**Submitted by: Ayush Agrawal**

**Date: 12/01/2022**

**MongoDB Lab Assignments -Day 1**

**Section 1: - MongoDB Exercise in mongo shell**

Connect to a running mongo instance, use a database named **mongo\_practice.**

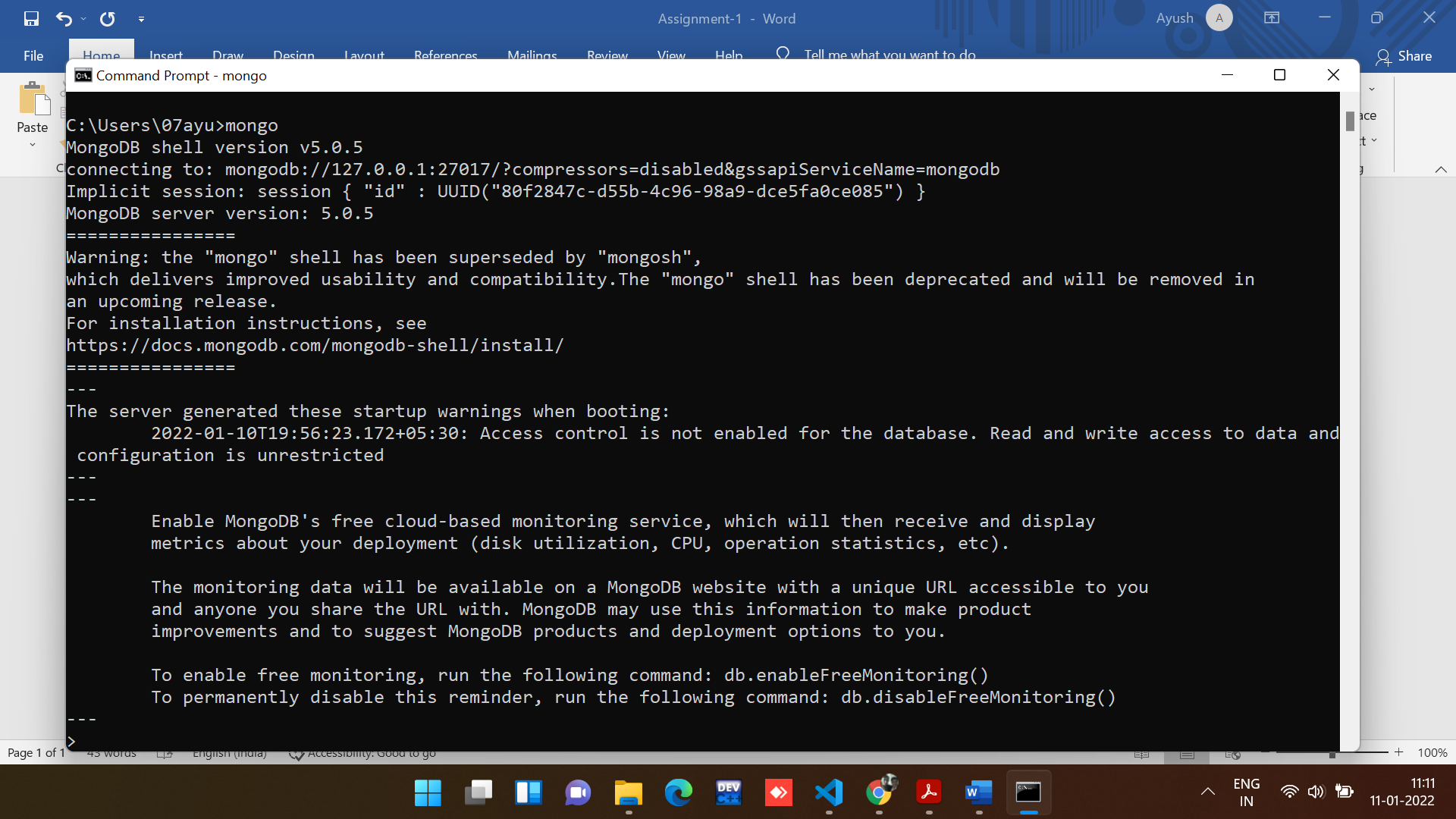
Document all your queries in a javascript file to use as a reference.

**Command to open mongo shell in command prompt:**

mongo

**Output:**

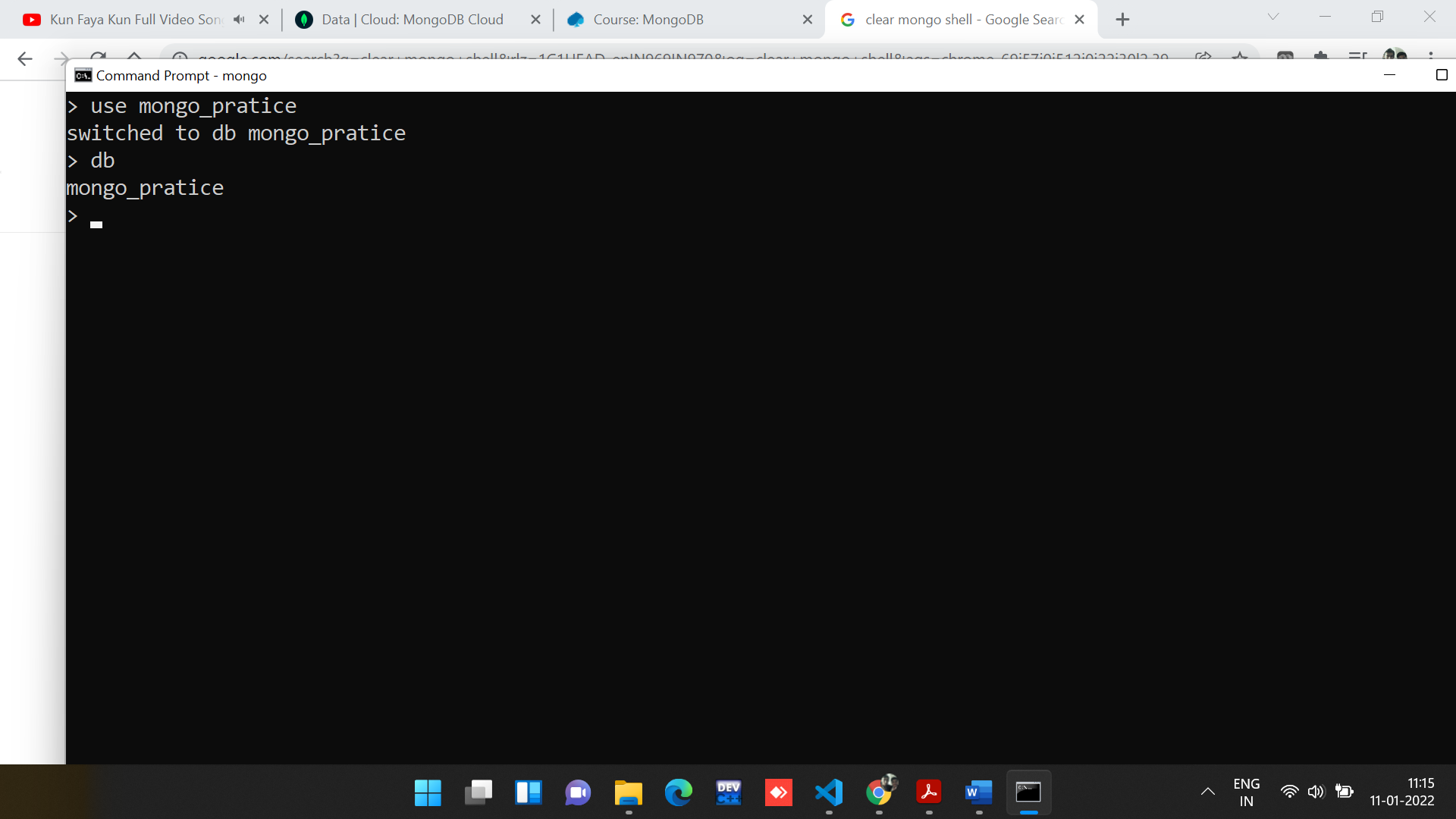
Command line shell i.e mongo has been instantized in the command prompt.



**Command to create and switch to database named ‘mongo\_practice’:**

use mongo\_pratice

**Output:**



**Note:** command ‘db’ will show the current database we are working on.

**Section-2: - Insert Documents**

Insert the following documents into a movies collection.

**Command to create a collection named ‘movies’:**

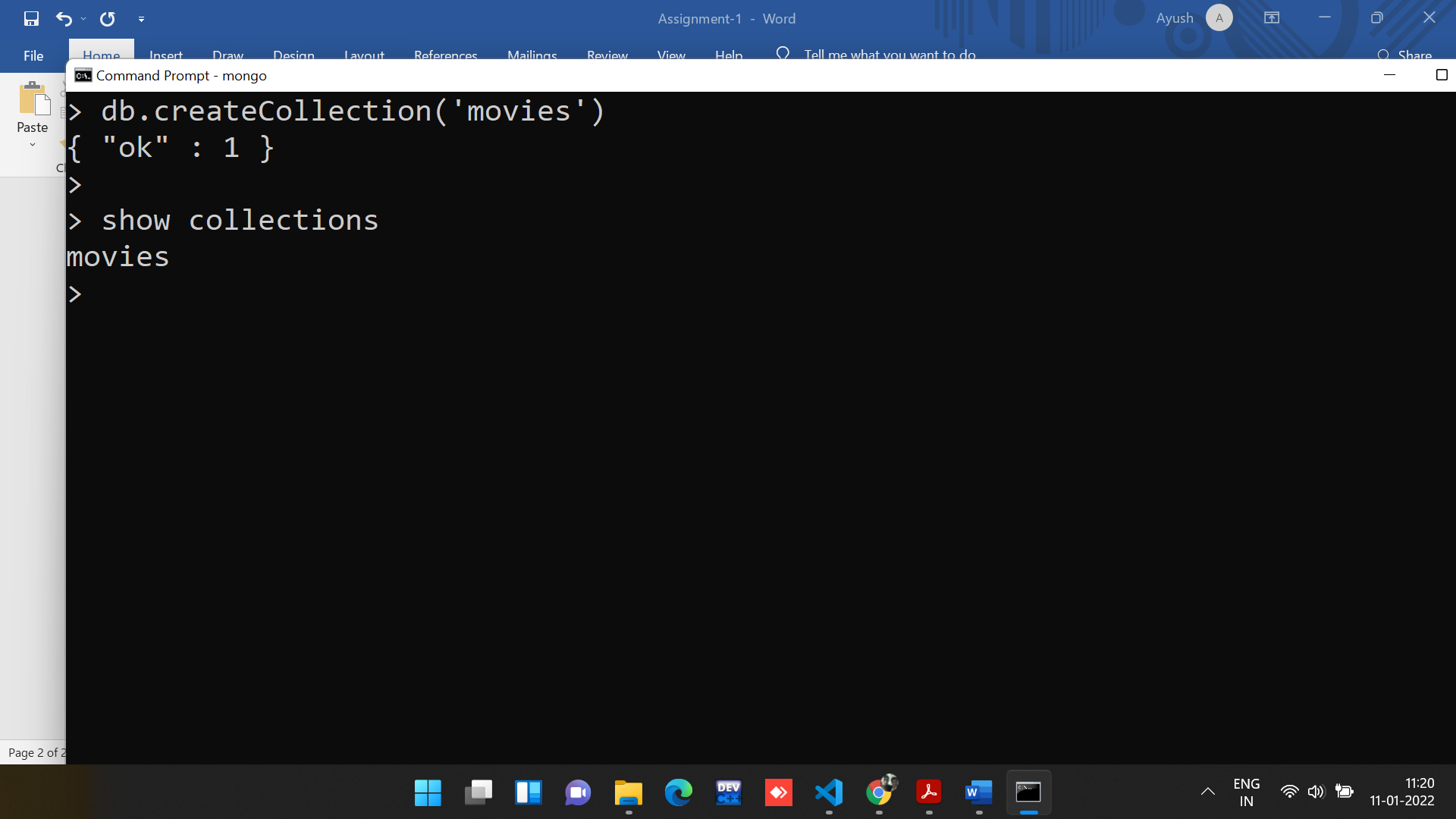
*//Command to create a collection.*

db.createCollection('movies')

*//Command to display all the collections in the database.*

show collections

**Output:**



Insert the provided data in the collection ‘movies’.

**Command to insert many rows in the collection:**

*//command to insert many rows in the database.*

db.movies.insertMany([{

    "title" : "Fight Club",

    "writer" : "Chuck Palahniuko",

    "year" : 1999,

    "actors" : [ "Brad Pitt", "Edward Norton" ]

},

{

    "title" : "Pulp Fiction",

    "writer" : "Quentin Tarantino",

    "year" : 1994,

    "actors" : [ "John Travolta", "Uma Thurman" ]

},

{

    "title" : "Inglorious Basterds",

    "writer" : "Quentin Tarantino",

    "year" : 2009,

    "actors" : [ "Brad Pitt", "Diane Kruger","Eli Roth" ]

},

{

    "title" : "The Hobbit : An Unexpected Journey",

    "writer" : "J.R.R. Tolkein",

    "year" : 2012,

    "franchise" : "The Hobbit"

},

{

    "title" : "The Hobbit: The Desolation of Smaug",

    "writer" : "J.R.R. Tolkein",

    "year" : 2013,

    "franchise" : "The Hobbit"

},

{

    "title" : "The Hobbit: The Battle of the Five Armies writer ",

    "writer" : "J.R.R. Tolkein",

    "year" : 2012,

    "franchise" : "The Hobbit",

    "synopsis" : "Bilbo and Company are forced to engage in a war against an array of combatants and keep the Lonely Mountain from falling into the hands of a rising darkness."

},

{

    "title" : "Pee Wee Herman's Big Adventure"

},

{

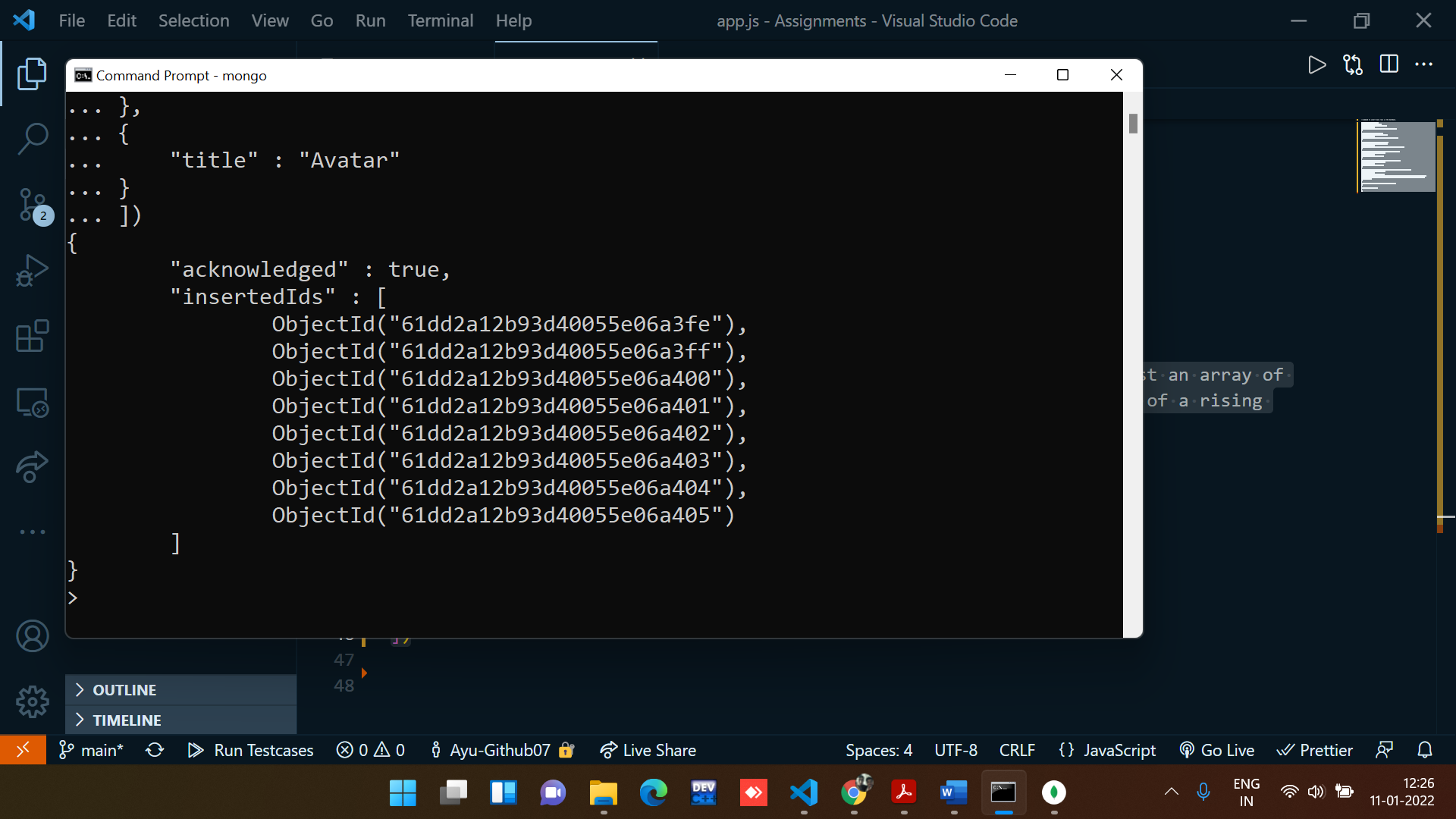
    "title" : "Avatar"

}

])

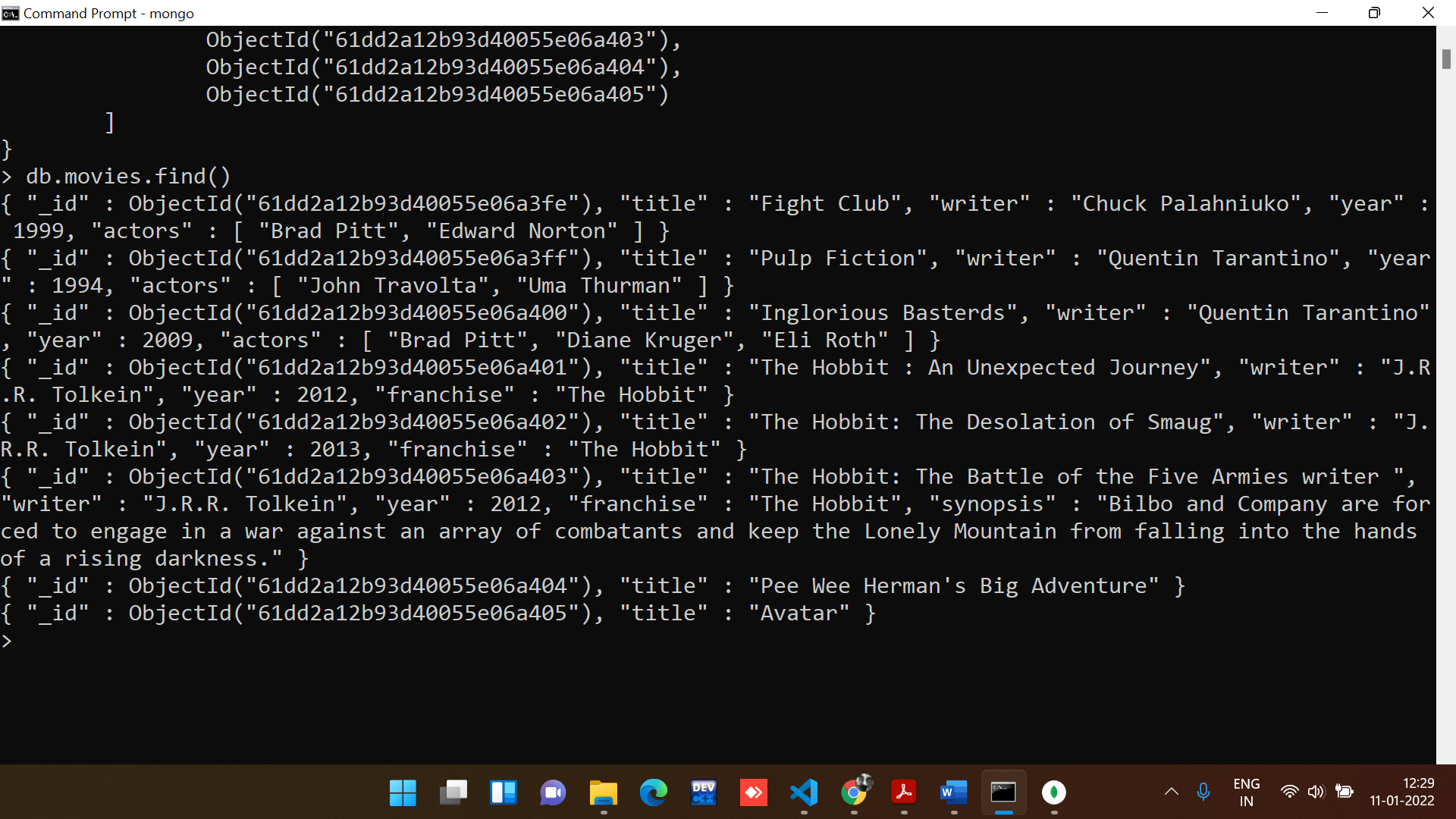
**Output:**

All the rows have been inserted with unique object id.

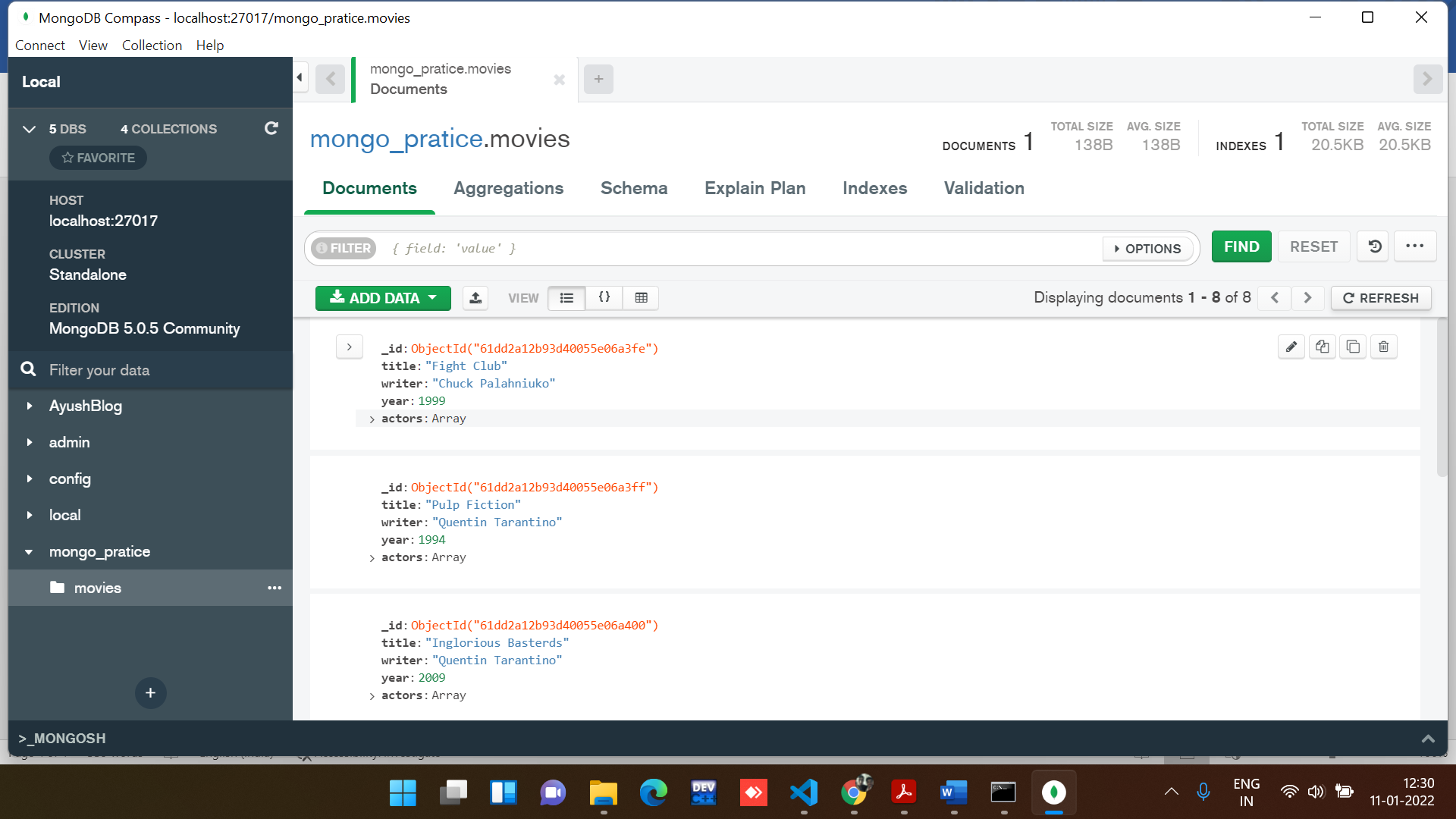


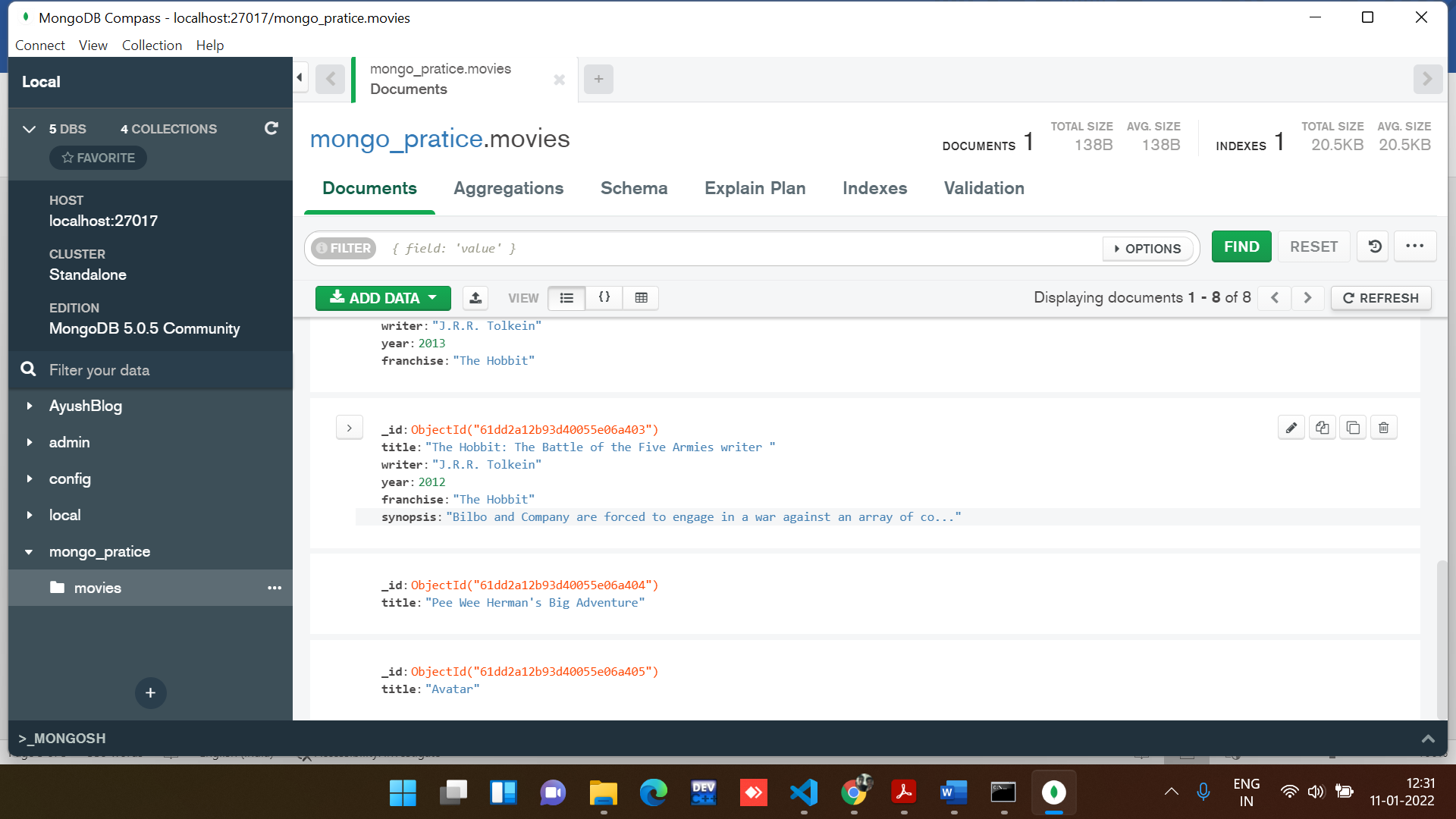
**Command:**

db.movies.find() :- To display all the inserted fields in the collection.



**Mongo dB Compass interface of database:**





**Query / Find Documents**

query the movies collection to

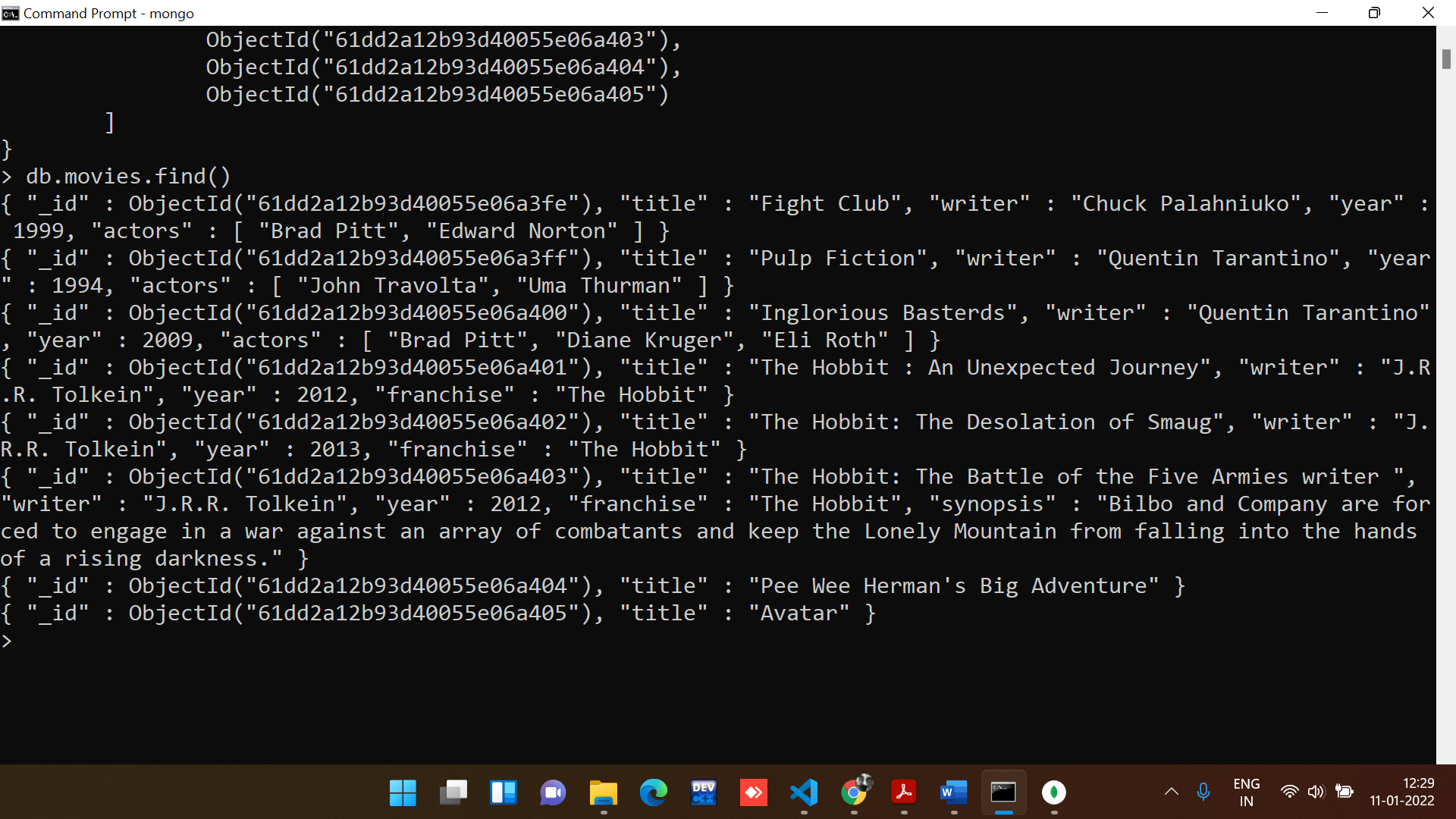
1. get all documents

**Command:**

*//Command to get all the documents in the collection.*

db.movies.find()

**Output:**



2. get all documents with writer set to "Quentin Tarantino"

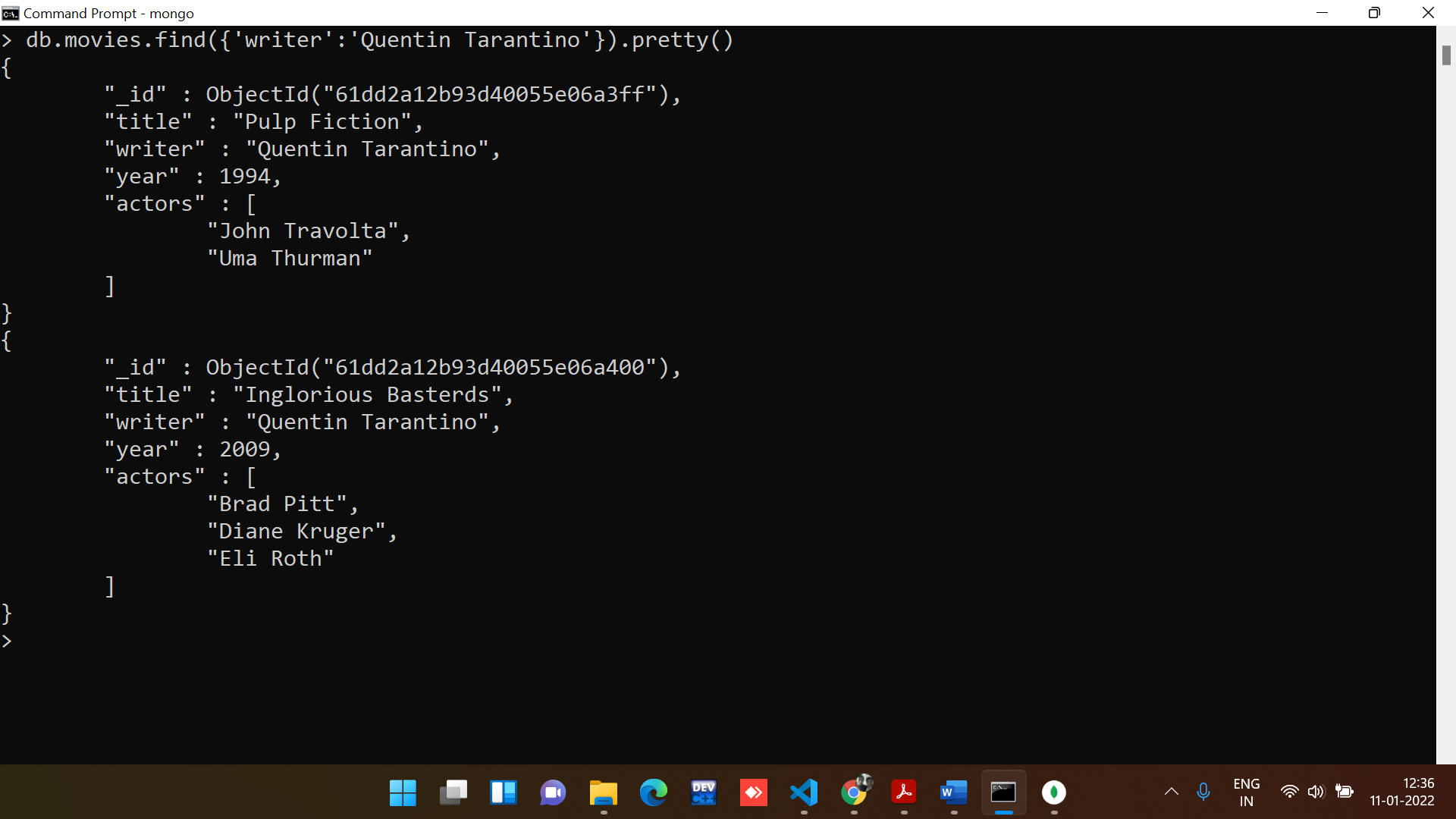
**Command:**

*//Commands to get all documents with writer set to "Quentin Tarantino"*

db.movies.find({'writer':'Quentin Tarantino'}).pretty()

**Output:**

**Note:** pretty() is a method, To display the results in a formatted way.



3. get all documents where actors include "Brad Pitt".

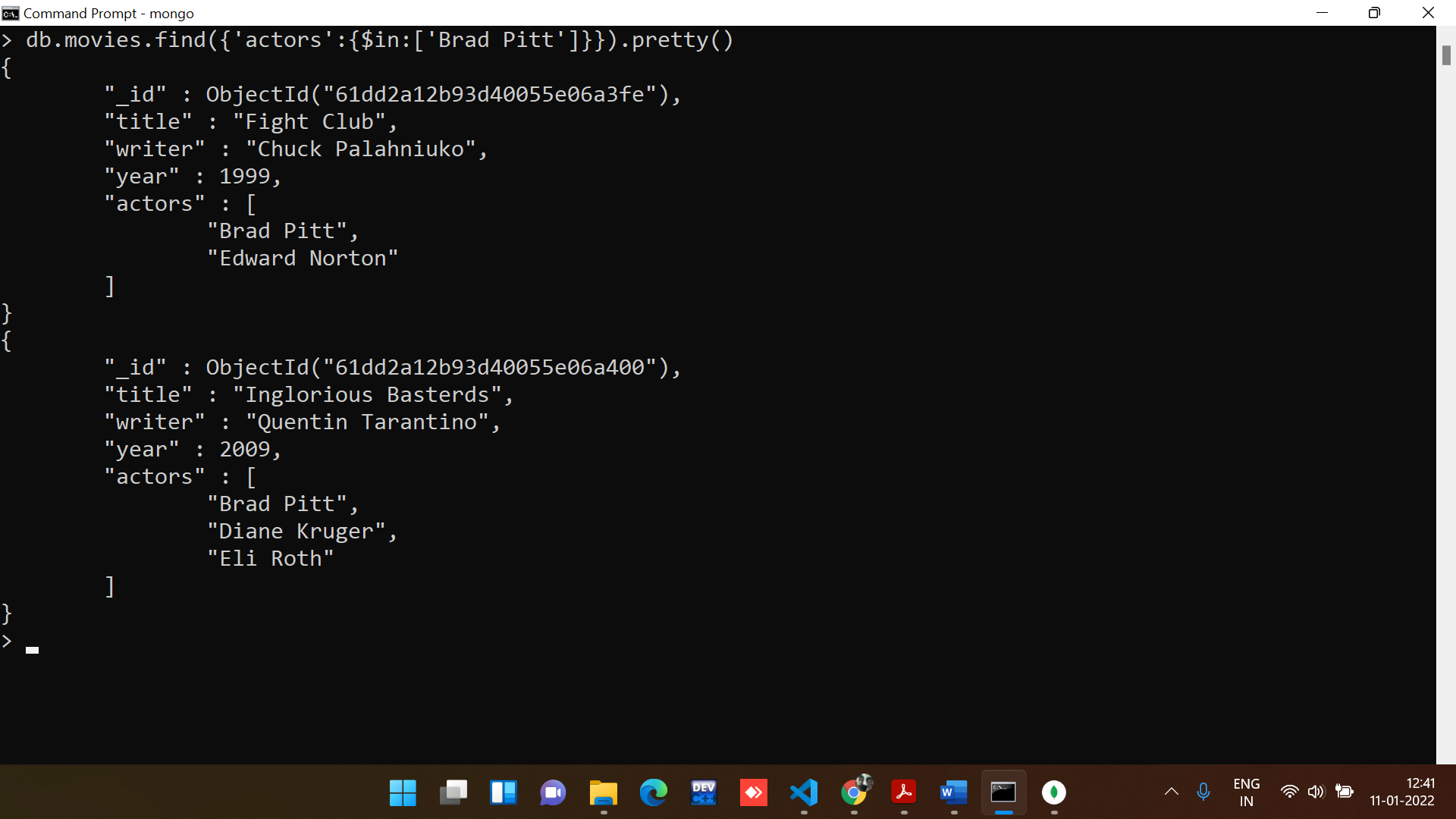
**Command:**

*//commands to get all documents where actors include "Brad Pitt"*

*//As datatype of the actors field is array. we need to use in operator to search for the name inside the actor array.*

db.movies.find({'actors':{$in:['Brad Pitt']}})

**Output:**



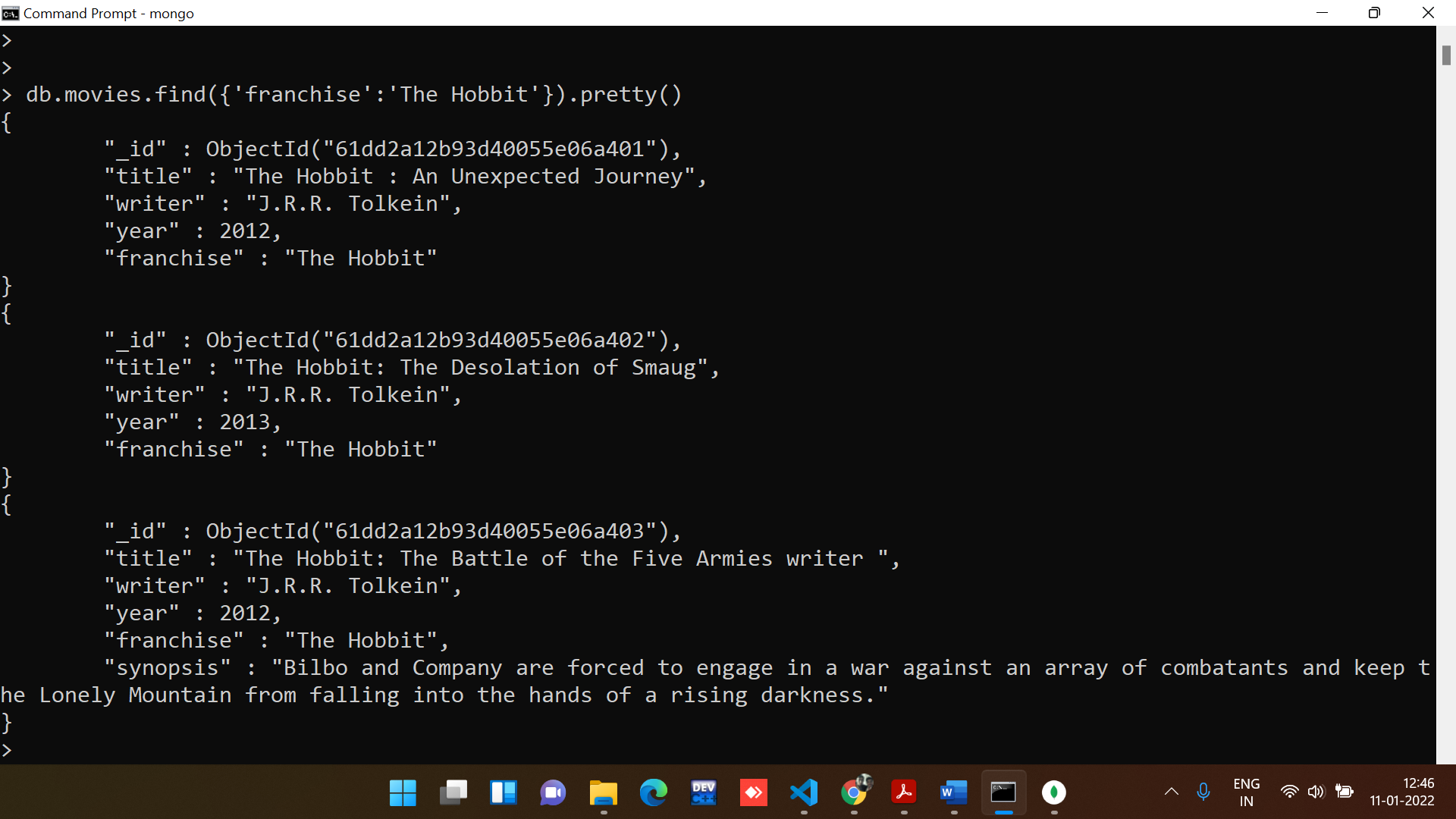
4. get all documents with franchise set to "The Hobbit"

**Command:**

*//Commands to get all documents with franchise set to "The Hobbit"*

db.movies.find({'franchise':'The Hobbit'}).pretty()

**Output:**



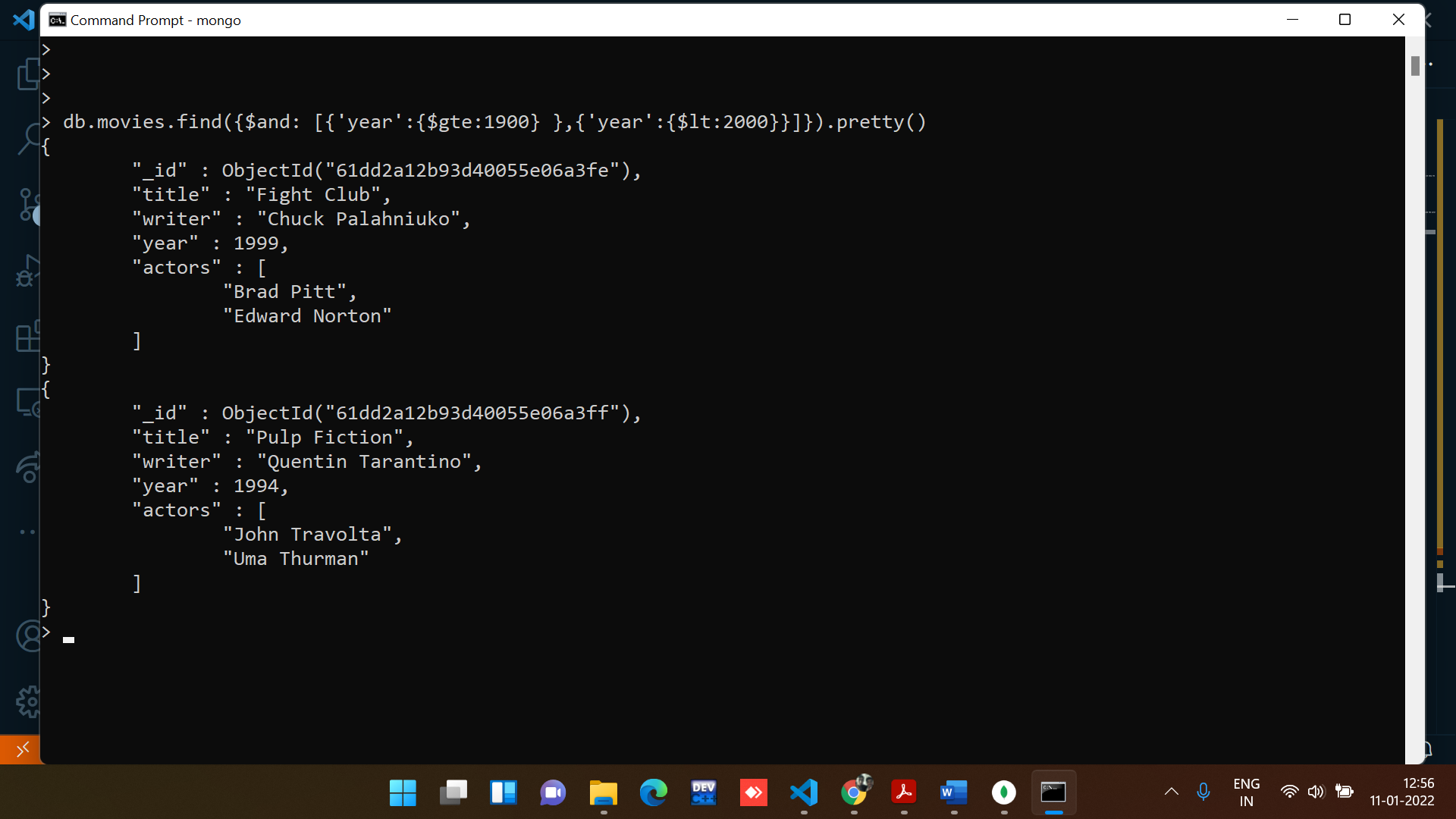
5. get all movies released in the 90s.

**Command:**

*//Commands to get all movies released in the 90s.*

db.movies.find({$and: [{'year':{$gte:1900}},{'year':{$lt:2000}}]})

**Output:**



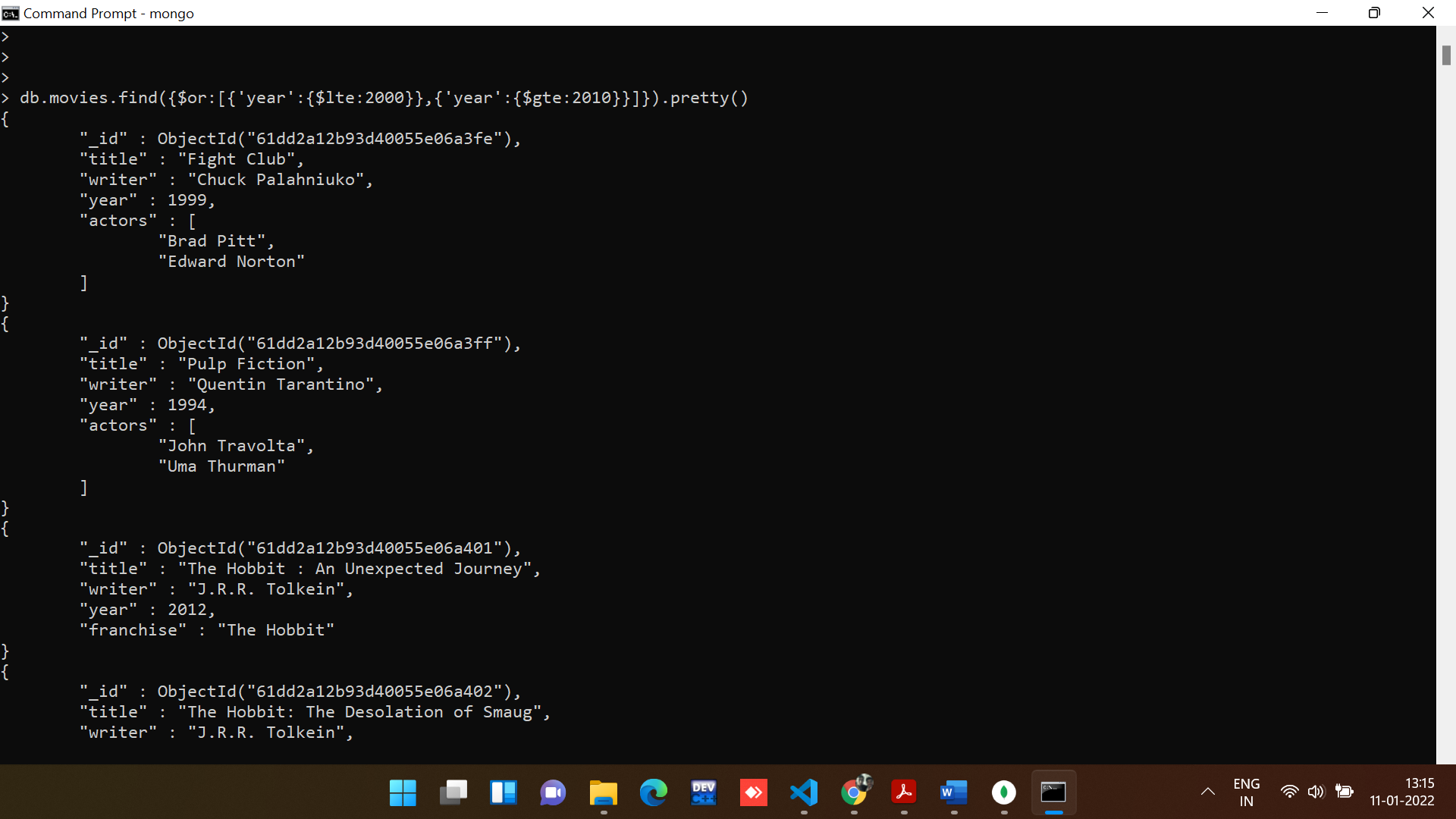
6. get all movies released before the year 2000 or after 2010

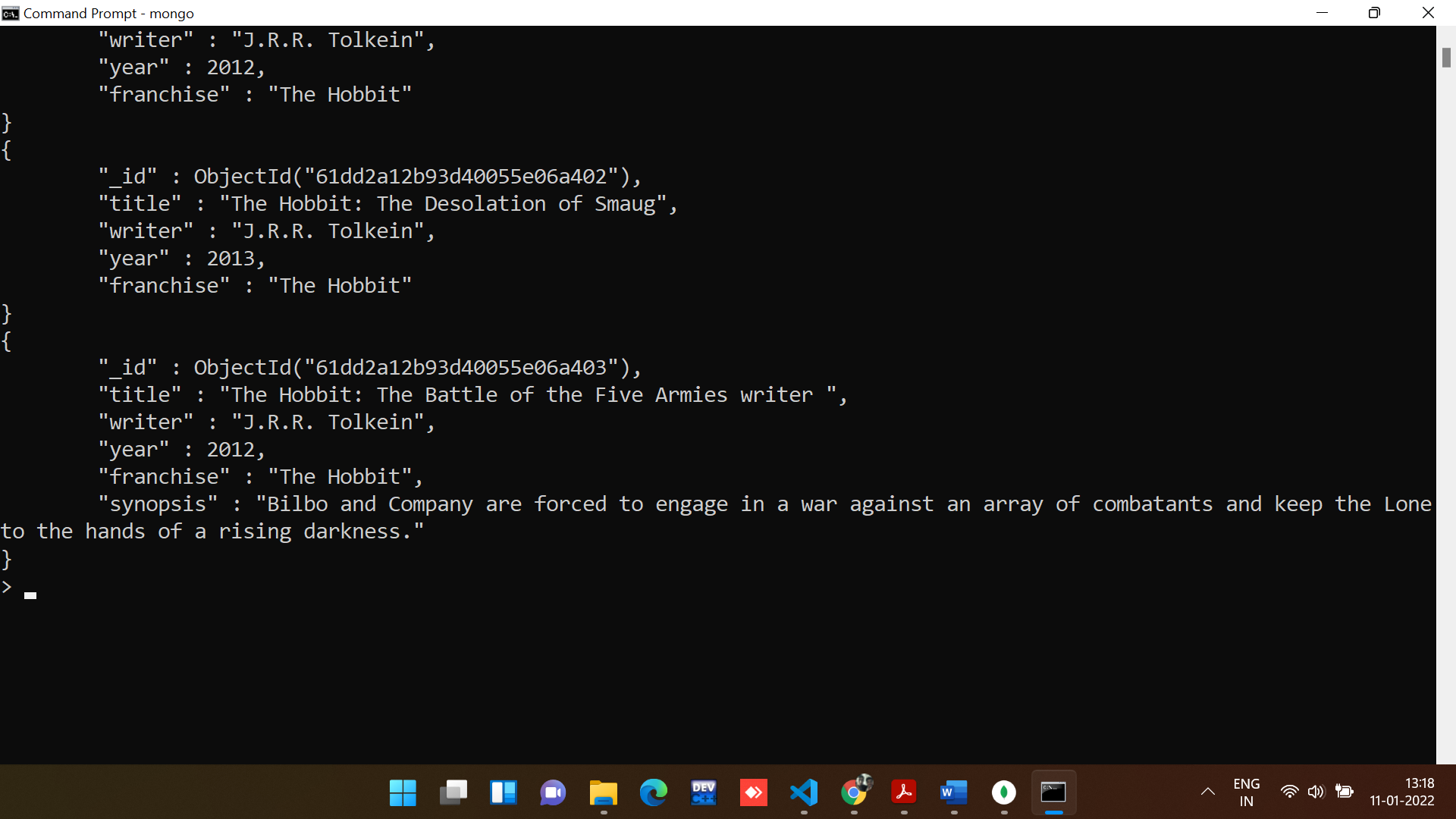
**Command:**

*//Commands to get all movies released before the year 2000 or after 2010*

db.movies.find({$or:[{'year':{$lte:2000}},{'year':{$gte:2010}}]})

**Output:**





**Section-3 :- Update Documents**

1. add a synopsis to "The Hobbit: An Unexpected Journey" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."

**Command:**

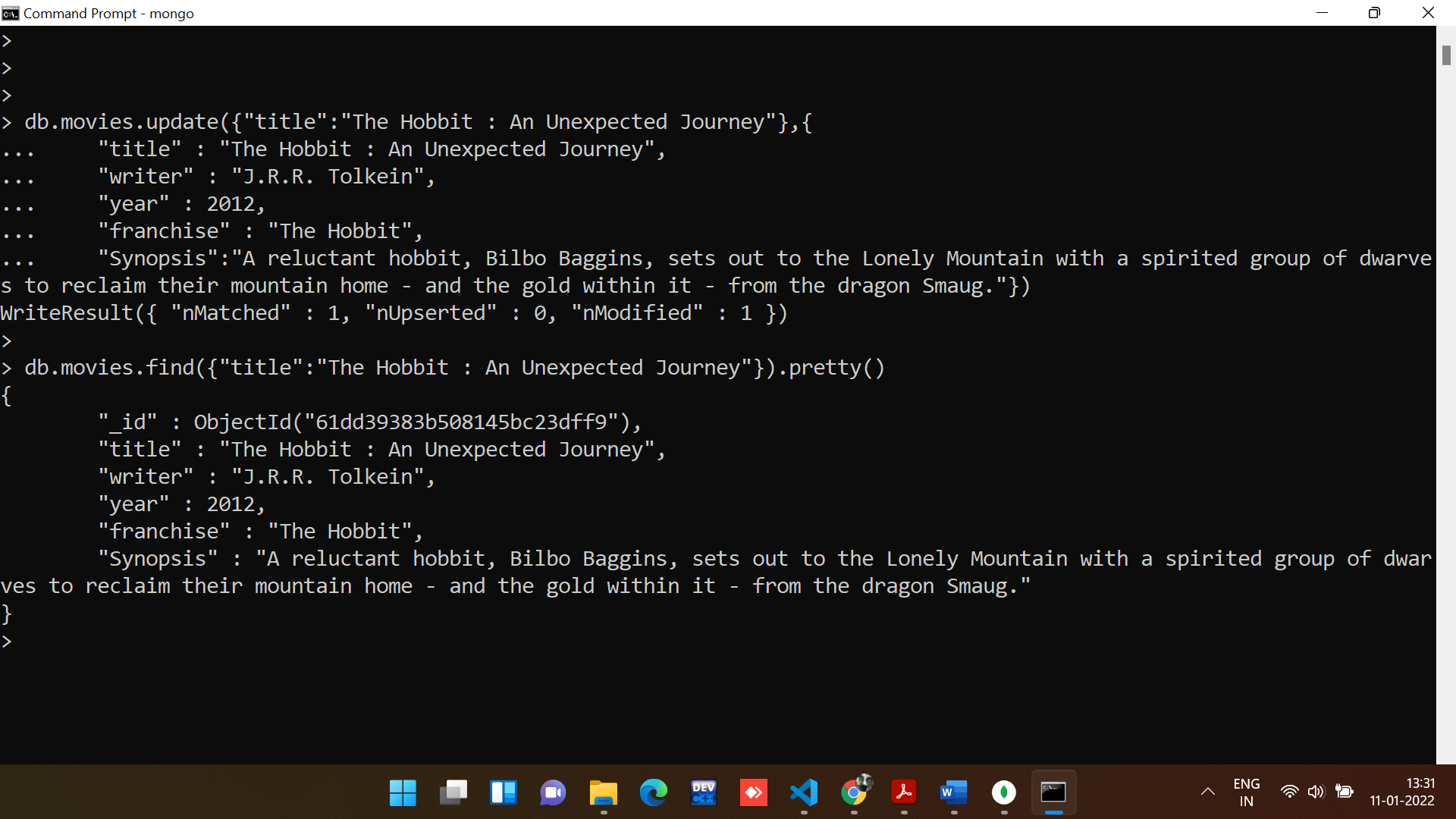
*//Command to add field in the row.*

db.movies.update({"title":"The Hobbit : An Unexpected Journey"},{

    $set:{"Synopsis":"A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."}

   })

**Output:**



2. add a synopsis to "The Hobbit: The Desolation of Smaug" : "The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."

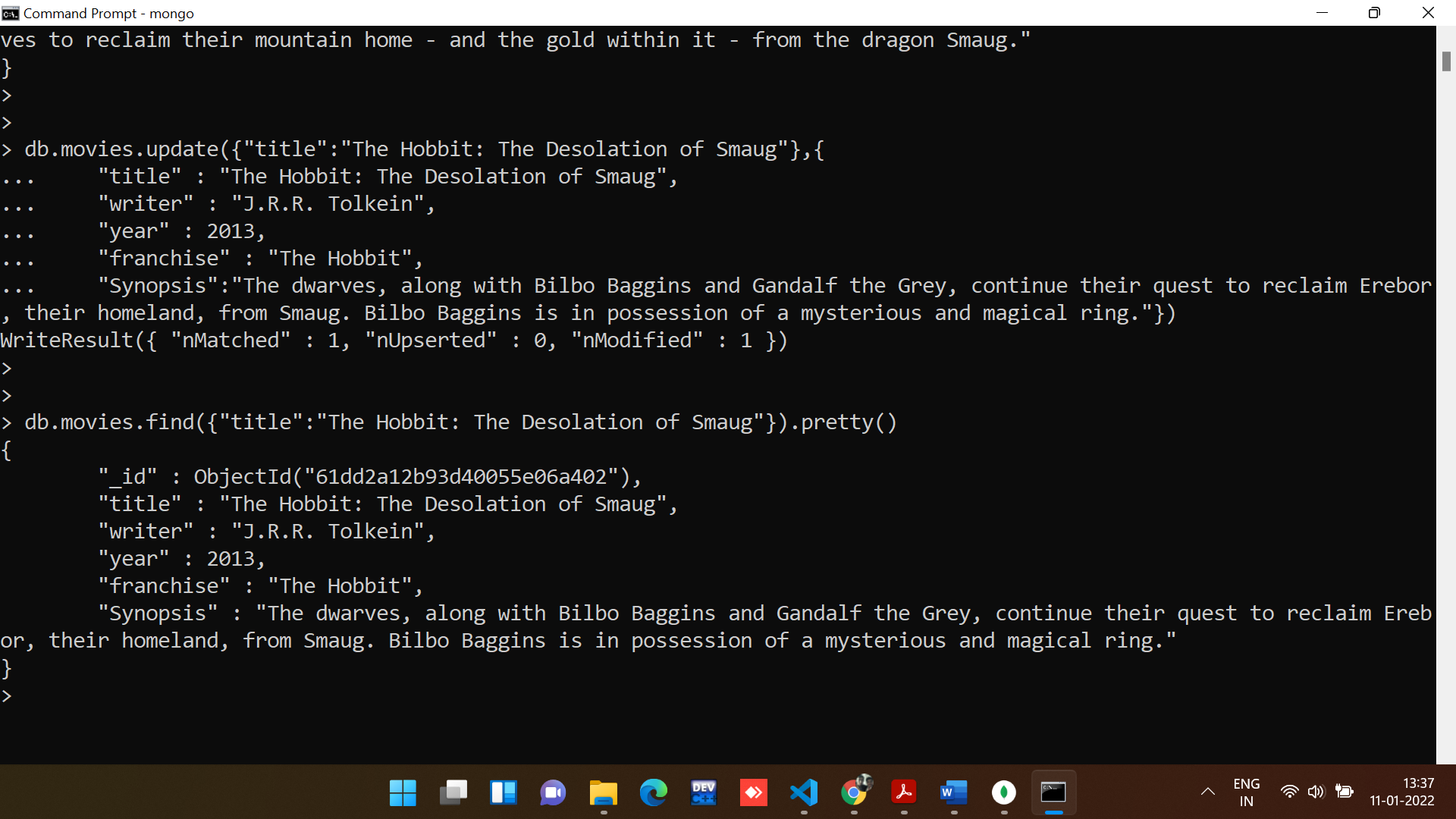
**Command:**

*//Command to add field in the row.*

db.movies.update({"title":"The Hobbit: The Desolation of Smaug"},{ $set:{

    "Synopsis":"The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."}})

**Output:**



3. add an actor named "Samuel L. Jackson" to the movie "Pulp Fiction".

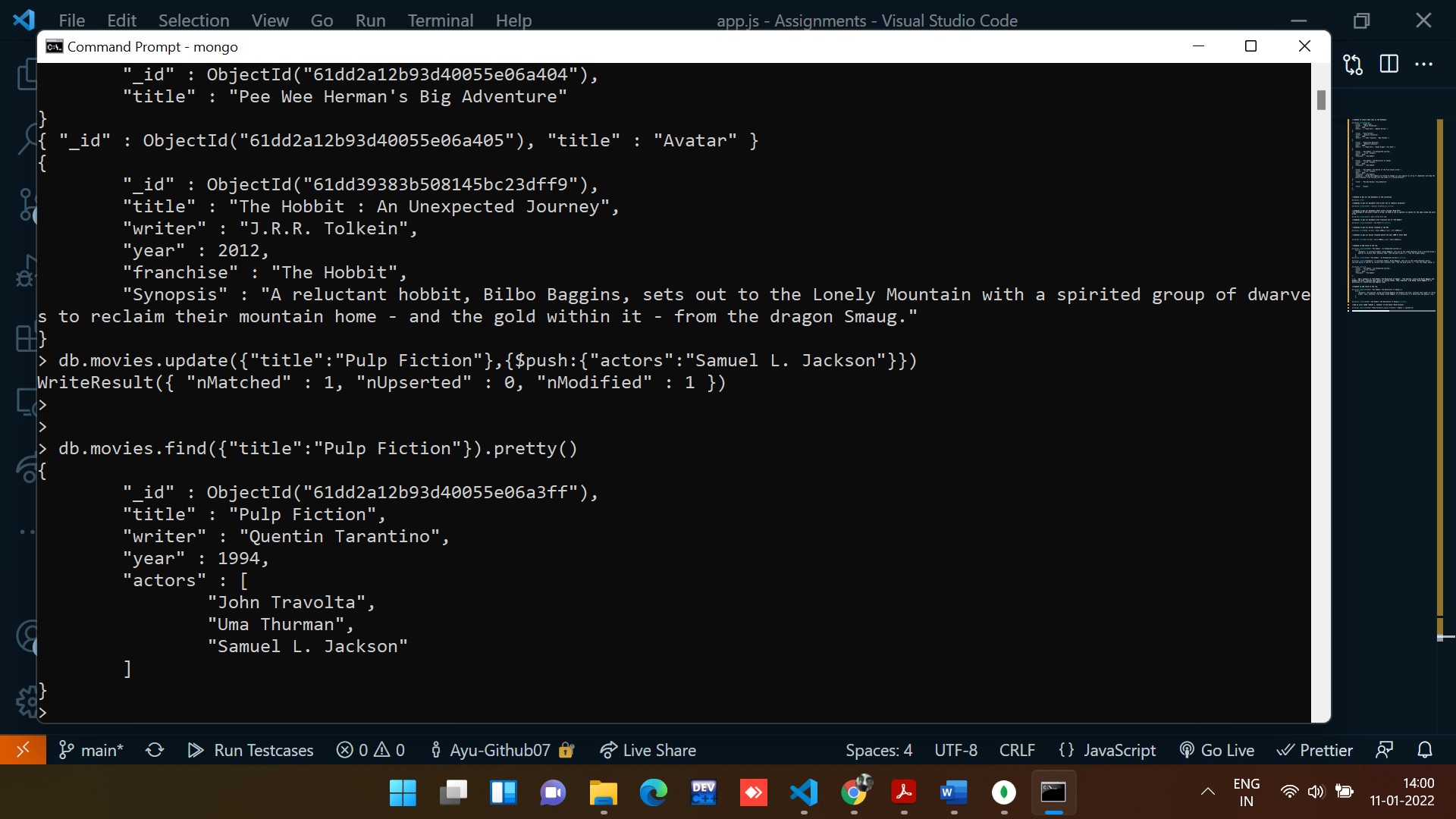
**Command:**

*//add an actor named "Samuel L. Jackson" to the movie "Pulp Fiction".*

db.movies.update({"title":"Pulp Fiction"},{$push:{"actors":"Samuel L. Jackson"}})

db.movies.find({"title":"Pulp Fiction"}).pretty()

**Output:**



**Section-4: - Text Search**

1. find all movies that have a synopsis that contains the word "Bilbo".

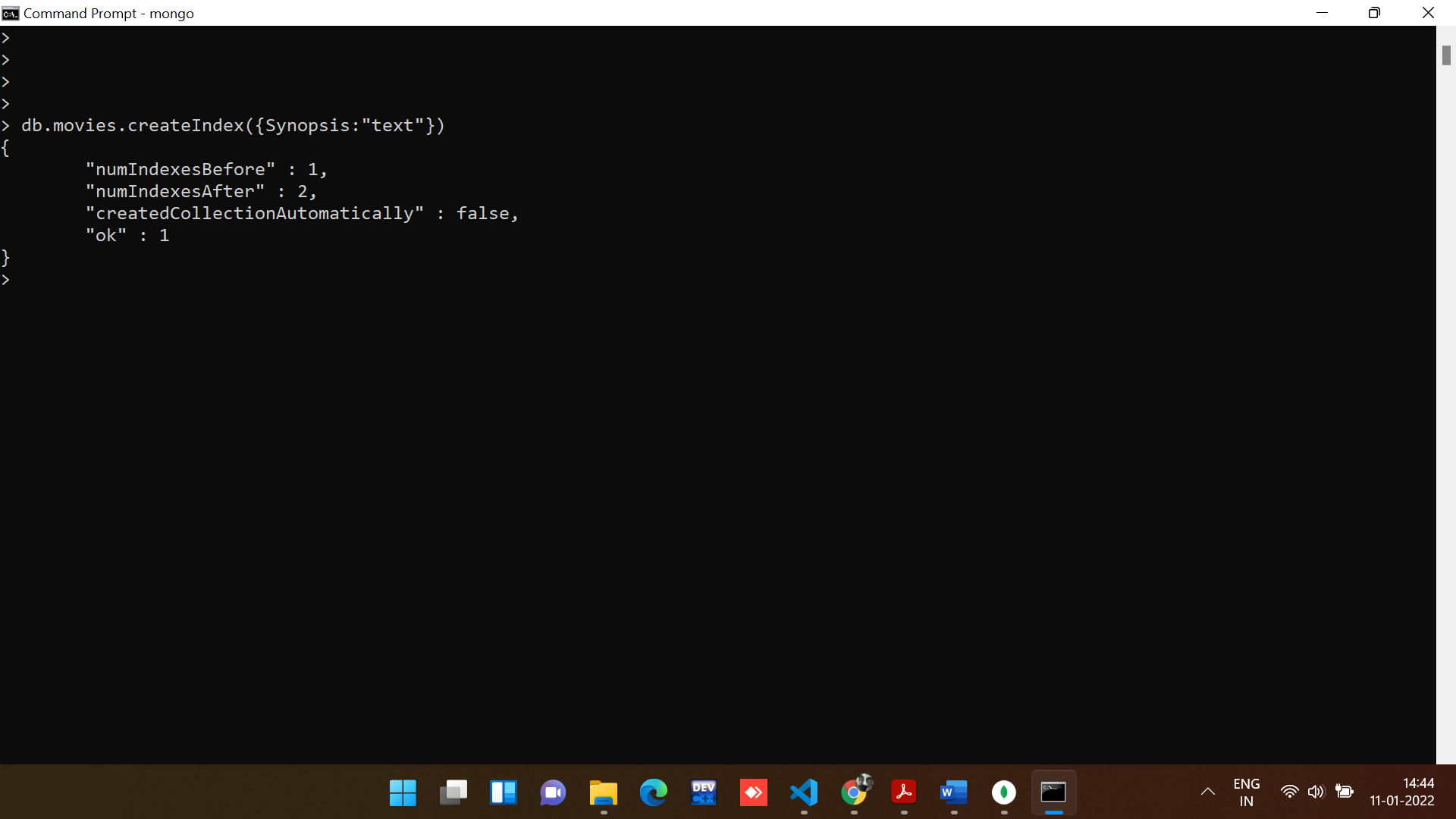
**Creating Text Index**

Consider the following document under posts collection containing the post text and its tags

**Command:**

db.movies.createIndex({Synopsis:"text"})

**Output:**

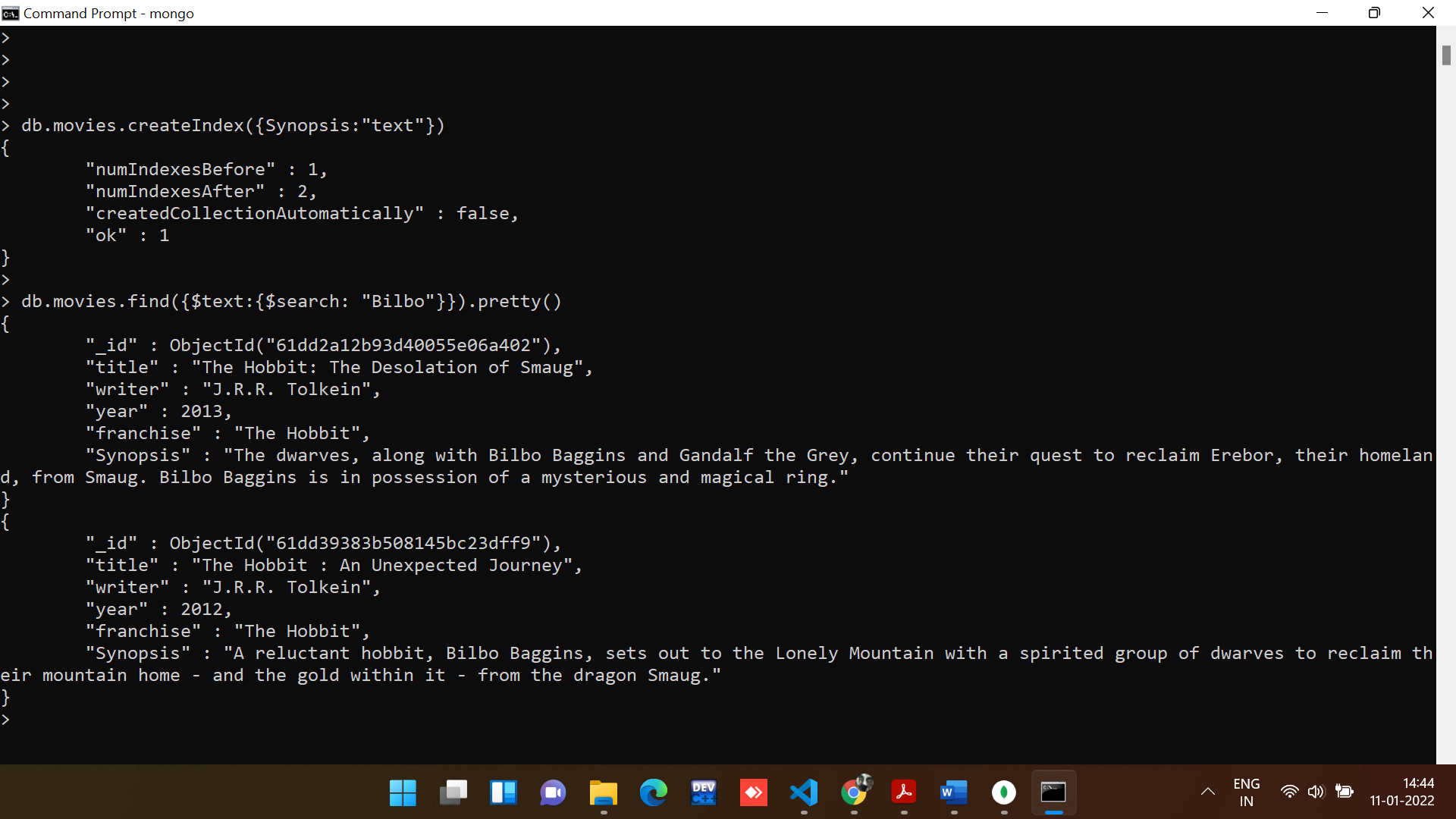


**Command:**

*//Command to search by text using indexes.*

db.movies.find({$text:{$search: "Bilbo"}}).pretty()

**Output:**



2. find all movies that have a synopsis that contains the word "Gandalf".

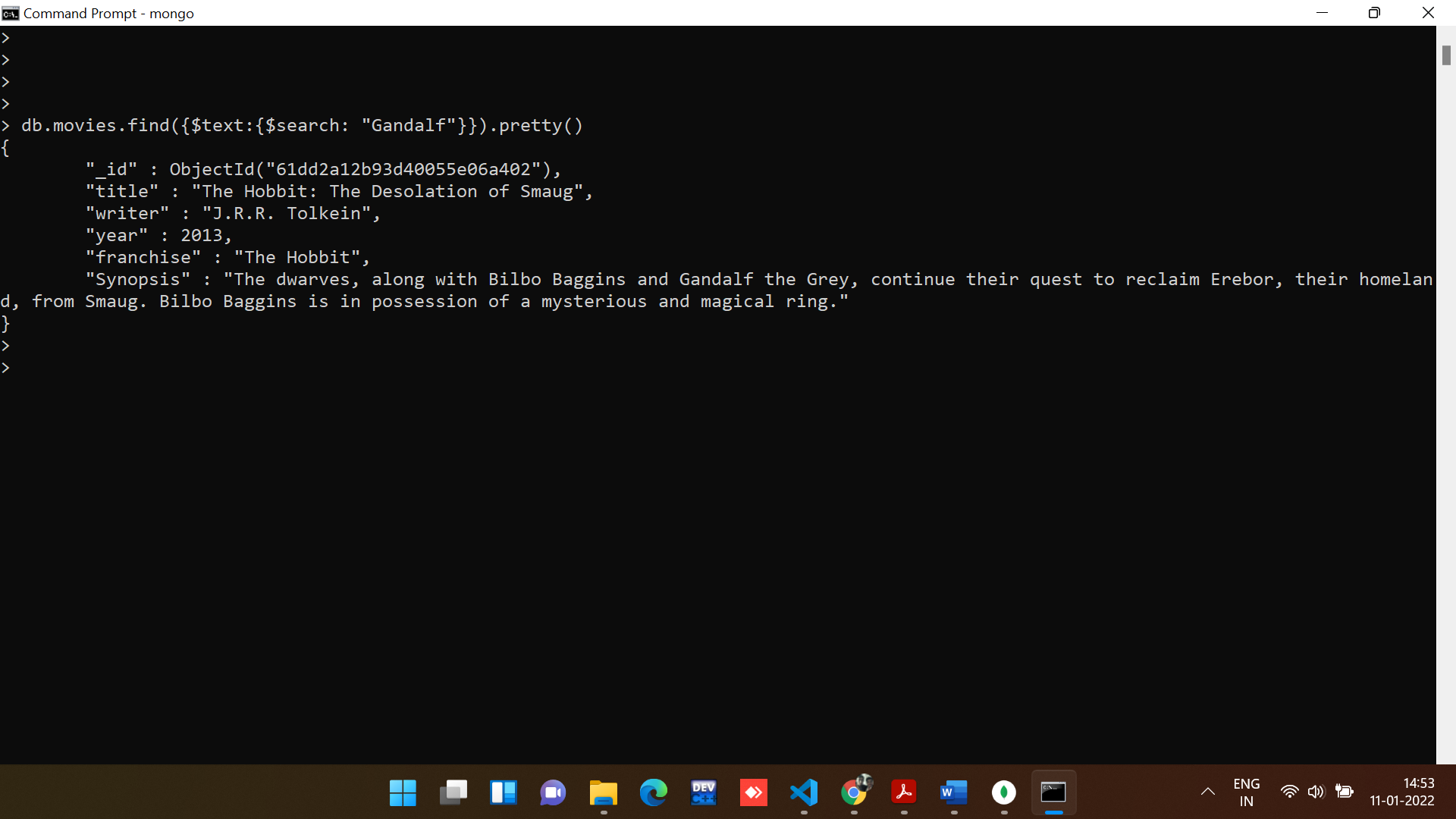
**Command:**

*// find all movies that have a synopsis that contains the word "Gandalf"*

*// Command to search by text using indexes.*

db.movies.find({$text:{$search: "Gandalf"}}).pretty()

**Output:**



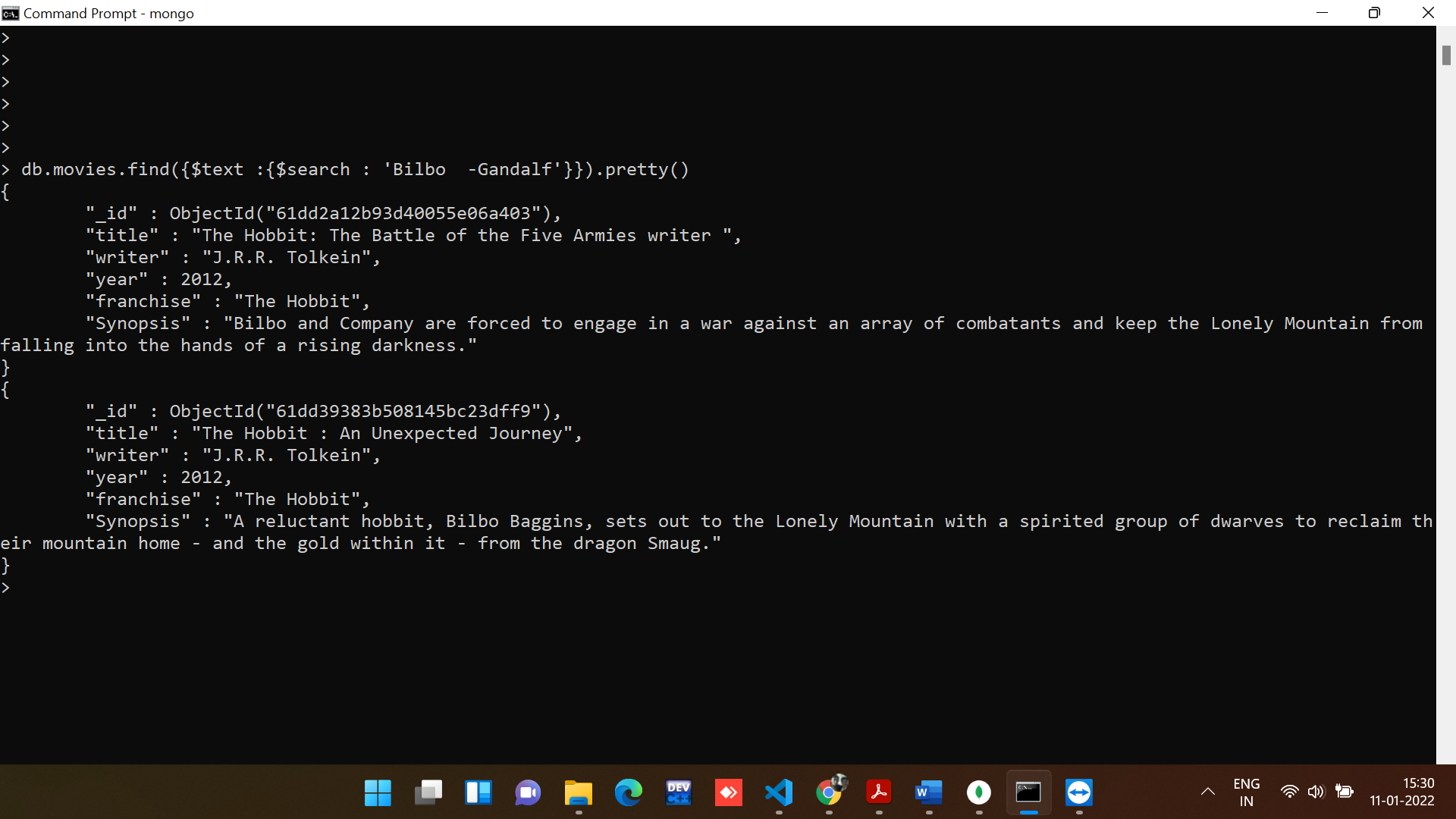
3. find all movies that have a synopsis that contains the word "Bilbo" and not the word "Gandalf"

**Command:**

*//find all movies that have a synopsis that contains the word "Bilbo" and not the word "Gandalf"*

db.movies.find({$text :{$search : 'Bilbo  -Gandalf'}}).pretty()

**Output:**



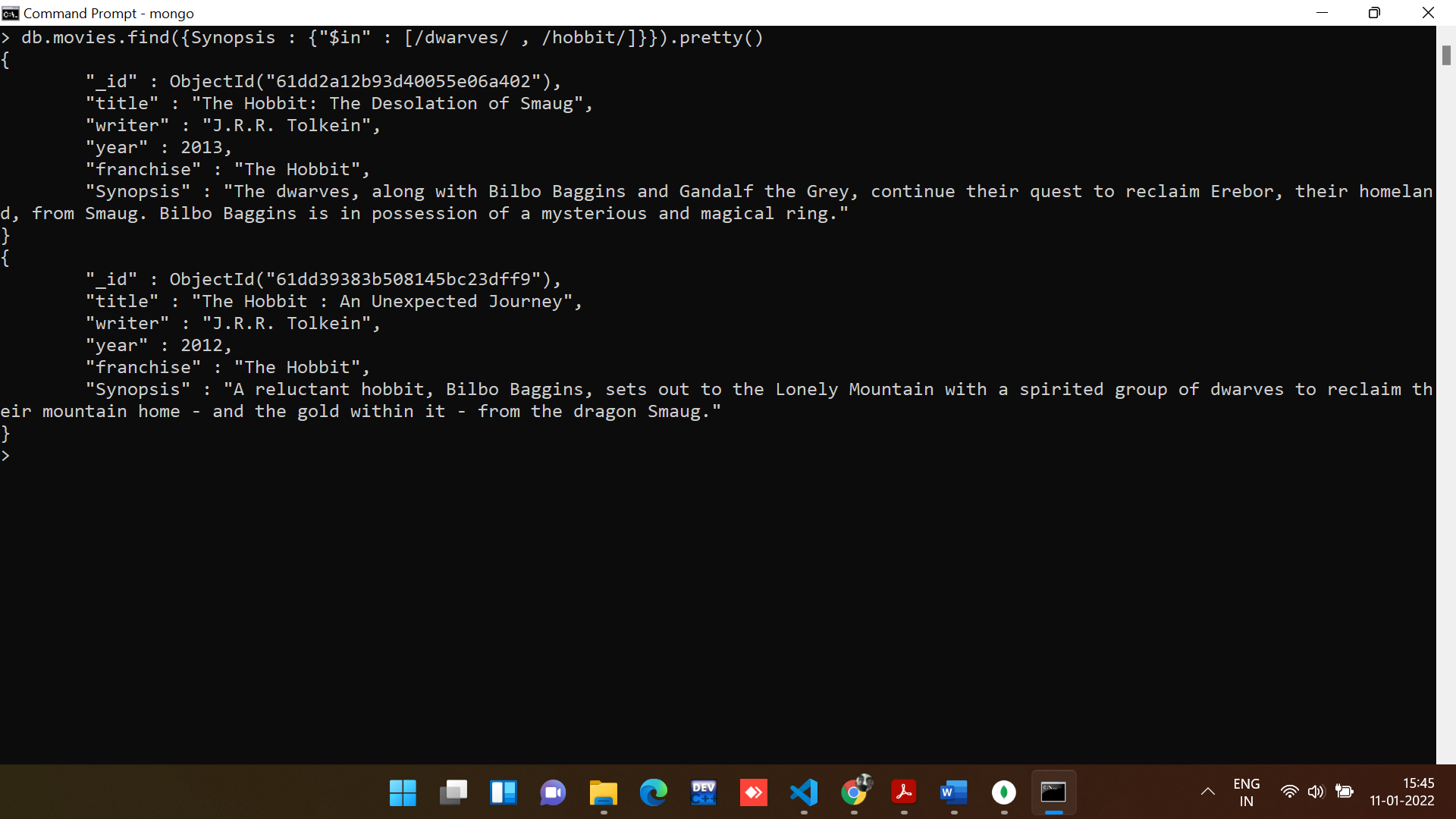
4. find all movies that have a synopsis that contains the word "dwarves" or "hobbit".

**Command:**

*// find all movies that have a synopsis that contains the word "dwarves" or "hobbit".*

db.movies.find({Synopsis : {"$in" : [/dwarves/ , /hobbit/]}})

**Output:**



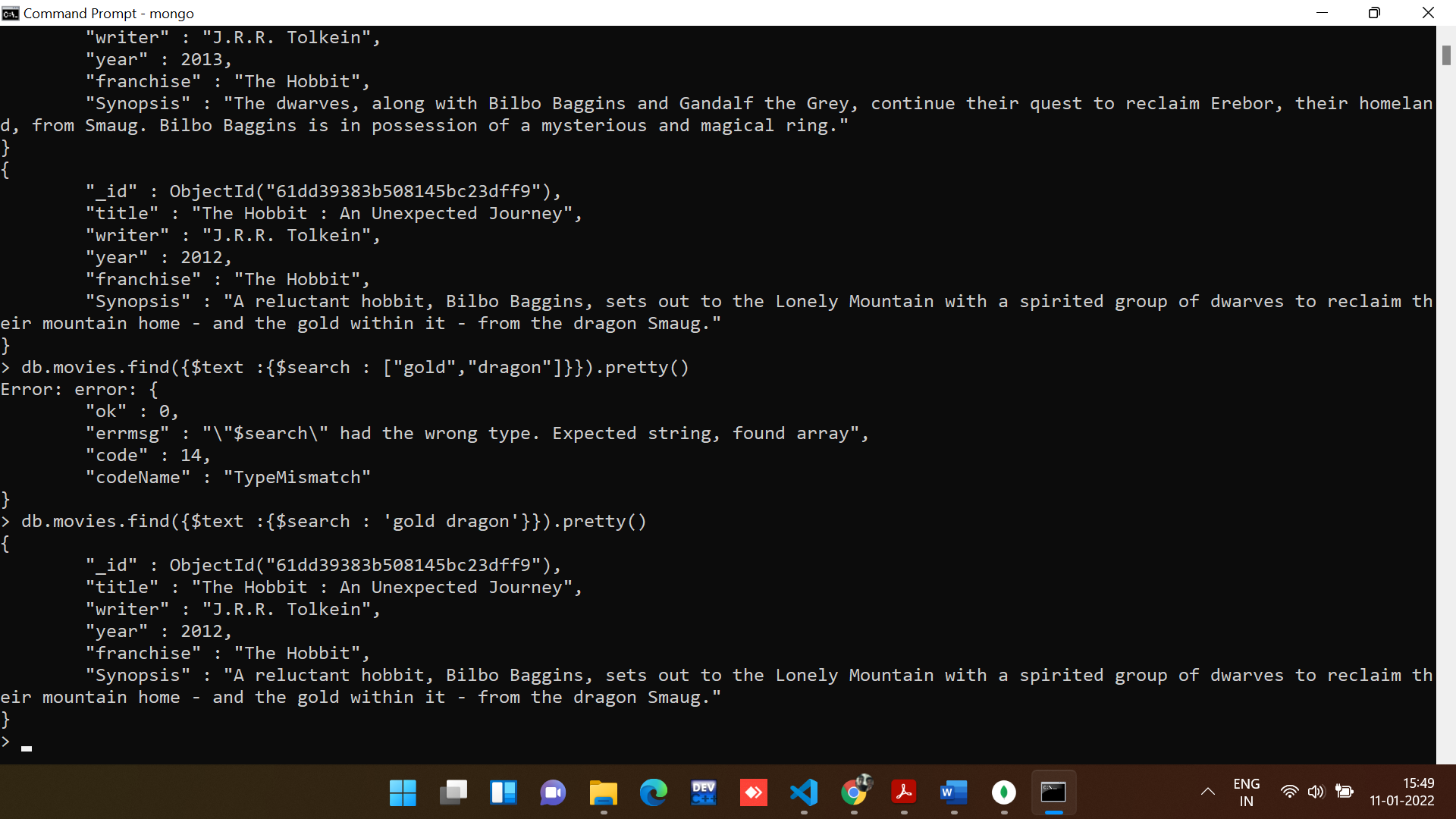
5. find all movies that have a synopsis that contains the word "gold" and "dragon".

**Command:**

*// find all movies that have a synopsis that contains the word "gold" and "dragon".*

db.movies.find({$text :{$search : 'gold dragon'}}).pretty()

**Output:**



**Section-5: - Delete Documents**

1. delete the movie "Pee Wee Herman's Big Adventure".

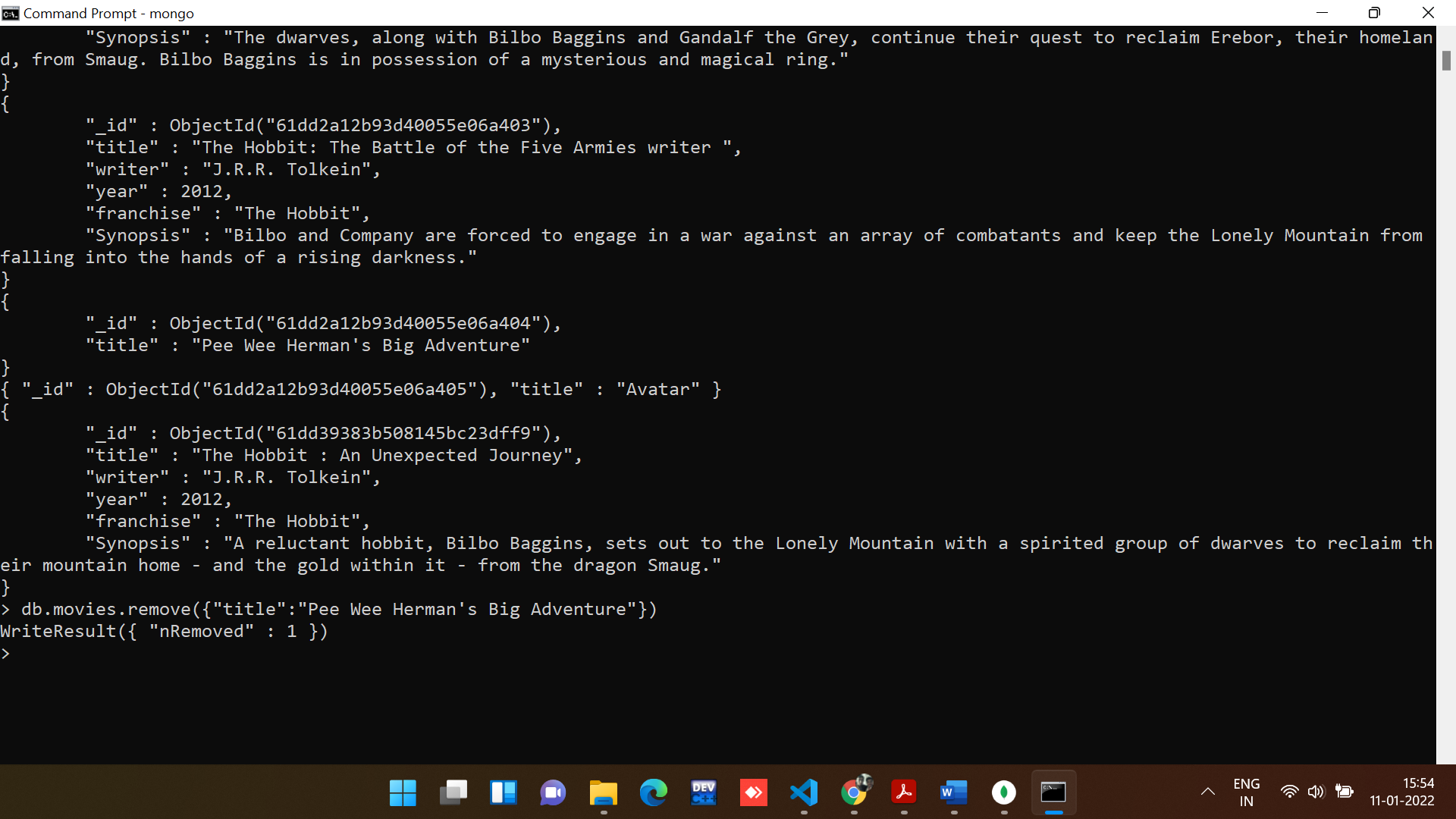
**Command:**

*//command to delete the movie name Pee Wee Herman's Big Adventure*

*// Pee Wee Herman's Big Adventure*

db.movies.remove({"title":"Pee Wee Herman's Big Adventure"})

**Output:**

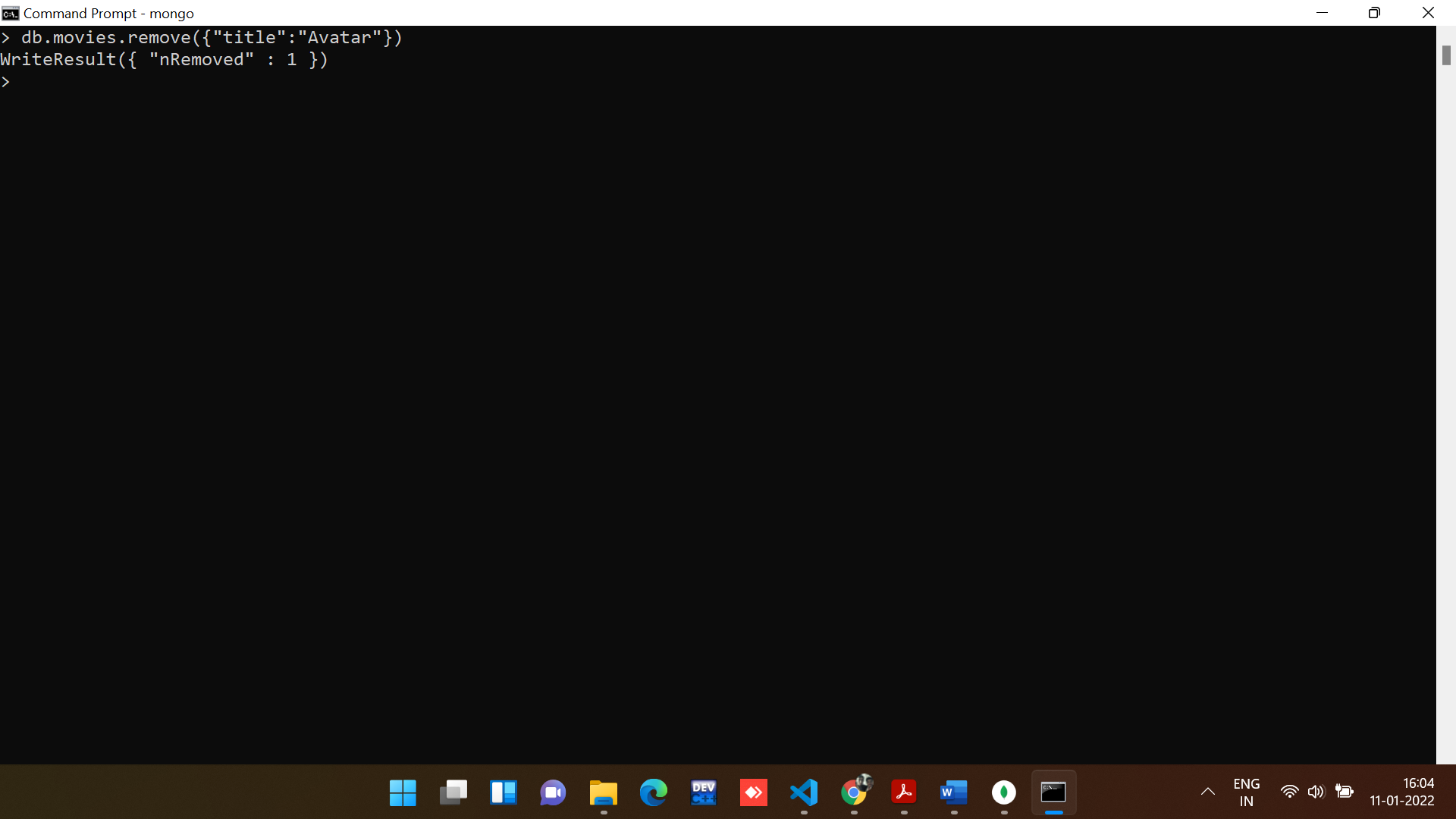


2. delete the movie "Avatar".

**Command:**

db.movies.remove({"title":"Avatar"})

**Output:**



**Section-6: - Relationships**

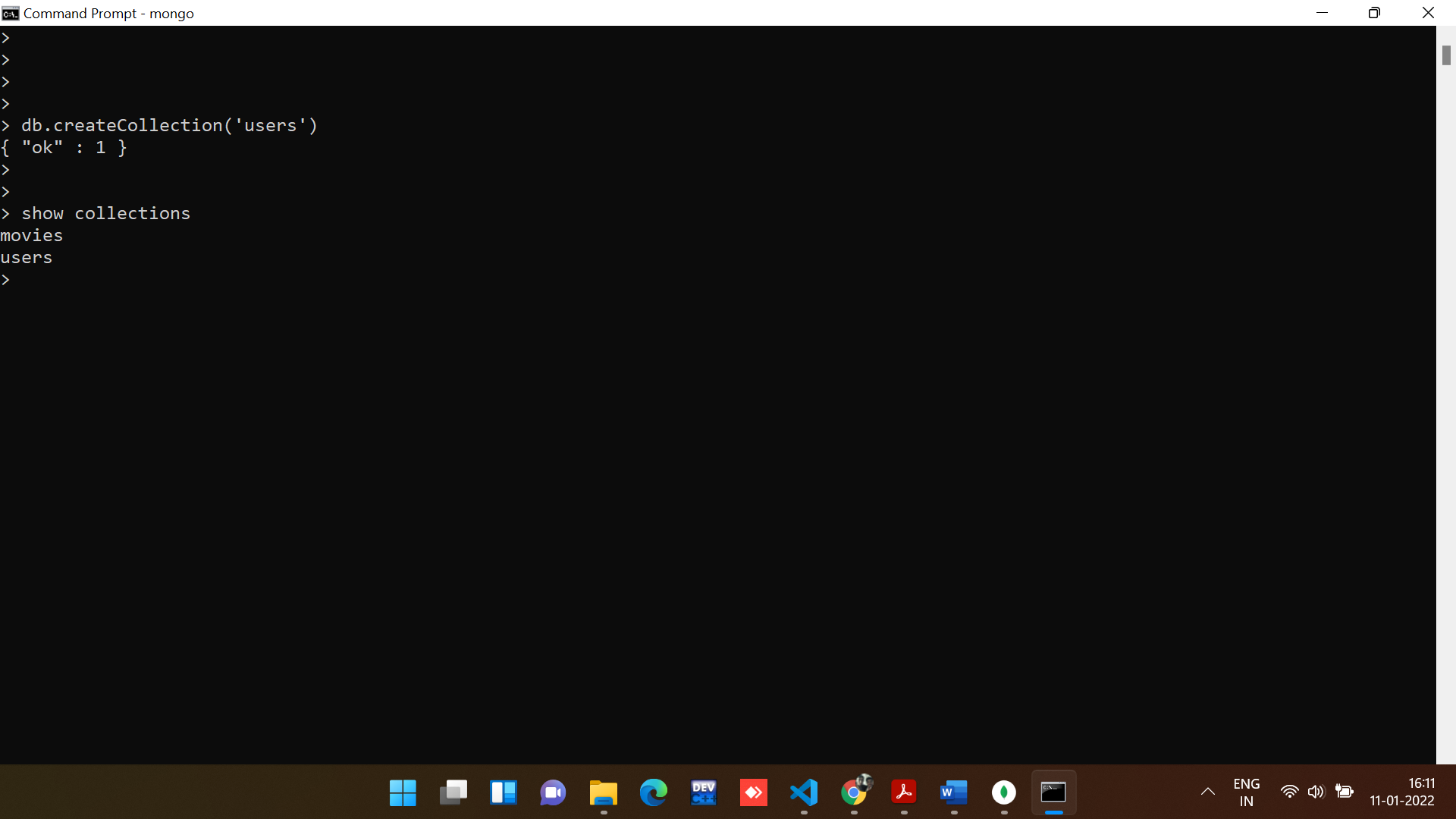
Insert the following documents into a users collection

Create a new collection named ‘users’ using the following command:

**Command:**

db.createCollection('users')

**Output:**



Insert rows inside the users collection:

**Command:**

db.users.insertMany([{

    "username" : "GoodGuyGreg",

    "first\_name" : "Good Guy",

    "last\_name" : "Greg"},

    {

        "username" : "ScumbagSteve",

        "full\_name" : {

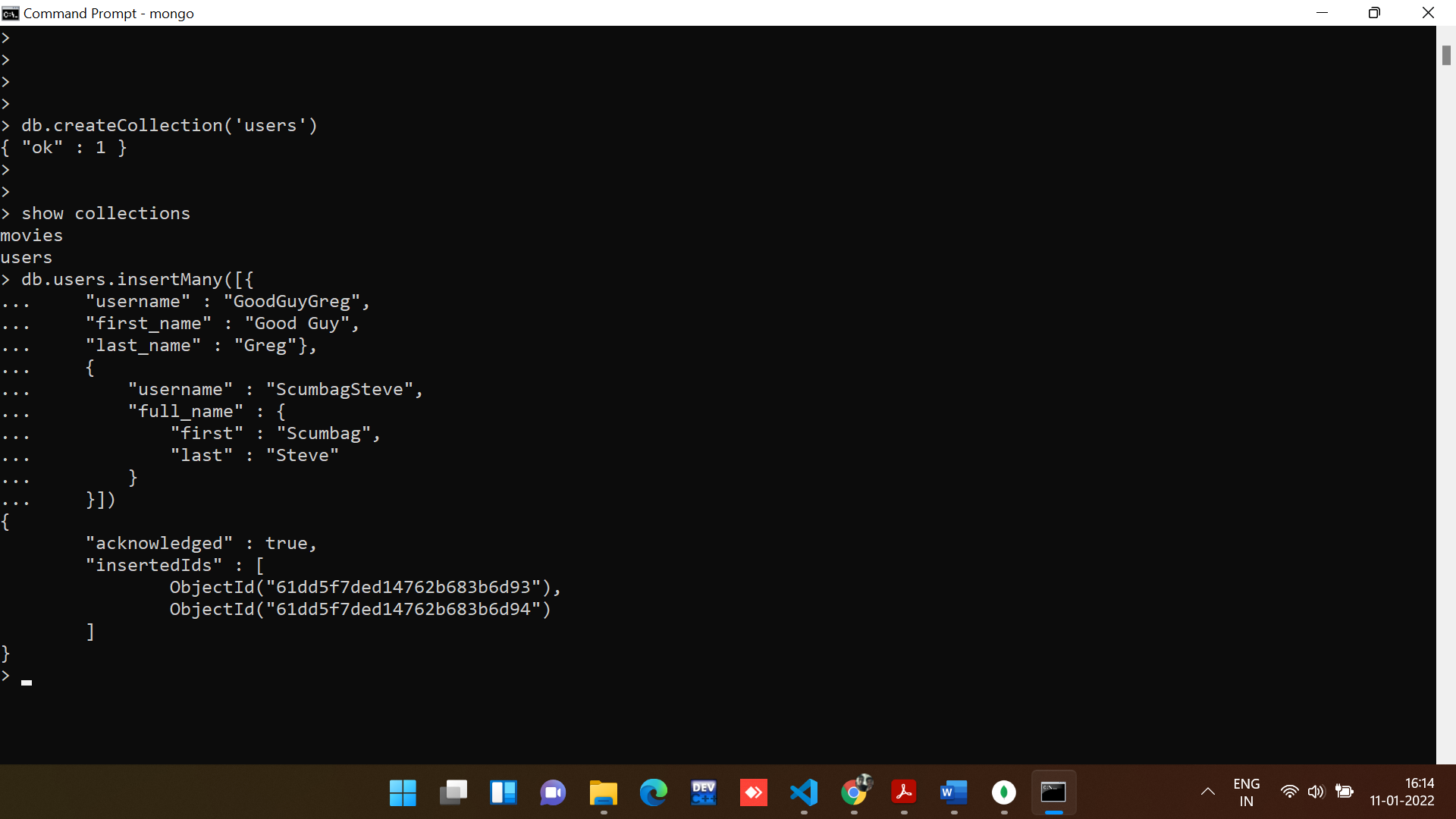
            "first" : "Scumbag",

            "last" : "Steve"

        }

    }])

**Output:**

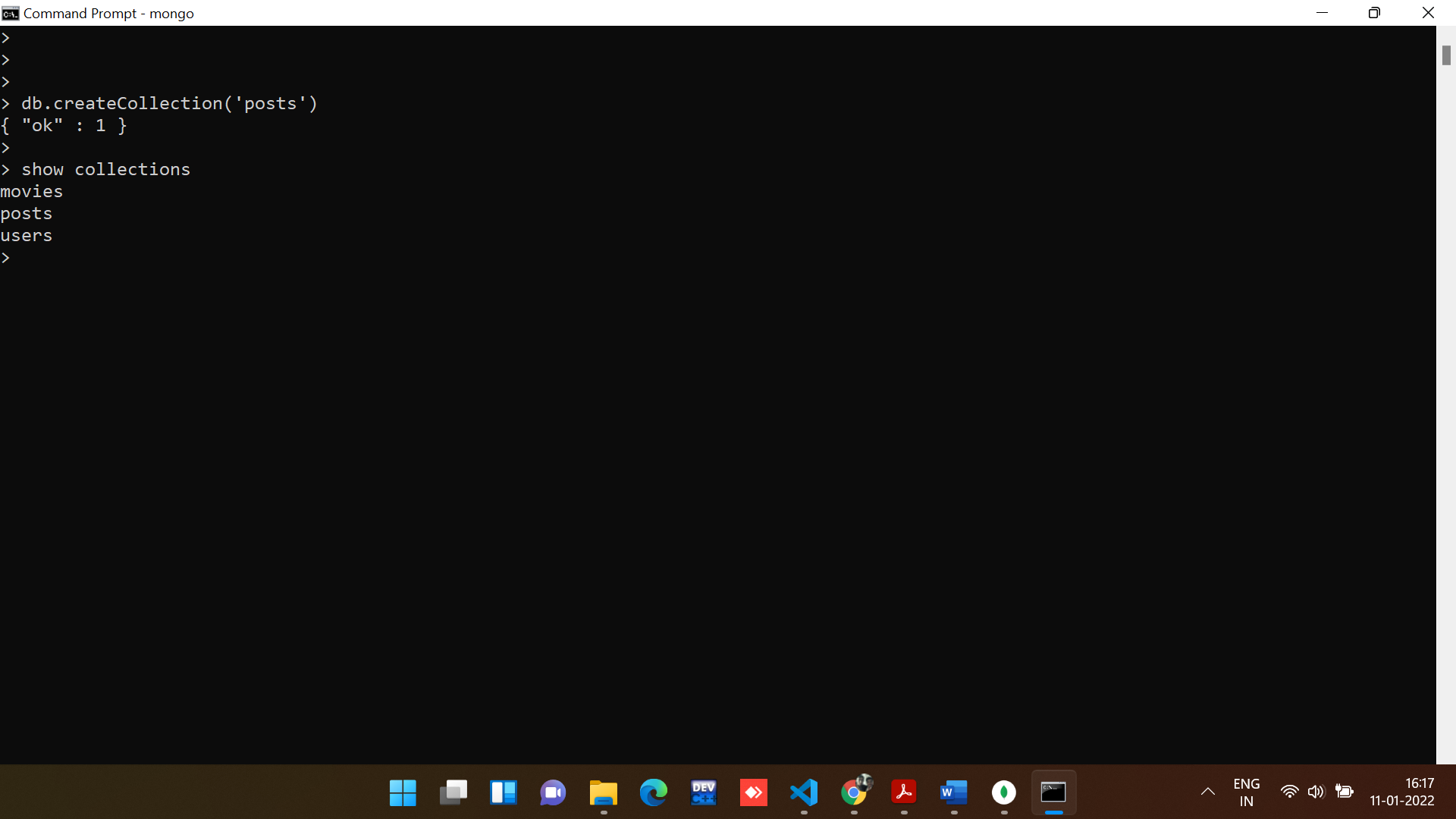


Insert the following documents into a posts collection.

**Command:**

db.createCollection('posts')

**Output:**



Insert rows inside the users collection:

**Command:**

db.posts.insertMany([

    {

    "username" : "GoodGuyGreg",

    "title" : "Passes out at party",

    "boby" : "Wakes up early and cleans house"

    },

    {

        "username" : "GoodGuyGreg",

        "title" : "Steals your identity",

        "boby" : "Raises your credit score"

    },

    {

        "username" : "GoodGuyGreg",

        "title" : "Reports a bug in your code",

        "boby" : "Sends you a Pull Request"

    },

    {

        "username" : "ScumbagSteve",

        "title" : "Borrows something",

        "boby" : "Sells it"

    },

    {

        "username" : "ScumbagSteve",

        "title" : "Borrows everything",

        "boby" : "The end"

    },

    {

        "username" : "ScumbagSteve",

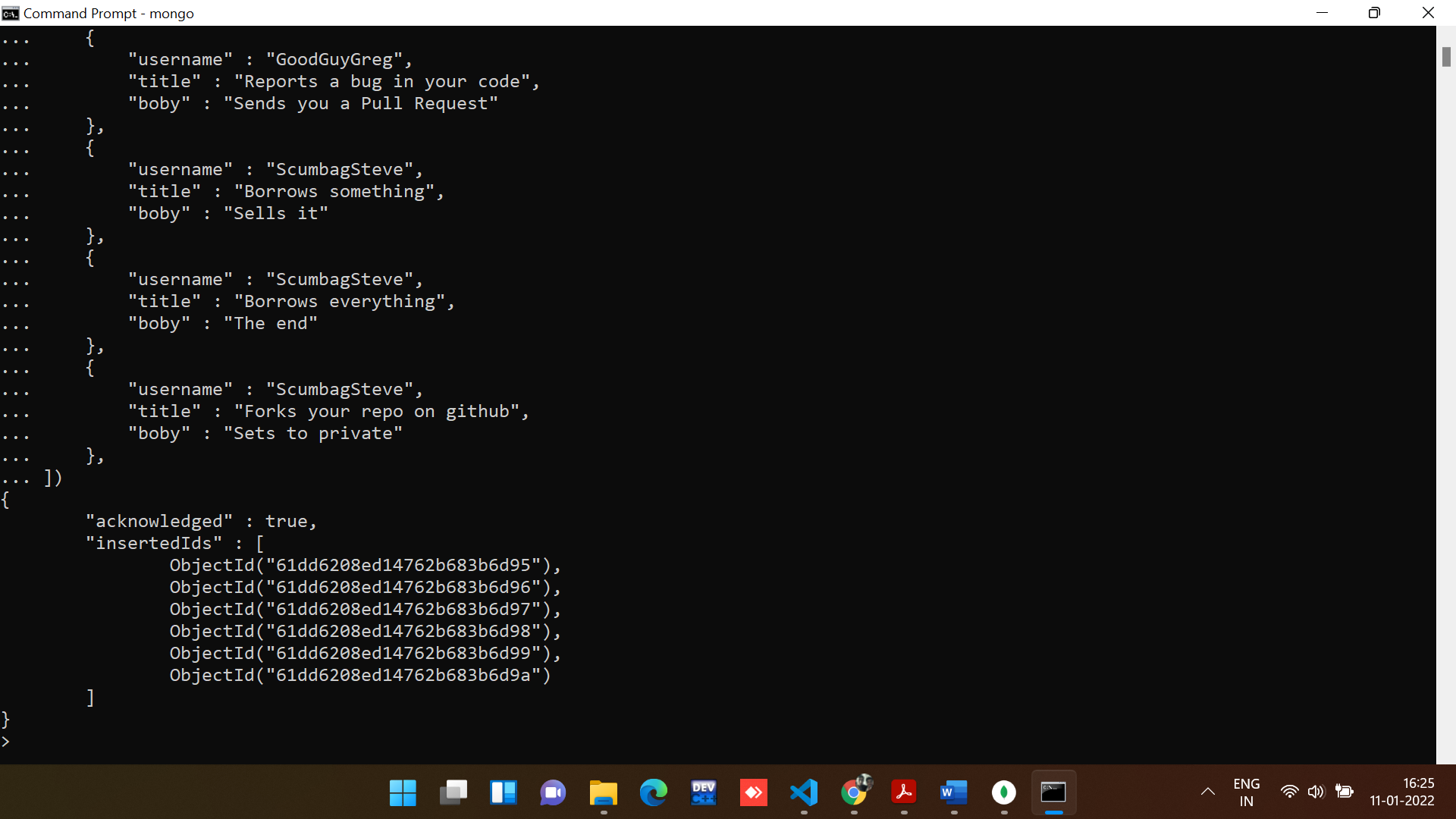
        "title" : "Forks your repo on github",

        "boby" : "Sets to private"

    },

])

**Output:**

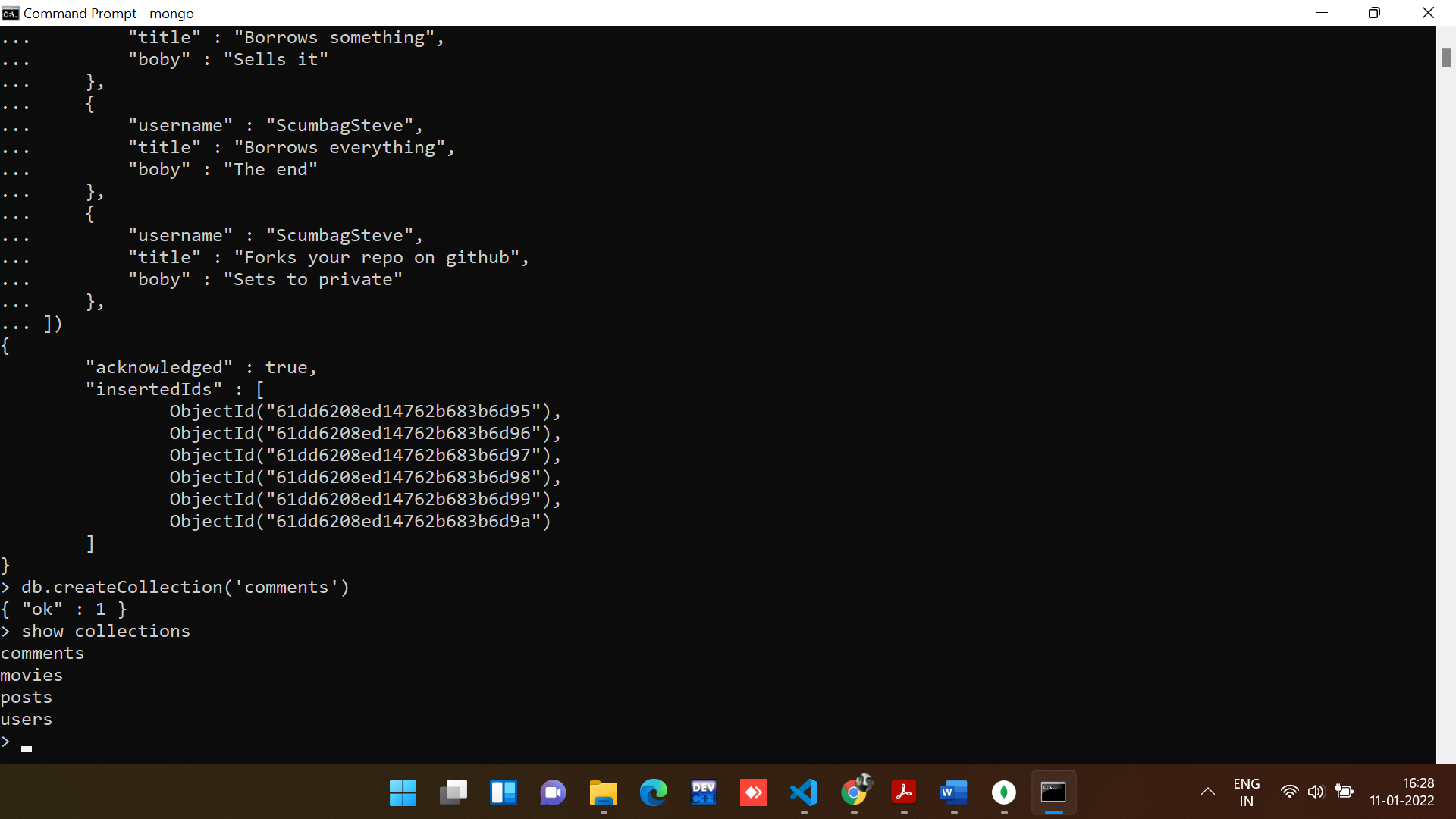


Insert the following documents into a comments collection.

**Command:**

db.createCollection('comments')

**Output:**



Insert rows inside the users collection:

**Command:**

*// Insert the following documents into a comments collection*

db.comments.insertMany([

  {

    username: "GoodGuyGreg",

    comment: "Hope you got a good deal!",

    post: [

      {

        $ref: "posts",

        $id: ObjectId("61dd6208ed14762b683b6d98"),

        $db: "mongo\_pratice",

      },

    ],

  },

  {

    username: "GoodGuyGreg",

    comment: "What's mine is yours!",

    post: [

      {

        $ref: "posts",

        $id: ObjectId("61dd6208ed14762b683b6d99"),

        $db: "mongo\_pratice",

      },

    ],

  },

  {

    username: "GoodGuyGreg",

    comment: "Don't violate the licensing agreement!",

    post: [

      {

        $ref: "posts",

        $id: ObjectId("61dd6208ed14762b683b6d9a"),

        $db: "mongo\_pratice",

      },

    ],

  },

  {

    username: "ScumbagSteve",

    comment: "It still isn't clean",

    post: [

      {

        $ref: "posts",

        $id: ObjectId("61dd6208ed14762b683b6d95"),

        $db: "mongo\_pratice",

      },

    ],

  },

  {

    username: "ScumbagSteve",

    comment: "Denied your PR cause I found a hack",

    post: [

      {

        $ref: "posts",

        $id: ObjectId("61dd6208ed14762b683b6d97"),

        $db: "mongo\_pratice",

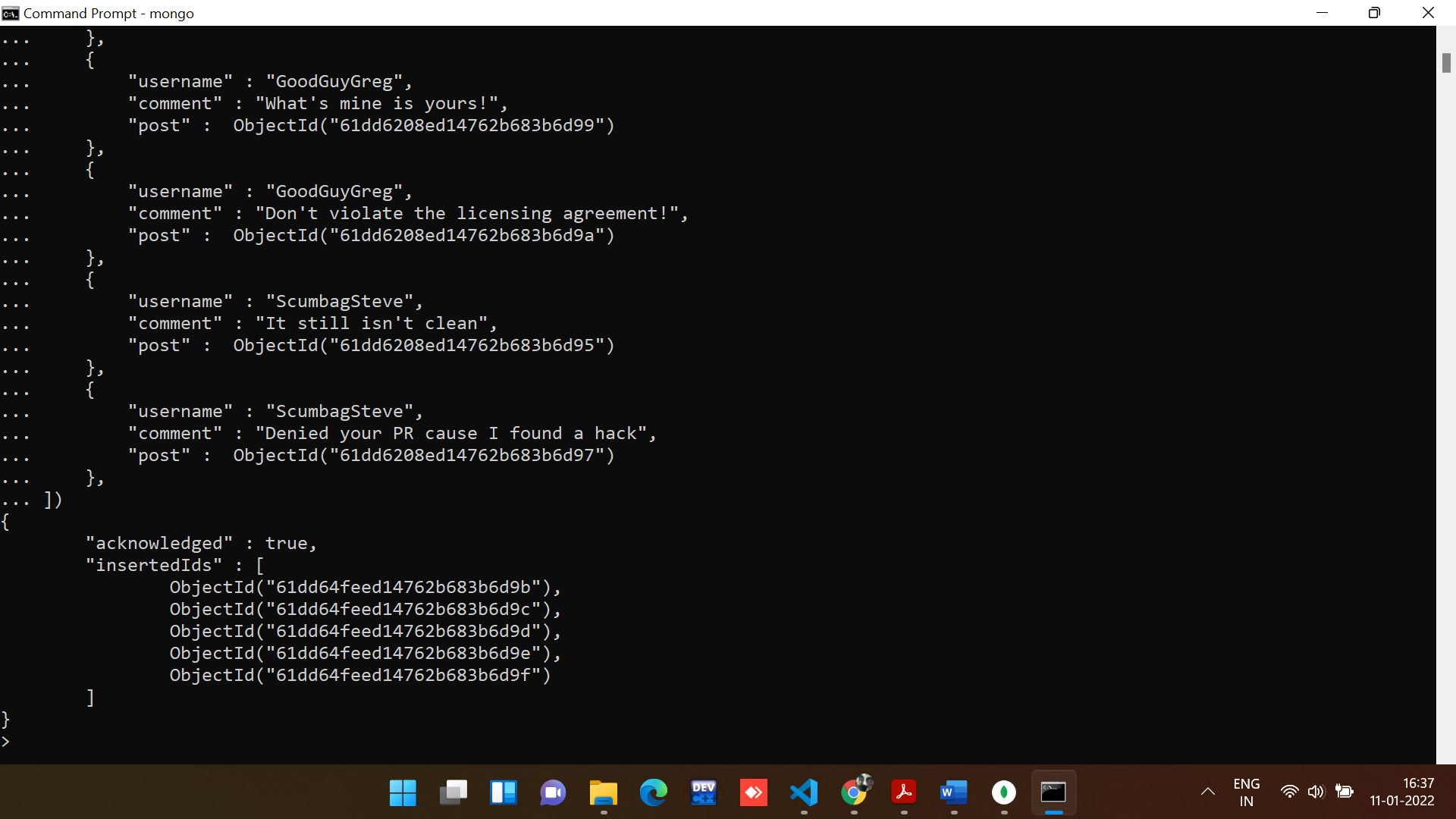
      },

    ],

  },

]);

**Output:**



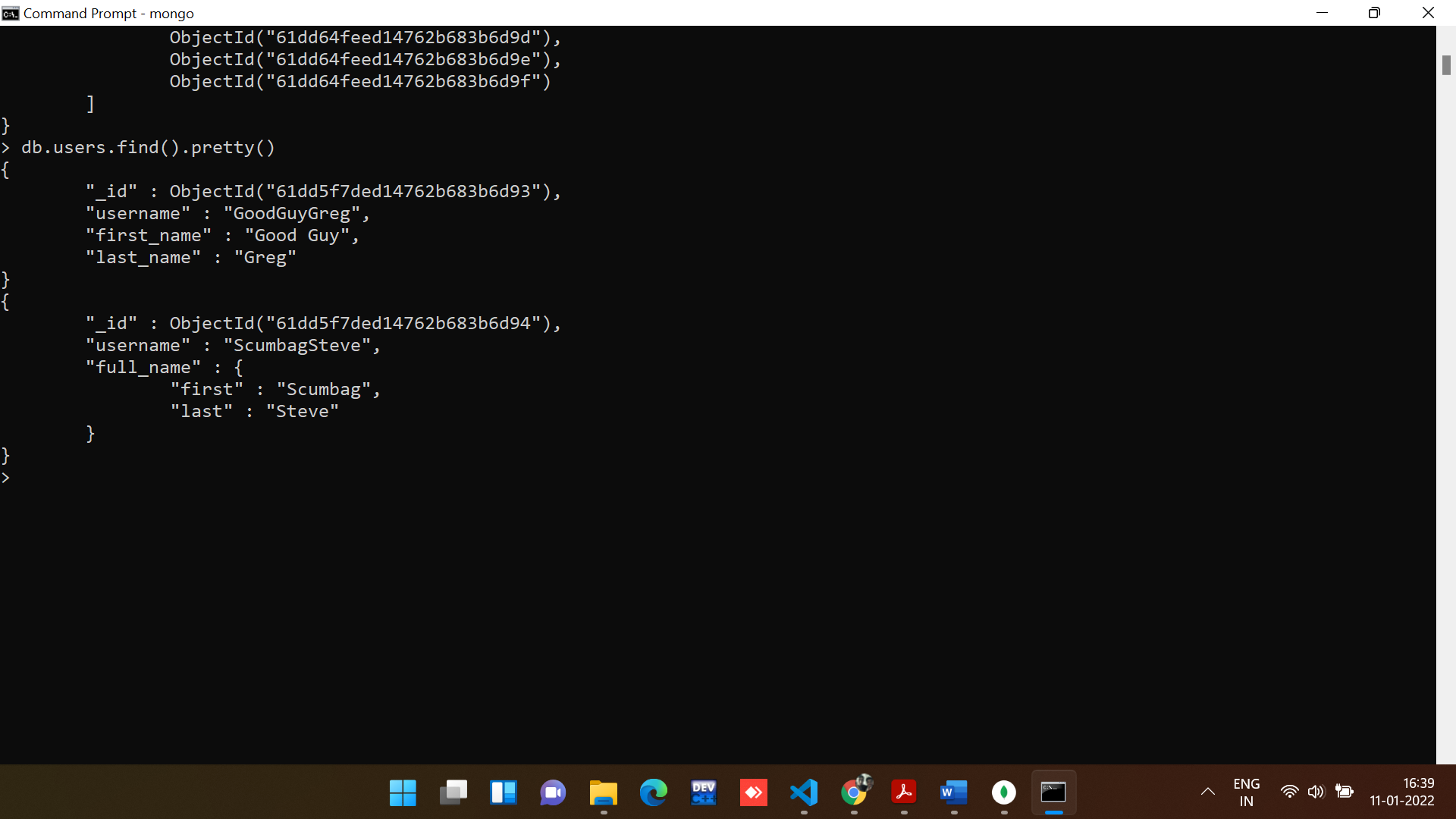
1. find all users

**Command:**

*//Command to find all the users*

db.users.find().pretty()

**Output:**



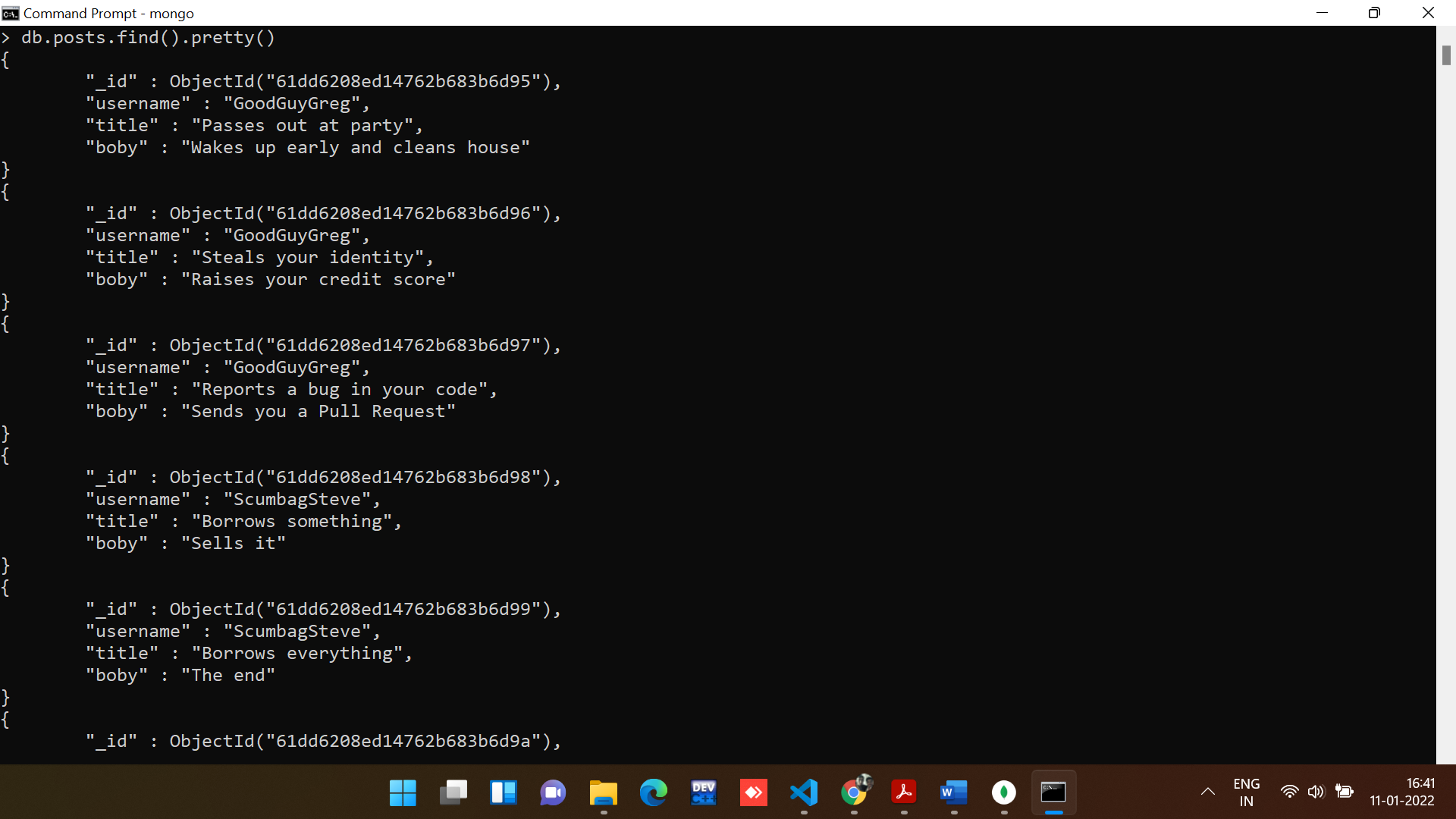
2. find all posts

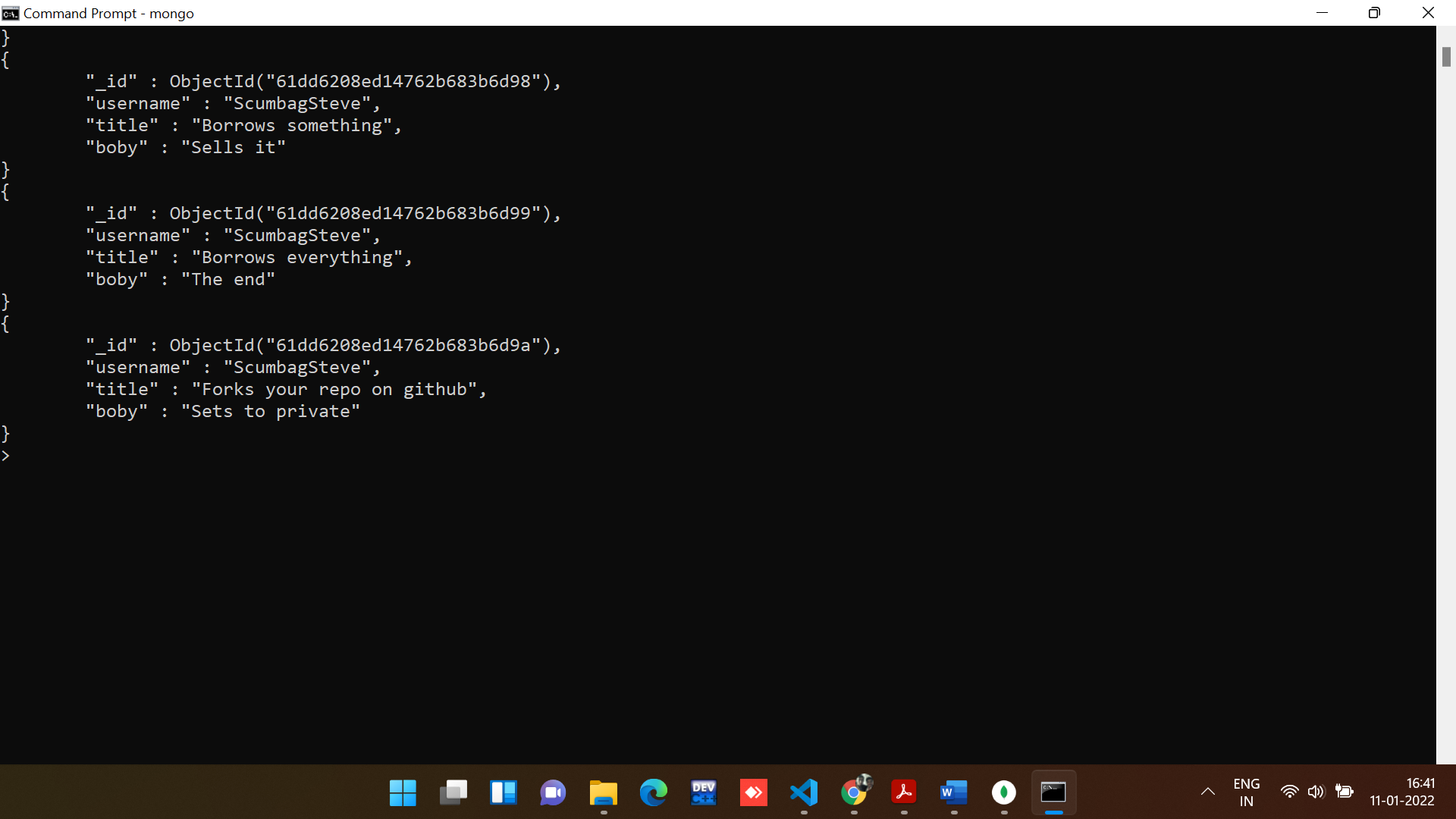
**Command:**

*//Command to find all the users*

db.posts.find().pretty()

**Output:**





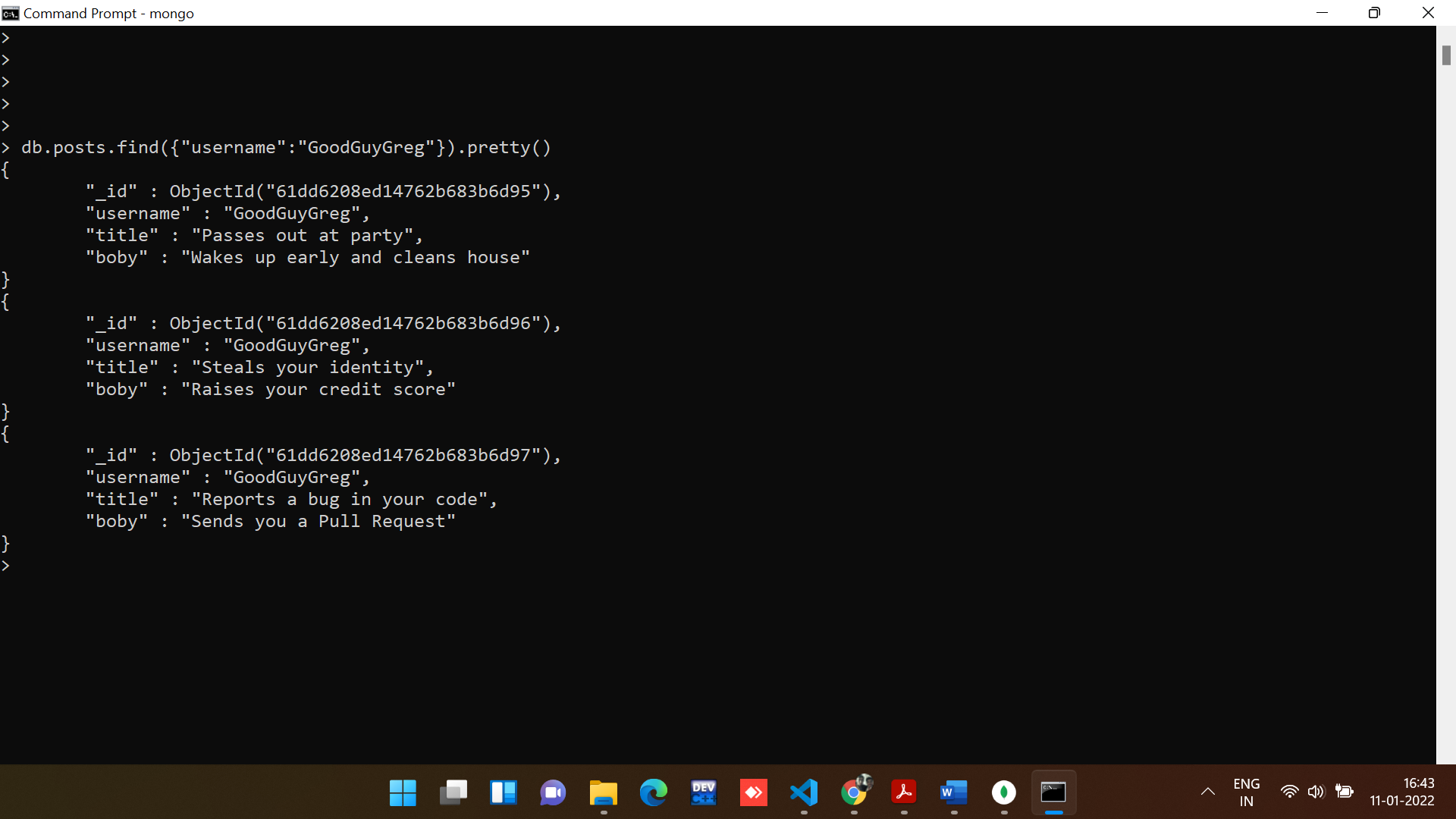
3. find all posts that was authored by "GoodGuyGreg"

**Command:**

*//Command to find all posts that was authored by "GoodGuyGreg"*

db.posts.find({"username":"GoodGuyGreg"})

**Output:**



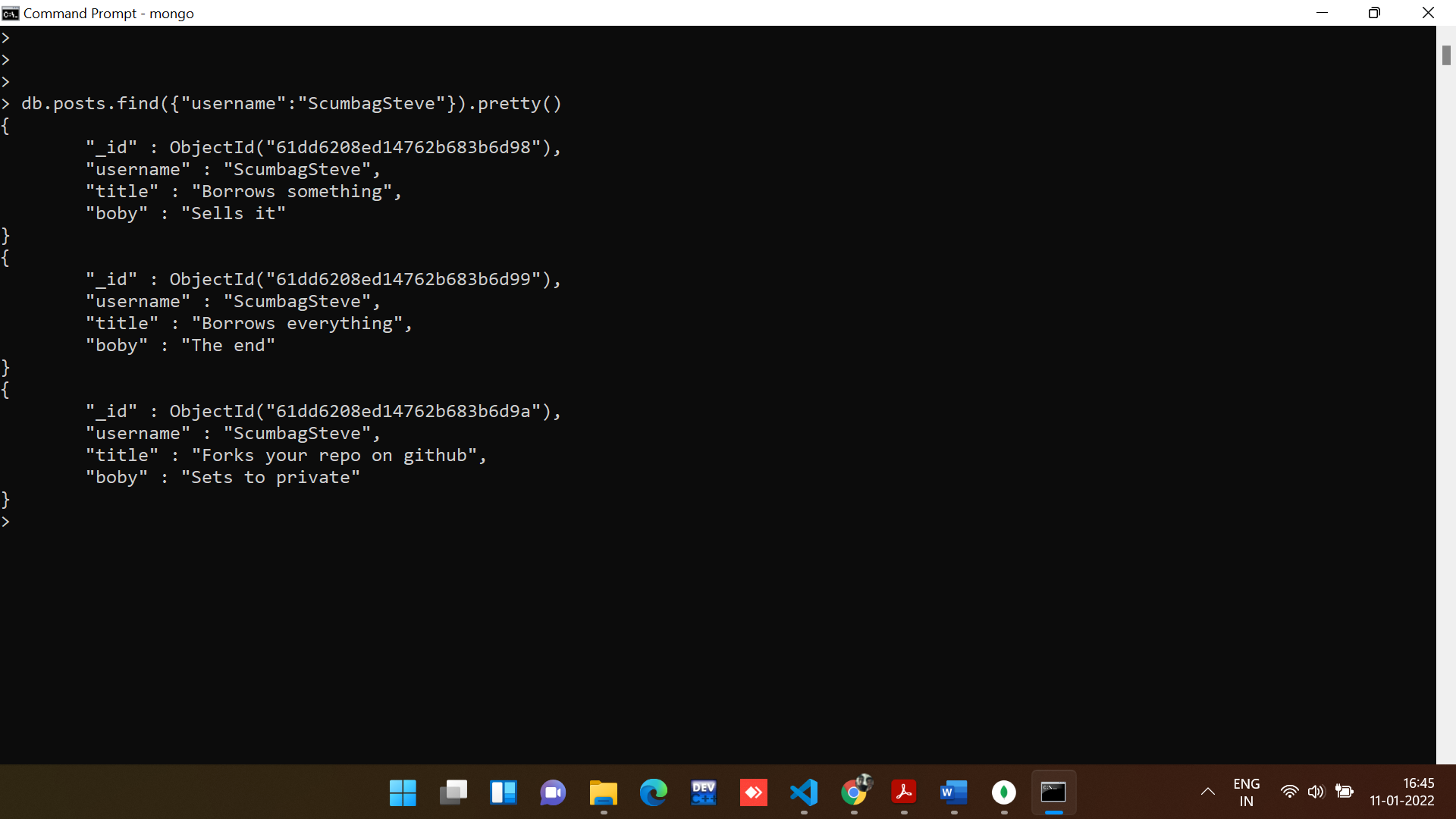
4. find all posts that was authored by "ScumbagSteve"

**Command:**

*//Command to find all posts that was authored by "ScumbagSteve"*

db.posts.find({"username":"ScumbagSteve"})

**Output:**



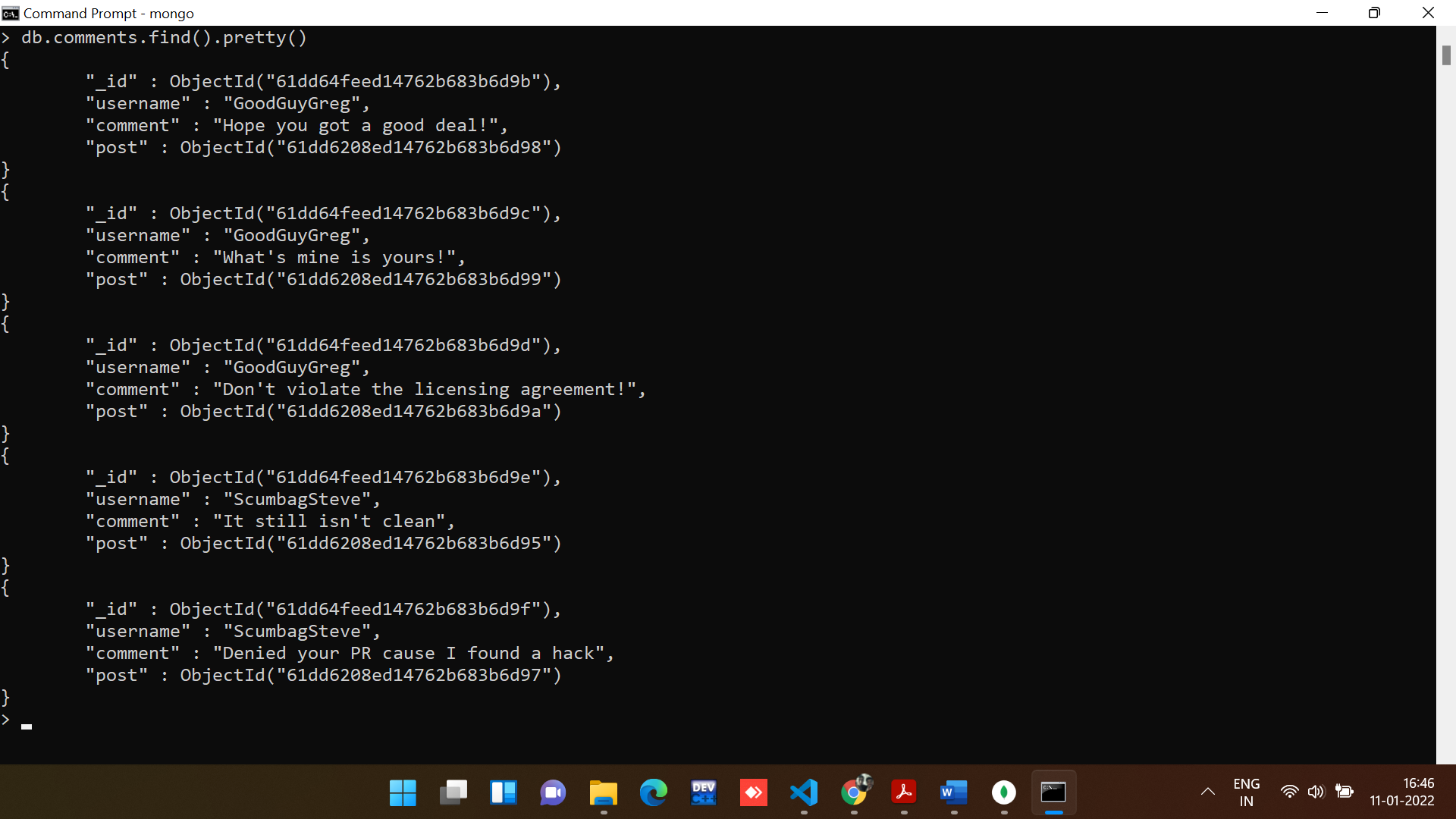
5. find all comments

**Command:**

*//Command to find all the users*

db.comments.find().pretty()

**Output:**



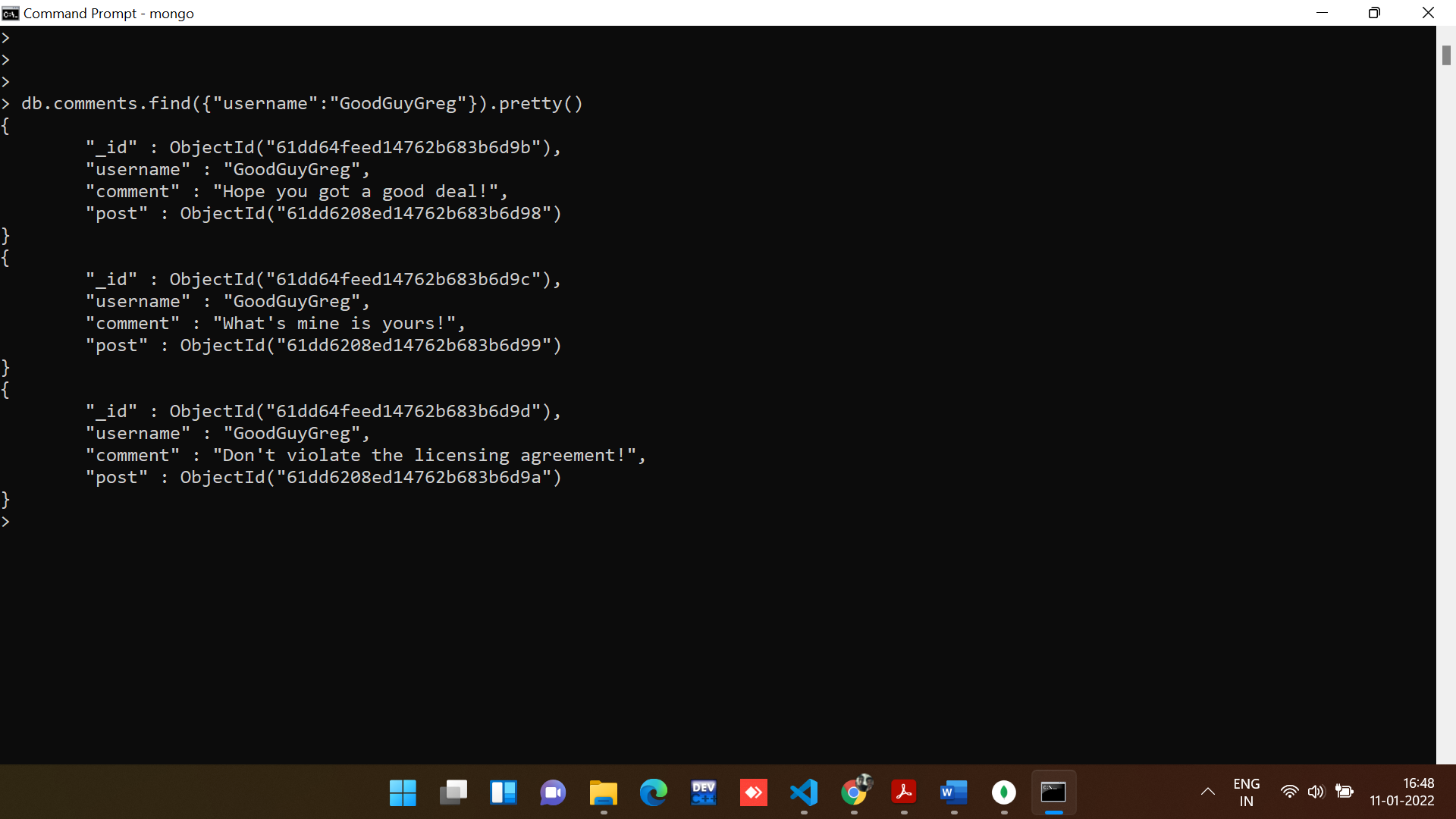
6. find all comments that was authored by "GoodGuyGreg"

**Command:**

*//Command to find all comments that was authored by "GoodGuyGreg"*

db.comments.find({"username":"GoodGuyGreg"})

**Output:**



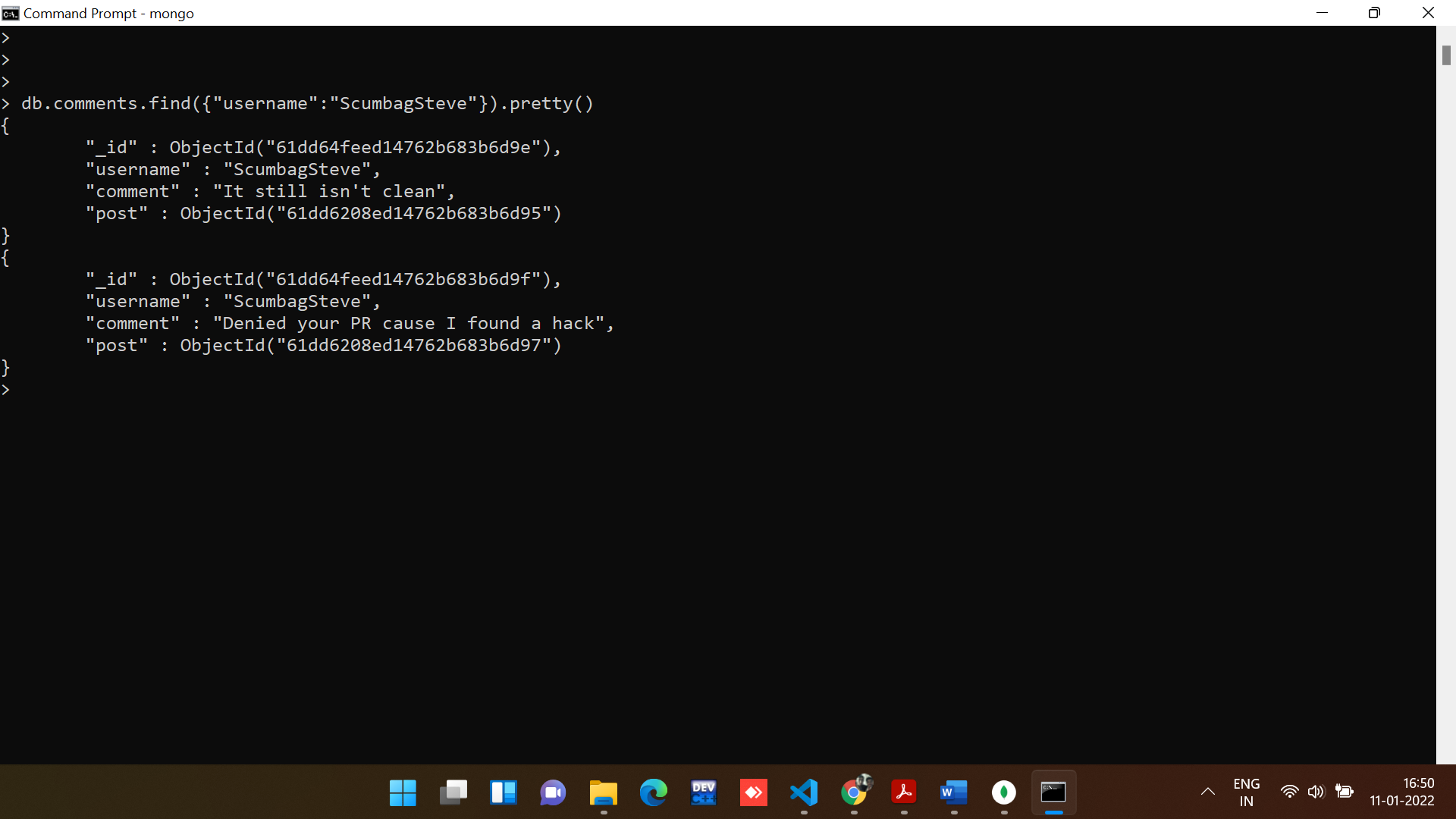
7. find all comments that was authored by "ScumbagSteve"

**Command:**

*//Command to find all comments that was authored by "ScumbagSteve"*

db.comments.find({"username":"ScumbagSteve"})

**Output:**



8. find all comments belonging to the post "Reports a bug in your code".

**Command:**

db.posts.aggregate([

{

$match: { title: 'Reports a bug in your code' }

},

{

$lookup: {

from: 'comments',

localField: '\_id',

foreignField: 'post',

as: 'comments'

}

}

]).pretty();

**Output:**

