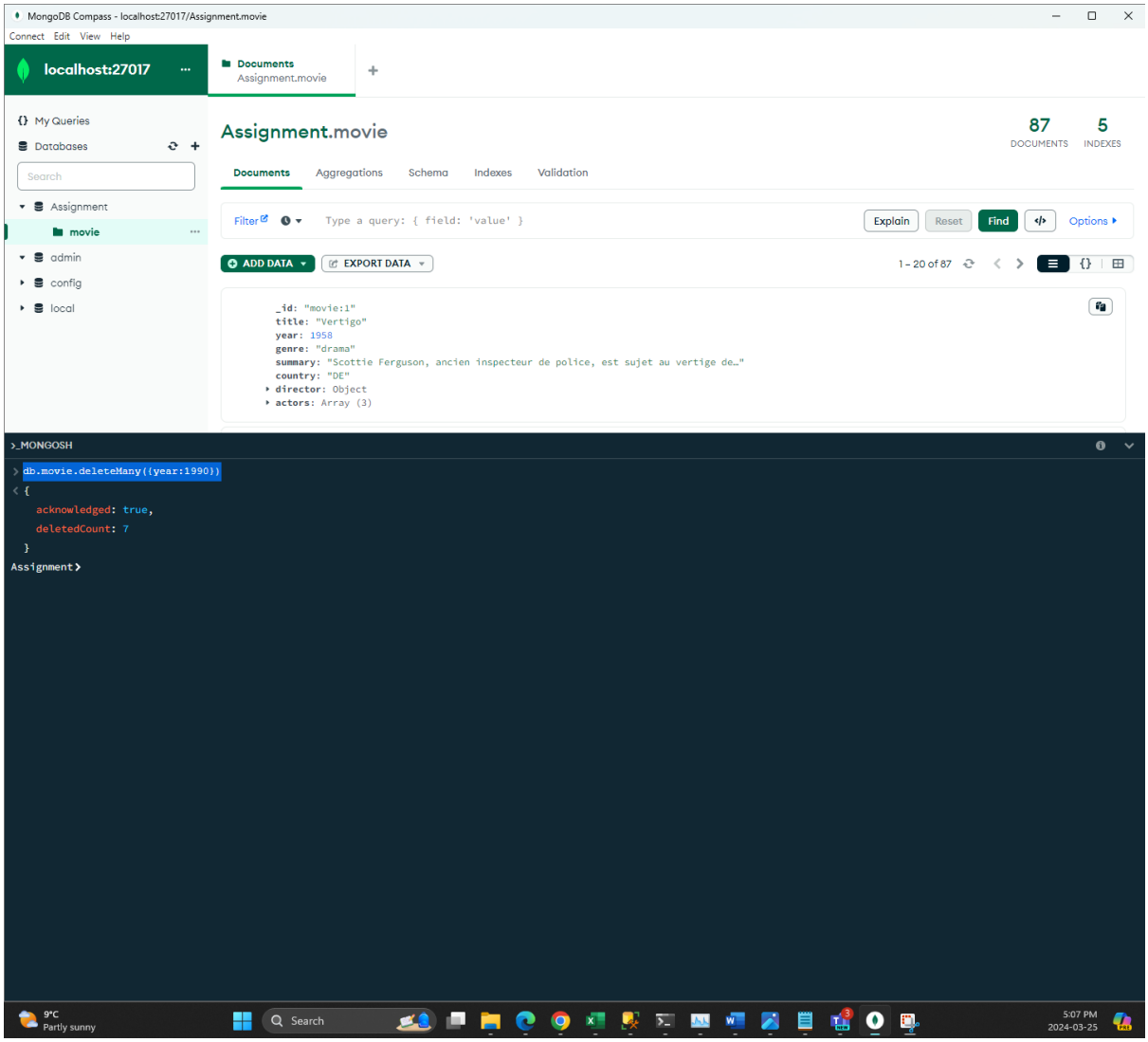
1. Delete the movie with title "Spider-Man".

Solution: db.movie.deleteOne({title:"Spider-Man"}) – we use deleteOne() method and specify the title to delete movie Spider-Man.



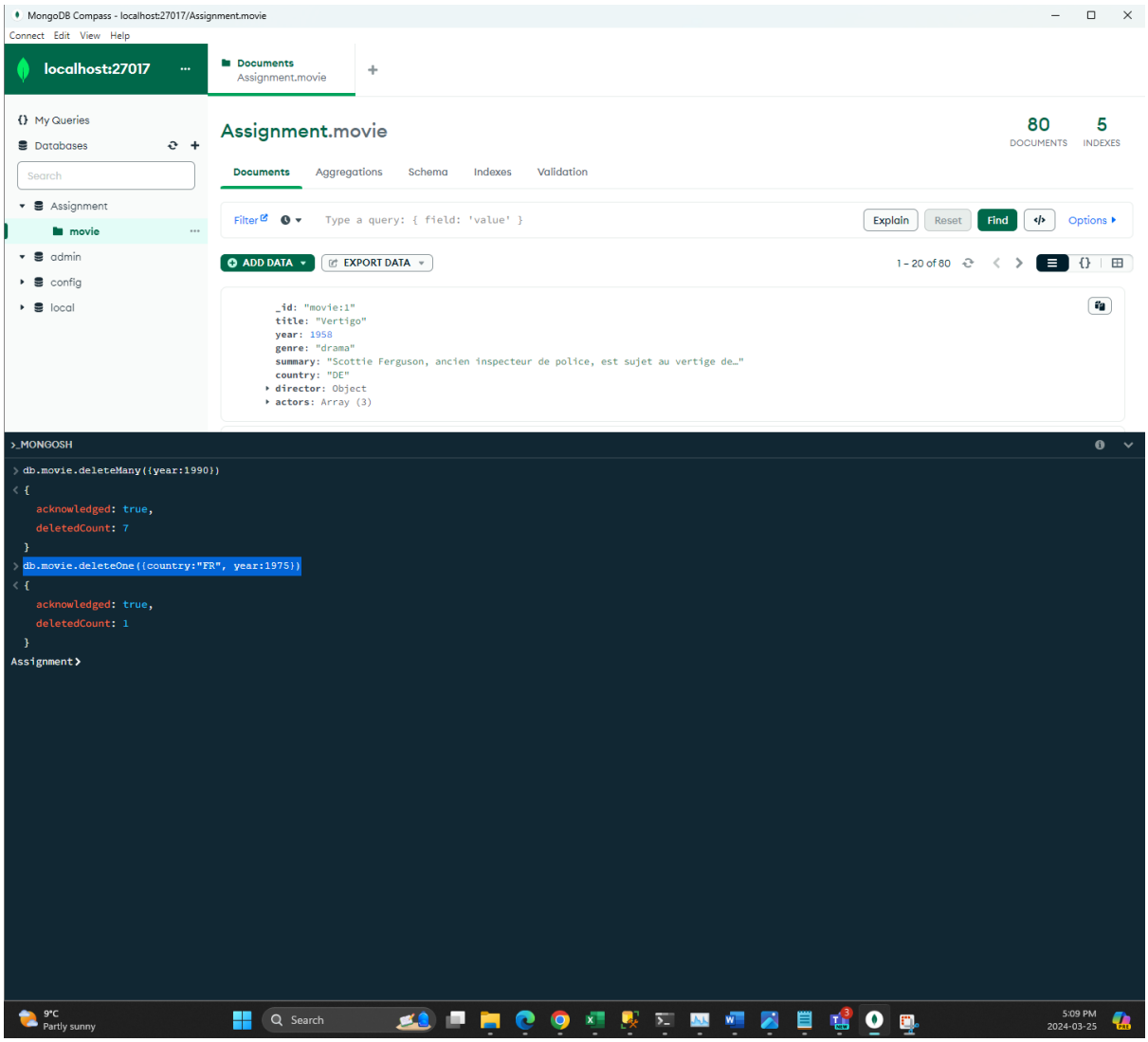
2. Delete the movies which is released in 1990.

Solution: db.movie.deleteMany({year:1990}) – We use DeleteMany() method and specify the year to delete all movies in 1990.



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

Solution: db.movie.deleteOne({country:"FR", year:1975}) – we use deleteOne() method and specify the country and year of the movie document we want to delete.



4. Delete the movies with genre is drama.

Solution: db.movie.deleteMany({genre:"drama"}) – we use the deleteMany() method and specify the genre of movies to delete.